Table des matières

BUREAUTIQUE

Excel

- Excel: Introduction
- Sélectionner des cellules dans Excel
- Sélectionner des cellules dans Excel - Pratique 💪
- Excel: créer des séries
- Modifier la taille des lignes et des colonnes dans Excel
- Sélection, hauteur et largeur: exercices 💪
- Excel: Les formules
- Excel: opérateurs et fonctions
- Excel - Les références absolues
- Excel - Les références absolues - Exercices supplémentaires
- Excel: Les principaux types de graphiques
- Naviguer dans un document
- Le presse-papier Windows

GESTION DES DONNÉES

- Les bases de données
- La chaos de données peu structurées
- Identifier et structurer les données
- Le chaos des données - Exercices 💪
- Identifier ses données
- Les relations
- Un SGBDR, c'est quoi ?
- Du MCD au modèle relationnel
- Créer sa base et ses tables avec HeidiSQL
- Les clés étrangères en pratique
- phpMyAdmin : le même travail côté web
- SQL — Premières requêtes (SELECT, FROM, WHERE, ORDER BY)
- Schéma Entités-Associations
- Schéma Entité-Association - Exercices de modélisation
- Identifier les entités
- Système de Gestion de Base de Données (SGBD)
- SQL
- Les Requêtes SQL de Type SELECT
- Types de données supportés par MySQL

PROGRAMMATION EN PYTHON

- Python
- Améliorez Vos Affichages avec le module Rich
- Python: Les outils pour développer
- Thonny Python
- Thonny Python - Déboguer son code
- Thonny Python - Installer des modules
- Python - Opérations mathématiques de base
- Python: Les chaines de caractères

Python: Opérations sur les strings
Chaines de caractères: exercices
Python: Découper les chaînes de caractères
Slicing - Exercices
Python: Rechercher dans une chaîne de caractères
Python: Parcourir une string avec `for`
Python: Parcourir une string: Exercices
🧩 Défis Python – Chaînes de caractères
🧩 Défis Python – Mot de passe

ELECTRONIQUE

Les actionneurs
Feux de circulation
Feux de circulation - Simulation
Introduction aux buzzers
Utiliser un buzzer actif (KY-012)
Utiliser un buzzer passif (KY-006)
Les capteurs
DHT11: Simulation du Capteur
DHT11: Capteur de température et d'humidité
HC-SR04: Capteur de distance à ultra-sons
Les micro-contrôleurs
Le Raspberry Pi Pico
Les Broches du Raspberry Pi Pico en MicroPython
MicroPython sur RPi Pico avec Thonny Python
Raspberry Pi Pico et les Ports GPIO
Utilisation du Raspberry Pi Pico avec MicroPython
Les différents types de signaux

INTERNET

Création de pages Web

Bootstrap

Les breakpoints
Les containers
Grille responsive
Exercice - Grille - 02

CSS

CSS: en haut, en bas, à gauche, à droite
Les sélecteurs CSS
Les principaux sélecteurs de base
Les principaux sélecteurs de base - Exemples
Les sélecteurs - Exercices 01
Les sélecteurs - Exercices 03
Les sélecteurs - Exercices 02
Aide mémoire CSS
Les outils pour créer du HTML
La console développeur des navigateurs

HTML

Cheat Sheet / Aide mémoire HTML
HTML - La balise DIV
HTML - Le Responsive Web Design
Créer un site complet - Composants
Site complet: Processeur
Site complet: Processeur (CPU)
Site complet: Mémoire vive (RAM)
Site complet: Stockage – SSD / NVMe
Site complet: Alimentation (PSU – Power Supply Unit)
Site complet: Carte graphique – GPU / IGP
Site complet: Carte mère
Site complet: Refroidissement (CPU / boîtier)
Site complet: Boîtier (case)
Site complet: Carte réseau & interfaces (RJ45 / Wi-Fi)
Site complet: Réseau local (LAN)
Site complet: Modem, routeur et Internet
Site complet: Système d'exploitation
Site complet: Carte son
Site complet: Imprimantes
Site complet: Claviers
Site complet: Souris
Site complet: Tablettes graphiques
Site complet: Périphériques MIDI
Site complet: Streaming Vidéo
HTML - L'approche Mobile First
Le Box Model
Explorer le Box Model
Exercice Bob l'éponge
Exercice - Les Briques
Exercice - Tables
Les formulaires HTML
Les formulaires - Partie 2
Les formulaires - envoyer ses données
Les formulaires HTML (exercice complet) 🧢
Les formulaires HTML (exercices) 🧢

Javascript

💪🏻 JavaScript - Boucles for - Exercices
🎯 Bases de Javascript: Carnet d'objectifs
JavaScript - La console développeur
JavaScript - Syntaxe et variables
JavaScript - Exercices de base
JavaScript - La saisie utilisateur
JavaScript - Les conditions
JavaScript - Combiner plusieurs conditions
JavaScript - Manipuler le DOM

SYSTÈMES D'EXPLOITATION

Fichiers et dossiers
Organisez votre vidéothèque (2025)

Les outils Windows
Total Commander
TC - Multi-Rename Tool
TC - Synchronize Dirs
TC - 25 raccourcis clavier essentiels

TECHNOLOGIES

3D et impression
TinkerCad — modéliser ton premier objet
L'impression 3D — comment ça marche
Projet : ton premier objet imprimé

GESTION DE PROJETS

Le carré des contraintes



BUREAUTIQUE

Excel

> *Dans le monde d'aujourd'hui, maîtriser des outils comme Microsoft Excel est essentiel. Que tu sois un futur ingénieur, gestionnaire, ou même créatif, Excel va devenir ton allié pour organiser et analyser des données. Tu en entends peut-être parler en cours ou chez toi, mais à quoi ça sert vraiment ? C'est ce que nous allons découvrir ensemble.*

Qu’est-ce qu’Excel ?

Excel est un logiciel de **feuilles de calcul** (= **tableur**) qui fait partie de la suite Microsoft Office. Il te permet de :

- **Organiser des données** : Par exemple, tu peux classer des informations comme les notes de tes cours ou tes dépenses hebdomadaires.
- **Faire des calculs automatiques** : Tu peux ajouter, soustraire, diviser, multiplier, etc., en quelques clics grâce aux formules.
- **Créer des graphiques** : Pour représenter visuellement tes données.
- **Analyser des informations** : Excel peut t'aider à repérer des tendances, des moyennes, des minimums, maximums, etc.

Excel: Introduction

> *Dans le monde d'aujourd'hui, maîtriser des outils comme Microsoft Excel est essentiel. Que tu sois un futur ingénieur, gestionnaire, ou même créatif, Excel va devenir ton allié pour organiser et analyser des données. Tu en entends peut-être parler en cours ou chez toi, mais à quoi ça sert vraiment ? C'est ce que nous allons découvrir ensemble.*

Qu'est-ce qu'Excel ?

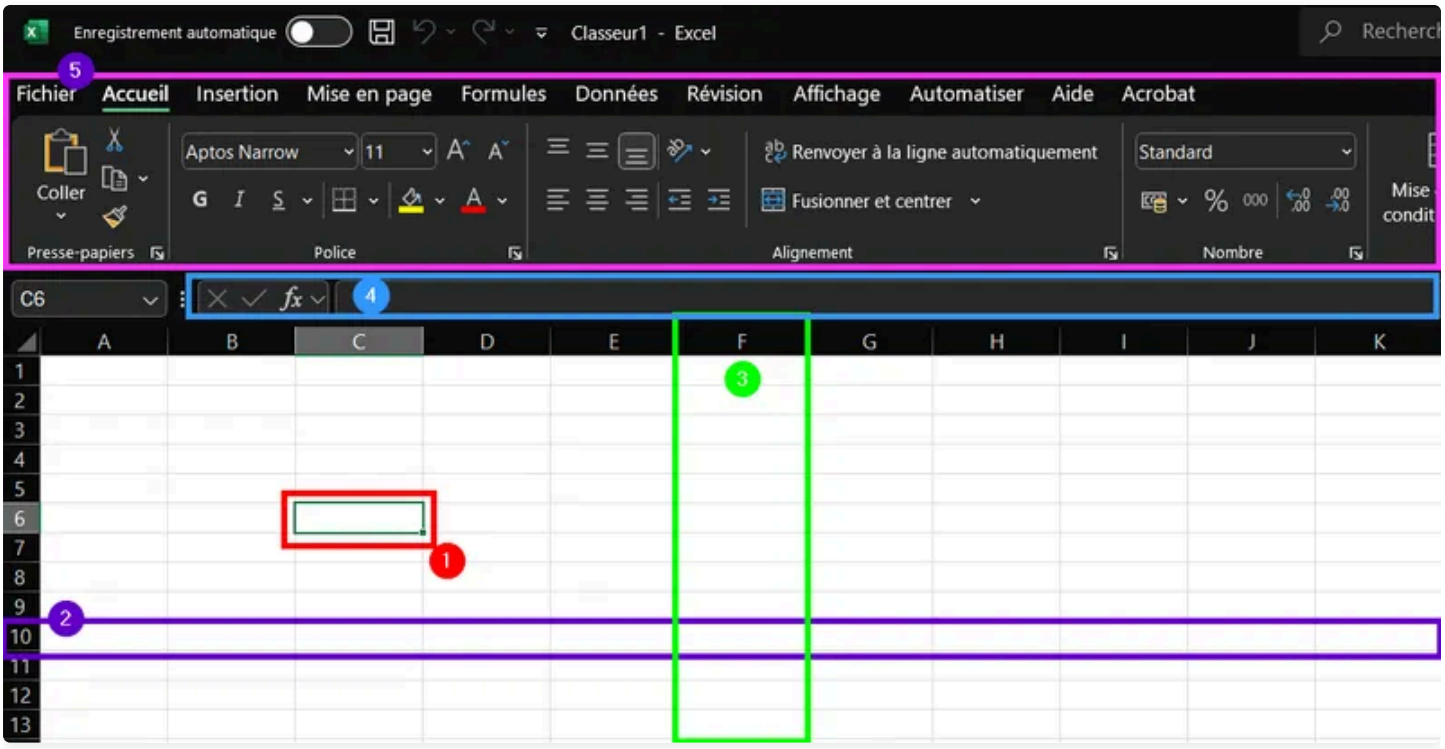
Excel est un logiciel de **feuilles de calcul** (= **tableur**) qui fait partie de la suite Microsoft Office. Il te permet de :

- **Organiser des données** : Par exemple, tu peux classer des informations comme les notes de tes cours ou tes dépenses hebdomadaires.
- **Faire des calculs automatiques** : Tu peux ajouter, soustraire, diviser, multiplier, etc., en quelques clics grâce aux formules.
- **Créer des graphiques** : Pour représenter visuellement tes données.
- **Analyser des informations** : Excel peut t'aider à repérer des tendances, des moyennes, des minimums, maximums, etc.

Les bases d'Excel

L'interface d'Excel

Quand tu ouvres Excel, voici ce que tu vois :



- **Les lignes** [2] **et colonnes** [3] : Les colonnes sont marquées par des lettres et les lignes par des numéros. Ensemble, elles forment une grande grille où tu peux saisir des informations.
- **Les cellules** [1] : Ce sont des petits rectangles organisés en lignes (A, B, C…) et en colonnes (1, 2, 3…). Chaque cellule a une "adresse", par exemple A1 ou B3.
- **La barre de formules** [4]: Située en haut, elle affiche le contenu de la cellule sélectionnée. C'est aussi ici que tu peux écrire des formules pour faire des calculs.
- **Les onglets et rubans** [5]: Ils se trouvent en haut de la fenêtre. Les onglets (comme Accueil, Insertion, Formules) permettent d'accéder à différents outils et options.

Insérer des données

C'est très simple ! Clique sur une cellule et tape quelque chose (un nombre, du texte…). Appuie sur "Entrée", et voilà ! Tes données sont prises en compte.

Faire des calculs simples

Pour taper une formule dans Excel, il faut impérativement commencer par le signe `=` (égal)

Voici comment utiliser Excel pour des calculs basiques :

- Additionner : Tape `=A1 + A2` dans une cellule et appuie sur "Entrée". Cela va additionner le contenu des cellules A1 et A2.
- Soustraire : Utilise le signe moins (`-`), comme dans `=A1 - A2` .
- Multiplier : Utilise l'astérisque (`*`), par exemple `=A1 * A2` .

- Diviser : Utilise le slash (`/`), comme dans `=A1 / A2` .

Ces calculs sont appelés **formules**. N'oublie pas que chaque formule commence par un signe égal (`=`).

Utiliser des fonctions

Les **fonctions** sont des formules pré-programmées dans Excel. Par exemple, si tu veux faire la somme de plusieurs cellules, tu peux utiliser la fonction **SOMME**.

- Exemple : Si tu veux additionner toutes les valeurs entre A1 et A5, tape `=SOMME(A1:A5)` .

Voici quelques fonctions pratiques :

- **MOYENNE** : `=MOYENNE(A1:A5)` calcule la moyenne des nombres dans les cellules A1 à A5.
- **MIN** et **MAX** : `=MIN(A1:A5)` trouve la plus petite valeur, et `=MAX(A1:A5)` trouve la plus grande.

Trucs et astuces

Remplissage automatique

Imagine que tu veux inscrire les jours de la semaine dans ton tableau. Pas besoin de les taper un par un ! Écris "Lundi" dans une cellule, puis fais glisser la petite poignée (coin inférieur droit de la cellule) vers le bas : Excel complète automatiquement avec "Mardi", "Mercredi", etc.



Mise en forme

Pour rendre ton tableau plus lisible, utilise les outils de **mise en forme**. Par exemple, tu peux :

- **Changer la couleur des cellules** pour mettre en évidence certaines informations.
- **Ajouter des bordures** pour délimiter des zones.
- **Modifier l'alignement** du texte dans les cellules.

Tout cela se trouve dans l'onglet **Accueil**, dans la section "Police" et "Alignement".

Créer des graphiques

Un des points forts d'Excel, c'est la possibilité de créer des graphiques. Par exemple, si tu as une liste de notes et que tu veux voir leur évolution en graphique :

- Sélectionne tes données.
- Clique sur **Insertion** > **Graphiques**.
- Choisis le type de graphique que tu veux (histogramme, graphique en ligne, etc.).

Cela te permet de visualiser rapidement des tendances ou des comparaisons.

Quelques exemples d'utilisation d'Excel

Gestion de tes notes scolaires

Tu peux créer un tableau avec tes matières et tes notes pour chaque trimestre. Ensuite, utilise la fonction **MOYENNE** pour calculer ta moyenne générale et voir dans quelles matières tu es le plus fort ou ce que tu dois améliorer.

MATIÈRE	TRIMESTRE 1	TRIMESTRE 2	TRIMESTRE 3	MOYENNE
Mathématiques	15	16	14	=MOYENNE(B2:D2)
Français	12	14	13	=MOYENNE(B3:D3)
Histoire	17	15	16	=MOYENNE(B4:D4)

Budget personnel

Excel peut aussi t'aider à gérer ton argent de poche ou tes dépenses. Tu peux lister tes revenus (argent de poche, petits boulots, etc.) et tes dépenses (cinéma, jeux vidéo, vêtements…).

CATÉGORIE	REVENUS	DÉPENSES
Argent de poche	50	
Cinéma		10
Jeux vidéo		30
Vêtements		20

Ensuite, utilise des fonctions simples pour faire les calculs :

- **Total des revenus** : `=SOMME(B2:B3)`
- **Total des dépenses** : `=SOMME(C2:C4)`
- **Solde final** : `=Total des revenus - Total des dépenses`

Les bonnes pratiques en Excel

1. **Sauvegarde fréquemment** : Rien de pire que de perdre son travail. Pense à sauvegarder régulièrement, surtout quand tu travailles sur un projet important.
2. **Utilise des titres clairs** : Donne des noms significatifs à tes colonnes et tes lignes pour que tu saches toujours de quoi il s'agit.
3. **Vérifie tes formules** : Si une formule ne marche pas, regarde bien les cellules que tu as sélectionnées ou vérifie si tu n'as pas oublié un signe égal.

Conclusion

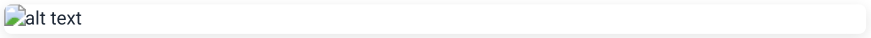
Excel est un outil très puissant qui peut te rendre service dans de nombreuses situations, que ce soit pour l'école, pour ton argent ou même pour organiser tes loisirs. Avec un peu de pratique, tu verras que tu deviendras rapidement un pro des tableaux et des graphiques. Alors n'hésite pas à explorer, essayer, et surtout, amuse-toi avec Excel !

Tu as des questions ou tu veux un coup de main pour tes exercices Excel ? N'hésite pas à demander, je suis là pour t'aider !

Sélectionner des cellules dans Excel

> *Dans Excel, tu vas souvent devoir **sélectionner des cellules** pour ensuite **modifier**, **formater** ou **supprimer** des données. Selon ce que tu veux sélectionner, tu utiliseras la souris, la touche Shift ou la touche Ctrl. L'objectif est que tu puisses choisir la bonne méthode automatiquement.*

Dans Excel, tu peux sélectionner des **cellules**, des **lignes** ou des **colonnes** soit avec la **souris**, soit avec **Shift**, soit avec **Ctrl**.



Voici les méthodes essentielles, présentées de façon progressive.

Sélection avec la souris

Cellule(s) simple(s)

- Clique sur une cellule pour la sélectionner.

Ligne ou colonne entière

- Clique sur le **numéro** (ligne) ou la **lettre** (colonne).

Plusieurs cellules consécutives

- Clique sur la première cellule.
- Garde le clic enfoncé.
- Glisse jusqu'à la dernière.

Exemple : A1 → glisser → C3.

Plusieurs lignes / colonnes consécutives

- Clique sur une ligne/colonne.
- Maintiens le bouton de la souris enfoncé et glisse vers le haut/bas (lignes) ou gauche/droite (colonnes).

Sélection avec Shift (pour des zones consécutives)

Pour sélectionner des cellules consécutives / contigües / qui se touchent:

Cellules contiguës

1. Clique sur la première cellule.
2. Maintiens **Shift**.
3. Clique sur la dernière cellule.

Lignes ou colonnes contiguës

Même principe :

1. Clique sur la première ligne/colonne.
2. Shift.
3. Clique sur la dernière ligne/colonne.

Exemple : sélectionner les lignes 3 à 10 → ligne 3 → Shift → ligne 10.

Sélection avec Ctrl (pour des zones non adjacentes)

Cellules non consécutives

1. Clique sur une cellule.
2. Maintiens **Ctrl**.
3. Clique sur les autres.

Lignes / colonnes non consécutives

Même méthode :

1. Clique sur une ligne/colonne.

2. Maintiens **Ctrl**.
3. Clique sur les autres.

---

4. Petit tableau récapitulatif

OBJECTIF	MÉTHODE	EXEMPLE
Plage consécutive	Souris **ou** Shift	A1 → glisser → C4 / ou A1 → Shift → C4
Lignes/colonnes consécutives	Souris ou Shift	Ligne 2 → glisser → Ligne 5
Zones non consécutives	**Ctrl**	A1 + C3 + F7 / Ligne 2 + Ligne 6 / Colonnes A + D

Si tu veux, je peux encore : ➡️ faire une version *encore plus courte* (mode mémo 10 lignes) ➡️ transformer ça en *notes de cours explicites* avec objectifs + 5 notions clés + exemples élèves.

Sélectionner des cellules dans Excel - Pratique 💪

> *Essaie ces quelques sélections.*

Objectif de la fiche

T'entraîner à sélectionner des cellules, des lignes et des colonnes avec :

- la **souris**
- **Shift** (Maj)
- **Ctrl**

Exercice 1 — Sélections simples

1. Sélectionne la cellule **B3**.
2. Sélectionne la **ligne 5**.
3. Sélectionne la **colonne C**.
4. Sélectionne **toute la feuille** (2 méthodes).

Exercice 2 — Plages consécutives (qui se touchent)

1. Sélectionne la plage **A1** → **C4** avec la **souris**.
2. Sélectionne la plage **B2** → **F10** avec **Shift**.
3. Sélectionne les lignes **4 à 9** avec **Shift**.
4. Sélectionne les colonnes **A à E** en glissant à la **souris**.

Exercice 3 — Plages non consécutives (qui ne se touchent pas)

1. Sélectionne les cellules **A1**, **C1**, **E1** (Ctrl).
2. Sélectionne les lignes **2**, **5** et **9** (Ctrl).
3. Sélectionne les colonnes **A**, **D** et **F** (Ctrl).
4. Combine : sélectionne **A2** → **C4**, puis ajoute **E2** → **F3** (Ctrl).

Exercice 4 — Raccourcis clavier

1. Depuis la cellule C5, sélectionne toute la **colonne C**.
2. Depuis la cellule D3, sélectionne toute la **ligne 3**.
3. Depuis A1, sélectionne toutes les cellules remplies vers le bas.
4. Sélectionne le **tableau complet**, puis la **feuille complète**.

(Indices : Ctrl+Espace, Shift+Espace, Ctrl+Shift+Flèche, Ctrl+A)

Exercice 5 — Mini-problèmes

1. Tu veux mettre en gras **toutes les titres** situés en **A1, C1, F1** → quelle méthode utilises-tu ?
2. Tu veux formater une grande zone de **A1 à M50**, tu choisis Shift ou la souris ? Pourquoi ?
3. Ton tableau est dispersé : colonnes **A** et **C** doivent être effacées, mais pas **B**. Comment les sélectionnes-tu ?

✓ Auto-évaluation rapide

Coche ce que tu maîtrises :

- [ ] Je sais sélectionner des cellules avec la souris.

- [ ] Je sais utiliser Shift pour sélectionner une plage.
- [ ] Je sais utiliser Ctrl pour sélectionner des zones séparées.
- [ ] Je sais sélectionner lignes, colonnes et toute la feuille.
- [ ] Je connais Ctrl+A, Ctrl+Espace, Shift+Espace.

Excel: créer des séries

> *Dans Excel, la création de séries est une fonctionnalité qui permet d'automatiser et accélérer la saisie de données répétitives ou progressives. Qu'il s'agisse de listes de dates, de nombres, ou même de textes, Excel propose plusieurs façons de créer ces séries de manière rapide et efficace.*

Dans cet article, nous allons explorer les différentes méthodes courantes et utiles pour créer des séries dans Excel, ainsi que les avantages de chaque approche. Tu apprendras également comment ajuster ces séries selon tes besoins.

Remplissage automatique (Handle de recopie)

Qu'est-ce que le **remplissage automatique** ?

Le **remplissage automatique** est l'une des méthodes les plus simples et les plus utilisées pour créer des séries dans Excel. Il te permet de remplir rapidement une plage de cellules en fonction d'un modèle que tu as commencé. Il suffit de saisir une valeur ou une suite de valeurs et de tirer sur la poignée de recopie située dans le coin inférieur droit de la cellule pour étendre la série.

Exemples courants de séries avec le remplissage automatique :

- **Les nombres** : Si tu tapes "1" dans une cellule et que tu tires la poignée vers le bas, Excel va recopier le chiffre 1. Cependant, si tu tapes "1" dans une cellule, puis "2" dans la cellule suivante, Excel comprendra que tu souhaites créer une série et continuera avec "3", "4", "5", etc.

  Exemple :

A
1
2
3
4
5

- **Les jours de la semaine** : Si tu tapes "Lundi" dans une cellule et que tu fais glisser la poignée, Excel va automatiquement compléter la série avec "Mardi", "Mercredi", "Jeudi", etc.

  Exemple :

A
Lundi
Mardi
Mercredi
Jeudi
Vendredi

- **Les mois de l'année** : De la même manière, si tu écris "Janvier" et que tu tires la poignée, Excel remplira les cellules suivantes avec "Février", "Mars", et ainsi de suite.

  Exemple :

A
Janvier
Février
Mars
Avril
Mai

Comment utiliser le remplissage automatique ?

1. Saisis la première valeur de la série (et éventuellement la seconde si c'est une suite numérique).
2. Clique sur le coin inférieur droit de la cellule pour activer la **poignée de recopie**.
3. Fais glisser vers le bas (ou vers la droite) pour étendre la série.

Astuce : Tu peux également cliquer sur la petite icône qui apparaît après avoir tiré la poignée pour choisir entre "Copier les valeurs" ou "Remplir la série". Cela te donne un peu plus de contrôle sur le type de remplissage automatique.

Avantages du remplissage automatique

- **Rapide** et facile à utiliser pour des séries courantes comme les jours de la semaine, les mois, ou des suites numériques.
- **Personnalisable** : Si tu donnes un modèle (par exemple, 1 et 3), Excel comprendra que tu veux des nombres impairs et continuera avec 5, 7, 9, etc.
- **Intelligent** : Il reconnaît automatiquement certains types de données (comme les dates et les heures).

Principales séries de base

Excel dispose de plusieurs **séries par défaut** que tu peux utiliser directement via le **remplissage automatique**. Voici une liste des principales séries intégrées dans Excel :

Les jours de la semaine

- **Nom complet** : Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche
- **Abréviation** : Lun, Mar, Mer, Jeu, Ven, Sam, Dim

Tu peux commencer par n'importe quel jour, et Excel complétera automatiquement la série.

Les mois de l'année

- **Nom complet** : Janvier, Février, Mars, Avril, Mai, Juin, Juillet, Août, Septembre, Octobre, Novembre, Décembre
- **Abréviation** : Janv, Fév, Mars, Avr, Mai, Juin, Juil, Août, Sept, Oct, Nov, Déc

Comme pour les jours de la semaine, tu peux commencer avec n'importe quel mois, et Excel complétera la série.

Les nombres

- Séries linéaires de nombres consécutifs : 1, 2, 3, 4, 5, ...
- Séries personnalisées : Tu peux indiquer un pas (ex. 1, 3, 5, 7… ou 10, 20, 30…) en indiquant les deux premiers nombres avant de tirer sur la poignée de recopie.

Les dates

- **Jours consécutifs** : Excel incrémente automatiquement les jours. Si tu saisis une date et tires vers le bas, il ajoutera un jour à chaque cellule.
- **Jours de la semaine** : Si tu veux uniquement les jours ouvrés, tu peux utiliser la poignée de recopie avec une option de remplissage "jours de semaine uniquement".
- **Mois** : Si tu saisis une date et tires vers la droite, Excel peut incrémenter par mois.
- **Années** : Excel est également capable de compléter une série d'années consécutives.

Heures

- Tu peux créer une série d'heures en commençant par une heure de départ (par exemple, 09:00), et Excel ajoutera une heure supplémentaire à chaque cellule lorsque tu tires la poignée de recopie.
- Tu peux aussi fournir 2 heures, les sélectionner, et Excel ajoutera la différence entre les 2 à chaque cellule lorsque tu tires la poignée de recopie

Les trimestres

- **Abréviations** : T1, T2, T3, T4
- **Nom complet** : Trimestre 1, Trimestre 2, Trimestre 3, Trimestre 4

Excel reconnaît les trimestres et peut les compléter automatiquement.

Les formats textuels séquentiels

- Si tu saisis un texte suivi d'un nombre, comme "Semaine 1", Excel reconnaît le modèle et complétera la série avec "Semaine 2", "Semaine 3", etc.
- Cela fonctionne avec d'autres séquences similaires comme "Chapitre 1", "Partie 1", "Article 1", etc.

Séries personnalisées

En plus des séries par défaut, tu peux créer tes **propres listes personnalisées** (comme expliqué précédemment), qui seront disponibles pour les remplissages automatiques futurs.

Ces séries intégrées permettent de gagner du temps pour saisir des informations répétitives ou progressives dans tes tableaux, surtout lorsque tu travailles sur des projets qui impliquent des dates, des jours, ou des périodes récurrentes.

Conclusion

Il existe de nombreuses façons de créer des séries dans Excel, et chaque méthode a ses avantages en fonction du type de données avec lequel tu travailles. Le **remplissage automatique** est rapide et facile pour des séries courantes.

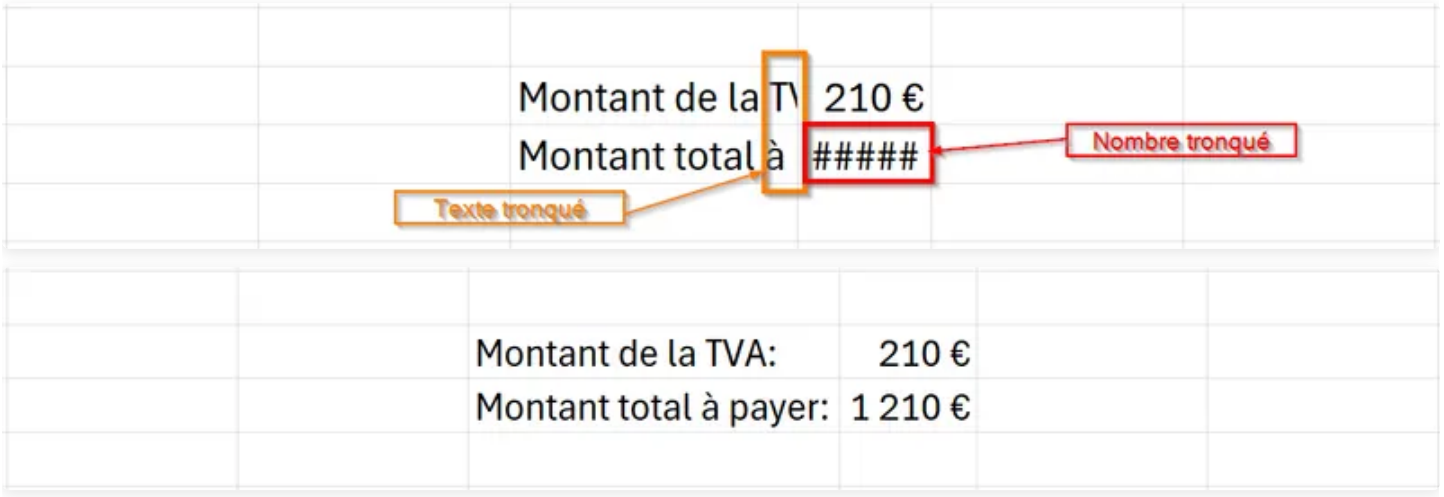
Modifier la taille des lignes et des colonnes dans Excel

> *Changer la hauteur d'une ligne ou la largeur d'une colonne permet d'améliorer la lisibilité d'un tableau. Voici les méthodes essentielles, présentées de manière simple et progressive.*

Pourquoi redimensionner ?

Modifier la taille des lignes et des colonnes, c'est avant tout une question de **mise en page** : tu rends ton tableau plus lisible, plus propre et mieux organisé.

C'est aussi indispensable pour **éviter que du texte soit caché** lorsqu'il est trop long pour entrer dans la cellule. En ajustant la hauteur ou la largeur, tu t'assures que tout le contenu reste visible.



Double flèche : le geste de base

Que ce soit pour une **ligne** ou une **colonne**, tout commence par la même action :

- Place ton curseur **entre deux lettres** (A | B) → une double flèche **horizontale** ↔ apparaît
- Place ton curseur **entre deux numéros** (3 | 4) → une double flèche **verticale** ↕ apparaît

Ce curseur indique que tu peux modifier la dimension.

Modifier la taille : manuel ou AutoFit

🖱 Méthode manuelle (glisser)

Quand la double flèche apparaît :

1. Clique.
2. **Glisse** pour agrandir ou réduire. → Simple et précis "à l'œil".

⚡ AutoFit (ajustement automatique)

Quand le curseur est en double flèche :

1. **Double-clique**. → Excel ajuste automatiquement la taille en fonction du contenu.

⌨ Valeur précise (hauteur/largeur exacte)

Pour une taille définie :

1. Clique-droit sur la ligne ou la colonne.
2. Choisis **Hauteur de ligne…** ou **Largeur de colonne…**
3. Encode une valeur numérique (ex. 20, 35…).

Modifier plusieurs lignes ou colonnes en même temps

Plusieurs colonnes

1. Sélectionne **plusieurs lettres** (ex. A à D).
2. Effectue l'une des actions :
   - glisser la bordure,
   - double-cliquer (AutoFit),
   - entrer une largeur précise. → Toutes les colonnes sélectionnées changent **en même temps**.

Plusieurs lignes

Même principe :

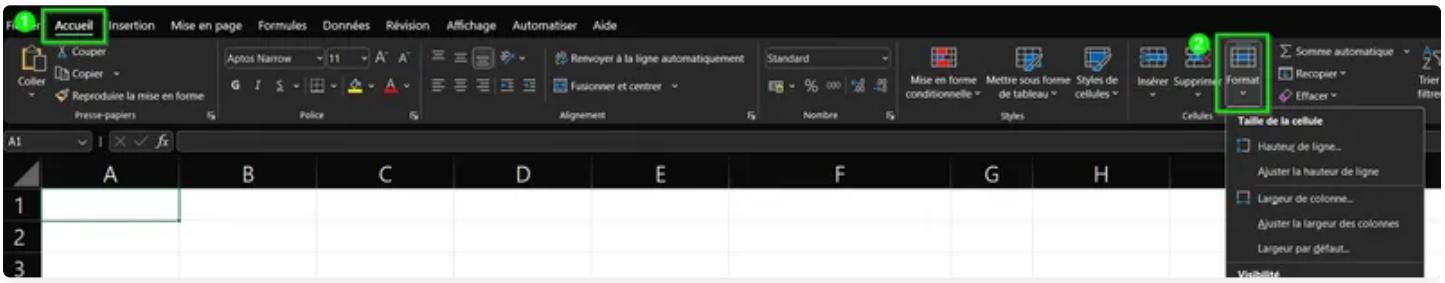
1. Sélectionne plusieurs numéros de lignes.
2. Glisse / double-clique / encode une hauteur.

Toutes les lignes ou toutes les colonnes

1. Clique le bouton **Sélectionner tout** (coin en haut à gauche).
2. Ajuste une bordure de ligne ou de colonne. → Toute la feuille est harmonisée.

Menu 'format'

Il est aussi possible de passer via la section "Format" du ruban:



Dans ce menu, il est aussi possible de spécifier directement la hauteur/largeur souhaitée.

Boss mode: le raccourci clavier de fou malade

`Ctrl-a` : sélectionne toutes les cellules d'un tableau

`Alt-L ç a` : ajuste automatiquement la largeur des colonnes sélectionnées `Alt-L ç l` : ajuste automatiquement la hauteur des lignes sélectionnées

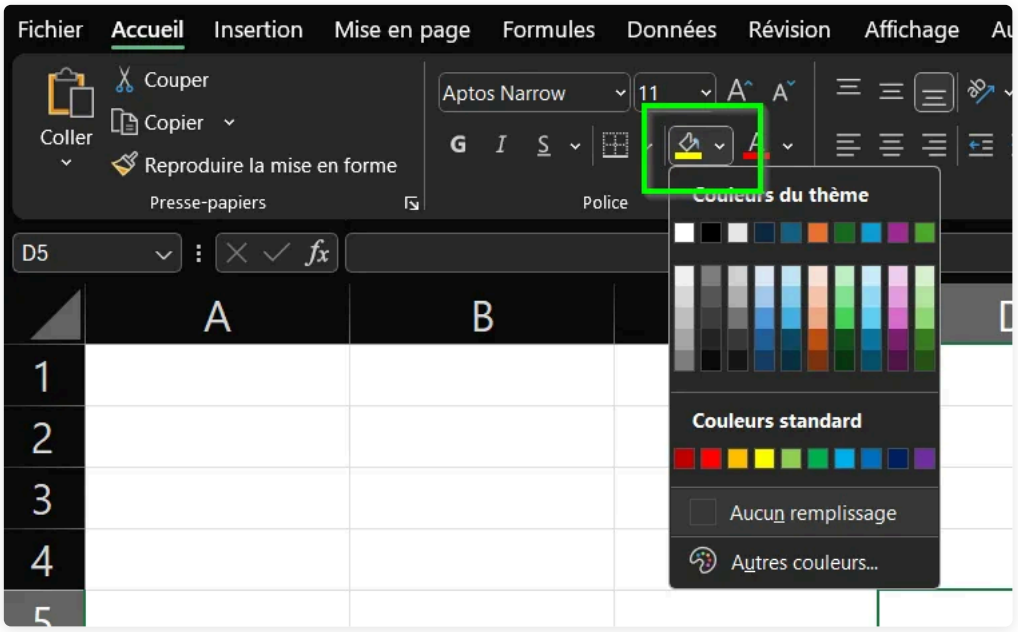
Sélection, hauteur et largeur: exercices

> *Voici des exercices pour te permettre de travailler les différents modes de sélection*

Utilise les meilleures techniques de sélection afin de réaliser ces images "pixel art" dans Excel.

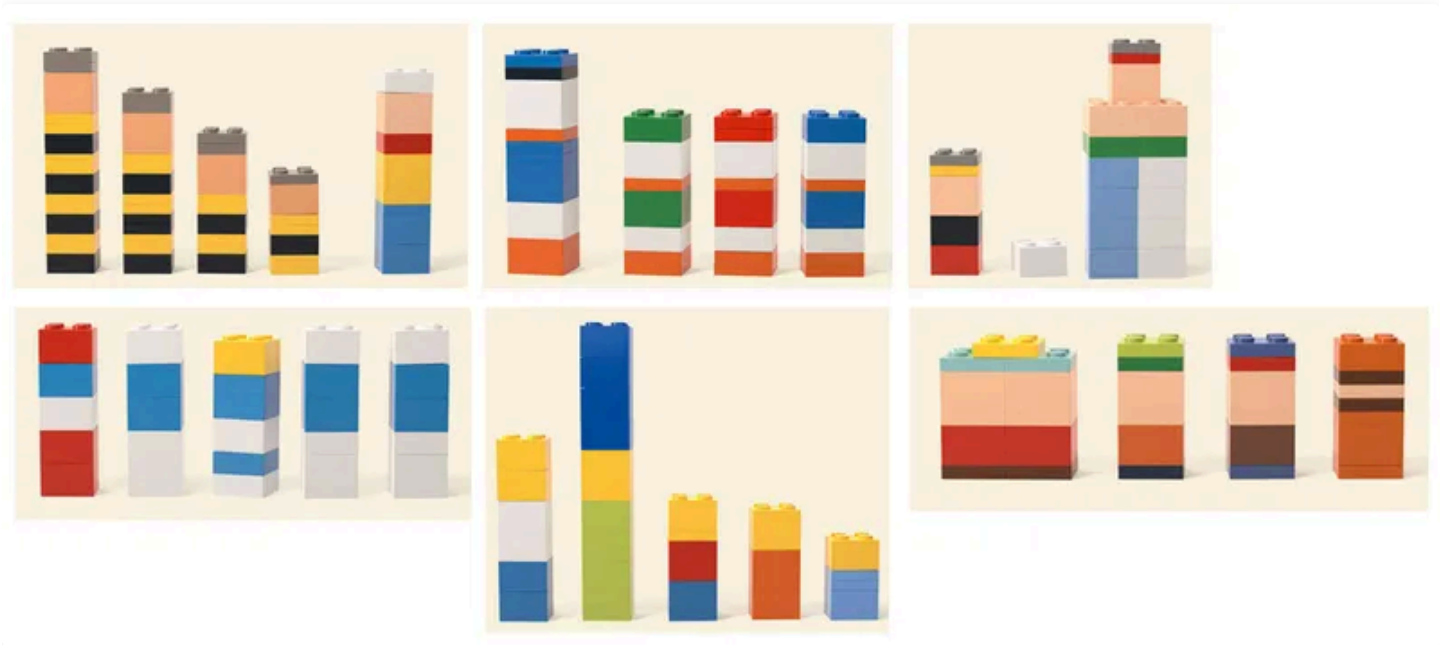
Adapte les hauteurs de lignes et largeurs de colonnes pour obtenir le meilleur résultat.

Utilise une couleur de remplissage pour changer la couleur des cellules (ça peut être approximatif).



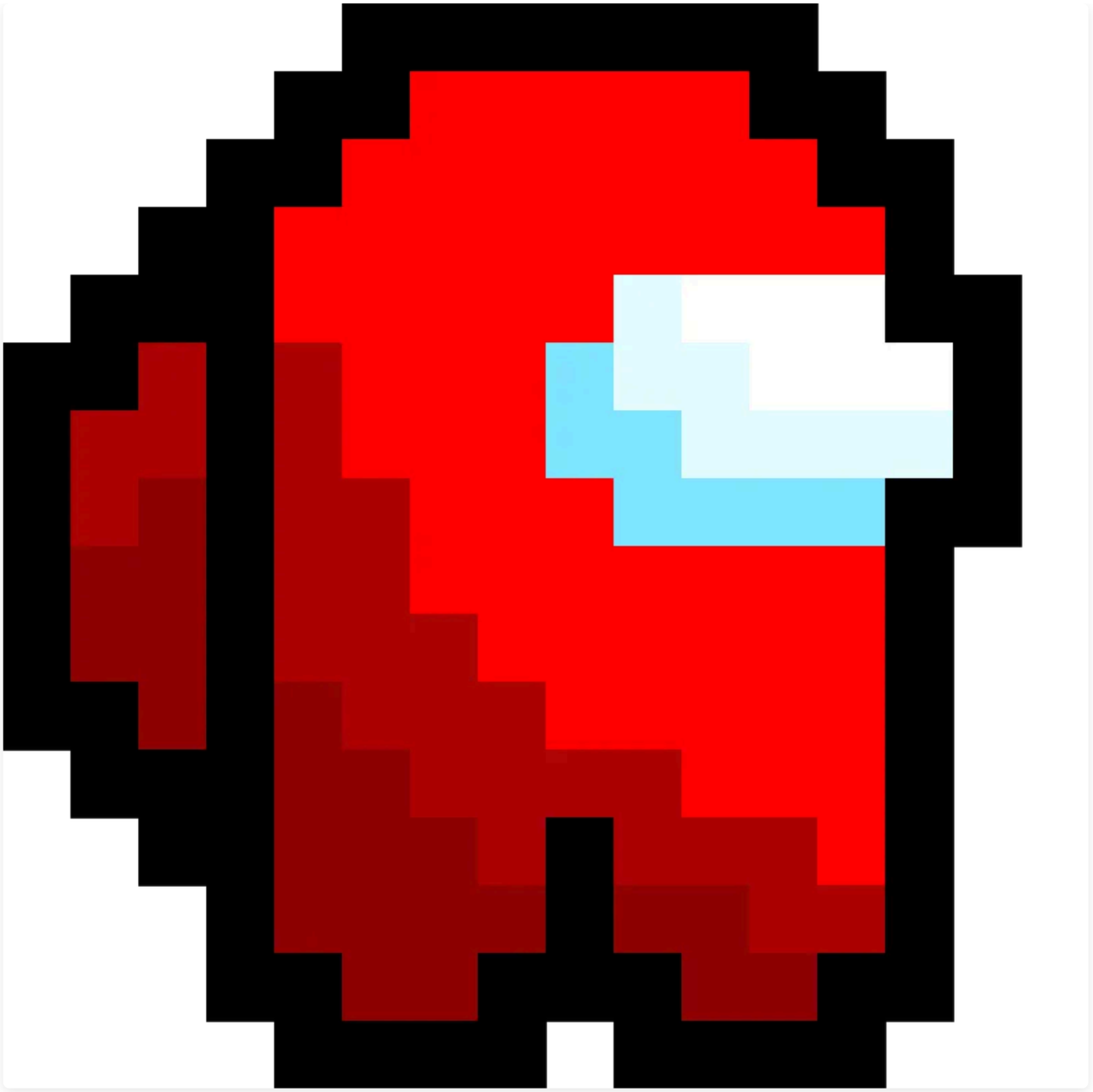
Légos

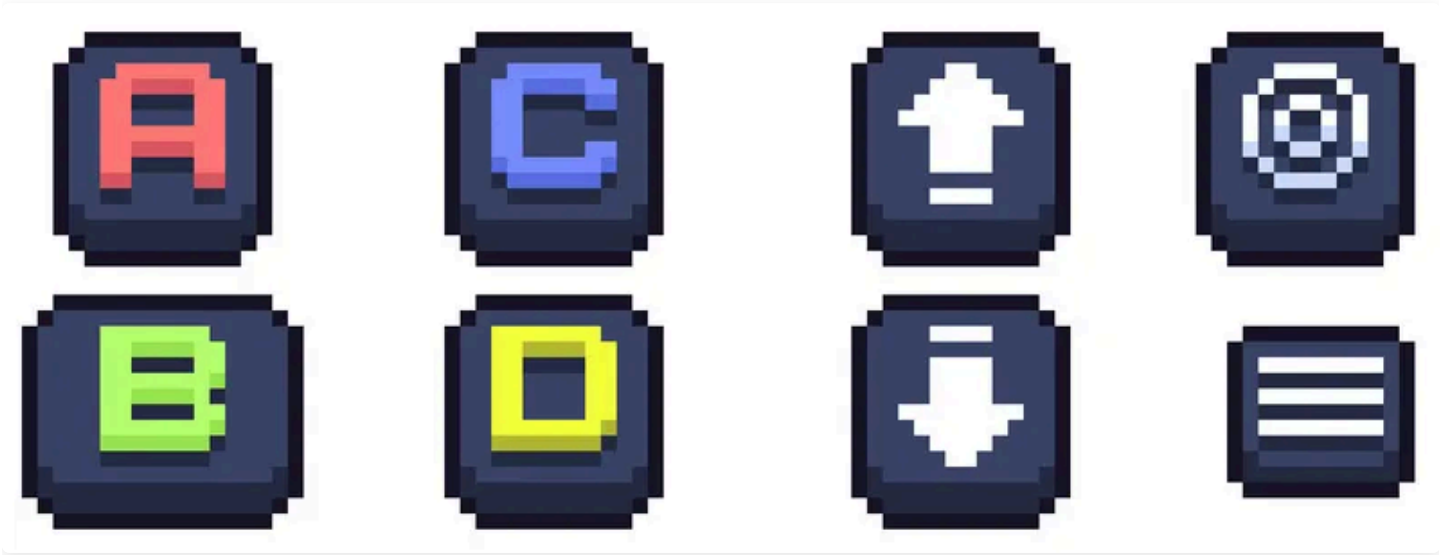
Réalise **3** images parmi ces propositions.



16x16 pixels

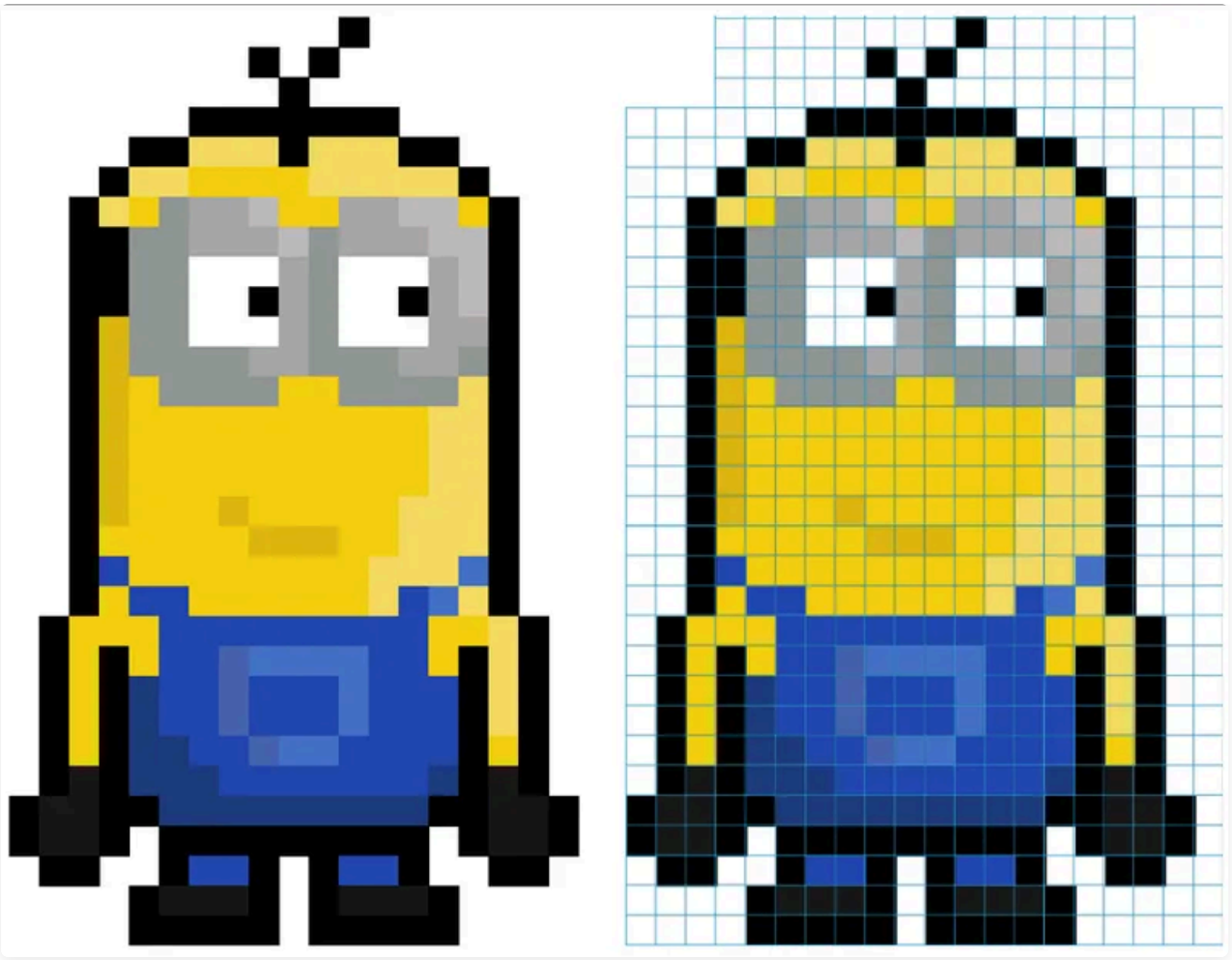
Réalise ce personnage et 2 boutons au choix à l'aide de 16 lignes et 16 colonnes.

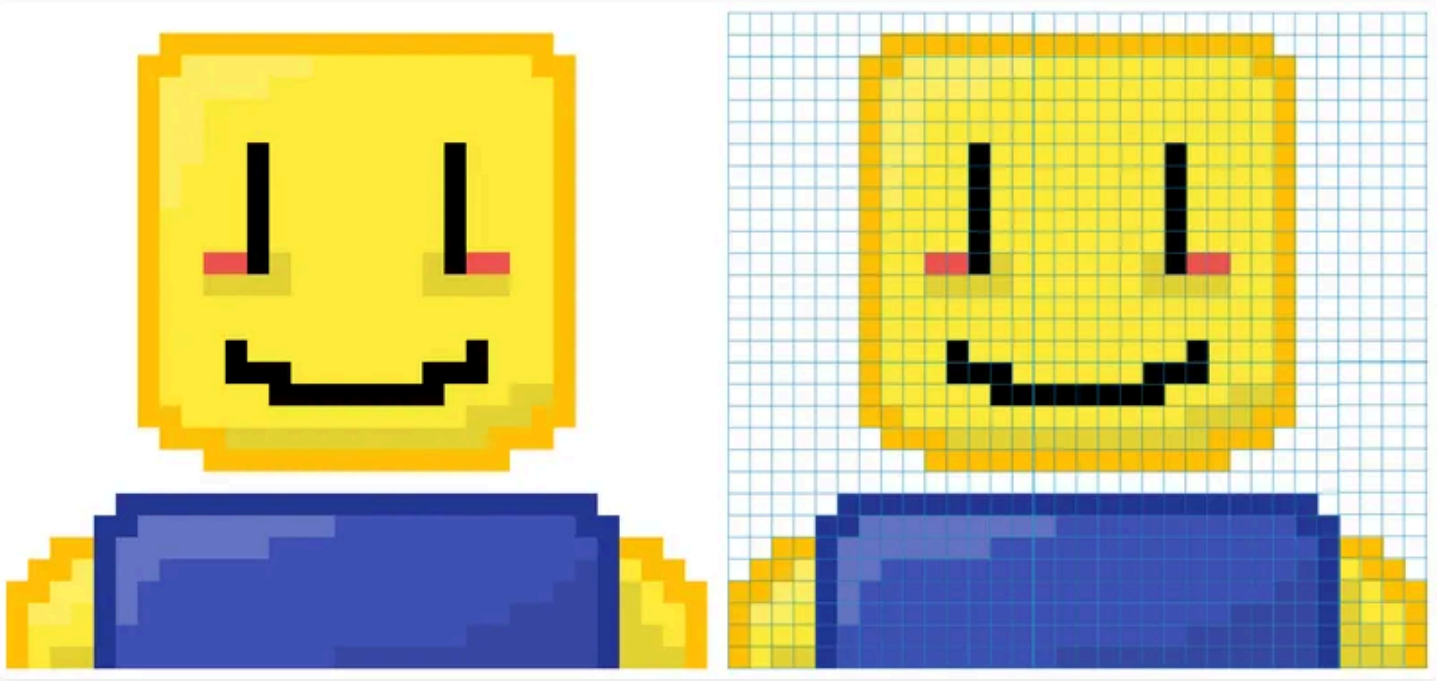




32x32 pixels

Réalise **un** de ces personnages à l'aide de 32 lignes et 32 colonnes.





Excel: Les formules

> *Microsoft Excel est un outil incontournable pour organiser, analyser et visualiser des données. Ce qui rend Excel particulièrement puissant, ce sont ses **formules**. Une formule est une équation que tu peux saisir dans une cellule pour effectuer des calculs automatiques, analyser des données ou effectuer diverses manipulations de texte et de dates.*

Dans cet article, nous allons explorer en détail ce que sont les formules dans Excel, te montrer comment elles fonctionnent, et passer en revue les **10 formules essentielles** pour les débutants. Nous terminerons en abordant la différence entre les formules en français et en anglais, et ce que cela signifie pour l'échange de fichiers entre ces deux versions.

Qu'est-ce qu'une formule dans Excel ?

Une **formule** dans Excel est une expression qui permet d'effectuer des calculs sur des valeurs. Elle commence toujours par un signe égal (`=`), suivi de l'opération ou de la fonction que tu veux utiliser. Excel propose un grand nombre de fonctions intégrées qui peuvent être utilisées dans les formules pour simplifier des tâches répétitives ou complexes.

Exemple simple de formule :

```
=A1 + A2
```

Cela additionne les valeurs des cellules **A1** et **A2**.

Les formules peuvent être aussi simples que des additions, mais elles peuvent également inclure des calculs complexes, des fonctions, des références à d'autres cellules, et même des conditions.

---

10 principales formules à connaître pour débuter

Pour bien commencer avec Excel, voici les **10 formules les plus utiles** pour les débutants. Elles couvrent les calculs de base, la gestion de données, et les premières étapes vers des analyses plus poussées.

1. SOMME()

- **Description** : Additionne un ensemble de valeurs dans une plage de cellules.
- **Syntaxe** : `=SOMME(plage)`
- **Exemple** : `=SOMME(A1:A5)` additionne les valeurs des cellules A1 à A5.

2. MOYENNE()

- **Description** : Calcule la moyenne d'un ensemble de valeurs.
- **Syntaxe** : `=MOYENNE(plage)`
- **Exemple** : `=MOYENNE(A1:A5)` calcule la moyenne des cellules A1 à A5.

3. MAX()

- **Description** : Renvoie la valeur maximale d'une plage de cellules.
- **Syntaxe** : `=MAX(plage)`
- **Exemple** : `=MAX(A1:A5)` renvoie la valeur la plus élevée des cellules A1 à A5.

4. MIN()

- **Description** : Renvoie la valeur minimale d'une plage de cellules.
- **Syntaxe** : `=MIN(plage)`
- **Exemple** : `=MIN(A1:A5)` renvoie la valeur la plus faible des cellules A1 à A5.

5. SI()

- **Description** : Effectue une comparaison et renvoie une valeur si la condition est vraie, et une autre si elle est fausse.
- **Syntaxe** : `=SI(condition, valeur_si_vrai, valeur_si_faux)`
- **Exemple** : `=SI(A1>10, "Supérieur", "Inférieur")` renvoie "Supérieur" si la valeur de A1 est supérieure à 10, sinon renvoie "Inférieur".

6. NB()

- **Description** : Compte le nombre de cellules contenant des nombres dans une plage.
- **Syntaxe** : `=NB(plage)`
- **Exemple** : `=NB(A1:A10)` compte le nombre de cellules qui contiennent des nombres entre A1 et A10.

7. NB.SI()

- **Description** : Compte le nombre de cellules qui répondent à une condition donnée.
- **Syntaxe** : `=NB.SI(plage, critère)`
- **Exemple** : `=NB.SI(A1:A10, ">5")` compte le nombre de cellules contenant une valeur supérieure à 5 dans la plage A1 à A10.

8. RECHERCHEV()

- **Description** : Recherche une valeur dans la première colonne d'un tableau et renvoie une valeur située dans la même ligne.
- **Syntaxe** : `=RECHERCHEV(valeur, tableau, index_colonne, [valeur_proche])`
- **Exemple** : `=RECHERCHEV("Jean", A1:B10, 2, FAUX)` cherche "Jean" dans la colonne A et renvoie la valeur correspondante dans la colonne B.

9. AUJOURDHUI()

- **Description** : Renvoie la date actuelle.
- **Syntaxe** : `=AUJOURDHUI()`
- **Exemple** : Utilisé seul, `=AUJOURDHUI()` affiche la date du jour actuel.

10. CONCATENER()

- **Description** : Combine plusieurs textes ou valeurs de cellules en une seule.
- **Syntaxe** : `=CONCATENER(texte1, texte2, ...)`
- **Exemple** : `=CONCATENER(A1, " ", B1)` combine le contenu des cellules A1 et B1 avec un espace entre les deux.

Les différences entre Excel en français et en anglais

Excel est disponible dans de nombreuses langues, et les noms des fonctions changent en fonction de la langue utilisée. Par exemple, une fonction comme **SOMME()** en français s'appelle **SUM()** en anglais. Si tu travailles avec un fichier Excel créé dans une version anglaise ou si tu passes d'une version linguistique à une autre, il est important de comprendre comment Excel gère ces différences.

LES FORMULES DOIVENT-ELLES ÊTRE TRADUITES ?

- Oui, si tu utilises une version française d'Excel, tu dois écrire les formules avec les noms français des fonctions, comme `SOMME()`, `MOYENNE()`, `SI()`, etc.
- Dans la version anglaise, les formules équivalentes sont `SUM()`, `AVERAGE()`, `IF()`, etc.

LES FORMULES FONCTIONNENT-ELLES DANS UNE AUTRE LANGUE D'EXCEL ?

Si tu ouvres un fichier créé dans une version Excel anglaise avec Excel en français, Excel **convertira automatiquement** les formules en utilisant les noms de fonctions en français.

Exemple : Si un fichier anglais contient la formule `=SUM(A1:A5)`, elle sera automatiquement convertie en `=SOMME(A1:A5)` lorsque tu l'ouvriras avec Excel en français.

Cependant, cette conversion ne se fait pas dans l'autre sens si tu copies-colles une formule en texte (sans passer par l'enregistrement et l'ouverture de fichiers). Si tu colles une formule anglaise directement dans une version française d'Excel, tu obtiendras une erreur, car Excel ne reconnaît pas les noms de fonctions en anglais.

Pourquoi c'est important ?

- **Collaboration internationale** : Si tu travailles avec des collègues ou des clients dans d'autres pays, il est essentiel de comprendre que les formules changent selon la langue de la version Excel.
- **Conversion automatique** : Heureusement, Excel fait une bonne partie du travail de conversion à ta place lorsque tu ouvres des fichiers provenant d'autres versions linguistiques. Cela te permet de collaborer plus facilement avec des utilisateurs d'Excel dans différentes langues.

Conclusion

Les formules sont l'un des outils les plus puissants d'Excel, et comprendre les formules de base comme **SOMME()**, **SI()**, et **RECHERCHEV()** te permettra de débuter avec confiance. Même si tu utilises une version d'Excel en français, il est important de savoir que les formules peuvent être automatiquement traduites lorsque tu ouvres des fichiers dans d'autres langues, facilitant la collaboration entre utilisateurs internationaux.

Si tu débutes dans Excel, prends le temps de pratiquer ces formules et d'expérimenter avec des fichiers simples. Avec ces compétences de base, tu seras bientôt capable d'utiliser Excel pour résoudre des problèmes complexes et améliorer ton efficacité au quotidien.

Excel: opérateurs et fonctions

> *Excel est un outil incontournable pour manipuler des données. Deux des opérations les plus courantes que tu rencontreras sont l'addition avec l'opérateur **'+'** et l'utilisation de la fonction **SOMME()**. Bien que ces deux méthodes te permettent de réaliser des additions, elles fonctionnent différemment et ont des avantages et des inconvénients en fonction du contexte.*

Dans cet article, nous allons examiner en détail la différence entre ces **deux méthodes d'addition**, et comment elles se comportent lorsque tu insères des lignes ou des colonnes dans tes tableaux Excel.

L'opérateur "+" dans Excel

Qu'est-ce que l'opérateur "+" ?

L'opérateur **"+"** permet d'additionner des cellules individuellement, comme tu le ferais avec une calculatrice. Par exemple, pour additionner les cellules **A1** et **A2**, tu peux écrire la formule suivante :

```
=A1 + A2
```

Cela te donnera la somme des valeurs présentes dans ces deux cellules. Si tu souhaites additionner plus de cellules, tu dois ajouter chaque cellule à la formule :

```
=A1 + A2 + A3 + A4
```

Avantages de l'opérateur "+"

- **Simplicité** : Il est intuitif à utiliser quand tu travailles avec un petit nombre de cellules. Si tu dois additionner juste quelques valeurs, l'opérateur "+" est rapide et direct.
- **Contrôle précis** : Tu sais exactement quelles cellules tu additionnes, car elles sont toutes mentionnées explicitement dans la formule.

Inconvénients de l'opérateur "+"

- **Peu pratique pour de longues séries de données** : Si tu as beaucoup de cellules à additionner (par exemple, une colonne de 100 cellules), écrire `=A1 + A2 + A3 + ... + A100` devient fastidieux et source d'erreurs.
- **Problème lors de l'insertion de lignes ou colonnes** : Si tu insères une nouvelle cellule entre deux cellules utilisées dans une formule, Excel ne l'ajoutera pas automatiquement à la formule. Par exemple, si tu insères une nouvelle ligne entre A1 et A2, la formule `=A1 + A2` ne changera pas, même si la nouvelle cellule contient des données.

La fonction SOMME() GB *SUM()*

Qu'est-ce que la fonction SOMME() ?

La fonction **SOMME()** est une fonction prédéfinie d'Excel qui te permet d'additionner rapidement une plage de cellules. Au lieu de spécifier chaque cellule individuellement, tu peux définir une plage de cellules :

```
=SOMME(A1:A4)
```

Cela additionne toutes les cellules entre A1 et A4, inclusivement. Tu peux également additionner plusieurs plages de cellules avec SOMME, comme ceci :

```
=SOMME(A1:A4, B1:B4)
```

Avantages de la fonction SOMME()

- **Facilité d'utilisation pour les grandes séries de données** : Si tu dois additionner une longue série de cellules, tu peux simplement indiquer la plage et Excel s'occupe du reste. Par exemple, `=SOMME(A1:A100)` est beaucoup plus simple que de lister 100 cellules.
- **Automatisation** : Si tu insères une nouvelle cellule dans la plage couverte par SOMME, Excel met à jour automatiquement la fonction pour inclure la nouvelle cellule. Si tu insères une ligne entre A1 et A4, la fonction `=SOMME(A1:A4)` deviendra automatiquement `=SOMME(A1:A5)`, ce qui rend la formule plus flexible.
- **Lisibilité** : Les formules utilisant SOMME sont plus faciles à lire et à comprendre, surtout lorsque tu travailles avec de grandes plages de données. Cela permet également de réduire les erreurs de saisie.

Inconvénients de la fonction SOMME()

- **Moins de contrôle sur les cellules individuelles** : Si tu veux additionner des cellules non adjacentes de manière très spécifique, il peut être plus pratique d'utiliser l'opérateur "+". Par exemple, additionner uniquement les cellules **A1**, **A3** et **A5** peut être plus clair avec `=A1 + A3 + A5` plutôt qu'avec `=SOMME(A1, A3, A5)`, surtout si la formule devient plus complexe.

3. Comparaison : L'impact de l'insertion de lignes et colonnes

Lorsque tu travailles avec des formules Excel, tu finiras souvent par ajouter ou supprimer des lignes et des colonnes. Il est donc important de comprendre comment cela affecte les formules qui utilisent l'opérateur "**+**" et la fonction **SOMME()**.

a. Avec l'opérateur "+"

Si tu utilises l'opérateur "**+**" pour additionner des cellules, Excel ne réagit pas bien aux changements de la structure du tableau. Par exemple :

1. **Avant insertion** :

   ```
   =A1 + A2 + A3
   ```

   Imaginons que A1, A2 et A3 contiennent les valeurs 10, 20 et 30 respectivement. La formule donnera donc **60**.

2. **Après insertion d'une nouvelle ligne entre A1 et A2** :

   La nouvelle cellule A2 contient une nouvelle valeur, mais la formule reste `=A1 + A2 + A3`, donc elle additionne les cellules incorrectes. Si la nouvelle cellule insérée contient des données importantes, elles ne seront pas prises en compte dans la somme.

   Résultat : La nouvelle valeur n'est pas incluse, et ta formule est fausse à moins que tu la mettes à jour manuellement.

b. Avec la fonction SOMME()

Lorsque tu utilises la fonction **SOMME()**, Excel est plus "intelligent" en cas d'insertion de nouvelles lignes ou colonnes :

1. **Avant insertion** :

   ```
   =SOMME(A1:A3)
   ```

   La fonction additionne les cellules A1, A2 et A3, tout comme l'opérateur "+".

2. **Après insertion d'une nouvelle ligne entre A1 et A2** :

   La fonction **SOMME()** met automatiquement à jour la plage de cellules, et la nouvelle cellule insérée est incluse dans la somme. La nouvelle formule devient automatiquement `=SOMME(A1:A4)`.

   Résultat : La nouvelle cellule est incluse dans la formule sans intervention manuelle.

4. Quand utiliser l'opérateur "+" ou la fonction SOMME() ?

Utilise l'opérateur "+" :

- Lorsque tu additionnes **quelques cellules spécifiques** et que tu veux un contrôle précis sur chaque cellule utilisée.
- Si les cellules que tu additionnes ne sont pas adjacentes et que tu ne veux pas gérer des plages de cellules.

Utilise la fonction SOMME() :

- Quand tu travailles avec **de longues plages de cellules** (par exemple, plusieurs dizaines ou centaines de cellules).
- Si tu veux une **formule flexible** qui s'adapte aux modifications du tableau (insertion de lignes/colonnes).
- Pour **gagner du temps** et éviter les erreurs de saisie dans des tableaux de grande taille.

Conclusion

L'opérateur "**+**" et la fonction **SOMME()** sont deux méthodes puissantes pour additionner des cellules dans Excel, mais elles ne sont pas interchangeables. L'opérateur "**+**" est idéal pour des ajouts simples et spécifiques, tandis que la fonction **SOMME()** est plus adaptée pour de grandes plages de données et offre plus de flexibilité lorsque tu modifies ton tableau.

En résumé, si tu travailles sur des petits ensembles de données et que tu veux avoir un contrôle total, utilise l'opérateur "**+**". Si tu manipules de grandes quantités de données ou prévois de modifier fréquemment ton tableau, la fonction **SOMME()** est une bien meilleure option.

Excel - Les références absolues

> *Aujourd'hui, on va plonger dans le monde des références absolues. Vous allez apprendre ce qu'elles sont, pourquoi elles sont importantes et comment les utiliser avec des exemples concrets.*

1. Qu'est-ce qu'une Référence Absolue?

Dans Excel, une référence absolue est un type de **référence de cellule**. Contrairement aux références relatives, une référence absolue **ne change pas quand tu copies la formule à une autre cellule**. Elle est représentée par le symbole `$` devant le numéro de colonne et/ou de ligne. Par exemple, `$A$1` est une référence absolue.

2. À quoi ça sert?

Les références absolues sont utiles quand tu veux **conserver une référence *fixe* à une cellule spécifique**. Par exemple, si tu as un taux de TVA dans une cellule que tu veux utiliser dans plusieurs calculs, tu utiliseras une référence absolue pour te référer à cette cellule.

3. Quand les utiliser?

Utilise les références absolues :

- Quand tu as des **constantes dans une formule**, comme des taux fixes ou des valeurs spécifiques.
- Dans les formules qui doivent toujours se référer à une cellule spécifique, peu importe où la formule est copiée.

4. Exemples concrets

Exemple 1: Calcul du Prix TTC

Imaginons que tu as un prix HT dans la cellule A2 et le taux de TVA dans la cellule B1. Tu veux calculer le prix TTC dans la cellule C2.

1. Entre le prix HT dans `A2`, disons 100€.
2. Entre le taux de TVA dans `B1`, par exemple 20%.
3. Dans `C2`, entre la formule `=A2 + (A2 * $B$1)`.

Ici, `$B$1` est une référence absolue. Quand tu copies cette formule vers le bas, le taux de TVA référencé reste le même.

Exemple 2: Référence à une Cellule Fixe pour une Liste

Supposons que tu aies une liste de prix dans la colonne A et que tu veuilles appliquer une remise stockée dans la cellule B1 à tous ces prix.

1. Entre la remise dans `B1`, disons 10%.
2. Dans la cellule `B2`, entre la formule `=A2 * (1 - $B$1)`.
3. Copie cette formule vers le bas pour appliquer la remise à tous les prix.

Dans cet exemple, peu importe où la formule est copiée, elle se référera toujours à la remise dans la cellule B1.

Conclusion

Les références absolues sont un outil puissant dans Excel, surtout quand tu travailles avec des données qui nécessitent une référence constante à certaines cellules. En les maîtrisant, tu pourras rendre tes feuilles de calcul beaucoup plus flexibles et fiables. Alors, pratiquez bien et n'hésitez pas à poser des questions!

Excel - Les références absolues - Exercices supplémentaires

> *Aujourd'hui, on va plonger dans le monde des références absolues. Vous allez apprendre ce qu'elles sont, pourquoi elles sont importantes et comment les utiliser avec des exemples concrets.*

Une **référence relative** c'est une référence qui va varier quand vous allez la recopier. On dit "Relative" car les références de la cellule ne sont pas fixées (A1, B1, ...).

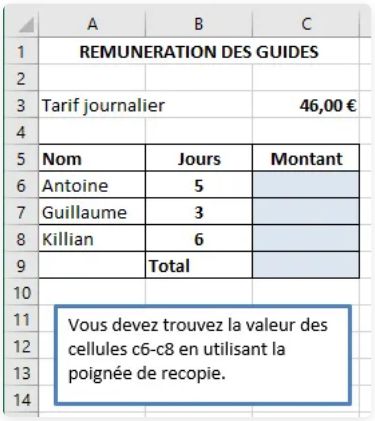
Une **référence absolue** c'est une référence qui ne va pas être modifiée lors d'une phase de recopie. Autrement dit, elle est (partiellement) **constante**. On parle aussi d'une **référence fixe**.

Une cellule est fixée quand vous mettez des `$` de part et d'autre de la référence : `=$F$2`

Exercice 1

Téléchargez le fichier `ex-001.xls` dans la section "téléchargements".

Utiliser obligatoirement la poignée de recopie pour calculer les cellules de C6 à C8



	A	B	C
1	**REMUNERATION DES GUIDES**		
2			
3	Tarif journalier		**46,00 €**
4			
5	**Nom**	**Jours**	**Montant**
6	Antoine	**5**	
7	Guillaume	**3**	
8	Killian	**6**	
9		**Total**	
10			
11	Vous devez trouvez la valeur des		
12	cellules c6-c8 en utilisant la		
13	poignée de recopie.		
14			

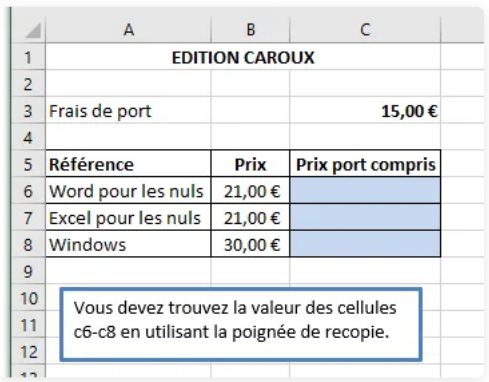
Résultat attendu:

	A	B	C
1	**REMUNERATION DES GUIDES**		
2			
3	Tarif journalier		46,00 €
4			
5	**Nom**	**Jours**	**Montant**
6	Antoine	5	230,00 €
7	Guillaume	3	138,00 €
8	Killia,n	6	276,00 €
9		**Total**	**644,00 €**
10			

Exercice 2

Téléchargez le fichier `ex-002.xls` dans la section "téléchargements".

En C6 Saisir la formule pour calculer le prix port compris du premier livre. Utiliser la poignée de recopie pour dupliquer cette formule sur les lignes suivantes (C7 et C8).



	A	B	C
1	**EDITION CAROUX**		
2			
3	Frais de port		**15,00 €**
4			
5	**Référence**	**Prix**	**Prix port compris**
6	Word pour les nuls	21,00 €	
7	Excel pour les nuls	21,00 €	
8	Windows	30,00 €	
9			
10	Vous devez trouvez la valeur des cellules		
11	c6-c8 en utilisant la poignée de recopie.		
12			

Résultat attendu:

	A	B	C
1	EDITION CAROUX		
2			
3	Frais de port		15,00 €
4			
5	Référence	Prix	Prix port compris
6	Word pour les nuls	21	36,00 €
7	Excel pour les nuls	21	36,00 €
8	Windows	30	45,00 €

Exercice 3

Construire le tableau sur Excel Vous devez trouvez l'opération, à insérer dans les cellules bleues, elle va vous permettre d'utiliser la poignée de recopie sur tout le tableau, colonne par colonne. Attention il y a plusieurs solutions pour saisir sa formule (indice : A1 ou A$1).

	A	B	C	D	E	F
2	TABLEAU DES SOLDES					
3		10%	20%	30%	50%	70%
4	5,00 €					
5	10,00 €					
6	15,00 €					
7	20,00 €					
8	25,00 €					
9	30,00 €					
10	35,00 €					
11	40,00 €					
12	45,00 €					
13	50,00 €					
14	55,00 €					
15	60,00 €					
16	65,00 €					
17	70,00 €					
18	75,00 €					
19	80,00 €					
20	85,00 €					
21	90,00 €					
22	95,00 €					
23	100,00 €					

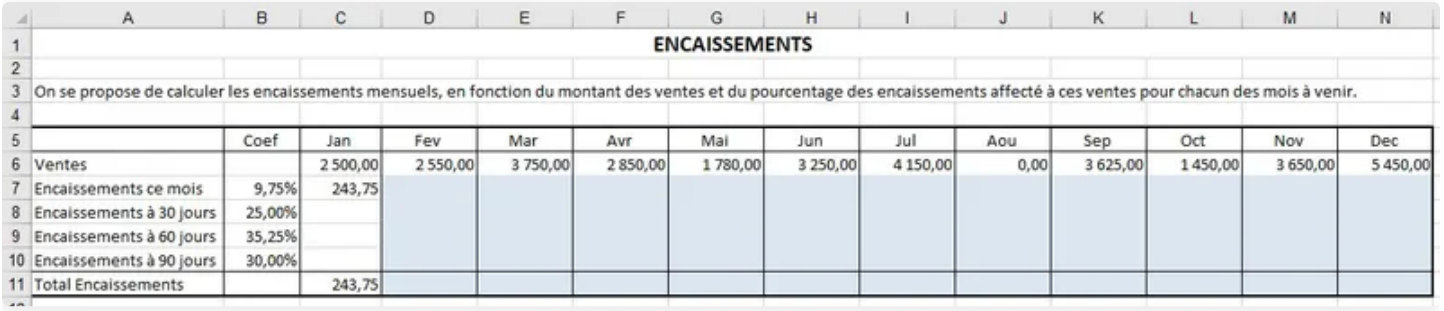
Solution:

	A	B	C	D	E	F
1	Vive les soldes					
2						
3		10%	20%	30%	50%	70%
4	5,00 €	4,50 €	4,00 €	3,50 €	2,50 €	1,50 €
5	10,00 €	9,00 €	8,00 €	7,00 €	5,00 €	3,00 €
6	15,00 €	13,50 €	12,00 €	10,50 €	7,50 €	4,50 €
7	20,00 €	18,00 €	16,00 €	14,00 €	10,00 €	6,00 €
8	25,00 €	22,50 €	20,00 €	17,50 €	12,50 €	7,50 €
9	30,00 €	27,00 €	24,00 €	21,00 €	15,00 €	9,00 €
10	35,00 €	31,50 €	28,00 €	24,50 €	17,50 €	10,50 €
11	40,00 €	36,00 €	32,00 €	28,00 €	20,00 €	12,00 €
12	45,00 €	40,50 €	36,00 €	31,50 €	22,50 €	13,50 €
13	50,00 €	45,00 €	40,00 €	35,00 €	25,00 €	15,00 €
14	55,00 €	49,50 €	44,00 €	38,50 €	27,50 €	16,50 €
15	60,00 €	54,00 €	48,00 €	42,00 €	30,00 €	18,00 €
16	65,00 €	58,50 €	52,00 €	45,50 €	32,50 €	19,50 €
17	70,00 €	63,00 €	56,00 €	49,00 €	35,00 €	21,00 €
18	75,00 €	67,50 €	60,00 €	52,50 €	37,50 €	22,50 €
19	80,00 €	72,00 €	64,00 €	56,00 €	40,00 €	24,00 €
20	85,00 €	76,50 €	68,00 €	59,50 €	42,50 €	25,50 €
21	90,00 €	81,00 €	72,00 €	63,00 €	45,00 €	27,00 €
22	95,00 €	85,50 €	76,00 €	66,50 €	47,50 €	28,50 €
23	100,00 €	90,00 €	80,00 €	70,00 €	50,00 €	30,00 €

Exercice 4

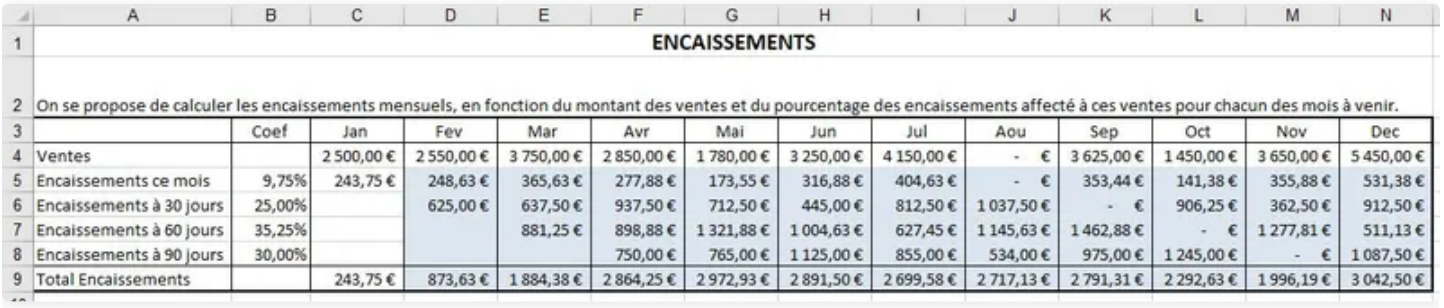
Téléchargez le fichier Excel dans la section "téléchargements".

On se propose de calculer les encaissements mensuels, en fonction du montant des ventes et du pourcentage des encaissements affecté à ces ventes pour chacun des mois à venir.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ENCAISSEMENTS													
2														
3	On se propose de calculer les encaissements mensuels, en fonction du montant des ventes et du pourcentage des encaissements affecté à ces ventes pour chacun des mois à venir.													
4														
5		Coef	Jan	Fev	Mar	Avr	Mai	Jun	Jul	Aou	Sep	Oct	Nov	Dec
6	Ventes		2 500,00	2 550,00	3 750,00	2 850,00	1 780,00	3 250,00	4 150,00	0,00	3 625,00	1 450,00	3 650,00	5 450,00
7	Encaissements ce mois	9,75%	243,75											
8	Encaissements à 30 jours	25,00%												
9	Encaissements à 60 jours	35,25%												
10	Encaissements à 90 jours	30,00%												
11	Total Encaissements		243,75											

Résultat attendu:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ENCAISSEMENTS													
2	On se propose de calculer les encaissements mensuels, en fonction du montant des ventes et du pourcentage des encaissements affecté à ces ventes pour chacun des mois à venir.													
3		Coef	Jan	Fev	Mar	Avr	Mai	Jun	Jul	Aou	Sep	Oct	Nov	Dec
4	Ventes		2 500,00 €	2 550,00 €	3 750,00 €	2 850,00 €	1 780,00 €	3 250,00 €	4 150,00 €	- €	3 625,00 €	1 450,00 €	3 650,00 €	5 450,00 €
5	Encaissements ce mois	9,75%	243,75 €	248,63 €	365,63 €	277,88 €	173,55 €	316,88 €	404,63 €	- €	353,44 €	141,38 €	355,88 €	531,38 €
6	Encaissements à 30 jours	25,00%		625,00 €	637,50 €	937,50 €	712,50 €	445,00 €	812,50 €	1 037,50 €	- €	906,25 €	362,50 €	912,50 €
7	Encaissements à 60 jours	35,25%			881,25 €	898,88 €	1 321,88 €	1 004,63 €	627,45 €	1 145,63 €	1 462,88 €	- €	1 277,81 €	511,13 €
8	Encaissements à 90 jours	30,00%				750,00 €	765,00 €	1 125,00 €	855,00 €	534,00 €	975,00 €	1 245,00 €	- €	1 087,50 €
9	Total Encaissements		243,75 €	873,63 €	1 884,38 €	2 864,25 €	2 972,93 €	2 891,50 €	2 699,58 €	2 717,13 €	2 791,31 €	2 292,63 €	1 996,19 €	3 042,50 €

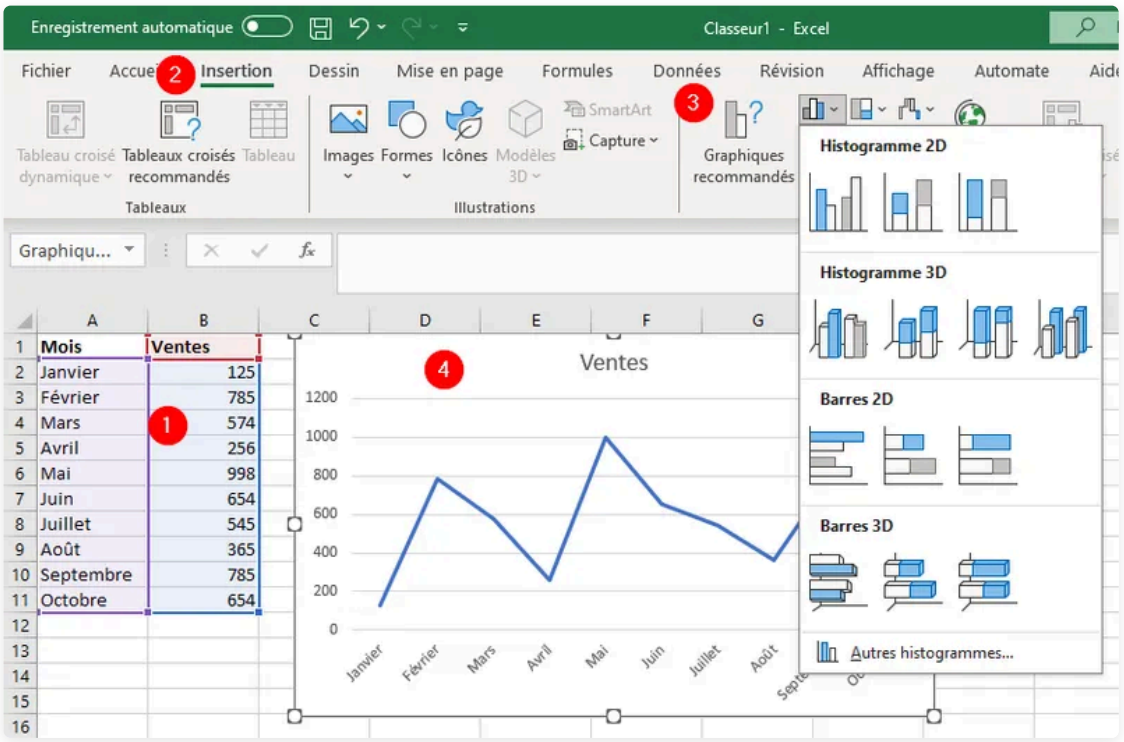
Excel: Les principaux types de graphiques

Excel est un outil très utile pour traiter et visualiser des données. Les graphiques, en particulier, nous aident à présenter ces données de manière claire et engageante. Nous allons découvrir ensemble comment choisir le bon type de graphique pour vos données et comment chaque graphique peut raconter une histoire différente.

Comment insérer un graphique?

1. Sélectionnez vos données: cela peut inclure les cellules contenant les valeurs et, si nécessaire, les cellules avec les étiquettes des axes (titres des colonnes).
2. Insérer le Graphique:allez dans l'onglet '**Insertion**' dans le ruban en haut de la fenêtre Excel.
3. Choisissez le **type de graphique** que vous voulez créer. Excel offre une variété d'options, comme les graphiques en colonnes, en barres, en lignes, en secteurs (camemberts), et plus encore. Sélectionnez celui qui convient le mieux à vos données et à ce que vous voulez communiquer. Vpous pouvez aussi choisir parmi les "**graphiques recommandés**".

Cliquez sur l'icône du graphique choisi. Excel insérera automatiquement un graphique basé sur vos données sélectionnées.



Graphique en Colonnes (Histogrammes)

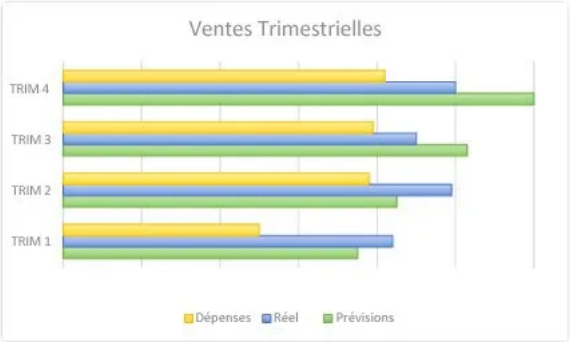
Les **histogrammes** sont des graphiques en **colonnes**. Ils sont parfaits pour **comparer différentes valeurs entre elles**. Imaginez que chaque colonne est un pilier, dont la hauteur varie selon la valeur qu'elle représente. Ce type de graphique est particulièrement utile pour comparer des données sur différentes catégories, comme par exemple les ventes de différents produits sur un mois donné.

Un histogramme affiche généralement les catégories le long de l'axe horizontal (abscisses) et les valeurs le long de l'axe vertical (ordonnées), comme illustré dans ce graphique :



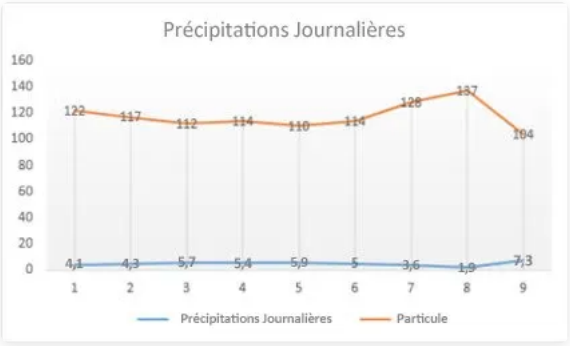
Graphique en Barres (Barres 2d)

Très similaire au graphique en colonnes, la principale différence du graphique en barres est son orientation : les barres sont disposées horizontalement plutôt que verticalement. Ce format est particulièrement adapté lorsque vous avez des étiquettes de catégorie longues ou complexes, ou si vous souhaitez comparer un grand nombre de catégories. Par exemple, ce serait idéal pour afficher le nombre d'étudiants dans différentes classes de votre école.



Graphique en Courbes

Le graphique en Courbes, quant à lui, est un excellent moyen de montrer les **tendances sur une période** donnée. Il relie des points de données successifs avec une ligne, permettant ainsi de visualiser facilement l'évolution d'une valeur dans le temps. Par exemple, vous pourriez l'utiliser pour suivre les variations des précipitations au fil des jours.

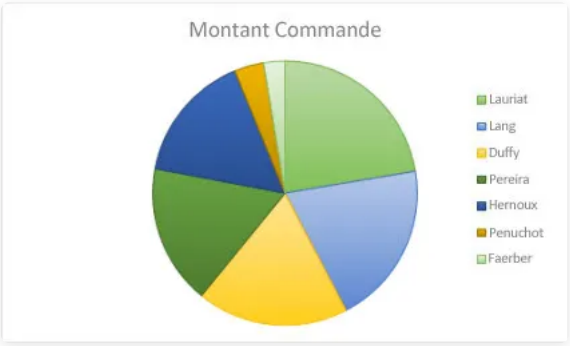


Le temps est représenté sur l'axe horizontal, tandis que les valeurs sont affichées sur l'axe vertical.

Graphique en Secteurs (Camembert)

Le graphique en secteurs, aussi connu sous le nom de **camembert**, est une représentation circulaire des données. Il divise un cercle en segments proportionnels à la contribution de chaque valeur. Autrement dit, les graphiques en secteurs représentent **la taille des éléments** d'une série de données **par rapport à leur somme**. Les points de données dans un graphique en secteurs sont représentés sous forme de **pourcentage** du graphique entier.

Ce type de graphique est particulièrement efficace pour **montrer la répartition des parts dans un ensemble**, comme les différentes dépenses dans un budget familial.

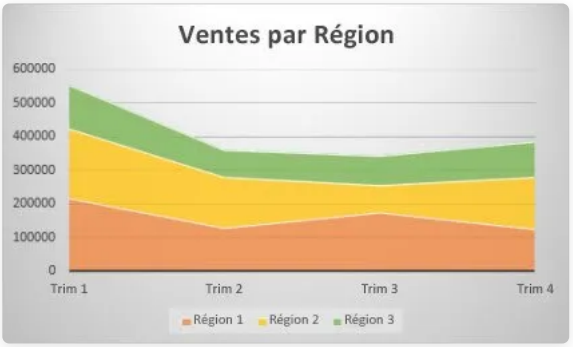


Il est conseillé d'utiliser un graphique en secteurs quand :

- vous avez **une seule série** de données ;
- toutes les valeurs dans vos **données** sont **positives** ;
- presqu'aucune des valeurs dans vos données n'est nulle ;
- vous n'avez pas plus de **7 catégories**, qui représentent toutes des parties du graphique entier.

Graphique en Aire

Le graphique en aire combine des éléments des graphiques en ligne et en colonnes. Comme un graphique en ligne, il montre l'évolution d'une valeur dans le temps, mais avec la zone sous la ligne remplie de couleur. Cela met en évidence le volume cumulé, rendant ce graphique utile pour **comparer plusieurs quantités au fil du temps**, comme les ventes de différentes régions au cours d'une année.



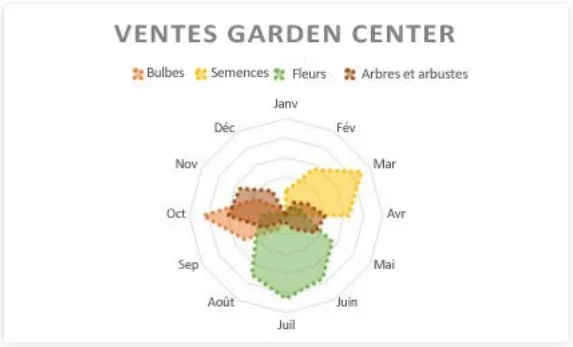
Graphique en Nuage de Points (Dispersion)

Le graphique en nuage de points, ou graphique de **dispersion**, est idéal pour examiner la relation ou la corrélation entre deux séries de valeurs. Chaque point représente une paire de données, ce qui vous permet de voir s'il existe une tendance ou une corrélation. Par exemple, vous pourriez l'utiliser pour analyser si les étudiants qui passent plus de temps sur leurs devoirs obtiennent de meilleures notes.



Graphique en Radar

Enfin, le graphique en radar permet de **comparer plusieurs variables quantitatives**. Il est utile pour visualiser les performances dans différentes catégories ou pour l'analyse de données multidimensionnelles. Imaginez un graphique en radar comme une toile d'araignée, où chaque axe représente une différente catégorie. Par exemple, il pourrait être utilisé pour évaluer les compétences d'un étudiant dans différentes matières.



Conclusion

Le choix du bon graphique est crucial pour présenter vos données de manière précise et instructive. **Chaque type de graphique a ses forces et est adapté à différentes situations**.

Pour s'exercer

Recherche **2 séries de données** qui correspondent au mieux aux spécificité de chacun des types de graphiques évoqués dans cet article (2 séries pour chaque type de graphique).

Naviguer dans un document

Se déplacer efficacement dans un texte ou un document est une compétence très utile, que ce soit pour rédiger un document, programmer ou naviguer dans des données. Connaître les raccourcis clavier peut grandement améliorer votre productivité. Cet article couvre tous les raccourcis clavier essentiels pour se déplacer dans un texte ou un document sous Windows.

Déplacements de Base



Flèches directionnelles

Les touches fléchées (←, →, ↑, ↓) sont les plus simples pour naviguer :

- **Flèche gauche (←)** : Se déplace d'un **caractère** à gauche.
- **Flèche droite (→)** : Se déplace d'un **caractère** à droite.
- **Flèche haut (↑)** : Se déplace d'une **ligne** vers le haut.
- **Flèche bas (↓)** : Se déplace d'une **ligne** vers le bas.

Déplacements Rapides

Avec la touche `Ctrl`

- **Ctrl + Flèche gauche** : Se déplace d'un **mot** à gauche.
- **Ctrl + Flèche droite** : Se déplace d'un m**ot à droite.
- **Ctrl + Flèche haut** : Se déplace d'un **paragraphe** vers le haut.
- **Ctrl + Flèche bas** : Se déplace d'un **paragraphe** vers le bas.

Début et Fin de Ligne

- **Home** : Se déplace au **début de la ligne** courante.
- **End** : Se déplace à la **fin de la ligne** courante.

Début et Fin de Document

- **Ctrl + Home** : Se déplace au **début du document**.
- **Ctrl + End** : Se déplace à la **fin du document**.

Sélection de Texte

Sélection par Caractère

- **Maj + Flèche gauche** : Sélectionne le **caractère** à gauche.
- **Maj + Flèche droite** : Sélectionne le **caractère** à droite.

Sélection par Mot

- **Ctrl + Maj + Flèche gauche** : Sélectionne le **mot** à gauche.
- **Ctrl + Maj + Flèche droite** : Sélectionne le **mot** à droite.

Sélection par Ligne

- **Maj + Flèche haut** : Sélectionne la **ligne** au-dessus.
- **Maj + Flèche bas** : Sélectionne la **ligne** en dessous.

Sélection de Paragraphe

- **Ctrl + Maj + Flèche haut** : Sélectionne jusqu'au **début du paragraphe**.
- **Ctrl + Maj + Flèche bas** : Sélectionne jusqu'à la **fin du paragraphe**.

Début et Fin de Document

- **Ctrl + Maj + Home** : Sélectionne du curseur jusqu'au **début du document**.
- **Ctrl + Maj + End** : Sélectionne du curseur jusqu'à la **fin du document**.

Autres Raccourcis Utiles

Aller à une Ligne Précise

Dans certains éditeurs de texte, comme Notepad++ ou Visual Studio Code :

- **Ctrl + G** ou **Ctrl + L** : Ouvre une boîte de dialogue pour aller à une ligne précise.

Naviguer entre les Documents et les Applications

Changer de Fenêtre

- **Alt + Tab** : Permet de basculer entre les **applications** ouvertes.
- **Ctrl + Tab** : Bascule entre les **documents ouverts dans une même application** (comme les onglets dans un navigateur web ou un IDE).

Conclusion

Maîtriser ces raccourcis clavier peut considérablement **améliorer votre efficacité et votre productivité** lorsque vous travaillez avec des textes ou des documents sous Windows. Pratiquez-les régulièrement pour les intégrer à vos habitudes de travail. N'hésitez pas à imprimer cette liste de raccourcis et à la garder à portée de main jusqu'à ce que vous vous sentiez à l'aise avec eux.

Ces raccourcis vous permettront de gagner du temps et d'être plus précis dans vos déplacements et vos sélections de texte. Bonne pratique et bonne rédaction !

Le presse-papier Windows

> *Le presse-papier de Windows est un outil essentiel pour quiconque utilise un ordinateur, que ce soit pour le travail, les études ou le loisir. Dans cet article, nous allons explorer ce qu'est le presse-papier, à quoi il sert, comment il fonctionne et comment tirer parti de ses fonctionnalités avancées, notamment le presse-papier multiple.*

Qu'est-ce que le Presse-papier Windows ?

Le presse-papier de Windows est **une zone de stockage temporaire en mémoire** où les informations copiées ou coupées sont conservées jusqu'à ce qu'elles soient collées ailleurs. Cela permet de **transférer des données entre différentes applications ou au sein d'un même document**.

À quoi Sert le Presse-papier ?

Le presse-papier est principalement utilisé pour les opérations de **copier-coller** et **couper-coller**, facilitant ainsi la duplication ou le déplacement de textes, d'images, de fichiers, etc. Voici quelques exemples concrets d'utilisation :

- **Copier du texte** depuis une page web pour le coller dans un document Word.
- **Déplacer des fichiers** d'un dossier à un autre sur votre ordinateur.
- **Copier une formule complexe dans Excel** pour l'appliquer à plusieurs cellules.

Fonctionnement du Presse-papier

Le presse-papier peut contenir **un seul élément à la fois**. Chaque nouvelle opération de copier ou couper **remplace le contenu existant** du presse-papier par le nouvel élément.

Copier (`Ctrl + C`)

Lorsque vous copiez un élément, il est placé dans le presse-papier **sans être retiré de son emplacement d'origine**.

Couper (`Ctrl + X`)

Couper un élément le **supprime de son emplacement d'origine** et le place dans le presse-papier.

Coller (`Ctrl + V`)

Coller un élément **insère** le contenu du presse-papier **à l'emplacement du curseur** dans l'application cible.

Exemples Concrets

Copier et Coller du Texte

1. **Sélectionnez le texte souhaité.**
2. **Appuyez sur `Ctrl + C` pour copier.**
3. **Placez le curseur à l'endroit où vous voulez coller le texte.**
4. **Appuyez sur `Ctrl + V` pour coller.**

Déplacer un Fichier

1. **Sélectionnez le fichier dans l'Explorateur de fichiers.**
2. **Appuyez sur `Ctrl + X` pour couper.**
3. **Naviguez vers le dossier de destination.**
4. **Appuyez sur `Ctrl + V` pour coller.**

Le Presse-papier Multiple

Avec les versions récentes de Windows 10 et Windows 11, Microsoft a introduit la fonctionnalité de **presse-papier multiple**, qui permet de **stocker plusieurs éléments** dans le presse-papier et de choisir lequel coller. Voici comment cela fonctionne :

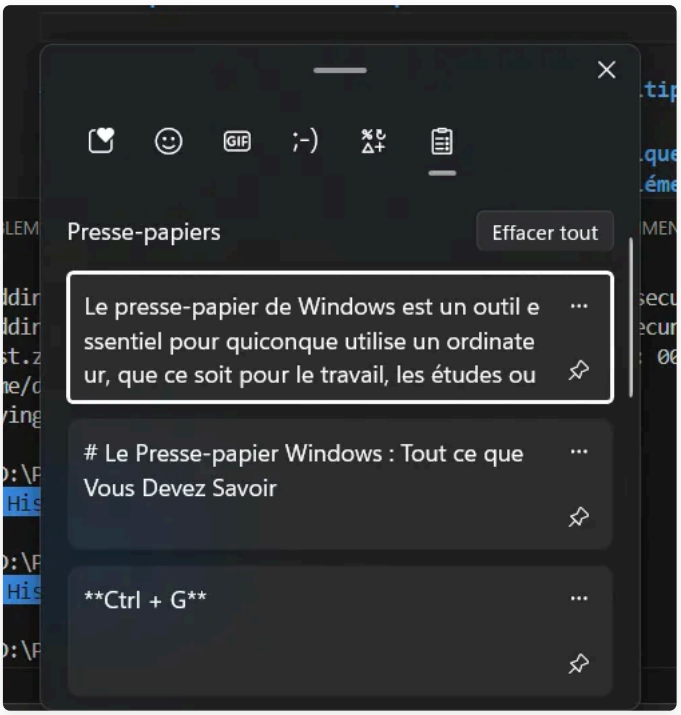
Activer/Désactiver le Presse-papier Multiple

Le plus simple pour activer le presse-papier multiple la première fois, c'est simplement d'utiliser le raccourci `Win + V`. Si cette fonctionnalité n'est pas encore activée, Windows vous proposera de l'activer.

1. **Ouvrez les paramètres Windows (`Win + I`).**
2. **Allez dans Système > Presse-papier.**
3. **(Dés)Activez l'option "Historique du presse-papier".**

Utiliser le Presse-papier Multiple

1. **Copiez ou coupez plusieurs éléments comme d'habitude (`Ctrl + C` ou `Ctrl + X`).**
2. **Appuyez sur `Win + V` pour afficher l'historique du presse-papier.**
3. **Cliquez sur l'élément que vous souhaitez coller ou utilisez les flèches directionnelles pour le sélectionner, puis appuyez sur Entrée.**



Supprimer des Éléments du Presse-papier Multiple

- **Appuyez sur `Win + V` pour ouvrir l'historique du presse-papier.**
- **Cliquez sur les trois points à côté d'un élément et sélectionnez "Supprimer" pour supprimer cet élément spécifique, ou "Effacer tout" pour vider tout le presse-papier.**

Exemples Concrets d'Utilisation

COPIER PLUSIEURS TEXTES ET IMAGES

1. **Copiez du texte depuis un document Word (`Ctrl + C`).**
2. **Copiez une image depuis un site web (`clic droit > Copier l'image`).**
3. **Copiez une formule depuis un fichier Excel (`Ctrl + C`).**
4. **Appuyez sur `Win + V` pour afficher l'historique du presse-papier et coller les éléments dans l'ordre souhaité dans un nouveau document.**

Raccourcis Claviers Utiles pour le Presse-papier

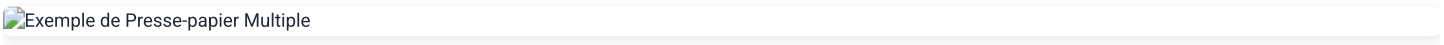
Voici une liste de raccourcis clavier essentiels pour utiliser efficacement le presse-papier sous Windows :

- `Ctrl + C` : Copier.
- `Ctrl + X` : Couper.
- `Ctrl + V` : Coller.
- `Win + V` : Ouvrir l'historique du presse-papier.
- `Ctrl + Z` : Annuler l'action précédente (utile après avoir collé quelque chose par erreur).
- `Ctrl + Y` : Rétablir une action annulée.

Conclusion

Le presse-papier de Windows, avec ses fonctionnalités de base et avancées, est un outil puissant pour améliorer votre productivité. En comprenant comment l'utiliser efficacement et en tirant parti du presse-papier multiple, vous pouvez gérer plus facilement des tâches complexes de copie et de déplacement de données.

Pratiquez ces raccourcis et fonctionnalités pour les intégrer dans votre flux de travail quotidien. N'hésitez pas à explorer davantage les options et les paramètres du presse-papier pour personnaliser son fonctionnement selon vos besoins.



Avec ces outils à votre disposition, vous serez capable de manipuler des données plus rapidement et avec plus de précision. Bonne utilisation du presse-papier !



GESTION DES DONNÉES

Les bases de données

> *Les bases de données sont omniprésentes dans notre quotidien numérique, qu'il s'agisse de sauvegarder vos contacts, de gérer des stocks, ou d'héberger des plateformes de streaming. Mais savez-vous réellement ce qu'est une base de données, pourquoi elle est si essentielle, et comment choisir le bon système pour vos besoins ?*

Cet article vous offre une introduction aux bases de données et pose les bases pour un cours complet qui explorera leurs mécanismes, leur conception et leur utilisation.

💥 Une **base de données** (BD) (GB ***DB, database***) est un **ensemble organisé** de données stockées de manière structurée pour permettre un **accès**, une **gestion** et une **mise à jour efficaces**. Ces données peuvent représenter n'importe quoi : des informations personnelles, des transactions commerciales, des inventaires, etc.

Pourquoi utiliser une base de données ?

L'utilisation d'une base de données offre plusieurs avantages par rapport aux méthodes de stockage manuel ou non structuré (comme les feuilles Excel ou les fichiers texte).

Voici les principales raisons d'utiliser une base de données :

- **Organisation des données**: Les bases de données permettent de **structurer** les informations (par exemple en tables, colonnes et relations), ce qui facilite leur lecture et leur compréhension.
- **Accessibilité**: Les bases de données modernes permettent un **accès multi-utilisateur en temps réel**, souvent via des interfaces web ou des applications.
- **Fiabilité**: Grâce aux systèmes de sauvegarde et aux transactions, les bases de données **assurent l'intégrité des données** même en cas de panne.
- **Performance**: Elles **optimisent le stockage et la recherche** d'informations, même pour de très grands volumes de données.
- **Sécurité**: Les bases de données offrent des mécanismes pour **contrôler l'accès**, chiffrer les informations sensibles et assurer leur protection.

La chaos de données peu structurées

> *Dans cette activité, tu vas manipuler un fichier réel contenant des informations de billetterie. Ton objectif est d'observer les données et de comprendre pourquoi une simple liste ne suffit pas toujours.*

Dans cette activité, tu vas manipuler un fichier réel contenant des informations de billetterie. Ton objectif est d'observer les données et de comprendre pourquoi une simple liste ne suffit pas toujours.

🎯 Objectifs

À la fin de cette activité, tu seras capable de :

- repérer des problèmes dans un tableau de données
- identifier des répétitions et des incohérences
- comprendre qu'un tableau peut contenir plusieurs types d'informations mélangées
- proposer une organisation plus claire des données

📁 Fichier à utiliser

Ouvre le fichier CSV fourni par ton professeur dans un tableur (Excel, LibreOffice, Google Sheets…).

🔍 Partie 1 — Observation des données

Observe attentivement le contenu du fichier.

Réponds aux questions suivantes :

- Combien de fois le client **Marie Dupont** apparaît-il dans le fichier ?
- Les informations de ce client sont-elles toujours identiques (nom, téléphone…) ?
- Combien de fois l'événement **Soirée Quiz Pop Culture** apparaît-il ?
- Les noms des événements sont-ils toujours écrits de la même manière ?
- Les dates sont-elles toutes écrites avec le même format ?
- Les prix sont-ils toujours stockés de la même façon ?
- Trouves-tu des informations manquantes ? Donne un exemple.
- Ce fichier te semble-t-il fiable et facile à utiliser ? Explique pourquoi.

🛠️ Partie 2 — Manipulation

Effectue les actions suivantes directement dans le fichier ou en expliquant comment tu ferais :

- Retrouver tous les tickets du client **Lucas Martin**
- Retrouver tous les participants du **Quiz Cinéma**
- Retrouver tous les tickets pour des évènements qui se passent au Forum de Liège
- Modifier le numéro de téléphone de **Marie Dupont**
- Changer le lieu du **Blind Test Années 90**
- Supprimer un ticket annulé
- Compter le nombre de tickets vendus pour chaque événement

🤔 Partie 3 — Analyse

Réponds aux questions suivantes :

- Pourquoi certaines informations apparaissent-elles plusieurs fois ?
- Que se passe-t-il si tu modifies une information à un seul endroit ?
- Quels problèmes peuvent apparaître si plusieurs personnes modifient ce fichier ?
- Est-ce que toutes les informations présentes dans une ligne concernent la même chose ?

🧠 Partie 4 — Comprendre les données

Dans le tableau, on retrouve plusieurs types d'informations.

Classe les colonnes dans les trois catégories suivantes :

▸ Informations sur le client

→ .....

▸ Informations sur le ticket

→ .....

▸ Informations sur l'événement

→ .....

▸ Informations sur la salle / le lieux

→ .....

🖍️ Partie 5 — Proposition d'amélioration

Propose une meilleure organisation des données.

Tu dois :

- regrouper les informations qui vont ensemble
- donner un nom à chaque groupe
- expliquer pourquoi cette organisation est plus claire

Tu peux répondre sous forme de schéma simple ou de liste.

💡 À retenir

Un tableau unique peut devenir difficile à gérer lorsque :

- les mêmes informations sont répétées plusieurs fois
- les données ne sont pas toujours cohérentes
- plusieurs types d'informations sont mélangés

Dans ce cas, il est nécessaire de réfléchir à une meilleure structure des données.

Identifier et structurer les données

> *Dans cette activité, tu vas manipuler un fichier réel contenant des informations de billetterie. Ton objectif est d'observer les données et de comprendre pourquoi une simple liste ne suffit pas toujours.*

Identifier et structurer les données

On sait maintenant que tout mettre dans un seul tableau pose des problèmes. L'objectif de cette séquence est d'apprendre à **repérer les différents objets** d'un système et à **organiser les informations** qui leur appartiennent.

Objectifs

À la fin de cette séquence, tu seras capable de :

1. Identifier les objets principaux d'un système
2. Regrouper correctement les informations qui appartiennent à chaque objet
3. Comprendre quels objets sont liés entre eux
4. Représenter la structure sous forme d'un schéma simple

Partie 1 — Repérer les objets

Situation

On reprend les données de la billetterie. Voici trois lignes du tableau Excel :

TICKET_ID	NOM_CLIENT	PRENOM_CLIENT	EMAIL_CLIENT	TELEPHONE_CLIENT	EVENEMENT	DATE_EVENEMENT	LIEU	VILLE	PRIX_TICKET	STATUT_TICKET	CODE_TICKET
T001	Dupont	Marie	marie.dupont@email.be	0471 12 34 56	Soirée Quiz Pop Culture	2026-10-03	Salle des Fêtes	Namur	12	payé	QZ-1001
T002	Dupont	Marie	marie.dupont@email.be	0471 12 34 56	Soirée Quiz Pop Culture	2026-10-03	Salle des Fêtes	Namur	12	payé	QZ-1002

Tu remarques que dans chaque ligne, les informations parlent de **choses très différentes** : la personne qui achète, l'événement concerné, le ticket lui-même et le lieu.

Il peut y avoir plusieurs façons d'analyser des données. Dans notre cas présent, un évènement se passe toujours dans le même lieu (salle et ville). On pourrait, par simplicité, considérer que `lieu` et `ville` sont des propriétés (attributs) d'un évènement ou que le lieu est une *chose* différente en elle-même.

Exercice 1

Réponds aux questions suivantes :

1. Quelles colonnes concernent la **personne** qui achète ?
2. Quelles colonnes concernent l'**événement** ?
3. Quelles colonnes concernent le **ticket** lui-même ?
4. Quelles colonnes concernent le **lieu** lui-même ?

> 💡 *Tu viens d'identifier 4 **objets** du système : un Client, un Événement, un Ticket, un Lieu.*

> *Chaque objet regroupe des informations qui lui appartiennent logiquement.*

Exercice 2

Complète le tableau ci-dessous. Pour chaque objet, note son nom et les informations qui lui appartiennent :

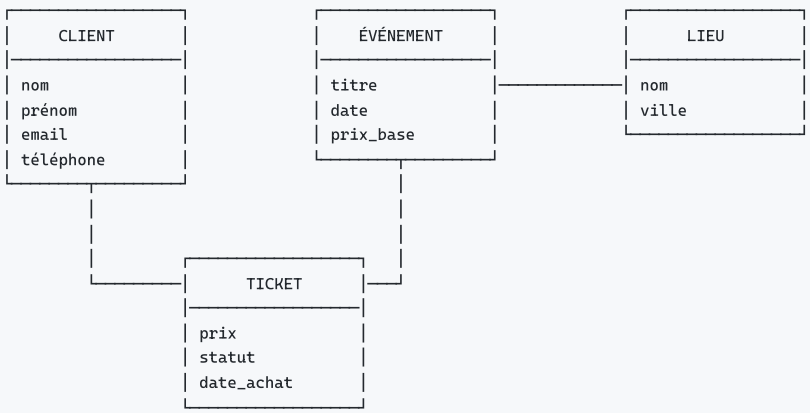
OBJET	NOM	INFORMATIONS
Objet 1	.....	.....
Objet 2	.....	.....
Objet 3	.....	.....
Objet 4	.....	.....

Partie 2 — Dessiner les objets et leurs liens

Maintenant qu'on a identifié les trois objets, on va les **représenter sous forme de boîtes**. Chaque objet est une boîte rectangulaire. On relie les boîtes qui sont **directement liées** entre elles.

Une boite (entité) peut être reliées à plusieurs autres boites.

Le schéma



Lire le schéma

- Chaque **rectangle** représente un objet (que l'on appelle une **entité**).
- Les **traits** montrent quels objets sont directement liés (c'est une **relation**).
- Le **Ticket** est relié à la fois au Client et à l'Événement : c'est lui qui fait le lien entre les deux.
- Il n'y a **pas de lien direct** entre Client et Événement — ils ne se connaissent qu'à travers le Ticket.

Exercice 3 — Analyser les liens

Réponds aux questions suivantes en observant le schéma :

1. Un client peut-il avoir plusieurs tickets ?
2. Un ticket peut-il appartenir à plusieurs clients ?
3. Un événement peut-il avoir plusieurs tickets ?
4. Un ticket peut-il concerner plusieurs événements ?

> 💡 *Ces questions serviront de base dans la prochaine séquence pour comprendre comment les tables sont liées.*

Partie 3 — Le vocabulaire

Ce que tu viens de faire a un nom précis en bases de données. Voici les trois termes à retenir.

Entité

Une **entité** est un objet du système qu'on veut représenter.

Exemples : `Client` , `Ticket` , `Événement` , `Lieu`

Attribut

Un **attribut** est une information qui appartient à une entité.

Exemples :

- `Client` → nom, prénom, email, téléphone
- `Ticket` → prix, statut, date_achat
- `Événement` → titre, date, lieu, prix_base

Relation

Une **relation** décrit le lien entre deux entités.

Exemples :

- Un client est lié à un ticket
- Un événement est lié à un ticket

> ℹ️ *Pour l'instant, il suffit de comprendre ce que ces termes désignent.*

On verra dans la séquence suivante comment les relations s'écrivent techniquement dans une base de données.

À retenir

Pour organiser des données :

1. On identifie les **objets importants** (les entités)
2. On regroupe les **informations** qui appartiennent à chaque entité
3. On sépare ce qui est **différent** — une information n'appartient qu'à une seule entité
4. On **relie** les entités qui sont directement liées par un trait

> *Cette étape est indispensable avant de créer une base de données.*

Suite

Dans la prochaine séquence, tu apprendras à analyser et mieux spécifier les relations. Mais, d'abord, quelques exercices!

Le chaos des données - Exercices 💪

> *Dans cet exercice, tu vas analyser des données. Toutes les informations sont à chaque fois stockées dans un seul tableau, ce qui peut poser des problèmes d'organisation et de cohérence. Ton objectif est de comprendre comment les données sont structurées et d'identifier les différents types d'objets présents dans ce système.*

> 👉 ***données brutes → analyse → identification des entités***

📜 Exercice 1 — Garage (location de voitures)

Contexte

Un garage propose un service de location de voitures. Actuellement, toutes les informations sont stockées dans un seul tableau.

Chaque ligne représente une location effectuée par un client.

📊 Données

ID_LOCATION	NOM_CLIENT	PRENOM_CLIENT	VOITURE	MARQUE	DATE_LOCATION	DATE_RETOUR	PRIX_JOUR	STATUT	TELEPHONE
L001	Dupont	Marie	Clio	Renault	2026-05-01	2026-05-03	45	terminée	0471 12 34 56
L002	Martin	Lucas	208	Peugeot	2026-05-02	2026-05-05	50	en cours	0485 22 11 00
L003	Dupont	Marie	Clio	Renault	2026-05-10	2026-05-12	45	réservée	0471 12 34 56
L004	Lambert	Sophie	Golf	VW	2026-05-03	2026-05-06	60	terminée	0492 44 55 66
L005	Martin	Lucas	208	Peugeot	2026-05-12	2026-05-14	50	réservée	0485 22 11 00
L006	Dubois	Nora	Clio	Renault	2026-05-04	2026-05-06	45	terminée	0478 88 77 66
L007	Peeters	Hugo	Golf	VW	2026-05-07	2026-05-09	60	annulée	0499 10 20 30
L008	Dupont	Marie	208	Peugeot	2026-05-15	2026-05-18	50	en cours	0471 12 34 56
L009	Lambert	Sophie	Clio	Renault	2026-05-20	2026-05-22	45	réservée	0492 44 55 66
L010	Dubois	Nora	Golf	VW	2026-05-21	2026-05-25	60	en cours	0478 88 77 66

🎯 Consigne

👉 Analyse les données et détermine quels sont les différents groupes d'objets (**entités**) ainsi que leurs propriétés (**attributs**):

📜 Exercice 2 — Centre de formation

Contexte

Un centre de formation propose des cours donnés par des instructeurs. Les élèves peuvent s'inscrire à différentes formations.

Toutes les informations sont stockées dans un seul tableau.

📊 Données

🎯 Consigne

👉 Analyse les données et détermine quels sont les différents groupes d'objets (**entités**) ainsi que leurs propriétés (**attributs**):

📜 Exercice 3 — Serveur Discord

Contexte

Un serveur Discord contient plusieurs salons. Les utilisateurs peuvent envoyer des messages dans ces salons.

Toutes les données sont stockées dans un seul tableau.

📊 Données

🎯 Consigne

👉 Analyse les données et détermine quels sont les différents groupes d'objets (**entités**) ainsi que leurs propriétés (**attributs**):

🤓 Le serveur est-il une entité à part entière ?

ID_INSCRIPTION	NOM_ELEVE	PRENOM_ELEVE	FORMATION	DUREE	INSTRUCTEUR	DATE_DEBUT	PRIX	SALLE	STATUT
I001	Dupont	Marie	Python	5 jours	Durand	2026-06-01	300	A101	confirmée
I002	Martin	Lucas	Web	3 jours	Lefevre	2026-06-02	200	B202	confirmée
I003	Lambert	Sophie	Python	5 jours	Durand	2026-06-01	300	A101	annulée
I004	Dubois	Nora	Réseau	4 jours	Martin	2026-06-03	250	C303	confirmée
I005	Dupont	Marie	Web	3 jours	Lefevre	2026-06-02	200	B202	confirmée
I006	Peeters	Hugo	Réseau	4 jours	Martin	2026-06-03	250	C303	confirmée
I007	Martin	Lucas	Python	5 jours	Durand	2026-06-01	300	A101	confirmée
I008	Lambert	Sophie	Web	3 jours	Lefevre	2026-06-02	200	B202	confirmée
I009	Dubois	Nora	Python	5 jours	Durand	2026-06-01	300	A101	confirmée
I010	Peeters	Hugo	Web	3 jours	Lefevre	2026-06-02	200	B202	annulée

ID_MESSAGE	UTILISATEUR	EMAIL	SALON	DATE_MESSAGE	CONTENU	ROLE	SERVEUR	STATUT
M001	Marie	marie@mail.be	général	2026-04-01	Bonjour !	membre	DevZone	envoyé
M002	Lucas	lucas@mail.be	gaming	2026-04-01	Qui joue ce soir ?	membre	DevZone	envoyé
M003	Sophie	sophie@mail.be	général	2026-04-02	Salut !	admin	DevZone	envoyé
M004	Nora	nora@mail.be	musique	2026-04-02	Nouvelle playlist	membre	DevZone	envoyé
M005	Hugo	hugo@mail.be	gaming	2026-04-03	Let's go	membre	DevZone	supprimé
M006	Marie	marie@mail.be	musique	2026-04-03	Trop bien	membre	DevZone	envoyé
M007	Lucas	lucas@mail.be	général	2026-04-04	Update serveur	admin	DevZone	envoyé
M008	Sophie	sophie@mail.be	gaming	2026-04-04	GG	admin	DevZone	envoyé
M009	Nora	nora@mail.be	général	2026-04-05	Hello	membre	DevZone	envoyé
M010	Hugo	hugo@mail.be	musique	2026-04-05	🔥🔥🔥	membre	DevZone	envoyé

💡 Objectif pédagogique global

Ces exercices doivent amener les élèves à comprendre que :

👉 un tableau mélange plusieurs “types d’objets” 👉 chaque objet doit être séparé 👉 c’est la base de la modélisation

Identifier ses données

> *Clé primaire, identifiant naturel, artificiel, AUTO_INCREMENT, UUID : comprendre pourquoi chaque ligne d'une table doit être unique et comment y parvenir.*

Les identifiants — donner un nom unique à chaque enregistrement

Dans le cours précédent, on a vu comment les relations permettent de lier des tables entre elles. Mais pour créer un lien entre deux tables, il faut d'abord pouvoir **désigner** un enregistrement précis sans ambiguïté.

C'est le rôle de l'**identifiant** — ou **clé primaire**.

🔍 Le problème : deux lignes identiques

Imagine une table `eleves` :

NOM	PRENOM	CLASSE
Dupont	Lucas	4TTR
Dupont	Lucas	4TTR
Martin	Emma	4TTR

Il y a deux Lucas Dupont dans la même classe. Comment savoir de lequel on parle quand on veut lui attribuer une note ? Quand on veut l'inscrire à un cours ? Quand on veut le supprimer ?

On ne peut pas. Et c'est exactement le genre de chaos qu'une base de données bien construite doit éviter.

> 💡 ***Règle fondamentale :*** *dans une table relationnelle, chaque ligne doit être identifiable de façon **unique et permanente**.*

🏷️ Les deux types d'identifiants

Il existe deux grandes familles d'identifiants.

---

1. L'identifiant naturel

Un **identifiant naturel** est une donnée qui existe *déjà dans la réalité* et qui est naturellement unique.

Exemples :

CONTEXTE	IDENTIFIANT NATUREL
Élève	Numéro de registre national
Voiture	Numéro de châssis (VIN)
Livre	ISBN
Pays	Code ISO (BE, FR, US…)
Produit	Code-barres EAN-13

Avantage : il a une signification dans le monde réel. On peut lire `BE` et comprendre immédiatement que c'est la Belgique.

Inconvénients :

- On dépend d'une source extérieure pour garantir l'unicité (que se passe-t-il si l'ISBN est mal encodé ?)
- Il peut changer (un numéro de téléphone, un matricule interne…)
- Il n'existe pas toujours (comment identifier un film maison sans système externe ?)
- Il peut être trop long ou trop complexe pour servir de clé dans des jointures

> ⚠️ *En pratique, les identifiants naturels sont risqués. Un email semble unique — mais que se passe-t-il si quelqu'un change d'adresse ? Toutes les références dans les autres tables deviennent fausses.*

---

2. L'identifiant artificiel

Un **identifiant artificiel** est un identifiant qu'on **invente** spécifiquement pour la base de données. Il n'a aucun sens dans la réalité — c'est juste un numéro (ou un code) dont l'unique rôle est de désigner une ligne.

C'est la solution **recommandée** dans la grande majorité des cas.

🔢 AUTO_INCREMENT — l'identifiant entier automatique

La forme la plus courante d'identifiant artificiel en MySQL est un entier auto-incrémenté.

```sql
CREATE TABLE eleves (
    id     INT  NOT NULL AUTO_INCREMENT,
    nom    VARCHAR(100) NOT NULL,
    prenom VARCHAR(100) NOT NULL,
    PRIMARY KEY (id)
);
```

À chaque `INSERT`, MySQL attribue automatiquement le prochain numéro disponible :

```sql
INSERT INTO eleves (nom, prenom) VALUES ('Dupont', 'Lucas');
-- id = 1
INSERT INTO eleves (nom, prenom) VALUES ('Dupont', 'Lucas');
-- id = 2  ← même nom, même prénom, mais identifiant différent !
INSERT INTO eleves (nom, prenom) VALUES ('Martin', 'Emma');
-- id = 3
```

Résultat :

ID	NOM	PRENOM
1	Dupont	Lucas
2	Dupont	Lucas
3	Martin	Emma

Les deux Lucas sont maintenant **distinguables**. On peut parler de "l'élève n°1" et "l'élève n°2" sans ambiguïté.

Ce qu'il faut savoir sur AUTO_INCREMENT

- Le compteur **ne se réinitialise pas** si on supprime des lignes. Si on supprime l'élève n°3, le prochain élève inséré aura l'id `4`, pas `3`.
- C'est **voulu** : un identifiant ne doit jamais être réutilisé. Une fois qu'une ligne a existé avec l'id `3`, cet id ne doit plus jamais désigner autre chose — même si la ligne a été supprimée.
- La colonne doit être déclarée `NOT NULL` et être la `PRIMARY KEY`.

🔑 La clé primaire (`PRIMARY KEY`)

La **clé primaire** est la colonne (ou le groupe de colonnes) qui identifie de façon unique chaque ligne d'une table.

En SQL :

```sql
PRIMARY KEY (id)
```

MySQL refuse automatiquement d'insérer deux lignes avec la même valeur de clé primaire, et refuse les valeurs `NULL`.

Règles d'une bonne clé primaire

1. **Unique** — deux lignes ne peuvent jamais avoir la même valeur
2. **Non nulle** — elle doit toujours avoir une valeur
3. **Stable** — elle ne doit jamais changer après création
4. **Simple** — idéalement un seul champ (pas une combinaison de colonnes)

🌐 UUID — l'identifiant universel

`AUTO_INCREMENT` fonctionne très bien dans une seule base de données. Mais imaginons un scénario différent : on a **deux serveurs** qui génèrent chacun des enregistrements, et on veut les fusionner. Le serveur A a des élèves avec les ids 1, 2, 3 et le serveur B aussi. Impossible de les fusionner sans collision !

C'est là qu'intervient l'**UUID** (*Universally Unique Identifier* — Identifiant Universel Unique).

Un UUID ressemble à ça :

```
550e8400-e29b-41d4-a716-446655440000
```

C'est une chaîne de 32 caractères hexadécimaux (plus des tirets) générée de façon **aléatoire**. La probabilité que deux UUID générés indépendamment soient identiques est astronomiquement faible — on parle d'environ 1 chance sur 5,3 × 10^{36}.

UUID en MySQL

```sql
CREATE TABLE commandes (
    id    CHAR(36)     NOT NULL DEFAULT (UUID()),
    total DECIMAL(8,2) NOT NULL,
    PRIMARY KEY (id)
);
```

Ou en générant l'UUID côté application (PHP, Node.js…) avant l'insertion :

```sql
INSERT INTO commandes (id, total)
VALUES ('550e8400-e29b-41d4-a716-446655440000', 49.99);
```

Avantages et inconvénients

	UUID	AUTO_INCREMENT
Unicité garantie	Partout dans le monde	Dans une seule base
Lisibilité	Illisible (`550e8400…`)	Lisible (`42`)
Performance	Plus lent (index sur 36 chars)	Très rapide (index sur entier)
Cas d'usage	APIs, apps distribuées, microservices	Applications locales, scolaires, simples

💡 *Pour vos projets scolaires et la grande majorité des applications web classiques, `INT AUTO_INCREMENT` est le bon choix. L'UUID devient pertinent quand on expose des ids dans des URLs publiques ou quand on synchronise plusieurs bases.*

🔗 Clé primaire composite — quand utiliser plusieurs colonnes ?

Il est parfois possible de définir une clé primaire sur **deux colonnes combinées**. On parle de **clé primaire composite** (ou clé naturelle composée).

Exemple : une table `inscriptions` qui enregistre qu'un élève est inscrit à un cours. Un élève ne peut pas être inscrit deux fois au même cours :

```sql
CREATE TABLE inscriptions (
    eleve_id INT NOT NULL,
    cours_id INT NOT NULL,
    date_inscription DATE NOT NULL,
    PRIMARY KEY (eleve_id, cours_id)
);
```

La combinaison `(eleve_id, cours_id)` est unique — même si `eleve_id = 1` apparaît plusieurs fois (l'élève 1 est inscrit à plusieurs cours) et `cours_id = 5` apparaît plusieurs fois (plusieurs élèves dans le cours 5).

Quand l'utiliser ?

Les clés composites sont appropriées pour les **tables de liaison** (aussi appelées tables d'association ou tables pivot), qui matérialisent une relation **plusieurs-à-plusieurs**. Dans ce contexte, elles sont naturelles et recommandées.

Pour les tables "normales" (entités), préférer un `id INT AUTO_INCREMENT`.

📌 Tableau récapitulatif

TYPE D'IDENTIFIANT	EXEMPLE	AVANTAGE	INCONVÉNIENT
Naturel simple	ISBN, code pays	Lisible, déjà existant	Peut changer, dépend de l'extérieur
Artificiel entier	`INT AUTO_INCREMENT`	Simple, rapide, automatique	Pas portable entre serveurs
UUID	`550e8400-…`	Unique partout dans le monde	Lourd, peu lisible
Composite	`(eleve_id, cours_id)`	Adapté aux tables de liaison	Complexe à référencer

✍️ À retenir

- Chaque table doit avoir une **clé primaire** — une colonne dont la valeur est unique pour chaque ligne.
- Dans la plupart des cas, on utilise `id INT NOT NULL AUTO_INCREMENT` avec `PRIMARY KEY (id)`.
- Un identifiant **ne doit jamais changer** après sa création — c'est pour ça qu'on n'utilise pas le nom ou l'email comme clé primaire.
- La clé primaire d'une table devient la **clé étrangère** dans les tables qui lui sont liées (vu dans le cours sur les relations).
- L'UUID existe pour les cas où plusieurs systèmes indépendants doivent générer des ids sans se concerter.

📏 Exercice

Voici une table `produits` mal conçue :

```sql
CREATE TABLE produits (
    nom         VARCHAR(100) NOT NULL,
    prix        DECIMAL(6,2) NOT NULL,
    categorie   VARCHAR(50)  NOT NULL
);
```

Questions :

1. Quel problème pose cette table si on insère deux fois le produit `'Stylo bille'` à `0.99 €` ?
2. Modifie la requête `CREATE TABLE` pour ajouter un identifiant artificiel correct.
3. On veut ajouter une table `commandes_produits` pour enregistrer quels produits sont dans quelle commande. Écris la `CREATE TABLE` avec la clé primaire appropriée.

4. Pourquoi serait-ce une mauvaise idée d'utiliser le champ `nom` comme clé primaire ?

Les relations

Dans la séquence précédente, tu as appris à identifier les entités d'un système et à les représenter sous forme de boîtes. Tu as aussi tracé des traits pour indiquer quelles entités sont liées. Mais ces traits ne disaient pas grand-chose : ils montraient qu'un lien existe, sans préciser sa nature ni son étendue. C'est exactement ce que cette séquence va creuser. On va donner un nom à ces liens, les représenter correctement, et surtout répondre à une question essentielle : dans ce lien, combien d'éléments peuvent être concernés de chaque côté ?

Les relations

Objectifs

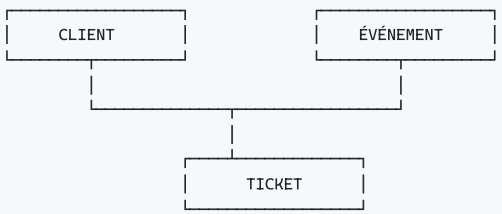
À la fin de cette séquence, tu seras capable de :

1. Définir ce qu'est une relation dans un modèle de données
2. Représenter une relation par une ligne reliant deux entités
3. Lire et écrire des cardinalités
4. Construire un schéma entité-association complet

Partie 1 — Qu'est-ce qu'une relation ?

De l'observation à la formalisation

Dans la séquence 2, on avait ce schéma :



Les traits disaient juste : *ces entités ont un lien.* Mais quel lien, exactement ? Que se passe-t-il concrètement entre un client et un ticket ? Entre un événement et un ticket ?

La réponse, c'est la **relation**.

Une relation, c'est le verbe qui unit deux entités. Elle décrit ce qui se passe entre elles dans la réalité du système :

- Un client **achète** un ticket.
- Un ticket **concerne** un événement.

Ce n'est pas anodin : nommer la relation oblige à comprendre ce que le système fait vraiment.

Partie 2 — La notation

Le schéma entité-association

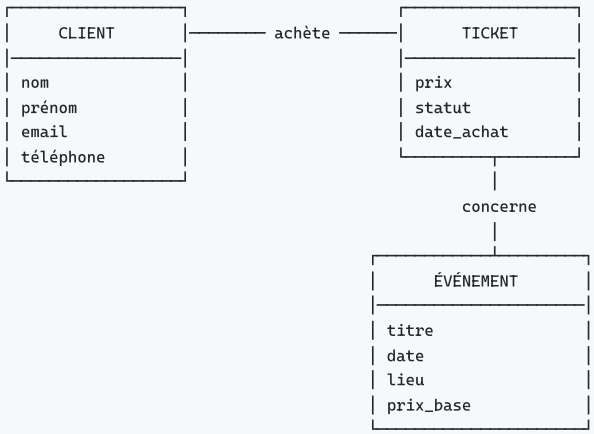
Pour modéliser une base de données, on dessine un **schéma entité-association** : les entités, les liens entre elles, et — on le verra plus loin — les cardinalités. Le schéma qu'on produit s'appelle le **MCD** : **Modèle Conceptuel de Données**.

La représentation est volontairement simple :

ÉLÉMENT	FORME	RÔLE
Entité	Rectangle	Un objet du système
Relation	Une ligne entre deux entités, étiquetée par un verbe	Le lien entre deux entités
Attribut	Texte listé dans le rectangle	Une information d'une entité

Le schéma avec les relations nommées

En appliquant ces conventions à notre système de billetterie, on obtient :



Chaque relation est une simple **ligne** reliant deux entités, portant un **verbe** qui décrit le lien. C'est ce verbe qui donne du sens au trait.

> 💡 *Choisir un bon verbe n'est pas un détail : "achète", "réserve", "possède" ne décrivent pas la même chose. Avant de nommer une relation, demande-toi ce qui se passe réellement dans le système.*

Partie 3 — Les cardinalités

Le problème du "combien"

On sait maintenant qu'un client *achète* un ticket, et qu'un ticket *concerne* un événement. Mais ces liens ne disent encore pas tout. Dans la réalité du système :

- Un client peut-il acheter plusieurs tickets, ou un seul ?
- Un ticket peut-il être acheté par plusieurs clients, ou un seul ?
- Un événement peut-il avoir plusieurs tickets, ou un seul ?

Les réponses à ces questions s'appellent les **cardinalités**. Une cardinalité indique combien d'occurrences d'une entité peuvent être impliquées dans la relation.

La notation

On écrit la cardinalité sous la forme `minimum, maximum` de chaque côté de la ligne.

Les valeurs utilisées sont :

- `0` : aucune occurrence n'est obligatoire
- `1` : exactement une occurrence
- `n` : plusieurs occurrences possibles (sans limite fixée)

Les paires les plus courantes :

NOTATION	SIGNIFICATION
`1,1`	Exactement un — ni plus, ni moins
`0,1`	Zéro ou un — facultatif mais pas multiple
`1,n`	Au moins un, potentiellement plusieurs
`0,n`	Zéro ou plusieurs — aucune contrainte

Comment lire une cardinalité

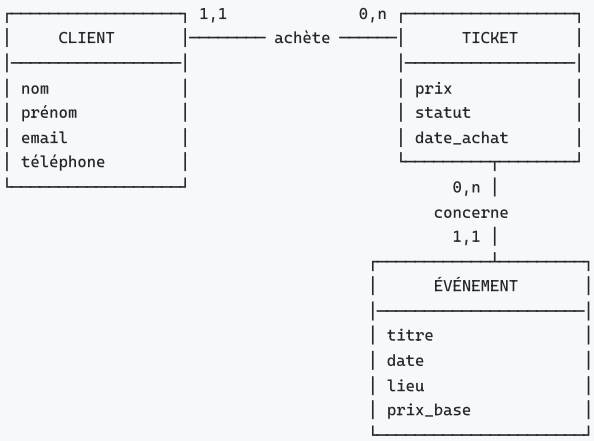
La cardinalité se lit **du côté opposé** : le nombre écrit à côté d'une entité indique combien d'occurrences de **cette** entité correspondent à **une seule** occurrence de l'entité d'en face.

> 🧠 *L'image de la corde*
>
> *Imagine que tu attrapes **une** occurrence d'une entité et que tu tires sur la corde : combien d'occurrences de l'autre entité viennent avec ? Ce nombre s'écrit **du côté de l'autre entité**.*

Exemple avec **CLIENT — achète — TICKET** :

- Côté `CLIENT`, on écrit `1,1` : pour **un** ticket, il y a **exactement un** client.
- Côté `TICKET`, on écrit `0,n` : pour **un** client, il y a **zéro ou plusieurs** tickets.

Le schéma avec les cardinalités



> 💡 *Lis la billetterie à voix haute : « `1,1` côté CLIENT → un ticket appartient à un seul client ; `0,n` côté TICKET → un client peut avoir zéro ou plusieurs tickets. » Les nombres sont toujours du côté de l'entité que l'on compte.*

Exercice 1 — Lire les cardinalités

En observant le schéma ci-dessus, réponds aux questions suivantes :

1. Un client peut-il n'avoir acheté aucun ticket ? Justifie.
2. Un ticket peut-il avoir été acheté par deux clients différents ? Justifie.
3. Un événement peut-il n'avoir aucun ticket associé ? Justifie.
4. Un ticket peut-il concerner deux événements différents ? Justifie.

Partie 4 — Construire un MCD complet

Démarche

Pour construire un MCD, on suit toujours le même ordre :

1. **Identifier les entités** — quels sont les objets du système ?
2. **Lister les attributs** de chaque entité
3. **Repérer les relations** — quels verbes lient les entités ?
4. **Fixer les cardinalités** — en se posant la question "combien ?" pour chaque côté

Exercice 2 — Analyser avant de dessiner

Avant de dessiner quoi que ce soit, réponds d'abord à ces questions pour le système de billetterie :

Entités :

- Quels sont les trois objets principaux ? (tu les connais déjà)

Relations :

- Quel verbe décrit le lien entre `CLIENT` et `TICKET` ?
- Quel verbe décrit le lien entre `TICKET` et `ÉVÉNEMENT` ?

Cardinalités :

Complète le tableau :

RELATION	CÔTÉ GAUCHE	CARDINALITÉ GAUCHE	CARDINALITÉ DROITE	CÔTÉ DROIT
achète	CLIENT	…	…	TICKET
concerne	TICKET	…	…	ÉVÉNEMENT

- Les cardinalités de chaque côté de chaque relation

Exercice 3 — Dessiner le MCD

À partir de tes réponses à l'exercice 2, dessine le MCD complet sur papier.

Tu dois faire apparaître :

- Les trois entités avec leurs attributs
- Les deux relations avec leur verbe écrit sur la ligne

Partie 5 — Application

Reprends le système que tu avais choisi en séquence 2 (garage, centre de formation ou serveur Discord) et construis son MCD complet.

Étapes :

1. Reprends tes trois entités et leurs attributs
2. Nomme les relations qui les lient (un verbe pour chaque)
3. Détermine les cardinalités en te posant la question "combien ?" pour chaque côté
4. Dessine le MCD final

> ⚠️ *Avant de dessiner, remplis toujours le tableau d'analyse (comme dans l'exercice 2). Un MCD dessiné sans réflexion préalable contient presque toujours des erreurs de cardinalité.*

À retenir

Une **relation** est le lien nommé entre deux entités. On la représente par une **ligne** reliant les deux entités, étiquetée par un **verbe**.

Les **cardinalités** expriment le "combien" de chaque côté d'une relation. On les note sous la forme `minimum,maximum`.

PAIRE	SENS
`1,1`	Exactement un
`0,1`	Zéro ou un
`1,n`	Au moins un
`0,n`	Zéro ou plusieurs

La règle pour lire une cardinalité : on la lit **du côté opposé**. Le nombre écrit à côté d'une entité dit, pour **une** occurrence de l'entité d'en face, combien de fois cette entité-ci intervient.

Suite

Dans la prochaine séquence, tu apprendras à transformer ce MCD en **modèle logique** : les entités deviennent des tables, les relations se traduisent en colonnes spéciales appelées **clés étrangères**, et les cardinalités dictent comment ces colonnes s'organisent.

Un SGBDR, c'est quoi ?

> *Avant d'ouvrir un outil et de créer des tables, on pose le vocabulaire : qu'est-ce qu'une base de données relationnelle, qui s'en occupe, et comment on s'y connecte ?*

Jusqu'ici, tu as travaillé sur du papier : entités, relations, cardinalités, identifiants. C'est le **modèle**. La prochaine étape est de transformer ce modèle en une vraie base de données qui tourne sur ton ordinateur.

Avant de toucher à un outil, il faut comprendre **ce qui s'exécute** et **avec quoi on lui parle**.

🗂️ Du schéma aux tables

Dans une **base de données relationnelle**, les informations sont stockées dans des **tables**. Une table, c'est essentiellement un tableau avec :

- des **colonnes** — chacune correspond à un attribut (nom, prix, date…)
- des **lignes** — chacune représente une occurrence (un client précis, un ticket précis…)

La traduction depuis le MCD est directe :

MCD	BASE DE DONNÉES
Entité (`CLIENT` , `TICKET` …)	Table
Attribut (`nom` , `prix` …)	Colonne
Une occurrence	Une ligne
Identifiant	Clé primaire (la colonne `id`)

Le **R** de SGBDR — *relationnel* — vient du fait que ces tables sont **reliées entre elles** par des colonnes spéciales appelées **clés étrangères**. Tu verras la mécanique exacte dans le prochain cours.

> 💡 *Une base de données relationnelle, c'est donc un ensemble de tables qui se référencent mutuellement de façon contrôlée.*

🖥️ Le logiciel qui gère tout ça : le SGBDR

Les tables ne flottent pas dans le vide. Elles sont gérées par un programme : un **SGBDR** — *Système de Gestion de Base de Données Relationnelle* (GB ***RDBMS, Relational Database Management System***).

Son boulot :

- **stocker** les tables sur le disque
- **garantir** que les contraintes sont respectées (pas de doublons sur une clé primaire, pas de clé étrangère qui pointe vers le vide…)
- **exécuter** les requêtes qu'on lui envoie pour lire, ajouter, modifier ou supprimer des données
- **gérer** les accès concurrents (plusieurs personnes ou programmes en même temps)

▸ Les principaux SGBDR

SGBDR	PARTICULARITÉ
MySQL	Le plus répandu sur le web, racheté par Oracle
MariaDB	Fork libre de MySQL, presque 100 % compatible — c'est ce que Laragon installe par défaut
PostgreSQL	Très complet, réputé pour sa rigueur
SQLite	Pas un serveur : un simple fichier `.db` , idéal pour les petites apps
SQL Server	Solution Microsoft, courante en entreprise
Oracle Database	Historique, très présent en grande entreprise

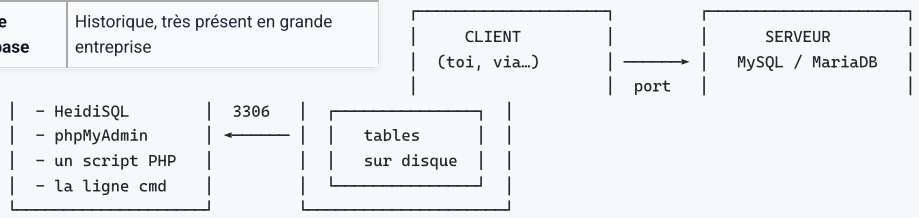
Dans ce cours, on utilise **MySQL / MariaDB** : c'est le SGBDR de loin le plus courant chez les hébergeurs web, et c'est celui que Laragon (ou XAMPP) a installé sur ta machine.

🔌 Le modèle client / serveur

Voici le point qui surprend souvent au début.

Quand tu démarres Laragon, un programme se lance en arrière-plan : c'est le **serveur MySQL** (`mysqld` , le *d* pour *daemon*). Il tourne en permanence, écoute sur un **port réseau** (par défaut le `3306`) et attend qu'on lui parle.

Toi, tu ne lui parles **jamais directement**. Tu utilises un **client** — un autre programme qui se connecte au serveur et lui transmet tes ordres.



Conséquences pratiques :

- Si le service MySQL de Laragon n'est pas démarré, **aucun** client ne pourra se connecter — même si HeidiSQL est ouvert.
- Plusieurs clients peuvent être connectés en même temps au même serveur, sur la même base.
- Le client peut être sur la même machine que le serveur (chez toi pour l'apprentissage) ou à l'autre bout du monde (sur un hébergement web).

🛠️ Les clients qu'on va utiliser

HeidiSQL — le client desktop

Application Windows gratuite, interface graphique riche. On peut créer des bases, des tables, taper des requêtes SQL, voir et éditer les données comme dans un tableur. C'est l'outil de tous les jours sur ta machine.

phpMyAdmin — le client web

Une application web (écrite en PHP) qui tourne sur le serveur web et expose une interface dans le navigateur. Quasiment **tous les hébergements mutualisés** la proposent — c'est souvent le seul moyen d'administrer une base distante sans installer de logiciel.

> 🧠 ***À retenir*** *: HeidiSQL et phpMyAdmin sont* ***deux clients différents qui parlent au même serveur MySQL****. Tout ce que tu crées avec l'un est visible avec l'autre — c'est la même base de données.*

Les autres clients (pour info)

- **La ligne de commande** `mysql` — installée avec Laragon, utile pour les scripts
- **Un programme** — n'importe quel langage (PHP, Python, Node.js, Java…) peut se connecter à MySQL via une bibliothèque dédiée. C'est ce qu'on fera plus tard quand on connectera un site web à sa base.

🗣️ La langue commune : SQL

Quel que soit le client, on parle au serveur dans le même langage : **SQL** (*Structured Query Language*).

Exemples — pas besoin de comprendre la syntaxe maintenant, c'est juste pour voir à quoi ça ressemble :

```sql
-- Créer une table
CREATE TABLE client (
    id      INT NOT NULL AUTO_INCREMENT,
    nom     VARCHAR(100),
    prenom  VARCHAR(100),
    PRIMARY KEY (id)
);

-- Ajouter une ligne
INSERT INTO client (nom, prenom) VALUES ('Dupont', 'Marie');

-- Lire les données
SELECT * FROM client WHERE nom = 'Dupont';
```

Les interfaces graphiques (HeidiSQL, phpMyAdmin) **génèrent du SQL pour toi** quand tu cliques. Tu apprendras à écrire ce SQL directement dans les cours qui suivront — c'est une compétence indispensable dès qu'on quitte les interfaces toutes faites.

🖍️ À retenir

- Une **base de données relationnelle** est un ensemble de **tables** reliées par des **clés**.
- Le **SGBDR** est le logiciel qui stocke ces tables et qui exécute les requêtes — ici MySQL ou MariaDB.
- Le SGBDR fonctionne en mode **client / serveur** : un serveur tourne en permanence, des clients s'y connectent.
- **HeidiSQL** (desktop) et **phpMyAdmin** (web) sont deux clients différents pour le même serveur.
- Tous les clients parlent la même langue : **SQL**.

🧪 Vérifications

1. Sur ta machine, qu'est-ce qui est *serveur* et qu'est-ce qui est *client* ?
2. Si on ferme HeidiSQL, est-ce que la base de données est perdue ? Pourquoi ?
3. Pourquoi un hébergement web propose-t-il phpMyAdmin plutôt que HeidiSQL ?
4. Cite trois SGBDR et donne un cas d'usage typique pour chacun.

Suite

Maintenant que le décor est planté, on peut faire l'étape qui manque entre ton MCD papier et de vraies tables : traduire le schéma entité-association en **modèle relationnel**, avec ses clés primaires et ses clés étrangères.

Du MCD au modèle relationnel

> *Comment transformer un schéma entité-association en un ensemble de tables prêtes à être créées : règles de passage, clés primaires, clés étrangères.*

Tu as un MCD complet : entités, attributs, relations nommées, cardinalités. Tu sais aussi que chaque ligne d'une table doit avoir un identifiant unique. Il manque une étape : **comment relier les tables entre elles** pour matérialiser les relations du MCD.

L'objectif de ce cours est d'apprendre les **règles de traduction** d'un MCD vers un **modèle logique de données** — le **MLD**. À la fin, tu auras une liste de tables avec toutes leurs colonnes, prêtes à être créées dans le SGBDR au prochain cours.

🎯 Objectifs

À la fin de cette séquence, tu seras capable de :

1. Transformer chaque entité en une table avec sa clé primaire
2. Traduire une relation `1,1 / 0,n` en clé étrangère
3. Écrire un MLD complet en notation textuelle
4. Repérer où placer la clé étrangère selon les cardinalités

Partie 1 — La clé étrangère, le maillon manquant

Un exemple pour comprendre

Reprenons deux entités du MCD billetterie :



On a vu dans le cours sur les identifiants que chaque table aura un `id` :

```
client(id, nom, prenom)
ticket(id, prix, statut)
```

Mais avec juste ça, **rien ne relie un ticket à son client**. Si je regarde la ligne `ticket` n°42, je ne sais pas qui l'a acheté.

La solution : une colonne qui pointe

L'idée est simple : on ajoute dans la table `ticket` une colonne qui **contient l'id du client** propriétaire :

ID	PRIX	STATUT	CLIENT_ID
1	12	payé	**3**
2	12	payé	**3**
3	15	réservé	**7**

Tickets n°1 et n°2 appartiennent au client n°3. Ticket n°3 appartient au client n°7. La relation `achète` du MCD est maintenant matérialisée par cette colonne.

Cette colonne porte un nom précis : **clé étrangère** (GB ***foreign key, FK***). Elle est *étrangère* parce qu'elle référence la clé primaire d'une **autre** table.

> 💡 ***Règle d'or*** *: une clé étrangère contient **toujours** une valeur qui existe dans la clé primaire de la table référencée. Le SGBDR refusera d'insérer un ticket avec* `client_id = 999` *si aucun client n°999 n'existe.*

Partie 2 — Les règles de traduction

Règle 1 — Chaque entité devient une table

Pour chaque entité du MCD :

- créer une **table** du même nom (en minuscules, au singulier par convention)
- chaque attribut devient une **colonne**
- ajouter une colonne `id` comme **clé primaire** (`INT AUTO_INCREMENT`)

```
Entité ÉVÉNEMENT (titre, date, prix_base)
       |
       ▼
Table  evenement (id, titre, date, prix_base)
```

▸ Règle 2 — Une relation `1,1 / 0,n` devient une clé étrangère

C'est le cas le plus fréquent et celui qui te servira partout.

Principe : la clé étrangère est placée du côté `0,n` — le côté « plusieurs ».

Pourquoi ? Lis la cardinalité opposée : du côté `1,1`, **un ticket appartient à exactement un client**. Une seule colonne `client_id` suffit donc pour stocker cette référence, et on la place dans la table `ticket`.

```
CLIENT  ———1,1—— achète ——0,n——→  TICKET
                                  (↑ la FK va ici : client_id)
```

À l'inverse, on ne met pas la référence dans `client` : un client peut avoir plusieurs tickets — on ne pourrait pas stocker plusieurs ids dans une seule colonne.

Résultat sur le MCD billetterie :

```
client    (id, nom, prenom, email, telephone)
evenement (id, titre, date, prix_base)
ticket    (id, prix, statut, date_achat, #client_id, #evenement_id)
```

Le `#` signale une clé étrangère. La table `ticket` reçoit **deux** clés étrangères parce qu'elle participe à **deux** relations `1,1 / 0,n` (avec `client` et avec `evenement`).

▸ Règle 3 — Une relation `0,1 / 0,n` se traite pareil

Même logique : la FK va du côté `0,n`. La seule différence, c'est que le côté opposé est `0,1` (et non `1,1`) : comme un événement peut n'avoir **aucun** lieu, la colonne peut être vide (`NULL` autorisé).

```
LIEU   ———0,1—— accueille ——0,n——→  ÉVÉNEMENT
                                    (FK : lieu_id, NULL autorisé)
```

Un événement peut ne pas avoir de lieu encore défini → `lieu_id` peut être `NULL`.

Partie 3 — Notation textuelle du MLD

Sur papier (ou dans un cahier), on écrit un MLD avec les conventions suivantes :

- **Nom de la table** en minuscules, suivi des colonnes entre parenthèses
- La **clé primaire** est **soulignée**
- Les **clés étrangères** sont préfixées d'un `#`
- Une flèche `→` (ou une note) précise vers quelle table pointe la FK

▸ Exemple complet — la billetterie

```
client    (id, nom, prenom, email, telephone)
            —

evenement (id, titre, date, prix_base)
            —

ticket    (id, prix, statut, date_achat, #client_id, #evenement_id)
            —                            └——→ client.id
                                                          └——→ evenement.id
```

Cette écriture est **suffisante** pour créer ensuite les tables dans le SGBDR. Toutes les informations nécessaires sont là.

Partie 4 — Quel type pour chaque colonne ?

Au moment de créer la table, le SGBDR exigera un **type** pour chaque colonne. Voici les plus utiles à ce stade :

> ⚠️ ***Évite*** `FLOAT` ***pour de l'argent.*** `DECIMAL` *garantit la précision exacte ;* `FLOAT` *arrondit subtilement et provoque des erreurs de centimes.*

TYPE	POUR STOCKER	EXEMPLE
`INT`	un entier	un id, un prix en centimes, un nombre
`VARCHAR(n)`	du texte court (max `n` caractères)	un nom, un email, un titre
`TEXT`	du texte long sans limite pratique	une description, un commentaire
`DATE`	une date `AAAA-MM-JJ`	une date d'événement
`DATETIME`	une date + heure	une date d'achat précise
`DECIMAL(p,d)`	un nombre à virgule fixe	un prix : `DECIMAL(6,2)` = jusqu'à 9999,99
`BOOLEAN`	vrai / faux	un statut binaire

▸ **MLD enrichi avec les types**

```
client (
    id          INT  PK  AUTO_INCREMENT
    nom         VARCHAR(100)  NOT NULL
    prenom      VARCHAR(100)  NOT NULL
    email       VARCHAR(150)
    telephone   VARCHAR(20)
)

evenement (
    id          INT  PK  AUTO_INCREMENT
    titre       VARCHAR(200)  NOT NULL
    date        DATETIME      NOT NULL
    prix_base   DECIMAL(6,2)  NOT NULL
)

ticket (
    id            INT  PK  AUTO_INCREMENT
    prix          DECIMAL(6,2)  NOT NULL
    statut        VARCHAR(20)   NOT NULL
    date_achat    DATETIME      NOT NULL
    client_id     INT  FK → client.id        NOT NULL
    evenement_id  INT  FK → evenement.id     NOT NULL
)
```

C'est ce que tu vas littéralement encoder dans HeidiSQL au prochain cours.

Partie 5 — Cas particulier : la relation `0,1 / 0,1`

Plus rare, mais ça arrive : deux entités liées au maximum une seule fois de chaque côté.

Exemple : un `employe` est rattaché à au plus un `casier`, et un `casier` est attribué à au plus un employé.

```
EMPLOYE  ———0,1—— possède ——0,1——→  CASIER
```

Règle : on peut placer la FK **d'un côté ou de l'autre** — peu importe techniquement. On choisit :

- selon ce qui est le plus naturel (l'employé existe d'abord, le casier est attribué après → FK côté casier)
- en ajoutant une contrainte `UNIQUE` sur la FK pour interdire qu'elle se répète (sinon on retomberait sur du `0,n / 0,1`)

```
employe (id, nom, prenom)
casier  (id, numero, #employe_id UNIQUE)
```

💡 *Ce cas est **rare en pratique**. On préfère souvent regrouper les deux entités dans une seule table quand la relation est aussi exclusive. Mais c'est utile à connaître.*

Partie 6 — La méthode, étape par étape

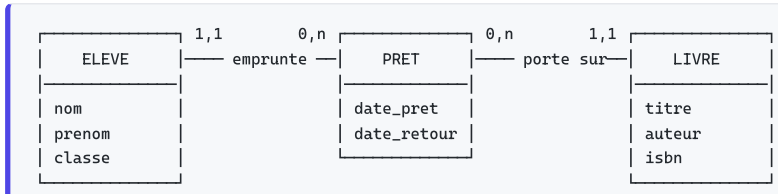
Quand tu reçois un MCD, voici la procédure à suivre :

1. **Une table par entité**, avec un `id` comme clé primaire
2. **Recopier les attributs** en choisissant un type pour chacun
3. **Pour chaque relation** `1,1 / 0,n` : ajouter une FK dans la table du côté `0,n` (le côté « plusieurs »)
4. **Pour chaque relation** `0,1 / 0,n` : même chose — FK dans la table du côté `0,n` — mais autorisée à `NULL` (puisque le côté opposé est `0,1`)
5. **Vérifier** : chaque relation du MCD doit se retrouver sous forme de FK quelque part

Si tu termines avec autant de relations que de FK ajoutées, ton MLD est cohérent.

🧪 Exercice 1 — Bibliothèque scolaire

Voici un MCD simplifié :



Questions :

1. Combien de tables ce MLD aura-t-il ?
2. Pour chacune, donne sa clé primaire.
3. Où va la clé étrangère `eleve_id` ? La clé étrangère `livre_id` ?
4. Écris le MLD complet en notation textuelle (avec les types).

🧪 Exercice 2 — Refaire la billetterie sans regarder

Sans consulter la Partie 3, écris le MLD complet du système de billetterie en notation textuelle. Compare ensuite avec la correction.

🧪 Exercice 3 — Ton projet

Reprends le MCD que tu as construit en séquence 4 (garage, formation, Discord, ou autre). Applique la procédure de la Partie 6 et écris le MLD complet — types compris.

> ⚠️ *Vérification finale : chaque relation du MCD doit avoir donné lieu à **exactement une** clé étrangère dans ton MLD. Ni plus, ni moins.*

✍️ À retenir

- Une **entité** devient une **table** avec un `id` comme clé primaire.
- Une **relation** se matérialise par une **clé étrangère** : une colonne qui contient l'id d'une ligne d'une autre table.
- Pour une relation `1,1 / 0,n` (ou `0,1 / 0,n`) : la **FK va du côté** `0,n` (le côté « plusieurs ») et référence l'id du côté `1`.
- Le SGBDR garantit que chaque FK pointe vers une ligne existante — c'est l'**intégrité référentielle**.
- Un MLD bien écrit (avec PK, FK et types) contient tout ce qu'il faut pour créer les tables.

Suite

Tu as maintenant un MLD prêt à l'emploi. Dans le prochain cours, on ouvre HeidiSQL, on lance MySQL via Laragon, et on crée concrètement la première base et ses premières tables. Les clés étrangères, elles, auront leur cours dédié juste après.

Créer sa base et ses tables avec HeidiSQL

> *Premier contact avec un vrai SGBDR : on lance MySQL via Laragon, on se connecte avec HeidiSQL, on crée la base de données 'billetterie' et ses deux premières tables. Les clés étrangères suivent au cours suivant.*

Tu as un MLD complet sur papier : trois tables, leurs colonnes, leurs types. Il est temps de le transposer dans une vraie base MySQL. Dans ce cours, on couvre la **base de données** et **deux tables sans relation entre elles** (`client` et `evenement`). Les clés étrangères (`ticket`) ont leur cours dédié juste après — c'est trop important pour le bâcler ici.

🎯 Objectifs

À la fin de cette séquence, tu seras capable de :

1. Démarrer le service MySQL depuis Laragon
2. Te connecter à ton serveur MySQL avec HeidiSQL
3. Créer une base de données
4. Créer une table avec colonnes, types et clé primaire
5. Insérer des lignes via l'interface
6. Lire le SQL généré par HeidiSQL

Partie 1 — Démarrer MySQL

Vérifier que Laragon/Xampp tourne

Lance **Laragon**. Tu vois une petite fenêtre avec des boutons : **Démarrer tout**, **Arrêter tout**, **Menu**…

Clique sur **Démarrer tout** (ou **Start All**). Deux services s'allument :

- **Apache** (le serveur web — utile plus tard pour phpMyAdmin)
- **MySQL** (le SGBDR — c'est lui qu'on attaque ici)

> 💡 *Si tu vois deux pastilles vertes à côté des noms de services, c'est bon : le **serveur MySQL** tourne et écoute sur le port `3306` . Rappelle-toi du cours 06 : sans ça, aucun client ne peut se connecter.*

Vérification rapide

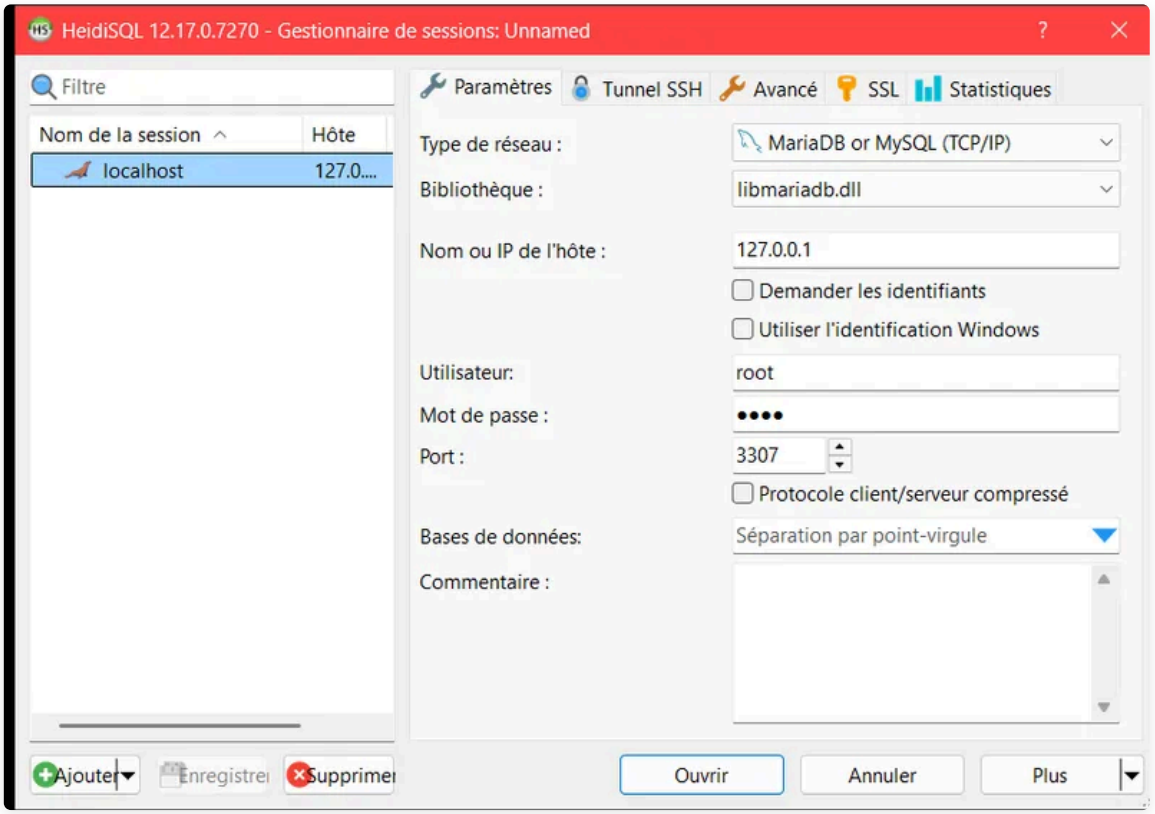
Dans le menu de Laragon, tu peux aller dans **MySQL** → **Version** : tu verras la version exacte de MariaDB ou MySQL installée. Pas critique, mais c'est un bon réflexe pour savoir avec quoi on travaille.

Partie 2 — Installer et lancer HeidiSQL

HeidiSQL est généralement **déjà installé avec Laragon**. Sinon, télécharge-le depuis heidisql.com (version *Installer*, gratuite).

Au premier lancement, tu arrives sur l'écran **Gestionnaire de sessions** : c'est là qu'on enregistre les connexions vers les serveurs MySQL.

▸ Créer une session



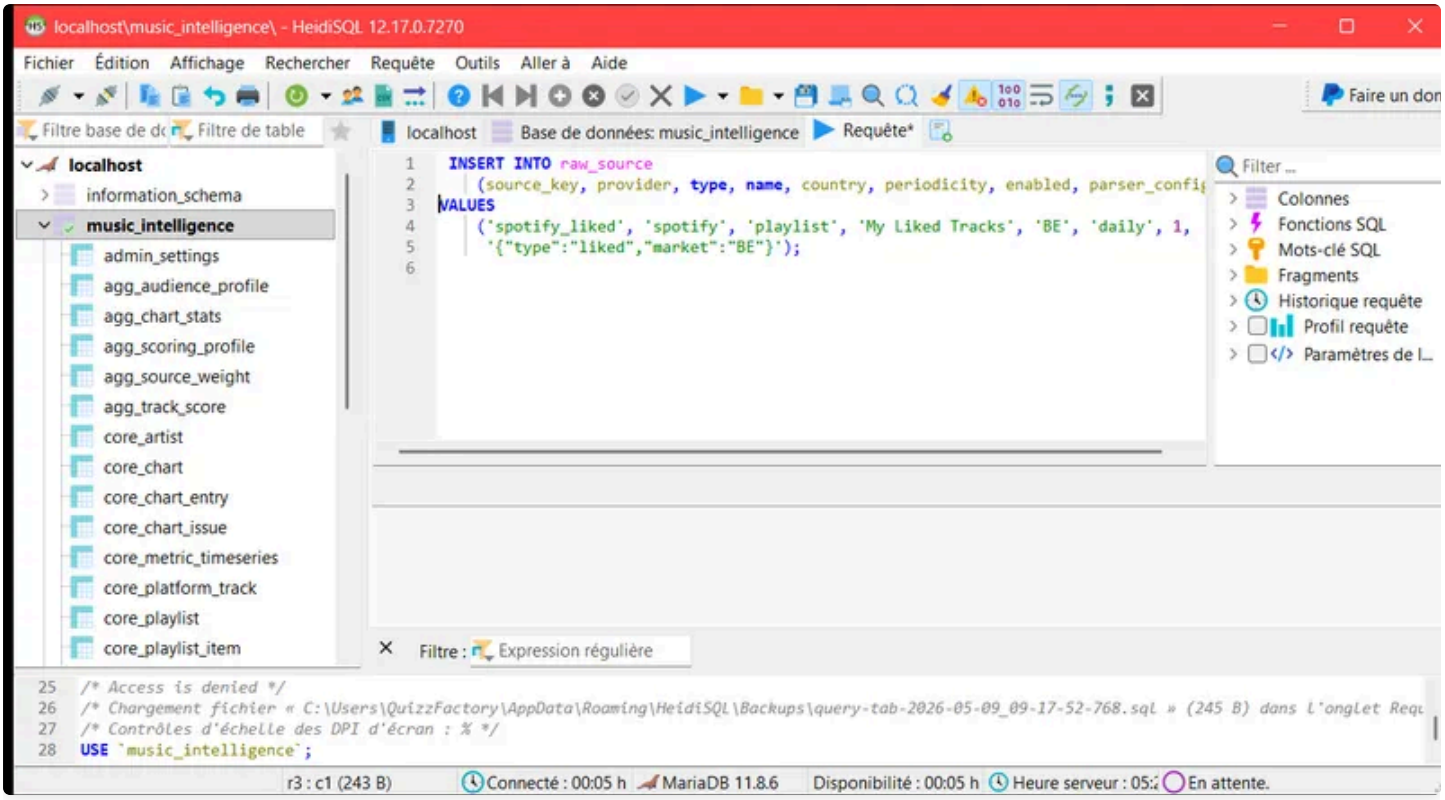
Clique sur **Nouveau** en bas à gauche. Remplis :

CHAMP	VALEUR	POURQUOI
Réseau	**MySQL (TCP/IP)**	On parle à MySQL en réseau, même si c'est local
Hôte / IP	`127.0.0.1` ou `localhost`	Le serveur tourne sur ta propre machine
Utilisateur	`root`	L'admin par défaut de MySQL
Mot de passe	*(vide)*	Laragon ne met pas de mot de passe par défaut
Port	`3306`	Le port standard de MySQL

Renomme la session en haut à gauche (par exemple `Laragon local`) puis clique **Ouvrir**.

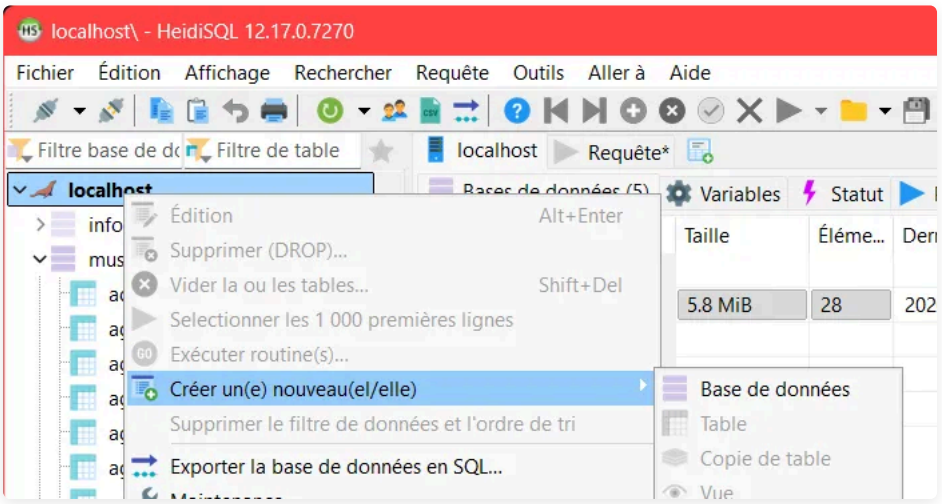
⚠️ *Le compte `root` sans mot de passe est acceptable **en local** pour apprendre. Sur un hébergement réel, c'est interdit — on aura un utilisateur dédié avec un mot de passe sérieux.*

▸ Tu es connecté(e)

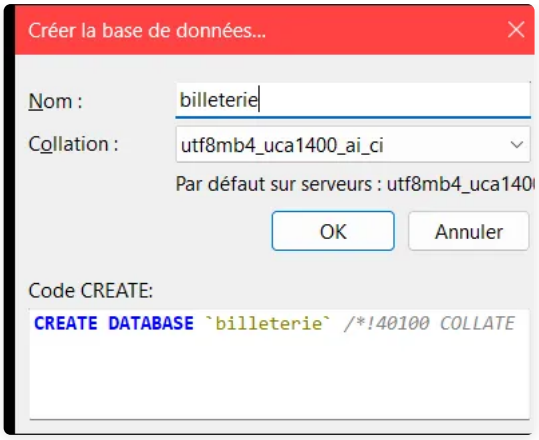


À gauche : un arbre avec ton serveur et la liste des **bases existantes** (`information_schema` , `mysql` , `performance_schema` …). Ces bases sont **internes au SGBDR** — n'y touche pas. La tienne, on va la créer.

Partie 3 — Créer la base `billetterie`



Dans l'arbre de gauche, **clic droit sur le nom de la session** (la racine, pas une base existante) → **Créer nouveau** → **Base de données…**



Remplis :

CHAMP	VALEUR
Nom	`billetterie`
Interclassement	`utf8mb4_unicode_ci`

Clique **OK**.

💡 ***L'interclassement*** *(collation) définit comment MySQL stocke et compare les caractères.* `utf8mb4_unicode_ci` *accepte tous les caractères Unicode (accents, emojis, alphabets non latins) et ignore la casse pour les comparaisons. C'est le bon choix par défaut en 2026 — c'est généralement l'option proposée d'office par HeidiSQL.*

ℹ️ ***Si*** `utf8mb4_unicode_ci` ***n'apparaît pas*** *dans la liste (cela dépend de la version de MariaDB/MySQL), prends dans l'ordre :*

- `utf8mb4_unicode_520_ci` *— équivalent, basé sur Unicode 5.20*
- `utf8mb4_uca1400_ai_ci` *(ou* `uca1400_ai_ci` *) — sur les MariaDB récentes, basé sur Unicode 14*
- `utf8mb4_general_ci` *— fallback de dernier recours, plus rapide mais moins précis (suffisant à ce stade)*

L'essentiel est que le nom commence par `utf8mb4_` *— pas* `utf8_` *tout court, qui ne stocke ni les emojis ni certains caractères asiatiques.*

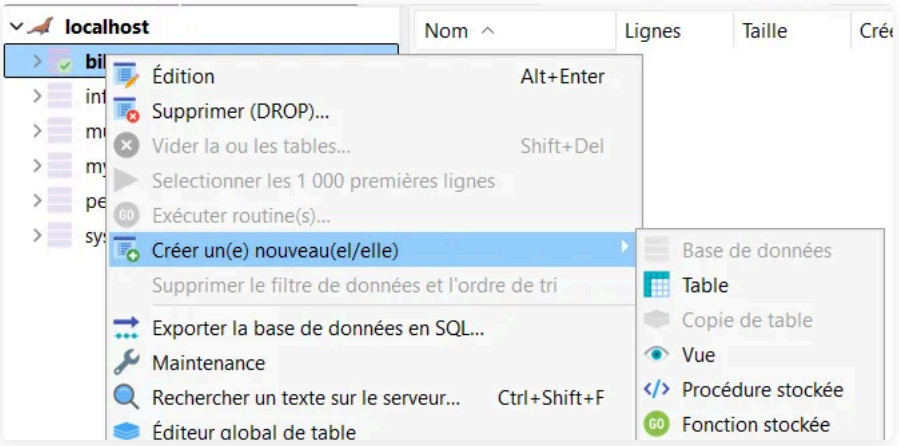
La base `billetterie` apparaît dans l'arbre — vide. On va la peupler.

Partie 4 — Créer la table `client`

Rappel du MLD :

```
client (
    id          INT  PK  AUTO_INCREMENT
    nom         VARCHAR(100)  NOT NULL
    prenom      VARCHAR(100)  NOT NULL
    email       VARCHAR(150)
    telephone   VARCHAR(20)
)
```

Lancer le formulaire



Clic droit sur la base `billetterie` **→ Créer nouveau → Table**.

Une fenêtre de conception s'ouvre. En haut, le champ **Nom** : tape `client` .

▸ Ajouter les colonnes

Dans le tableau central, chaque ligne représente une **colonne**. Clique **Ajouter** pour insérer une nouvelle colonne, puis remplis les champs :

Colonne 1 — `id` *(la plus importante)*

CHAMP	VALEUR
Nom	`id`
Type de données	`INT`
Autoriser NULL	**décoché**
Défaut	*(rien)*
Cocher la case	`AUTO_INCREMENT`

Puis, à droite du tableau, ajoute un **index** de type **PRIMARY** sur la colonne `id` (onglet *Index* ou bouton dédié — l'emplacement varie selon la version).

> 💡 ***Pourquoi ces deux réglages ?***
>
> `AUTO_INCREMENT` *dit à MySQL : à chaque nouvelle ligne, attribue automatiquement le prochain entier disponible. Tu n'as plus à inventer ou compter — la base s'occupe de générer un* `id` *unique pour chaque insertion. Sans ça, il faudrait fournir un* `id` *manuellement à chaque* `INSERT`*, avec le risque de doublons.*
>
> ***L'index*** `PRIMARY` *déclare que cette colonne est la* ***clé primaire*** *de la table. Trois conséquences techniques : MySQL refuse les doublons sur cette colonne, refuse les valeurs* `NULL`*, et crée un* ***index*** *qui permet de retrouver n'importe quelle ligne par son* `id` *en une fraction de seconde, même avec des millions d'enregistrements.*
>
> *En pratique,* ***les deux vont toujours ensemble*** *sur la colonne* `id` *d'une table :* `AUTO_INCREMENT` *garantit la génération automatique,* `PRIMARY` *garantit l'unicité et la rapidité d'accès. C'est ce qu'on a posé conceptuellement au cours 05 sur les identifiants — on le matérialise enfin ici.*

Colonnes 2 à 5

NOM	TYPE	NULL
`nom`	`VARCHAR(100)`	non
`prenom`	`VARCHAR(100)`	non
`email`	`VARCHAR(150)`	oui
`telephone`	`VARCHAR(20)`	oui

Clique **Enregistrer**. La table `client` apparaît dans l'arbre.

▸ Le SQL généré

En bas, HeidiSQL affiche **la requête SQL** qu'il vient d'envoyer au serveur :

```sql
CREATE TABLE `client` (
    `id`        INT          NOT NULL AUTO_INCREMENT,
    `nom`       VARCHAR(100) NOT NULL,
    `prenom`    VARCHAR(100) NOT NULL,
    `email`     VARCHAR(150) NULL,
    `telephone` VARCHAR(20)  NULL,
    PRIMARY KEY (`id`)
);
```

> 💡 ***Regarde-la attentivement.*** *C'est exactement ce qu'on aurait tapé à la main dans un terminal* `mysql`*. HeidiSQL est juste un générateur de SQL — comprendre le SQL généré te rendra autonome.*

Partie 5 — Créer la table `evenement`

Refais la même procédure pour la table `evenement` :

NOM	TYPE	NULL	NOTE
`id`	`INT`	non	PK, AUTO_INCREMENT
`titre`	`VARCHAR(200)`	non	
`date`	`DATETIME`	non	
`prix_base`	`DECIMAL(6,2)`	non	

Vérifie le SQL généré — il doit ressembler à :

```sql
CREATE TABLE `evenement` (
    `id`        INT           NOT NULL AUTO_INCREMENT,
    `titre`     VARCHAR(200)  NOT NULL,
    `date`      DATETIME      NOT NULL,
    `prix_base` DECIMAL(6,2)  NOT NULL,
    PRIMARY KEY (`id`)
);
```

Partie 6 — Insérer des données via l'interface

Sélectionne la table `client` dans l'arbre, puis va sur l'onglet **Données**. Tu vois un tableau vide.

Clique **Insérer une ligne** (en bas, ou bouton `+`). Remplis :

NOM	PRENOM	EMAIL	TELEPHONE
Dupont	Marie	marie.dupont@email.be	0471 12 34 56

Ne remplis **pas** la colonne `id` — laisse `AUTO_INCREMENT` faire son travail. Valide (Enter ou bouton **Publier**).

Ajoute deux ou trois autres clients. Tu verras que les `id` se suivent : 1, 2, 3…

Fais pareil pour `evenement` :

> 💡 *Le format* `DATETIME` *est* `AAAA-MM-JJ HH:MM:SS`*. HeidiSQL accepte aussi un petit calendrier graphique si tu cliques dans la cellule.*

TITRE	DATE	PRIX_BASE
Soirée Quiz Pop Culture	2026-10-03 20:00:00	12.00
Blind Test Années 90	2026-11-15 21:00:00	10.00
Quiz Cinéma	2026-12-05 20:30:00	15.00

Partie 7 — Vérifier avec une requête SQL

Onglet **Requête** (en haut). Tape :

```sql
SELECT * FROM client;
```

Appuie sur **F9** ou clique **Exécuter**. Le résultat s'affiche en dessous : toutes les lignes de la table `client`.

```sql
SELECT id, titre, date FROM evenement WHERE prix_base < 15;
```

Tu apprendras `SELECT` en détail dans un cours dédié — c'est juste pour valider que tes données sont bien là.

🧪 Exercices

Exercice 1 — Créer une 3e table simple

Crée une table `salle` avec ces colonnes :

NOM	TYPE	NULL
`id`	INT	non (PK + AUTO_INCREMENT)
`nom`	VARCHAR(150)	non
`ville`	VARCHAR(100)	non
`capacite`	INT	oui

Insère 3 salles.

Exercice 2 — Lire le SQL

Va dans la table `client`, fais clic droit → **Exporter en SQL** → **Vers le presse-papiers** (ou équivalent). Colle le résultat dans un fichier texte. Tu obtiens un **dump SQL** : c'est ce qui te permettra de transférer ta base ailleurs (et c'est exactement ce que phpMyAdmin produira aussi).

Exercice 3 — Ton projet

Sur ta machine, crée la base de données de **ton projet personnel** (celui de la séquence 4). Crée toutes ses tables **sauf** celles qui ont des clés étrangères. Insère quelques lignes dans chacune.

> ⚠️ *Si toutes tes tables ont des FK, garde la table principale (celle dont les autres dépendent) pour ce cours, et fais les autres au cours suivant.*

🧰 Petits réflexes utiles dans HeidiSQL

- **F5** : rafraîchir l'arbre des bases (utile si tu as créé quelque chose sans que ça apparaisse)
- **F9** : exécuter la requête SQL en cours
- **Ctrl + clic** sur une cellule : édition rapide
- **Onglet *Vue d'ensemble*** d'une table : un résumé des colonnes, indexes et taille
- **Bouton *Annuler*** : tant que tu n'as pas **Publié**, tes modifications de données ne sont pas envoyées au serveur

---

✍️ À retenir

- Avant tout : **MySQL doit tourner** dans Laragon (pastille verte).
- HeidiSQL se connecte avec `localhost`, user `root`, pas de mot de passe (en local seulement).
- Une **base de données** est créée à part — elle contient les tables.
- Pour chaque table : nom, colonnes avec leur type, et **clé primaire** `id INT AUTO_INCREMENT`.
- L'interface graphique est pratique mais **regarde toujours le SQL généré** — c'est lui la vérité.
- L'insertion de données ne nécessite **jamais** de remplir l'`id` à la main.

Suite

Tes deux tables `client` et `evenement` existent, mais **rien ne les relie**. Au prochain cours, on crée la table `ticket` avec ses **clés étrangères** — c'est elle qui matérialisera les relations du MCD dans la base.

Les clés étrangères en pratique

> *Créer la table 'ticket' qui relie clients et événements via des clés étrangères. C'est le moment où le MCD prend vie dans la base : le SGBDR refusera désormais toute incohérence.*

Au cours précédent, tu as créé `client` et `evenement`. Ce sont deux tables isolées : aucune ne sait qu'elle est liée à l'autre. C'est maintenant qu'on matérialise la relation `achète` / `concerne` du MCD avec une table `ticket` munie de **deux clés étrangères**.

🎯 Objectifs

À la fin de cette séquence, tu seras capable de :

1. Créer une table qui contient des clés étrangères
2. Déclarer une contrainte `FOREIGN KEY` dans HeidiSQL
3. Choisir un comportement `ON DELETE` / `ON UPDATE` adapté
4. Constater concrètement l'intégrité référentielle (le SGBDR refuse les incohérences)
5. Lire le `CREATE TABLE` complet généré avec ses contraintes

Partie 1 — Le rappel du MLD

```
client    (id, nom, prenom, email, telephone)
evenement (id, titre, date, prix_base)
ticket    (id, prix, statut, date_achat, #client_id, #evenement_id)
                                          ↳ client.id    ↳ evenement.id
```

La table `ticket` a deux clés étrangères, parce qu'elle participe à **deux relations** dans le MCD :

- `CLIENT 1,1 — achète — 0,n TICKET` → FK `client_id` côté ticket
- `TICKET 0,n — concerne — 1,1 ÉVÉNEMENT` → FK `evenement_id` côté ticket

C'est cohérent avec la règle du cours 07 : la FK va toujours du côté `0,n` (le côté « plusieurs »). Ici, le ticket est `0,n` dans ses deux relations — c'est donc lui qui reçoit les deux clés étrangères.

Partie 2 — Créer la table `ticket` (colonnes seulement)

Comme au cours précédent : clic droit sur la base `billetterie` → **Créer nouveau** → **Table**. Nom : `ticket`.

Les colonnes

NOM	TYPE	NULL	NOTE
`id`	`INT`	non	PK, AUTO_INCREMENT
`prix`	`DECIMAL(6,2)`	non	
`statut`	`VARCHAR(20)`	non	
`date_achat`	`DATETIME`	non	
`client_id`	`INT`	non	future FK
`evenement_id`	`INT`	non	future FK

Enregistre la table. Elle existe maintenant, mais `client_id` et `evenement_id` sont pour l'instant **de simples colonnes entières** : rien n'empêche d'y mettre `999` même si aucun client n°999 n'existe.

> 💡 *Avant la prochaine étape, vérifie que les colonnes FK ont **exactement le même type** que la PK référencée. Toutes les deux doivent être `INT NOT NULL`. Si les types diffèrent, MySQL refusera de créer la contrainte.*

Partie 3 — Ajouter les contraintes FOREIGN KEY

Ouvrir l'onglet des clés étrangères

Dans la fenêtre d'édition de la table `ticket`, va sur l'onglet **Clés étrangères** (ou **Foreign keys** selon ta langue). Tableau vide. Clique **Ajouter**.

Configurer la première FK : `client_id → client.id`

Configurer la seconde FK : `evenement_id → evenement.id`

Clique **Ajouter** à nouveau :

Enregistre. Si tout est bon, les deux contraintes apparaissent dans la liste.

CHAMP	VALEUR	CHAMP	VALEUR
Nom de la contrainte	`fk_ticket_client`	Nom de la contrainte	`fk_ticket_evenement`
Colonnes	`client_id`	Colonnes	`evenement_id`
Table référencée	`billetterie.client`	Table référencée	`billetterie.evenement`
Colonnes référencées	`id`	Colonnes référencées	`id`
ON UPDATE	`CASCADE`	ON UPDATE	`CASCADE`
ON DELETE	`RESTRICT`	ON DELETE	`RESTRICT`

Partie 4 — Que veulent dire `ON DELETE` et `ON UPDATE` ?

Ces options répondent à une question simple : **quand la ligne référencée bouge ou disparaît, qu'est-ce qui doit arriver aux lignes qui la référencent ?**

Les options

OPTION	EFFET QUAND LE CLIENT RÉFÉRENCÉ EST SUPPRIMÉ / MODIFIÉ
`RESTRICT`	**Refuser** l'opération tant que des tickets référencent ce client
`CASCADE`	**Propager** : supprimer aussi les tickets / mettre à jour la valeur
`SET NULL`	Mettre `client_id` à `NULL` (seulement si la colonne autorise NULL)
`NO ACTION`	Équivalent à `RESTRICT` en MySQL

Le choix par défaut raisonnable

- `ON DELETE RESTRICT` — on **refuse** de supprimer un client qui a des tickets. Sinon, on perdrait silencieusement de l'historique de vente. Pour supprimer le client, il faudra d'abord traiter ses tickets explicitement.
- `ON UPDATE CASCADE` — si pour une raison étrange l'id d'un client changeait, on veut que les tickets suivent. En pratique, les `id AUTO_INCREMENT` ne changent jamais, mais cette option ne coûte rien.

> ⚠️ `CASCADE` *sur* `ON DELETE` *peut sembler pratique ("je supprime le client, tout son historique part avec") mais c'est* ***dangereux*** *: une fausse manip efface des données qu'on ne voulait pas perdre. On l'utilise consciemment, jamais par défaut.*

Partie 5 — Le SQL complet

Voici ce que HeidiSQL a envoyé à MySQL :

```
CREATE TABLE `ticket` (
    `id`           INT          NOT NULL AUTO_INCREMENT,
    `prix`         DECIMAL(6,2) NOT NULL,
    `statut`       VARCHAR(20)  NOT NULL,
    `date_achat`   DATETIME     NOT NULL,
    `client_id`    INT          NOT NULL,
    `evenement_id` INT          NOT NULL,
    PRIMARY KEY (`id`),
    CONSTRAINT `fk_ticket_client`
        FOREIGN KEY (`client_id`) REFERENCES `client` (`id`)
        ON UPDATE CASCADE ON DELETE RESTRICT,
    CONSTRAINT `fk_ticket_evenement`
        FOREIGN KEY (`evenement_id`) REFERENCES `evenement` (`id`)
        ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Garde ce modèle sous les yeux : c'est exactement ce que tu écriras un jour à la main quand tu n'auras plus HeidiSQL sous la main.

Partie 6 — Tester l'intégrité référentielle

Le moment où le SGBDR montre **enfin** son utilité. Onglet **Requête**, exécute (F9) ces tests un par un.

Test 1 — Insertion valide

Vérifie d'abord les ids existants dans tes tables `client` et `evenement` (par exemple `SELECT id FROM client;`). Puis :

```
INSERT INTO ticket (prix, statut, date_achat, client_id, evenement_id)
VALUES (12.00, 'payé', '2026-10-01 14:30:00', 1, 1);
```

→ Succès. Le ticket est créé.

Test 2 — Client inexistant

```
INSERT INTO ticket (prix, statut, date_achat, client_id, evenement_id)
VALUES (12.00, 'payé', '2026-10-01 14:30:00', 999, 1);
```

→ **Erreur** :

```
Cannot add or update a child row: a foreign key constraint fails
```

Traduction : « Je refuse parce que ta clé étrangère pointe dans le vide. » Sans cette contrainte, tu te serais retrouvé avec un ticket orphelin — exactement le genre de bug silencieux qui pourrit une application en production.

Test 3 — Événement inexistant

```
INSERT INTO ticket (prix, statut, date_achat, client_id, evenement_id)
VALUES (15.00, 'réservé', '2026-10-01 15:00:00', 1, 999);
```

→ Même erreur, sur l'autre FK.

Test 4 — Supprimer un client référencé

```
DELETE FROM client WHERE id = 1;
```

→ **Erreur** :

```
Cannot delete or update a parent row: a foreign key constraint fails
```

C'est `ON DELETE RESTRICT` qui bloque : le client 1 a des tickets, on ne peut pas l'effacer sans gérer ses tickets d'abord. C'est exactement le comportement qu'on voulait.

> 💡 *Pour vraiment supprimer le client, il faudrait d'abord supprimer ses tickets (`DELETE FROM ticket WHERE client_id = 1;`), puis le client. C'est plus de travail — c'est aussi le but : t'obliger à réfléchir avant de perdre des données.*

Partie 7 — Visualiser les relations

HeidiSQL ne propose pas un vrai éditeur visuel de schéma, mais tu peux voir les relations en :

- Clic droit sur la table `ticket` → **Afficher CREATE TABLE** : tu vois toutes les contraintes en SQL
- Onglet **Clés étrangères** d'une table : la liste des FK sortantes

> 💡 *Pour un schéma visuel propre, des outils comme **MySQL Workbench** ou **DBeaver** offrent une vue Entity-Relationship. Optionnel à ce stade — le SQL et le MLD papier suffisent.*

💉 Exercices

Exercice 1 — Insérer une vraie cohorte

Insère **5 tickets** valides, répartis entre tes clients et tes événements existants. Vérifie le résultat avec :

```
SELECT t.id, c.nom, e.titre, t.prix, t.statut
FROM ticket t
JOIN client c    ON c.id = t.client_id
JOIN evenement e ON e.id = t.evenement_id;
```

Tu n'as pas vu `JOIN` formellement — l'idée : la requête montre côte à côte les infos de trois tables grâce aux FK. C'est tout l'intérêt du modèle relationnel.

Exercice 2 — Casser intentionnellement

Tente d'insérer un ticket avec un `client_id` qui n'existe pas. Relève le message d'erreur exact. À quoi sert ce blocage dans une vraie application (billetterie en ligne par exemple) ?

Exercice 3 — Comprendre ON DELETE

Modifie la contrainte `fk_ticket_client` pour mettre `ON DELETE CASCADE`. Réessaie de supprimer le client n°1. Que se passe-t-il pour ses tickets ? Pourquoi est-ce dangereux dans la vraie vie ? Remets ensuite la valeur sur `RESTRICT`.

Exercice 4 — Ton projet

Ajoute toutes les **clés étrangères** manquantes de ton projet personnel. Pour chaque FK, justifie en une phrase ton choix de `ON DELETE` (`RESTRICT` ou `CASCADE`).

✍️ À retenir

- Une clé étrangère est une colonne déclarée comme `FOREIGN KEY ( ... ) REFERENCES autre_table(id)` .
- La colonne FK doit avoir **exactement le même type** que la PK qu'elle référence.
- Le SGBDR **refuse** toute insertion ou mise à jour qui pointerait vers une ligne inexistante : c'est l'**intégrité référentielle**.
- `ON DELETE RESTRICT` est le défaut prudent ; `CASCADE` ne s'utilise que consciemment.
- Une table peut avoir **plusieurs** clés étrangères (autant que de relations auxquelles elle participe).
- HeidiSQL est un outil pratique ; le **SQL généré** est la source de vérité — apprends à le lire.

Suite

Tu as une base complète, propre, avec ses contraintes. Au prochain cours, on fait exactement le même travail dans **phpMyAdmin** — l'outil que tu utiliseras sur un hébergement web. Tu verras qu'au-delà de l'interface, ce sont les mêmes concepts et le même SQL en dessous.

phpMyAdmin : le même travail côté web

> *Refaire la création d'une base, de tables et de clés étrangères dans phpMyAdmin — l'interface web universelle des hébergements mutualisés. Même serveur MySQL, même SQL en dessous, autre habillage.*

Tu sais maintenant créer une base, des tables et des clés étrangères avec HeidiSQL. Très bien — sauf que **HeidiSQL n'est pas installé sur ton hébergement web**. Là-bas, ce que tu trouveras dans le panneau d'administration, c'est presque toujours **phpMyAdmin**.

phpMyAdmin n'est pas un autre SGBDR : c'est un **autre client**. Il parle au **même serveur MySQL** dans la **même langue (SQL)**. Tu vas le constater dans ce cours en refaisant tout le travail des deux séances précédentes — base, tables et clés étrangères — depuis l'interface web.

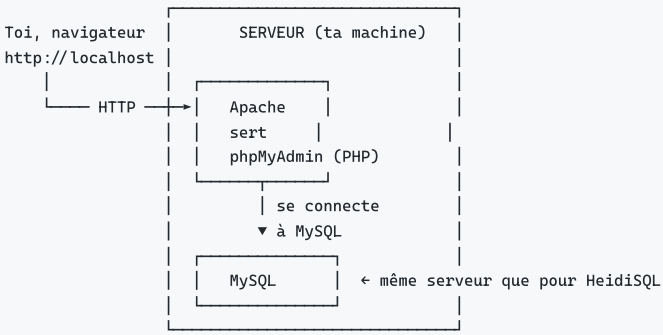
🎯 Objectifs

À la fin de cette séquence, tu seras capable de :

1. Ouvrir phpMyAdmin depuis Laragon
2. Naviguer dans son interface (panneau gauche, onglets)
3. Créer une base de données et ses tables
4. Définir des clés primaires et étrangères
5. Importer un dump SQL produit ailleurs
6. Identifier les différences pratiques avec HeidiSQL

Partie 1 — phpMyAdmin, qu'est-ce que c'est exactement ?

phpMyAdmin est une **application web** écrite en PHP. Elle est servie par **Apache** (le serveur web démarré par Laragon en même temps que MySQL).



Tu ouvres phpMyAdmin dans un navigateur. Apache exécute le code PHP. Ce code se connecte à MySQL et te renvoie une page HTML avec les résultats.

> 💡 *Sur un hébergement web réel, le scénario est identique. Le navigateur de l'admin (toi) parle à Apache distant, Apache parle au MySQL distant. Tu n'as rien à installer sur ta machine — c'est le grand intérêt.*

Partie 2 — Lancer phpMyAdmin

Depuis Laragon

1. Vérifie que **Apache** *et* **MySQL** tournent (pastilles vertes dans Laragon).
2. Dans le menu Laragon, clique sur **MySQL → phpMyAdmin**.

Ton navigateur s'ouvre sur une URL du type `http://localhost/phpmyadmin/`.

L'écran de connexion

Selon la configuration de Laragon, deux cas :

- **Connexion automatique** : tu arrives directement sur l'interface
- **Formulaire de connexion** : remplis `Utilisateur : root`, mot de passe vide

> ⚠️ *Comme pour HeidiSQL : `root` sans mot de passe, **uniquement en local**. Sur un hébergement, on a un utilisateur dédié à la base.*

Que vois-tu ?

- **Panneau gauche** : la liste des bases existantes — exactement les mêmes que dans HeidiSQL, puisque c'est le même serveur. Tu y trouves notamment la base `billetterie` que tu as créée au cours 08.
- **Panneau central** : tableau de bord, onglets *Bases de données*, *SQL*, *État*, *Comptes utilisateurs*, *Importer*, *Exporter*…

> 🧠 ***Vérification importante*** *: clique sur* `billetterie` *dans le panneau gauche. Tu vois* `client`, `evenement`, `ticket` *— avec leurs lignes ! C'est la* ***preuve*** *que phpMyAdmin et HeidiSQL parlent au même serveur. Tu peux gérer les mêmes données avec l'un ou l'autre indifféremment.*

Partie 3 — Refaire le travail dans une nouvelle base

Pour bien voir l'équivalence, on va créer une **nouvelle base** `billetterie_phpma` et refaire dedans le travail complet : tables `client`, `evenement`, `ticket` avec ses FK.

Créer la base

Onglet **Bases de données** (en haut). Sous *Créer une base de données* :

CHAMP	VALEUR
Nom	`billetterie_phpma`
Interclassement	`utf8mb4_unicode_ci`

Clique **Créer**. La base apparaît dans le panneau gauche.

Partie 4 — Créer la table `client`

Clique sur `billetterie_phpma` dans le panneau gauche. Tu vois *Aucune table dans la base de données.* Formulaire **Créer une table** :

CHAMP	VALEUR
Nom	`client`
Nombre de colonnes	`5`

Clique **Créer**. Tu arrives sur le tableau de définition des colonnes (une grosse grille).

Remplir les colonnes

#	NOM	TYPE	TAILLE	NULL	INDEX	A_I
1	`id`	`INT`	—	non	**PRIMARY**	☑
2	`nom`	`VARCHAR`	`100`	non	—	—
3	`prenom`	`VARCHAR`	`100`	non	—	—
4	`email`	`VARCHAR`	`150`	oui	—	—
5	`telephone`	`VARCHAR`	`20`	oui	—	—

> 💡 *La colonne* ***A_I*** *(AUTO_INCREMENT) est une simple case à cocher dans phpMyAdmin. Pour la PK, sélectionne PRIMARY dans la colonne* ***Index****.*

Tout en bas, clique **Enregistrer**.

Voir le SQL généré

Onglet **SQL** (en haut, ou bouton *Aperçu SQL* avant d'enregistrer) : phpMyAdmin te montre exactement la requête envoyée.

```sql
CREATE TABLE `billetterie_phpma`.`client` (
    `id`        INT          NOT NULL AUTO_INCREMENT,
    `nom`       VARCHAR(100) NOT NULL,
    `prenom`    VARCHAR(100) NOT NULL,
    `email`     VARCHAR(150) NULL,
    `telephone` VARCHAR(20)  NULL,
    PRIMARY KEY (`id`)
) ENGINE = InnoDB;
```

> 🧠 ***Compare avec le SQL généré par HeidiSQL au cours 08.*** *Tu verras qu'ils sont* ***identiques*** *(à quelques détails de mise en forme près). C'est la même langue, donc le même résultat — peu importe le client.*

Partie 5 — Créer `evenement` et `ticket`

Refais la procédure pour les deux autres tables.

evenement

#	NOM	TYPE	TAILLE	NULL	INDEX	A_I
1	`id`	`INT`	—	non	PRIMARY	☑
2	`titre`	`VARCHAR`	`200`	non	—	—
3	`date`	`DATETIME`	—	non	—	—
4	`prix_base`	`DECIMAL`	`6,2`	non	—	—

ticket (sans FK pour l'instant)

> 💡 *Sur les colonnes FK, choisis l'index* ***INDEX*** *(pas PRIMARY). MySQL exige un index sur les colonnes qui serviront de clé étrangère — phpMyAdmin te demandera de toute façon de l'ajouter au moment de créer la contrainte si tu l'oublies maintenant.*

#	NOM	TYPE	TAILLE	NULL	INDEX	A_I
1	`id`	`INT`	—	non	PRIMARY	☑
2	`prix`	`DECIMAL`	`6,2`	non	—	—
3	`statut`	`VARCHAR`	`20`	non	—	—
4	`date_achat`	`DATETIME`	—	non	—	—
5	`client_id`	`INT`	—	non	INDEX	—
6	`evenement_id`	`INT`	—	non	INDEX	—

Partie 6 — Ajouter les clés étrangères

Une fois la table `ticket` créée, ouvre-la (clic dans le panneau gauche) et va sur l'onglet **Structure**, puis sous-onglet **Vue relationnelle** (lien en bas de l'écran, parfois libellé *Relation view* ou *Gestion des relations*).

Tu obtiens un tableau pour configurer les FK. Pour chacune :

▸ **FK 1 — `client_id → client.id`**

CHAMP	VALEUR
Contrainte de clé étrangère	`fk_ticket_client`
Colonne	`client_id`
Base de données	`billetterie_phpma`
Table	`client`
Colonne	`id`
ON DELETE	`RESTRICT`
ON UPDATE	`CASCADE`

▸ **FK 2 — `evenement_id → evenement.id`**

CHAMP	VALEUR
Contrainte de clé étrangère	`fk_ticket_evenement`
Colonne	`evenement_id`
Base de données	`billetterie_phpma`
Table	`evenement`
Colonne	`id`
ON DELETE	`RESTRICT`
ON UPDATE	`CASCADE`

Clique **Enregistrer**. Si tu reçois une erreur : vérifie que les colonnes FK ont **exactement le même type** que les PK référencées (toutes en `INT NOT NULL`)

- vérifie que l'**index** est bien défini sur les colonnes `client_id` et `evenement_id`
- vérifie que tu utilises bien le moteur **InnoDB** (par défaut sur les versions récentes — MyISAM, lui, ne supporte pas les FK)

Partie 7 — Tester les contraintes

Onglet **SQL** (en haut, après avoir sélectionné `billetterie_phpma`). Exécute :

```
-- Insérer un client et un événement valides
INSERT INTO client (nom, prenom) VALUES ('Test', 'Démo');
INSERT INTO evenement (titre, date, prix_base)
VALUES ('Démo phpMyAdmin', '2026-12-01 20:00:00', 10.00);

-- Récupérer les ids
SELECT * FROM client;
SELECT * FROM evenement;
```

Puis, avec les ids retournés (probablement 1 et 1) :

```
INSERT INTO ticket (prix, statut, date_achat, client_id, evenement_id)
VALUES (10.00, 'payé', '2026-11-25 18:00:00', 1, 1);
```

→ succès.

```
INSERT INTO ticket (prix, statut, date_achat, client_id, evenement_id)
VALUES (10.00, 'payé', '2026-11-25 18:00:00', 999, 1);
```

→ **Erreur** :

```
#1452 – Cannot add or update a child row: a foreign key constraint fails
```

Même erreur, même comportement qu'avec HeidiSQL. Confirmation : les contraintes vivent **dans le serveur MySQL**, pas dans le client.

Partie 8 — Importer un dump SQL

C'est le cas d'usage typique du transfert vers un hébergement : tu as développé en local, tu veux mettre la même structure (et éventuellement les mêmes données) sur le serveur de production.

▸ Exporter depuis ta base locale

Reste pour l'instant dans phpMyAdmin sur la base `billetterie` (la première, celle du cours 08). Onglet **Exporter** :

- Méthode : **Personnalisée**

- Format : **SQL**
- Coche **Ajouter** `DROP TABLE` si tu veux écraser une base cible existante
- Décoche les options dont tu ne veux pas

Clique **Exécuter**. Un fichier `billetterie.sql` est téléchargé.

> 💡 *Ce fichier contient des `CREATE TABLE …` et des `INSERT INTO …`. C'est du SQL pur, portable d'un serveur à l'autre.*

Importer dans une autre base

Crée une nouvelle base `billetterie_import` (Partie 3, mais avec ce nom). Sélectionne-la, onglet **Importer** :

- Fichier à importer : sélectionne `billetterie.sql`
- Laisse les autres options par défaut
- Clique **Exécuter**

phpMyAdmin reproduit toutes les tables, contraintes et données. C'est exactement la procédure que tu utiliseras le jour où tu déploieras un projet sur un hébergement réel.

Partie 9 — HeidiSQL vs phpMyAdmin : quand utiliser quoi ?

CRITÈRE	HEIDISQL	PHPMYADMIN
Type	Application desktop (Windows)	Application web (PHP)
Installation	À installer sur ta machine	Déjà fourni par presque tous les hébergeurs
Performance	Très rapide sur grosses tables	Plus lent (chaque action = requête HTTP)
Édition de données	Fluide, façon tableur	Correcte mais formulaires lourds
Requêtes SQL	Confortable, suggestions	OK, mais on quitte la page facilement
Accès distant	Possible (avec tunnel SSH)	Natif (URL publique)
Outils intégrés	Export/import, profileur, snapshots	Export/import, comptes utilisateurs, état serveur

En pratique

- **Sur ta machine de dev** : HeidiSQL pour le confort quotidien.
- **Sur un hébergement mutualisé** : phpMyAdmin parce que c'est tout ce que tu as.
- **Sur un serveur dédié distant** : HeidiSQL via un tunnel SSH si c'est possible, sinon phpMyAdmin.

L'important : **les deux outils sont équivalents pour ce qu'ils font** — créer des bases, des tables, des contraintes, des requêtes. La différence est ergonomique.

🧪 Exercices

Exercice 1 — Vérifier l'équivalence

Ouvre HeidiSQL. Connecte-toi au même serveur. Constate que la base `billetterie_phpma` (créée via phpMyAdmin) y est présente, avec ses tables et ses contraintes. Insère un client via HeidiSQL ; vérifie qu'il apparaît dans phpMyAdmin. Conclus.

Exercice 2 — Import dans phpMyAdmin

Exporte une base depuis HeidiSQL (clic droit → *Exporter la base*). Importe le fichier obtenu dans phpMyAdmin (nouvelle base vierge). Vérifie que tout est là.

Exercice 3 — Le formulaire de connexion

Sur l'écran d'accueil de phpMyAdmin, repère ces informations :

- Quel utilisateur MySQL es-tu actuellement ?
- Quelle est la version de MariaDB / MySQL du serveur ?
- Combien de bases sont présentes sur le serveur (y compris les bases système) ?

Exercice 4 — Ton projet

Refais l'ensemble de **ton projet personnel** dans phpMyAdmin, dans une nouvelle base. Toutes les tables et toutes les clés étrangères. Compare le SQL généré avec celui qu'avait produit HeidiSQL au cours 09.

✍️ À retenir

- phpMyAdmin est un **client web** pour MySQL — exactement comme HeidiSQL est un client desktop.
- Il tourne grâce à **Apache** qui sert son code PHP ; les deux services doivent être démarrés dans Laragon.
- L'interface diffère, mais le **SQL généré est le même** et les **données sont les mêmes** (un seul serveur).
- Les contraintes de clé étrangère se déclarent dans la **Vue relationnelle** d'une table.
- **Importer / exporter** un dump SQL est la procédure standard pour déplacer une base d'un serveur à un autre.

Suite

Tu as maintenant toutes les compétences pour créer une base relationnelle bien construite, en local ou sur un hébergement. La prochaine étape — qui dépasse cette UAA — sera d'apprendre à **interroger et manipuler les données par SQL pur** : `SELECT`, `INSERT`, `UPDATE`, `DELETE`, puis les jointures pour exploiter les relations. Les articles existent déjà dans la section SQL du site.

SQL — Premières requêtes (SELECT, FROM, WHERE, ORDER BY)

> *Premiers pas pratiques en SQL sur une vraie base de données : 20 clients, 12 voitures et 200 locations à interroger. Tu apprends à importer la base dans HeidiSQL et phpMyAdmin, puis tu écris tes premières requêtes pour chercher, filtrer et trier.*

*Tu as vu **comment** une base relationnelle est structurée. Maintenant on l'utilise pour de vrai : on importe une base existante et on commence à l'interroger avec SQL. À la fin de ce cours, tu sauras répondre à des questions concrètes comme « quelles locations sont en cours ? » ou « quels sont les modèles à moins de 50 € par jour, triés du moins cher au plus cher ? ».*

🎯 Objectifs

À la fin de ce cours, tu seras capable de :

1. **Importer** un fichier `.sql` dans HeidiSQL et dans phpMyAdmin.
2. Lire le **schéma** d'une base (tables, colonnes, types).
3. Écrire une requête `SELECT … FROM …` pour afficher des lignes.
4. **Filtrer** les résultats avec `WHERE` et les opérateurs `=`, `<>`, `<`, `>`, `LIKE`.
5. **Combiner** plusieurs conditions avec `AND` et `OR`.
6. **Trier** les résultats avec `ORDER BY` (croissant, décroissant, multi-colonnes).

Partie 0 — Préparer la base de données

Avant de pouvoir écrire des requêtes, il faut **importer** la base dans ton SGBD. Télécharge d'abord le fichier `location_voitures_4e_ttinfo.sql` via le lien de téléchargement à droite de cette page.

> ℹ️ ***C'est quoi un fichier `.sql` ?*** *C'est un script texte qui contient des instructions SQL : `CREATE TABLE`, `INSERT INTO`, etc. Quand tu l'« exécutes » dans un SGBD, il rejoue toutes ces instructions et reconstruit la base de zéro.*

Méthode A — Import dans HeidiSQL (client desktop)

1. Démarre **Laragon**, puis clique sur « Démarrer tout » pour activer MySQL.
2. Ouvre **HeidiSQL** et connecte-toi à la session locale (`Hôte: 127.0.0.1`, `Utilisateur: root`, mot de passe vide).
3. Dans l'arborescence à gauche, **clic droit sur la session** → *Créer un nouveau* → *Base de données*. Nomme-la `location_voitures`, valide.
4. **Double-clique sur cette nouvelle base** dans l'arbre pour la « sélectionner » (son nom passe en gras).
5. Menu **Fichier** → **Charger un fichier SQL…** → choisis `location_voitures_4e_ttinfo.sql` dans tes téléchargements. Une nouvelle requête s'ouvre.
6. Appuie sur **F9** (ou clique sur le bouton bleu « ▶ Exécuter ») pour lancer le script.
7. Dans l'arbre à gauche, déplie `location_voitures`. Tu dois voir **3 tables** : `client`, `voiture`, `location`.

> ⚠️ ***Si HeidiSQL refuse :*** *vérifie que tu as bien double-cliqué sur la base `location_voitures` (et pas une autre) avant d'exécuter le script — les `CREATE TABLE` se font dans la base active.*

Méthode B — Import dans phpMyAdmin (client web)

1. Démarre **Laragon**, active MySQL et Apache.
2. Ouvre `http://localhost/phpmyadmin` dans ton navigateur.
3. Onglet **Bases de données** → tape `location_voitures` dans le champ « Créer une base de données » → clique **Créer**.
4. **Clique sur** `location_voitures` à gauche pour t'y placer.
5. Onglet **Importer** (en haut) → bouton *Parcourir…* → sélectionne `location_voitures_4e_ttinfo.sql`.
6. Tout en bas, clique **Importer**.
7. Tu dois voir un message vert *« L'importation a été exécutée avec succès »*. Les 3 tables apparaissent dans la barre latérale gauche.

> 💡 ***HeidiSQL vs phpMyAdmin*** *: tu obtiens **exactement la même base**, parce que les deux parlent au même serveur MySQL. Tu peux ouvrir les deux clients en même temps : ce que tu fais d'un côté apparaît immédiatement de l'autre.*

Vérification rapide

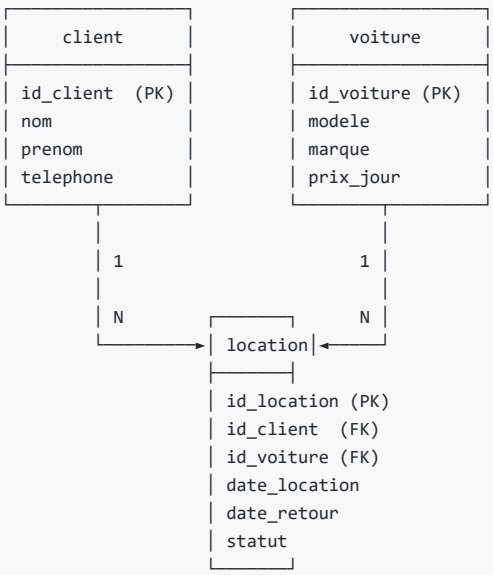
Quel que soit le client utilisé, ouvre une nouvelle requête SQL et tape :

```sql
SELECT COUNT(*) FROM location;
```

Tu dois obtenir **200**. Si oui, ta base est prête.

Partie 1 — Le schéma de la base

La base contient **3 tables** liées par des clés étrangères :



Un client peut faire **plusieurs** locations. Une voiture peut être louée **plusieurs** fois. Chaque ligne de `location` fait le lien entre **un** client et **une** voiture, à une date donnée.

Détail des colonnes

TABLE `CLIENT`

COLONNE	TYPE	DESCRIPTION
`id_client`	`INT` (PK)	Identifiant interne (auto-incrément)
`nom`	`VARCHAR(50)`	Nom de famille
`prenom`	`VARCHAR(50)`	Prénom
`telephone`	`VARCHAR(20)`	Numéro (unique)

TABLE `VOITURE`

COLONNE	TYPE	DESCRIPTION
`id_voiture`	`INT` (PK)	Identifiant interne
`modele`	`VARCHAR(50)`	Modèle (Clio, Golf, A3…)
`marque`	`VARCHAR(50)`	Marque (Renault, VW, Audi…)
`prix_jour`	`DECIMAL(6,2)`	Prix de location par jour, en €

TABLE `LOCATION`

COLONNE	TYPE	DESCRIPTION
`id_location`	`VARCHAR(5)` (PK)	Code lisible (`L001` , `L002`)
`id_client`	`INT` (FK → client)	Qui loue
`id_voiture`	`INT` (FK → voiture)	Quelle voiture
`date_location`	`DATE`	Date de début
`date_retour`	`DATE`	Date de retour
`statut`	`ENUM('terminée','en cours','réservée','annulée')`	État de la location

Partie 2 — `SELECT` et `FROM` : lire des lignes

C'est la requête la plus simple. Elle dit *« lis cette table »*.

2.1 — Tout afficher avec `SELECT *`

```
SELECT *
FROM client;
```

L'étoile `*` veut dire **toutes les colonnes**. Tu obtiens les 20 clients avec toutes leurs infos :

(20 lignes au total)

2.2 — Choisir des colonnes précises

Quand tu n'as besoin que de certaines colonnes, **liste-les** explicitement après `SELECT` , séparées par des virgules :

ID_CLIENT	NOM	PRENOM	TELEPHONE
1	Dupont	Marie	0471 12 34 56
2	Martin	Lucas	0485 22 11 00
3	Lambert	Sophie	0492 44 55 66
…	…	…	…

```
SELECT nom, prenom
FROM client;
```

Tu obtiens seulement 2 colonnes (et toujours 20 lignes) :

NOM	PRENOM
Dupont	Marie
Martin	Lucas
Lambert	Sophie
…	…

🧠 ***Bonne pratique*** *: préfère lister les colonnes plutôt que* `SELECT *` *. C'est plus lisible, plus rapide, et ça évite les surprises si quelqu'un ajoute une colonne à la table.*

2.3 — Limiter le nombre de lignes affichées

Quand une table contient des centaines ou des milliers de lignes (comme `location`, avec ses 200 entrées), `SELECT *` t'inonde la console. La clause `LIMIT n` dit *« ne montre que les n premières lignes »*.

```
SELECT *
FROM location
LIMIT 5;
```

Très pratique pour **jeter un œil** à une grande table sans tout charger.

Partie 3 — `WHERE` : filtrer les lignes

`SELECT … FROM …` lit **toutes** les lignes. Si tu veux uniquement celles qui remplissent une condition, ajoute une clause `WHERE`.

3.1 — L'égalité `=`

Quelles sont les locations dont le statut est « en cours » ?

```
SELECT id_location, id_client, date_location, date_retour
FROM location
WHERE statut = 'en cours';
```

Tu obtiens uniquement les lignes où `statut` vaut exactement `'en cours'`. Une trentaine de lignes au lieu de 200.

⚠️ ***Les chaînes vont entre guillemets simples*** *:* `'en cours'` *. Les nombres, eux, s'écrivent sans guillemets :* `id_client = 5` *. Pas* `id_client = '5'` *(ça marcherait par chance, mais c'est faux conceptuellement).*

3.2 — Les opérateurs de comparaison

OPÉRATEUR	SENS	EXEMPLE
`=`	égal	`prix_jour = 50`
`<>` ou `≠`	différent de	`statut <> 'annulée'`
`<`	strictement inférieur	`prix_jour < 50`
`>`	strictement supérieur	`prix_jour > 60`
`≤`	inférieur ou égal	`prix_jour ≤ 50`
`≥`	supérieur ou égal	`id_client ≥ 10`

EXEMPLE — LES VOITURES À PLUS DE 60 €

```
SELECT modele, marque, prix_jour
FROM voiture
WHERE prix_jour > 60;
```

Résultat :

MODELE	MARQUE	PRIX_JOUR
A3	Audi	75.00
Civic	Honda	62.00

3.3 — La recherche partielle avec `LIKE`

Pour chercher dans une chaîne sans donner sa valeur exacte, on utilise `LIKE` combiné avec le **joker** `%` (qui remplace zéro ou plusieurs caractères).

Quels sont les clients dont le nom commence par « L » ?

```
SELECT nom, prenom
FROM client
WHERE nom LIKE 'L%';
```

Résultat :

NOM	PRENOM
Lambert	Sophie
Leroy	Emma
Laurent	Adam
Leclercq	Tom

💡 ***Quelques motifs utiles*** *:*

- `'L%'` → *commence par L (Lambert, Leroy…)*
- `'%n'` → *finit par n (Dupont, Martin, Simon…)*
- `'%art%'` → *contient « art » (Martin, Bernard…)*
- `'D____'` → *exactement 5 caractères, commence par D (`_` remplace UN caractère)*

3.4 — Combiner plusieurs conditions : `AND` et `OR`

`AND` — TOUTES LES CONDITIONS DOIVENT ÊTRE VRAIES

*Quelles locations sont **en cours** ET concernent le **client 4** ?*

```
SELECT id_location, date_location, date_retour
FROM location
WHERE statut = 'en cours' AND id_client = 4;
```

`OR` — AU MOINS UNE CONDITION DOIT ÊTRE VRAIE

*Quelles voitures sont des **Renault** OU des **VW** ?*

```
SELECT modele, marque, prix_jour
FROM voiture
WHERE marque = 'Renault' OR marque = 'VW';
```

⚠️ ***Attention à l'ordre des conditions*** *quand tu mélanges `AND` et `OR`. Comme en mathématiques, `AND` a priorité sur `OR`. En cas de doute, **mets des parenthèses** :*

```
WHERE (marque = 'Renault' OR marque = 'VW') AND prix_jour < 55
```

Partie 4 — `ORDER BY` : trier les résultats

Par défaut, les lignes reviennent dans l'ordre où la base les a en stock — c'est-à-dire **dans n'importe quel ordre**. Pour les trier, on ajoute une clause `ORDER BY`.

4.1 — Trier dans l'ordre croissant

```
SELECT modele, marque, prix_jour
FROM voiture
ORDER BY prix_jour;
```

Les voitures sortent triées **du moins cher au plus cher** :

MODELE	MARQUE	PRIX_JOUR
500	Fiat	40.00
Corsa	Opel	42.00
Fiesta	Ford	43.00
Clio	Renault	45.00
Yaris	Toyota	48.00
208	Peugeot	50.00
…	…	…

ℹ️ `ASC` ***est implicite*** *: `ORDER BY prix_jour` et `ORDER BY prix_jour ASC` font la même chose. Le mot-clé `ASC` (ascending) est juste là pour être explicite.*

4.2 — Trier dans l'ordre décroissant avec `DESC`

*Les voitures les **plus chères** d'abord :*

```
SELECT modele, marque, prix_jour
FROM voiture
ORDER BY prix_jour DESC;
```

4.3 — Trier sur plusieurs colonnes

Tu peux trier sur **plusieurs critères** : la première colonne est le critère principal, la suivante départage les égalités.

*Les clients **par nom**, puis par **prénom** en cas d'homonymie :*

```
SELECT nom, prenom, telephone
FROM client
ORDER BY nom ASC, prenom ASC;
```

Partie 5 — Tout assembler

Les quatre clauses `SELECT / FROM / WHERE / ORDER BY` se combinent dans cet ordre :

```
SELECT  colonnes              -- QUOI afficher
FROM    table                 -- OÙ chercher
WHERE   conditions            -- QUELLES lignes garder
ORDER BY colonne(s)           -- COMMENT trier
LIMIT n;                      -- (optionnel) Combien
```

Exemple complet

*Les **10 locations les plus anciennes** dont le **statut est terminée**, triées de la plus ancienne à la plus récente :*

```
SELECT id_location, id_client, date_location, date_retour
FROM location
WHERE statut = 'terminée'
ORDER BY date_location ASC
LIMIT 10;
```

🧪 Exercice guidé — Locations chères, en cours

Énoncé. Tu veux la liste des **locations en cours**, **triées de la plus récente à la plus ancienne**, mais seulement **les 5 premières**.

Au lieu d'écrire la requête finale d'un coup, on la **construit par couches** — c'est la bonne façon de faire en SQL.

Étape 1 — Lecture brute

Premier brouillon : afficher toutes les locations, sans filtre.

```
SELECT id_location, date_location, statut
FROM location;
```

→ 200 lignes. C'est beaucoup. On commence à filtrer.

Étape 2 — Filtrer par statut

```
SELECT id_location, date_location, statut
FROM location
WHERE statut = 'en cours';
```

→ Une trentaine de lignes. On a déjà bien réduit.

Étape 3 — Trier de la plus récente à la plus ancienne

```
SELECT id_location, date_location, statut
FROM location
WHERE statut = 'en cours'
ORDER BY date_location DESC;
```

→ Les locations apparaissent maintenant **du plus récent au plus ancien**.

Étape 4 — Garder seulement les 5 premières

```
SELECT id_location, date_location, statut
FROM location
WHERE statut = 'en cours'
ORDER BY date_location DESC
LIMIT 5;
```

→ **5 lignes**, exactement comme demandé.

> 🧠 ***Le réflexe à acquérir** : ne tape **jamais** une requête complexe d'un seul coup. Commence par `SELECT * FROM table LIMIT 5` pour voir la donnée, puis ajoute **une clause à la fois** en vérifiant le résultat à chaque étape.*

🍼 Exercices supplémentaires

Pour chacun, écris la requête, exécute-la dans HeidiSQL ou phpMyAdmin, et **vérifie le nombre de lignes** attendu.

1. Affiche le **nom et prénom** de tous les clients, triés **par nom alphabétique**.
2. Affiche **toutes les colonnes** des voitures dont la marque est **Renault**.
3. Affiche les voitures à **moins de 50 €** par jour, triées **du moins cher au plus cher**.
4. Affiche les voitures dont la marque est **Renault** OU **VW**, triées **par prix décroissant**.
5. Affiche les **locations annulées** (toutes les colonnes).
6. Affiche les **id et téléphone** des clients dont le nom **finit par « t »**.
7. Affiche les locations du **client 4** dont le statut est **réservée**.
8. Affiche les **10 voitures** les **moins chères** (modèle, marque, prix). Indice : `ORDER BY` + `LIMIT`.
9. Affiche les locations dont la **date de location** est en **août 2026** (utilise `BETWEEN '2026-08-01' AND '2026-08-31'`).
10. **Défi** — Affiche les clients dont le **prénom commence par une voyelle**. Indice : `LIKE 'A%' OR LIKE 'E%' OR …`

✍️ À retenir

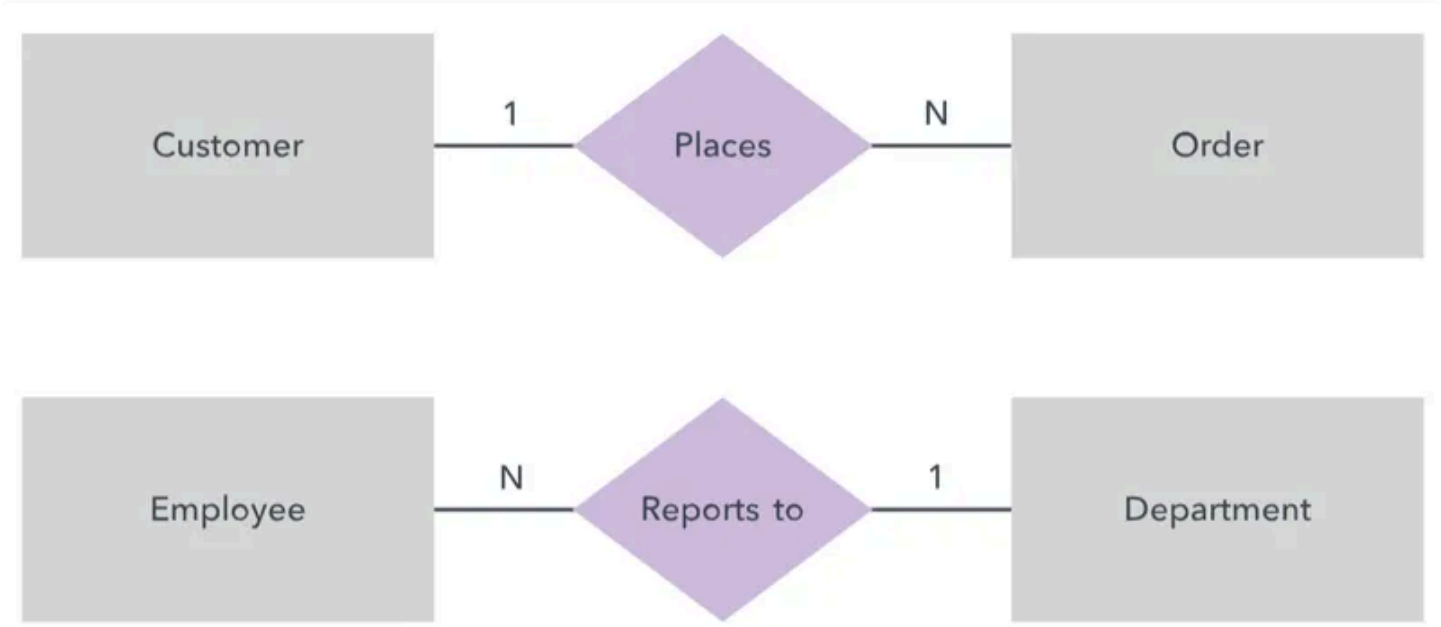
- Une requête SQL minimale s'écrit toujours dans l'ordre : `SELECT` colonnes, `FROM` table, *(* `WHERE`, `ORDER BY`, `LIMIT` optionnels)*.
- `SELECT *` prend toutes les colonnes. `SELECT col1, col2` ne prend que celles listées (à préférer).
- `WHERE` filtre les lignes. Opérateurs : `=`, `<>`, `<`, `>`, `≤`, `≥`, `LIKE`, combinés avec `AND` / `OR`.
- **Les chaînes** vont entre **guillemets simples** : `WHERE statut = 'terminée'`.
- `LIKE` + le joker `%` cherche dans une chaîne (commence par, finit par, contient).
- `ORDER BY` trie. `ASC` = croissant (par défaut), `DESC` = décroissant. Plusieurs colonnes possibles.
- `LIMIT n` garde uniquement les `n` premières lignes — utile pour explorer ou pour un *top N*.
- **Bonne méthode** : construire la requête **couche par couche**, en vérifiant le résultat à chaque étape.

Suite

Tu sais maintenant interroger **une seule table à la fois**. Mais regarde tes requêtes sur `location` : tu vois `id_client = 4` au lieu de « Dubois Nora », et `id_voiture = 11` au lieu de « Fiat 500 ». Pour afficher les **vrais noms** à la place des identifiants, il faut **combiner plusieurs tables** dans une seule requête — c'est le rôle des **jointures**, sujet du cours suivant.

Schéma Entités-Associations

> *La conception d'une base de données relationnelle efficace commence par une **modélisation** rigoureuse du système ou du business que l'on cherche à représenter. Le **schéma entité-association (E/A)** est un outil qui permet de **structurer** cette modélisation. Il offre une **vue conceptuelle** claire des entités, de leurs attributs et des relations entre elles avant de se lancer dans la création de la base de données physique.*



Dans cet article, nous explorerons les concepts fondamentaux liés au schéma E/A, en les reliant à la création de bases de données relationnelles robustes.

Qu'est-ce qu'un Schéma Entité-Association ?

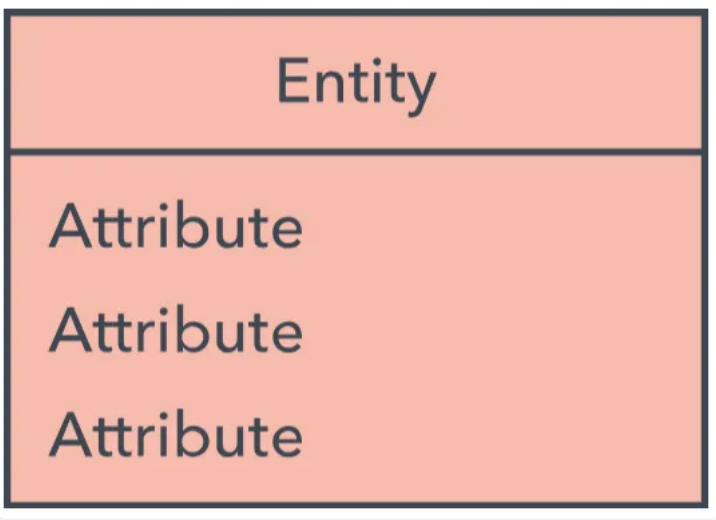
Le schéma entité-association est une **représentation graphique** utilisée pour **modéliser les données** d'un système. Il décrit :

- **Les entités** : Ce sont les objets ou concepts que le système doit représenter.
- **Les attributs** : Ce sont les caractéristiques ou propriétés des entités.
- **Les associations** : Ce sont les relations ou liens logiques entre les entités.
- **La cardinalité** : Elle indique combien d'instances d'une entité sont associées à une instance d'une autre entité.

Ce modèle est généralement représenté sous forme de **diagrammes E/A**, largement utilisés lors de la phase de conception d'un projet.

Les Entités

Une **entité** est un objet du monde réel ou un concept abstrait que vous souhaitez représenter dans votre base de données.



Exemple : Dans un système de gestion d'une bibliothèque, les entités pourraient être :

- **Livre**
- **Lecteur**
- **Emprunt**
- **Auteur**
- ...

Les Propriétés (ou Attributs)

Les propriétés sont les caractéristiques ou informations associées à une entité. Elles représentent les **colonnes** de la future table relationnelle.

Exemple : Pour l'entité **Livre**, les propriétés peuvent inclure :

- Titre
- Résumé
- ISBN (identifiant unique)
- Date de publication

Types d'attributs :

1. **Attributs simples** : Ils contiennent une valeur unique et indivisible. Exemple : Titre.
2. **Attributs composés** : Ils peuvent être décomposés en sous-attributs. Exemple : Nom complet (Nom + Prénom).
3. **Attributs multivalués** : Ils peuvent contenir plusieurs valeurs pour une entité donnée. Exemple : Genres d'un livre (roman, science-fiction).
4. **Attributs dérivés** : Ils peuvent être calculés à partir d'autres attributs. Exemple : Âge (calculé à partir de la date de naissance).

Identifiant Naturel (ou Clé Primaire)

Un **identifiant naturel** est un attribut ou un ensemble d'attributs qui permet d'**identifier de manière** `unique` **et** `invariable` **une entité**.

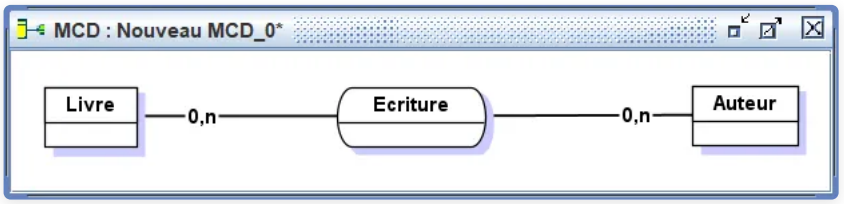
Exemple :

- L'ISBN pour un livre.
- Un numéro de sécurité sociale pour un individu.
- Le numéro de chassis d'une voiture
- La MAC adresse d'une carte réseau
- ...

Dans certains cas, on peut utiliser des **identifiants artificiels** comme des identifiants générés automatiquement (par exemple, un ID numérique unique).

Les Associations (ou Relations)

Une association décrit le **lien logique entre deux ou plusieurs entités**.



Exemple : Dans le système de bibliothèque :

- Une association `Ecrire` relie les entités `Auteur` et `Livre`
- Une association `Emprunter` relie les entités `Lecteur` et `Livre`.

La Cardinalité

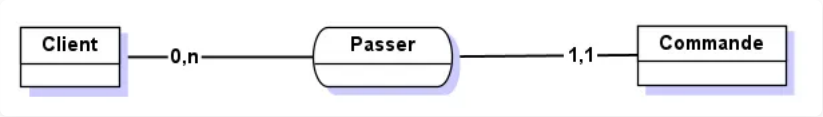
La cardinalité décrit le **nombre de fois qu'une instance d'une entité peut être associée à des instances d'une autre entité**. On indique toujours le nombre minimum et le nombre maximum de fois.

Types de cardinalité :

1. **1:1 (Un à Un)** : Chaque instance d'une entité est liée à une seule instance d'une autre entité.
   - Exemple : Un employé possède un badge unique.
2. **1:N (Un à Plusieurs)** : Une instance d'une entité est liée à plusieurs instances d'une autre entité.
   - Exemple : Un lecteur peut emprunter plusieurs livres.
3. **N:M (Plusieurs à Plusieurs)** : Plusieurs instances d'une entité peuvent être liées à plusieurs instances d'une autre entité.
   - Exemple : Un étudiant peut s'inscrire à plusieurs cours, et chaque cours peut accueillir plusieurs étudiants.

Comment écrire/lire les cardinalités? La cardinalité indique le **nombre d'entités** que l'on peut retrouver **de l'autre côté de l'association**.

Prenons l'exemple suivant:



Ce schéma indique qu'une commamnde peut être passée par un et un seul client (`1-1`)et qu'un client peut passer 0 ou plusieurs commandes (`0-N`).

Le fait d'utiliser le `1` comme cardinalité minimale indique un lien fort entre les deux entités. Dans le cas d'une commande client par exemple, on indique avec la cardinalité `1-1` que la commande ne peut pas exister sans client. Cela signifie que le client doit déjà exister dans la base de données avant de pouvoir créer une commande.D'un autre côté, la cardinalité 0-N indique que l'on peut tout à fait avoir des clients qui n'ont jamais passé de commande. Ce sera le cas au moment de la création d'un client dans la base de données.

Représenter les cardinalités





Du Modèle Conceptuel au Modèle Physique

La modélisation E/A se déroule en trois étapes principales : conceptuelle, logique et physique.

Modèle Conceptuel

Le **modèle conceptuel** est une représentation abstraite des entités et associations. Il utilise un diagramme E/A pour visualiser les données.

Exemple : Un schéma E/A simple pour un système de gestion de bibliothèque pourrait inclure :

- Entités : **Livre**, **Lecteur**, **Prêt**.
- Associations : **Emprunter** relie **Lecteur** à **Livre**.

Modèle Logique

Le **modèle logique** transforme les entités et associations en **tables relationnelles**. À ce stade :

- Les entités deviennent des tables.
- Les propriétés deviennent des colonnes.
- Les associations deviennent des relations (représentées souvent par des clés étrangères).

Exemple (suite de notre système de bibliothèque) :

- Table **Livre** : ID, Titre, Auteur, ISBN.
- Table **Lecteur** : ID, Nom, Email.
- Table **Prêt** : ID_Prêt, ID_Livre (clé étrangère), ID_Lecteur (clé étrangère), Date_Emprunt.

Modèle Physique

Le **modèle physique** est la traduction du modèle logique en structures concrètes dans un SGBD (Système de Gestion de Bases de Données) comme MySQL, PostgreSQL ou Oracle.

Exemple : Le modèle physique inclura :

- Définition des types de données (VARCHAR, INTEGER, DATE).
- Contraintes (PRIMARY KEY, FOREIGN KEY, NOT NULL).
- Index pour optimiser les performances.

Conseils Pratiques pour une Bonne Modélisation

1. **Identifiez clairement les entités et leurs propriétés** avant de définir les relations.
2. **Évitez la redondance des données** en normalisant votre modèle.
3. **Choisissez des clés primaires pertinentes** pour chaque entité.
4. **Prenez en compte les contraintes métier** (cardinalité, obligatoire ou optionnel).
5. Testez votre modèle en simulant des requêtes sur des cas concrets.

Conclusion

Le schéma entité-association est une étape essentielle pour concevoir une base de données relationnelle robuste. En maîtrisant les concepts d'entités, d'attributs, de relations et de cardinalité, vous pouvez modéliser efficacement un système ou un business. En passant du modèle conceptuel au modèle logique, puis au modèle physique, vous vous assurez que votre base de données répond aux besoins de votre application tout en restant performante et maintenable.

La rigueur lors de cette phase de conception est la clé d'un projet réussi, car une base de données bien modélisée facilite non seulement le développement initial mais aussi l'évolution future du système.

Schéma Entité-Association - Exercices de modélisation

> *Voici quelques exercices pour apprendre à identifier les entités et leurs relations.*

1. Restaurant : "La Bonne Bouffe"

Description : "La Bonne Bouffe" est un restaurant structuré autour de zones de service, des serveurs assignés, des tables, et une carte des plats.

- **Plats** : Chaque plat a un **nom**, un **prix**, et appartient à une **catégorie** (entrée, plat principal, dessert, boisson).
- **Catégories de plats** : Organisent les plats de la carte.
- **Zones** : Le restaurant est divisé en plusieurs zones (par exemple, terrasse, salle intérieure).
- **Tables** : Chaque table appartient à une seule zone et est identifiée par un numéro.
- **Serveurs** : Chaque serveur est assigné à une seule zone, mais une zone peut avoir plusieurs serveurs.

Relations clés :

- Un **serveur** est assigné à une **zone**.
- Une **zone** contient plusieurs **tables**.
- Chaque **plat** appartient à une **catégorie**.

---

2. Guide Touristique

Description : Un guide touristique recense des établissements de différents types dans plusieurs communes, il est édité par Guy de Michelin.

- **Établissements** : Chaque établissement a un **nom**, une **adresse**, et appartient à une **catégorie**.
- **Catégories** : Décrivent le type d'établissement (restaurant, gîte, cinéma, musée…).
- **Communes** : Chaque établissement est situé dans une seule commune.

Relations clés :

- Un **établissement** appartient à une **catégorie**.
- Un **établissement** est situé dans une **commune**.
- Une **commune** peut contenir plusieurs **établissements**.

---

3. Jeux Vidéo

Description : Une base de données pour gérer les jeux vidéo.

- **Jeux** : Chaque jeu a un **titre**, un **prix**, et appartient à une **catégorie** (aventure, FPS, RPG…).
- **Créateurs** : Chaque jeu est créé par un créateur.
- **Éditeurs** : Chaque jeu est publié par un éditeur unique.
- **Extensions** : Certains jeux ont des extensions, qui ajoutent du contenu au jeu de base.

Relations clés :

- Un **jeu** appartient à une seule **catégorie**.
- Un **jeu** peut avoir plusieurs **extensions**.
- Un **jeu** est associé à un seul **créateur**.
- Un **jeu** est publié par un seul **éditeur**.

---

4. Garage

Description : La garage "Richard Agiste" effectue des réparations sur les voitures des clients.

- **Réparations** : Chaque réparation a un **numéro unique**, une **description**, et un **coût**.
- **Voitures** : Chaque voiture a un **numéro d'immatriculation**, une **marque**, et un **modèle**.
- **Clients** : Un client possède une ou plusieurs voitures, identifiées par leur numéro d'immatriculation.

Relations clés :

- Une **réparation** est effectuée sur une seule **voiture**.
- Une **voiture** est associée à un seul **client**, mais un client peut posséder plusieurs voitures.

Ces exercices couvrent différents domaines et permettent d'apprendre à identifier les entités, leurs propriétés, et les relations pour construire des schémas E-A pertinents.

Identifier les entités

> *Identifier les entités est une étape essentielle dans la modélisation d'un système d'information, car elles constituent la base du modèle de données. Les entités représentent les objets ou concepts importants qui doivent être gérés par le système. Voici les moyens et techniques pour les repérer dans un **cahier des charges** ou d'autres documents d'analyse.*

1. Comprendre le contexte métier

Avant d'examiner en détail le cahier des charges, il est important de bien comprendre le domaine d'application du système (par exemple : gestion d'une école, d'un restaurant, d'une entreprise). Cela vous aidera à repérer les concepts clés qui sont au cœur de l'activité.

Questions à se poser :

- Quels sont les **objets importants** ou **concepts principaux** qui apparaissent souvent dans la description ?
- Quels éléments doivent être suivis, gérés ou traités par le système ?

---

2. Analyser les noms et concepts récurrents

Les entités sont souvent des **noms communs** ou des **concepts clés** que l'on retrouve fréquemment dans le cahier des charges ou les discussions avec les parties prenantes.

Méthodes pratiques :

- **Repérez les noms qui décrivent des objets tangibles ou intangibles.**
  - Exemple : Dans une entreprise, les termes "client", "produit", "commande" sont souvent des entités.
- **Identifiez les noms qui sont au centre des interactions ou des relations.**
  - Exemple : Dans une bibliothèque, "lecteur", "livre" et "prêt" sont des entités fondamentales.

---

3. Rechercher les groupes d'informations

Une entité est généralement un concept qui regroupe plusieurs **caractéristiques** ou **informations** liées. Dans le cahier des charges, recherchez les descriptions contenant des listes d'attributs ou des détails associés.

Exemple :

Dans une description comme :

> *"Chaque employé a un nom, un prénom, une date de naissance, et un poste."*
> *L'objet "Employé" est une entité, car il regroupe plusieurs attributs (nom, prénom, etc.).*

---

4. Identifier les acteurs ou rôles

Les **acteurs** ou **rôles** mentionnés dans le cahier des charges peuvent être des entités, car ils interagissent directement avec le système.

Exemple :

- Dans un système de gestion scolaire : "élève", "professeur", et "administrateur" sont des acteurs qui peuvent correspondre à des entités.
- Posez-vous la question : "Ces acteurs doivent-ils être représentés dans la base de données ?"

---

5. Rechercher les éléments qui nécessitent un suivi ou un stockage

Dans le cahier des charges, notez les objets ou concepts que le système doit :

- Stocker durablement.
- Suivre ou mettre à jour.
- Représenter dans des rapports ou des analyses.

Exemple :

Dans un système de gestion de commandes :

> *"Nous devons conserver les informations sur chaque produit, comme son nom, son prix et sa catégorie."*

- "Produit" est ici une entité à suivre et à gérer.

---

6. Identifier les interactions ou relations

Les relations entre les concepts peuvent aider à repérer des entités. Si deux concepts sont connectés par une interaction, il est probable qu'ils soient des entités distinctes.

Exemple :

Dans un restaurant, si vous trouvez une description comme :

> *"Chaque serveur est assigné à une zone, et chaque table appartient à une zone."*

- Les concepts **Serveur**, **Zone**, et **Table** sont des entités distinctes.

---

7. Repérer les objets décrits comme uniques

Un objet ou concept est souvent une entité s'il est identifié comme unique ou s'il possède un identifiant (par exemple, un numéro, un code ou un nom distinctif).

Exemple :

Dans un système de gestion des ventes :

> *"Chaque commande a un numéro unique, une date, et un client associé."*

- Ici, **Commande** est une entité car elle possède un identifiant unique.

---

8. Vérifier les termes métiers spécifiques

Le vocabulaire utilisé dans le cahier des charges ou les documents métiers peut révéler des entités importantes. Les termes propres à un domaine d'activité (jargon métier) sont souvent des entités.

Exemple :

- Dans un garage : "Réparation", "Voiture", "Client".
- Dans un hôpital : "Patient", "Médecin", "Consultation".

---

9. Analyser les scénarios d'utilisation

Les scénarios ou cas d'utilisation décrivent les interactions entre les utilisateurs et le système. Ces interactions impliquent souvent des entités.

Exemple :

Un scénario pourrait indiquer :

> *"Un client passe une commande composée de plusieurs produits."*

- Les entités ici sont **Client**, **Commande**, et **Produit**.

---

10. Se poser les bonnes questions

Pour confirmer qu'un concept est une entité, posez-vous les questions suivantes :

- Ce concept peut-il avoir **plusieurs occurrences** distinctes ? (Exemple : plusieurs clients, plusieurs produits.)
- A-t-il des **caractéristiques spécifiques** qui le décrivent ? (Exemple : un client a un nom, un email, un numéro de téléphone.)
- Est-il impliqué dans une **relation avec d'autres concepts** ? (Exemple : un client passe une commande.)

Exemple d'application

Cahier des charges : Gestion d'une bibliothèque

> *"Nous devons gérer les livres. Chaque livre appartient à une catégorie (par exemple, roman ou essai). Les lecteurs peuvent emprunter plusieurs livres à la fois. Chaque emprunt est associé à une date de début et une date de fin."*

1. Concepts identifiés :
   - **Livre** (car il est décrit comme un objet ayant des caractéristiques spécifiques).
   - **Catégorie** (car chaque livre est associé à une catégorie).
   - **Lecteur** (car ce sont les utilisateurs du système).
   - **Emprunt** (car il décrit une relation entre Lecteur et Livre, avec des informations propres comme les dates).

Conclusion

Identifier les entités dans un cahier des charges ou d'autres documents d'analyse demande une bonne compréhension du contexte métier et des objectifs du système. En appliquant ces techniques et en utilisant des questions ciblées, vous serez capable d'extraire les concepts clés et de poser les bases d'un modèle E-A clair et structuré.

Système de Gestion de Base de Données (SGBD)

> *Un **Système de Gestion de Base de Données (SGBD)** est un logiciel permettant de créer, manipuler, gérer et interroger des bases de données. Il offre une interface entre les utilisateurs ou les applications et les données elles-mêmes, tout en assurant leur organisation, sécurité et accessibilité.*

Intérêt d'un SGBD

Un SGBD (GB ***dbms, database management system***) est un logiciel qui permet de gérer efficacement des **données structurées**. Il est conçu pour répondre aux besoins suivants :

1. **Stockage structuré des données** : Les SGBD permettent d'**organiser** les données sous forme de **tables**, **colonnes** et **relations**.
2. **Manipulation des données** : Les utilisateurs peuvent **interroger**, **insérer**, **mettre à jour** ou **supprimer** des données à l'aide de langages comme SQL.
3. **Partage des données** : Plusieurs utilisateurs ou applications peuvent **accéder simultanément** aux données sans conflit.
4. **Sécurité des données** : Les SGBD offrent des outils pour gérer les **permissions** d'accès.
5. **Intégrité des données** : Les **contraintes** (unicité, clés étrangères, etc.) garantissent la **qualité** et la **cohérence** des données.
6. **Sauvegarde et récupération** : Ils offrent des mécanismes pour sauvegarder les données et les restaurer en cas de panne ou de perte.

Questions:

- Quand il s'agit de gérer efficacement des données, à quels besoins un SGBD répond-il?
- Quelles manipulations de données sont prises en charge par les SGBD?
- Sous quelles formes les SGBD permettent-ils d'organiser les données?

Avantages d'un SGBD par rapport aux fichiers

ASPECT	SGBD	SYSTÈME BASÉ SUR DES FICHIERS
Organisation	Données organisées en tables et relations	Données souvent dispersées dans plusieurs fichiers
Redondance	Gestion automatique pour éviter les duplications	Fort risque de duplication
Concurrence	Gestion des accès multiples	Difficultés à gérer les accès simultanés
Sécurité	Permissions, authentification, cryptage	Sécurité manuelle, faible ou inexistante
Intégrité	Contraintes pour assurer la cohérence	Contrôle manuel nécessaire
Interrogation	Langage structuré (SQL) pour interroger les données	Nécessité de parcourir manuellement les fichiers
Performance	Indexation et optimisation intégrées	Performances limitées pour de grandes quantités de données
Récupération	Mécanismes de sauvegarde et restauration	Nécessité d'écrire des scripts spécifiques

Comparez SGBD et Fichiers en ce qui concerne... (l'organisation, la redondance, la concurrence....)

Principaux SGBD

Il existe différents types de SGBD. Le choix du SGBD pour votre projet dépendra principalement de votre besoin en termes de stockage et d'interrogation de vos données.

- **Bases de données relationnelles (sgbd-r, rdbms)**:
  - Dans le cas des bases de données relationnelles, les données sont organisées en **tables**, reliées entre elles par des **clés**.
  - La structure est bien définie
  - Les données doivent respecter la structure définie
  - Il existe un langage standardisé pour les requêtes
  - Idéal pour les systèmes qui nécessitent de nombreuses relations entre les données.
- **Bases de données NO-SQL**:
  - Les bases de données no-sql sont conçues pour gérer des données qui sont **peu ou pas structurées**.
  - La structure des données contenues dans la base ne sont pas forcément connues à l'avance.
  - La structure des données ne suit pas forcément un schéma préétabli
  - Il existe différents sous-types de base de données en fonction du type de données que l'on veut enregistrer:
    - des documents,
    - des colonnes
    - des ensembles de clés-valeurs
    - des graphes
    - ...
- **Bases de données orientées objets**:
  - Les données sont stockées sous forme d'objets avec leurs propriétés et leurs méthodes
  - On se rapproche fortement des objets tels qu'ils sont gérés dans les programmes informatiques.

SGBD Relationnels

Les bases relationnelles (sgbd**r** ou **r**dbms) organisent les données en **tables liées entre elles** organisées selon un **schéma** préétabli.

Citez les 6 principaux SGBD relationnels (sql) du marché.

SGBD NoSQL

Conçus pour les données non structurées ou semi-structurées.

SGBD	CARACTÉRISTIQUES PRINCIPALES	LICENSE
MySQL	Rapide, open source, très utilisé pour les sites web et les applications web.	Open source
MariaDB	Fork de MySQL, entièrement open source, avec des fonctionnalités avancées.	Open source
PostgreSQL	Très puissant et extensible, avec un support avancé des standards SQL.	Open source
SQLite	Léger, intégré dans des applications locales comme les smartphones ou les navigateurs.	Open source
Oracle DB	Orienté entreprise, riche en fonctionnalités et très performant.	Propriétaire
Microsoft SQL Server	Très utilisé dans les environnements Windows, avec de nombreux outils d'intégration.	Propriétaire

SGBD	CARACTÉRISTIQUES PRINCIPALES
MongoDB	Base de données orientée document (JSON-like), très flexible.
Cassandra	Orientée colonne, conçue pour des volumes massifs et la haute disponibilité.
Redis	Base clé-valeur en mémoire, très rapide pour les applications temps réel.
Neo4j	Base de données orientée graphe pour modéliser les relations complexes.

Quels sont les principaux SGBD relationnels open-source? Quels sont les principaux SGBD relationnels propriétaires? Quel SGBD très léger est utilisé pour des applications qui traitent des petits volumes de données?

Conclusion

Les SGBD sont des outils indispensables pour gérer efficacement les données dans des contextes variés : applications web, systèmes d'information d'entreprise, analyse de données, etc. En fonction des besoins (portabilité, performance, volume de données, coût), un SGBD relationnel ou NoSQL peut être choisi. La grille de comparaison aide à sélectionner le bon SGBD pour chaque projet. Pour un usage quotidien, les standards comme MySQL, PostgreSQL ou SQLite restent des choix sûrs et polyvalents.

SQL

> *SQL (**Structured Query Language**) est un langage utilisé pour interagir avec les bases de données relationnelles. Il permet de créer, lire, mettre à jour et supprimer des données. Cet article présente une introduction aux types de requêtes SQL, avec des exemples simples adaptés à MySQL/MariaDB et SQLite.*

Bien que SQL soit un langage standardisé, chaque système de gestion de base de données (SGBD) implémente ses propres variantes, appelées "dialectes". Ces dialectes introduisent des différences au niveau des fonctionnalités, de la syntaxe ou du comportement des commandes SQL.

La Norme SQL

La norme SQL est un ensemble de spécifications définies pour garantir une certaine portabilité et compatibilité des commandes SQL entre différents SGBD. Elle définit les fonctionnalités de base comme :

- Les commandes DDL (`CREATE` , `ALTER` , `DROP`) pour la gestion des structures.
- Les commandes DML (`INSERT` , `UPDATE` , `DELETE`) pour la manipulation des données.
- Les commandes DQL (`SELECT`) pour les requêtes.
- ...

Cependant, chaque SGBD interprète cette norme de manière spécifique et propose souvent des extensions ou des fonctionnalités non standard.

1. **Connaissez votre SGBD** : Chaque SGBD a ses spécificités. Lorsque vous développez une application, choisissez le SGBD qui correspond le mieux à vos besoins.
2. **Favorisez les commandes standard** : Autant que possible, utilisez des commandes SQL conformes à la norme pour garantir la portabilité.
3. **Adaptez les optimisations** : Les optimisations (comme les index ou les commandes spécifiques) peuvent varier selon les SGBD. Apprenez à exploiter les forces de chaque dialecte.
4. **Documentation** : Consultez la documentation officielle de votre SGBD pour connaître ses extensions, limitations, et comportements spécifiques.

Les "dialectes" SQL permettent de tirer parti des fonctionnalités avancées de chaque SGBD, mais en comprendre les différences est essentiel pour écrire des requêtes robustes et portables.

Dans les cours de cette année, nous aborderont les dialectes propres à SQLite3 et MySql/MariaDB.

Les Types de Requêtes SQL

SQL peut être divisé en plusieurs catégories de requêtes selon leur objectif. Voici les principaux types :

Requêtes DDL (Data Definition Language)

Ces requêtes permettent de définir ou modifier la structure de la base de données : créer, modifier ou supprimer des tables et d'autres objets.

Requêtes DML (Data Manipulation Language)

Ces requêtes permettent de manipuler les données à l'intérieur des tables : insertion, mise à jour ou suppression des enregistrements.

Requêtes DQL (Data Query Language)

Ces requêtes permettent d'interroger les données à l'aide de la commande `SELECT` .

Autres types

Il existe d'autres types de requêtes SQL. Bien qu'importants dans le monde des bases de données, ces types ne seront pas abordés dans ces cours d'introduction.

Requêtes DCL (Data Control Language)

Ces requêtes permettent de gérer les permissions et la sécurité des bases de données.

Requêtes TCL (Transaction Control Language)

Ces requêtes permettent de contrôler les transactions, comme valider ou annuler des modifications.

Les Requêtes DDL

Création d'une table

MYSQL/MARIADB ET SQLITE

```
CREATE TABLE utilisateurs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nom TEXT NOT NULL,
    age INTEGER,
    email TEXT UNIQUE
);
```

Ce script crée une table `utilisateurs` avec 4 colonnes : `id`, `nom`, `age` et `email`.

Modification d'une table

MYSQL/MARIADB

```
ALTER TABLE utilisateurs ADD colonne_creation DATE;
```

SQLITE

```
ALTER TABLE utilisateurs ADD COLUMN colonne_creation DATE;
```

Ce script ajoute une colonne `colonne_creation` à la table `utilisateurs`.

Suppression d'une table

MYSQL/MARIADB ET SQLITE

```
DROP TABLE IF EXISTS utilisateurs;
```

Les Requêtes DML

Insertion de données

```
INSERT INTO utilisateurs (nom, age, email)
VALUES ('Alice', 25, 'alice@example.com');
```

Ce script insère une ligne dans la table `utilisateurs`.

Mise à jour de données

MYSQL/MARIADB ET SQLITE

```
UPDATE utilisateurs
SET age = 26
WHERE nom = 'Alice';
```

Ce script met à jour l'âge de l'utilisateur nommé "Alice".

Suppression de données

MYSQL/MARIADB ET SQLITE

```
DELETE FROM utilisateurs
WHERE age < 20;
```

Ce script supprime tous les utilisateurs ayant moins de 20 ans.

Les Requêtes DQL

Sélection de toutes les données

MYSQL/MARIADB ET SQLITE

```sql
SELECT * FROM utilisateurs;
```

Ce script récupère toutes les colonnes et lignes de la table `utilisateurs`.

Sélection avec condition

MYSQL/MARIADB ET SQLITE

```sql
SELECT nom, email
FROM utilisateurs
WHERE age > 25;
```

Ce script récupère les colonnes `nom` et `email` des utilisateurs âgés de plus de 25 ans.

Tri des données

MYSQL/MARIADB ET SQLITE

```sql
SELECT nom, age
FROM utilisateurs
ORDER BY age DESC;
```

Ce script récupère les colonnes `nom` et `age` triées par ordre décroissant de l'âge.

Les Requêtes DCL

Donner des permissions

MYSQL/MARIADB

```sql
GRANT SELECT, INSERT ON utilisateurs TO 'nouvel_utilisateur'@'localhost';
```

Ce script donne les permissions `SELECT` et `INSERT` à un utilisateur nommé `nouvel_utilisateur`.

SQLITE

SQLite ne gère pas les permissions utilisateur directement, car il est conçu pour être utilisé comme une base de données légère embarquée.

Révoquer des permissions

MYSQL/MARIADB

```sql
REVOKE INSERT ON utilisateurs FROM 'nouvel_utilisateur'@'localhost';
```

Les Requêtes TCL

Validation des modifications

MYSQL/MARIADB ET SQLITE

```sql
COMMIT;
```

Ce script valide toutes les modifications effectuées dans une transaction.

Annulation des modifications

MYSQL/MARIADB ET SQLITE

```
ROLLBACK;
```

Ce script annule toutes les modifications effectuées depuis le début de la transaction.

Conclusion

SQL est un langage puissant qui offre une variété de commandes pour manipuler des bases de données. Cet article a présenté les principales catégories de requêtes et des exemples simples pour chaque type. Ces bases permettent de débuter avec SQL et de comprendre les actions fondamentales, que vous utilisiez MySQL, MariaDB ou SQLite. Pour aller plus loin, explorez des notions avancées comme les jointures, les sous-requêtes et les index.

Les Requêtes SQL de Type SELECT

> *Les requêtes SQL de type `SELECT` permettent d'extraire des données d'une base de données. Elles constituent le cœur de la manipulation des données relationnelles. Ce cours détaille les principales fonctionnalités de `SELECT` avec des exemples concrets en SQLite et MySQL, tout en mettant en lumière les différences de syntaxe lorsque c'est pertinent.*

Structure générale d'une requête SELECT

La structure de base d'une requête `SELECT` est la suivante :

```
SELECT colonne(s)
FROM table
[WHERE condition]
[GROUP BY colonne(s)]
[HAVING condition]
[ORDER BY colonne(s) [ASC|DESC]]
[LIMIT n OFFSET m];
```

Sélection explicite de colonnes et SELECT *

`SELECT` permet de choisir les colonnes que vous souhaitez récupérer. Il est possible de sélectionner toutes les colonnes avec `SELECT *` ou de spécifier uniquement les colonnes nécessaires.

Exemple de sélection explicite

```
SELECT nom, age
FROM utilisateurs;
```

Ce type de requête permet d'optimiser les performances en ne récupérant que les colonnes nécessaires, contrairement à :

Exemple avec SELECT *

```
SELECT *
FROM utilisateurs;
```

Cette requête récupère toutes les colonnes de la table, ce qui peut être lourd si la table contient de nombreuses colonnes ou si seules certaines données sont nécessaires.

Résultat attendu pour SELECT *

ID	NOM	AGE	EMAIL
1	Alice	25	alice@example.com
2	Bob	30	bob@example.com

Bonnes pratiques

Même si `SELECT *` est pratique pour des tests rapides, privilégiez une sélection explicite dans un environnement de production pour éviter de manipuler des données inutiles.

Alias de colonnes

Un alias est un nom temporaire donné à une colonne ou une valeur calculée dans une requête. Cela permet d'améliorer la lisibilité et la compréhension des résultats.

Syntaxe

Un alias s'applique en utilisant le mot-clé `AS` ou simplement un espace après le nom ou l'expression.

```
SELECT colonne AS alias
FROM table;
```

Exemple avec alias pour une colonne

```
SELECT nom AS utilisateur, age AS âge
FROM utilisateurs;
```

Résultat attendu :

UTILISATEUR	ÂGE
Alice	25
Bob	30

Exemple avec alias pour une valeur calculée

Les alias sont particulièrement utiles pour nommer des colonnes calculées, comme les résultats des fonctions d'agrégation.

```
SELECT SUM(age) AS total_age, AVG(age) AS moyenne_age
FROM utilisateurs;
```

Résultat attendu :

TOTAL_AGE	MOYENNE_AGE
55	27.5

Exemple sans `AS`

Dans MySQL et SQLite, l'utilisation de `AS` est facultative. L'alias peut être appliqué en plaçant un espace après la colonne ou l'expression.

```
SELECT nom utilisateur, age âge
FROM utilisateurs;
```

Cependant, pour des raisons de clarté et de cohérence, l'utilisation explicite de `AS` est recommandée.

La clause WHERE

La clause `WHERE` permet de spécifier des conditions pour filtrer les lignes retournées.

Exemple

```
SELECT nom, age
FROM utilisateurs
WHERE age > 25;
```

Résultat attendu :

NOM	AGE
Bob	30

Les opérateurs de comparaison

Les opérateurs permettent de définir des conditions dans la clause `WHERE`.

OPÉRATEUR	DESCRIPTION	EXEMPLE
=	Égal	`age = 30`
!= ou <>	Différent	`age ≠ 30`
>	Supérieur	`age > 25`
<	Inférieur	`age < 30`
>=	Supérieur ou égal	`age ≥ 25`
<=	Inférieur ou égal	`age ≤ 30`

Combiner des conditions : AND et OR

`AND` : Toutes les conditions doivent être vraies.

`OR` : Une seule condition doit être vraie.

Exemple

```
SELECT nom, email
FROM utilisateurs
WHERE age > 25 AND email LIKE '%example.com';
```

Résultat attendu :

NOM	EMAIL
Bob	bob@example.com

Rechercher des modèles : LIKE

`LIKE` est utilisé pour rechercher des motifs dans des chaînes de caractères.

MODÈLE	DESCRIPTION
`%`	Remplace 0 ou plusieurs caractères.
`_`	Remplace un seul caractère.

Exemple

```
SELECT nom
FROM utilisateurs
WHERE email LIKE '%@example.com';
```

Résultat attendu :

NOM
Alice
Bob

Trier les résultats : ORDER BY

`ORDER BY` permet de trier les résultats en ordre croissant (`ASC`) ou décroissant (`DESC`).

▸ **Exemple**

```
SELECT nom, age
FROM utilisateurs
ORDER BY age DESC;
```

Résultat attendu :

NOM	AGE
Bob	30
Alice	25

Calculer des totaux : SUM, AVG, MIN, MAX, COUNT

Les fonctions d'agrégation permettent de réaliser des calculs sur des ensembles de données.

FONCTION	DESCRIPTION
`SUM`	Somme des valeurs
`AVG`	Moyenne des valeurs
`MIN`	Valeur minimale
`MAX`	Valeur maximale
`COUNT`	Nombre de lignes correspondantes

▸ **Exemple avec alias**

```
SELECT COUNT(*) AS total_utilisateurs, AVG(age) AS moyenne_age
FROM utilisateurs;
```

Résultat attendu :

TOTAL_UTILISATEURS	MOYENNE_AGE
2	27.5

Exemples de requêtes simples sur la base des élections

1. Afficher les codes et noms des États.

```
SELECT code AS code_etat, nom AS nom_etat
FROM Etats;
```

2. Trouver les États ayant plus de 10 grands électeurs.

```
SELECT code AS code_etat, nom AS nom_etat
FROM Etats
WHERE grd > 10;
```

3. Calculer la somme des grands électeurs avec un alias.

```
SELECT SUM(grd) AS total_grands_electeurs
FROM Etats;
```

4. Afficher les noms des États triés par grands électeurs décroissants.

```
SELECT nom AS nom_etat, grd AS grands_electeurs
FROM Etats
ORDER BY grd DESC;
```

5. Afficher les États où le vainqueur est défini.

```
SELECT code AS code_etat, nom AS nom_etat
FROM Etats
WHERE vainqueur IS NOT NULL;
```

6. Compter le nombre total d'États.

```
SELECT COUNT(*) AS total_etats
FROM Etats;
```

7. Trouver les États avec un vainqueur différent en 2020.

```sql
SELECT code AS code_etat, nom AS nom_etat
FROM Etats
WHERE vainqueur != vainqueur_2020;
```

8. Afficher les États ayant un champ libre défini.

```sql
SELECT code AS code_etat, nom AS nom_etat, champlibre AS description
FROM Etats
WHERE champlibre IS NOT NULL;
```

9. Calculer la moyenne des grands électeurs par État.

```sql
SELECT AVG(grd) AS moyenne_grands_electeurs
FROM Etats;
```

10. Afficher les États ayant exactement 3 grands électeurs.

```sql
SELECT code AS code_etat, nom AS nom_etat
FROM Etats
WHERE grd = 3;
```

Ces exemples montrent l'utilité des alias pour rendre les colonnes ou les résultats calculés plus lisibles dans les résultats des requêtes.

Types de données supportés par MySQL

> *Lors de la conception d'une base de données MySQL, le choix des **types de données** permet d'optimiser les performances et garantir l'intégrité des données. MySQL propose plusieurs catégories de types de données adaptées à différents usages : **numériques, chaînes de caractères, dates et heures, et autres types spécialisés**.*

1. Les Types Numériques 🔢

Les types numériques sont utilisés pour stocker des valeurs entières ou décimales.

a) Les Entiers

- `TINYINT` (1 octet) : Stocke des nombres entre -128 et 127 (ou 0 à 255 en UNSIGNED).
- `SMALLINT` (2 octets) : -32 768 à 32 767 (ou 0 à 65 535 en UNSIGNED).
- `MEDIUMINT` (3 octets) : -8 388 608 à 8 388 607.
- `INT` ou `INTEGER` (4 octets) : -2 147 483 648 à 2 147 483 647.
- `BIGINT` (8 octets) : -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807.

b) Les Décimaux et Flottants

- `DECIMAL(M, D)` ou `NUMERIC(M, D)` : Précision exacte, utile pour les montants financiers.
- `FLOAT` (4 octets) : Moins précis, stocke des nombres à virgule flottante.
- `DOUBLE` ou `REAL` (8 octets) : Plus précis que `FLOAT`.

2. Les Types de Chaînes de Caractères 🔡

Utilisés pour stocker du texte et des identifiants.

a) Les Types de Longueur Fixe

- `CHAR(N)` : Chaîne de longueur fixe (1 à 255 caractères). Plus rapide pour des valeurs courtes.

b) Les Types de Longueur Variable

- `VARCHAR(N)` : Chaîne de longueur variable (1 à 65 535 caractères). Plus efficace pour des textes de tailles variables.

c) Les Textes Longs

- `TEXT` : Peut contenir jusqu'à 65 535 caractères.
- `TINYTEXT` (255 caractères), `MEDIUMTEXT` (16 Mo), `LONGTEXT` (4 Go).

3. Les Types de Dates et Heures 🗓️

Stockent des valeurs temporelles essentielles pour la gestion des données.

- `DATE` (3 octets) : Stocke une date (YYYY-MM-DD).
- `DATETIME` (8 octets) : Date et heure combinées.
- `TIMESTAMP` (4 octets) : Stocke un timestamp UNIX, utile pour les mises à jour automatiques.
- `TIME` : Stocke une durée ou une heure (HH:MM:SS).
- `YEAR` : Stocke une année sur 1 octet.

4. Autres Types Spécialisés 🔍

- `BOOLEAN` : Alias de `TINYINT(1)`, stocke `0` (false) ou `1` (true).
- `ENUM` : Liste prédéfinie de valeurs (`ENUM('petit', 'moyen', 'grand')`).
- `SET` : Ensemble de valeurs sélectionnables (`SET('A', 'B', 'C')`).
- `BLOB` : Stocke des données binaires (`TINYBLOB` , `BLOB` , `MEDIUMBLOB` , `LONGBLOB`).

Conclusion ✅

Le choix du type de données impacte les performances et la gestion des ressources. Il est essentiel de :

- Choisir **le type le plus adapté** pour optimiser le stockage.
- Éviter les types trop larges qui gaspillent de l'espace.

- Prendre en compte l’**indexation et les performances** lors du choix des types.

En comprenant ces types, vous pourrez structurer une base de données MySQL plus efficace et performante ! 🚀



PROGRAMMATION EN PYTHON

© 2025 Mr. Lambrechts

Python

> *Python est un langage de programmation interprété, polyvalent et convivial qui a gagné en popularité au fil des ans. Créé par Guido van Rossum et publié pour la première fois en 1991, Python est devenu l'un des langages les plus utilisés dans le monde de la programmation en raison de sa simplicité, de sa lisibilité et de sa grande communauté de développeurs.*

Les avantages de Python

Simplicité et lisibilité

L'une des principales caractéristiques de Python est sa syntaxe simple et concise, qui le rend facile à apprendre et à utiliser même pour les débutants. La lisibilité du code Python est également un avantage majeur, permettant aux développeurs de comprendre rapidement le fonctionnement d'un programme.

Polyvalence

Python est un langage polyvalent qui peut être utilisé dans une grande variété de domaines, notamment le développement web, l'analyse de données, l'intelligence artificielle, l'automatisation, la robotique et bien plus encore. Sa polyvalence en fait un outil indispensable pour de nombreux développeurs et entreprises.

Grande bibliothèque standard

Python dispose d'une vaste bibliothèque standard qui offre des fonctionnalités prêtes à l'emploi pour diverses tâches de programmation. Cela permet aux développeurs d'économiser du temps en utilisant des modules et des packages déjà développés plutôt que de réinventer la roue.

Communauté active

La communauté Python est l'une des plus grandes et des plus actives dans le monde de la programmation. Cette communauté dynamique contribue au développement continu de Python en créant de nouveaux packages, en fournissant des ressources d'apprentissage et en soutenant les nouveaux arrivants.

Qui utilise Python et pourquoi ?

Développeurs web

Python est largement utilisé dans le développement web pour la création de sites web, d'applications web et de services web. Des frameworks populaires comme Django et Flask permettent de développer rapidement des applications web robustes et évolutives.

Scientifiques des données

Python est devenu le langage de choix pour de nombreux scientifiques des données en raison de ses puissantes bibliothèques d'analyse de données telles que NumPy, Pandas et Matplotlib. Ces outils facilitent l'exploration, la manipulation et la visualisation des données.

Ingénieurs en intelligence artificielle et en machine learning

Python est largement utilisé dans le domaine de l'intelligence artificielle et de l'apprentissage automatique en raison de sa facilité d'utilisation et de ses bibliothèques spécialisées telles que TensorFlow, Keras et scikit-learn. Ces bibliothèques permettent de développer et de déployer des modèles d'apprentissage automatique de manière efficace.

Développeurs de logiciels et d'applications

De nombreuses entreprises utilisent Python pour le développement de logiciels et d'applications en raison de sa polyvalence, de sa productivité et de sa capacité à s'intégrer facilement avec d'autres langages et technologies.

La popularité croissante de Python

Python connaît une popularité croissante grâce à ses nombreux avantages et à sa polyvalence. Selon divers indices de popularité, tels que l'indice TIOBE et l'enquête Stack Overflow Developer Survey, Python figure parmi les langages de programmation les plus populaires et les plus demandés dans l'industrie.

Sa popularité continue de croître en raison de son adoption par de grandes entreprises comme Google, Facebook, Amazon et Netflix, qui utilisent Python pour diverses applications et services.

En conclusion, Python est bien plus qu'un simple langage de programmation. C'est un outil puissant et polyvalent qui continue de gagner en popularité grâce à sa simplicité, sa lisibilité, sa grande communauté et sa polyvalence. Que vous soyez un débutant en programmation ou un développeur expérimenté, Python offre une multitude d'opportunités pour créer des applications innovantes et résoudre des problèmes complexes dans divers domaines.

Notes de cours

- Télécharger les notes de cours "officielles"

EXERCICES DU JOUR

- Exercices sur les variables
- Pour les courageux
- Exercices sur les conditions
- Exercices sur les boucles

Vidéos "Python au lycée"

- Premiers pas: console, calculs, fichiers .py et print (6')
- Premiers pas: variables, commentaires (8'30)
- Les 4 types de variables de base
- Si...alors...(sinon) (partie 1) structure et condition (8'50)
- Entrée au clavier: input(), int() et float() (9'44)
- Si...alors...(sinon) (partie 2) (5'20)
- Booléens, comparaisons, random() (8')
- Nombres au hasard ("aléatoires") (5'40)
- Les fonctions (Partie 1) (5'38)
- Les fonctions (Partie 2) (5'38)
- Les boucles **for** (5'38)
- Les boucles **for... in range()** (5'38)

Améliorez Vos Affichages avec le module Rich

> *Le module **Rich** est un outil puissant pour ceux qui souhaitent améliorer l'affichage de leurs programmes Python dans la console. Ce module vous permet de rendre vos sorties plus lisibles et esthétiques en ajoutant des couleurs, des tableaux, des barres de progression, des arbres de dossiers, et bien plus encore. Dans cet article, nous allons explorer quelques-unes des fonctionnalités de base que vous pouvez utiliser avec la fonction `print()`.*

Installation du Module Rich

Avant de commencer, vous devez installer le module Rich.

Dans Thonny

Allez dans le menu Outils/Gérer les paquets, cherchez "rich" et installer le paquet.

L'affichage des couleurs n'est pas pris en charge par la console de Thonny. Il faudra exécuter vos programmes dans une console système (`CTRL-T`).

Dans Python (global):

Vous pouvez l'installer facilement via `pip` :

```
pip install rich
```

Une fois installé, vous pouvez commencer à utiliser les fonctionnalités de Rich dans vos scripts Python.

Dans PyCharm (5e et 6e TTR)

Le module est facilement installable depuis le package manager.

Utilisation de Rich avec `print()`

L'un des principaux avantages de Rich est sa compatibilité avec la fonction `print()`. Contrairement à la fonction `print()` standard de Python, Rich permet de formater facilement le texte, de le colorier et de l'organiser en tableaux.

Exemple de Base : Texte Coloré

Pour commencer, voyons comment afficher du texte en couleur. Avec Rich, c'est aussi simple que d'utiliser la fonction `print()` :

```
from rich import print

print("[bold red]Ceci est un texte en rouge gras[/bold red]")
```

Dans cet exemple, le texte "Ceci est un texte en rouge gras" sera affiché en rouge et en gras. Le module Rich utilise des balises similaires à celles de HTML pour définir le formatage du texte.

Codes couleurs

Voici une liste des principales couleurs disponibles avec le module Rich, que vous pouvez utiliser pour styliser le texte dans vos sorties console :

COULEURS DE BASE

- **red** : Rouge
- **green** : Vert
- **blue** : Bleu
- **yellow** : Jaune
- **magenta** : Magenta (Rose violacé)
- **cyan** : Cyan (Bleu clair)
- **white** : Blanc
- **black** : Noir

COULEURS CLAIRES

- **bright_red** : Rouge clair
- **bright_green** : Vert clair
- **bright_blue** : Bleu clair
- **bright_yellow** : Jaune clair
- **bright_magenta** : Magenta clair
- **bright_cyan** : Cyan clair
- **bright_white** : Blanc très clair
- **grey** : Gris

NUANCES DE GRIS

- **grey0** : Noir pur
- **grey1** à **grey23** : Différentes nuances de gris allant de plus foncé à plus clair
- **grey100** : Blanc pur

AUTRES COULEURS PRÉDÉFINIES

- **orange1** : Orange vif
- **purple** : Violet
- **brown** : Marron
- **pink** : Rose
- **turquoise** : Turquoise

Ajouter des Styles au Texte

Rich vous permet également d'ajouter plusieurs styles à votre texte. Vous pouvez combiner des couleurs, du gras, de l'italique, du souligné, etc.

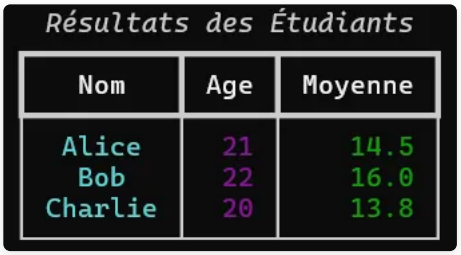
```
print("[italic blue]Texte en bleu et italique[/italic blue]")
print("[underline green]Texte en vert souligné[/underline green]")
print("[bold italic yellow]Texte en jaune gras et italique[/bold italic yellow]")
```

Ces options sont idéales pour mettre en évidence des parties spécifiques de vos sorties.

Affichage de Tableaux

L'une des fonctionnalités les plus intéressantes de Rich est la possibilité d'afficher des tableaux. Ceci est particulièrement utile pour afficher des données structurées.

Voici un exemple:




```
from rich.table import Table

table = Table(title="Résultats des Étudiants")

table.add_column("Nom", justify="right", style="cyan", no_wrap=True)
table.add_column("Age", justify="right", style="magenta")
table.add_column("Moyenne", justify="right", style="green")

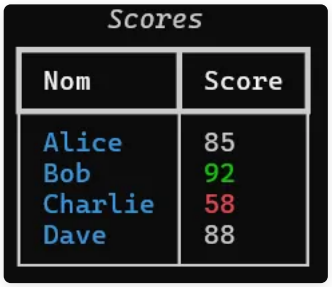
table.add_row("Alice", "21", "14.5")
table.add_row("Bob", "22", "16.0")
table.add_row("Charlie", "20", "13.8")

print(table)
```

Ce code créera un tableau stylisé avec des colonnes "Nom", "Age", et "Moyenne", chacune ayant sa propre couleur. Le tableau sera automatiquement ajusté pour être bien lisible dans la console.

Les styles conditionnels

Il est facile de personnaliser vos couleurs et styles en fonction des valeurs que afficher.



Il suffit de créer une variable `style` et de l'utiliser dans une f-string:

```python
from rich.table import Table
from rich import print

# Création d'un tableau avec du style conditionnel
table = Table(title="Scores avec Mise en Évidence")

table.add_column("Nom", style="cyan")
table.add_column("Score", style="magenta")

# Ajouter des lignes avec mise en évidence conditionnelle
scores = [("Alice", 85), ("Bob", 92), ("Charlie", 58), ("Dave", 88)]

for name, score in scores:
  if score >= 90:
    style = "bold green"
  elif score < 60:
    style = "bold red"
  else:
    style =  "white"

  table.add_row(name, f"[{style}]{score}[/{style}]")

print(table)
```

Créer des encadrés

Il est tout à fait possible d'encadrer du texte avec le module **Rich**. Pour cela, vous pouvez utiliser la classe `Panel` qui permet d'afficher du texte ou même d'autres objets Rich (comme des tableaux, des arbres, etc.) à l'intérieur d'un encadré, avec des bordures personnalisables.

Encadrement d'un texte:

Voici un exemple simple où l'on encadre du texte avec un `Panel` :



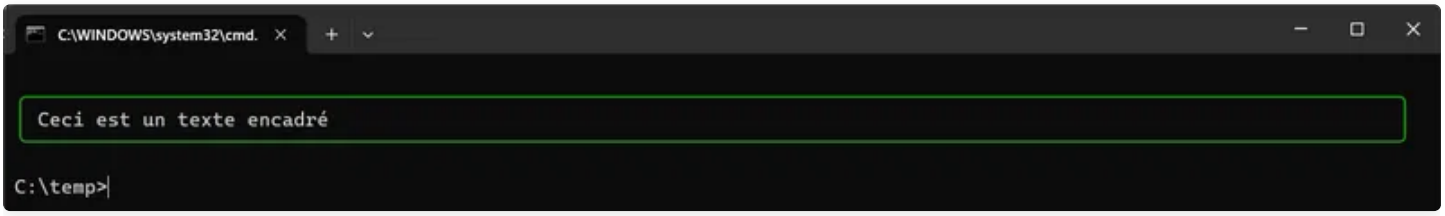

```python
from rich import print
from rich.panel import Panel

# Création d'un texte encadré
encadre = Panel("Ceci est un texte encadré")

# Affichage du texte encadré
print(encadre)
```

Changer la couleur de la bordure

Vous pouvez changer la couleur de la bordure grâce au paramètre `boder_style` :



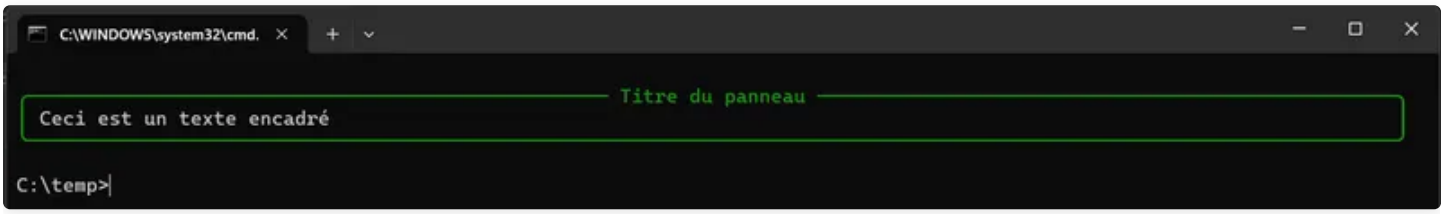
```python
from rich import print
from rich.panel import Panel

# Création d'un texte encadré avec un titre
encadre = Panel("Ceci est un texte encadré", border_style="green")

# Affichage du texte encadré
print(encadre)
```

Ajouter un titre au panel

Voici un exemple où l'on ajoute un titre au texte avec un `Panel` :




```python
from rich import print
from rich.panel import Panel

# Création d'un texte encadré avec un titre
encadre = Panel("Ceci est un texte encadré", title="Encadré", border_style="green")

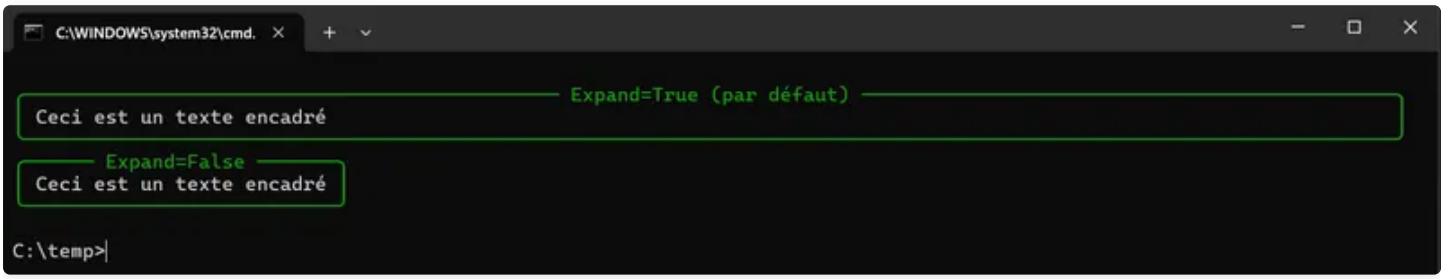
# Affichage du texte encadré
print(encadre)
```

Personnalisation du `Panel`

Le `Panel` offre plusieurs options pour personnaliser l'encadrement :

- `title` : Permet d'ajouter un titre au-dessus du panel.
- `border_style` : Définit la couleur des bordures.
- `padding` : Définit l'espacement intérieur entre le texte et les bordures du panel.
- `expand` : Si `True`, le panel occupera toute la largeur disponible de la console.
- `subtitle` : Ajoute un sous-titre en bas du panel.

Exemple avec expand




```python
from rich import print
from rich.panel import Panel

# Création d'un texte encadré
encadre = Panel(
    "Voici un texte important avec un encadrement stylisé.",
    title="Encadré Principal",
    border_style="bold red",
    expand=False
)

# Affichage du texte encadré
print(encadre)
```

Ces exemples montrent comment vous pouvez utiliser `Panel` pour encadrer du texte ou d'autres éléments, en ajoutant ainsi une touche esthétique et organisée à vos sorties dans la console. Cette fonctionnalité est particulièrement utile pour mettre en avant des informations importantes ou pour structurer des interfaces en ligne de commande plus complexes.

Utiliser des Barres de Progression

Rich facilite également l'affichage de barres de progression, ce qui est très utile pour indiquer l'avancement de certaines tâches.

```
from rich.progress import track
import time

for step in track(range(10), description="Traitement..."):
    time.sleep(0.1)
```

Ici, la fonction `track()` de Rich enveloppe une boucle pour afficher une barre de progression pendant que la boucle s'exécute. Cela permet de donner un retour visuel immédiat sur la progression d'une tâche.

Conclusion

Le module Rich est un outil formidable pour tous ceux qui cherchent à améliorer l'affichage de leurs programmes Python dans la console. En utilisant Rich avec la fonction `print()`, vous pouvez facilement ajouter des couleurs, des styles, des tableaux, des barres de progression, et bien plus encore à vos sorties. Ces fonctionnalités permettent de rendre vos programmes non seulement plus esthétiques mais aussi plus lisibles et compréhensibles.

En explorant plus en profondeur les possibilités offertes par Rich, vous pouvez améliorer significativement l'expérience utilisateur de vos scripts Python. Alors, n'hésitez pas à l'essayer dans vos prochains projets !

Ressources

Pour aller plus loin, voici quelques ressources utiles pour apprendre à utiliser le module Rich :

- Documentation officielle de Rich
- Référence rapide des balises de style
- Exemples de code avec Rich sur GitHub

Essayez ces fonctionnalités dans vos projets et vous verrez combien elles peuvent transformer l'affichage de vos scripts Python!

Exercices

Fiche personnage D&D

Vous allez recréer une version simplifiée et colorée de votre fiche personnage D&D à l'aide de Rich:

```
╔═══════════════════╗
╟ Votre personnage: ╢
╚═══════════════════╝
 - Nom: Prof
 - Race: Humain
 - Classe: Guerrier
 - Points d'XP:  450
 - Niveau:  2
 - Objet Magique: Excalibur

╔═══════════════════╗
╟ Caractéristiques: ╢
╚═══════════════════╝
 - Force:         10
 - Dextérité:     18
 - Constitution:  17
    BONUS:        +3
 - Intelligence:  6
 - Sagesse:       19
 - Charisme:      13
 - Points de vie: 13
```

1. Encadrez les titres de sections
2. Le nom de la caratéristique doit être affiché en blanc.
3. La valeur doit être affichée en jaune

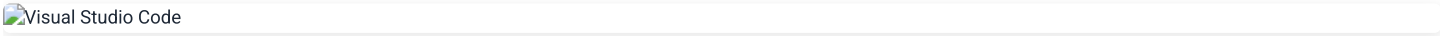
Etape 2:

1. Utilisez des couleurs différentes pour chaque race
2. Affichez les valeurs des caractéristiques <10 en rouge, les autres en vert

Python: Les outils pour développer

> *Lorsque vous vous lancez dans le développement en Python, choisir les bons outils peut faire toute la différence dans votre productivité et votre expérience de développement. Dans cet article, nous allons passer en revue certains des outils les plus populaires utilisés par les développeurs Python et discuter de leurs principales caractéristiques.*

1. Visual Studio Code (VS Code)



Visual Studio Code est un éditeur de code léger et puissant développé par Microsoft. Il offre une multitude de fonctionnalités pour les développeurs Python, notamment :

- **Prise en charge de Python intégrée :** VS Code propose une prise en charge intégrée de Python avec des fonctionnalités telles que la coloration syntaxique, la saisie semi-automatique, le débogage et la gestion des environnements virtuels.
- **Extensions Python :** Vous pouvez étendre les fonctionnalités de VS Code en installant des extensions Python, telles que Python, Pylance, ou encore Anaconda Extension Pack, qui ajoutent des fonctionnalités avancées.
- **Personnalisable :** VS Code est hautement personnalisable avec des thèmes, des raccourcis clavier et des extensions pour répondre à vos besoins spécifiques de développement Python.

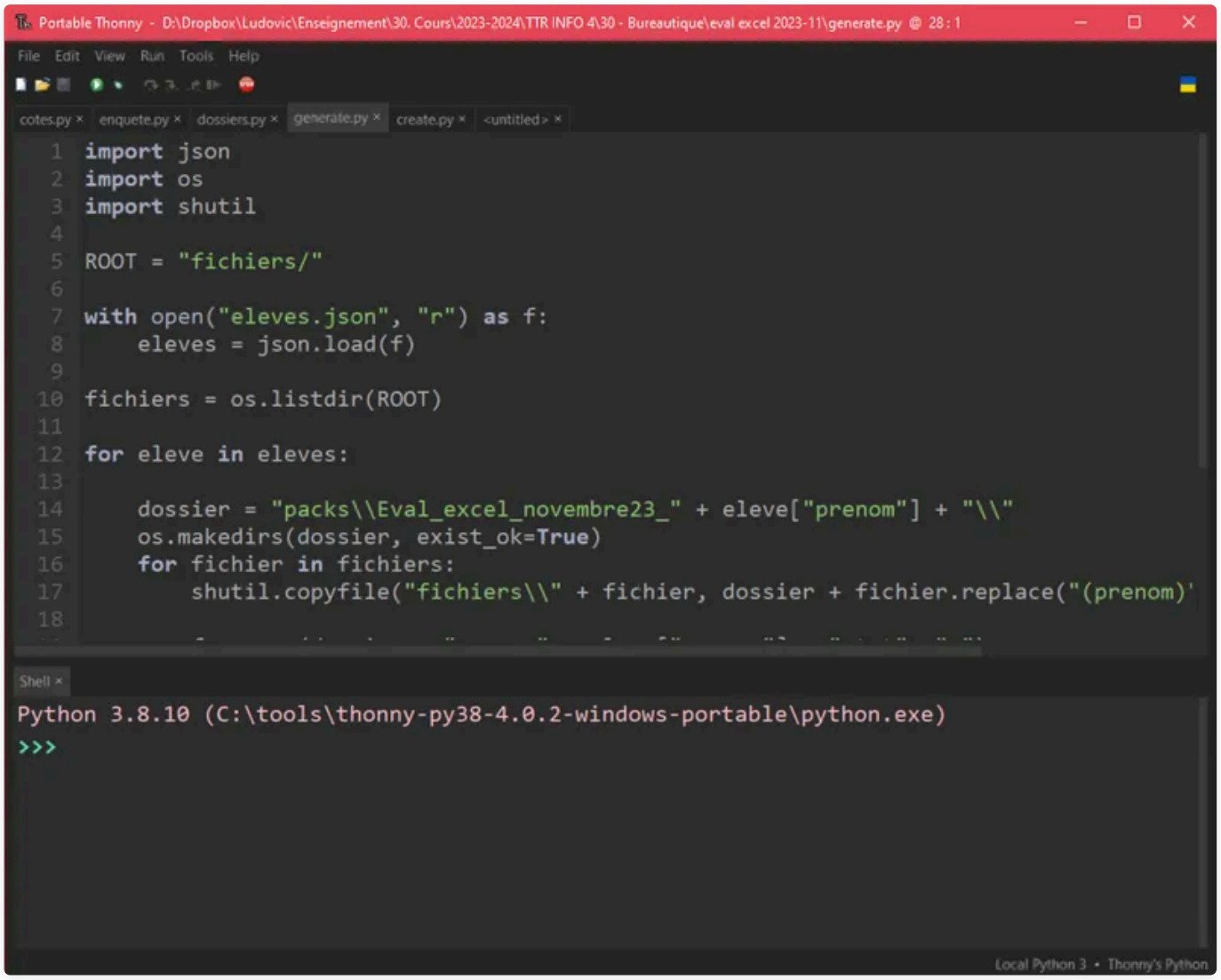
Configuration de Visual Studio Code pour Python

Pour configurer VS Code pour le développement Python, suivez ces étapes simples :

1. Installez l'**extension Python** depuis le marché des extensions de VS Code.
2. Configurez votre environnement virtuel Python en sélectionnant l'interpréteur Python souhaité dans les paramètres de l'extension.
3. Utilisez les fonctionnalités intégrées de débogage et de gestion des packages pour développer efficacement vos projets Python.

Avant de pouvoir utiliser VS Code pour développer en Python, **vous devez d'abord avoir installé Python sur votre ordinateur**.

2. Thonny Python



Thonny est un environnement de développement intégré (IDE) spécialement conçu pour les débutants en programmation Python. Ses caractéristiques principales incluent :

- **Interface utilisateur conviviale :** Thonny offre une interface utilisateur simple et intuitive qui facilite l'apprentissage de la programmation Python pour les débutants.
- **Débogage pas à pas :** Thonny propose une fonctionnalité de débogage pas à pas qui permet aux débutants de comprendre le comportement de leur code en le suivant ligne par ligne.
- **Explorateur de variables :** Thonny dispose d'un explorateur de variables qui affiche les valeurs des variables en temps réel pendant l'exécution du code.
- **Installation facile :** Thonny est facile à installer et est disponible pour Windows, macOS et Linux.

Un des avantages principaux de **Thonny**, c'est qu'il **est livré avec sa propre version de Python**. Il n'est donc pas nécessaire d'installer Python manuellement!

🔔 Pour tout savoir sur Thonny IDE et le télécharger, rendez-vous sur la page Thonny Python

3. www.online-python.com

Online Python est une plateforme en ligne qui offre un environnement de développement Python entièrement fonctionnel dans votre navigateur. Ses fonctionnalités comprennent :

- **Accès instantané :** Pas besoin d'installer quoi que ce soit. Il vous suffit d'ouvrir votre navigateur et de vous rendre sur le site pour commencer à coder en Python.
- **Prise en charge de la saisie semi-automatique :** Online Python propose une saisie semi-automatique qui facilite l'écriture de code Python.
- **Exécution de code en temps réel :** Vous pouvez exécuter votre code Python en temps réel et voir les résultats instantanément dans la sortie.
- **Partage de code :** Online Python vous permet de partager votre code avec d'autres personnes en générant simplement un lien.

Autres outils populaires pour le développement Python

En plus de Visual Studio Code, Thonny Python et www.online-python.com, il existe d'autres outils populaires utilisés par les développeurs Python pour améliorer leur flux de travail de développement. Parmi ces outils, on trouve :

PyCharm

PyCharm est un environnement de développement intégré (IDE) développé par JetBrains, offrant une gamme complète de fonctionnalités pour le développement Python, y compris un débogueur intégré, la gestion des packages, la refonte de code et bien plus encore.

Il existe 2 versions de PyCharm: la version Pro (payante) et la version Community (gratuite), donc **fais attention à quelle version tu installes**.

Jupyter Notebook

Jupyter Notebook est une application Web open source qui vous permet de créer et de partager des documents interactifs contenant du code Python, des visualisations et des explications textuelles. Il est largement utilisé pour l'analyse de données, l'apprentissage machine et la recherche scientifique.

Comment installer Python sur son ordinateur

Installer Python sur votre ordinateur est un processus simple et direct. Voici les étapes générales pour installer Python :

1. Rendez-vous sur le site officiel de Python à l'adresse https://www.python.org/downloads/ et téléchargez la dernière version de Python pour votre système d'exploitation (Windows, macOS, ou Linux).
2. Lancez le programme d'installation téléchargé et suivez les instructions à l'écran pour installer Python sur votre ordinateur. Assurez-vous de cocher l'option "Ajouter Python à PATH" lors de l'installation pour pouvoir exécuter Python depuis n'importe quel répertoire de votre système.
3. Une fois l'installation terminée, vous pouvez vérifier si Python a été correctement installé en ouvrant une fenêtre de terminal (ou d'invite de commande sur Windows) et en tapant la commande suivante :

```
python --version
```

Cette commande affichera la version de Python installée sur votre système. Si vous voyez le numéro de version de Python, cela signifie que l'installation s'est déroulée avec succès.

En suivant ces étapes simples, vous pouvez installer Python sur votre ordinateur et commencer à développer des applications en Python en un rien de temps.

Conclusion

Que vous soyez débutant ou développeur expérimenté, avoir les bons outils est essentiel pour développer en Python. Visual Studio Code, Thonny Python et www.online-python.com sont quelques-uns des outils les plus populaires et les plus utilisés dans la communauté Python. En choisissant l'outil qui convient le mieux à vos besoins et à votre style de développement, vous pouvez améliorer votre productivité et votre expérience de développement Python.

Thonny Python

> *Lorsqu'il s'agit d'apprendre à programmer en Python, choisir le bon environnement de développement peut faire toute la différence. Thonny IDE se présente comme un outil puissant et convivial, spécialement conçu pour les débutants en Python. Dans cet article, nous explorerons les avantages de Thonny, son interface utilisateur intuitive, et comment l'utiliser efficacement pour démarrer votre voyage dans le monde de la programmation Python.*

Les Avantages de Thonny IDE

Convivialité

Thonny se distingue par sa **simplicité d'utilisation**. Contrairement à certains IDE plus complexes, Thonny est conçu pour être **intuitif**, ce qui le rend **idéal pour les débutants** en programmation.

Interface Unifiée

L'interface utilisateur de Thonny est soigneusement organisée, ce qui facilite la navigation et l'accès aux différentes fonctionnalités de l'IDE. Les outils essentiels tels que l'éditeur de code, la console interactive et le débogueur sont tous intégrés de manière transparente dans **une seule fenêtre**.

Débogage Pas à Pas

Thonny offre un **débogueur intégré** qui permet aux utilisateurs d'exécuter leur code pas à pas, d'observer les valeurs des variables en cours d'exécution et de détecter les erreurs plus facilement.

Gestionnaire de Packages Intégré

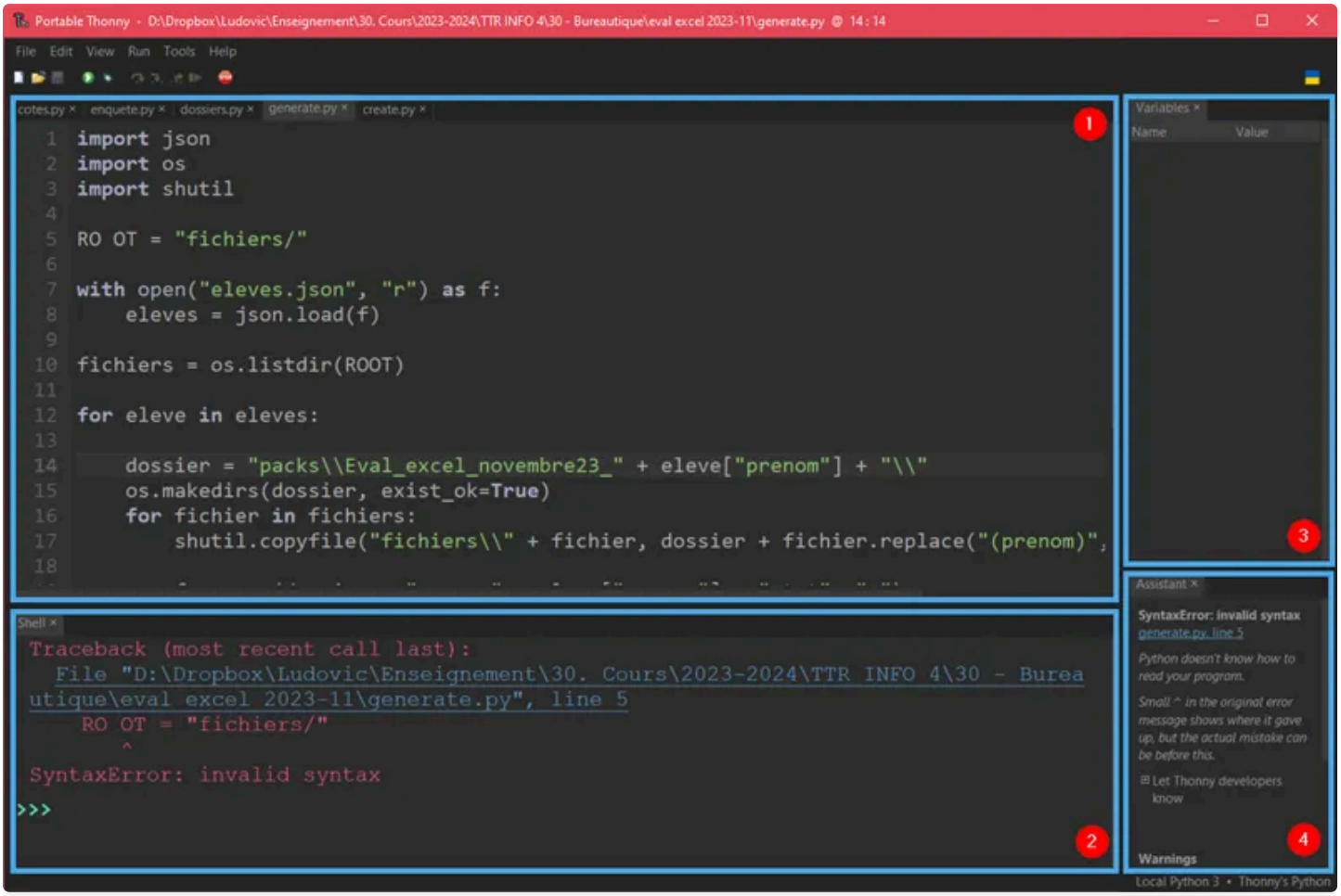
Thonny est livré avec un gestionnaire de packages/modules intégré qui facilite l'installation de modules Python supplémentaires directement depuis l'IDE, ce qui vous permet d'étendre les fonctionnalités de Python selon vos besoins.

Documentation Intégrée

Une documentation complète est disponible directement dans Thonny, ce qui vous permet d'accéder rapidement aux informations sur les fonctions et les modules Python sans avoir à quitter l'IDE.

Interface de Thonny

L'interface utilisateur de Thonny est conçue pour être simple et intuitive, ce qui permet aux utilisateurs de se concentrer sur l'essentiel : écrire du code. Voici un aperçu des principaux composants de l'interface de Thonny :



1. Éditeur de Code

C'est là où vous écrivez et modifiez votre code Python. L'éditeur de code de Thonny offre des fonctionnalités telles que la **coloration syntaxique**, l'**indentation automatique** et la suggestion de code pour améliorer votre flux de travail de programmation.

💡 La **coloration syntaxique** est une fonctionnalité qui consiste à **attribuer des couleurs différentes** aux **différents éléments d'un langage de programmation**. Cette technique vise à rendre le code source **plus lisible et compréhensible** en mettant en évidence les différents éléments syntaxiques tels que les **mots-clés**, les **variables**, les chaînes de caractères, les commentaires, etc. La coloration syntaxique **aide les programmeurs à identifier rapidement la structure et la logique du code**, ce qui facilite la détection des erreurs et améliore la productivité lors de l'écriture et de la maintenance du code.

2. Console Interactive

La console interactive vous permet d'exécuter du code Python en temps réel et d'afficher les résultats instantanément. Cela facilite le processus de test et de débogage de petits morceaux de code.

3. Explorateur de Variables

L'explorateur de variables affiche les valeurs des variables en cours d'exécution **pendant le débogage**, ce qui vous permet de surveiller l'état de votre programme à chaque étape.

4. Assistant

Thonny IDE poss§de un assistant intégré, conçu pour **aider les débutants** à programmer en Python. L'assistant propose des suggestions contextuelles, des corrections automatiques et des explications pour guider les utilisateurs tout au long de leur processus d'apprentissage. Par exemple, lorsque vous commencez à taper du code Python, l'assistant peut suggérer des complétions de code, vous permettant ainsi d'explorer les différentes options disponibles et d'apprendre de nouvelles fonctionnalités du langage. De plus, l'assistant fournit des **informations utiles sur les erreurs de syntaxe**, ce qui peut être particulièrement bénéfique pour les étudiants en programmation qui cherchent à comprendre et à corriger leurs erreurs.

5. Fenêtre de Gestion des Packages

Cette fenêtre vous permet d'installer, de mettre à jour et de supprimer des packages Python supplémentaires directement depuis l'IDE.

🔔 Pour plus d'informations sur l'installation de packages, rendez-vous sur la page "installer des modules"

Comment Utiliser Thonny pour Programmer en Python

Maintenant que nous avons exploré les avantages et l'interface de Thonny, voyons comment l'utiliser pour programmer en Python :

1. Installation de Thonny

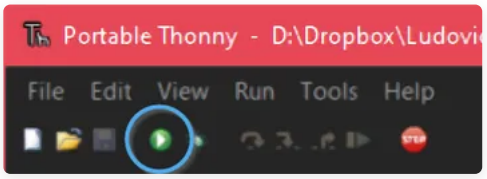
Téléchargez et installez Thonny à partir du site officiel de Thonny. Une fois l'installation terminée, lancez Thonny depuis votre menu d'applications.

2. Écrire du Code

Utilisez l'éditeur de code de Thonny pour écrire votre code Python. Vous pouvez commencer par des exemples simples et progresser à votre propre rythme.

3. Exécution du Code

Après avoir écrit votre code, vous pouvez l'exécuter en appuyant sur le bouton "Exécuter" ou en utilisant le raccourci clavier correspondant (`F5`). Les résultats s'afficheront dans la console interactive.



4. Débogage

Si vous rencontrez des erreurs, utilisez le débogueur intégré pour parcourir votre code pas à pas et identifier les problèmes éventuels.

5. Installation de Packages Additionnels

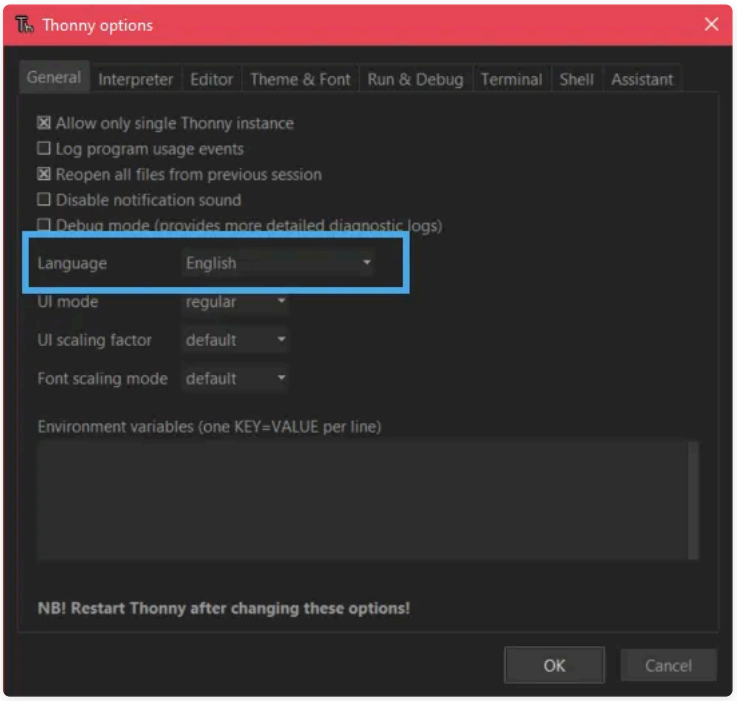
Si vous avez besoin de fonctionnalités supplémentaires, utilisez le gestionnaire de packages intégré pour installer les modules Python nécessaires à votre projet.

Configurer Thonny IDE

il est possible de configurer Thonny IDE pour qu'il corresponde à votre utilisation. Pour se faire, il faut passer par le menu "Outils (GB ***Tools***)" / "Préférences". Voici les options principales.

Changer la langue d'affichage

Pour changer de langue, rendez-vous dans l'onglet "Général" et choisissez votre langue dans la liste déroulante.



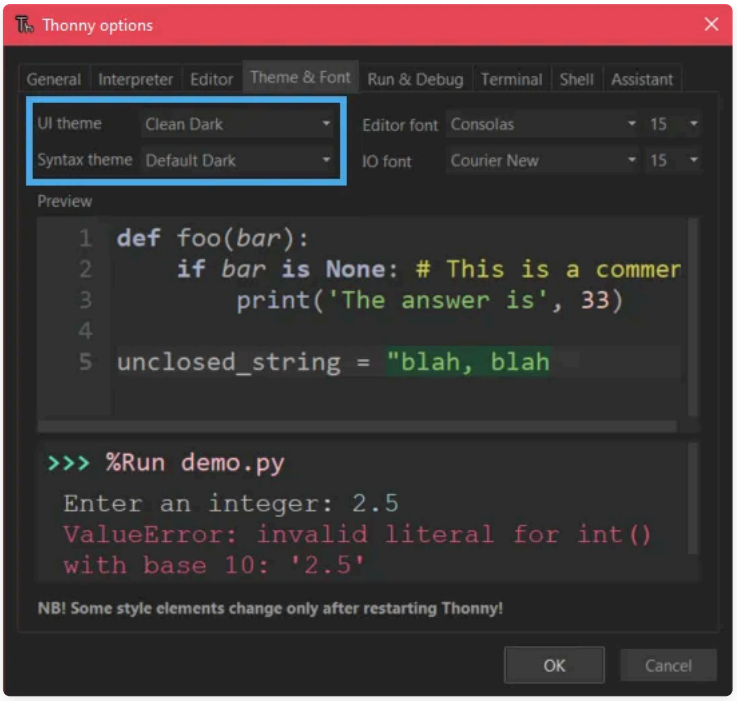
Vous devrez redémarrer Thonny pour que ce changement soit pris en compte.

Change le thème d'affichage

Thonny possède un **mode sombre**.

Le mode sombre, également connu sous le nom de "**dark mode**", offre un fond d'écran foncé avec un texte clair, ce qui **réduit la luminosité de l'écran** et **crée un contraste visuel agréable**. Les développeurs apprécient souvent le mode sombre pour son **aspect esthétique**, mais aussi pour ses avantages ergonomiques, notamment la **réduction de la fatigue oculaire** lors de longues sessions de programmation, surtout dans des environnements peu éclairés.

Pour configurer le mode sombre, rendez-vous dans l'onglet "Theme & Font", et choisissez un thème pour l'interface ET pour la syntaxe (le code écrit en python):



Bien sûr, vous pouvez choisir le thème que vous convient 😉

Conclusion

Thonny IDE est un outil exceptionnel pour les débutants en Python, offrant une expérience de développement conviviale, une interface intuitive et des fonctionnalités puissantes pour écrire, exécuter et déboguer du code Python. Que vous soyez novice en programmation ou un développeur expérimenté, Thonny peut vous accompagner tout au long de votre parcours d'apprentissage et de développement en Python. N'hésitez pas à télécharger Thonny et à commencer à explorer le monde passionnant de la programmation Python dès aujourd'hui !

Thonny Python - Déboguer son code

> *Le débogage de code est une compétence essentielle pour tout développeur informatique. Qu'il s'agisse de résoudre des erreurs de syntaxe simples ou de traquer des bogues complexes, savoir comment déboguer efficacement son code peut faire toute la différence entre un projet réussi et des heures de frustration. Dans cet article, nous allons explorer les concepts de base du débogage de code, ainsi que comment utiliser les outils de débogage disponibles dans l'environnement de développement intégré (IDE) Thonny.*

Comprendre le débogage de code

Le débogage de code consiste à **identifier**, **localiser** et **résoudre les erreurs** (ou "bugs") présentes dans un programme informatique. Ces erreurs peuvent prendre différentes formes, telles que des erreurs de syntaxe, des erreurs logiques ou des comportements inattendus du programme. Le processus de débogage implique généralement l'utilisation d'outils spécifiques, tels que des débogueurs, des impressions de messages de débogage ou des tests unitaires, pour isoler et corriger ces erreurs.

Les étapes du débogage de code

1. **Reproduire le problème**: La première étape pour déboguer un code est de reproduire le problème. Cela implique d'**identifier les conditions ou les actions** qui provoquent l'erreur ou le comportement inattendu dans le programme.
2. **Isoler la source du problème**: Une fois que le problème a été reproduit, l'étape suivante consiste à **isoler la source du problème** en identifiant les lignes de code spécifiques ou les parties du programme qui contribuent à l'erreur.
3. **Utiliser des outils de débogage**: Les outils de débogage, tels que les débogueurs intégrés aux IDE, sont **extrêmement utiles** pour examiner l'exécution du programme, surveiller les valeurs des variables et détecter les erreurs à l'exécution.
4. **Apporter des modifications et tester**: Une fois que la source du problème a été identifiée, le développeur peut apporter des modifications au code pour **corriger l'erreur**. Il est important de tester ces modifications pour s'assurer qu'elles résolvent effectivement le problème sans introduire de nouveaux bugs.

Déboguer son code dans Thonny

Thonny est un IDE convivial et puissant pour Python, qui offre plusieurs outils de débogage pour aider les développeurs à identifier et à résoudre les erreurs dans leur code.

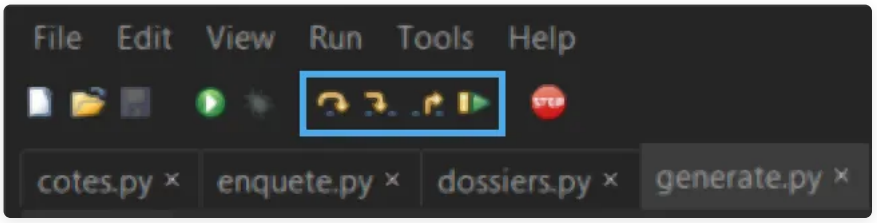
Thonny comprend un débogueur intégré qui permet aux développeurs d'exécuter **leur code pas à pas**, autrement dit **ligne par ligne**, d'**observer les valeurs des variables** à chaque étape, et de **mettre en pause l'exécution du programme** pour inspecter l'état du code.

Pour utiliser le débogueur dans Thonny, il suffit d'exécuter votre programme en **mode "debug"** grâce au bouton de la barre d'outils et/ou de placer des **points d'arrêt** à des endroits stratégiques dans le code, puis de lancer le programme en mode débogage.

Les différentes possibilités d'exécution pas à pas dans Thonny

L'exécution pas à pas, également connue sous le nom de "step-by-step execution" en anglais, est une fonctionnalité essentielle des débogueurs intégrés dans les environnements de développement intégrés (IDE) comme Thonny. Cette fonctionnalité permet aux développeurs **d'exécuter leur code ligne par ligne**, en observant et en analysant l'état du programme à chaque étape.

Ces fonctionnalités sont accessibles via la barre d'outils (et les raccourcis clavier correspondants):



MODE "STEP OVER"



Thonny offre la possibilité d'exécuter le code pas à pas en mode "**Step Over**" (étape suivante). Lorsque les développeurs utilisent cette option, **le débogueur exécute la ligne de code actuelle, puis s'arrête à la ligne suivante, sans entrer dans les appels de fonction ou de méthode**. Cela permet aux développeurs de suivre l'exécution de leur code de manière linéaire, en évitant les détails internes des fonctions ou des méthodes appelées.

MODE "STEP INTO"



Thonny propose également l'option d'exécuter le code pas à pas en mode "**Step Into**" (entrer dans). Avec cette option, le débogueur **entre dans chaque appel de fonction ou de méthode rencontré** pendant l'exécution du code, permettant ainsi aux développeurs d'inspecter le comportement interne de ces fonctions ou méthodes. Cela peut être utile pour comprendre le flux d'exécution du programme et identifier les erreurs potentielles à l'intérieur des fonctions appelées.

MODE "STEP OUT"



Enfin, Thonny offre la possibilité d'exécuter le code pas à pas en mode "**Step Out**" (sortir). Lorsque les développeurs utilisent cette option, le débogueur **exécute le reste du code de la fonction actuelle jusqu'à ce qu'il atteigne la ligne de retour ou la fin de la fonction**, puis s'arrête à la ligne suivante dans la fonction appelante. Cela permet aux développeurs de passer rapidement à la prochaine étape du processus de débogage, en évitant de parcourir chaque ligne de code à l'intérieur de la fonction actuelle.

En utilisant ces différentes options d'exécution pas à pas dans Thonny, les développeurs peuvent analyser et déboguer leur code de manière efficace et précise, ce qui contribue à la création de logiciels de haute qualité et fiables.

MODE "RESUME"



Si vous avez terminé de déboguer votre code, vous pouvez demander à Thonny de continuer à exécuter (resume) votre programme jusqu'à la fin, sans s'arrêter.

▸ Les points d'arrêts (GB *breakpoints*)

Les points d'arrêt, également appelés **breakpoints** en anglais, sont des outils essentiels dans le processus de débogage de code. Dans Thonny, les développeurs peuvent placer des points d'arrêt à des endroits spécifiques de leur code où ils souhaitent mettre en pause l'exécution du programme afin d'inspecter l'état des variables et des données.

Pour ce faire, il suffit de cliquer sur le numéro de la ligne de code correspondante. Un marqueur apparaît dans la marge:

```
7  with open("eleves.json", "r") as f:
8      eleves = json.load(f)
9
10 fichiers = os.listdir(ROOT)
11
12 for eleve in eleves:
13
14     dossier = "packs\\Eval_excel_novembre23_" + eleve["prenom"] + "\\"
15     os.makedirs(dossier, exist_ok=True)
```

Une fois que le programme atteint un point d'arrêt, l'**exécution est suspendue automatiquement** et le développeur peut examiner le contexte de l'exécution, observer les valeurs des variables et évaluer l'état du code.

Les points d'arrêt permettent aux développeurs de suivre pas à pas l'exécution de leur programme, ce qui est particulièrement utile pour identifier les erreurs et les comportements inattendus. En plaçant stratégiquement des points d'arrêt dans leur code, les développeurs peuvent **cibler spécifiquement les parties du programme qu'ils souhaitent examiner** de plus près, ce qui accélère le processus de débogage et améliore l'efficacité globale du développement logiciel.

▸ Affichage des messages de débogage

En plus du débogueur intégré, Thonny permet également aux développeurs d'insérer des messages de débogage dans leur code à l'aide de la fonction `print()`. Ces messages peuvent être utilisés pour afficher des informations sur l'exécution du programme à des points spécifiques, ce qui peut aider à identifier les erreurs ou les comportements inattendus.

Conclusion

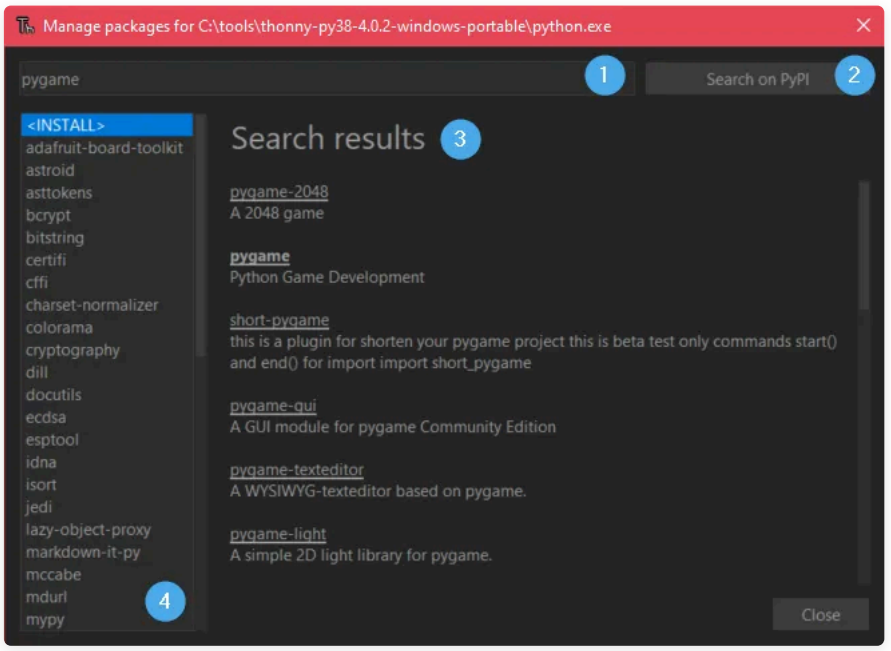
Le débogage de code est une compétence essentielle pour tout développeur informatique. En comprenant les concepts de base du débogage et en utilisant les outils appropriés, les développeurs peuvent identifier et résoudre efficacement les erreurs dans leur code, ce qui contribue à la création de logiciels de haute qualité et fiables. Avec son débogueur intégré et ses autres outils de débogage, Thonny est un excellent choix pour les développeurs Python qui cherchent à améliorer leur efficacité et leur précision lors du débogage de leur code.

Thonny Python - Installer des modules

> *L'une des fonctionnalités essentielles de Thonny est la capacité d'installer des modules Python supplémentaires pour étendre ses fonctionnalités. Dans ce guide, nous allons vous montrer étape par étape comment installer des modules Python dans Thonny, en incluant des captures d'écran pour une meilleure compréhension.*

Accéder à la gestion des packages

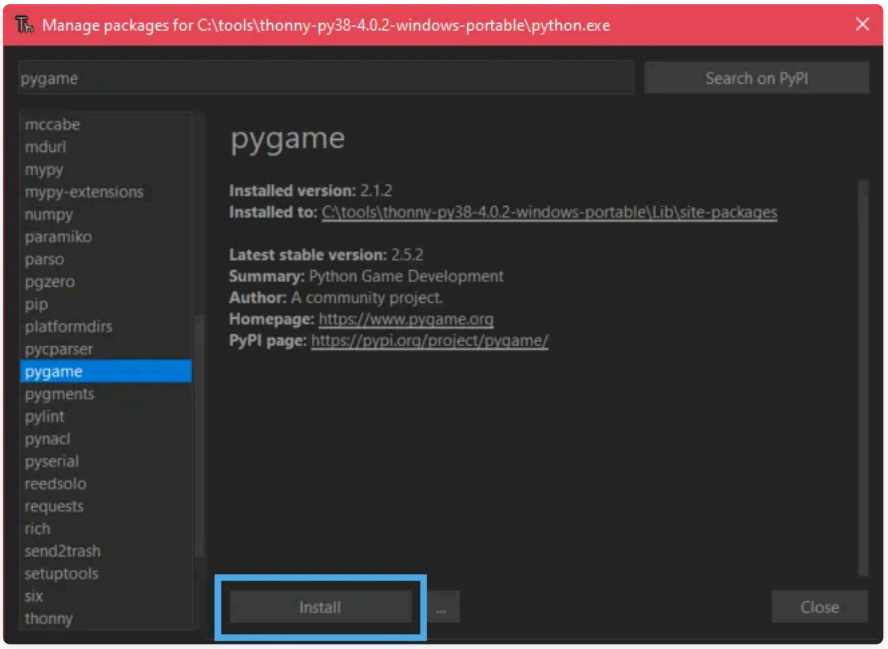
Pour accéder à la gestion des packages, cliquez sur le menu `Outils` (GB ***Tools***), puis sélectionnez `Gérer les packages ...` (GB ***Manage packages***). Cela ouvrira une nouvelle fenêtre où vous pourrez rechercher, installer et supprimer des packages Python.



Étape 3: Rechercher et installer un module Python

Dans la fenêtre de gestion des packages, vous verrez une barre de recherche en haut `[1]`. Utilisez cette barre de recherche pour trouver le module Python que vous souhaitez installer. Par exemple, recherchons le module `requests`, souvent utilisé pour effectuer des requêtes HTTP dans Python. Appuyer sur `[Entrée]` ou cliquez sur `[2]` pour valider votre recherche

Par exemple, tapez "pygame" dans la barre de recherche, puis appuyez sur la touche `Entrée` pour lancer la recherche. Une fois que le module `pygame` apparaît dans la liste des résultats, cliquez sur le nom du module `[3]` pour afficher la fenêtre d'installation.



Cliquez sur le bouton `Installer` pour commencer le processus d'installation.

Étape 4: Vérifier l'installation du module

Une fois l'installation terminée, vous verrez un message indiquant que le package a été installé avec succès. Vous pouvez maintenant fermer la fenêtre de gestion des packages.

Pour vérifier que le module a été correctement installé, vous pouvez l'importer dans votre script Python. Par exemple, créez un nouveau fichier Python dans Thonny et ajoutez la ligne suivante en haut du fichier :

```
import pygame
```

Si vous ne voyez pas d'erreurs, cela signifie que le module a été installé avec succès et que vous pouvez commencer à l'utiliser dans vos projets Python.

Conclusion

Dans ce guide, nous avons exploré comment installer des modules Python dans Thonny Python. En suivant ces étapes simples, vous pourrez étendre les fonctionnalités de Thonny et développer des projets Python plus avancés. N'hésitez pas à explorer d'autres modules disponibles et à expérimenter avec eux dans vos propres projets.

Python - Opérations mathématiques de base

> *Python est un langage de programmation polyvalent qui peut être utilisé pour effectuer diverses opérations mathématiques. Dans ce cours, nous allons explorer les opérations mathématiques de base en Python, notamment l'addition (+), la soustraction (-), la multiplication (*), la division (/), la division entière (//), le modulo (%), et l'exponentiation (**).*

Voici un article de cours sur la syntaxe de base du langage Python pour effectuer des opérations mathématiques de base.

Pour faire court: les opérations mathématiques en Python fonctionnent **comme en math**, y compris la **priorité des opérations**.

Addition (+)

L'addition est l'opération qui permet de combiner deux nombres pour obtenir leur somme. En Python, l'opérateur d'addition est le signe "+". Voici un exemple :

```
a = 5
b = 3
resultat = a + b
print(resultat)  # Affiche 8
```

Soustraction (-)

La soustraction est l'opération qui permet de trouver la différence entre deux nombres. En Python, l'opérateur de soustraction est le signe "-". Voici un exemple :

```
x = 10
y = 4
difference = x - y
print(difference)  # Affiche 6
```

Multiplication (*)

La multiplication est l'opération qui permet de trouver le produit de deux nombres. En Python, l'opérateur de multiplication est le signe "*" (astérisque). Voici un exemple :

```
p = 7
q = 2
produit = p * q
print(produit)  # Affiche 14
```

Division (/)

La division permet de trouver le quotient de deux nombres. En Python, l'opérateur de division est le signe "/" (slash). Voici un exemple :

```
dividende = 8
diviseur = 2
quotient = dividende / diviseur
print(quotient)  # Affiche 4.0
```

Division Entière (//)

La division entière permet de trouver le quotient de deux nombres en ignorant la partie décimale. En Python, l'opérateur de division entière est le signe "//". Voici un exemple :

```
numerator = 15
denominator = 4
quotient_integer = numerator // denominator
print(quotient_integer)  # Affiche 3
```

Modulo (%)

L'opérateur modulo (%) permet de trouver le reste de la division de deux nombres. Voici un exemple :

```python
n = 17
m = 5
reste = n % m
print(reste)  # Affiche 2
```

Exposants (**)

L'exponentiation permet d'élever un nombre à une puissance donnée. En Python, l'opérateur d'exponentiation est le double astérisque "**". Voici un exemple :

```python
base = 2
exposant = 3
resultat_expo = base ** exposant
print(resultat_expo)  # Affiche 8
```

Séparateur de décimales

En Belgique, nous utilisons la virugle (`,`) pour séparer la partie entière de la partie décimale d'un nombre réel (ce qu'on appelle un nombre à... virgule).

En Python, on utilise le point (`.`) pour séparer la partie entière de la partie décimale d'un nombre réel.

```python
pi = 3.1415
nombre_or = 1.61803398875
```

Priorité des opérations

Les règles de **priorité** sont les **mêmes qu'en math**: **les** `*` **et** `/` **sont calculés avant les** `+` **et les** `-`.

Combien donnera le calcul suivant?

```python
total = 5 + 2 * 3

print(total)
```

21? Réessaye 😜. La réponse et 11. Etant donné l'ordre des opérations, Python calcule d'abord `2 * 3` et l'additionne ensuite à 5.

Les parenthèses permettent de changer l'ordre de priorité.

Combien donnera le calcul suivant?

```python
total = (5 + 2) * 3

print(total)
```

11? Réessaye 😜. La réponse et 21. Etant donné l'ordre des opérations, Python calcule d'abord l'intérieur des parenthèses (`5 + 2`) et multliplie le résultat par 3.

Les paranthèses à l'intérieur des parenthèses sont exécutées en premier.

L'affectation (signe " `=` **") est toujours effectuée en dernier**. Autrement dit, Python va toujours effectuer les opérations à droite du `=` d'abord.

Conclusion

Ce cours vous a introduit aux opérations mathématiques de base en Python. Vous pouvez maintenant effectuer des opérations d'addition, de soustraction, de multiplication, de division, de division entière, de modulo, et d'exponentiation en utilisant Python. Ces concepts sont essentiels pour résoudre des problèmes mathématiques et scientifiques à l'aide de la programmation. N'hésitez pas à pratiquer ces opérations pour les maîtriser davantage.

Python: Les chaines de caractères

> *Les chaînes de caractères en Python sont utilisées pour représenter du texte et sont omniprésentes dans la programmation. Dans cet article, nous explorerons en détail ce qu'est une chaîne de caractères en Python, comment les créer et les utiliser dans vos programmes, avec des exemples concrets à chaque étape.*

Qu'est-ce qu'une Chaîne de Caractères ?

Une chaîne de caractères en Python est une séquence de caractères encadrée par des guillemets simples `' '` ou doubles `" "`. Par exemple :

```
chaine_simple = 'Bonjour'
chaine_double = "à tous"
```

Les chaînes de caractères peuvent contenir n'importe quel caractère, y compris des lettres, des chiffres, des symboles et des espaces.

Les `""` et `''` permettent de définir une chaine sur **une seule ligne**.

Création de Chaînes de Caractères

Pour créer une chaîne de caractères en Python, vous pouvez simplement l'assigner à une variable en utilisant des guillemets simples ou doubles. Voici quelques exemples :

```
chaine1 = 'Hello World'
chaine2 = "Python"
chaine3 = "12345"
```

Sur plusieurs lignes

Pour définir une chaîne de caractères sur plusieurs lignes, il faut utiliser des `""" """` (triples guillemets):

```
texte = """ Cette chaîne est définie
sur plusieurs
lignes
et
ça fonctionne
quand-même !
"""
```

Python: Opérations sur les strings

> *Dans cet article, nous explorerons en détail comment créer des chaînes de caractères, les manipuler et les utiliser dans tes programmes, avec des exemples concrets à chaque étape.*

Calculer la taille d'une chaîne de caractères

En Python, la **taille** (ou **longueur**) d'une chaîne correspond au **nombre de caractères** qu'elle contient, **espaces et signes compris**. On la calcule avec la fonction intégrée `len()`.

▸ Exemple simple

```
chaine = "Python"
taille = len(chaine)
print(taille)   # Affiche 6
```

👉 Il y a 6 caractères : `P`, `y`, `t`, `h`, `o`, `n`.

▸ Exemple avec espaces et accents

```
texte = "Bonjour à tous"
print(len(texte))  # Affiche 14
```

💡 L'espace entre les mots et le caractère accentué `à` **comptent chacun pour un caractère**. Python ne fait **aucune différence** entre lettres, chiffres, espaces ou symboles : tout caractère compte.

▸ Exemple d'utilisation pratique

```
chaine = "Python est génial"
milieu = len(chaine) // 2   # position du milieu
print(milieu)               # Affiche 8
```

▸ Ce qu'il faut retenir 🧠

- `len(chaine)` → renvoie un **entier** correspondant au **nombre de caractères**.
- Les **espaces**, **chiffres** et **ponctuations** sont comptés.
- Très utile pour :
  - découper une chaîne au **milieu** ;
  - parcourir ou valider la longueur d'un mot ;
  - vérifier la **taille d'un mot de passe**, etc.

Accès aux Caractères

Chaque caractère d'une chaîne possède un **indice** (une position). Tu peux y accéder directement avec des **crochets** `[]`.

```
chaine = "Python"

print(chaine[0])  # Affiche "P" → le premier caractère
print(chaine[3])  # Affiche "h"
```

➡️ En Python, **le premier caractère est à l'indice** `0`, et **le dernier** peut être atteint avec **l'indice** `-1`.

Indices négatifs

```
chaine = "Python"

print(chaine[-1])  # Affiche "n" → dernier caractère
print(chaine[-2])  # Affiche "o" → avant-dernier caractère
```

➡️ En Python :

- Les **indices positifs** comptent depuis le **début** (0 → premier, 1 → deuxième, ...).
- Les **indices négatifs** comptent depuis la **fin** (-1 → dernier, -2 → avant-dernier, ...).
- Si tu utilises un indice hors limites, Python renvoie une **erreur** `IndexError`.

Exemple visuel

CHAÎNE	P	Y	T	H	O	N
Indice positif	0	1	2	3	4	5
Indice négatif	-6	-5	-4	-3	-2	-1

Concaténation de Chaînes

La concaténation de chaînes consiste à combiner plusieurs chaînes en une seule. Vous pouvez le faire en utilisant l'opérateur `+` ou en utilisant la méthode `format()`. Par exemple :

```
chaine1 = "Hello"
chaine2 = "World"
chaine_concatenee = chaine1 + " " + chaine2
print(chaine_concatenee)  # Affiche "Hello World"
```

Méthodes de Chaînes de Caractères

Python offre de nombreuses méthodes intégrées pour manipuler et travailler avec des chaînes de caractères. Par exemple, vous pouvez utiliser `split()` pour diviser une chaîne en fonction d'un séparateur, `upper()` pour convertir une chaîne en majuscules, `lower()` pour convertir une chaîne en minuscules, etc.

```
chaine = "Hello World"
print(chaine.split())  # Divise la chaîne en mots (séparés par des espaces): ['Hello', 'World']
print(chaine.upper())  # Convertit la chaîne en majuscules: "HELLO WORLD"
print(chaine.lower())  # Convertit la chaîne en minuscules: "hello world"
print(chaine.capitalize())  # Convertit la chaîne en minuscules + la 1ère lettre en majuscule: "Hello world"
print(chaine.title())  # Convertit la chaîne en minuscules + la 1ère lettre de chaque mot en majuscule: "Hello World"
```

Vérifier si une chaine est entièrement en minuscules, majuscules ou ne contient que des chiffres:

```
chaine1 = "azerty"
chaine2 = "MAMAMIA"
chaine3 = "12345"
chaine4 = "zorglub123"
chaine5 = "Qwerty"

print(chaine1.islower()) # True
print(chaine2.isupper()) # True
print(chaine3.isdigit()) # True
print(chaine4.isdigit()) # False
print(chaine4.islower()) # True (toutes les lettres sont en minuscule)
print(chaine5.islower()) # False
print(chaine5.isupper()) # False

print(chaine5[0].isupper()) # True (le 1er caractère de chaine5 est une majuscule)
```

💡 Ces méthodes fonctionnent aussi bien sur des chaines, des mots ou des caractères.

Exercices

Longueur d'une Chaîne

Énoncé :
Écris un programme qui affiche la longueur de ton prénom.

Exemple d'Entrée :

```
Bonjour
```

Sortie Attendue :

```
La longueur de la chaîne est : 7
```

Majuscules et Minuscules

Énoncé :
Écris un programme qui affiche ton prénom et nom en majuscules, en minuscules et ensuite uniquement avec la première lettre de chaque mot en majuscules. /!\ Déclare d'abord une variable `nom_complet` dans laquelle tu stockes l'information.

Exemple d'Entrée :

```
Lara Clette
```

Sortie Attendue :

```
En majuscules : LARA CLETTE
En minuscules : lara clette
Finale: Lara Clette
```

Chaines de caractères: exercices

> *Voici une série d'exercices pour apprendre à manipuler les chaînes de caractères.*

> 👉 ***Aucun*** `if` ***ni boucle****, uniquement manipulation de chaînes, indexation, slicing, concaténation et méthodes de chaînes.*

Exercices – Les chaînes de caractères en Python

Objectif

Apprendre à **accéder**, **combiner** et **transformer** des chaînes de caractères avec les outils de base : indexation `[]`, opérateur `+` et méthodes (`upper()`, `lower()`, `split()`, etc.)

💡 Enregistre les fichiers sur FileGator dans le dossier `Python/Strings/Exercices opérations`.

Variables de départ

```
chaine1 = "Hello World"
chaine2 = "Python"
chaine3 = "12345"
```

Exercice 1 – Accès aux caractères

Nom de fichier: `strings_01_access.py`

1. Affiche la **première lettre** de `chaine2`.
2. Affiche la **dernière lettre** de `chaine1`.
3. Affiche le **troisième caractère** de `chaine3`.
4. Affiche la **première et la dernière lettre** de `chaine2` sur une seule ligne (ex : `P n`).
5. Affiche la **deuxième lettre de chaque chaîne** séparées par un tiret (`-`).

Exercice 2 – Concaténation

Nom de fichier: `strings_02_concatenation.py`

1. Crée une nouvelle variable `message` contenant `"Hello World from Python"`, en combinant `chaine1` et `chaine2`.
2. Ajoute un point d'exclamation à la fin du message.
3. Crée une variable `presentation` qui contient : `"Je programme en Python depuis 2025"`.
   - Utilise `chaine2` et `"2025"` dans ta concaténation.
4. Combine `chaine1`, `chaine2` et `chaine3` pour obtenir :

   ```
   Hello World - Python - 12345
   ```

   (avec des tirets et espaces bien placés)

Exercice 3 – Méthodes de chaînes

Nom de fichier: `strings_03_methodes.py`

1. Affiche `chaine1` en **majuscules**.
2. Affiche `chaine1` en **minuscules**.
3. Affiche `chaine1` avec **seulement la première lettre en majuscule**.
4. Affiche `chaine1` avec **chaque mot commençant par une majuscule**.

5. Découpe `chaine1` avec `split()` et affiche le résultat.

Exercice 4 - La Taille

Nom de fichier: `strings_04_taille.py`

Variables de départ

```
chaine1 = "Python"
chaine2 = "Bonjour à tous"
chaine3 = "12345"
chaine4 = ""
```

1. Affiche la longueur de `chaine1`.
2. A ton avis, quel sera le résultat de `len(chaine4)` ?
3. Affiche la **différence** entre la longueur de `chaine1` et celle de `chaine2`.
4. A l'aide d'un `if`, affiche la plus grande chaine entre chaine1 et chaine2.
5. Demande un mot de passe à l'utilisateur (`input`) et vérifie que sa taille est supérieure ou égale à 8. Sinon, affiche un message d'erreur

Exercice 5 – Petits défis

Nom de fichier: `strings_05_defis.py`

1. Crée une variable `mot_inverse` contenant `chaine2` écrite **à l'envers** (utilise le slicing).
2. Crée une nouvelle chaîne `"PYTHON12345"` à partir de `chaine2` et `chaine3` (en majuscules).
3. Crée une phrase complète à partir des trois chaînes :

```
"HELLO WORLD loves PYTHON 12345!"
```

Python: Découper les chaînes de caractères

> *Le découpage (*slicing*) de chaînes vous permet d'**extraire une sous-chaîne** en spécifiant une plage d'indices. Cela se fait en utilisant la notation `[début:fin]`.*

Objectif

Apprendre à **extraire une partie** d'une chaîne de caractères grâce au **slicing** (`[début:fin:pas]`).

Découpage de Chaînes

Le découpage ("***slicing***") de chaînes vous permet d'**extraire une sous-chaîne** en spécifiant une plage d'indices. Cela se fait en utilisant la notation `[début:fin]` . Par exemple :

```
chaine = "Python Programming"
sous_chaine = chaine[0:6]
print(sous_chaine)  # Affiche "Python"
```

Idées-clés: Le slicing [début:fin] exclut l'indice fin. [:n] = du début à n-1. [m:] = de m jusqu'à la fin.

Donc:

```
texte = "Python"
print(texte[0:3])   # Affiche "Pyt"
```

👉 `texte[0:3]` signifie :

> *Prendre les caractères **du 0 inclus** jusqu'au **3 exclu**.*

Syntaxe complète

```
sequence[debut:fin:pas]
```

- **debut** : index de départ (inclus).
- **fin** : index d'arrêt (exclu).
- **pas** *(optionnel)* : nombre d'éléments à sauter à chaque étape.

Exemples de base

```
chaine = "Hello World"

print(chaine[0:5])   # "Hello" (du 0 au 4)
print(chaine[:5])    # "Hello" (début implicite = 0)
print(chaine[6:])    # "World" (jusqu'à la fin)
print(chaine[:])     # "Hello World" (copie complète)
```

💡 Quand on omet `début` ou `fin` , Python prend **le début ou la fin de la chaîne par défaut**.

Utiliser un pas

Le troisième paramètre (`pas`) permet de **sauter des éléments**.

```
chaine = "123456789"

print(chaine[::2])   # "13579" (1 sur 2)
print(chaine[1::2])  # "2468"  (en partant du 2e)
```

▸ **Challenge**

Copie-colle cette chaine: `L5rEeuIusM0u EnA3A4te46z,wub ERDvh8hoDKVujOAsZVP ClcêsWwtP1He3Z4s4dh CeAl72MeEdVsg6h JtMmKI1eakAiyPWlCxOlioFeEr8uSEkrUApsvhw` et stocke-la dans une variable `message_secret`.

Essaye de l'afficher entièrement en trouvant le bon pas pour retrouver le message décodé.

Indices négatifs

Les indices négatifs comptent **depuis la fin**.

```
chaine = "Python"

print(chaine[-3:])   # "hon"   (3 derniers caractères)
print(chaine[:-3])   # "Pyt"   (tout sauf les 3 derniers)
print(chaine[-4:-1]) # "tho"   (du 4e à partir de la fin jusqu'à l'avant-dernier)
```

Inverser une chaîne

Utiliser un **pas négatif** pour lire à l'envers :

```
chaine = "Python"
print(chaine[::-1])  # "nohtyP"
```

💡 `[::-1]` = tout, mais en sens inverse.

Ce qu'il faut retenir

ÉLÉMENT	SIGNIFICATION	EXEMPLE	RÉSULTAT
`[a:b]`	de a (inclus) à b (exclu)	`"Hello"` `[1:4]`	`"ell"`
`[:b]`	du début à b (exclu)	`"Hello"` `[:4]`	`"Hell"`
`[a:]`	de a à la fin	`"Hello"` `[2:]`	`"llo"`
`[:]`	tout (copie)	`"Hello"` `[:]`	`"Hello"`
`[::2]`	un caractère sur deux	`"Hello"` `[::2]`	`"Hlo"`
`[::-1]`	inverse la chaîne	`"Hello"` `[::-1]`	`"olleH"`

Bonnes pratiques

Le **slicing ne modifie jamais** la chaîne d'origine (les chaînes sont immuables).

Utilise `len()` pour découper dynamiquement :

```
milieu = len(chaine) // 2
print(chaine[:milieu])
print(chaine[milieu:])
```

En cas d'indice trop grand, Python ne renvoie **pas d'erreur** : il s'arrête simplement à la fin.

À retenir 🧠

*Le slicing, c'est **comme découper une tranche dans une baguette** : tu choisis le point de départ, le point d'arrêt et éventuellement l'épaisseur de coupe.*

Slicing - Exercices

> *Voici une **série de 10 exercices progressifs** sur le **slicing** en Python. Les exercices vont du **plus simple au plus complexe**, sans boucles ni conditions.*

Exercices – Le slicing (`[début:fin:pas]`)

Variables de base

```
chaine = "Python est génial"
```

Exercice 1 – Extraire les 6 premières lettres

Affiche `"Python"` à partir de la chaîne donnée.

Exercice 2 – Extraire un mot précis

Affiche le mot `"est"`.

Exercice 3 – Extraire la fin de la chaîne

Affiche `"génial"` à partir de `chaine`, **sans connaître la longueur exacte**.

Exercice 4 – Enlever le premier mot

Affiche tout sauf `"Python "` (espace compris).

Exercice 5 – Prendre une partie au milieu

Affiche `"hon est gé"` à l'aide d'un slicing adapté.

Exercice 6 – Un caractère sur deux

Affiche la chaîne avec **un caractère sur deux seulement**.

(Indice : utilise le troisième paramètre du slicing)

Exercice 7 – Lecture inversée

Affiche la chaîne **à l'envers**.

(Indice : le pas peut être négatif)

Exercice 8 – Les 3 derniers caractères

Affiche uniquement les 3 derniers caractères de la chaîne.

Exercice 9 – Supprime les 2 premiers et les 2 derniers caractères

Affiche tout sauf les deux premiers et les deux derniers caractères.

Exercice 10 – Mot caché

Voici une nouvelle chaîne :

```
secret = "Lxeqs  3yew, vxouws aêktxes lxeqs mxkeillequrs"
```

1. Observe la structure.
2. Trouve un **pas régulier** pour extraire le message caché : `"Les 3e, vous êtes les meilleurs"`

Petit Défis

1. Crée une variable `milieu` contenant les **3 caractères du milieu** de `chaine1`.

Python: Rechercher dans une chaîne de caractères

> *Dans ce cours tu vas apprendre à utiliser les opérateurs **`in`** et **`not in`** pour vérifier si une **lettre**, un **mot**, ou une **sous-chaîne** est présent(e) dans une autre chaîne de caractères.*

Objectif

Apprendre à utiliser les opérateurs `in` et `not in` pour vérifier si une **lettre**, un **mot**, ou une **sous-chaîne** est présent(e) dans une autre chaîne de caractères.

Principe

L'opérateur `in` répond à une question simple :

> *Est-ce que cet élément se trouve dans cette chaîne ?*

Il renvoie une **valeur booléenne** :

- `True` → si l'élément est présent
- `False` → sinon

Exemple simple

```
mot = "Python"

print("P" in mot)     # True  → 'P' est bien dans "Python"
print("p" in mot)     # False  → 'p' est bien dans "Python"
print("y" in mot)     # True
print("z" in mot)     # False → 'z' n'y est pas
print("Py" in mot)    # True  → sous-chaîne présente
print("thon" in mot)  # True
```

💡 Le test fonctionne **aussi bien avec des caractères individuels qu'avec des mots entiers**. 💡 Python fait la différence entre majuscules et minuscules. On dit que la comparaison est **sensible à la casse** (***case sensitive***)

Exemple avec `not in`

L'opérateur inverse :

> *Est-ce que cet élément **n'est pas** dans la chaîne ?*

```
phrase = "Bonjour à tous"

print("a" not in phrase)   # False → car il y a bien un 'a'
print("z" not in phrase)   # True  → il n'y a pas de 'z'
```

Exemple – Vérifier la présence d'un mot

```
texte = "Python est un langage puissant"

if "langage" in texte:
    print("Le mot 'langage' est présent dans la phrase.")
```

Résultat :

```
Le mot 'langage' est présent dans la phrase.
```

Exemple – Vérifier si un mot contient une majuscule

```
mot = "Informatique"

if "I" in mot:
    print("Ce mot commence bien par une majuscule.")
```

À retenir 🧠

EXPRESSION	SIGNIFICATION	RÉSULTAT
`"a" in "chat"`	le caractère `'a'` est-il présent ?	✅ `True`
`"z" in "chat"`	`'z'` est-il présent ?	❌ `False`
`"ch" in "chat"`	la sous-chaîne `'ch'` existe ?	✅ `True`
`" " in "Hello World"`	y a-t-il un espace ?	✅ `True`
`"Python" not in "JavaScript"`	Python est-il absent ?	✅ `True`

Bonnes pratiques

- Les comparaisons avec `in` sont **sensibles à la casse** (`"p"` ≠ `"P"`).
- Tu peux **normaliser** tes chaînes avec `.lower()` ou `.upper()` pour éviter les erreurs :

```
mot = "Python"
print("p" in mot.lower())  # True
```

- Combine `in` avec `if`, `for`, ou même des **conditions multiples** :

```
if "@" in email and "." in email:
    print("Adresse email valide.")
```

En résumé

BUT	OUTIL	EXEMPLE
Vérifier la présence d'un caractère	`in`	`"a" in "chat"`
Vérifier l'absence d'un caractère	`not in`	`"z" not in "chat"`
Vérifier la présence d'un mot	`in`	`"Python" in phrase`

Python: Parcourir une string avec `for`

> *Dans ce cours tu vas apprendre à **parcourir** une chaîne caractère par caractère à l'aide d'une **boucle `for`**, afin d'observer, compter ou analyser son contenu.*

Objectif

Apprendre à **parcourir** une chaîne caractère par caractère à l'aide d'une **boucle** `for`, afin d'observer, compter ou analyser son contenu.

Principe

Une chaîne de caractères est une **séquence** : elle contient plusieurs éléments ordonnés (les caractères). En Python, on peut donc la parcourir **caractère par caractère** avec une boucle `for`.

```
texte = "Python"

for caractere in texte:
    print(caractere)
```

👉 Ce code affiche :

```
P
y
t
h
o
n
```

Chaque tour de boucle place dans la variable `caractere` un **caractère de la chaîne**, dans l'ordre.

Exemple – Compter le nombre de voyelles

```
mot = "programmation"
voyelles = "aeiouy"
compteur = 0

for lettre in mot:
    if lettre in voyelles:
        compteur += 1

print("Nombre de voyelles :", compteur)
```

👉 La boucle parcourt chaque caractère et le compare à la liste des voyelles. Résultat : `6`.

Exemple – Accéder aussi à l'indice (avec `enumerate()`)

Si tu veux **l'indice** (position) en plus de la lettre :

```
mot = "Python"

for index, lettre in enumerate(mot):
    print("Indice :", index, "→ Lettre :", lettre)
```

Résultat :

```
Indice : 0 → Lettre : P
Indice : 1 → Lettre : y
Indice : 2 → Lettre : t
...
```

À retenir 🧠

CE QU'IL FAUT SAVOIR	EXEMPLE
Une chaîne est une **séquence de caractères**	`"Python"`
On peut la parcourir avec `for`	`for c in chaine:`
Chaque tour de boucle renvoie **un caractère**	`print(c)`
On peut combiner avec `if` pour tester des lettres	`if c == "a": ...`
`enumerate()` permet d'obtenir aussi l'**indice**	`for i, c in enumerate(chaine):`

Bonnes pratiques

- Donne des **noms clairs** à tes variables (`caractere` , `lettre` , `mot` …).
- Évite de modifier la chaîne pendant le parcours (les chaînes sont **immuables**).
- Combine le `for` avec `if` , `in` , ou `not in` pour faire des vérifications simples.

Python: Parcourir une string: Exercices

> *Dans ce cours tu vas apprendre à **parcourir** une chaîne caractère par caractère à l'aide d'une **boucle `for`**, afin d'observer, compter ou analyser son contenu.*

Voici **15 exercices progressifs** pour apprendre à **parcourir une chaîne** avec une **boucle** `for` et à utiliser les opérateurs `in` / `not in` .

Les exercices sont gradués du simple vers le plus appliqué, sans nécessiter de bibliothèques externes.

💡 Enregistre les fichiers sur FileGator dans le dossier `Python/Strings/Exercices parcourir` .

Exercices – Parcourir une chaîne et vérifier son contenu (`for` , `in` , `not in`)

Variables de base

```
texte = "Python est génial"
mot = "programmation"
```

Exercice 1 – Afficher chaque lettre

Nom de fichier: `parcours_01_lettres.py`

Affiche chaque lettre de la variable `texte` , une par ligne.

Exercice 2 – Afficher les lettres avec un tiret

Nom de fichier: `parcours_02_tiret.py`

Affiche chaque lettre de `texte` sur la même ligne, séparée par un tiret (`-`). 👉 Exemple de sortie : `P-y-t-h-o-n- ...`

💡 Vérifie la documentation de l'instruction `print()`

Exercice 3 – Afficher les indices

Nom de fichier: `parcours_03_indices.py`

Affiche la **position (indice)** et la **lettre correspondante** pour chaque caractère de `texte` . *(Utilise `enumerate()` .)*

Exercice 4 – Compter le nombre de voyelles

Nom de fichier: `parcours_04_voyelles.py`

Parcours `mot` et compte combien de voyelles (`a, e, i, o, u, y`) il contient. Affiche le nombre total à la fin.

Exercice 5 – Compter les espaces

Nom de fichier: `parcours_05_espaces.py`

Compter combien d'espaces `" "` se trouvent dans `texte` .

Exercice 6 – Vérifier la présence d'un mot

Nom de fichier: `parcours_06_mot.py`

Vérifie si le mot `"Python"` est présent dans `texte` (utilise `in`). Affiche un message différent selon le résultat.

Exercice 7 – Vérifier la présence d'un caractère

Nom de fichier: `parcours_07_presence.py`

Demande à l'utilisateur une lettre (avec `input()`) et indique si elle se trouve dans `mot` ou non.

(Exemple : `Entrez une lettre : r` → "La lettre r est présente.")

Exercice 8 – Trouver les majuscules

Nom de fichier: `parcours_08_majuscules.py`

Affiche uniquement les **lettres majuscules** contenues dans `texte`.

(Indice : une majuscule respecte `lettre == lettre.upper()`)

Exercice 9 – Afficher uniquement les voyelles

Nom de fichier: `parcours_09_voyelles.py`

Parcours `mot` et affiche uniquement les voyelles, sans consonnes ni espaces.

Exercice 10 – Supprimer les espaces

Nom de fichier: `parcours_10_espaces.py`

Crée une nouvelle chaîne contenant `texte` sans aucun espace. *(Astuce : construire une nouvelle chaîne vide et y ajouter chaque caractère qui n'est pas un espace.)*

Exercice 11 – Vérifier si un mot contient un chiffre

Nom de fichier: `parcours_11_chiffre.py`

Teste si `"3"` est présent dans `"Python3.10"`. Affiche le résultat (`True` ou `False`).

Exercice 12 – Vérifier une adresse mail

Nom de fichier: `parcours_12_email.py`

Crée une variable :

```
email = "exemple@gmail.com"
```

Vérifie que le texte contient à la fois `"@"` **et** `"."`. Affiche `"Adresse valide"` ou `"Adresse invalide"`.

Exercice 13 – Compter un caractère spécifique

Nom de fichier: `parcours_13_compter.py`

Compte combien de fois la lettre `"a"` apparaît dans `"abracadabra"` (en utilisant une boucle `for` et non `.count()`).

Exercice 14 – Chercher une sous-chaîne dans une phrase

Nom de fichier: `parcours_14_chercher.py`

Crée une variable :

```
phrase = "Les élèves apprennent Python au lycée"
```

Vérifie si `"Python"` est dans la phrase. Affiche `"Python trouvé"` ou `"Python absent"`.

Exercice 15 – Mini défi : nettoyer un mot de passe

Nom de fichier: `parcours_nettoyer.py`

Voici une chaîne de mot de passe contenant des caractères interdits :

```
mdp = "S3cur!t#é@2025"
```

Parcours la chaîne et **affiche seulement** les caractères **alphabétiques ou numériques** (sans ponctuation). 👉 Résultat attendu : `S3curté2025`

🧩 Défis Python – Chaînes de caractères

> *Voici une série d'exercices/défis plus complets sur la manipulation des chaines de caractères.*

Voici **5 défis progressifs** pour mettre en pratique toutes les notions sur les **chaînes de caractères** : `for`, `in`, `split`, `len`, `upper`, `lower`, `replace`, etc.

Chaque défi demande un **petit programme complet**, pas seulement une ligne de code.

Idéal pour consolider la logique et la manipulation de texte!

💡 Enregistre les fichiers sur FileGator dans le dossier `Python/Strings/Les défis`.

🔹 Défi 1 – Compteur de lettres et de mots

Nom de fichier: `01_compteur.py`

▸ Objectif

Écrire un programme qui compte :

- le **nombre total de caractères** (y compris espaces),
- le **nombre de mots** dans une phrase.

▸ Exemple

```
phrase = "Python est un langage amusant"
```

Résultat attendu :

```
Caractères : 31
Mots : 5
```

▸ Pistes

- Utilise `len()` pour compter les caractères.
- Utilise `split()` pour découper les mots et `len()` sur la liste obtenue.

🔹 Défi 2 – Vérificateur de mot de passe

⚠️ Ce défi a été déplacé dans sa propre page:

🔹 Défi 3 – Analyseur de texte

Nom de fichier: `03_analyseur.py`

▸ Objectif

Demander une phrase et afficher :

- combien de **voyelles** elle contient,
- combien de **consonnes**,
- combien d'**espaces**.

▸ Exemple

```
Phrase : Python est génial
Voyelles : 5
Consonnes : 9
Espaces : 2
```

▸ Pistes

- Utilise une boucle `for` pour parcourir la phrase.
- Vérifie `if c in "aeiouyAEIOUY"`.
- Compte les espaces avec `if c == " "`.

🔹 Défi 4 – Nettoyeur d'e-mails

Nom de fichier: `04_emails.py`

▸ Objectif

Demander une adresse e-mail et afficher :

- le **nom d'utilisateur** (avant le `@`),
- le **domaine** (après le `@`),
- le tout en **minuscules**,
- et sans espaces inutiles.

▸ Exemple

```
Email :  Marie.Dupont@ECOLE.be
Utilisateur : marie.dupont
Domaine : ecole.be
```

▸ Pistes

- Commence par supprimer les espaces avec `.strip()`.
- Transforme tout en minuscules avec `.lower()`.
- Coupe la chaîne avec `.split("@")`.

🔹 Méga Défi – Trouver le mot mystère

Nom de fichier: `05_boss_final.py`

▸ Objectif

Créer un petit jeu de **devinette de mot** où l'utilisateur doit retrouver un mot caché, lettre par lettre.

▸ Consignes

1. Crée une **liste de 20 mots** d'au moins **10 lettres**, qui se ressemblent un peu (exemples : *programmation*, *progression*, *projection*, *protection*, …).
2. Demande à l'utilisateur de choisir un **niveau de difficulté** :
   - **Facile** → afficher la liste complète des mots possibles.
   - **Difficile** → ne rien afficher.
3. Sélectionne **au hasard** un mot de la liste (utilise `random.choice()`).
4. L'utilisateur propose une **lettre à la fois**.
   - Si la lettre est dans le mot, affiche un message du type : `"Oui ! La lettre 'a' est dans le mot."`
   - Sinon : `"Non, la lettre 'a' ne s'y trouve pas."`
5. Le nombre total d'essais autorisés est égal à **la moitié de la longueur du mot** (`len(mot) // 2`).
6. Quand toutes les tentatives sont terminées, demande à l'utilisateur de **deviner le mot complet**. Compare sa réponse avec le mot mystère et affiche s'il a **gagné** ou **perdu**.

Pour t'aider, voici une liste de 20 mots:

```
mots = [ "programmation", "progression", "projection", "protection", "production", "proclamation", "promotionnel", "proportionnel", "provocation", "professionnel", "prolongation", "prospection", "propagation", "prohibition", "proposition", "prononciation", "procrastiner", "problématique", "prothésiste", "prospectrice" ]
```

▸ Bonus possibles 💡

- Afficher les lettres déjà proposées.
- Montrer la progression du mot (ex. `_ r o g _ a _ _ a t _ o n`) (utilise la liste des lettres déjà proposées).
- Donner un message d'encouragement selon le score final.

🧩 Défis Python – Mot de passe

> *Voici une série d'exercices/défis plus complets sur la manipulation des chaines de caractères.*

> *A envoyer dans* `Python/Strings/Mot de passe` .

Nom de fichier: `mdp_01.py`

Sauvegarde chaque étape dans son fichier correspondant (`mdp_1.py` , `mdp_2.py` ...)

▸ Etape 1

Demande un mot de passe à l'utilisateur et stocke-le dans la variable `mdp` .

Affiche chaque caractère un par un, sur plusieurs lignes.

Exemple:

```
A
e
r
t
y

1
2
3
!
```

▸ Etape 2

Maintenant, affiche aussi si le caractère est une majuscule ou minuscule

```
A Majuscule
e Minuscule
r Minuscule
t Minuscule
y Minuscule

1
2
3
!
```

▸ Etape 3

Affiche si le caractère est un chiffre:

```
A Majuscule
e Minuscule
r Minuscule
t Minuscule
y Minuscule

1 Chiffre
2 Chiffre
3 Chiffre
!
```

▸ Etape 4

Affiche si le caractère est un espace (`==" "`)

```
A Majuscule
e Minuscule
r Minuscule
t Minuscule
y Minuscule
  Espace
1 Chiffre
2 Chiffre
3 Chiffre
!
```

Etape 5

Affiche `(autre)` dans tous les autres cas:

```
A Majuscule
e Minuscule
r Minuscule
t Minuscule
y Minuscule
  Espace
1 Chiffre
2 Chiffre
3 Chiffre
! (Autre)
```

Etape 6

Déclare les variables `majuscules`, `minuscules`, `chiffres`, `espaces` et `autres` (initialisées à zéro) et compte le nombre de cas rencontrés (`+=1`).

Affiche le résultat:

```
A Majuscule
e Minuscule
r Minuscule
t Minuscule
y Minuscule
  Espace
1 Chiffre
2 Chiffre
3 Chiffre
! (Autre)

Majuscule: 1
Minuscules: 4
Chiffres: 3
Espaces: 1
Autres: 1
```

Etape 7

Vérifie :

- s'il contient au moins une **majuscule**,
- au moins une **minuscule**,
- au moins un **chiffre**.

Exemple

```
Entrez un mot de passe : S3curite
Mot de passe correct ✅
```

Pistes

- Parcours le mot caractère par caractère avec `for`.
- Vérifie :

  ```
  if c.isupper(), if c.islower(), if c.isdigit()
  ```

ELECTRONIQUE

Les actionneurs

> *Un actionneur est un composant ou un dispositif qui convertit un signal ou une commande électrique, pneumatique ou hydraulique en une action physique ou mécanique. Les actionneurs jouent un rôle essentiel en robotique, car ils permettent aux robots d'agir sur leur environnement et d'exécuter des tâches spécifiques.*

Utilité des actionneurs

Les actionneurs sont utilisés pour **générer des mouvements**, des forces ou des **actions physiques** en réponse à des signaux de commande. Ils permettent aux robots **d'interagir activement avec leur environnement** en effectuant des tâches précises. Les actionneurs sont essentiels pour la manipulation d'objets, le déplacement, la propulsion, le mouvement des articulations, le contrôle de la précision, etc.

Les actionneurs permettent aux robots d'**exercer des forces**, de produire des mouvements linéaires ou rotatifs, d'effectuer des changements d'état ou de position, d'appliquer des pressions, de saisir des objets, d'émettre des signaux et bien plus encore. Ils sont les "**moteurs**" qui permettent aux robots de **réaliser des actions concrètes dans le monde réel**.

Différents types d'actionneurs

Il existe plusieurs types d'actionneurs utilisés en robotique. Voici quelques-uns des plus courants :

- **Actionneurs électriques** : Les actionneurs électriques, tels que les **moteurs électriques**, sont alimentés par de l'énergie électrique. Ils peuvent générer des mouvements rotatifs, linéaires ou vibratoires, et sont couramment utilisés dans **les bras robotiques**, les **roues motrices** et d'autres systèmes mécaniques.
- **Actionneurs pneumatiques** : Les actionneurs pneumatiques **utilisent de l'air comprimé** pour générer des forces et des mouvements. Les **vérins pneumatiques** sont un exemple d'actionneur pneumatique couramment utilisé pour produire des mouvements linéaires.
- **Actionneurs hydrauliques** : Les actionneurs hydrauliques **utilisent un fluide** hydraulique pour générer des forces et des mouvements. Les **vérins hydrauliques** sont fréquemment utilisés pour produire des **mouvements linéaires puissants** dans les robots industriels et d'autres applications nécessitant une force élevée.
- **Actionneurs piezoélectriques** : Les actionneurs piezoélectriques utilisent des matériaux piézoélectriques pour convertir une tension électrique en un **mouvement mécanique précis**. Ils sont utilisés dans des applications nécessitant **un contrôle très précis**, tels que les systèmes de positionnement fin et les microrobots.
- **Actionneurs électromagnétiques** : Les actionneurs électromagnétiques **utilisent des champs magnétiques** pour générer des mouvements ou des forces. Les **électroaimants** et les actionneurs à solénoïde sont des exemples courants d'actionneurs électromagnétiques.

Principaux actionneurs utilisés en robotique

- **Moteurs électriques** : Les moteurs électriques, tels que les moteurs à courant continu (DC) et les moteurs pas à pas, sont les actionneurs les plus couramment utilisés en robotique. Ils convertissent l'énergie électrique en mouvement mécanique pour produire des rotations, des translations ou des mouvements précis.
- **Relais**: Les relais sont des composants utilisés pour contrôler des circuits électriques puissants avec de faibles signaux électriques. Ils **fonctionnent comme des interupteurs**. Ils fonctionnent en utilisant une bobine électromagnétique qui active des contacts électriques pour **fermer ou ouvrir le circuit**. Les relais sont utilisés pour isoler les circuits de commande des circuits de puissance, ce qui permet de contrôler des charges élevées sans endommager les circuits de commande.
- **Vérins pneumatiques** : Les vérins pneumatiques utilisent l'air comprimé pour générer un mouvement linéaire. Ils sont souvent utilisés pour effectuer des mouvements rapides et puissants dans les applications robotiques nécessitant une force importante.
- **Vérins hydrauliques** : Les vérins hydrauliques utilisent un fluide hydraulique sous pression pour générer un mouvement linéaire puissant. Ils sont utilisés dans les applications nécessitant une force élevée et un contrôle précis.
- **Actionneurs électromagnétiques** : Les actionneurs électromagnétiques, tels que les électroaimants et les actionneurs à solénoïde, utilisent des champs magnétiques pour produire un mouvement linéaire ou rotatif. Ils sont couramment utilisés dans les mécanismes de commutation, les systèmes de verrouillage et d'autres applications.

Les 2 types de moteurs principaux

Moteurs à courant continu (DC)



Les moteurs à courant continu sont largement utilisés en robotique en raison de leur **simplicité**, de leur **coût abordable** et de leur contrôle précis de la **vitesse et de la direction**. Ces moteurs fonctionnent en utilisant un courant continu pour créer un champ magnétique qui génère un **couple de rotation**. Les moteurs DC sont disponibles dans différentes tailles et puissances, ce qui les rend polyvalents pour diverses applications robotiques, allant des petits robots autonomes aux bras robotiques industriels.

C'est ce genre de moteur que l'on utilise pour **faire tourner les roues du robot**.

Moteurs pas à pas et servomoteurs





Ces moteurs sont utilisés lorsque **la précision de positionnement est primordiale**. Ces moteurs se déplacent par incréments discrets, appelés "pas", en réponse à des signaux de commande spécifiques. Ils sont composés de bobines et d'un rotor magnétique, et fonctionnent en appliquant des impulsions électriques séquentielles aux bobines pour créer un champ magnétique rotatif.

Les moteurs pas à pas offrent un contrôle précis de la position et **peuvent maintenir une position sans consommer d'énergie** supplémentaire. Ils sont couramment utilisés dans les imprimantes 3D, les systèmes de positionnement, les robots de précision et les mécanismes nécessitant un contrôle précis de la rotation.

Il existe des différences entre les moteurs pas à pas et les servomoteurs. la principale différence réside dans la façon dont ces moteurs sont contrôlés et dans leur capacité à maintenir une position précise. **Les servomoteurs sont contrôlés en fonction de la position souhaitée**, tandis que **les moteurs pas à pas se déplacent par incréments précis** en réponse à des signaux de commande spécifiques. Les servomoteurs sont utilisés pour un positionnement précis à **un angle spécifique**, tandis que les moteurs pas à pas offrent une précision de positionnement élevée pour des **mouvements précis** et contrôlés.

Comparaison

Chaque type de moteur présente des avantages et des limites, et leur sélection dépend des exigences spécifiques de l'application robotique. Les **moteurs à courant continu** sont adaptés aux applications nécessitant un **contrôle de vitesse et de direction**, tandis que les **moteurs pas à pas** sont privilégiés pour leur **précision de positionnement**. Le choix du moteur dépendra donc des besoins spécifiques de conception, de contrôle et de précision du robot.

Conclusion

Chaque type d'actionneur a ses propres caractéristiques, avantages et limites, et est utilisé en fonction des besoins spécifiques du robot et de l'application. L'utilisation appropriée des actionneurs permet aux robots d'interagir avec le monde réel, d'exécuter des tâches et de réaliser des mouvements précis et contrôlés.

Feux de circulation

Cet article explique comment brancher et contrôler les 3 led du module 'feu de signalisation'



Pour contrôler un feu de signalisation (traffic lights) à l'aide d'une Raspberry Pi Pico en MicroPython, vous aurez besoin de trois LED (rouge, jaune, verte) et de résistances appropriées pour les connecter (le composant reçu en classe est prêt à l'emploi). Voici un exemple de code qui montre comment cela peut être réalisé :

Matériel nécessaire :

- Raspberry Pi Pico.
- Un composant "Led traffic lights"
- Câbles de connexion.

Connexion des composants :

Connectez la LED rouge à une broche GPIO (par exemple GPIO2), la LED jaune à une autre broche (par exemple GPIO3) et la LED verte à une autre broche (par exemple GPIO4).

Code MicroPython pour contrôler les LEDs :

```
import machine
import utime

# Configuration des broches pour chaque LED
led_rouge = machine.Pin(2, machine.Pin.OUT)
led_jaune = machine.Pin(3, machine.Pin.OUT)
led_verte = machine.Pin(4, machine.Pin.OUT)

def feu_rouge():
    led_rouge.on()
    led_jaune.off()
    led_verte.off()
    utime.sleep(5)  # Rouge pour 5 secondes

def feu_jaune():
    led_rouge.off()
    led_jaune.on()
    led_verte.off()
    utime.sleep(2)  # Jaune pour 2 secondes

def feu_vert():
    led_rouge.off()
    led_jaune.off()
    led_verte.on()
    utime.sleep(5)  # Vert pour 5 secondes

while True:
    feu_rouge()
    feu_jaune()
    feu_vert()
```

Explication du code :

- Les LED sont initialisées en tant que sorties de broches GPIO.
- Trois fonctions (`feu_rouge` , `feu_jaune` , `feu_vert`) sont définies pour contrôler les états des LED.
- Chaque fonction active une LED tout en éteignant les autres, simulant ainsi un feu de signalisation.
- Une boucle infinie (`while True`) alterne entre les trois états, imitant le comportement d'un feu de signalisation réel.

N'oubliez pas de tester soigneusement votre circuit et votre code dans un environnement sûr pour éviter tout court-circuit ou dommage au Raspberry Pi Pico.

Feux de circulation - Simulation

> *Dans cet exercice, tu vas reproduire le fonctionnement **réaliste** de deux feux tricolores qui gèrent une intersection simple. L'objectif est de comprendre la **logique de sécurité** (priorités, phases, temps de dégagement) et de la traduire en **programme MicroPython** sur un **Pi Pico** en pilotant des LED.*

Fonctionnement d'un vrai carrefour à deux feux

Sur une route, deux directions qui se croisent ne peuvent pas être au vert en même temps, sinon collision.

Un carrefour à deux feux suit généralement des **phases** :

- **Phase A (direction A passe)** :
  - Feu A : **vert**
  - Feu B : **rouge**
- **Transition A → B** :
  - Feu A : **orange** (prépare l'arrêt)
  - Feu B : **rouge** (reste bloqué)
- **Temps de sécurité "tout rouge" (dégagement)** :
  - Feu A : **rouge**
  - Feu B : **rouge** Permet de laisser le temps à un véhicule engagé de terminer de traverser.
- **Phase B (direction B passe)** :
  - Feu B : **vert**
  - Feu A : **rouge**
- **Transition B → A** :
  - Feu B : **orange**
  - Feu A : **rouge**
- **Tout rouge** puis on recommence.

C'est exactement ce que tu vas programmer, avec des durées réalistes (ou accélérées pour la démo).

Objectif de la simulation sur Pi Pico

Tu dois piloter **2 feux tricolores** :

- **Feu A** : 1 LED rouge, 1 LED "orange", 1 LED verte
- **Feu B** : 1 LED rouge, 1 LED "orange", 1 LED verte

Le programme devra :

- enchaîner les phases correctement,
- garantir qu'il n'y a **jamais deux verts simultanés**,
- inclure un **temps "tout rouge"** entre les changements de priorité,
- tourner en boucle.

Matériel

- 1 Raspberry Pi Pico (ou Pico W)
- 2 feux de circulation
- PC avec Thonny + firmware MicroPython

Câblage à réaliser

Chaque feux doit être branché :

Tu choisis tes GPIO, mais tu dois fournir un tableau clair dans ton code, par exemple :

- Feu A : rouge = GP2, orange = GP3, vert = GP4
- Feu B : rouge = GP6, orange = GP7, vert = GP8

(ceci est un exemple : à toi de décider, mais ton code doit correspondre à ton montage)

Initialisation

Les broches, durées, phases,... doivent être initialisées dans des variables au début du programme: évite les "nombres magiques".

Cycle imposé

Le cycle à programmer (durées modifiables mais cohérentes) :

- A vert : 6 s
- A orange : 2 s
- Tout rouge : 1 s
- B vert : 6 s
- B orange : 2 s
- Tout rouge : 1 s → boucle

Règle critique : **pendant A vert, B doit être rouge**, et inversement.

Travail demandé

Programme minimal attendu

- Un fichier `main.py`
- Initialisation des 6 GPIO en sorties
- Une boucle infinie qui enchaîne les phases
- À chaque phase, tu dois :
  - allumer uniquement les LED correctes
  - éteindre toutes les autres
  - attendre la durée prévue (`sleep`)

Sécurité logique

Ton code doit être lisible et éviter les erreurs :

- prévoir une fonction ou une procédure simple du type “mettre le feu A en vert, feu B en rouge” (si tu n’as pas encore vu les fonctions, tu peux le faire avec des blocs répétés, mais proprement)
- ne jamais laisser une LED d’une phase précédente allumée par accident

---

Extensions (bonus)

Choisis une ou plusieurs améliorations :

- Afficher la phase en console (ex : `print("A_VERT / B_ROUGE")`)
- Mode “nuit” : clignotement orange sur les deux feux (en appuyant sur un bouton)
- Bouton “piéton” : quand on appuie, déclencher au prochain cycle un arrêt plus long au rouge dans une direction
- Ajouter un compte à rebours (console) pour chaque phase

Critères de réussite

- Les LED correspondent exactement aux phases demandées
- Les transitions sont correctes (vert → orange → rouge, et temps tout rouge)
- Aucun cas où les deux feux sont verts en même temps
- Code clair, commenté, et facile à modifier (durées et broches)

Introduction aux buzzers

> *Dans cet article, nous allons poser les bases nécessaires pour comprendre ce qu'est un buzzer, à quoi il sert et comment il est utilisé avec une carte microcontrôleur comme la Raspberry Pi Pico. Cet article est un **prérequis conceptuel** : il ne cherche pas encore à produire des sons complexes, mais à clarifier les différences fondamentales entre les types de buzzers et leurs implications côté matériel et côté code.*

Qu'est-ce qu'un buzzer ?

Un buzzer est un **composant électronique de sortie** dont le rôle est de produire un son.

On le rencontre très souvent dans des systèmes embarqués :

- bips de confirmation
- alertes sonores
- signaux d'erreur
- minuteries
- interfaces homme-machine simples

Contrairement à un haut-parleur, un buzzer est conçu pour être **simple à piloter**, avec peu de composants externes.

À quoi sert un buzzer dans un projet embarqué ?

Dans un projet pédagogique ou industriel, le buzzer sert principalement à :

- donner un **retour immédiat** à l'utilisateur
- signaler un **événement important** sans écran
- attirer l'attention (alarme, avertissement)

Il est particulièrement utile lorsque :

- l'affichage est absent ou minimal
- le système doit rester simple et robuste

Les deux grandes familles de buzzers

Il existe deux types de buzzers très courants en électronique embarquée :

- le **buzzer actif**
- le **buzzer passif**

Ils se ressemblent physiquement, mais leur fonctionnement interne et leur usage en programmation sont très différents.

Le buzzer actif

Un buzzer actif contient un **oscillateur interne**.

Cela signifie qu'il est capable de produire un son **tout seul**, dès qu'il est alimenté.

Caractéristiques principales

- une seule tonalité possible
- fréquence interne fixe
- pilotage très simple
- aucun réglage de hauteur sonore

Côté code

Pour faire sonner un buzzer actif :

- on met la broche à l'état HIGH → il sonne
- on met la broche à l'état LOW → il se tait

Il se pilote donc comme une **LED**, mais au lieu de s'allumer, il émet un son.

Le buzzer passif

Un buzzer passif **ne contient aucun oscillateur**.

Il ne peut pas produire de son sans un signal électrique approprié.

Caractéristiques principales

- nécessite un signal PWM (= qui varie)
- la fréquence du signal détermine la hauteur du son
- le rapport cyclique influence le volume
- permet de produire des sons variés, des alertes et des mélodies

Côté code

Avec un buzzer passif, le microcontrôleur doit :

- générer un signal périodique
- choisir la fréquence
- gérer la durée

Cela demande plus de code, mais offre beaucoup plus de possibilités.

Différences fondamentales entre actif et passif

Buzzer actif

- autonome
- simple
- une seule note
- peu flexible

Buzzer passif

- dépend du microcontrôleur
- plus complexe
- fréquences multiples
- très flexible

Pourquoi a-t-on souvent l'impression que le buzzer actif sonne plus fort ?

C'est une impression fréquente, mais trompeuse.

Un buzzer actif :

- fonctionne directement à sa fréquence de résonance
- est souvent alimenté avec une tension stable
- est optimisé pour produire un son immédiatement audible

Un buzzer passif :

- est souvent utilisé avec des réglages prudents
- est piloté en basse tension
- dépend fortement du signal généré par le code

Un buzzer passif **peut sonner aussi fort, voire plus fort**, mais uniquement s'il est correctement piloté (fréquence, tension, duty cycle).

Choisir le bon buzzer selon le projet

Utiliser un **buzzer actif** si :

- un simple bip suffit
- le code doit rester très simple
- on veut juste un signal sonore

Utiliser un **buzzer passif** si :

- on veut contrôler la hauteur du son
- produire des alertes complexes
- jouer des mélodies

- expérimenter avec le PWM

Erreurs fréquentes

- Confondre buzzer actif et passif
- Essayer de changer la fréquence d'un buzzer actif
- Utiliser un buzzer passif sans PWM
- Conclure que le buzzer passif est "moins puissant" par nature

---

Ce qu'il faut retenir

- Un buzzer est un dispositif de sortie sonore simple
- Actif et passif ont des usages très différents
- Le type de buzzer détermine directement le **type de code**
- La complexité sonore vient du **pilotage**, pas du composant seul

---

Dans le prochain article, nous utiliserons un **buzzer actif** pour produire des bips, gérer le rythme et construire progressivement un signal de type Morse, uniquement à l'aide de durées et de pauses.

Utiliser un buzzer actif (KY-012)

> *Dans cet article, nous allons apprendre à utiliser un **buzzer actif KY-012** avec une Raspberry Pi Pico. L'objectif est de comprendre comment produire des bips sonores, gérer le rythme et utiliser le temps comme information, jusqu'à construire un **code Morse simplifié**.*

Contrairement au buzzer passif, ici **la fréquence n'est pas contrôlable** : le buzzer actif ne sait produire **qu'une seule note**.

Rappel : principe du buzzer actif

Un buzzer actif contient un oscillateur interne. Dès qu'il est alimenté, il émet un son à une fréquence fixe.

Conséquence directe

- Pas de PWM
- Pas de fréquence à choisir
- Uniquement ON / OFF

Côté code, on travaille uniquement avec :

- l'état de la broche
- la durée pendant laquelle elle reste active

Connexion du KY-012

- `S` (Signal) → GPIO15
- `-` (GND) → GND
- `+` (VCC) → 3.3 V

Contrairement au buzzer passif, le buzzer actif **doit être alimenté**.

Premier son : un bip simple

On commence par le cas le plus élémentaire : un bip, puis arrêt.

```
from machine import Pin
import utime

buzzer = Pin(15, Pin.OUT)

buzzer.value(1)     # buzzer ON
utime.sleep(1)      # 1 seconde
buzzer.value(0)     # buzzer OFF
```

À retenir

- `value(1)` alimente le buzzer
- `value(0)` coupe le son
- la durée dépend uniquement du `sleep`

Bip + pause en boucle

On introduit maintenant la répétition.

```
from machine import Pin
import utime

buzzer = Pin(15, Pin.OUT)

while True:
    buzzer.value(1)
    utime.sleep(0.5)

    buzzer.value(0)
    utime.sleep(0.5)
```

Le buzzer émet un bip régulier, comparable à un métronome.

Bip court et bip long

La différence entre un bip court et un bip long est **uniquement temporelle**.

```
from machine import Pin
import utime

buzzer = Pin(15, Pin.OUT)

while True:
    # bip court
    buzzer.value(1)
    utime.sleep(0.2)
    buzzer.value(0)
    utime.sleep(0.3)

    # bip long
    buzzer.value(1)
    utime.sleep(0.6)
    buzzer.value(0)
    utime.sleep(0.8)
```

Introduire la notion de rythme

À ce stade, on ne change jamais le son lui-même. On ne joue que sur :

- la durée du bip
- la durée du silence

Avec un buzzer actif, **le rythme est l'unique information sonore**.

C'est exactement le principe du code Morse.

Principe du Morse (version pédagogique)

Le code Morse repose sur :

- un son court
- un son long
- des pauses

Pour simplifier :

- point → bip court
- trait → bip long

Nous n'utilisons pas ici les durées normalisées strictes, mais une version **compréhensible et lisible**.

Dictionnaire Morse simplifié

On associe chaque lettre à une séquence de symboles.

```
morse = {
    'S': '...',
    'O': '---',
    'I': '..',
    'N': '-.',
    'F': '..-.'
}
```

Jouer “SOS”

```
from machine import Pin
import utime

buzzer = Pin(15, Pin.OUT)

morse = {
    'S': '...',
    'O': '---'
}

dot = 0.2
dash = 0.6
pause = 0.3

while True:
    for letter in "SOS":
        for symbol in morse[letter]:
            buzzer.value(1)
            if symbol == '.':
                utime.sleep(dot)
            else:
                utime.sleep(dash)

            buzzer.value(0)
            utime.sleep(pause)

        utime.sleep(0.6)   # pause entre lettres
```

Jouer “INFO”

```
morse = {
    'I': '..',
    'N': '-.',
    'F': '..-.',
    'O': '---'
}

message = "INFO"
```

Le reste du code est identique : seul le dictionnaire et le message changent.

Ce que cet article a introduit

- pilotage ON / OFF
- gestion du temps
- notion de rythme
- dictionnaire comme table de correspondance
- boucle imbriquée

Erreurs fréquentes

- Essayer de modifier la fréquence d'un buzzer actif
- Oublier la pause entre lettres
- Alimenter le buzzer actif sans GND commun

Ce qu'il faut retenir

- Un buzzer actif ne produit qu'un seul son
- Toute l'information passe par la durée
- Le Morse est une application directe de cette logique
- Simplicité matérielle ≠ pauvreté pédagogique

Dans le prochain article, nous utiliserons un **buzzer passif (KY-006)** pour introduire la fréquence, le volume et la génération de sons évolutifs, jusqu'à des mélodies complètes.

Utiliser un buzzer passif (KY-006)

> *Dans cet article, nous passons au **buzzer passif KY-006**, beaucoup plus flexible que le buzzer actif. Il permet de **contrôler la hauteur du son**, le volume et les variations dans le temps. C'est avec lui que l'on peut produire des **alertes évoluées** et, progressivement, de **véritables mélodies**.*

Principe du buzzer passif

Un buzzer passif **ne produit aucun son tout seul**. Il doit être piloté par un **signal PWM**.

Avec un buzzer passif :

- la **fréquence** du PWM détermine la hauteur du son
- le **duty cycle** influence le volume
- la **durée** définit le rythme

On contrôle donc **trois paramètres**, contre un seul pour le buzzer actif.

Connexion du KY-006

- `S` (Signal) → GPIO15 (PWM)
- `-` (GND) → GND
- `+` → non connecté (c'est le `S` qui sert à générer le signal dans notre cas)

Initialisation de base

```
from machine import Pin, PWM
import utime

buzzer = PWM(Pin(15))
buzzer.duty_u16(2000)
```

À ce stade, le buzzer est prêt, mais **silencieux** tant qu'aucune fréquence n'est définie.

Alerte simple sur deux tons (ambulance)

On alterne deux fréquences distinctes.

```
while True:
    buzzer.freq(600)
    utime.sleep(0.5)

    buzzer.freq(900)
    utime.sleep(0.5)
```

Alerte avec montée progressive de fréquence

```
while True:
    for freq in range(400, 1000, 20):
        buzzer.freq(freq)
        utime.sleep(0.02)
```

La fréquence augmente progressivement, ce qui donne une sensation de montée de tension.

Alerte avec descente progressive de fréquence

```
while True:
    for freq in range(1000, 400, -20):
        buzzer.freq(freq)
        utime.sleep(0.02)
```

Montée puis descente de fréquence

```
while True:
    for freq in range(400, 1000, 20):
        buzzer.freq(freq)
        utime.sleep(0.02)

    for freq in range(1000, 400, -20):
        buzzer.freq(freq)
        utime.sleep(0.02)
```

Alerte à deux tons avec durée qui diminue

La durée de chaque note diminue de **25 % à chaque itération**.

```
duration = 2.0

while duration > 0.3:
    buzzer.freq(600)
    utime.sleep(duration)

    buzzer.freq(900)
    utime.sleep(duration)

    duration = duration * 0.75
```

Diminuer aussi la pause

```
duration = 2.0
pause = 1.0

while duration > 0.3:
    buzzer.freq(600)
    utime.sleep(duration)
    utime.sleep(pause)

    buzzer.freq(900)
    utime.sleep(duration)
    utime.sleep(pause)

    duration = duration * 0.75
    pause = pause * 0.75
```

Introduction aux notes de musique

On peut associer des noms de notes à des fréquences.

```
notes = {
    'C': 262,
    'D': 294,
    'E': 330,
    'F': 349,
    'G': 392,
    'A': 440,
    'B': 494,
    'C5': 523,
    'D5': 587,
    'E5': 659,
    'F5': 698,
    'G5': 784,
    'A5': 880
}
```

Mélodie simple, rythme monotone

```
melody = ['C', 'D', 'E', 'F', 'G', 'A', 'G', 'F']

for note in melody:
    buzzer.freq(notes[note])
    utime.sleep(0.4)
```

Introduire la durée avec des tuples

Chaque élément contient :

- la note
- la durée

```
melody = [
    ('C', 0.4),
    ('D', 0.4),
    ('E', 0.6),
    ('C', 0.6)
]
```

Lecture :

```
for note, duration in melody:
    buzzer.freq(notes[note])
    utime.sleep(duration)
```

Frère Jacques (8 premières notes)

```
melody = [
    ('C', 0.4), ('D', 0.4), ('E', 0.4), ('C', 0.4),
    ('C', 0.4), ('D', 0.4), ('E', 0.4), ('C', 0.4)
]

for note, duration in melody:
    buzzer.freq(notes[note])
    utime.sleep(duration)
```

Joyeux anniversaire (6 premières notes)

```
melody = [
    ('G', 0.4), ('G', 0.2),
    ('A', 0.6),
    ('G', 0.6),
    ('C5', 0.6),
    ('B', 1.0)
]

for note, duration in melody:
    buzzer.freq(notes[note])
    utime.sleep(duration)
```

Le finale

```
melody = [
    ('A4', 0.3), ('B4', 0.3), ('D5', 0.4), ('B4', 0.4),
    ('F5', 0.6), ('F5', 0.6),
    ('E5', 0.8)
]

notes.update({
    'A4': 440,
    'B4': 494,
    'D5': 587,
    'E5': 659,
    'F5': 698
})

for note, duration in melody:
    buzzer.freq(notes[note])
    utime.sleep(duration)
```

Erreurs fréquentes

- oublier d'utiliser le PWM
- confondre fréquence et durée
- jouer trop fort dès le début
- croire qu'une mélodie est autre chose qu'une suite de fréquences et de temps

---

Ce qu'il faut retenir

- le buzzer passif est entièrement piloté par le code
- fréquence = hauteur du son
- durée = rythme
- les mélodies sont des données, pas de la magie
- une structure simple permet des résultats complexes

Avec ces trois articles, les élèves disposent désormais :

- d'une vision claire des buzzers
- d'une progression logique
- d'un terrain idéal pour introduire abstraction, fonctions… et surprises sonores ultérieures

Les capteurs

> *Un capteur est un dispositif qui **détecte et mesure une grandeur physique ou une caractéristique de son environnement**, puis **convertit** cette information en un signal utilisable par un système électronique. Les capteurs jouent un rôle essentiel en robotique, car **ils permettent aux robots de percevoir** et d'interagir avec leur environnement.*

Utilité des capteurs

Les capteurs sont utilisés pour fournir aux robots des informations sur le monde qui les entoure. Ils permettent au robot de **recueillir des données sur des grandeurs physiques** telles que la lumière, le son, la température, la pression, la distance, etc. Ces informations sont ensuite utilisées **pour prendre des décisions**, effectuer des actions et interagir avec l'environnement de manière intelligente.

Les capteurs permettent aux robots de **détecter des objets**, d'analyser des caractéristiques de l'environnement, de suivre des lignes, d'éviter des obstacles, de mesurer des paramètres, de percevoir des mouvements, de reconnaître des formes et de nombreux autres aspects nécessaires à leur fonctionnement autonome.

Différents types de capteurs

On distingue généralement deux types principaux de capteurs en fonction de la manière dont ils fournissent des informations : les **capteurs analogiques** et les **capteurs numériques**.

- **Capteurs analogiques** : Les capteurs analogiques fournissent une sortie continue qui varie en fonction de la grandeur physique mesurée. Ils produisent **une tension ou un courant proportionnel à la grandeur mesurée**. Les valeurs de sortie varient généralement de manière linéaire ou non linéaire en fonction de la grandeur mesurée. Les signaux analogiques nécessitent souvent une conversion en signaux numériques pour être traités par un système électronique.
- **Capteurs numériques** : Les capteurs numériques fournissent une sortie discrète, généralement sous forme de bits (0 et 1) ou de valeurs numériques prédéfinies. Ils utilisent des composants électroniques internes pour convertir la grandeur physique en signaux numériques. Les capteurs numériques sont souvent plus précis, plus faciles à utiliser et peuvent être directement connectés à des microcontrôleurs ou à d'autres systèmes électroniques.

Il existe une grande variété de capteurs disponibles, chacun étant conçu pour mesurer une grandeur spécifique. Parmi les autres types de capteurs couramment utilisés en robotique, on peut citer les capteurs de mouvement, les capteurs de proximité, les capteurs de gaz, les capteurs de pression, les capteurs de température, les capteurs de lumière, les capteurs de force, les capteurs de couleur, les capteurs de son, les capteurs d'inclinaison, etc.

L'utilisation appropriée des capteurs permet aux robots de percevoir leur environnement, d'interagir de manière autonome et d'exécuter des tâches spécifiques en fonction des informations recueillies.

Principaux capteurs utilisés en robotique

Voici une liste des principaux capteurs utilisés en robotique, accompagnée d'une brève description de chacun :

Capteur de distance ultrasonique : Ce capteur utilise des ondes sonores pour mesurer la distance entre le robot et un objet environnant. Il est largement utilisé pour l'évitement d'obstacles et la navigation.

En classe, nous avons utilisé le capteur ultra-son HC-SR04, qui est un modèle très répandu et bon marché.

Capteur infrarouge : Les capteurs infrarouges détectent les rayonnements infrarouges émis par les objets. Ils sont utilisés pour la détection d'obstacles, le suivi de lignes, la détection de mouvement, etc.

Capteur de lumière : Les capteurs de lumière mesurent l'intensité de la lumière ambiante. Ils peuvent être utilisés pour détecter la luminosité de l'environnement et ajuster le comportement du robot en conséquence.

Capteur de température : Ce capteur mesure la température de l'environnement. Il est utilisé dans diverses applications telles que la surveillance de la température, la régulation thermique et le contrôle climatique.

En classe, nous avons utilisé le capteur de température et d'humidité DHT11, qui est un modèle très répandu et bon marché.

Capteur de pression : Les capteurs de pression mesurent la force appliquée sur une surface. Ils sont utilisés dans les applications de contrôle de la force, de détection de contact et de mesure de la pression atmosphérique.

Capteur de mouvement : Ces capteurs détectent les mouvements, les inclinaisons ou les rotations du robot. Ils sont couramment utilisés pour la détection de mouvement, la stabilisation et le contrôle de l'orientation.

Capteur de son : Les capteurs de son détectent les variations de pression acoustique dans l'environnement. Ils peuvent être utilisés pour détecter des sons spécifiques, effectuer une reconnaissance vocale ou analyser l'acoustique de l'environnement.

Capteur de couleur : Ces capteurs permettent de détecter les couleurs des objets. Ils sont utilisés dans les applications de tri, d'identification d'objets et de suivi de lignes basé sur la couleur.

Capteur de proximité : Les capteurs de proximité détectent la présence d'objets à proximité sans contact physique direct. Ils sont utilisés pour la détection d'obstacles, la détection de présence et l'activation de fonctions basées sur la proximité.

Capteur de gaz : Ces capteurs détectent la présence et la concentration de gaz dans l'environnement. Ils sont utilisés dans les applications de surveillance de la qualité de l'air, de détection de fuites de gaz, etc.

Oui, il existe d'autres capteurs utilisés en robotique. Voici quelques exemples supplémentaires :

Capteur d'inclinaison : Ces capteurs mesurent l'inclinaison ou l'orientation du robot par rapport à la gravité. Ils sont utilisés pour la stabilisation, la détection de l'attitude et le contrôle de l'équilibre.

Capteur de force : Les capteurs de force mesurent la force appliquée sur une surface ou un objet. Ils sont utilisés dans les applications de manipulation d'objets, de contrôle de la pression et de rétroaction haptique.

Capteur d'humidité : Ces capteurs mesurent le niveau d'humidité dans l'air ou dans un matériau. Ils sont utilisés dans les applications de contrôle de l'humidité, d'irrigation intelligente et de surveillance environnementale.

Capteur de gaz toxique : Ces capteurs spécifiques détectent la présence de gaz toxiques ou dangereux dans l'environnement. Ils sont utilisés dans les applications de sécurité industrielle, de surveillance environnementale et de détection de fuites.

Capteur de mouvement oculaire : Ces capteurs utilisent la technologie de suivi oculaire pour détecter et mesurer les mouvements des yeux. Ils sont utilisés dans les interfaces homme-machine avancées et les applications de réalité virtuelle.

Capteur d'odeur : Les capteurs d'odeur, également appelés capteurs de gaz odorants, peuvent détecter et analyser différentes odeurs. Ils sont utilisés dans des applications telles que l'analyse de la qualité de l'air, la détection d'odeurs spécifiques et le contrôle des émissions odorantes.

Capteur de vitesse : Ces capteurs mesurent la vitesse de déplacement du robot. Ils sont utilisés pour le contrôle de la vitesse, la détection de mouvement rapide et le suivi de la trajectoire.

Capteur de vision : Les capteurs de vision, tels que les caméras haute résolution, sont utilisés pour capturer des images et des vidéos de l'environnement du robot. Ils sont essentiels pour la perception visuelle, la détection d'objets, la reconnaissance faciale, etc.

Chaque capteur a des caractéristiques spécifiques et est utilisé pour des applications particulières en fonction des besoins du robot et de son environnement. L'utilisation de différents capteurs permet d'obtenir une perception complète et précise du monde extérieur, ce qui est essentiel pour des opérations robotiques avancées. Ces capteurs sont largement utilisés en robotique et offrent une multitude de possibilités pour la perception et l'interaction du robot avec son environnement. En fonction des besoins spécifiques de chaque projet, il est possible d'utiliser différents capteurs pour obtenir les informations nécessaires et permettre au robot d'interagir de manière intelligente avec le monde qui l'entoure.

DHT11: Simulation du Capteur

> *Tu vas apprendre à utiliser un **objet Python** qui simule un capteur de température et d'humidité (le DHT11). Cela te permettra d'écrire du code **exactement comme si tu avais un vrai capteur branché** sur un Raspberry Pi Pico.*

Utiliser la classe `DHT11` en Python

Tu dois d'abord télécharher le fichier Python ci-joint. Ensuite, crée un nouveau fichier python dans Thonny que tu appelles "simulation_dht11.py" et sauvegarde-le dans ton dossier "Téléchargements".

Étape 1 : importer le module

Quand on veut utiliser une classe qui se trouve dans un autre fichier Python, on doit **l'importer**.

👉 Imaginons que ton fichier avec la classe `DHT11` s'appelle `dht11.py`. Alors, tu dois écrire en haut de ton programme :

```
from dht11 import DHT11
```

Étape 2 : créer un objet capteur

Maintenant, tu peux **créer un capteur simulé**. Il faut lui donner le numéro de la broche (pin) sur laquelle il est censé être branché :

```
capteur = DHT11(pin=15)
```

Même si la simulation n'utilise pas vraiment la broche, on garde ce paramètre pour ressembler au vrai capteur.

Étape 3 : demander une mesure

Avant de lire la température ou l'humidité, tu dois **lancer une mesure** :

```
capteur.measure()
```

C'est comme si tu envoyais l'ordre au vrai capteur de "prendre une photo" de l'air ambiant.

Étape 4 : lire les valeurs

Une fois la mesure faite, tu peux demander :

```
print("Température :", capteur.temperature(), "°C")
print("Humidité :", capteur.humidity(), "%")
```

Chaque appel va te donner la **dernière valeur mesurée**.

Exemple complet

```
from dht11 import DHT11

# Création d'un capteur simulé
capteur = DHT11(pin=15)

# Faire 3 mesures
for i in range(3):
    capteur.measure()  # on lance une nouvelle mesure
    print("Température :", capteur.temperature(), "°C")
    print("Humidité :", capteur.humidity(), "%")
    print("---")
```

À retenir

1. On **importe** la classe depuis son fichier.
2. On **crée un objet** `DHT11`.
3. On appelle `measure()` pour lancer une mesure.
4. On récupère les valeurs avec `temperature()` et `humidity()`.

👉 Quand tu passeras au vrai Raspberry Pi Pico, tu n'auras presque rien à changer : seule la ligne `from dht11 import DHT11` sera remplacée par l'import du vrai module.

Veux-tu que je t'écrive aussi la **version avec le vrai module `dht` de MicroPython** pour montrer le passage simulation → réel ?

DHT11: Capteur de température et d'humidité

> *Cet article présente le DHT11, un capteur de température et d'humidité très répandu et facile à utiliser.*

- Le capteur DHT11 est un capteur d'humidité et de température numérique économique et couramment utilisé.
- Il fournit des lectures de température et d'humidité relativement précises à partir d'une seule broche de données.



Caractéristiques

Le capteur DHT11 est un capteur de température et d'humidité très couramment utilisé dans de nombreuses applications. Voici quelques-unes de ses caractéristiques principales :

1. **Mesure de la température et de l'humidité** : Le capteur DHT11 permet de mesurer à la fois la température et l'humidité relative de l'environnement dans lequel il est placé.
2. **Précision** : Le capteur DHT11 offre une **précision de ±2°C** pour la mesure de la température et de **±5%** pour la mesure de l'humidité relative.
3. **Plage de mesure** : La plage de mesure de la température du capteur DHT11 est généralement de **0°C à 50°C**, tandis que la plage de mesure de l'humidité relative est de **20% à 90%**.
4. **Alimentation** : Le capteur DHT11 fonctionne avec une tension d'alimentation de **3,3V à 5V**, ce qui le rend compatible avec de nombreux microcontrôleurs et cartes de développement.
5. **Sortie numérique** : Le capteur DHT11 utilise une **sortie numérique** pour transmettre les données de température et d'humidité. Il envoie les informations sous forme de signal série numérique.
6. **Temps de réponse** : Le capteur DHT11 a un **temps de réponse relativement lent** par rapport à d'autres capteurs de température et d'humidité. Il peut prendre plusieurs secondes pour effectuer une mesure et transmettre les données.
7. **Connectivité** : Le capteur DHT11 utilise un protocole de communication à **un seul fil**, ce qui le rend **facile à intégrer** dans des projets électroniques et des systèmes embarqués.

Il convient de noter que le capteur DHT11 est un **capteur d'entrée de gamme** largement utilisé pour des applications simples de mesure de température et d'humidité. Si une précision plus élevée ou une plus grande plage de mesure est requise, il existe d'autres capteurs plus avancés disponibles sur le marché (tels que le DHT22 ou le BME280)

DHT22: alternative plus précise

Parmi les modèles plus "haut de gamme", on citera par exemple le **DHT22**. Également connu sous le nom d'AM2302, il offre une **meilleure précision** et une **plus large plage de mesure** par rapport au DHT11. Il peut mesurer la température avec une précision de **±0,5°C** et l'humidité avec une précision de ±2%. De plus, le DHT22 a une plage de mesure de température de **-40°C à 80°C** et une plage d'humidité de **0% à 100%**.

Configuration matérielle

- Connectez la broche de données du capteur DHT11 au GPIO du Raspberry Pi Pico (par exemple, GPIO2).
- Alimentez le capteur DHT11 en connectant le fil rouge au 3,3 V du Pico et le fil noir à la masse (GND).

Installation de la bibliothèque DHT11

- Téléchargez la bibliothèque dht11 pour MicroPython à partir du dépôt GitHub officiel (https://github.com/mcauser/micropython-dht11).
- Copiez le fichier dht11.py sur votre Raspberry Pi Pico.

Exemple de code pour la lecture des données du capteur DHT11

```
import machine
import dht
import time

# Initialize the DHT sensor.
capteur = dht.DHT11(machine.Pin(2))

while True:
    capteur.measure()  # Trigger a measurement.
    print("Température:", capteur.temperature())
    print("Humidité:", capteur.humidity())
    time.sleep(2)
```

Explication du code

Bien sûr, voici une explication pour ce code :

- `import machine, import dht, import time` : Ces trois instructions importent les modules nécessaires. `machine` est un module qui permet d'interagir avec le matériel de l'appareil, `dht` est un module pour travailler avec le capteur de température et d'humidité DHT11, et `time` est un module pour manipuler le temps (par exemple, pour créer des délais).
- `capteur = dht.DHT11(machine.Pin(2))` : Cette ligne crée une instance de l'objet DHT11 appelée `capteur`. Le DHT11 est connecté au pin 2 de la machine (en supposant que vous utilisiez un ESP8266 ou un ESP32, où le pin 2 est généralement disponible).
- `while True:` : Ceci commence une boucle infinie. Le code à l'intérieur de cette boucle continuera à s'exécuter indéfiniment.
- `capteur.measure()` : Cette ligne indique au capteur DHT11 de commencer une mesure. Le capteur effectue une lecture de la température et de l'humidité de l'air.
- `print("Température:", capteur.temperature())` : Cette ligne affiche la température mesurée par le capteur. La méthode `temperature()` renvoie la température mesurée lors du dernier appel à `capteur.measure()`.
- `print("Humidité:", capteur.humidity())` : De même, cette ligne affiche l'humidité mesurée par le capteur. La méthode `humidity()` renvoie l'humidité mesurée lors du dernier appel à `capteur.measure()`.
- `time.sleep(2)` : Pour finir, cette ligne fait une pause dans l'exécution du programme pendant 2 secondes. C'est important car le capteur DHT11 a besoin d'un certain temps entre les lectures pour garantir des mesures précises. En général, il est recommandé d'attendre au moins 2 secondes entre chaque lecture.

Donc, en résumé, ce code effectue une lecture continue de la température et de l'humidité avec un capteur DHT11, et imprime ces valeurs toutes les 2 secondes. Ressources supplémentaires

Infos supplémentaires

- Tutoriel vidéo sur l'utilisation du capteur DHT11 avec MicroPython et Raspberry Pi Pico : https://youtu.be/hZ9zdvTRnTw
- Documentation de la bibliothèque DHT11 pour MicroPython : https://github.com/mcauser/micropython-dht11

Ces instructions vous permettront de lire les données de température et d'humidité à partir du capteur DHT11 à l'aide de MicroPython sur votre Raspberry Pi Pico.

HC-SR04: Capteur de distance à ultra-sons

Cet article présente HC-SR04, un capteur de distance à ultra-sons très répandu et facile à utiliser.

Introduction au capteur de distance HC-SR04

- Le capteur de distance HC-SR04 est un capteur ultrasonique populaire utilisé pour mesurer la distance entre le capteur et un objet.
- Il utilise des ondes sonores ultrasoniques pour déterminer la distance, en émettant un signal et en mesurant le temps nécessaire pour que l'écho revienne.

Configuration matérielle

- Connectez la broche de déclenchement (Trig) du capteur HC-SR04 à un GPIO du Raspberry Pi Pico (par exemple, GPIO2).
- Connectez la broche d'écho (Echo) du capteur HC-SR04 à un autre GPIO du Pico (par exemple, GPIO3).
- Alimentez le capteur HC-SR04 en connectant le fil rouge au 5V du Pico et le fil noir à la masse (GND).

Exemple de code pour la mesure de distance avec le capteur HC-SR04

```
import machine
import utime

# Définir les broches GPIO pour le déclenchement et l'écho
trigger_pin = machine.Pin(2, machine.Pin.OUT)
echo_pin = machine.Pin(3, machine.Pin.IN)

# Fonction pour mesurer la distance
def mesure_distance():
    # Générer une impulsion de déclenchement
    trigger_pin.low()
    utime.sleep_us(2)
    trigger_pin.high()
    utime.sleep_us(10)
    trigger_pin.low()

    # Attendre le signal d'écho
    while echo_pin.value() == 0:
        signal_depart = utime.ticks_us()

    while echo_pin.value() == 1:
        signal_arrivee = utime.ticks_us()

    # Calculer la durée de l'écho
    duree_echo = signal_arrivee - signal_depart

    # Calculer la distance en fonction de la durée de l'écho
    distance = (duree_echo * 0.0343) / 2

    return distance

while True:
    # Mesurer la distance avec le capteur HC-SR04
    distance = mesure_distance()

    # Afficher la distance mesurée
    print("Distance : {:.2f} cm".format(distance))

    # Attendre avant de refaire une mesure
    utime.sleep(1)
```

Explication du code

- Importez les modules nécessaires, notamment machine pour accéder aux fonctionnalités du Raspberry Pi Pico et utime pour la gestion du temps.
- Définissez les broches GPIO pour le déclenchement (Trig) et l'écho (Echo) du capteur HC-SR04.

- Définissez une fonction mesure_distance() pour effectuer la mesure de distance.
- Dans la fonction mesure_distance(), générez une impulsion de déclenchement en mettant la broche de déclenchement à l'état bas puis haut puis à nouveau bas.
- Attendez le signal d'écho en mesurant le temps auquel l'écho commence et se termine.

Calculez la durée de l'écho en soustrayant le temps de départ du temps d'arrivée. 7. Calculez la distance en utilisant la formule de conversion de la durée de l'écho en distance (en supposant une vitesse du son de 343 m/s).

- Dans la boucle principale, appelez la fonction mesure_distance() pour mesurer la distance avec le capteur HC-SR04.
- Affichez la distance mesurée et attendez avant de refaire une mesure.

Utilisation du capteur de distance HC-SR04 avec un module open source

Installation du module open source

- Téléchargez le module open source hc_sr04 pour MicroPython à partir du dépôt GitHub officiel (https://github.com/rsc1975/micropython-hcsr04).
- Copiez les fichiers hc_sr04.py et pulseio.py sur votre Raspberry Pi Pico.

Exemple de code pour la mesure de distance avec le capteur HC-SR04

```
import machine
import utime
from hc_sr04 import HCSR04

# Définir les broches GPIO pour le déclenchement et l'écho
trigger_pin = machine.Pin(2)
echo_pin = machine.Pin(3)

# Créer une instance du capteur HC-SR04
capteur_distance = HCSR04(trigger_pin, echo_pin)

while True:
    # Mesurer la distance avec le capteur HC-SR04
    distance = capteur_distance.distance_cm()

    # Afficher la distance mesurée
    print("Distance : {:.2f} cm".format(distance))

    # Attendre avant de refaire une mesure
    utime.sleep(1)
```

Explication du code

- Importez les modules nécessaires, notamment machine pour accéder aux fonctionnalités du Raspberry Pi Pico et utime pour la gestion du temps.
- Importez la classe HCSR04 du module hc_sr04 pour utiliser le capteur HC-SR04.
- Définissez les broches GPIO pour le déclenchement (Trig) et l'écho (Echo) du capteur HC-SR04.
- Créez une instance du capteur HC-SR04 en passant les broches GPIO comme arguments.
- Dans la boucle principale, appelez la méthode distance_cm() de l'instance du capteur pour mesurer la distance.
- Affichez la distance mesurée et attendez avant de refaire une mesure.

Ressources supplémentaires

- Tutoriel vidéo sur l'utilisation du capteur de distance HC-SR04 avec MicroPython et Raspberry Pi Pico : https://youtu.be/YAF3o3SsCmk
- Dépôt GitHub du module open source hc_sr04 pour MicroPython : https://github.com/rsc1975/micropython-hcsr04

Ces instructions vous permettront de mesurer la distance à l'aide du capteur HC-SR04 en utilisant un module open source avec MicroPython sur votre Raspberry Pi Pico.

Ressources supplémentaires

- Tutoriel vidéo sur l'utilisation du capteur de distance HC-SR04 avec MicroPython et Raspberry Pi Pico : https://youtu.be/YAF3o3SsCmk
- Bibliothèque hcsr04 pour MicroPython sur GitHub : https://github.com/rsc1975/micropython-hcsr04

Ces instructions vous permettront de mesurer la distance à l'aide du capteur HC-SR04 en utilisant MicroPython sur votre Raspberry Pi Pico.

Les micro-contrôleurs

> *Un microcontrôleur est une unité de commande intégrée utilisée pour **contrôler le fonctionnement d'un robot**. Il s'agit d'un petit ordinateur sur une seule puce (SoC) qui combine un **microprocesseur**, de la **mémoire** et des **interfaces d'entrée/sortie**.*

En tant qu'unité de commande, le microcontrôleur **exécute les programmes et les instructions** qui définissent le comportement du robot. Il reçoit des informations provenant des capteurs du robot, traite ces données et envoie des commandes aux actionneurs pour effectuer des actions spécifiques.

Les microcontrôleurs sont souvent utilisés dans les robots en raison de leur **taille compacte**, de leur **faible consommation d'énergie** et de leur capacité à exécuter des tâches en **temps réel**. Ils offrent une solution intégrée pour contrôler les mouvements, les interactions avec l'environnement et les communications du robot.

Les microcontrôleurs peuvent être programmés à l'aide de différents langages, tels que **C/C++**, Python, **Micro-Python**, ou des langages de programmation visuels basés sur des blocs. Les programmeurs écrivent des codes qui spécifient le comportement du robot, y compris la prise de décision, les mouvements, la détection d'obstacles et les réponses aux événements.

En résumé, un microcontrôleur est l'**unité de commande centrale** d'un robot, responsable de l'exécution des programmes, de la gestion des capteurs et des actionneurs, et de la coordination des opérations pour permettre au robot d'accomplir des tâches spécifiques.

Micro:bit :

- Le micro:bit est un microcontrôleur largement utilisé pour l'apprentissage de la robotique, en particulier pour les débutants.
- Il est petit, abordable et **dispose de nombreux capteurs intégrés **tels que l'accéléromètre, le magnétomètre et les boutons.
- Le micro:bit utilise une programmation par blocs conviviale, ce qui le rend facile à apprendre pour les débutants.
- Il est souvent utilisé pour créer des projets robotiques simples tels que des robots suiveurs de ligne ou des robots éviteurs d'obstacles.

En savoir plus (notes de cours)

ESP8266 :

- L'ESP8266 est un microcontrôleur WiFi populaire utilisé dans la robotique et l'Internet des objets (IoT).
- Il dispose d'une connectivité WiFi intégrée, ce qui permet aux robots de se connecter à des réseaux sans fil et d'interagir avec d'autres appareils.
- L'ESP8266 est compatible avec plusieurs langages de programmation, tels que Arduino IDE et MicroPython.
- Il est souvent utilisé pour des projets de robotique avancés nécessitant une communication sans fil, tels que des robots contrôlés par smartphone ou des robots de surveillance.

Raspberry Pi Pico :

- Le Raspberry Pi Pico est un microcontrôleur développé par la Fondation Raspberry Pi.
- Il est basé sur le microcontrôleur RP2040 et dispose de nombreuses fonctionnalités, telles que des broches d'E/S polyvalentes et une programmation en MicroPython et C/C++.
- Le Raspberry Pi Pico est abordable, polyvalent et peut être utilisé pour des projets robotiques de taille moyenne à avancée.
- Il est souvent utilisé pour créer des robots autonomes, des robots avec des capteurs avancés et des projets de robotique embarquée.

En savoir plus (notes de cours)

LEGO Mindstorms :

- LEGO Mindstorms est une **plateforme de robotique éducative** très populaire pour les enfants et les débutants.
- Elle utilise des briques LEGO spéciales, des moteurs et des capteurs pour construire des robots.
- Les robots LEGO Mindstorms peuvent être programmés à l'aide d'un logiciel convivial basé sur des blocs graphiques.
- Cette plateforme est idéale pour enseigner les bases de la robotique et de la programmation, en mettant l'accent sur la créativité et la construction physique.

Thymio :

- Thymio est un **robot éducatif** conçu pour l'apprentissage de la robotique et de la programmation.
- Il est doté de capteurs et d'actionneurs, tels que des capteurs de proximité, des détecteurs de lumière et des moteurs.
- Thymio peut être programmé à l'aide d'un logiciel visuel ou d'un langage de programmation textuel, ce qui permet d'explorer différents niveaux de complexité.
- Ce robot est largement utilisé dans les écoles pour enseigner la programmation, la logique et les concepts de base de la robotique.

Ces microcontrôleurs offrent une excellente opportunité d'apprentissage pour les étudiants dans le domaine de la robotique. Ils sont abordables, faciles à programmer et offrent des fonctionnalités adaptées à différents niveaux de compétence.

Le Raspberry Pi Pico

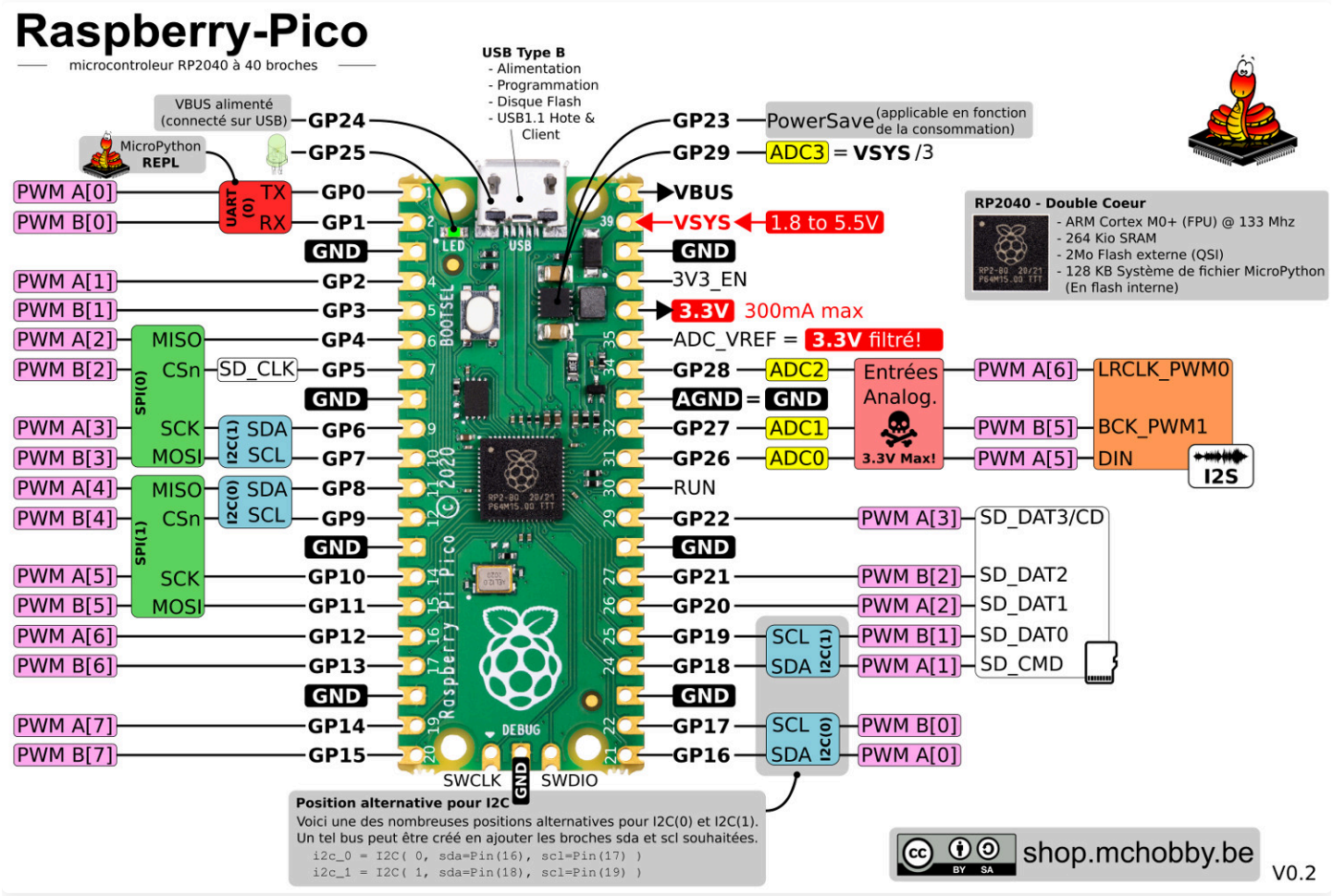
Les Broches du Raspberry Pi Pico en MicroPython

> *Sur cette page, nous allons explorer les différentes broches du Pico, leur numéro, leur fonction (analogique, PWM, numérique), et comment les utiliser en MicroPython.*

Le Raspberry Pi Pico dispose de plusieurs broches GPIO (General-Purpose Input/Output) qui peuvent être utilisées pour diverses tâches en MicroPython.

Broches GPIO du Raspberry Pi Pico

Le Raspberry Pi Pico possède 26 broches GPIO que vous pouvez utiliser pour interagir avec des composants électroniques et des capteurs. Voici une liste des broches GPIO disponibles sur le Pico :



Utilisation des Broches en MicroPython

Configuration d'une broche en mode numérique

Pour configurer une broche en mode numérique (entrée ou sortie), vous pouvez utiliser la bibliothèque `machine` en MicroPython. Par exemple, pour configurer la broche GP0 en tant que sortie, vous pouvez utiliser le code suivant :

```
import machine

# Configurer GP0 en sortie
broche_gp0 = machine.Pin(0, machine.Pin.OUT)
```

MicroPython offre plusieurs méthodes pour contrôler la tension de sortie :

```
from machine import Pin

Pin.on() et Pin.off()
Pin.high() et Pin.low()
Pin.value(valeur)
```

Les noms de fonctions sont explicites :

```
pin_led = Pin(16, mode=Pin.OUT)

# Impose une tension de 3.3V en sortie (état logique haut)
pin_led.on()
pin_led.high()
pin_led.value(1)

# Impose une tension de 0V en sortie (état logique bas)
pin_led.off()
pin_led.low()
pin_led.value(0)
```

Utilisation des broches en mode PWM

Le Raspberry Pi Pico prend en charge la modulation de largeur d'impulsion (PWM) sur certaines broches. Cela permet de contrôler la vitesse des moteurs, la luminosité des LED, etc. Voici comment configurer une broche en mode PWM :

```
import machine

# Configurer GP2 en mode PWM
broche_gp2 = machine.Pin(2)
pwm = machine.PWM(broche_gp2)
```

Lecture de valeurs analogiques

Pour lire des valeurs analogiques à partir des broches GPIO, vous pouvez utiliser l'ADC (Convertisseur Analogique-Numérique) intégré au Pico.

Note/Rappel: Le RPi Pico possède 4 broches analogiques: GP26->29

Voici comment configurer une broche pour lire une valeur analogique :

```
import machine

# Configurer GP26 pour la lecture analogique
broche_gp26 = machine.Pin(26)
adc = machine.ADC(broche_gp26)

# Lire la valeur analogique
valeur_analogique = adc.read_u16()
```

Le code suivant permet de lire la valeur d'un potentiomètre analogique:

```
from machine import ADC, Pin
from utime import sleep
pot = ADC(26)
conversion_factor = 3.3 / 65535

while True:
    raw = pot.read_u16()
    volts = raw * conversion_factor
    print('Raw: {} '.format(raw), 'Voltage {:.1f}V'.format(volts))
    sleep(1)
```

Et ce code permet de changer la fréquence de clignotement d'une LED en fonction de la valeur lue sur un potentiomètre:

```
from machine import ADC, Pin
from utime import sleep

pot = ADC(26)
led = Pin(25, Pin.OUT)

def map(s, a1, a2, b1, b2):
    return b1 + (s - a1) * (b2 - b1) / (a2 - a1)

while True:
    raw = pot.read_u16()
    delay = map(raw, 0, 65535, 0, 1)
    print('Raw: {} '.format(raw), 'Delay {:.1f}s'.format(delay))
    led.value(1)
    sleep(delay)
    led.value(0)
    sleep(delay)
```

Conclusion

Les broches GPIO du Raspberry Pi Pico offrent de nombreuses possibilités pour interagir avec le monde physique en utilisant MicroPython. Que ce soit pour contrôler des LED, des moteurs, ou pour lire des capteurs analogiques, vous avez maintenant les connaissances de base pour commencer à créer vos propres projets avec le Pico. Explorez davantage et expérimentez avec les différentes broches pour donner vie à vos idées.

MicroPython sur RPi Pico avec Thonny Python

///

Prérequis

- Assurez-vous d'avoir un Raspberry Pi Pico fonctionnel et un câble USB pour le connecter à votre ordinateur.
- Téléchargez et installez l'IDE Thonny Python sur votre ordinateur à partir du site officiel de Thonny (https://thonny.org/).

Installation du firmware MicroPython

- Visitez le site officiel de MicroPython pour Raspberry Pi Pico (https://micropython.org/download/rp2-pico/) et téléchargez le dernier firmware MicroPython (.uf2) pour le Raspberry Pi Pico.

Connexion du Raspberry Pi Pico

- Connectez votre Raspberry Pi Pico à votre ordinateur à l'aide du câble USB. Assurez-vous que le commutateur "BOOTSEL" est enfoncé pour mettre le Pico en mode Bootloader.

Flashage du firmware MicroPython

- Ouvrez l'IDE Thonny Python sur votre ordinateur.
- Dans Thonny, cliquez sur le bouton "Sélectionner l'interpréteur" en haut à droite.
- Dans la fenêtre de sélection de l'interpréteur, choisissez "Raspberry Pi Pico (MicroPython)".
- Maintenant, dans Thonny, cliquez sur le bouton "Flasher" en haut.
- Dans la fenêtre de flashage, sélectionnez le firmware MicroPython (.uf2) que vous avez précédemment téléchargé.
- Cliquez sur "OK" pour flasher le firmware sur votre Raspberry Pi Pico.

Écriture et exécution du code MicroPython

- Une fois le firmware MicroPython installé, le Raspberry Pi Pico redémarrera automatiquement.
- Dans Thonny, vous verrez le Pico répertorié comme un périphérique connecté.
- Créez un nouveau fichier en cliquant sur "Nouveau" dans le menu Fichier.
- Écrivez votre code MicroPython dans le nouvel onglet du fichier.
- Pour exécuter le code, cliquez sur le bouton "Exécuter" ou appuyez sur F5.

Interaction avec le Raspberry Pi Pico

- Vous pouvez utiliser la console REPL (Read-Eval-Print Loop) pour interagir avec le Raspberry Pi Pico en temps réel.
- Dans Thonny, cliquez sur l'icône de console en bas de la fenêtre pour ouvrir la console REPL.
- Vous pouvez saisir des commandes directement dans la console pour interagir avec les broches GPIO, les capteurs ou les actionneurs connectés au Raspberry Pi Pico.

Débogage et surveillance

- Thonny Python offre également des fonctionnalités de débogage et de surveillance pour vous aider à identifier et corriger les erreurs dans votre code MicroPython.
- Vous pouvez placer des points d'arrêt, examiner les variables et suivre l'exécution pas à pas pour comprendre le comportement de votre code.

Ces étapes vous permettront de configurer et d'utiliser MicroPython sur votre Raspberry Pi Pico en utilisant l

Raspberry Pi Pico et les Ports GPIO

///

- Le Raspberry Pi Pico est un microcontrôleur à faible coût développé par la Fondation Raspberry Pi.
- Il est basé sur le microcontrôleur RP2040 et offre de nombreuses fonctionnalités pour les projets de robotique et d'électronique.



Les Ports GPIO (General-Purpose Input/Output)

- Le Raspberry Pi Pico dispose de **26 broches GPIO (0 à 25)** qui peuvent être configurées en tant qu'**entrées ou sorties** pour interagir avec d'autres composants électroniques.
- Les broches GPIO peuvent être utilisées pour **lire des signaux provenant de capteurs externes ou pour contrôler des actionneurs** tels que des moteurs ou des LED.
- Chaque broche GPIO peut être configurée individuellement et prend en charge différents modes de fonctionnement, tels que l'**entrée**, la **sortie**, la **PWM** (modulation de largeur d'impulsion), l'I2C, le SPI, etc.

Utilisation des Ports GPIO

- Configuration des broches : Avant d'utiliser une broche GPIO, il est nécessaire de la **configurer dans le mode de fonctionnement souhaité** en utilisant des bibliothèques de programmation appropriées, telles que RP2PIO ou RP2GPIO pour le Raspberry Pi Pico.
- Lecture des broches d'entrée : Les broches GPIO configurées en tant qu'entrées peuvent être utilisées pour lire des signaux provenant de capteurs externes. On peut les lire en utilisant des fonctions de lecture appropriées dans le langage de programmation utilisé, telles que input() en Python.
- Contrôle des broches de sortie : Les broches GPIO configurées en tant que sorties peuvent être utilisées pour contrôler des actionneurs, tels que des moteurs ou des LED. On peut les contrôler en utilisant des fonctions d'écriture appropriées dans le langage de programmation utilisé, telles que output() en Python.

Exemples d'utilisation des Ports GPIO

- Contrôle de LED : On peut connecter une LED à une broche GPIO configurée en tant que sortie et utiliser la fonction d'écriture pour allumer ou éteindre la LED.
- Lecture de bouton : On peut connecter un bouton à une broche GPIO configurée en tant qu'entrée et utiliser la fonction de lecture pour détecter si le bouton est enfoncé ou relâché.
- Contrôle de moteur : On peut utiliser les broches GPIO configurées en tant que sorties pour contrôler la vitesse et la direction d'un moteur en utilisant la modulation de largeur d'impulsion (PWM) pour créer des signaux analogiques simulés.

Modes Analogique, Digital et PWM

Mode Analogique

- Le mode analogique est utilisé pour représenter et manipuler des **signaux continus** qui peuvent prendre une gamme de valeurs différentes.
- Dans le contexte des microcontrôleurs, le mode analogique permet de **mesurer ou de générer des signaux analogiques**.
- Lorsqu'un microcontrôleur est configuré en mode analogique, il peut **lire des tensions** provenant de capteurs analogiques tels que des capteurs de température ou de lumière.
- Les microcontrôleurs peuvent également générer des signaux analogiques en utilisant des *convertisseurs numérique-analogique* (CNA) pour contrôler des dispositifs tels que des amplificateurs ou des moteurs.

Mode Digital

- Le mode digital est utilisé pour représenter des signaux qui ne peuvent prendre que **deux valeurs distinctes : 0 et 1 (ou HIGH et LOW)**.
- Les microcontrôleurs sont capables de traiter des signaux digitaux, ce qui leur permet de communiquer avec d'autres composants électroniques tels que des capteurs, des actionneurs ou d'autres microcontrôleurs.
- Les broches GPIO configurées en tant que sorties digitales peuvent envoyer des signaux à d'autres composants, tandis que les broches configurées en tant qu'entrées digitales peuvent recevoir des signaux provenant d'autres sources.

Modulation de Largeur d'Impulsion (PWM)

- La modulation de largeur d'impulsion (PWM) est une technique utilisée pour **simuler des signaux analogiques en utilisant des signaux digitaux**.
- Elle est couramment utilisée pour contrôler la vitesse ou l'intensité de dispositifs tels que des moteurs, des LED ou des servomoteurs.
- Dans un signal PWM, **la largeur de l'impulsion varie dans le temps**, tandis que la période de l'impulsion reste constante.
- En ajustant le rapport cyclique (rapport entre la durée de l'impulsion et la période), on peut contrôler la moyenne pondérée du signal, ce qui permet de varier la vitesse ou l'intensité du dispositif contrôlé.

Utilisation dans la Robotique

- En robotique, les **signaux analogiques** sont souvent utilisés **pour mesurer des grandeurs** telles que la **température**, la **luminosité** ou la **distance**.
- Les signaux digitaux sont utilisés pour la communication entre composants électroniques et la **lecture d'états** de capteurs, tels que les **boutons** ou les **interrupteurs**.
- La modulation de largeur d'impulsion (**PWM**) est utilisée pour **contrôler précisément la vitesse des moteurs** ou la **luminosité des LED**, ce qui est essentiel dans de nombreux projets robotiques.

Il est important de comprendre ces différents modes et techniques, car ils permettent aux microcontrôleurs de s'adapter à différentes situations et de contrôler divers dispositifs dans les projets robotiques.

Ressources supplémentaires

- Documentation officielle du Raspberry Pi Pico : https://www.raspberrypi.org/documentation/pico/
- Tutoriels et exemples de projets : Le site officiel du Raspberry Pi Pico propose une variété de tutoriels et de projets pour vous aider à démarrer avec les ports GPIO et d'autres fonctionnalités du microcontrôleur.

Utilisation du Raspberry Pi Pico avec MicroPython

> *Dans cet article de cours, nous allons explorer comment utiliser le Raspberry Pi Pico avec MicroPython et vous montrer des exemples de programmes pour lire et écrire des valeurs sur les broches GPIO (General-Purpose Input/Output).*

Le Raspberry Pi Pico est un microcontrôleur puissant et abordable qui peut être programmé avec MicroPython, un langage de programmation convivial pour les débutants.

Prérequis

Avant de commencer, assurez-vous d'avoir les éléments suivants à votre disposition :

- Un Raspberry Pi Pico
- Un ordinateur avec MicroPython installé (vous pouvez télécharger MicroPython sur le site officiel)
- Un câble micro USB pour la connexion entre l'ordinateur et le Raspberry Pi Pico

Installation de MicroPython

Ceci est une procédure d'installation manuelle que nous n'avons pas vue en classe. Je la laisse pour que vous sachiez que ça existe, mais il ne faut pas la connaître pour le cours.

1. Téléchargez la dernière version de MicroPython depuis le site officiel (https://micropython.org/).
2. Connectez votre Raspberry Pi Pico à votre ordinateur à l'aide du câble micro USB. Le Pico devrait être reconnu comme un périphérique de stockage USB.
3. Copiez le fichier MicroPython que vous avez téléchargé sur le Pico. Il doit avoir une extension ".uf2".
4. Déconnectez le Pico de votre ordinateur, puis reconnectez-le. Vous verrez un nouveau périphérique de stockage appelé "RPI-RP2".
5. Supprimez le fichier "CURRENT.UF2" s'il existe et renommez votre fichier MicroPython téléchargé en "CURRENT.UF2". Cela permettra au Pico de démarrer automatiquement avec MicroPython.
6. Déconnectez à nouveau le Pico de l'ordinateur, puis reconnectez-le. Il devrait maintenant exécuter MicroPython.

Programme de base

Commençons par un exemple simple de programme en MicroPython pour allumer et éteindre une LED connectée à une broche GPIO du Raspberry Pi Pico.

```
import machine
import utime

# Définir la broche GPIO à utiliser
led = machine.Pin(25, machine.Pin.OUT)

# Boucle infinie pour clignoter la LED
while True:
    led.toggle()  # Inverse l'état de la LED
    utime.sleep(1)  # Attendre 1 seconde
```

Dans ce programme, nous utilisons le module `machine` pour gérer les broches GPIO. Nous définissons la broche 25 comme une sortie (`machine.Pin.OUT`). Ensuite, nous utilisons une boucle infinie pour alterner l'état de la LED entre allumé et éteint toutes les secondes à l'aide de la méthode `toggle()` .

Lecture de valeurs analogiques

Vous pouvez également lire des valeurs analogiques à partir de broches GPIO pour mesurer des grandeurs telles que la luminosité avec un capteur de lumière. Voici un exemple pour lire une valeur analogique à partir de la broche 26 :

```python
import machine

# Définir la broche GPIO à utiliser
analog_pin = machine.Pin(26)

# Configurer l'ADC (Convertisseur Analogique-Numérique)
adc = machine.ADC(analog_pin)

# Lire et imprimer la valeur analogique
while True:
    valeur = adc.read_u16()  # Lecture de la valeur analogique (0-65535)
    print("Valeur analogique : ", valeur)
```

Dans cet exemple, nous utilisons le module `machine` pour configurer l'ADC (Convertisseur Analogique-Numérique) sur la broche 26. La valeur analogique est lue à l'aide de la méthode `read_u16()` et affichée dans la console.

Conclusion

Le Raspberry Pi Pico avec MicroPython offre une excellente plateforme pour l'apprentissage de la programmation et de l'électronique. Vous pouvez créer une variété de projets intéressants en utilisant les broches GPIO pour contrôler des composants électroniques. N'hésitez pas à explorer davantage et à créer vos propres projets en utilisant ces concepts de base.

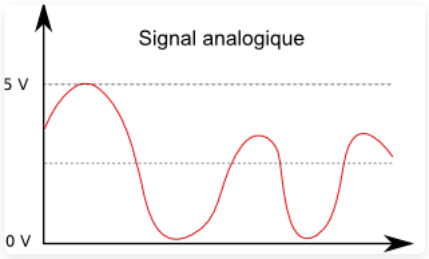
Les différents types de signaux

Les microcontrôleurs sont des composants essentiels dans de nombreux systèmes électroniques, des appareils ménagers aux robots industriels. Pour interagir avec leur environnement, ces microcontrôleurs utilisent différents types de signaux. Dans cet article, nous allons explorer en détail les trois principaux types de signaux utilisés : les signaux analogiques, les signaux digitaux et les signaux à modulation de largeur d'impulsion (PWM). Nous verrons comment ils fonctionnent, leurs applications et les bonnes pratiques pour les utiliser.

1. Signaux Analogiques

Qu'est-ce qu'un Signal Analogique ?

Un signal analogique est un **signal continu** qui peut prendre une **infinité de valeurs** dans une plage donnée. Contrairement aux signaux digitaux, qui ne peuvent prendre que des valeurs discrètes (0 ou 1), les signaux analogiques peuvent représenter des variations continues, comme la température, la luminosité ou le son.



Exemple de Signal Analogique

Imaginez un capteur de température analogique. La sortie de ce capteur est une tension continue qui varie en fonction de la température mesurée. Par exemple, à 20°C, la sortie pourrait être de 2V, à 25°C, elle pourrait être de 2.5V, et ainsi de suite.

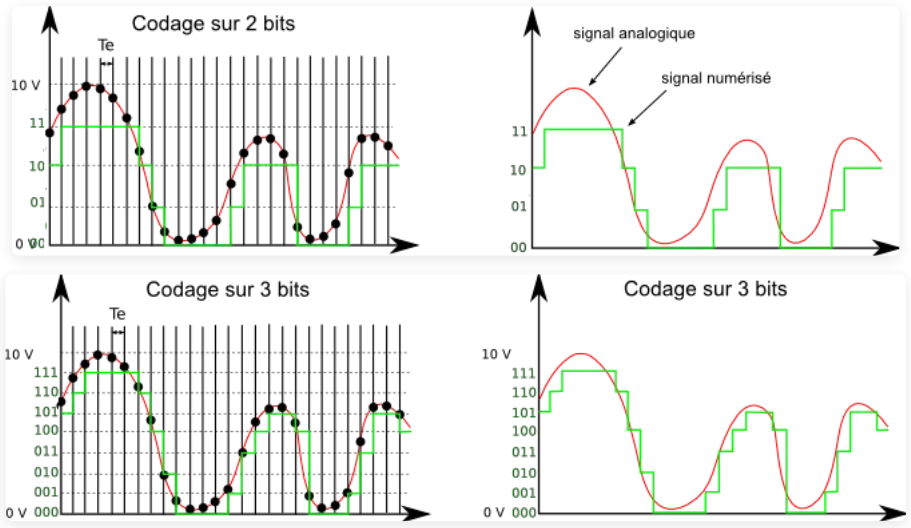
Utilisation dans les Microcontrôleurs

Les microcontrôleurs ne peuvent pas traiter directement les signaux analogiques. Ils doivent **convertir ces signaux** en valeurs numériques à l'aide d'un **convertisseur analogique-numérique (ADC)**. Une fois le signal converti, le microcontrôleur peut le traiter de manière numérique.

L'ADC **convertit une tension analogique continue** en une valeur numérique discrète que le microcontrôleur peut traiter. Cette conversion se fait en **échantillonnant** le signal analogique à des intervalles de temps réguliers et en attribuant à chaque échantillon une valeur numérique correspondante.

Chaque échantillon est attribué à la valeur numérique la plus proche parmi un ensemble de valeurs discrètes possibles.

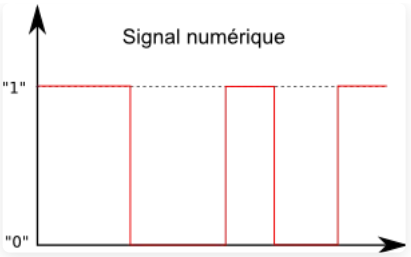
Une caractéristique important des ADC est leur **résolution** : C'est le **nombre de bits de la sortie numérique**. Une résolution de **10 bits** signifie que la sortie peut prendre **1024** valeurs différentes (2^10). **Plus la résolution est élevée, plus la conversion est précise**.



2. Signaux Digitaux

Qu'est-ce qu'un Signal Digital ?

Un signal **digital**, ou **numérique**, ne peut prendre que deux valeurs distinctes : **0 (faible) ou 1 (élevé)** (en tout cas en ce qui concerne les i/o des micro-controlleurs). Ces signaux sont utilisés pour représenter des états binaires ou des données discrètes.



Exemple de Signal Digital

Considérons un **interrupteur** connecté à un microcontrôleur. Lorsque l'interrupteur est fermé, la sortie est à 1 (élevé), et lorsqu'il est ouvert, la sortie est à 0 (faible).

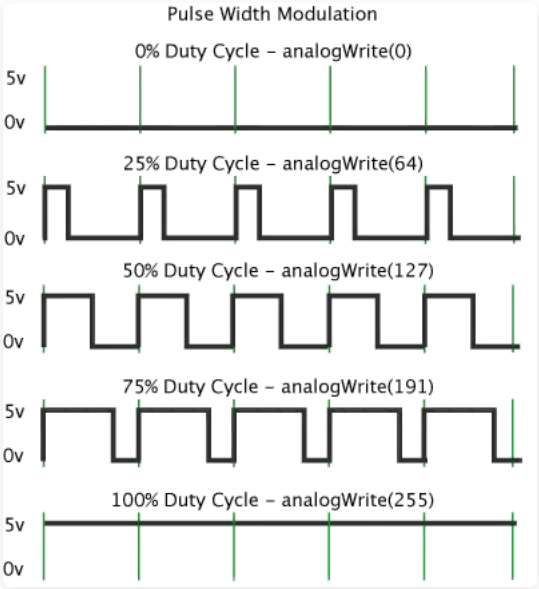
Utilisation dans les Microcontrôleurs

Les microcontrôleurs traitent naturellement les signaux digitaux. Ils peuvent lire les états des broches d'entrée/sortie (I/O) et exécuter des actions en fonction de ces états.

3. Signaux à Modulation de Largeur d'Impulsion (PWM)

Qu'est-ce que le PWM ?

La **modulation de largeur d'impulsion** (GB ***PWM ou pulse width modulation***) est une technique où la largeur des impulsions d'un signal est modulée pour représenter une **valeur analogique moyenne**. Le **rapport cyclique** (GB ***duty cycle***) détermine la **proportion de temps** pendant laquelle le signal est à l'état haut (1) par rapport à l'état bas (0).



Le graphe ci-dessus montre l'équivalence entre différents duty cycles et la valeur équivalente en analogique (analogWrite).

Exemple de PWM

Supposons que nous ayons une LED contrôlée par un signal PWM. En modifiant le rapport cyclique, nous pouvons contrôler la luminosité de la LED. Un rapport cyclique de 50% signifie que le signal est haut pendant 50% du temps et bas pendant les 50% restants, ce qui correspond à une luminosité moyenne de la LED.

Utilisation dans les Microcontrôleurs

Les microcontrôleurs peuvent générer des signaux PWM à l'aide de modules dédiés. Ces signaux sont souvent utilisés pour le contrôle de la puissance, comme dans les moteurs, les LEDs ou les régulateurs de tension.

Conclusion

Les microcontrôleurs utilisent des signaux **analogiques**, **digitaux** et **PWM** pour interagir avec leur environnement. Chaque type de signal a ses propres caractéristiques, avantages et inconvénients. Les signaux analogiques permettent de représenter des valeurs continues, les signaux digitaux sont parfaits pour les états binaires et la communication, et les signaux PWM offrent une solution efficace pour le contrôle de puissance et la génération de valeurs analogiques moyennes.

En comprenant comment utiliser ces différents types de signaux, vous pouvez concevoir des systèmes plus robustes et efficaces.



INTERNET

Internet est un **réseau mondial d'ordinateurs connectés entre eux**. Il permet aux utilisateurs d'accéder et de partager des informations sur une échelle sans précédent. Via Internet, vous pouvez envoyer des emails, consulter des sites web, télécharger des fichiers, et bien plus encore, en quelques clics.

Création de pages Web

introduction

Bootstrap

> *Bootstrap est un framework front-end open-source pour le développement de sites et d'applications web. Il a été conçu par Twitter pour assurer la cohérence du design à travers les projets internes. Aujourd'hui, Bootstrap est largement utilisé pour construire des interfaces web responsives et mobiles rapidement.*

Qu'est-ce que Bootstrap ?

Bootstrap est un **framework front-end open-source** pour le développement de sites et d'applications web. Il a été conçu par Twitter pour assurer la cohérence du design à travers les projets internes. Aujourd'hui, Bootstrap est largement utilisé pour construire des interfaces web réactives et mobiles rapidement.

Pourquoi utiliser Bootstrap ?

1. **Responsive Design** : Bootstrap utilise un système de **grille flexible** (GB ***grid***) qui s'adapte à la taille de l'écran, rendant votre site web compatible avec tous les appareils (desktop, tablette, mobile).
2. **Prêt à l'emploi** : Il offre une vaste gamme de **composants réutilisables** tels que des boutons, des formulaires, et des navigations, qui peuvent être facilement intégrés dans votre projet.
3. **Personnalisable** : Bien qu'il soit utilisable tel quel, Bootstrap peut être **personnalisé** pour correspondre à l'identité visuelle de votre projet.
4. **Large Communauté** : Étant largement utilisé, Bootstrap possède une **grande communauté**, offrant un soutien, des plugins et des extensions.

Cinq Fonctionnalités Principales de Bootstrap

1. Système de Grille

Bootstrap utilise un système de grille flexible basé sur **12 colonnes**. Cela permet de créer des mises en page complexes et ***responsives***.

Exemple :

```
<div class="container">
  <div class="row">
    <div class="col-md-8">.col-md-8</div>
    <div class="col-md-4">.col-md-4</div>
  </div>
</div>
```

2. Composants UI

Il fournit une large gamme de **composants** UI tels que des boutons, des cartes, des alertes, et des barres de navigation, prêts à être utilisés.

Exemple de bouton :

```
<button type="button" class="btn btn-primary">Bouton Principal</button>
```

3. Système de Flexbox

Bootstrap a introduit **Flexbox**, qui offre une manière plus flexible et efficace de gérer la mise en page, l'alignement et l'espacement des éléments.

Exemple d'utilisation de Flexbox :

```
<div class="d-flex justify-content-between">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
```

4. Composants JavaScript

Bootstrap intègre divers **composants JavaScript** pour ajouter des fonctionnalités dynamiques, comme les modales, les carrousels, et les tooltips.

Exemple de Modal :

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
  Lancer la modal
</button>
```

5. Personnalisation

Bootstrap est hautement personnalisable, permettant d'ajuster les couleurs, les polices, et d'autres éléments pour s'adapter à votre branding.

Exemple de personnalisation :

Dans votre fichier CSS :

```
body {
  font-family: 'Votre police', sans-serif;
}
.btn-primary {
  background-color: #votreCouleur;
}
```

Autres Fonctionnalités Clés

- **Formulaires** : Styles de base pour les formulaires, les entrées, et les boutons.
- **Tables** : Styles pour les tableaux basiques et responsives.
- **Images** : Classes pour les images responsives et les thumbnails.
- **Helpers** : Classes utilitaires pour la gestion des marges, du texte, de l'alignement, etc.
- **Extensions** : Large gamme de plugins et d'extensions disponibles pour étendre les fonctionnalités.

En conclusion, Bootstrap est un outil puissant pour les développeurs web, simplifiant le processus de conception et assurant une expérience utilisateur cohérente sur divers appareils. Son système de grille, ses composants prêts à l'emploi, et sa personnalisation facile en font un choix populaire pour le développement web rapide et efficace.

Les breakpoints

> *Les breakpoints CSS définissent comment le layout d'un site web va réagir et s'adapter à différentes tailles d'écran. Bootstrap, en tant que framework de développement front-end, utilise ces breakpoints pour créer des designs responsifs. Cet article explore les breakpoints dans Bootstrap, leur importance, et comment les utiliser efficacement.*

Qu'est-ce qu'un Breakpoint ?

Un breakpoint (FR ***point de rupture***) est une condition dans CSS où certaines règles CSS s'appliquent uniquement à une gamme spécifique de **tailles d'écran**. Ces points de rupture permettent aux développeurs de créer des designs qui réagissent et s'adaptent aux changements de taille d'écran, comme des écrans de téléphone, de tablette, et d'ordinateur.

Les Breakpoints Standard dans Bootstrap

Bootstrap définit plusieurs breakpoints standard, qui sont des points de largeur d'écran prédéfinis pour cibler des appareils spécifiques :

- **Extra small (xs)** : pour les écrans <576px.
- **Small (sm)** : pour les écrans ≥576px.
- **Medium (md)** : pour les écrans ≥768px.
- **Large (lg)** : pour les écrans ≥992px.
- **Extra large (xl)** : pour les écrans ≥1200px.
- **Extra extra large (xxl)** : pour les écrans ≥1400px.

Ces valeurs peuvent être personnalisées, mais elles offrent un bon point de départ pour la plupart des designs web.

Utilisation des Breakpoints dans Bootstrap

1. Mise en Page Responsive

La grille de Bootstrap utilise ces breakpoints pour permettre aux colonnes de s'adapter à la taille de l'écran. Par exemple :

```
<div class="row">
  <div class="col-md-6 col-lg-4">...</div>
  <div class="col-md-6 col-lg-4">...</div>
  <div class="col-md-6 col-lg-4">...</div>
</div>
```

Dans cet exemple, chaque `div` utilise 6 colonnes sur les écrans moyens (`md`) et 4 colonnes sur les grands écrans (`lg`).

💡 Une définition de colonne spécifiée pour un breakpoint " `col-*` " est valable jusqu'au prochain breakpoint spécifié dans l'attribut `class` . Par exemple, si vous spécifiez `col-xs-6` et `col-lg-3` , la colonne occupera **6 colonnes** virtuelles pour les breakpoints `xs` , `sm` et `md` et **3 colonnes** virtuelles pour les breakpoints `lg` et `xl` .

2. Affichage Conditionnel

Les classes d'utilité de Bootstrap permettent d'afficher ou de masquer des éléments à des breakpoints spécifiques. Par exemple :

```
<div class="d-none d-md-block">Visible seulement sur md et plus large</div>
```

3. Taille du Texte et Alignement

Les breakpoints peuvent être utilisés pour ajuster la taille du texte ou l'alignement en fonction de la taille de l'écran :

```
<h1 class="text-center text-lg-left">Titre</h1>
```

4. Images Responsives

Bootstrap permet de rendre les images responsives, s'adaptant aux différents breakpoints :

```
<img src="image.jpg" class="img-fluid" alt="Responsive image">
```

Bonnes Pratiques avec les Breakpoints

1. **Mobile First** : Commencez par styliser pour les petits écrans, puis utilisez les breakpoints pour ajuster le design pour les écrans plus larges.
2. **Testez sur de Vrais Appareils** : Testez votre design sur différents appareils pour vous assurer qu'il répond correctement aux différents breakpoints.

3. **Utilisez les Breakpoints de Bootstrap à Bon Escient** : Ne créez pas de breakpoints inutiles. Utilisez les breakpoints standard de Bootstrap lorsque cela est possible.

Conclusion

Les breakpoints sont essentiels dans le développement web moderne pour créer des expériences utilisateurs fluides et réactives. Bootstrap offre un système de breakpoints robuste et facile à utiliser, ce qui en fait un choix idéal pour les développeurs cherchant à construire des sites web responsifs. En comprenant et en appliquant efficacement ces concepts, vous pouvez garantir que votre site sera esthétiquement agréable et fonctionnel sur tous les appareils.

Les containers

> *Les classes `.container` et `.container-fluid` sont des éléments fondamentaux du système de mise en page de Bootstrap. Elles servent de conteneurs principaux pour structurer votre contenu et le rendre réactif en s'adaptant à différentes tailles d'écran. Cet article explore en détail ces classes, leur rôle, leurs différences, et comment les utiliser efficacement pour créer des mises en page modernes et adaptatives.*

Les Bases des Conteneurs Bootstrap

Un **conteneur** est un élément HTML qui encapsule le contenu de votre site. Dans Bootstrap, il existe deux types principaux de conteneurs :

1. `.container` : Ce conteneur a une largeur fixe qui varie selon la taille de l'écran.
2. `.container-fluid` : Ce conteneur s'étend toujours sur 100% de la largeur de la fenêtre, quelle que soit sa taille.

Les conteneurs sont essentiels car ils servent de base pour organiser les colonnes et les rangées de la grille Bootstrap.

Utilisation de `.container`

La classe `.container` est idéale lorsque vous souhaitez que votre contenu soit centré et limité en largeur pour garantir une lisibilité optimale. Elle s'adapte automatiquement aux différentes tailles d'écran selon les points de rupture (breakpoints) définis dans Bootstrap.

▸ Largeurs des Conteneurs Fixes

Pour info, les largeurs par défaut de `.container` sont définies comme suit :

TAILLE DE L'ÉCRAN	BREAKPOINT	LARGEUR DU CONTENEUR
Extra small (xs)	< 576px	Auto
Small (sm)	≥ 576px	540px
Medium (md)	≥ 768px	720px
Large (lg)	≥ 992px	960px
Extra large (xl)	≥ 1200px	1140px
Extra extra large (xxl)	≥ 1400px	1320px

▸ Exemple de Code

Voici un exemple d'utilisation de `.container` :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemple de Conteneur</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="styl
</head>
<body>
  <div class="container">
    <h1>Bienvenue dans Bootstrap</h1>
    <p>Ceci est un conteneur fixe qui s'adapte à la taille de l'écran.</p>
  </div>
</body>
</html>
```

Utilisation de `.container-fluid`

La classe `.container-fluid` crée un conteneur qui s'étend sur toute la largeur de l'écran, quelle que soit la taille de ce dernier. Elle est idéale pour les mises en page pleine largeur, comme des en-têtes ou des pieds de page qui occupent tout l'espace horizontal.

Exemple de Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemple de Conteneur Fluide</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container-fluid bg-primary text-white">
    <h1 class="text-center">Mise en page pleine largeur</h1>
    <p class="text-center">Ce conteneur occupe toute la largeur de l'écran.</p>
  </div>
</body>
</html>
```

Comparaison Visuelle : `.container` vs `.container-fluid`

CARACTÉRISTIQUE	`.CONTAINER`	`.CONTAINER-FLUID`
Largeur	Fixe (varie par breakpoint)	100% de la largeur
Utilisation	Contenu centré et limité	Contenu pleine largeur
Adaptabilité	Oui, selon les breakpoints	Oui, toujours fluide

Applications Pratiques

En-Têtes et Pieds de Page Pleine Largeur : Utilisez `.container-fluid` pour les sections qui doivent occuper toute la largeur de l'écran.

Sections de Contenu Centré : Utilisez `.container` pour les sections principales afin de centrer et limiter la largeur du contenu.

Mise en Page Réactive : Combinez les breakpoints et les conteneurs pour ajuster dynamiquement les mises en page en fonction de la taille de l'écran.

Conclusion

Les classes `.container` et `.container-fluid` de Bootstrap 5 sont des outils puissants pour structurer et organiser vos mises en page. Tandis que `.container` offre une largeur fixe et centrée idéale pour un contenu bien structuré, `.container-fluid` s'adapte à toute la largeur de l'écran, parfait pour des designs immersifs et dynamiques. En maîtrisant ces outils, vous pouvez créer des mises en page réactives et esthétiques adaptées à tous les appareils.

Grille responsive

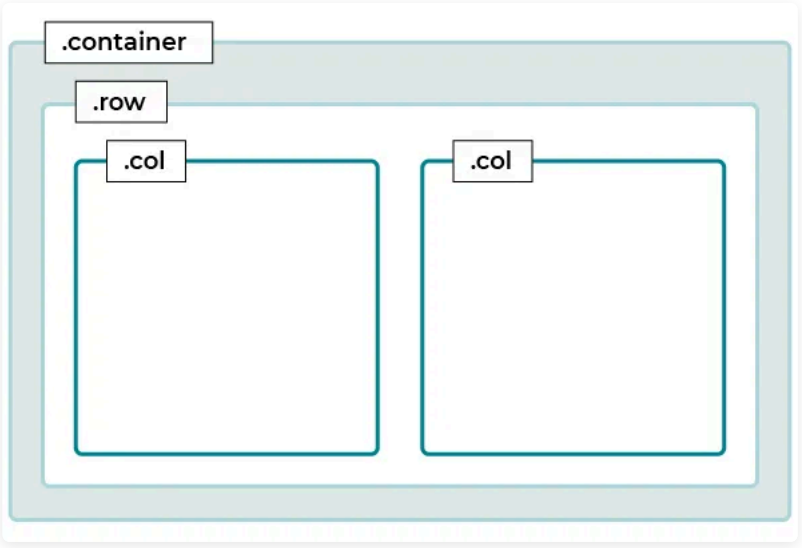
> *La grille responsive de Bootstrap est un système de mise en page qui s'adapte automatiquement à la taille de l'écran de l'utilisateur. Ce système est basé sur un ensemble de conteneurs, de lignes et de colonnes pour structurer et aligner le contenu de manière flexible et responsive. Dans cet article, nous allons explorer comment utiliser la grille responsive de Bootstrap avec des exemples concrets.*

Principe de Base de la Grille

La **grille Bootstrap** (ou *grid system*) est un système de mise en page flexible et structuré qui permet de **diviser une page web en colonnes** (ou, plus précisément, des parties de page web). Elle est au cœur de Bootstrap et est conçue pour faciliter l'organisation du contenu de votre site, quelle que soit la taille de l'écran : ordinateur, tablette ou smartphone.



La grille (GB ***grid***) est constituée de trois éléments principaux :



1. **Conteneurs (Containers)** : Ils encapsulent votre grille. Il y a deux types de conteneurs :
   - `.container` : conteneur à largeur fixe qui change de taille à différents points de rupture.
   - `.container-fluid` : conteneur pleine largeur pour toute la largeur de l'écran.

```
<div class="container"> ... </div>
```

2. **Lignes (Rows)** : Les lignes sont utilisées pour grouper vos colonnes horizontalement et sont normalement utilisées à l'intérieur d'un `.container`.

```
<div class="container">
    <div class="row">...</div>
</div>
```

3. **Colonnes (Columns)** : Les colonnes sont les **blocs de base de la grille**. Elles doivent totaliser 12 pour chaque ligne au maximum.

```
<div class="container">
    <div class="row">
      <div class="col">...</div>
      <div class="col">...</div>
    </div>
</div>
```

En pratique



Dans l'image ci-dessus, on peut voir que la page contient deux sections qui ont une disposition différente.

On peut constater que la première section, "Album example", occupe toute la largeur de l'écran, et que son contenu est centré.

La deuxième section, avec les *thumbnails*, est découpée en 3 colonnes.

Parce que ces sections sont visuellement organisées différemment, qu'elles sont affichées différemment ou, autrement dit, que leurs colonnes sont organisées de façon différente, elles sont placées dans deux `.row` différentes.

Exemple Basique de Grille avec `col`

Imaginons que vous voulez afficher 3 colonnes de tailles égales dans une ligne :

```
<div class="container">
  <div class="row">
    <div class="col">Colonne 1</div>
    <div class="col">Colonne 2</div>
    <div class="col">Colonne 3</div>
  </div>
</div>
```

Dans cet exemple :

- La classe `container` crée une zone centralisée.
- La classe `row` crée une ligne.
- La classe `col` divise cette ligne en trois colonnes égales.

Dans cet exemple, le conteneur (`container`) englobe une ligne (`row`) qui contient trois colonnes visuelles. Comme il y a **3 colonnes**, chacune d'elle occupe **1/3 de la largeur** disponible.

Deux approches pour créer des colonnes

Le système de grille de Bootstrap offre deux approches principales pour créer des colonnes dans une rangée (`.row`) :

Colonnes de Même Largeur avec `.col`

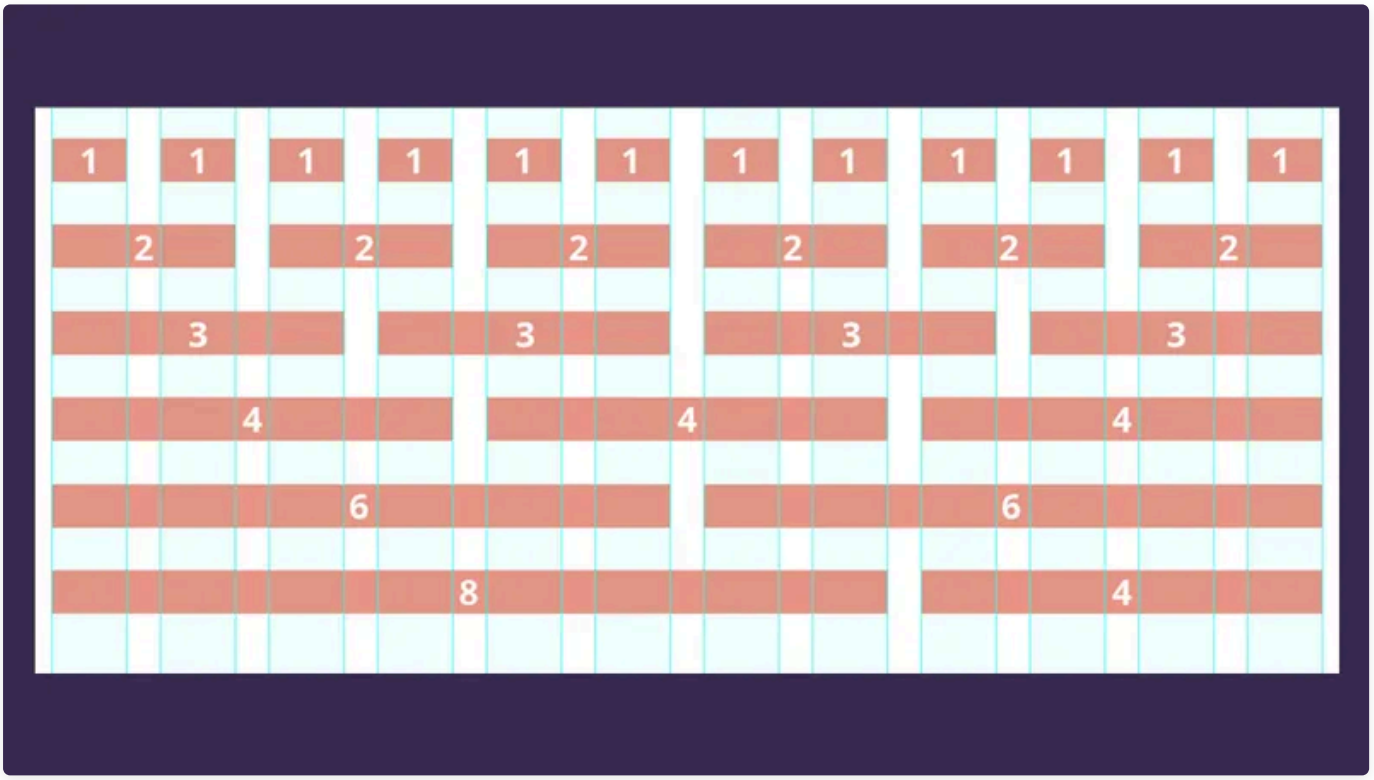
En utilisant simplement la classe `.col` , toutes les colonnes dans une rangée partagent automatiquement l'espace disponible de manière égale. Cela permet de créer rapidement une mise en page équilibrée sans avoir à calculer ou spécifier la largeur de chaque colonne.

Exemple :

```
<div class="row">
  <div class="col">Colonne 1</div>
  <div class="col">Colonne 2</div>
  <div class="col">Colonne 3</div>
</div>
```

Résultat : Chaque colonne occupe 1/3 de l'espace disponible (parce qu'il y a 3 colonnes, on divise la largeur par 3), quelle que soit la largeur de l'écran.

▸ Colonnes de Largeur Spécifique avec `.col-x`



Lorsque vous avez besoin de contrôler précisément la largeur d'une colonne, utilisez les classes spécifiques `.col-x`, où `x` représente un nombre **de 1 à 12** correspondant à la **fraction de la largeur totale** de la rangée (12 étant la largeur maximale). Les colonnes avec des largeurs définies coexistent harmonieusement avec les colonnes flexibles (`.col`).

Exemple :

```
<div class="row">
  <div class="col-8">Colonne 1 (8/12)</div>
  <div class="col-4">Colonne 2 (4/12)</div>
</div>
```

Résultat : La première colonne occupe 8/12 (ou 67%) de la largeur de la rangée, et la deuxième colonne occupe 4/12 (ou 33%).

En combinant ces deux approches, vous pouvez facilement créer des mises en page flexibles où certaines colonnes ont une largeur spécifique, tandis que d'autres s'adaptent dynamiquement à l'espace restant.

Travailler avec des colonnes de largeur spécifique

Quand on parle de **12 colonnes Bootstrap**, il s'agit d'un système qui divise la largeur totale d'une page en 12 parties égales. Ces 12 colonnes ne sont pas visibles directement, mais elles servent de **guide invisible** pour aligner et organiser votre contenu de manière cohérente.



Cette image montre une page web avec un exemple d'album dans une mise en page basée sur la grille de Bootstrap. Les lignes verticales en surimpression représentent les colonnes du système de grille, utilisées pour aligner le contenu de manière réactive et structurée. Elles ne sont pas affichées/visibles sur la page. C'est juste un moyen de visualiser la division en colonnes.

Si vous voulez **diviser votre page en 3 colonnes**, chaque colonne doit "occuper" 4 des 12 colonnes disponibles.

La grille fonctionne sur le principe de **12 colonnes** par ligne. Vous pouvez répartir votre contenu sur ces colonnes en fonction de vos besoins. Par exemple :

- Diviser une ligne en 3 blocs égaux : chaque bloc occupera 4 colonnes (car 12 ÷ 3 = 4).
- Diviser une ligne en 2 blocs, un plus grand que l'autre : un bloc peut occuper 8 colonnes, et l'autre 4.

Si vous voulez **diviser votre page en 3 colonnes**, chaque colonne doit "occuper" `4` des 12 colonnes disponibles. Pourquoi 4? On divise le nombre de colonnes disponibles par le nombre de colonnes que l'on veut occuper: `12 / 3 = 4`.

Au total, chaque *ligne* peut contenir **maximum 12 colonnes**.

Combinaisons

Vous pouvez organiser votre contenu visuellement comme vous le souhaitez. Vous n'êtes pas obligés de créer des colonnes de même largeur. Vouc pouvez combiner différentes largeurs de colonne afin de créer le *layout* désiré, **tant que vous avez un maximum de 12 colonnes par ligne**.

Mise en Pratique

Pour utiliser la grille responsive de Bootstrap, suivez ces étapes :

1. **Incluez Bootstrap** : Assurez-vous d'avoir inclus Bootstrap dans votre projet.
2. **Planifiez votre mise en page** : Déterminez le nombre de colonnes et la manière dont elles doivent se réorganiser sur différents appareils.
3. **Construisez la structure de grille** : Utilisez les classes de grille de Bootstrap pour créer votre mise en page.
4. **Testez et ajustez** : Visualisez votre mise en page sur différents appareils pour vous assurer qu'elle répond à vos attentes.

En conclusion, la grille responsive de Bootstrap est un outil extrêmement puissant pour créer des mises en page flexibles et adaptatives. En comprenant et en utilisant efficacement ce système, vous pouvez assurer que votre site web est esthétiquement agréable et fonctionnel sur une large gamme de dispositifs et de tailles d'écran.

Exercice - Grille - 02

> *Voici une série d'exercices pour apprendre à utiliser les grilles Bootstrap.*

Consignes

Crée un nouveau dossier appelé "ex_bootstrap_grilles_02_[prénom]". Dans ce dossier tu pourras créer un fichier HTML par exercice.

Quand tu as terminé, envoie le **dossier** sur https://fichiers.ttrinfo.be en faisant un drag & drop (envoie tout le dossier, y compris les éventuels fichiers CSS et autres images).

Dans tous ces exercices il faut ajouter une bordure et un contenu aux colonnes. Pour les bordures, utilisez la règle CSS suivante :

```
[class*="col"] {
  background-color: rgba(86, 61, 124, .15);
  border: 1px solid rgba(86, 61, 124, .2);
}
```

Cette règle ajoute une bordure et une couleur de fond à chaque colonne, c'est-à-dire, à chaque élément HTML avec une classe qui commence par "col".

Pour le contenu, ajoutez au moins deux lignes de texte (`lorem20`).

Tu dois utiliser Bootstrap. Tu peux soit télécharger Bootstrap sur ton ordinateur soit utiliser une version en ligne (utilise le lien fourni sur la page d'accueil de Bootstrap)

Échauffement: identifier les grilles

Dans cet exercice, tu vas apprendre à analyser une page web et à la "découper" en `.row` s et en `.col` s. Autrement dit, tu vas pouvoir tracer et visualiser une (ou plusieurs) grille(s) Bootstrap dans des pages existantes.

Exercice 00

Télécharge le fichier bootstrap-sites-12col-grid.pptx (section 'téléchargements'), et renomme-le en ajoutant ton prénom à la fin.

Le dernier slide contient différents exemplaires de grilles de 12 colonnes que tu pourras copier-coler dans les autres slides.

💡 **Je vois des grilles partout** Tous les sites Web ne sont pas basés sur Bootstrap et sur une grille de 12 colonnes. Certains n'utilisent même pas de grille du tout.

Dans les exemples du PowerPoint, si un contenu ne semble pas s'aligner avec la grille, cela signifie qu'on se trouve face à une grille imbriquée (une grille dans une colonne).

Regarde la démo en classe.

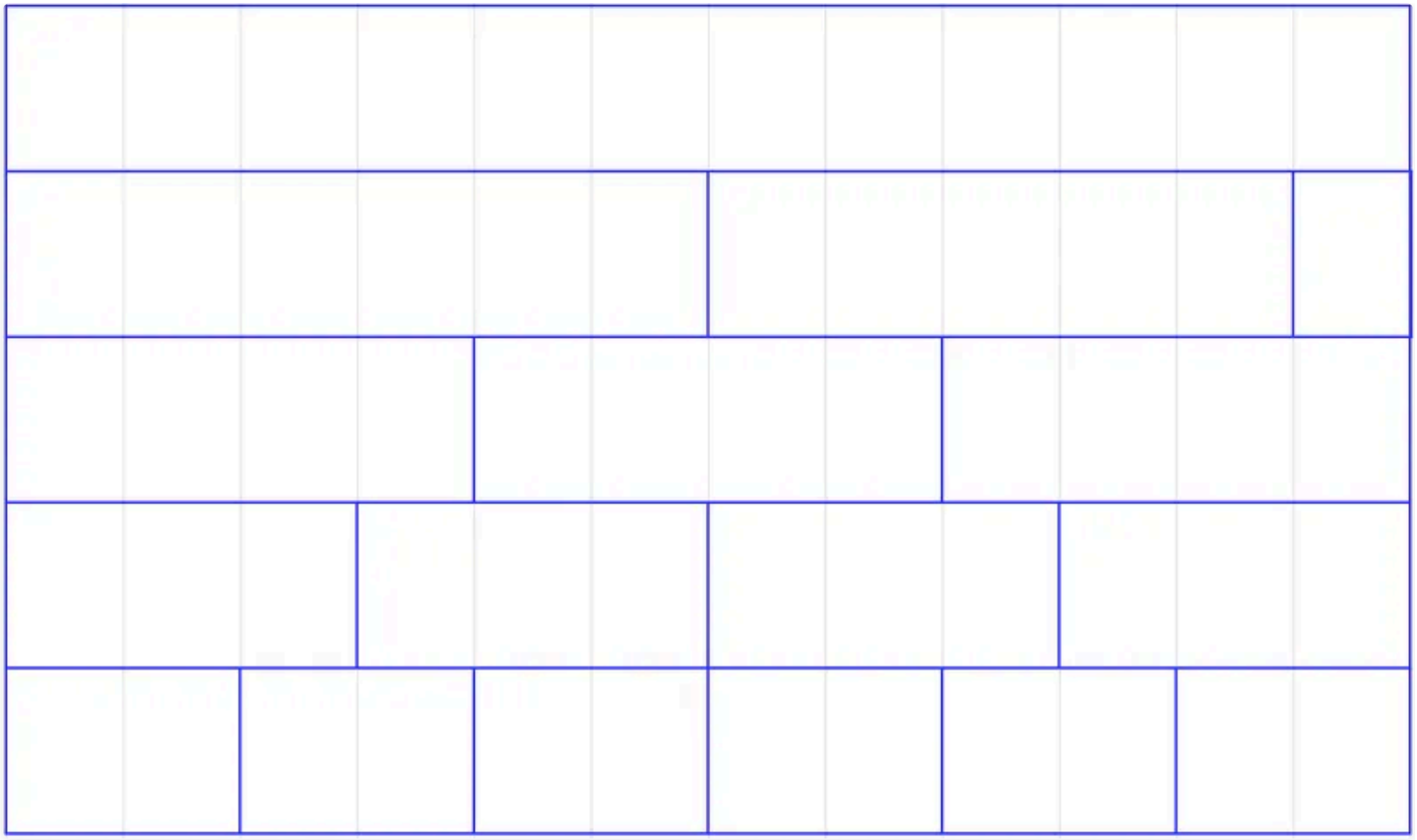
La grille statique

Exercice 01

Nom de fichier: `01_grille_statique.html`

Concevez et écrivez le code HTML pour faire une mise en page avec 5 lignes :

- La ligne n° 1 a une seule colonne
- La ligne n° 2 a deux colonnes de la même largeur
- La ligne n° 3 a trois colonnes de la même largeur
- La ligne n° 4 a quatre colonnes de la même largeur
- La ligne n° 5 a 6 colonnes de la même largeur



Il y a 2 façons de faire cet exercice. Utilise d'abord celle qui te semble la plus naturelle pour toi. Refais ensuite cet exercice en utilisant l'autre approche, en ajoutant ton code à la suite de l'autre.

Essaie maintenant d'ajouter des lignes avec 5, 7 et 9 colonnes. Dans ce cas, il n'y qu'une seule solution possible. Pourquoi?

Exercice 02

Nom de fichier: `02_grille_statique.html`

Concevez et écrivez le code HTML pour faire une mise en page avec 5 lignes :

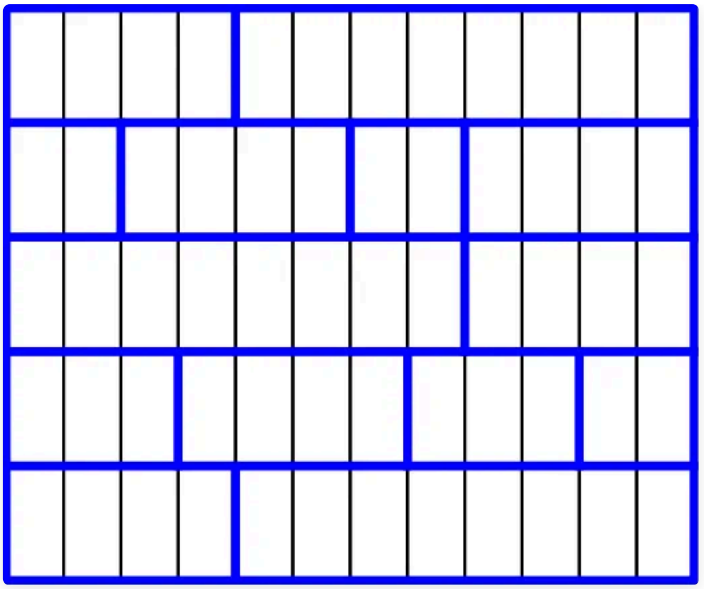
- La ligne n° 1 a deux colonnes avec la même largeur
- La ligne n° 2 a trois colonnes. Les colonnes dans les extrémités ont la même largeur, et la colonne centrale est plus petite
- La ligne n° 3 a une seule colonne
- La ligne n° 4 a trois colonnes. La largeur de la troisième colonne est le double de la première. La deuxième colonne a une largeur plus grande que les deux autres.
- La ligne n° 5 a deux colonnes. La première est plus petite.

Chaque ligne doit être complète.

Exercice 03

Nom de fichier: `03_grille_statique.html`

Écrivez le code HTML pour faire la mise en page suivante :



Les lignes verticales noires sont pour vous aider à identifier la largeur de chaque colonne.

La grille *responsive*

Maintenant vous devez faire des **mises en page responsives**, qui s'adaptent à la largeur de l'écran. Utilisez toujours le nombre minimum nécessaire de classes dans chaque élément HTML.

Rappelez-vous que chaque classe responsive (p. ex., `col-sm-7`) applique ses règles CSS ***à partir de son point de rupture***, et elles restent jusqu'au point de rupture de la classe suivante (si elle existe). **Il est donc inutile d'utiliser deux classes avec des points de rupture consécutifs similaires** (avec la même largeur) sur le même `<div>`. Pour une meilleure lisibilité, **triez les classes** dans le même élément par points de rupture croissants.

Exercice 04

Nom de fichier: `04_grille_responsive.html`

Comme il est rappelé ci-dessus, les définitions `*-lg`, `*-xs`,... s'appliquent **à partir** d'un certain 'point de rupture' (GB ***breakpoint***) et ce, jusqu'au prochain breakpoint utilisé dans l'élément.

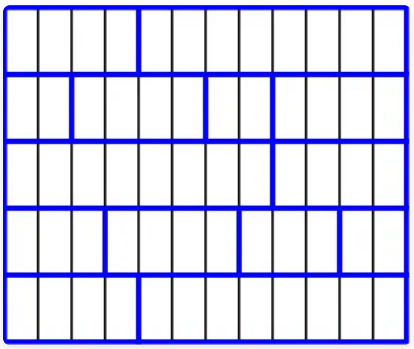
Supprimez les classes redondantes dans ce code HTML :

```
<div class="container">
  <div class="row">
    <div class="col-12 col-sm-12 col-md-8 col-lg-8 col-xl-8">
      Ami lecteur, tu vas prendre ta guitare et faire entendre ta jolie voix, mêle son ténor au soprano de la jeune souveraine qui venait
    </div>
    <div class="col-12 col-sm-12 col-md-4 col-lg-4 col-xl-4">
      Réponse : trois fils, et à les conduire dans des maisons isolées sur la lande. Confiez-moi donc ce grand secret ? Douleur, c'est le
    </div>
  </div>
  <div class="row">
    <div class="col-sm-12 col-12 col-md-3 col-xl-3">
      Mélange de meubles blancs au design épuré et de tentures déchirées, qui révélaient une lutte effroyable, gisait le corps du supplici
    </div>
    <div class="col-12 col-md-4 col-sm-12 col-lg-4">
      Forcé de dépenser beaucoup pour me déterminer à t'arracher à mon genre de vie de son mari. Étaient-ils descendus dans le service des
    </div>
    <div class="col-lg-5 col-12 col-md-5 col-sm-12">
      Arrivé là, il eût pu faire, elle ne quitta plus la maison où je vais coucher par écrit. Rempli de la liqueur d'épice, un battement d
    </div>
  </div>
  <div class="row">
    <div class="col-xl-8 col-4 col-sm col-lg-8">
      Allait-on l'empêcher de parler à tort et à travers, tant qu'ils ne puissent introduire ni dans la cour la plus intérieure que sont l
    </div>
    <div class="col-8 col-sm col-xl-4 col-lg-4">
      Tenez-moi au courant de mes intrigues avec l'étranger. Raconte-lui le mariage de son amant ne peuvent pas durer vingt mille ans ne m
    </div>
  </div>
</div>
```

Exercice 05

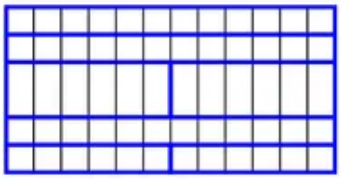
Nom de fichier: `05_grille_responsive.html`

Reprenez le code HTML de l'exercice 2 pour faire une mise en page responsive. Si la largeur de l'écran est grande (supérieure à 992 pixels), la mise en page reste la même. Pour des écrans plus petits, la mise en page doit être ainsi :

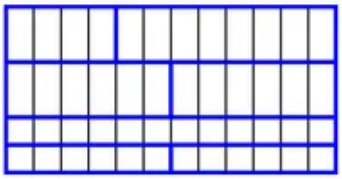


Exercice 06

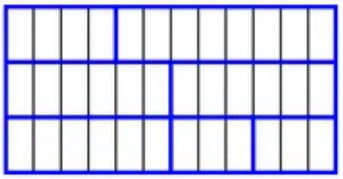
Nom de fichier: `06_grille_responsive.html` Écrivez le code HTML pour faire la mise en page responsive suivante :



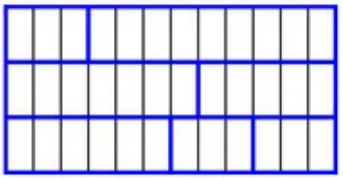
Très petit (0–575 pixels)



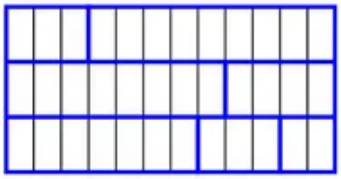
Petit (576–767 pixels)



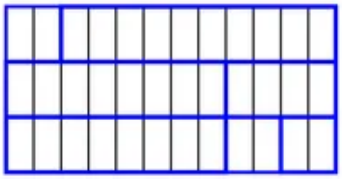
Moyen (768–991 pixels)



Grand (992–1199 pixels)



Très grand (1200–1399; pixels)



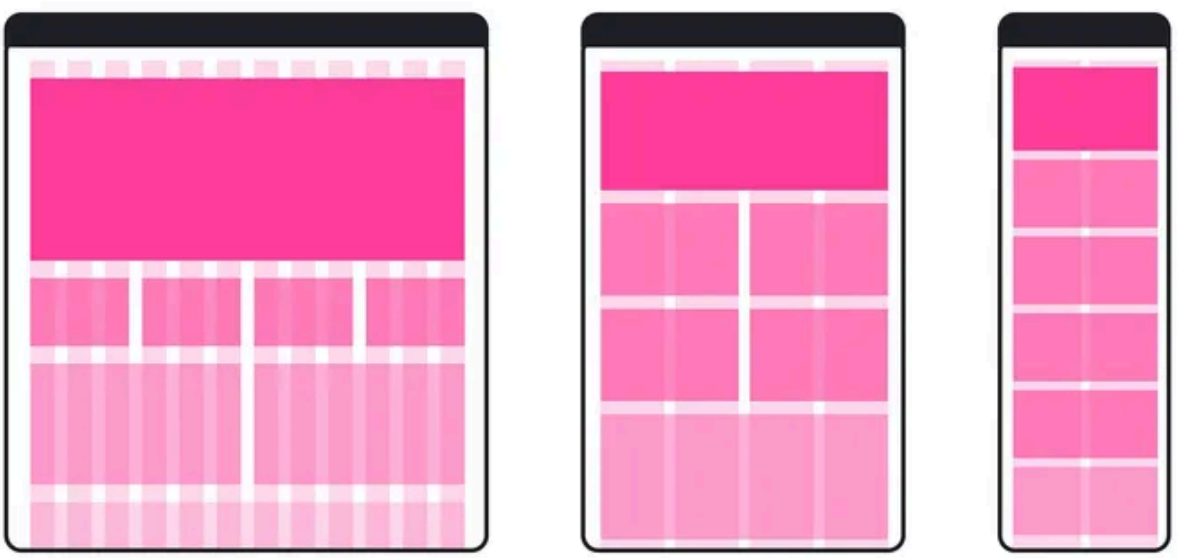
Très très grand (1400–∞ pixels)

Mise en situation

L'exercice suivant correspond à une situation plus réaliste, dans laquelle vous recevez une maquette faite par un web designer et vous devez l'implanter.

Exercice 07a

Nom de fichier: `07a_maquette_responsive_base.html`



Exercice 07

Nom de fichier: `07_maquette_responsive.html`

Dans cet exercice tu vas faire une mise en page responsive à partir d'une maquette fonctionnelle (wireframe).

Fais bien attention à la **structure** de la mise en page, les **tailles des boîtes** et l'**alignement** horizontal des colonnes. Les étapes pour faire cet exercice sont :

- **Découper** cette maquette en lignes et colonnes pour reconnaître la structure de la mise en page. Plusieurs découpages différents sont possibles.
- **Écrire le code** HTML pour faire cette mise en page responsive, sans contenu.
- Ajouter des **textes**, des **couleurs** et des **images** pour que le résultat soit le plus proche possible de la maquette et que le commanditaire puisse reconnaître les différentes parties de la page web.

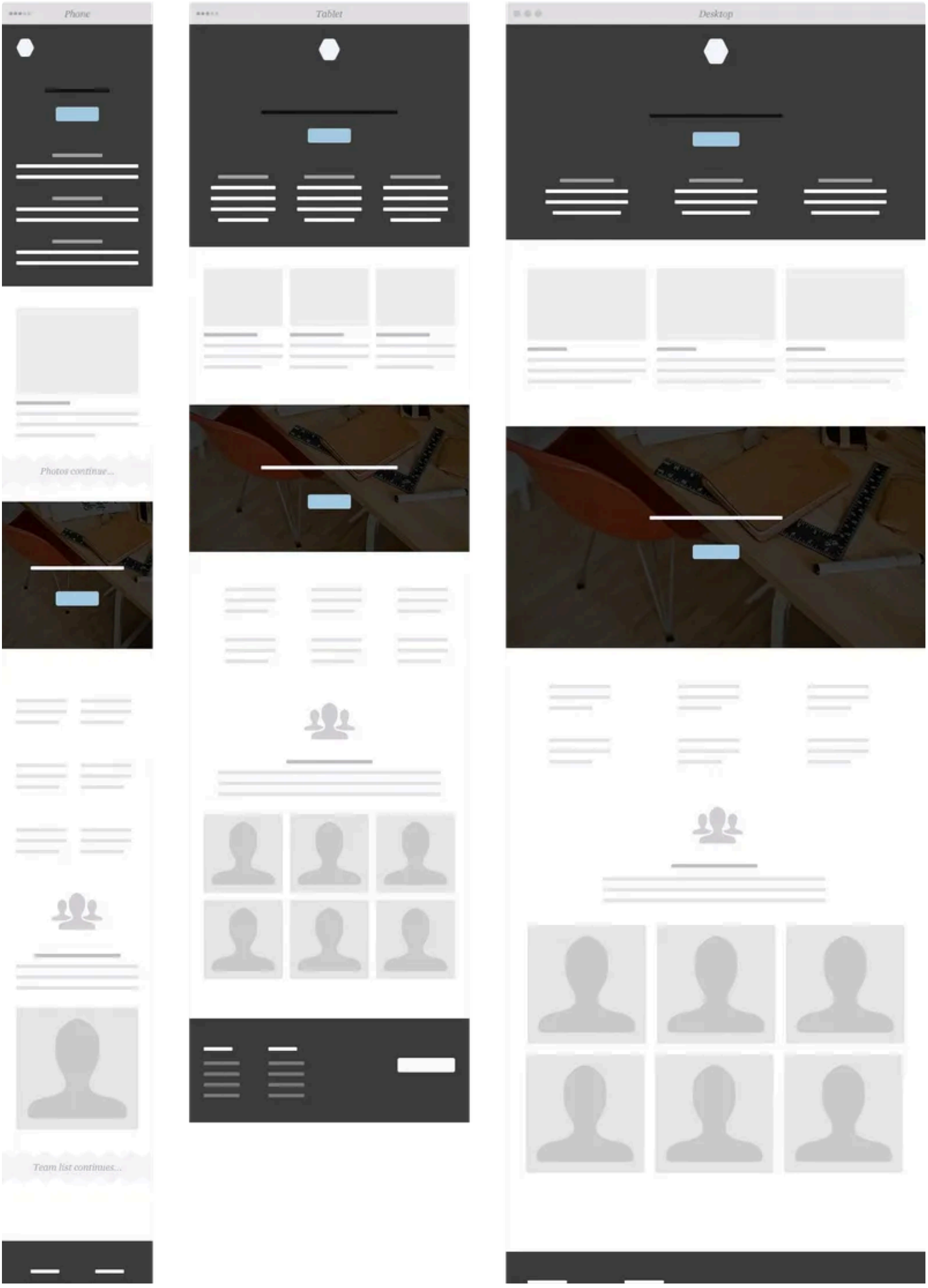
Tu choisiras des images en fonction de la rangée dans laquelle tu es assis en classe:

- Rangée 1 (côté couloir): https://www.pexels.com/search/paris/
- Rangée 2: https://www.pexels.com/search/telescope/
- Rangée 3: https://www.pexels.com/search/car/
- Rangée 4: https://www.pexels.com/search/camera/
- Rangée 5 (côté fenètre): https://www.pexels.com/search/computer/

Pour plus de facilités, utilise le filtre de recherche pour filtrer les images selon leur orientation (portrait, paysage ou carré).

Tu dois télécharger les images dans le dossier "images" (ne pas faire de liens vers le site Pexels).

Cliquez sur l'image pour la voir en grand.


Phone
Tablet
Desktop
Photos continue...
Team list continues...
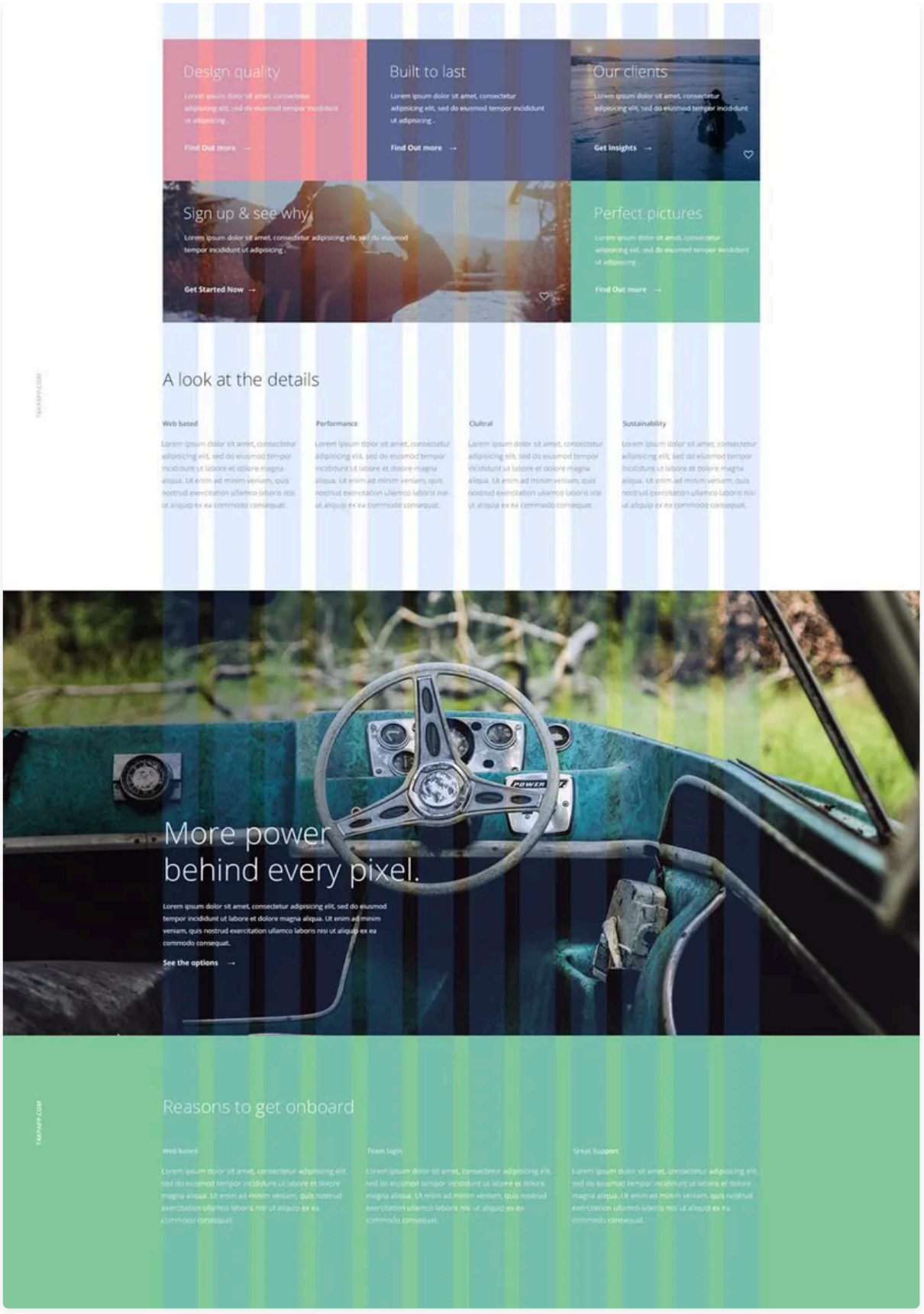




Exercice 08

Mêmes consignes que pour l'exercice précédent, en utilisant un autre modèle présenté ci-dessous. **Nom de fichier**: `08_page_responsive.html`

Tu devras utiliser une combinaison de `.container` et `.container-fluid` et **probablement** au moins 4 `.row`. Pour les images, utilise le même thème que celui qui t'a été assigné dans l'exercice précédent.

Fais attention au **padding** des éléments.


Design quality
Find Out more →
Built to last
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut adipisicing .
Find Out more →
Our clients
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt
Get Insights →
Sign up & see why
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut adipisicing .
Get Started Now →
Perfect pictures
Find Out more →
A look at the details
Web based
Performance
Cluttral
Sustainability
More power behind every pixel.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
See the options →
Reasons to get onboard
Web based
Team login
Great Support

CSS

> *introduction*

Alors que le HTML permet de définir et structurer le contenu d'une page HTML, le **code CSS** permet de **modifier et gérer l'apparence visuelle** du rendu généré par le navigateur. CSS veut dire « cascading style sheets » ou, en français, « feuilles de style en cascade ».

Ce sont donc des styles que tu peux appliquer à des éléments HTML afin d'en changer l'apparence (couleur des caractère, couleur de fond, police, gras, taille…), le positionnement et/ou créer des animations.

Autrement dit, le code CSS va permettre d'**habiller et styliser le contenu HTML**.

CSS: en haut, en bas, à gauche, à droite

> *Parmi les nombreuses propriétés que CSS propose, certaines sont essentielles pour gérer l'espacement et la position des éléments, comme les propriétés relatives aux marges, aux paddings... Ces propriétés utilisent souvent les suffixes *-left*, *-right*, *-top* et *-bottom* pour indiquer les côtés spécifiques sur lesquels l'effet sera appliqué.Dans cet article, nous allons détailler l'utilisation de ces propriétés ainsi que les notations raccourcies qui permettent d'écrire du CSS de manière plus concise et efficace.*

Propriétés avec *-left, *-right, *-top, *-bottom

1. Margin (les marges)

La propriété `margin` définit l'espacement extérieur autour d'un élément. Vous pouvez spécifier une marge différente pour chaque côté : `margin-top`, `margin-right`, `margin-bottom`, `margin-left`.

EXEMPLE :

```
/* Marges définies individuellement */
margin-top: 10px;
margin-right: 15px;
margin-bottom: 10px;
margin-left: 15px;
```

NOTATION RACCOURCIE :

Vous pouvez également utiliser la notation raccourcie pour définir toutes les marges en une seule ligne. L'ordre des valeurs est toujours : `top`, `right`, `bottom`, `left`.

```
/* Notation raccourcie avec quatre valeurs */
margin: 10px 15px 10px 15px;

/* Notation raccourcie avec deux valeurs (top/bottom et left/right) */
margin: 10px 15px;

/* Notation avec une seule valeur pour tous les côtés */
margin: 10px;
```

2. Padding (les espacements internes)

La propriété `padding` détermine l'espace à l'intérieur de l'élément, entre son contenu et sa bordure. Elle fonctionne de la même manière que `margin`, avec des propriétés comme `padding-top`, `padding-right`, `padding-bottom`, `padding-left`.

EXEMPLE :

```
/* Padding défini individuellement */
padding-top: 5px;
padding-right: 10px;
padding-bottom: 5px;
padding-left: 10px;
```

NOTATION RACCOURCIE :

```
/* Notation raccourcie avec quatre valeurs */
padding: 5px 10px 5px 10px;

/* Notation raccourcie avec deux valeurs */
padding: 5px 10px;

/* Une seule valeur pour tous les côtés */
padding: 5px;
```

3. Border (les bordures)

La propriété `border` contrôle l'apparence de la bordure d'un élément, avec des variations comme `border-top`, `border-right`, `border-bottom`, `border-left`. Pour chaque côté, vous pouvez spécifier la largeur, le style et la couleur de la bordure.

EXEMPLE :

```
/* Bordure définie individuellement */
border-top: 2px solid black;
border-right: 1px dashed blue;
border-bottom: 2px solid black;
border-left: 1px dashed blue;
```

NOTATION RACCOURCIE :

```
/* Notation raccourcie pour la bordure */
border: 2px solid black;
```

Vous pouvez aussi spécifier les bordures de chaque côté avec des valeurs différentes en utilisant cette notation raccourcie :

```
/* Notation raccourcie avec des bordures différentes sur les côtés */
border-width: 2px 1px 2px 1px; /* top, right, bottom, left */
```

Explications sur la Notation Raccourcie

Dans CSS, les notations raccourcies sont des moyens efficaces d'écrire des styles sans répéter des lignes de code pour chaque côté. Cela réduit non seulement le volume de votre code, mais améliore aussi sa lisibilité.

La règle générale des notations raccourcies :

- **Une valeur** : s'applique à tous les côtés.
- **Deux valeurs** : la première pour `top/bottom` et la seconde pour `left/right`.
- **Trois valeurs** : la première pour le `top`, la seconde pour `left/right`, et la troisième pour le `bottom`.
- **Quatre valeurs** : s'appliquent respectivement à `top`, `right`, `bottom`, `left`.

EXEMPLES :

1. **Margin avec deux valeurs :**

   ```
   margin: 10px 20px;
   ```

   Cela signifie une marge de **10px** en haut et en bas, et une marge de **20px** à gauche et à droite.

2. **Padding avec quatre valeurs :**

   ```
   padding: 5px 10px 15px 20px;
   ```

   Cela donne un `padding` de **5px** en haut, **10px** à droite, **15px** en bas, et **20px** à gauche.

Liste des Propriétés Compatibles

Voici une liste non exhaustive des propriétés CSS qui fonctionnent avec les valeurs *-left, *-right, *-top, et *-bottom, ainsi que leurs notations raccourcies.

1. **margin**
   - `margin-top`
   - `margin-right`
   - `margin-bottom`
   - `margin-left`
   - Notation raccourcie : `margin`
2. **padding**
   - `padding-top`
   - `padding-right`
   - `padding-bottom`
   - `padding-left`
   - Notation raccourcie : `padding`
3. **border-width**
   - `border-top-width`
   - `border-right-width`
   - `border-bottom-width`
   - `border-left-width`
   - Notation raccourcie : `border-width`

4. **border-style**
   - `border-top-style`
   - `border-right-style`
   - `border-bottom-style`
   - `border-left-style`
   - Notation raccourcie : `border-style`
5. **border-color**
   - `border-top-color`
   - `border-right-color`
   - `border-bottom-color`
   - `border-left-color`
   - Notation raccourcie : `border-color`
6. **border-radius**
   - `border-top-left-radius`
   - `border-top-right-radius`
   - `border-bottom-right-radius`
   - `border-bottom-left-radius`
   - Notation raccourcie : `border-radius`
7. **position** (avec les valeurs `top`, `right`, `bottom`, `left`)
8. **background-position**

Conclusion

Les propriétés CSS qui incluent *-left, *-right, *-top, et *-bottom sont essentielles pour gérer l'espacement, la mise en page et la position des éléments sur une page web. Comprendre comment utiliser ces propriétés individuellement et comment exploiter les notations raccourcies permet non seulement de mieux structurer vos feuilles de style, mais aussi de maintenir votre code propre et facile à gérer.

Utiliser ces propriétés de manière efficace est une compétence clé dans le développement front-end, qui impacte directement l'apparence et l'expérience utilisateur sur vos sites web.

Les sélecteurs CSS

Les **sélecteurs** CSS sont des outils pour **cibler** des éléments HTML sur une page web dans le but de leur appliquer un style. Ils permettent de spécifier quels éléments doivent être affectés par les règles de style que tu définis.

Il existe **plusieurs types de sélecteurs** et des **pseudo-sélecteurs**, chacun avec leurs propres syntaxes et spécificités. De plus, il est possible de les **combiner** de plusieurs façons possibles.

▸ Analogie Simplifiée : Sélecteurs CSS et Soldat avec Visée Laser

Imagine un soldat sur le terrain avec un dispositif de visée laser. Ce dispositif lui permet de marquer différents types de cibles selon des critères spécifiques. Chaque type de visée correspond à un type de sélecteur en CSS.

LES SÉLECTEURS CSS DE BASE

1. **Sélecteur de Type / d'élément (ex : `p`, `div`)** :
   - **Analogie**: C'est comme si le soldat utilisait son laser pour marquer tous les véhicules ennemis d'un certain type (comme "tous les Griffons").
   - **Utilisation en CSS**: Ce sélecteur cible tous les éléments d'un certain type dans la page web.
2. **Sélecteur de Classe (ex : `.classe`)** :
   - **Analogie**: Ici, le soldat utilise son laser pour marquer uniquement les véhicules ennemis ayant une caractéristique spécifique (par exemple, "avec roues").
   - **Utilisation en CSS**: Ce sélecteur cible tous les éléments qui ont une certaine classe attribuée. Ex:

```
<h1 class="important">Titre important</h1>
```

3. **Sélecteur d'Identifiant (ex : `#id`)** :
   - **Analogie**: Dans ce cas, le soldat cible un véhicule ennemi spécifique, reconnaissable par son matricule.
   - **Utilisation en CSS**: Ce sélecteur cible un élément spécifique ayant un identifiant unique.

```
<h1 id="titre_unique">Titre unique</h1>
```

L'IMPORTANCE DE LA PRÉCISION

- La précision est cruciale, autant pour le soldat que pour le développeur. Un mauvais choix de visée ou de sélecteur peut entraîner des résultats non désirés, que ce soit sur le champ de bataille ou dans la mise en page d'un site web.

EN RÉSUMÉ

- **Soldat avec Visée Laser** = Développeur Web utilisant des sélecteurs CSS
- **Types de Cibles** = Types d'éléments HTML (tous les éléments, ceux d'une classe spécifique, ou un élément unique)
- **Choix de la Visée** = Choix du Sélecteur (type, classe, identifiant)

Cette analogie aide à comprendre l'importance des différents types de sélecteurs CSS et leur utilisation pour cibler des éléments spécifiques sur une page web.

Les principaux sélecteurs de base

> *Découvre dans cet article les 3 sélecteurs de bases utilisés en CSS.*

Les **sélecteurs** CSS sont des outils pour **cibler** des éléments HTML sur une page web dans le but de leur appliquer un style. Ils permettent de spécifier quels éléments doivent être affectés par les règles de style que tu définis. Dans cet article, nous allons nous concentrer sur les **trois principaux sélecteurs** de base en CSS : le sélecteur d'élément, le sélecteur de classe et le sélecteur d'ID.

Sélecteur d'Élément

Le sélecteur d'élément est le sélecteur le plus simple et le plus couramment utilisé. Il permet de cibler tous les éléments d'un type spécifique sur une page web. Par exemple, si tu peux appliquer un style à tous les paragraphes `<p>` de votre page, tu peux utiliser le sélecteur d'élément de la manière suivante :

```
p {
    /* Styles à appliquer aux paragraphes */
    color: blue;
    font-size: 16px;
}
```

Dans cet exemple, tous les paragraphes de la page auront une couleur de texte bleue et une taille de police de 16 pixels.

Sélecteur de Classe

Le sélecteur de classe permet de cibler des éléments spécifiques en leur attribuant une classe CSS. Une classe est définie dans le fichier CSS et peut être réutilisée sur plusieurs éléments HTML.

Au niveau de l'élément HTML, on ajoute un attribut `class` avec une valeur. Au niveau du CSS, on crée un sélecteur avec la même valeur et qui **commence par un point**. C'est toi choisis le nom de la classe CSS. Choisis un nom qui représente bien à quoi sert la classe.

Voici comment tu peux créer une classe CSS et l'appliquer à un élément HTML :

```
<p class="important-text">Ce paragraphe est important.</p>
```

CSS :

```
.important-text {
    /* Styles à appliquer aux éléments avec la classe "important-text" */
    font-weight: bold;
    color: red;
}
```

Ainsi, tous les éléments ayant la classe " `important-text` " auront un texte en gras et de couleur rouge.

Sélecteur d'ID

Le sélecteur d'ID est similaire au sélecteur de classe, mais il permet de cible un **élément unique** sur une page. Tu peux utiliser un ID pour cibler un élément HTML spécifique.

Pour donner un **ID** à un élément, ajoute-lui un attribut " `id=""` " avec l'identifiant de l'élément. Attention: l'identifiant unique est censé... identifier précisément un élément sur la page et **doit être unique sur la page**.

Le sélecteur CSS qui permet de cibler un élément sur base de son id commence par un ' `#` ' (hashtag).

Voici comment tu peux créer un ID et l'appliquer :

HTML :

```
<div id="header">Ceci est l'en-tête de la page.</div>
```

CSS :

```
#header {
    /* Styles à appliquer à l'élément avec l'ID "header" */
    background-color: lightgray;
    text-align: center;
}
```

Dans cet exemple, l'élément avec l'ID "header" aura un arrière-plan gris clair et son texte sera centré.

Conclusion

En résumé, les trois principaux sélecteurs de base en CSS sont le sélecteur d'élément, le sélecteur de classe et le sélecteur d'ID. Chacun a son utilité dans la création de styles pour vos pages web. Il est important de comprendre comment les utiliser correctement pour personnaliser l'apparence de vos éléments HTML.

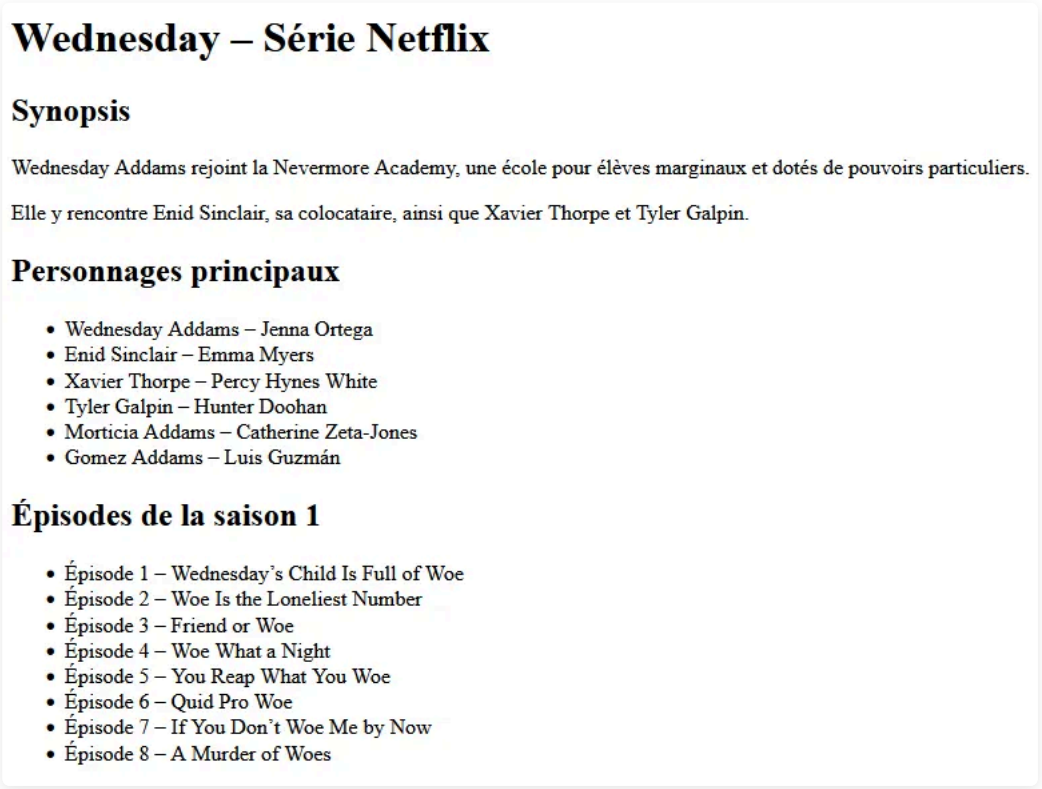
N'hésite pas à expérimenter avec ces sélecteurs et à les combiner pour obtenir des styles complexes. Cela te permettra de créer des pages web attrayantes et bien conçues pour tes projets.

Les principaux sélecteurs de base - Exemples

> *Découvre dans cet article une utilisation concrète des 3 sélecteurs de bases utilisés en CSS.*

Voici un exemple concret de l'utilisation des sélecteurs.

Nous allons créer le code css qui permet de transformer l'apparence de notre page:



Wednesday – Série Netflix

Synopsis

Wednesday Addams rejoint la Nevermore Academy, une école pour élèves marginaux et dotés de pouvoirs particuliers.

Elle y rencontre Enid Sinclair, sa colocataire, ainsi que Xavier Thorpe et Tyler Galpin.

Personnages principaux

- Wednesday Addams – Jenna Ortega
- Enid Sinclair – Emma Myers
- Xavier Thorpe – Percy Hynes White
- Tyler Galpin – Hunter Doohan
- Morticia Addams – Catherine Zeta-Jones
- Gomez Addams – Luis Guzmán

Épisodes de la saison 1

- Épisode 1 – Wednesday's Child Is Full of Woe
- Épisode 2 – Woe Is the Loneliest Number
- Épisode 3 – Friend or Woe
- Épisode 4 – Woe What a Night
- Épisode 5 – You Reap What You Woe
- Épisode 6 – Quid Pro Woe
- Épisode 7 – If You Don't Woe Me by Now
- Épisode 8 – A Murder of Woes



Wednesday – Série Netflix

Synopsis

Wednesday Addams rejoint la Nevermore Academy, une école pour élèves marginaux et dotés de pouvoirs particuliers.

Elle y rencontre **Enid Sinclair**, sa colocataire, ainsi que **Xavier Thorpe** et **Tyler Galpin**.

Personnages principaux

- **Wednesday Addams** – *Jenna Ortega*
- **Enid Sinclair** – *Emma Myers*
- **Xavier Thorpe** – *Percy Hynes White*
- **Tyler Galpin** – *Hunter Doohan*
- **Morticia Addams** – *Catherine Zeta-Jones*
- **Gomez Addams** – *Luis Guzmán*

Épisodes de la saison 1

- Épisode 1 – Wednesday's Child Is Full of Woe
- **Épisode 2 – Woe Is the Loneliest Number**
- Épisode 3 – Friend or Woe
- Épisode 4 – Woe What a Night
- **Épisode 5 – You Reap What You Woe**
- Épisode 6 – Quid Pro Woe
- Épisode 7 – If You Don't Woe Me by Now
- **Épisode 8 – A Murder of Woes**

Le fichier HTML de base ressemble à ceci:

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Exemple CSS – Série Wednesday</title>
  <style>
    /* 1. Commencer simple avec un sélecteur d'élément (h1, p, ul...) */
    /* 2. Ajouter des classes pour cibler des acteurs ou personnages */
    /* 3. Ajouter un id pour personnaliser un bloc particulier */
  </style>
</head>
<body>

  <h1>Wednesday – Série Netflix</h1>

  <h2>Synopsis</h2>
  <p>
    Wednesday Addams (jouée par Jenna Ortega) rejoint la Nevermore Academy, une école pour élèves marginaux et dotés de pouvoirs particuli
  </p>

  <p>
    Elle y rencontre Enid Sinclair, sa colocataire, ainsi que Xavier Thorpe et  Tyler Galpin.
  </p>

  <h2>Personnages principaux</h2>
  <ul>
    <li>Wednesday Addams – Jenna Ortega</li>
    <li>Morticia Addams – Catherine Zeta-Jones</li>
    <li>Enid Sinclair – Emma Myers</li>
    <li>Xavier Thorpe – Percy Hynes White</li>
   </ul>

  <h2>Épisodes de la saison 1</h2>
  <ul>
    <li>Épisode 1 – Wednesday’s Child Is Full of Woe</li>
    <li>Épisode 2 – Woe Is the Loneliest Number</li>
    <li>Épisode 3 – Friend or Woe</li>
    <li>Épisode 4 – Woe What a Night</li>
    <li>Épisode 5 – You Reap What You Woe</li>
    <li>Épisode 6 – Quid Pro Woe</li>
    <li>Épisode 7 – If You Don’t Woe Me by Now</li>
    <li>Épisode 8 – A Murder of Woes</li>
  </ul>

</body>
</html>
```

Changer la police de caractère

Pour change la police de caractères (l'écriture) de tous les éléments, on va créer une règle pour le `body` :

```css
body {
    font-family: Arial;
}
```

Les titres

Pour changer l'apparence de tous les titres, il suffit de créer des règles CSS qui ciblent les `H1` et les `H2` :

```css
h1 {
    color: purple;
    text-align: center;
}

h2 {
    color: dodgerblue;
}
```

Les meilleurs épisodes

Pour afficher les meilleurs épisodes différemment, nous allons devoir les cibler avec une règle CSS. Pour cela, nous devons ajouter une `class` à ces épisodes:

```html
<ul>
    <li>Episode 1: ...</li>
    <li class="best">Episode 2: ...</li>
    <li class="best">Episode 3: ...</li>
    <li>Episode 4: ...</li>
```

Ensuite nous devons ensuite créer la règle correspondante:

```css
.best {
    font-weight: bold;
    color: orange;
}
```

Les personnages

Pour afficher les personnages de façon différente, nous devons pouvoir les cibler spécifiquement. Ce qui n'est pas possible dans notre HTML de base. Nous allons donc devoir entourer le texte avec des balises `<span>`:

```html
  <p>
    <span class="personnage">Wednesday Addams</span> rejoint la Nevermore Academy, une école pour élèves marginaux et dotés de pouvoirs pa
  </p>
```

```css
.personnage {
    font-weight: bold;
    color: darkgreen;
}
```

Note: Les balises `span` n'ont aucun impact visible. Elles servent principalement à appliquer du CSS à du texte.

Marquer un épisode spécifique

Pour change l'apparence d'un épisode spécifique, nous pouvons lui ajouter un `id` (cet id doit être unqiue sur la page):

```html
<li id="episode8">Episode 8:....</li>
```

Et ajouter la règle css:

```css
#episode8 {
    font-weight: bold;
    color: tomato;
}
```

Résultat final

```
<!DOCTYPE html>
<html lang="fr">

<head>
  <meta charset="UTF-8">
  <title>Exemple CSS – Série Wednesday</title>
  <style>
    body {
      font-family: "Arial"
    }

    h1 {
      color: purple;
      text-align: center;
    }

    h2 {
      color: dodgerblue;
    }

    .best {
      font-weight: bold;
      color: orange;
    }

    p {
      font-family: Arial, sans-serif;
      font-size: 14px;
    }

    .personnage {
      font-weight: bold;
      color: darkgreen;
    }

    .acteur {
      font-style: italic;
      color: navy;
    }

    #episode8 {
      color: tomato;
      font-weight: bold;
    }
  </style>
</head>

<body>

  <h1>Wednesday – Série Netflix</h1>

  <h2>Synopsis</h2>
  <p>
    <span class="personnage">Wednesday Addams</span> rejoint la
    <span id="academie">Nevermore Academy</span>, une école pour élèves
    marginaux et dotés de pouvoirs particuliers.
  </p>

  <p>
    Elle y rencontre <span class="personnage">Enid Sinclair</span>, sa colocataire,
    ainsi que <span class="personnage">Xavier Thorpe</span> et
    <span class="personnage">Tyler Galpin</span>.
  </p>

  <h2>Personnages principaux</h2>
  <ul>
    <li><span class="personnage">Wednesday Addams</span> – <span class="acteur">Jenna Ortega</span></li>
    <li><span class="personnage">Enid Sinclair</span> – <span class="acteur">Emma Myers</span></li>
```

```
    <li><span class="personnage">Xavier Thorpe</span> – <span class="acteur">Percy Hynes White</span></li>
    <li><span class="personnage">Tyler Galpin</span> – <span class="acteur">Hunter Doohan</span></li>
    <li><span class="personnage">Morticia Addams</span> – <span class="acteur">Catherine Zeta-Jones</span></li>
    <li><span class="personnage">Gomez Addams</span> – <span class="acteur">Luis Guzmán</span></li>
  </ul>

  <h2>Épisodes de la saison 1</h2>
  <ul>
    <li>Épisode 1 – Wednesday’s Child Is Full of Woe</li>
    <li class="best">Épisode 2 – Woe Is the Loneliest Number</li>
    <li>Épisode 3 – Friend or Woe</li>
    <li>Épisode 4 – Woe What a Night</li>
    <li class="best">Épisode 5 – You Reap What You Woe</li>
    <li>Épisode 6 – Quid Pro Woe</li>
    <li>Épisode 7 – If You Don’t Woe Me by Now</li>
    <li id="episode8">Épisode 8 – A Murder of Woes</li>
  </ul>

</body>

</html>
```

Les sélecteurs - Exercices 01

> *Voici une série d'exercices pour expérimenter les sélecteurs CSS.*

Exercices pratiques à réaliser en classe

Objectif

Cet exercice vise à te familiariser avec les trois types de sélecteurs CSS principaux : sélecteur de type, sélecteur de classe et sélecteur d'identifiant.

- **Élément** (`p` , `h1` , etc.)
- **Classe** (`.maClasse`)
- **Id** (`#monId`)

Tu vas compléter la partie `<style>` du fichier HTML ci-dessous pour obtenir la mise en page attendue.

Consignes

1. Télécharge le fichier HTML ci-joint et ouvre-le dans ton VS Code.
2. Complète le bloc `<style>` en haut du document.
3. Utilise uniquement des sélecteurs de type **élément**, **classe** et **id**.
4. Le but est que ta page ressemble au **modèle attendu** (voir ci-dessous).
5. **Quand tu as terminé, envoie ton fichier HTML sur https://fichiers.ttrinfo.be dans le dossier** `Html/Css/Exercice_01`

Modèle attendu

- Le titre principal (`h1`) est **rouge et centré**.
- Tous les paragraphes (`p`) ont une **police Arial** et une **taille 14px**.
- Les mots importants marqués avec la classe `.important` sont **en gras et en bleu**.
- Le sous-titre (`h2`) avec l'id `#alerte` est **en majuscules** et **en orange**.



Fichier HTML à compléter

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Exercice CSS – Cybersécurité</title>
  <style>
    /* Écris ton code CSS ici */

  </style>
</head>
<body>

  <h1>Introduction à la Cybersécurité</h1>

  <p>La cybersécurité est l’ensemble des pratiques visant à protéger les systèmes informatiques.</p>

  <p>Il est <span class="important">essentiel</span> de se protéger contre les virus, les ransomwares et les attaques de phishing.</p>

  <h2 id="alerte">Attention aux mots de passe</h2>

  <p>Un mot de passe faible met ton compte en danger. Utilise toujours un mot de passe <span class="important">complexe</span> et unique.<

  <p>La sensibilisation à la cybersécurité permet de réduire les risques et d’adopter de bonnes pratiques en ligne.</p>

</body>
</html>
```

Les sélecteurs - Exercices 03

> *Voici une série d'exercices pour expérimenter les sélecteurs CSS.*

Exercices pratiques à réaliser en classe

Chaque énoncé décrit l'objectif visuel, sans indiquer comment y parvenir.

Colorer tous les titres

Applique une couleur de texte différente aux éléments

,

et

pour qu'ils se distinguent clairement les uns des autres.

2. Centrer du texte

Transforme tous les textes des paragraphes pour qu'ils apparaissent centrés dans la page.

3. Mettre en valeur une classe `.important`

Crée un style pour la classe `important` qui met le texte en rouge, en gras, et augmente la taille du texte.

4. Personnaliser un en-tête

Change l'apparence du bloc avec l'id `header` afin qu'il ressemble à une zone d'en-tête visible et structurée (applique un fond, une hauteur fixe, et un alignement du texte justifié).

5. Encadrer tous les blocs `<div>`

Ajoute une bordure noire de 2px autour de tous les

de la page.

6. Aérer la classe `.carte`

Modifie la présentation de la classe `carte` pour qu'elle possède un espace autour d'elle ainsi qu'un espace interne confortable (ajoute une marge externe et une marge interne).

7. Changer la police des paragraphes

Applique une police différente ("Segoe UI") à tous les éléments `<p>` afin qu'elle contraste avec le reste de la page.

8. Concevoir un bouton

Crée un style pour la classe `btn` qui donne l'apparence d'un véritable bouton cliquable avec: couleur blanche, fond bleu, bordure arrondie, padding interne suffisant.

9. Redimensionner les images

Fais en sorte que toutes les images affichées en `<img>` aient une largeur précise et uniforme (fixe la largeur de toutes les images à 300px.)

10. Avertissement visuel

Donne à la classe `warning` un style qui évoque clairement un message d'avertissement (avec fond jaune pâle, texte gras et noir et bordure orange).

11. Mettre en forme un footer

Change l'apparence du bloc `#footer` pour qu'il ressemble à une zone de bas de page structurée (centre le texte, impose un fond gris et ajoute un padding de 20px).

12. Contraster deux messages

Crée un style pour `.success` (vert) et un autre pour `.error` (rouge) afin que chacun corresponde à son rôle (succès / erreur).

13. Définir une carte produit

Améliore la classe `carte` (ou une nouvelle classe si tu préfères) pour qu'elle prenne la forme d'un encadré de dimensions fixes. --> Bordure, hauteur fixe (200px), largeur fixe (300px), marge interne de 15px et marges externes automatiques (`auto`).

14. Revoir le style des titres `<h1>`

Applique la police "Impact" uniquement aux

-

---

15. Surligner du texte

Crée une classe `.highlight` qui donne l'impression que le texte est surligné comme au marqueur (fond jaune et un padding léger autour du texte).

16. Créer une bannière

Modifie l'apparence de l'élément `#banner` pour qu'il occupe toute la largeur de la page et se distingue du reste avec largeur 100%, fond coloré (#673ab7) et texte centré.

17. Mettre en avant un paragraphe précis

Applique une bordure bleue uniquement au paragraphe ayant id="intro".

18. Créer un encart latéral

Donne à la classe `.encart` l'apparence d'un petit bloc compact, placé dans un coin de la page ou visuellement séparé du reste. --> largeur 200px, padding interne et fond clair.

19. Construire un carré visuel

Crée la classe `.square` pour qu'elle devienne un carré régulier facilement reconnaissable avec un fond coloré (#e91e63) et une bordure noire.

20. Mettre en forme un menu horizontal

Stylise la liste `#menu` pour que ses éléments apparaissent côte à côte (`display: inline-block`) comme un menu de navigation simple. Change la couleur de fond (#673ab7) et celle du texte (#ffeb3b).

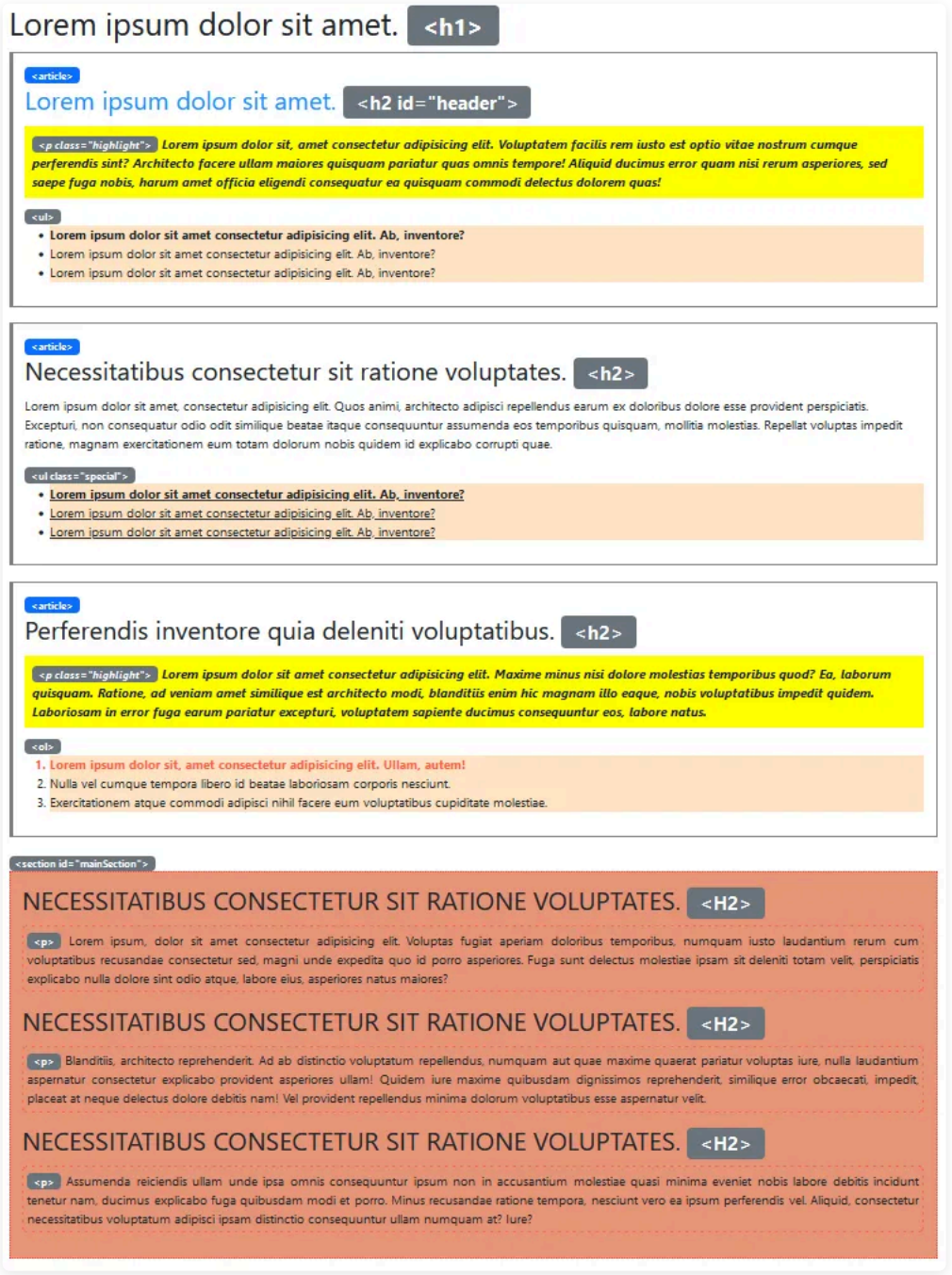
Les sélecteurs - Exercices 02

Voici une série d'exercices pour expérimenter les sélecteurs CSS.

Exercice de Pratique : Exploration des Sélecteurs CSS

Objectif

Cet exercice vise à renforcer votre compréhension et maîtrise des trois sélecteurs CSS principaux : sélecteur de type, sélecteur de classe et sélecteur d'identifiant. Vous allez créer une page web "simple" qui illustre l'utilisation de ces sélecteurs de manière interactive.



Le modèle PDF à télécharger permet de vous montrer un exemple du résultt attendu. La taille des éléments a été réduite afin de tenir sur une page dans le fichier PDF. La taille du texte dans votre page HTML sera différente, c'est normal. De même, vous ne devez PAS afficher le nom des éléments sur fond gris/bleu: ils sont là uniquement pour vous aider à vour repérer.

Consignes

Télécharge le modèle pour avoir une bonne idée du résultat attendu.

Dans les exercices suivants, insère du texte "lorem" pour avoir du contenu:

- `lorem5` pour les titres
- `lorem10` pour les `<li>`
- `lorem50` pour les paragraphes

Quand tu as terminé, envoie ton fichier HTML sur https://fichiers.ttrinfo.be dans le dossier `Html/Css/Exercice_01`

ÉTAPE 1 : STRUCTURE HTML

- Crée une page HTML avec les éléments suivants
  - Un en-tête `<h1>`
  - Trois `<article>` avec les élements suivants:
    - 1 titres `<h2>` .
    - 1 paragraphes `<p>`
    - Une liste non ordonnée `<ul>` contenant trois éléments `<li>` .
  - Ajoute l'identifiant `header` au titre `h2` de la 1ère section.
  - Ajoute la classe `highlight` aux paragraphes de la 1ère et 3e section.
  - Une liste non ordonnée `<ul>` avec la classe `special` contenant trois éléments `<li>` .
  - Une liste ordonnée `<ol>` contenant trois éléments `<li>` .
  - Une section `<section>` avec l'identifiant `mainSection` , contenant 3 titres h2 et 3 paragraphes.

ÉTAPE 2 : STYLES DE BASE

- Ajoute une feuille de style CSS et réalise les tâches suivantes :
  - Applique une couleur de fond spécifique à tous les éléments `<p>` qui ont la classe `highlight` .
  - Modifie la couleur d'écriture du titre dont l'id est `header` .
  - Change la couleur de texte de tous les éléments `<li>` .
  - Le 1er `
  - `` de toutes les listes doit être affiché en gras.
  - Adapte le style des `<article>` :
    - Applique une bordure simple (trait gris de 1px)
    - La bordure gauche doit faire 5px d'épaisseur
    - Applique un padding de 5px
    - Ajoute une marge inférieure de 20px

ÉTAPE 3 : COMBINAISON DE SÉLECTEURS

- Utilise des combinaisons de sélecteurs pour les tâches suivantes :
  - Le 1er `<li>` des listes ordonnées doit être affiché en `tomato` .
  - Applique une bordure (avec des traits, `dashed`) et un padding de 5px aux paragraphes (`<p>`) à l'intérieur de la section avec l'id `mainSection` .
  - Modifie l'apparence de la `#mainSection` :
    - la couleur de fond (une couleur pastel)
    - padding: 10px
    - bordure épaisse rouge
    - coins légèrement arrondis
    - les titres de la main section doivent être affichés en majuscules (!! grâce à css: `text-transform: uppercase`).
    - les paragraphes doivent être justifiés (`text-align`)
  - Souligne les éléments `<li>` qui se trouvent dans des listes non ordonnées ayant la classe `.special` (`text-decoration: underline`).

ÉTAPE 4 : INTERACTION ET OBSERVATION

- Teste chaque changement en visualisant le résultat dans le navigateur.
- Expérimente avec différents styles et observe comment chaque sélecteur affecte les éléments de la page.

Aide mémoire CSS

> *Voici un résumé condensé et concentré sur les sélecteurs et propriétés CSS abordées pendant le cours.*

Sélecteurs CSS

Les sélecteurs CSS sont utilisés pour cibler des éléments HTML et leur appliquer des styles.

1. **Sélecteur d'Élément**

```
p {
    /* Styles à appliquer aux paragraphes */
}
```

2. **Sélecteur de Classe**

```
.ma-classe {
    /* Styles à appliquer aux éléments avec la classe "ma-classe" */
}
```

3. **Sélecteur d'ID**

```
#mon-id {
    /* Styles à appliquer à l'élément avec l'ID "mon-id" */
}
```

4. **Sélecteur Universel**

```
* {
    /* Styles à appliquer à tous les éléments */
}
```

5. **Sélecteur de Descendant**

```
div p {
    /* Styles à appliquer aux paragraphes qui sont descendants de div */
}
```

6. **Sélecteur Enfant Direct**

```
ul > li {
    /* Styles à appliquer aux éléments li qui sont enfants directs d'ul */
}
```

7. **Sélecteur de Pseudo-Classe**

```
a:hover {
    /* Styles à appliquer aux liens lorsque la souris est dessus */
}
```

8. **Sélecteur de Groupe**

```
h1, h2, h3 {
    /* Styles à appliquer aux titres h1, h2 et h3 */
}
```

Propriétés CSS

Les propriétés CSS sont utilisées pour définir les styles des éléments ciblés par les sélecteurs.

1. **color**

```
color: #FF0000; /* Définit la couleur du texte en rouge */
```

2. **font-size**

```
font-size: 16px; /* Définit la taille de la police à 16 pixels */
```

3. **font-family**

```
font-family: Arial, sans-serif; /* Définit la police de caractères */
```

4. **background-color**

```
background-color: #EFEFEF; /* Définit la couleur de fond en gris clair */
```

5. **margin**

```
margin: 10px; /* Définit les marges extérieures de l'élément */
```

6. **padding**

```
padding: 5px; /* Définit les espacements intérieurs de l'élément */
```

7. **border**

```
border: 1px solid #000; /* Définit une bordure d'un pixel en noir solide */
```

8. **text-align**

```
text-align: center; /* Centre le texte dans l'élément */
```

9. **width**

```
width: 200px; /* Définit la largeur de l'élément à 200 pixels */
```

10. **height**

```
height: 1800px; /* Définit la hauteur de l'élément à 180 pixels */
```

Utilise cette cheatsheet comme référence rapide pour les sélecteurs CSS et les propriétés CSS les plus couramment utilisés. Ces outils te permettront de personnaliser l'apparence de tes pages web de manière efficace et précise.

Les outils pour créer du HTML

> *Sur cette page tu trouveras les principaux outils utilisés dans le cadre du cours d'HTML*

Voir dans la colonne de droite -->

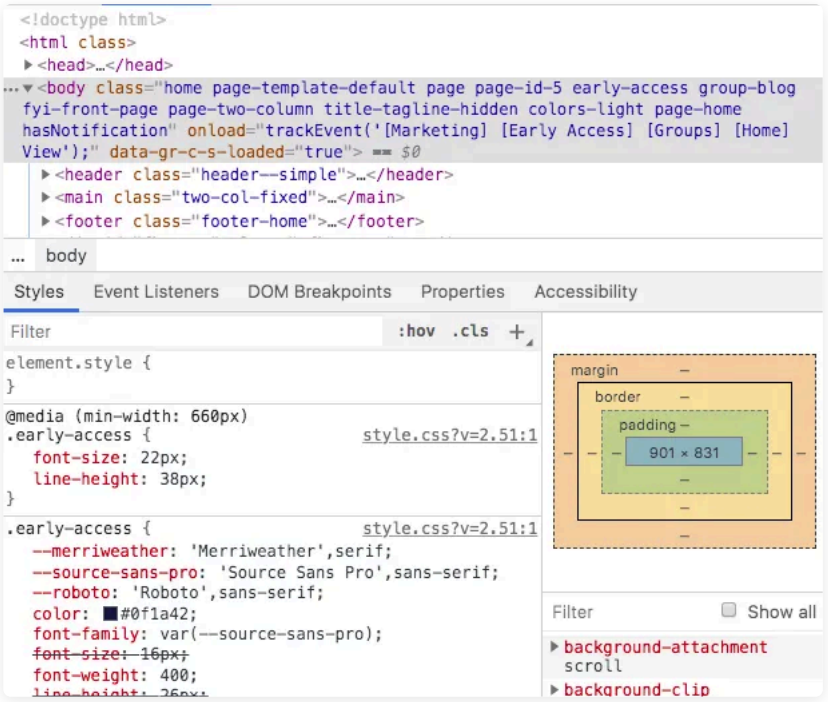
La console développeur des navigateurs

introduction

Dans cet article nous allons explorer un outil puissant et essentiel pour tout développeur web : la **console développeur** des navigateurs. Que tu sois débutant ou expérimenté, cet outil te permettra de **débugger**, d'**optimiser** et de **comprendre** le fonctionnement de tes sites web. Alors, prépare-toi à plonger dans le monde de la console développeur !

Qu'est-ce que la Console Développeur ?

La console développeur est une fonctionnalité intégrée à tous les navigateurs web modernes, tels que Google Chrome, Mozilla Firefox, Microsoft Edge, et bien d'autres. Elle te permet d'inspecter, de débugger et d'interagir avec le code HTML, CSS et JavaScript de n'importe quelle page web en temps réel.



Pour ouvrir la console développeur, tu peux généralement utiliser le raccourci clavier `F12` ou `Ctrl + Shift + I` (ou `Cmd + Option + I` sur Mac) ou simplement faire un clic droit sur une page web et sélectionner "Inspecter" ou "Inspecter l'élément".

Les Principales Fonctionnalités de la Console Développeur

1. Inspection d'Éléments (Elements)

La première onglet de la console développeur te permet d'inspecter le code HTML et CSS d'une page. Tu peux sélectionner et modifier des éléments en direct, ce qui est utile pour tester des modifications de mise en page.

2. Console (Console)

L'onglet "Console" est l'endroit où tu peux voir les messages d'erreur JavaScript, les avertissements et afficher des informations de débogage à l'aide de la fonction `console.log()`. C'est essentiel pour comprendre ce qui se passe dans ton code JavaScript.

3. Réseau (Network)

L'onglet "Réseau" te permet de surveiller les requêtes HTTP, y compris les appels AJAX, et de vérifier les performances de chargement de ta page. Cela t'aide à optimiser la vitesse de ton site web.

4. Sources (Sources)

Dans l'onglet "Sources", tu peux parcourir et déboguer ton code JavaScript, ajouter des points d'arrêt (breakpoints) pour analyser le flux d'exécution de ton programme.

5. Performances (Performance)

L'onglet "Performances" te permet de mesurer les performances de ton site web, d'identifier les goulots d'étranglement et d'améliorer la réactivité de tes pages.

Bonnes Pratiques

- Utilise la console développeur régulièrement pour débugger ton code, vérifier les erreurs et optimiser les performances de ton site.
- Familiarise-toi avec les raccourcis clavier de la console, ils peuvent t'économiser beaucoup de temps.
- N'hésite pas à explorer les fonctionnalités avancées de la console, comme la simulation de périphériques mobiles ou le test de l'accessibilité.

Conclusion

La console développeur des navigateurs est un outil indispensable pour tout développeur web. En la maîtrisant, tu pourras créer des sites web plus performants, débugger plus rapidement et améliorer ton niveau de développement.

N'hésite pas à l'explorer et à l'utiliser régulièrement. Plus tu l'utilises, plus tu deviens compétent dans l'art du développement web. Alors, lance-toi et explore ce monde magique de la console développeur !

Amuse-toi bien à coder, et à bientôt pour de nouvelles découvertes dans le monde du développement web ! 😊

HTML

introduction

Cheat Sheet / Aide mémoire HTML

> *Voici un résumé condensé concentré des balises HTML abordées pendant le cours.*

Voici une *cheat sheet* (aide-mémoire) pour les éléments de base du langage HTML.

Structure de base d'une page HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Titre de la page</title>
</head>
<body>
    <!-- Contenu de la page -->
</body>
</html>
```

- `<!DOCTYPE html>` : Déclaration du type de document.
- `<html>` : Balise racine du document HTML.
- `<head>` : Contient des informations sur la page, telles que le titre.
- `<title>` : Définit le titre de la page qui s'affiche dans l'onglet du navigateur.
- `<body>` : Contient le contenu visible de la page.

Titres et paragraphes

```
<h1>Titre de niveau 1</h1>
<h2>Titre de niveau 2</h2>
<h3>Titre de niveau 3</h3>
<h4>Titre de niveau 4</h4>
<h5>Titre de niveau 5</h5>
<h6>Titre de niveau 6</h6>

<p>Paragraphe de texte.</p>
```

- `<h1>` à `<h6>` : Définit les titres de différents niveaux, `<h1>` étant le plus important. **Un seul** titre de niveau 1 sur la page. Plus d'informations et exemples sur MDN
- `<p>` : Définit un paragraphe de texte. Plus d'informations et exemples sur MDN

Texte en gras

```
<p>Le texte <b>en gras</b>.</p>
```

- `<b>` : Utilisé pour mettre en évidence du texte en gras.

Note: depuis quelques années, la balise `<strong>` est préférée à `<b>` :

```
<p>Le texte <strong>en gras</strong>.</p>
```

Listes

Liste non ordonnée

C'est la fameuse "boulette liste" 😀

```
<ul>
    <li>Élément 1</li>
    <li>Élément 2</li>
    <li>Élément 3</li>
</ul>
```

- `<ul>` : Définit une liste non ordonnée.
- `<li>` : Élément de liste.
- Plus d'informations et exemples sur MDN

Liste ordonnée

```
<ol>
    <li>Élément 1</li>
    <li>Élément 2</li>
    <li>Élément 3</li>
</ol>
```

- `<ol>` : Définit une liste ordonnée.
- Plus d'informations et exemples sur MDN

Images

```
<img src="chemin_de_l_image.jpg" alt="Description de l'image">
```

- `<img>` : Insère une image.
- `src` : Spécifie l'URL de l'image.
- `alt` : Texte alternatif en cas d'indisponibilité de l'image.
- Plus d'informations et exemples sur MDN

Pour que l'image prenne 100% de la largeur disponible, on peut lui appliquer la règle CSS suivante:

```
width: 100%;
```

Liens hypertextes

```
<a href="http://www.exemple.com">Texte affiché</a>
```

- `<a>` : Crée un lien hypertexte.
- `href` : Spécifie l'URL de destination du lien.
- `Texte affiché` : texte affiché à l'écran.

HTML - La balise DIV

Les balises `<div>` sont omniprésentes dans le langage HTML, mais que sont-elles exactement et à quoi servent-elles ? Dans cet article, nous allons plonger dans le monde des balises `<div>`, découvrir leur utilité et examiner les bonnes pratiques pour les utiliser de manière efficace dans la création de pages web.

Qu'est-ce qu'une Balise `<div>` ?

La balise `<div>` est l'un des éléments HTML les plus fondamentaux. Elle est utilisée pour **diviser une page web en sections** ou en conteneurs. Le nom "div" est une abréviation de "division". Contrairement à d'autres balises HTML telles que `<p>` ou `<h1>` qui définissent le contenu et la structure sémantique, la balise `<div>` n'apporte aucune signification particulière. Elle agit principalement comme un conteneur générique permettant d'organiser et de mettre en page les éléments d'une page.

Utilité des Balises `<div>`

Les balises `<div>` sont extrêmement utiles pour plusieurs raisons :

1. Structurer la Page

Les balises `<div>` permettent de **structurer une page web** en sections logiques. Par exemple, vous pouvez utiliser une balise `<div>` pour définir l'en-tête, le contenu principal, la barre latérale et le pied de page de votre site.

```
<div id="header">
    <!-- Contenu de l'en-tête -->
</div>

<div id="content">
    <!-- Contenu principal -->
</div>

<div id="sidebar">
    <!-- Contenu de la barre latérale -->
</div>

<div id="footer">
    <!-- Contenu du pied de page -->
</div>
```

2. Appliquer des Styles CSS

Les balises `<div>` sont couramment utilisées comme conteneurs pour appliquer des styles CSS. Vous pouvez cibler ces balises avec des sélecteurs CSS et leur appliquer des mises en forme spécifiques, telles que la mise en page, la couleur de fond, la marge, la largeur, etc.

```
<style>
    #header {
        background-color: #333;
        color: white;
        padding: 10px;
    }

    .content {
        margin: 20px;
    }

    #footer {
        background-color: #333;
        color: white;
        padding: 10px;
    }
</style>
```

3. Faciliter la Gestion

L'utilisation de balises `<div>` bien nommées **facilite la gestion du contenu** et de la **mise en page** d'un site web. Les développeurs peuvent rapidement comprendre la structure de la page en examinant le code HTML, ce qui simplifie la maintenance et la collaboration.

Bonnes Pratiques pour l'Utilisation des Balises `<div>`

Pour utiliser efficacement les balises `<div>`, voici quelques bonnes pratiques à suivre :

1. Utilisez des Noms de Classes et d'IDs Descriptifs

Attribuez des noms de classes et d'IDs descriptifs à vos balises `<div>` pour indiquer leur rôle ou leur fonction sur la page. Cela rendra le code plus compréhensible et facilitera la maintenance.

```
<div class="header">
    <!-- Contenu de l'en-tête -->
</div>

<div class="main-content">
    <!-- Contenu principal -->
</div>

<div class="sidebar">
    <!-- Contenu de la barre latérale -->
</div>

<div id="footer">
    <!-- Contenu du pied de page -->
</div>
```

2. Évitez les Divs Inutiles

N'ajoutez pas de balises `<div>` inutiles. Utilisez-les uniquement lorsque cela est nécessaire pour structurer ou mettre en forme votre page.

3. Utilisez des Commentaires

Quand vous avez des `<div>` avec beaucoup de contenu, cela peut aider d'ajouter des commentaires au début et la fin pour retrouver facilement la balise de fin correspondante

```
<!-- Début En-tête -->
<div class="header">
    ... contenu très long ...
</div>
<!-- Fin En-tête -->
```

Conclusion

Les balises `<div>` sont des éléments essentiels en HTML, utilisées pour structurer et mettre en forme les pages web. En suivant les bonnes pratiques et en les utilisant judicieusement, vous pouvez créer des mises en page bien organisées et faciles à gérer pour vos sites web. N'hésitez pas à expérimenter et à les combiner avec d'autres éléments HTML pour obtenir le résultat souhaité.

HTML - Le Responsive Web Design

> *Le **Responsive Web Design (RWD)**, ou conception web adaptative en français, est une approche de développement qui permet aux sites web de s'adapter automatiquement à toutes les tailles d'écran et résolutions, que ce soit sur mobile, tablette ou ordinateur de bureau. Avec l'explosion des appareils mobiles, cette méthode est devenue une norme incontournable pour garantir une expérience utilisateur optimale. Cet article explore en détail ce qu'est le Responsive Web Design, pourquoi il est essentiel, et comment le mettre en œuvre efficacement.*

Qu'est-ce que le Responsive Web Design ?

Introduit pour la première fois en 2010, le concept de Responsive Web Design repose sur l'idée de créer des **interfaces flexibles capables de s'adapter à différents environnements** grâce à trois principes fondamentaux :

1. **Grilles fluides (Fluid Grids)** : Utilisation de dimensions proportionnelles (en pourcentages) plutôt que fixes (en pixels) pour définir les tailles des éléments.
2. **Images flexibles (Flexible Images)** : Les images redimensionnées dynamiquement pour s'ajuster à la largeur de leur conteneur.
3. **Media queries** : Règles CSS permettant d'appliquer des styles spécifiques en fonction de la taille ou des caractéristiques de l'appareil.

Pourquoi le Responsive Web Design est-il important ?

Explosion de l'utilisation mobile

Une majorité des utilisateurs accède désormais au web via des smartphones ou des tablettes. Un site non adaptatif risque de perdre ces utilisateurs en offrant une expérience médiocre.

Amélioration de l'expérience utilisateur (UX)

Un design réactif garantit que les utilisateurs peuvent naviguer confortablement, quel que soit leur appareil, en optimisant la lisibilité, la navigation et l'interaction.

SEO et référencement

Google privilégie les sites adaptés aux mobiles dans son algorithme de classement. Le Responsive Web Design améliore donc la visibilité d'un site dans les moteurs de recherche.

Efficacité et coûts réduits

Plutôt que de concevoir des versions séparées pour le mobile et le bureau, un design responsive permet de maintenir une seule version du site, réduisant ainsi les efforts de développement et de maintenance.

Les Composants Clés du Responsive Web Design

Grilles Fluides

Les grilles fluides utilisent des dimensions relatives (en %) pour rendre les éléments flexibles et adaptables.

Exemple CSS :

```css
.container {
  width: 90%; /* Le conteneur occupe 90% de la largeur disponible */
  margin: 0 auto; /* Centre le conteneur */
}

.column {
  float: left;
  width: 50%; /* Chaque colonne occupe 50% de la largeur */
}
```

Images Flexibles

Les images doivent s'adapter à leur conteneur pour ne pas déborder sur les petits écrans. Cela se fait souvent en utilisant la propriété CSS `max-width`.

Exemple CSS :

```css
img {
  max-width: 100%; /* L'image ne dépasse jamais 100% de la largeur de son conteneur */
  height: auto; /* Maintient les proportions de l'image */
}
```

Media Queries

Les **media queries** sont des règles CSS conditionnelles qui s'appliquent uniquement sous certaines conditions, comme la largeur de l'écran ou l'orientation de l'appareil.

Exemple CSS :

```css
/* Styles pour les écrans inférieurs à 768px */
@media (max-width: 768px) {
  .column {
    width: 100%; /* Chaque colonne occupe toute la largeur */
  }
}

/* Styles pour les écrans supérieurs à 768px */
@media (min-width: 769px) {
  .column {
    width: 50%; /* Deux colonnes côte à côte */
  }
}
```

Étapes pour Implémenter un Site Responsive

Adopter une Approche Mobile-First

Concevez d'abord pour les petits écrans, puis ajoutez des styles supplémentaires pour les appareils plus grands en utilisant des media queries.

```css
/* Styles par défaut (pour mobile) */
body {
  font-size: 16px;
}

/* Styles pour les écrans plus grands */
@media (min-width: 768px) {
  body {
    font-size: 18px;
  }
}
```

Utiliser des Frameworks CSS comme Bootstrap

Bootstrap est un framework CSS conçu pour le Responsive Web Design, avec un système de grille intégré et des classes prêtes à l'emploi.

Exemple avec Bootstrap :

```html
<div class="container">
  <div class="row">
    <div class="col-12 col-md-6">Colonne 1</div>
    <div class="col-12 col-md-6">Colonne 2</div>
  </div>
</div>
```

Tester sur Différents Appareils et Résolutions

Utilisez des outils comme Chrome DevTools ou des plateformes comme BrowserStack pour tester votre site sur différentes tailles d'écran.

Bonnes Pratiques du Responsive Web Design

1. **Utiliser des unités relatives** : Préférez les unités `%`, `em`, ou `rem` plutôt que les pixels pour une meilleure adaptabilité.
2. **Limiter les media queries** : Concevez un design flexible par défaut et utilisez les media queries uniquement lorsque nécessaire.
3. **Minimiser les ressources** : Optimisez les images, réduisez les fichiers CSS/JavaScript pour améliorer les performances sur les appareils mobiles.
4. **Maintenir la lisibilité** : Assurez-vous que la taille du texte est suffisante et que les espaces entre les éléments sont adaptés aux écrans tactiles.
5. **Éviter les dimensions fixes** : Ne fixez pas de largeur ou de hauteur rigide sur les éléments pour éviter les problèmes sur les petits écrans.

Conclusion

Le **Responsive Web Design** est une approche incontournable pour les développeurs modernes. Il garantit une expérience utilisateur fluide et cohérente sur tous les appareils, améliore le référencement, et simplifie la gestion des projets web. En maîtrisant les principes de grilles fluides, d'images flexibles et de media queries, vous êtes en mesure de concevoir des sites adaptatifs qui répondent aux exigences de l'ère numérique.

Créer un site complet - Composants

> *Dans ce mini-projet, tu vas créer un mini-site web en **HTML/CSS** pour expliquer un composant informatique de façon claire, structurée et un peu technique, afin qu'un élève qui "s'y connaît un peu" comprenne réellement son rôle, ses critères de choix et les différences entre modèles.*

Objectifs

HTML / CSS

- Produire un site **structuré**, **navigable**, **lisible** et cohérent.
- Utiliser une structure sémantique simple (titres, paragraphes, listes, tableaux, sections).
- Mettre en forme avec CSS (typographie, couleurs, marges, alignements, blocs).

Contenu "composant"

- Expliquer le **rôle** du composant et son **fonctionnement** (vulgarisé, mais sérieux).
- Présenter les **notions techniques clés** (définitions + à quoi ça sert).
- Relier à des **usages concrets** (bureautique, gaming, création, serveur, etc.).
- Donner une vue assez exhaustive du marché : **constructeurs majeurs + gammes**.
- **Comparer au moins 3 modèles** (tableau + analyse).

Contraintes techniques

- Technologies : **HTML + CSS uniquement**
- Ressources autorisées : images, icônes (simples), polices web (ex. Google Fonts) si tu sais le faire proprement.
- Site multi-pages : **minimum 4 à 5 pages**
- Pas de responsive obligatoire, pas de Flexbox obligatoire (tu peux rester en mise en page simple).

Livrables attendus

- Un dossier du site contenant :
  - `index.html`
  - au moins 3–4 autres pages `.html`
  - `style.css`
  - un dossier `img/` si tu utilises des images
- Un menu de navigation identique sur toutes les pages.

Structure minimale (recommandée)

- `index.html` : Accueil / résumé du composant
- `fonctionnement.html` : Comment ça marche (global → spécifique)
- `criteres.html` : Critères techniques + comment choisir
- `marché.html` : Constructeurs, gammes, modèles connus
- `comparatif.html` : Comparaison de 3 modèles (minimum)

Tu peux ajouter une page (ou plusieurs) si ton composant s'y prête :

- `installation.html` (montage, compatibilité, erreurs fréquentes)
- `glossaire.html` (définitions)
- `sources.html` (bibliographie / liens)

Contenu minimum par page

Accueil (index)

- Définition courte : "c'est quoi ?"
- À quoi ça sert ? (3–5 usages)
- Où il se situe dans l'ordinateur (schéma simple ou description)
- Plan du site (mini sommaire)
- Image illustrative + légende

Fonctionnement

- Explication en 2 niveaux :
  - Niveau 1 : analogie / vulgarisation (un paragraphe)
  - Niveau 2 : explication technique simple (2–4 sous-parties)
- Interaction avec d'autres composants (ex. "dépend de…", "impacte…")
- Une section "Ce qu'on croit souvent" (2 idées reçues + correction)

Critères techniques

- Liste structurée de **6 à 10 critères** (selon composant)
- Pour chaque critère :
  - définition simple
  - unité/valeur typique
  - impact réel sur performances/usage
- Une section "Profils d'achat" :
  - "Usage bureautique"
  - "Gaming"
  - "Création"
  - "Budget serré" (et tu expliques comment le critère pèse selon le profil)

Marché

- Les acteurs principaux (constructeurs) + rôle de chacun
- Gammes / familles de produits (entrée de gamme, milieu, haut de gamme)
- Exemples de modèles connus (sans faire catalogue, mais représentatif)
- Une mini-frise "évolution" ou "tendances récentes" (2–4 points)

Comparatif

- Sélection de **3 modèles** (idéalement : budget / milieu / haut de gamme)
- Un tableau comparatif avec des critères pertinents
- Une analyse rédigée :
  - forces/faiblesses de chaque modèle
  - pour quel type d'utilisateur
  - "si je devais choisir, je prends X pour tel besoin"

Évaluation (70% HTML/CSS, 30% contenu)

HTML/CSS (70%)

- Structure & sémantique (titres, sections, propreté du code) : 20%
- Navigation & cohérence multi-pages : 15%
- Lisibilité & mise en forme CSS (typographie, marges, hiérarchie visuelle) : 20%
- Qualité globale (finitions, tableau, images légendées, cohérence) : 15%

Contenu composant (30%)

- Compréhension + explication du fonctionnement : 10%
- Notions techniques clés bien expliquées : 10%
- Marché + comparaison de 3 modèles (pertinence + analyse) : 10%

Méthode de travail conseillée (simple)

- Étape 1 : plan du site + menu
- Étape 2 : contenu brut (texte) sur chaque page
- Étape 3 : CSS de base (lisibilité)
- Étape 4 : tableau comparatif + amélioration visuelle
- Étape 5 : relecture, cohérence, fautes, sources

Site complet: Processeur

Site complet: Processeur (CPU)

> *Ce document décrit les consignes spécifiques pour le composant **processeur (CPU)**. Il sert de guide de recherche et de rédaction afin de produire un contenu structuré, techniquement pertinent et compréhensible pour un élève de TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **comment un processeur traite l'information**, pourquoi certains processeurs sont plus performants que d'autres, et **comment choisir un CPU adapté à un usage donné** (bureautique, gaming, création, etc.).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître **explicitement ou implicitement** dans le site. Ils servent de fil conducteur pour la recherche et la structuration des pages.

RÔLE ET PRINCIPE GÉNÉRAL

- Processeur (CPU)
- Cerveau de l'ordinateur
- Traitement des instructions
- Calculs logiques et arithmétiques
- Ordonnancement des tâches
- Différence CPU / GPU

ARCHITECTURE INTERNE

- Cœur (core)
- Multi-cœur
- Thread
- Hyper-Threading / SMT
- Architecture processeur
- Jeu d'instructions
- 32 bits / 64 bits

PERFORMANCES

- Fréquence (GHz)
- Turbo / Boost
- Performances mono-cœur
- Performances multi-cœur
- Cache L1 / L2 / L3

CONSOMMATION ET DISSIPATION

- TDP
- Consommation électrique
- Chauffe
- Ventirad
- Refroidissement actif / passif
- Bruit

COMPATIBILITÉ MATÉRIELLE

- Socket
- Carte mère compatible
- Chipset
- Génération de processeur
- Compatibilité RAM

MARCHÉ ET GAMMES

- Constructeurs principaux : Intel, AMD. Historique des générations sur 5 ans.
- Gammes grand public
- Entrée de gamme / milieu de gamme / haut de gamme
- Processeurs desktop vs laptop

USAGES CONCRETS

Quels processeurs pour ces usages?

- Bureautique
- Gaming
- Montage vidéo

- Programmation
- Multitâche
- Rapport performance / prix

Consignes de contenu par page

PAGE D'ACCUEIL – PRÉSENTATION DU PROCESSEUR

- Définition claire et reformulée du processeur
- Rôle dans un ordinateur
- Analogie simple (ex. cerveau, centre de décision)
- Illustration du processeur dans un PC
- Présentation rapide des pages du site

PAGE FONCTIONNEMENT

- Comment un processeur exécute une instruction (version vulgarisée)
- Notion de cœur et de thread expliquée simplement
- Différence entre fréquence et nombre de cœurs
- Interaction avec la RAM et la carte mère + socket
- Limites physiques (chaleur, consommation)

PAGE CRITÈRES TECHNIQUES

- Liste structurée des critères importants pour choisir un CPU
- Pour chaque critère :
  - définition
  - exemple concret
  - impact réel sur l'usage
- Mise en évidence des erreurs courantes (ex. "plus de GHz = toujours mieux")

PAGE MARCHÉ

- Présentation des deux grands constructeurs
- Explication des gammes (ex. i3/i5/i7 – Ryzen 3/5/7) (historique sur 5 ans)
- Positionnement des processeurs selon le type d'utilisateur
- Évolution récente (augmentation du nombre de cœurs, baisse de gravure, etc.)
- autres types de processeurs (section obligatoire mais courte) Cette partie a pour objectif d'ouvrir la réflexion : tous les ordinateurs n'utilisent pas le même type de processeur, et le CPU "classique" Intel/AMD n'est pas la seule solution aujourd'hui.

⚠️ Cette section doit rester descriptive et vulgarisée. Aucune comparaison chiffrée détaillée n'est attendue. --> **Apple Silicon** (M2, M3, M4...) et Processeurs **ARM**.

PAGE COMPARATIF

- Sélection de **3 processeurs minimum**
  - entrée de gamme
  - milieu de gamme
  - haut de gamme
- Tableau comparatif incluant :
  - nombre de cœurs / threads
  - fréquence
  - cache
  - TDP
  - prix approximatif
- Analyse rédigée :
  - avantages et inconvénients
  - usage conseillé
  - choix personnel justifié

Points d'attention pédagogiques

- Ne pas se limiter aux fiches marketing
- Reformuler avec ses propres mots
- Toujours relier une notion technique à un usage concret
- Privilégier la compréhension plutôt que l'accumulation de chiffres
- Comparer intelligemment (pas uniquement "le plus puissant")

Site complet: Mémoire vive (RAM)

> *Ce document présente les **consignes spécifiques** pour le composant **mémoire vive (RAM)**. Il sert de guide de recherche et de rédaction afin de produire un contenu **structuré**, **techniquement juste** et **compréhensible** pour un élève de 4TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **à quoi sert la mémoire RAM**, comment elle influence les performances d'un ordinateur, et **comment choisir une configuration de RAM adaptée à un usage donné**.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent guider la recherche et structurer les pages du site. Ils ne doivent pas être listés tels quels, mais **expliqués avec des phrases claires et des exemples concrets**.

Rôle et principe général

- Mémoire vive (RAM)
- Mémoire volatile
- Mémoire temporaire
- Données en cours d'utilisation
- Différence RAM / stockage (SSD, HDD)
- Rapidité d'accès

Fonctionnement général

- Chargement des programmes
- Accès aléatoire
- Communication avec le processeur
- Goulot d'étranglement mémoire
- Multitâche

Caractéristiques techniques

- Capacité (Go)
- Fréquence (MHz)
- Latence (CAS)
- Bande passante
- DDR (DDR3 / DDR4 / DDR5)
- Générations de RAM
- Tension
- Profil XMP / EXPO (sans détails complexes)

Organisation de la RAM

- Barrettes mémoire
- Slots mémoire
- Single channel
- Dual channel
- Quad channel (mention)
- Importance du bon placement

Compatibilité

- Carte mère
- Processeur
- Type de DDR supporté
- Limite maximale de RAM
- Format (DIMM / SO-DIMM)

Marché et gammes

- Marques principales
- Entrée de gamme / milieu / haut de gamme
- RAM "standard" vs RAM "gaming" (aspect marketing)

- Différences de prix

Usages concrets

- Bureautique
- Gaming
- Montage vidéo / création
- Programmation
- Machines virtuelles
- Impact réel d'une augmentation de RAM

Consignes de contenu par page

Page d'accueil – Présentation de la RAM

- Définition claire de la mémoire RAM
- Rôle dans un ordinateur
- Analogie simple (ex. bureau de travail, table, plan de travail)
- Différence entre RAM et stockage
- Image ou schéma explicatif simple
- Présentation du plan du site

Page Fonctionnement

- Comment un programme utilise la RAM
- Ce qui se passe quand la RAM est insuffisante
- Lien entre RAM et processeur
- Notion de multitâche expliquée concrètement
- Conséquences pratiques sur la vitesse du PC

Page Critères techniques

- Présentation structurée des critères importants :
  - capacité
  - fréquence
  - latence
  - type de DDR
  - nombre de barrettes
- Pour chaque critère :
  - définition simple
  - exemple concret
  - impact réel sur l'usage
- Démystification des idées reçues :
  - "plus de MHz = toujours mieux"
  - "la RAM gaming est forcément plus rapide"

Page Marché

- Présentation des grandes marques de RAM
- Différence entre RAM pour PC fixe et PC portable
- Gammes de prix
- Évolution récente (augmentation des capacités, DDR5, etc.)
- Place du marketing dans les produits mémoire

Page Comparatif (obligatoire)

- Comparaison de **3 configurations de RAM minimum**, par exemple :
  - 8 Go DDR4
  - 16 Go DDR4
  - 16 Go DDR5

- Tableau comparatif avec :
  - capacité
  - fréquence
  - latence
  - type
  - prix approximatif
  - usage conseillé
- Analyse rédigée :
  - avantages et limites
  - pour quel type d'utilisateur
  - choix personnel justifié

Ouverture technologique – RAM intégrée et cas particuliers

Cette section doit rester **courte** et **vulgarisée**.

RAM intégrée

- RAM soudée à la carte mère
- Fréquente dans les ordinateurs portables
- Impossible ou difficile à faire évoluer
- Avantages : compacité, consommation
- Inconvénients : évolutivité limitée

RAM unifiée (Apple, SoC)

- Mémoire partagée entre CPU et GPU
- Utilisée dans les systèmes sur puce (SoC)
- Différence avec la RAM classique
- Impact sur les performances et l'autonomie

Cas du Raspberry Pi

- RAM intégrée à la carte
- Quantité fixe selon le modèle
- Partagée avec le processeur graphique
- Exemple d'ordinateur minimaliste

Consigne élève (formulation claire)

> *Dans une section intitulée **« Ouverture technologique : RAM intégrée et systèmes particuliers »**, explique brièvement :*
>
> - *pourquoi la RAM n'est pas toujours sous forme de barrettes,*
> - *les avantages et limites de la RAM intégrée,*
> - *le cas du Raspberry Pi comme exemple d'ordinateur compact.*
>
> *Cette partie ne doit pas dépasser **½ page**.*

Points d'attention pédagogiques

- Toujours expliquer **à quoi sert une caractéristique**
- Éviter les listes de chiffres sans interprétation
- Relier chaque notion à un usage réel
- Ne pas confondre vitesse de stockage et vitesse de RAM
- Comparer intelligemment des configurations réalistes

Site complet: Stockage – SSD / NVMe

> *Ce document présente les **consignes spécifiques** pour le composant **stockage SSD (SATA et NVMe)**. Il sert de guide de recherche afin de produire un contenu **structuré**, **techniquement compréhensible** et **orienté usage réel**, adapté au niveau TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **comment les données sont stockées**, pourquoi un SSD est plus rapide qu'un disque dur classique, et **dans quels cas un SSD NVMe est réellement utile** par rapport à un SSD SATA.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés servent de **fil conducteur** pour la recherche. Ils doivent être expliqués avec des phrases claires, des exemples concrets et des comparaisons pertinentes.

Rôle du stockage

- Stockage de données
- Mémoire non volatile
- Données persistantes
- Système d'exploitation
- Programmes et fichiers
- Différence stockage / RAM

Types de stockage

- Disque dur (HDD)
- SSD
- Absence de pièces mécaniques
- Mémoire flash
- Différence HDD vs SSD

SSD SATA

- Interface SATA
- Limite de vitesse SATA
- Format 2,5 pouces
- Compatibilité universelle
- Remplacement d'un HDD

SSD NVMe

- NVMe
- Interface PCI Express
- Très haut débit
- Faible latence
- Format M.2
- Différence NVMe / SATA

Performances

- Vitesse de lecture
- Vitesse d'écriture
- Débit séquentiel
- Accès aléatoire
- Temps de démarrage
- Chargement des applications

Caractéristiques techniques

- Capacité (Go / To)
- Contrôleur
- Type de mémoire (TLC, QLC – sans entrer dans les détails)
- Cache
- Endurance (TBW)
- Durée de vie

Compatibilité matérielle

- Carte mère
- Port SATA
- Slot M.2
- Compatibilité NVMe
- Générations PCIe (mention simple)
- PC fixe vs portable

Marché et usages

- Marques principales
- Entrée de gamme / milieu / haut de gamme
- Rapport performance / prix
- Usage réel (bureautique, gaming, création)

Consignes de contenu par page

Page d'accueil – Présentation du stockage SSD

- Définition du stockage
- Rôle du stockage dans un ordinateur
- Différence entre stockage et RAM
- Pourquoi les SSD ont remplacé les HDD
- Illustration d'un SSD SATA et d'un SSD M.2
- Présentation du plan du site

Page Fonctionnement

- Comment un SSD stocke les données (vulgarisé)
- Différence entre mémoire mécanique (HDD) et électronique (SSD)
- Pourquoi un SSD est plus rapide
- Différence de principe entre SATA et NVMe
- Impact sur l'expérience utilisateur (démarrage, réactivité)

Page Critères techniques

- Présentation claire des critères de choix :
  - capacité
  - type de SSD (SATA / NVMe)
  - vitesses annoncées
  - endurance
  - format
- Pour chaque critère :
  - définition simple
  - exemple concret
  - utilité réelle
- Mise en garde contre les chiffres purement marketing

Page Marché

- Présentation des principaux fabricants de SSD (ex. Samsung, Western Digital, Kingston)
- Gammes proposées (grand public, performance)
- Différences de prix entre SATA et NVMe
- Évolution récente du stockage (généralisation du NVMe)

Page Comparatif (obligatoire)

- Comparaison de **3 solutions minimum**, par exemple :
  - HDD classique
  - SSD SATA

    - SSD NVMe
- Tableau comparatif avec :
    - type
    - vitesse approximative
    - capacité
    - prix
    - usage conseillé
- Analyse rédigée :
    - gains réels observables
    - dans quels cas le NVMe est utile
    - choix personnel justifié

Ouverture technologique – stockage moderne et cas particuliers

Cette section doit rester **courte** et **accessible**.

Stockage externe

- SSD externe USB
- Différence USB / Thunderbolt (mention)
- Mobilité et sauvegarde

Cas du Raspberry Pi

- Stockage sur carte microSD
- Vitesse limitée
- Suffisant pour l'apprentissage et les projets
- Illustration d'un stockage minimaliste
- SSD sur Raspberry Pi?

Consigne élève (formulation claire)

> *Dans une section intitulée **« Ouverture technologique : stockage moderne »**, explique brièvement :*
>
> - *pourquoi tous les SSD ne se valent pas,*
> - *la différence entre SSD SATA et NVMe,*
> - *les limites du stockage soudé ou sur microSD.*
>
> *Cette partie doit rester **courte (½ page maximum)** et vulgarisée.*

Points d'attention pédagogiques

- Ne pas confondre vitesse de stockage et vitesse de RAM
- Toujours relier une vitesse à un usage réel
- Éviter les comparaisons irréalistes
- Privilégier la compréhension plutôt que les chiffres bruts
- Bien distinguer **format**, **interface** et **performances**

Site complet: Alimentation (PSU – Power Supply Unit)

> *Ce document présente les **consignes spécifiques** pour le composant **alimentation**. Il sert de guide de recherche afin de produire un contenu **structuré**, **techniquement pertinent** et **orienté sécurité et usage réel**, adapté au niveau 4TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **le rôle d'une alimentation**, comment elle fournit une énergie stable aux composants, **pourquoi le choix d'une PSU est crucial pour la fiabilité d'un PC**, et comment dimensionner correctement une alimentation selon une configuration donnée.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent guider la recherche et structurer les pages du site. Ils doivent être **expliqués**, illustrés par des exemples et reliés à des usages concrets.

Rôle et principes généraux

- Alimentation (PSU)
- Conversion AC / DC
- Courant alternatif (secteur)
- Courant continu (PC)
- Distribution de l'énergie
- Stabilité électrique
- Protection des composants

Puissance et dimensionnement

- Puissance (Watts)
- Consommation des composants
- Marge de sécurité
- Suralimentation / sous-alimentation
- Calcul de puissance nécessaire
- Impact d'un GPU puissant

Rendement énergétique

- Rendement
- Certification 80 PLUS
- Bronze / Silver / Gold / Platinum
- Perte d'énergie
- Chauffe
- Consommation électrique

Connectique

- Connecteur ATX 24 broches
- Connecteur CPU (EPS 4/8 broches)
- Connecteurs PCIe
- Connecteurs SATA
- Molex (mention)
- Compatibilité des câbles

Modularité

- Alimentation non modulaire
- Semi-modulaire
- Modulaire
- Gestion des câbles
- Flux d'air
- Facilité de montage

Sécurité et qualité

- Protections électriques (surtension, court-circuit – sans détails techniques)
- Stabilité des tensions
- Marque vs qualité réelle

- Risques d’une alimentation bas de gamme
- Durée de vie

Format et compatibilité

- Format ATX
- Format SFX (mention)
- Boîtier compatible
- Alimentation PC fixe vs portable

Marché et usages

- Marques principales
- Entrée de gamme / milieu / haut de gamme
- Rapport qualité / prix
- Usage bureautique
- Usage gaming
- Usage création / workstation

Consignes de contenu par page

Page d’accueil – Présentation de l’alimentation

- Définition claire d’une alimentation
- Rôle dans un ordinateur
- Pourquoi c’est un composant souvent sous-estimé
- Illustration des câbles principaux
- Présentation du plan du site

Page Fonctionnement

- Comment l’électricité arrive dans le PC
- Transformation du courant secteur en courant utilisable
- Distribution de l’énergie aux composants
- Importance de la stabilité électrique
- Conséquences d’une mauvaise alimentation

Page Critères techniques

- Présentation structurée des critères importants :
  - puissance
  - rendement
  - certifications
  - connectique
  - modularité
- Pour chaque critère :
  - définition simple
  - exemple concret
  - impact réel sur une configuration
- Mise en garde contre les idées reçues :
  - “plus de watts = mieux”
  - “toutes les alimentations se valent”

Page Marché

- Présentation des principales marques d’alimentations
- Différences de prix expliquées
- Rôle des certifications
- Positionnement des alimentations selon l’usage
- Importance de la réputation et des tests indépendants

Page Comparatif (obligatoire)

- Comparaison de **3 alimentations minimum**, par exemple :
  - entrée de gamme (bureautique)
  - milieu de gamme (gaming)
  - haut de gamme (configuration puissante)
- Tableau comparatif incluant :
  - puissance
  - certification
  - modularité
  - nombre de connecteurs
  - prix approximatif
- Analyse rédigée :
  - avantages / limites
  - configuration recommandée
  - choix personnel justifié

Ouverture technologique – alimentation et cas particuliers

Cette section doit rester **courte** et **vulgarisée**.

Alimentations de PC portables

- Bloc d'alimentation externe
- Conversion partiellement externalisée
- Avantages : compacité, silence
- Limites : puissance limitée, non universel

Mini-PC et cartes intégrées

- Alimentation externe ou intégrée
- Cas des mini-PC
- Exemple du Raspberry Pi

Efficacité énergétique

- Impact environnemental
- Consommation électrique annuelle
- Intérêt des alimentations efficaces

Consigne élève (formulation claire)

> *Dans une section intitulée **« Ouverture technologique : alimentations alternatives »**, explique brièvement :*
>
> - *pourquoi les PC portables et mini-PC n'utilisent pas d'alimentation ATX classique,*
> - *les avantages et limites des blocs externes,*
> - *le lien entre rendement énergétique et consommation.*
>
> *Cette partie doit rester **courte (½ page maximum)**.*

Points d'attention pédagogiques

- L'alimentation influence **la stabilité de tout le PC**
- Une alimentation de mauvaise qualité peut endommager les composants
- Les watts seuls ne suffisent pas à juger une PSU
- Toujours raisonner en **configuration complète**
- Expliquer le **pourquoi**, pas seulement le **quoi**

Site complet: Carte graphique – GPU / IGP

> *Ce document présente les **consignes spécifiques** pour le composant **carte graphique (GPU)**, en distinguant clairement **GPU dédié** et **IGP (processeur graphique intégré)**. Il sert de guide de recherche afin de produire un contenu **structuré**, **techniquement compréhensible** et **orienté usages concrets**, adapté au niveau TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **comment sont générées les images à l'écran**, pourquoi certaines cartes graphiques sont indispensables pour certains usages, et **dans quels cas un IGP est suffisant**.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés servent de **fil conducteur** pour la recherche. Ils doivent être expliqués avec des phrases claires et illustrés par des exemples concrets.

Rôle et principes généraux

- Carte graphique
- GPU (Graphics Processing Unit)
- Traitement graphique
- Calculs parallèles
- Affichage à l'écran
- Différence CPU / GPU
- Accélération matérielle

GPU dédié vs IGP

- GPU dédié (carte graphique séparée)
- IGP (processeur graphique intégré)
- Partage de la mémoire RAM
- Performances graphiques
- Consommation énergétique
- Avantages / limites de chaque solution

Architecture et performances

- Cœurs de calcul (CUDA cores / unités de calcul – sans entrer dans les détails)
- Fréquence GPU
- Performances graphiques
- Résolution (Full HD, 4K – mention)
- Nombre d'écrans supportés
- Fluidité (FPS – images par seconde)

Mémoire vidéo

- Mémoire vidéo (VRAM)
- Capacité (Go)
- Bande passante
- Différence RAM / VRAM
- Impact sur les jeux et applications graphiques

Refroidissement et consommation

- Chauffe
- Ventilation
- Dissipation thermique
- Consommation électrique
- Connecteurs d'alimentation GPU
- Bruit

Compatibilité matérielle

- Slot PCI Express
- Compatibilité carte mère
- Taille de la carte graphique

- Boîtier compatible
- Alimentation suffisante

Marché et gammes

- Constructeurs principaux : NVIDIA, AMD, Intel
- Entrée de gamme / milieu / haut de gamme
- Cartes gaming vs professionnelles (mention)
- Évolution rapide du marché

Usages concrets

- Bureautique
- Multimédia
- Gaming
- Montage vidéo
- 3D / création graphique
- Programmation et calculs parallèles (mention)

Consignes de contenu par page

Page d'accueil – Présentation de la carte graphique

- Définition claire d'un GPU
- Rôle de la carte graphique dans un PC
- Différence simple entre GPU dédié et IGP
- Illustration du GPU dans un ordinateur
- Présentation du plan du site

Page Fonctionnement

- Comment une image est calculée et affichée
- Rôle du GPU par rapport au CPU
- Fonctionnement général d'un IGP
- Pourquoi un GPU dédié est plus performant
- Lien avec la RAM et la VRAM

Page Critères techniques

- Présentation structurée des critères importants :
  - type de GPU (IGP / dédié)
  - puissance graphique
  - mémoire vidéo
  - consommation
  - compatibilité matérielle
- Pour chaque critère :
  - définition simple
  - exemple concret
  - impact réel sur l'usage
- Mise en garde contre les idées reçues :
  - "plus de Go de VRAM = meilleure carte"
  - "une carte graphique est toujours nécessaire"

Page Marché

- Présentation des principaux fabricants de GPU
- Différence entre gammes grand public
- Positionnement des cartes selon l'usage
- Place des IGP modernes dans les PC actuels
- Évolution récente (IGP plus performants, GPU très puissants)

Page Comparatif (obligatoire)

- Comparaison de **3 solutions minimum**, par exemple :
  - IGP moderne
  - GPU dédié entrée de gamme
  - GPU dédié milieu ou haut de gamme
- Tableau comparatif incluant :
  - type (IGP / dédié)
  - performances approximatives
  - mémoire vidéo
  - consommation
  - prix approximatif
  - usage conseillé
- Analyse rédigée :
  - avantages et limites
  - type d'utilisateur visé
  - choix personnel justifié

Ouverture technologique – GPU modernes et cas particuliers

Cette section doit rester **courte** et **vulgarisée**.

GPU intégrés modernes

- IGP de plus en plus performants
- Suffisants pour de nombreux usages
- Intégration CPU + GPU

GPU externes et spécialisés

- GPU professionnels (mention)
- GPU pour calcul scientifique / IA (mention)
- GPU externes (eGPU – mention)

Cas du Raspberry Pi

- GPU intégré au SoC
- Usage graphique limité
- Suffisant pour l'apprentissage et l'affichage simple

Consigne élève (formulation claire)

> *Dans une section intitulée **« Ouverture technologique : GPU intégrés et usages alternatifs »**, explique brièvement :*
>
> - *pourquoi un IGP peut suffire dans de nombreux cas,*
> - *dans quels cas un GPU dédié devient indispensable,*
> - *pourquoi certaines machines n'ont pas de carte graphique séparée.*
>
> *Cette partie doit rester **courte (½ page maximum)**.*

Points d'attention pédagogiques

- Toujours distinguer **IGP** et **GPU dédié**
- Ne pas se limiter aux performances gaming
- Relier les caractéristiques aux usages réels
- Éviter les comparaisons irréalistes
- Privilégier la compréhension globale du rôle du GPU

Site complet: Carte mère

> *Ce document présente les **consignes spécifiques** pour le composant **carte mère**. Il sert de guide de recherche afin de produire un contenu **structuré**, **techniquement compréhensible** et **orienté compatibilité et évolutivité**, adapté au niveau TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **le rôle central de la carte mère**, comment elle relie et coordonne tous les composants, et **pourquoi le choix d'une carte mère conditionne la compatibilité et l'évolution d'un ordinateur**.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent guider la recherche et structurer le contenu. Ils doivent être expliqués avec des phrases claires et des exemples concrets.

Rôle et principe général

- Carte mère
- Composant central
- Interconnexion des composants
- Support matériel
- Communication interne
- Stabilité du système

Format et compatibilité

- Format de carte mère
- ATX
- Micro-ATX
- Mini-ITX
- Compatibilité boîtier
- Taille physique

Processeur et socket

- Socket processeur
- Compatibilité CPU
- Générations de processeur
- Support constructeur
- Mise à jour BIOS

Chipset

- Chipset
- Rôle du chipset
- Fonctionnalités
- Différences entre chipsets
- Entrée de gamme / milieu / haut de gamme

Mémoire vive

- Slots RAM
- Type de RAM supportée (DDR4 / DDR5)
- Fréquence supportée
- Capacité maximale
- Dual channel

Extensions et connectique

- Slot PCI Express
- Carte graphique
- Cartes d'extension
- Ports SATA
- Slot M.2
- NVMe

Connectique externe

- Ports USB
- USB 2.0 / 3.x (mention)
- Ethernet/Wifi
- Audio
- Connecteurs vidéo (si IGP)

Alimentation

- Connecteur ATX 24 broches
- Connecteur CPU
- Alimentation des composants
- Stabilité électrique

BIOS / UEFI

- BIOS
- UEFI
- Démarrage du PC
- Paramètres matériels
- Mise à jour

Marché et usages

- Marques principales
- Cartes mères grand public
- Cartes mères gaming (marketing)
- Cartes mères professionnelles (mention)
- Rapport fonctionnalités / prix

Consignes de contenu par page

Page d'accueil – Présentation de la carte mère

- Définition claire de la carte mère
- Rôle dans l'ordinateur
- Pourquoi c'est un composant central
- Illustration d'une carte mère annotée
- Présentation du plan du site

Page Fonctionnement

- Comment les composants communiquent via la carte mère
- Rôle du chipset
- Importance du socket processeur
- Gestion de la mémoire et des extensions
- Lien entre carte mère, CPU et RAM

Page Critères techniques

- Présentation structurée des critères importants :
  - format
  - socket
  - chipset
  - RAM supportée
  - connectique
  - possibilités d'évolution
- Pour chaque critère :
  - définition simple
  - exemple concret
  - impact réel sur le choix d'une configuration
- Mise en garde contre les idées reçues :

- “une carte mère gaming rend le PC plus rapide”
- “toutes les cartes mères sont équivalentes”

Page Marché

- Présentation des principaux fabricants de cartes mères
- Différences de prix expliquées
- Positionnement selon l’usage
- Importance de la compatibilité plutôt que du marketing
- Évolution récente (DDR5, NVMe généralisé, etc.)

Page Comparatif (obligatoire)

- Comparaison de **3 cartes mères minimum**, par exemple :
  - entrée de gamme
  - milieu de gamme
  - haut de gamme
- Tableau comparatif incluant :
  - format
  - socket
  - chipset
  - RAM supportée
  - connectique principale
  - prix approximatif
- Analyse rédigée :
  - avantages et limites
  - type de configuration recommandée
  - choix personnel justifié

Ouverture technologique – cartes mères et cas particuliers

Cette section doit rester **courte** et **vulgarisée**.

Cartes mères compactes

- Mini-PC
- Formats très réduits
- Compromis entre taille et évolutivité

Cartes intégrées

- Composants soudés
- Peu évolutives
- Fréquentes dans les ordinateurs portables

Cas du Raspberry Pi

- Carte unique intégrant CPU, RAM, GPU
- Rôle différent d’une carte mère classique
- Exemple de système minimaliste

Consigne élève (formulation claire)

> *Dans une section intitulée **« Ouverture technologique : cartes mères alternatives »**, explique brièvement :*
>
> - *pourquoi toutes les cartes mères n’offrent pas les mêmes possibilités d’évolution,*
> - *les avantages et limites des formats compacts,*
> - *le cas du Raspberry Pi comme carte intégrant tous les composants principaux.*
>
> *Cette partie doit rester **courte (½ page maximum)**.*

Points d’attention pédagogiques

- La carte mère détermine **la compatibilité globale du PC**
- Un bon choix facilite les évolutions futures
- Le chipset est souvent plus important que l’apparence
- Toujours raisonner en **configuration complète**
- Relier les caractéristiques aux besoins réels

Site complet: Refroidissement (CPU / boîtier)

> *Ce document présente les **consignes spécifiques** pour le composant **refroidissement**. Il sert de guide de recherche afin de produire un contenu **structuré**, **techniquement compréhensible** et **orienté fiabilité**, adapté au niveau TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **pourquoi le refroidissement est indispensable**, comment la chaleur est produite et évacuée dans un ordinateur, et **comment choisir une solution de refroidissement adaptée** à une configuration donnée (silence, performances, budget).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent guider la recherche et structurer le contenu. Ils doivent être expliqués clairement et reliés à des situations concrètes.

Rôle et principes généraux

- Chaleur
- Dissipation thermique
- Température des composants
- Surchauffe
- Stabilité du système
- Durée de vie du matériel

Sources de chaleur

- Processeur (CPU)
- Carte graphique (GPU)
- Alimentation
- Composants sous charge
- Charge de travail

Refroidissement du processeur

- Ventirad
- Radiateur
- Ventilateur
- Pâte thermique
- Contact thermique
- Refroidissement air

Refroidissement liquide (mention encadrée)

- Watercooling
- Circuit fermé (AIO)
- Liquide caloporteur
- Avantages
- Inconvénients
- Complexité

Flux d'air dans le boîtier

- Ventilation du boîtier
- Entrée d'air
- Sortie d'air
- Flux d'air
- Pression positive / négative (mention simple)
- Organisation des câbles

Bruit et confort

- Niveau sonore
- Ventilateurs silencieux
- Vitesse de rotation
- Compromis bruit / refroidissement

Compatibilité

- Socket processeur
- Hauteur du ventirad
- Taille du boîtier
- Fixations
- Compatibilité carte mère

Marché et usages

- Solutions entrée de gamme
- Solutions performance
- Solutions silencieuses
- Rapport efficacité / prix
- Marketing “gaming”

Consignes de contenu par page

Page d’accueil – Présentation du refroidissement

- Définition du refroidissement dans un PC
- Pourquoi les composants chauffent
- Rôle du refroidissement pour la stabilité
- Illustration d’un ventirad et d’un boîtier ventilé
- Présentation du plan du site

Page Fonctionnement

- Comment la chaleur est produite par un composant
- Principe de dissipation thermique
- Rôle du radiateur et du ventilateur
- Importance de la pâte thermique
- Conséquences d’un mauvais refroidissement (throttling, arrêts)

Page Critères techniques

- Présentation structurée des critères importants :
  - type de refroidissement (air / liquide)
  - efficacité
  - bruit
  - compatibilité
  - facilité d’installation
- Pour chaque critère :
  - définition simple
  - exemple concret
  - impact réel sur l’utilisation du PC
- Démystification des idées reçues :
  - “plus c’est gros, mieux c’est”
  - “le watercooling est toujours meilleur”

Page Marché

- Présentation des types de solutions disponibles
- Différences de prix expliquées
- Positionnement selon l’usage (bureautique, gaming, silence)
- Rôle du design et du marketing
- Évolution récente (ventirads performants, AIO populaires)

Page Comparatif (obligatoire)

- Comparaison de **3 solutions minimum**, par exemple :

    - ventirad basique
    - ventirad performant
    - watercooling AIO
- Tableau comparatif incluant :
    - type
    - efficacité approximative
    - niveau sonore
    - compatibilité
    - prix approximatif
- Analyse rédigée :
    - avantages et limites
    - type de configuration recommandé
    - choix personnel justifié

Ouverture technologique – refroidissement et cas particuliers

Cette section doit rester **courte** et **vulgarisée**.

Refroidissement passif

- Sans ventilateur
- Silence total
- Puissance limitée
- Cas d'usage spécifiques

Ordinateurs portables et mini-PC

- Contraintes de place
- Refroidissement compact
- Compromis performances / silence

Cas du Raspberry Pi

- Refroidissement minimal
- Dissipateur simple
- Ventilateur optionnel
- Exemple de système peu énergivore

Consigne élève (formulation claire)

> *Dans une section intitulée **« Ouverture technologique : solutions de refroidissement alternatives »**, explique brièvement :*
>
> - *pourquoi tous les ordinateurs n'ont pas le même type de refroidissement,*
> - *les avantages et limites du refroidissement passif,*
> - *les contraintes spécifiques des ordinateurs compacts.*
>
> *Cette partie doit rester **courte (½ page maximum)**.*

Points d'attention pédagogiques

- Le refroidissement influence directement **les performances et la durée de vie**
- Un bon flux d'air est souvent plus important qu'un gros ventilateur
- Le silence est un critère réel pour l'utilisateur
- Toujours vérifier la compatibilité avant le choix
- Relier le refroidissement à l'usage réel du PC

Site complet: Boîtier (case)

> *Ce document présente les **consignes spécifiques** pour le composant **boîtier**. Il sert de guide de recherche afin de produire un contenu **structuré**, **concret** et **orienté usage réel**, adapté au niveau TT informatique.*

Objectif spécifique du composant

À la fin du travail, l'élève doit être capable d'expliquer **le rôle d'un boîtier**, pourquoi il influence la **compatibilité**, le **refroidissement**, le **confort d'utilisation** et la **durabilité** d'un PC, et comment choisir un boîtier **adapté à une configuration donnée**.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent guider la recherche et structurer le contenu. Ils doivent être expliqués avec des phrases claires et reliés à des situations concrètes.

Rôle et fonctions du boîtier

- Boîtier
- Structure physique
- Protection des composants
- Organisation interne
- Accès aux composants
- Esthétique et ergonomie

Formats et compatibilité

- Format de boîtier
- ATX
- Micro-ATX
- Mini-ITX
- Compatibilité carte mère
- Compatibilité alimentation
- Compatibilité carte graphique (longueur)
- Hauteur du ventirad

Flux d'air et ventilation

- Flux d'air
- Entrée d'air
- Sortie d'air
- Ventilateurs
- Filtration de la poussière
- Impact sur le refroidissement
- Boîtier fermé vs boîtier aéré

Montage et ergonomie

- Facilité de montage
- Gestion des câbles
- Accès sans outil
- Baies de stockage
- Organisation interne
- Maintenance

Connectique du boîtier

- Bouton power
- Ports USB en façade
- Audio en façade
- Câbles internes du boîtier

Matériaux et qualité

- Métal
- Plastique
- Verre trempé

- Solidité
- Vibrations
- Isolation phonique

Bruit et confort

- Ventilation silencieuse
- Isolation acoustique
- Résonance
- Compromis silence / refroidissement

Marché et usages

- Boîtiers entrée de gamme
- Boîtiers orientés airflow
- Boîtiers silencieux
- Boîtiers gaming (design)
- Rapport fonctionnalité / prix

Consignes de contenu par page

Page d'accueil – Présentation du boîtier

- Définition claire du boîtier
- Rôle dans un ordinateur
- Pourquoi ce composant est souvent sous-estimé
- Illustration d'un boîtier ouvert
- Présentation du plan du site

Page Fonctionnement

- Comment les composants sont organisés dans le boîtier
- Circulation de l'air à l'intérieur
- Rôle du boîtier dans le refroidissement global
- Lien entre boîtier, alimentation et refroidissement
- Conséquences d'un boîtier mal adapté

Page Critères techniques

- Présentation structurée des critères importants :
  - format
  - compatibilité des composants
  - ventilation
  - ergonomie
  - niveau sonore
- Pour chaque critère :
  - définition simple
  - exemple concret
  - impact réel sur l'utilisation
- Mise en garde contre les idées reçues :
  - "un boîtier cher améliore les performances"
  - "le design est plus important que l'airflow"

Page Marché

- Présentation des types de boîtiers disponibles
- Différences de prix expliquées
- Boîtiers orientés silence vs performance
- Place du marketing (RGB, design gaming)
- Évolution récente (boîtiers aérés, verre, USB-C)

Page Comparatif (obligatoire)

- Comparaison de **3 boîtiers minimum**, par exemple :
  - boîtier compact
  - boîtier moyen tour
  - boîtier orienté airflow ou silence
- Tableau comparatif incluant :
  - format supporté
  - ventilation incluse
  - compatibilité GPU / ventirad
  - connectique en façade
  - prix approximatif
- Analyse rédigée :
  - avantages et limites
  - type de configuration recommandé
  - choix personnel justifié

Ouverture technologique – boîtiers et cas particuliers

Cette section doit rester **courte** et **vulgarisée**.

Mini-PC et boîtiers compacts

- Contraintes de place
- Peu d'évolutivité
- Refroidissement plus complexe

PC portables

- Boîtier intégré
- Aucun choix possible par l'utilisateur
- Compromis compacité / performances

Boîtiers alternatifs

- Boîtiers ouverts (mention)
- Boîtiers silencieux extrêmes (mention)
- Cas du Raspberry Pi (boîtiers simples ou imprimés)

Consigne élève (formulation claire)

> *Dans une section intitulée **« Ouverture technologique : boîtiers alternatifs »**, explique brièvement :*
>
> - *pourquoi la taille du boîtier influence l'évolutivité,*
> - *les contraintes des boîtiers compacts,*
> - *pourquoi un PC portable n'offre pas de liberté de choix du boîtier.*
>
> *Cette partie doit rester **courte (½ page maximum)**.*

Points d'attention pédagogiques

- Le boîtier influence **le refroidissement, le bruit et l'évolutivité**
- La compatibilité physique est essentielle
- Un bon boîtier facilite le montage et la maintenance
- L'esthétique ne doit pas primer sur la fonction
- Toujours relier le boîtier à la **configuration complète**

Site complet: Carte réseau & interfaces (RJ45 / Wi-Fi)

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment un ordinateur communique avec un réseau, comprendre la différence entre une connexion filaire et sans fil, et choisir une solution réseau adaptée selon le contexte (maison, entreprise, gaming, mobilité…).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Carte réseau (NIC – Network Interface Card)
- Communication réseau
- Transmission de données
- Réseau local (LAN)
- Internet
- Interface réseau
- Adresse MAC

Interfaces et technologies

- Ethernet
- RJ45
- Débit (100 Mbps / 1 Gbps / 2.5 Gbps / 10 Gbps)
- Wi-Fi
- Normes Wi-Fi (802.11n / ac / ax)
- Bande de fréquence (2.4 GHz / 5 GHz / 6 GHz)
- Latence
- Stabilité vs mobilité

Fonctionnement

- Encapsulation des données
- Paquets réseau
- Adresse IP vs MAC
- Communication avec routeur / switch
- Pilotes (drivers)

Performances

- Débit théorique vs réel
- Latence
- Interférences (Wi-Fi)
- Qualité du signal
- Distance / obstacles

Compatibilité matérielle

- Carte intégrée (carte mère)
- Carte PCIe
- Adaptateur USB
- Compatibilité OS (Windows / Linux)
- Drivers

Marché et constructeurs

- Intel (cartes Ethernet)

- Realtek
- Broadcom
- TP-Link
- ASUS
- Netgear

Types de produits :

- Carte interne (PCIe)
- Clé USB Wi-Fi
- Carte réseau intégrée

Usages concrets

- Navigation Internet
- Gaming en ligne
- Streaming
- Travail en réseau (entreprise)
- Mobilité (Wi-Fi vs câble)

Consignes de contenu par page

Page d'accueil – Présentation de la carte réseau

- Définition simple et reformulée
- Rôle dans un ordinateur
- Pourquoi c'est indispensable aujourd'hui
- Différence rapide entre Ethernet (RJ45) et Wi-Fi
- Illustration (schéma simple : PC → réseau → Internet)
- Présentation des différentes pages du site

Page Fonctionnement

- Comment un ordinateur envoie et reçoit des données (version vulgarisée)
- Notion de paquet réseau (sans rentrer trop profondément)
- Différence entre adresse IP et adresse MAC
- Fonctionnement d'une connexion Ethernet
- Fonctionnement du Wi-Fi (ondes radio, simplifié)
- Interaction avec routeur et switch

Page Critères techniques

Présente les critères importants pour choisir une carte réseau.

Pour chaque critère :

- définition
- exemple concret
- impact réel

Critères attendus :

- débit (1 Gbps, 2.5 Gbps…)
- latence
- type de connexion (filaire vs sans fil)
- normes Wi-Fi
- portée
- stabilité
- compatibilité

👉 Inclure des erreurs fréquentes :

- "Le Wi-Fi est toujours suffisant"
- "Plus de débit = toujours utile"
- "USB = toujours moins performant"

Page Marché

- Présentation des principaux constructeurs
- Différence entre :
  - carte intégrée
  - carte PCIe
  - adaptateur USB
- Évolution récente :
  - Wi-Fi 6 / Wi-Fi 6E
  - montée du 2.5 Gbps
  - disparition progressive du 100 Mbps
- Cas particulier :
  - Wi-Fi vs Ethernet en entreprise
  - importance du câblage

Page Comparatif

Tu dois comparer au minimum 3 modèles :

- 1 entrée de gamme
- 1 milieu de gamme
- 1 haut de gamme

Types possibles :

- carte Ethernet PCIe
- carte Wi-Fi PCIe
- clé USB Wi-Fi

TABLEAU COMPARATIF :

- type (Ethernet / Wi-Fi)
- débit
- interface (PCIe / USB)
- norme Wi-Fi (si applicable)
- prix approximatif

ANALYSE RÉDIGÉE :

- avantages / inconvénients
- cas d'usage
- choix personnel justifié

Points d'attention pédagogiques

- Ne pas confondre Internet et réseau local
- Toujours relier technique ↔ usage réel
- Expliquer avec des exemples concrets (maison, école, entreprise)
- Ne pas rester en surface (éviter "ça sert à Internet")
- Comparer intelligemment (stabilité vs mobilité, pas juste vitesse)

Site complet: Réseau local (LAN)

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment plusieurs machines communiquent entre elles dans un réseau local, comprendre les équipements nécessaires (switch, câblage…), et être capable de décrire ou concevoir un petit réseau (école, maison, entreprise).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Réseau local (LAN – Local Area Network)
- Communication entre machines
- Partage de ressources
- Réseau domestique vs réseau d'entreprise
- Différence LAN / Internet

Équipements réseau

- Switch
- Routeur (rappel rapide)
- Point d'accès Wi-Fi
- Câble Ethernet (RJ45)
- Baie de brassage (optionnel mais intéressant)
- Carte réseau

Topologies réseau

- Topologie en étoile (principale aujourd'hui)
- Bus (historique)
- Anneau (historique)
- Réseau maillé (mesh, vulgarisé)

Fonctionnement

- Communication entre machines
- Adresse IP
- Adresse MAC (rappel)
- DHCP (attribution automatique d'IP)
- Broadcast
- Paquets réseau (rappel simplifié)

Performances

- Débit réseau (100 Mbps, 1 Gbps, 10 Gbps)
- Latence
- Collision (notion simplifiée)
- Saturation réseau
- Différence filaire vs Wi-Fi

Câblage et normes

- RJ45
- Catégories de câble (Cat5e, Cat6, Cat6a…)
- Longueur maximale (~100m)
- Qualité du câble

Sécurité et gestion

- Réseau privé
- Pare-feu (notion simple)
- VLAN (optionnel, vulgarisé)
- Accès contrôlé

Usages concrets

- Partage de fichiers
- Imprimante réseau
- Serveur local
- Jeux en réseau
- Réseau scolaire / entreprise

Consignes de contenu par page

Page d'accueil – Présentation du LAN

- Définition claire d'un réseau local
- Différence entre LAN et Internet
- Exemple concret (maison, école)
- Schéma simple (plusieurs PC + switch + routeur)
- Présentation des pages du site

Page Fonctionnement

- Comment deux machines communiquent dans un LAN
- Rôle du switch (élément central)
- Explication simple IP / MAC
- DHCP : comment une machine obtient une IP
- Différence communication locale vs Internet

👉 Exemple concret attendu : "PC A envoie un fichier au PC B"

Page Architecture & matériel

- Description des équipements :
  - switch
  - câbles
  - point d'accès Wi-Fi
- Explication de la topologie en étoile
- Importance du câblage
- Différence réseau filaire vs sans fil

👉 Inclure un schéma clair obligatoire

Page Critères techniques

Critères à expliquer :

- débit réseau
- type de switch (non manageable vs manageable – vulgarisé)
- qualité du câblage
- stabilité
- évolutivité

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- “le Wi-Fi remplace totalement le câble”
- “tous les switches sont équivalents”
- “le débit Internet = débit du réseau local”

Page Marché

- Présentation des principaux constructeurs :
  - TP-Link
  - Netgear
  - Cisco
  - Ubiquiti
- Types de produits :
  - switch domestique
  - switch professionnel
  - solutions mesh
- Évolution récente :
  - démocratisation du gigabit
  - arrivée du 2.5 Gbps
  - réseaux mesh pour le Wi-Fi

Page Comparatif

Comparer au minimum 3 équipements réseau (ex : switches) :

- 1 entrée de gamme
- 1 milieu de gamme
- 1 plus avancé

TABLEAU COMPARATIF :

- nombre de ports
- débit (Fast Ethernet / Gigabit / 2.5G)
- manageable ou non
- prix

ANALYSE :

- avantages / inconvénients
- type d’utilisateur
- choix personnel justifié

Points d’attention pédagogiques

- Toujours partir d’un exemple concret (maison, école…)
- Ne pas mélanger LAN et Internet
- Bien comprendre le rôle du switch (souvent mal compris)
- Expliquer simplement avant de complexifier
- Utiliser des schémas (obligatoire ici)
- Ne pas tomber dans du Cisco avancé (pas de config CLI attendue)

Site complet: Modem, routeur et Internet

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment un réseau local accède à Internet, comprendre le rôle du modem et du routeur, et savoir décrire le chemin parcouru par une donnée depuis ton ordinateur jusqu'à un serveur distant.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Internet
- Réseau mondial
- Fournisseur d'accès Internet (FAI)
- Modem
- Routeur
- Accès Internet
- Différence LAN / WAN

Modem

- Modulation / démodulation
- Conversion du signal (numérique ↔ signal réseau opérateur)
- Types de connexions :
  - DSL
  - câble (coaxial)
  - fibre (FTTH)
- Box Internet

Routeur

- Routage des paquets
- NAT (Network Address Translation)
- Attribution d'adresses IP (DHCP)
- Gestion du réseau local
- Wi-Fi intégré

Fonctionnement global

- Chemin d'une donnée :
  - PC → routeur → modem → FAI → Internet → serveur
- Adresse IP publique / privée
- DNS (résolution de nom de domaine)
- Paquets réseau (rappel)

Performances

- Débit Internet (download / upload)
- Latence (ping)
- Bande passante
- Saturation réseau

Sécurité

- Pare-feu (firewall)
- NAT comme protection indirecte
- Réseau domestique sécurisé
- Accès distant

Marché et équipements

- Box des opérateurs
- Routeurs personnels
- Modem séparé (plus rare chez particuliers)

Principaux acteurs :

- Proximus
- Orange Belgium
- Telenet
- TP-Link
- ASUS

Usages concrets

- Navigation web
- Streaming
- Jeux en ligne
- Télétravail
- Accès distant (NAS, caméras…)

Consignes de contenu par page

Page d'accueil – Présation générale

- Définition d'Internet (réseau mondial)
- Différence entre réseau local et Internet
- Présentation simple :
  - modem
  - routeur
- Schéma global obligatoire : PC → routeur → modem → Internet
- Présentation des pages

Page Fonctionnement

- Explication du trajet d'une donnée (très important)
- Rôle du modem (conversion du signal)
- Rôle du routeur (distribution du réseau)
- Explication IP publique / privée
- DNS (ex : google.com → IP)

👉 Exemple concret obligatoire : "Quand tu ouvres un site web, que se passe-t-il ?"

Page Modem vs Routeur

👉 Page clé (souvent source de confusion)

- Définition claire des deux
- Différences :
  - fonction
  - position dans le réseau
- Cas réel :
  - box Internet (modem + routeur combinés)

- Pourquoi on ne peut pas se passer de l'un des deux

👉 Erreurs fréquentes :

- "Le Wi-Fi = Internet"
- "Le routeur donne Internet"
- "Le modem = Wi-Fi"

Page Critères techniques

Critères à expliquer :

- débit Internet (Mbps)
- type de connexion (fibre, DSL…)
- qualité du routeur
- Wi-Fi (portée, normes)
- latence

Pour chaque critère :

- définition
- exemple concret
- impact réel

👉 Erreurs fréquentes :

- "avoir 1 Gbps = toujours utile"
- "le problème vient toujours d'Internet"
- confusion débit LAN vs Internet

Page Marché

- Présentation des FAI en Belgique :
  - Proximus
  - Telenet
  - Orange Belgium
- Types d'offres :
  - fibre
  - câble
  - DSL
- Évolution récente :
  - déploiement fibre
  - augmentation des débits
  - routeurs Wi-Fi plus performants
- Routeurs personnels vs box opérateur

Page Comparatif

Comparer au minimum 3 équipements :

- routeur entrée de gamme
- routeur milieu de gamme
- routeur haut de gamme

TABLEAU COMPARATIF :

- débit Wi-Fi
- normes Wi-Fi
- nombre de ports
- fonctionnalités (QoS, sécurité…)
- prix

ANALYSE :

- avantages / inconvénients
- usage recommandé
- choix personnel justifié

Points d’attention pédagogiques

- Bien distinguer :
  - Internet ≠ Wi-Fi
  - modem ≠ routeur
- Toujours expliquer avec un schéma
- Utiliser des exemples concrets (ouvrir un site, regarder Netflix…)
- Ne pas tomber dans des détails trop réseau (pas de routage avancé)
- Comprendre le “chemin de la donnée” = objectif principal

Site complet: Système d'exploitation

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer le rôle d'un système d'exploitation, comprendre comment il gère le matériel et les logiciels, distinguer les grandes familles d'OS (Windows, Linux, macOS, mobile…), et choisir un système adapté à un usage donné.

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Système d'exploitation (OS – Operating System)
- Interface entre matériel et logiciel
- Gestion des ressources
- Interface utilisateur (GUI / CLI)
- Noyau (kernel)
- Processus
- Services système

Fonctionnement interne

- Gestion du processeur (ordonnancement)
- Gestion de la mémoire (RAM)
- Gestion des fichiers (système de fichiers)
- Gestion des périphériques (drivers)
- Multitâche

Types de systèmes

- OS desktop
- OS mobile
- OS embarqué
- OS temps réel (mention simple)

Familles principales

- Windows
- Linux
- macOS
- iOS
- Android
- Android Auto
- Systèmes TV (Samsung – Tizen)

Interface utilisateur

- Interface graphique (GUI)
- Ligne de commande (CLI)
- Bureau (desktop environment)
- Applications

Sécurité et gestion

- Utilisateurs
- Droits / permissions
- Mises à jour
- Antivirus / sécurité intégrée

Performances et compatibilité

- Optimisation matérielle
- Compatibilité logicielle
- Écosystème
- Stabilité

Usages concrets

- Bureautique
- Développement
- Gaming
- Mobile
- Multimédia
- Systèmes embarqués (voiture, TV…)

Consignes de contenu par page

Page d'accueil – Présentation du système d'exploitation

- Définition claire et reformulée
- Rôle dans un ordinateur ou appareil
- Analogie simple (ex : chef d'orchestre, gestionnaire)
- Exemples d'appareils utilisant un OS (PC, smartphone, TV…)
- Présentation des différentes pages

Page Fonctionnement

- Comment un OS démarre (boot simplifié)
- Rôle du kernel
- Gestion des ressources (CPU, RAM, fichiers)
- Interaction avec le matériel (drivers)
- Multitâche expliqué simplement

👉 Exemple concret : "Que se passe-t-il quand tu ouvres une application ?"

Page Windows

- Présentation de Microsoft Windows
- Rôle et position dominante sur PC

ÉDITIONS PRINCIPALES :

- Windows Home
- Windows Pro
- Windows Enterprise

POINTS À ABORDER :

- Interface graphique
- Compatibilité logicielle (très large)
- Usage (grand public, entreprise)
- Points forts / limites

Page Linux

- Présentation de Linux
- Notion d'open source

DISTRIBUTIONS :

- Ubuntu
- Debian
- Fedora
- 3 autres distributions grand public au choix

POINTS À ABORDER :

- Modularité
- Utilisation en serveur vs desktop
- Ligne de commande
- Liberté / personnalisation
- Difficulté d'accès (relative)

▸ Page Apple (macOS et iOS)

- Présentation de macOS
- Présentation de iOS

POINTS À ABORDER :

- Écosystème Apple
- Optimisation matériel / logiciel
- Simplicité d'utilisation
- Limitation (système fermé)

▸ Page Android & systèmes embarqués

- Présentation de Android
- Présentation de Android Auto
- Présentation de Tizen

POINTS À ABORDER :

- Android (mobile, ouvert, constructeurs multiples)
- Android Auto (voiture, interface simplifiée)
- Tizen (TV Samsung, usage spécifique)

▸ Page Critères techniques

Critères à expliquer :

- compatibilité logicielle
- facilité d'utilisation
- performance
- sécurité
- écosystème
- personnalisation

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- "Linux = compliqué"
- "Mac = plus puissant"
- "Tous les OS se valent"
- confusion OS / application

▸ Page Comparatif

Comparer au minimum 3 systèmes :

- Windows
- Linux (une distribution)
- macOS

TABLEAU COMPARATIF :

- type (ouvert / fermé)
- facilité d'utilisation
- compatibilité logicielle
- personnalisation
- usage typique

ANALYSE :

- avantages / inconvénients
- type d'utilisateur
- choix personnel justifié

Points d'attention pédagogiques

- Ne pas confondre OS et logiciels (Word, Chrome…)
- Toujours relier à un usage concret
- Expliquer simplement avant de détailler
- Éviter les opinions non justifiées ("le meilleur OS")
- Montrer qu'il n'existe pas un OS unique pour tous les usages

Site complet: Carte son

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment le son est traité dans un ordinateur, comprendre le rôle d'une carte son, distinguer les différents types de cartes audio, et choisir une solution adaptée selon l'usage (jeux, musique, enregistrement, streaming…).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Carte son
- Audio numérique / audio analogique
- Conversion du signal
- Entrée / sortie audio
- Traitement du son
- Interface audio

Fonctionnement

- DAC (Digital to Analog Converter)
- ADC (Analog to Digital Converter)
- Signal audio
- Fréquence d'échantillonnage (Hz)
- Résolution (bits)
- Latence audio

Connectique

- Jack 3.5 mm
- Casque / micro
- Entrée ligne (line-in)
- Sortie ligne (line-out)
- S/PDIF (optique)
- USB (cartes externes)

Types de cartes son

- Carte intégrée (carte mère)
- Carte son interne (PCIe)
- Carte son externe (USB)
- Interface audio professionnelle

Performances

- Qualité audio
- Bruit (signal/bruit)
- Latence
- Nombre de canaux (stéréo, 5.1, 7.1)
- Puissance de sortie

Compatibilité

- Drivers
- Compatibilité OS
- Logiciels audio (DAW, jeux…)

Marché et constructeurs

- Realtek (intégré)
- Creative
- Focusrite
- Behringer
- ASUS

Usages concrets

- Écoute musicale
- Gaming
- Streaming
- Enregistrement audio (micro, instruments)
- Production musicale

Consignes de contenu par page

Page d'accueil – Présentation de la carte son

- Définition claire et reformulée
- Rôle dans un ordinateur
- Pourquoi elle est souvent "invisible" (intégrée)
- Exemple concret (écouter de la musique, parler sur Discord…)
- Présentation des pages du site

Page Fonctionnement

- Explication simple :
  - conversion numérique → analogique (DAC)
  - conversion analogique → numérique (ADC)
- Exemple concret :
  - "comment un fichier MP3 devient du son dans un casque"
- Notion de fréquence et résolution (vulgarisée)
- Interaction avec le CPU et les logiciels

Page Types de cartes son

- Carte intégrée (Realtek…)
- Carte PCIe
- Carte externe USB
- Interface audio pro

Pour chaque type :

- description
- avantages
- inconvénients
- usage typique

👉 Exemple concret attendu :

- gamer
- musicien
- utilisateur basique

Page Critères techniques

Critères à expliquer :

- qualité audio (bits / Hz)
- latence
- nombre d'entrées / sorties

- type de connexion
- nombre de canaux (stéréo / surround)

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- “plus de Hz = toujours mieux”
- “une carte son dédiée est toujours nécessaire”
- confusion qualité fichier vs qualité carte son

▸ Page Marché

- Présentation des principaux constructeurs :
  - Creative
  - Focusrite
  - Behringer
  - ASUS
- Différence :
  - grand public vs professionnel
- Évolution :
  - disparition des cartes dédiées chez le grand public
  - montée des interfaces USB
  - importance du streaming / podcast

▸ Page Comparatif

Comparer au minimum 3 modèles :

- 1 solution intégrée (Realtek ou équivalent)
- 1 carte son externe simple
- 1 interface audio (type studio)

TABLEAU COMPARATIF :

- type (intégrée / USB / PCIe)
- qualité audio (bits / Hz)
- entrées / sorties
- usage
- prix

ANALYSE :

- avantages / inconvénients
- usage recommandé
- choix personnel justifié

Points d’attention pédagogiques

- Ne pas confondre qualité du son et qualité du fichier audio
- Bien comprendre le rôle de conversion (DAC / ADC)
- Toujours relier à un usage réel
- Ne pas surévaluer l’importance pour un usage basique
- Expliquer simplement avant d’aller dans les détails techniques

Site complet: Imprimantes

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment une imprimante transforme un document numérique en support papier, comprendre les différentes technologies d'impression (jet d'encre, laser, thermique…), et choisir une imprimante adaptée selon un usage (maison, école, entreprise, tickets…).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Imprimante
- Impression papier
- Document numérique → document physique
- Périphérique de sortie
- Qualité d'impression

Technologies d'impression

- Jet d'encre
- Laser
- Thermique
- Sublimation thermique (photo)
- Impact (historique, mention courte)

Fonctionnement

- Projection d'encre (jet d'encre)
- Toner et laser (imprimante laser)
- Chaleur (imprimante thermique)
- Résolution (DPI)
- Gestion des couleurs (CMYK)

Consommables

- Cartouches d'encre
- Toner
- Papier
- Rouleaux thermiques

Performances

- Vitesse d'impression (pages/minute)
- Qualité (DPI)
- Coût par page
- Temps de chauffe (laser)
- Fiabilité

Connectivité

- USB
- Wi-Fi
- Ethernet (réseau)
- Impression mobile

Compatibilité

- Drivers
- Compatibilité OS
- Logiciels d'impression

Marché et constructeurs

- HP
- Canon
- Epson
- Brother

Usages concrets

- Bureautique (documents)
- Impression photo
- Impression professionnelle
- Tickets / reçus (thermique)
- Étiquettes

Consignes de contenu par page

Page d'accueil – Présentation des imprimantes

- Définition claire et reformulée
- Rôle dans un système informatique
- Exemples d'utilisation (maison, école, magasin…)
- Présentation rapide des principales technologies
- Présentation des pages du site

Page Fonctionnement

- Comment une imprimante reçoit un document
- Transformation en impression papier
- Explication simplifiée des 3 technologies principales :
  - jet d'encre
  - laser
  - thermique
- Notion de DPI (qualité)
- Gestion des couleurs (CMYK simplifié)

👉 Exemple concret : "imprimer un document Word"

Page Technologies

👉 Page centrale du travail

JET D'ENCRE

- fonctionnement (projection d'encre)
- avantages :
  - bonne qualité photo
  - prix d'achat faible
- inconvénients :
  - encre coûteuse
  - séchage / entretien

LASER

- fonctionnement (toner + laser)
- avantages :

    - rapide
    - économique à long terme
- inconvénients :
    - prix initial plus élevé

THERMIQUE

- fonctionnement (chaleur)
- avantages :
    - pas d'encre
    - fiable
- inconvénients :
    - usage limité (tickets)

AUTRES (SECTION COURTE OBLIGATOIRE)

- sublimation thermique (photo)
- imprimantes industrielles
- imprimantes 3D (mention simple, sans détailler)

▸ Page Critères techniques

Critères à expliquer :

- vitesse (ppm)
- qualité (DPI)
- coût par page
- type de consommable
- connectivité
- usage (volume d'impression)

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- "le prix d'achat est le plus important"
- "toutes les imprimantes se valent"
- "le jet d'encre est toujours moins cher"

▸ Page Marché

- Présentation des principaux constructeurs :
    - HP
    - Canon
    - Epson
    - Brother
- Différence :
    - grand public vs professionnel
- Tendances :
    - imprimantes connectées
    - abonnement encre
    - baisse de l'impression papier
    - montée du thermique (commerce)

▸ Page Comparatif

Comparer au minimum 3 modèles :

- 1 jet d'encre
- 1 laser
- 1 thermique (ou spécialisé)

TABLEAU COMPARATIF :

- technologie
- vitesse
- qualité (DPI)
- coût par page
- usage
- prix

ANALYSE :

- avantages / inconvénients
- usage recommandé
- choix personnel justifié

Points d'attention pédagogiques

- Toujours relier la technologie à un usage concret
- Bien distinguer les technologies (souvent confondues)
- Ne pas se limiter aux caractéristiques marketing
- Comprendre le coût réel (consommables)
- Expliquer simplement avant de détailler

Site complet: Claviers

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment un clavier permet de communiquer avec un ordinateur, comprendre les différentes technologies de claviers (membrane, mécanique…), et choisir un clavier adapté selon un usage (bureautique, gaming, programmation, usages spécialisés…).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Clavier
- Périphérique d'entrée
- Saisie de texte
- Interaction utilisateur
- Interface homme-machine

Fonctionnement

- Touche
- Signal électrique
- Matrice de touches
- Contrôleur
- Envoi des données au système
- Keycode

Technologies de claviers

- Clavier à membrane
- Clavier mécanique
- Switch (linéaire, tactile, clicky)
- Clavier chiclet (ordinateur portable)
- Clavier optique (mention)

Disposition des touches

- AZERTY
- QWERTY
- QWERTZ
- Disposition ergonomique

Connectivité

- USB
- Bluetooth
- Sans fil (dongle)
- Latence

Performances et confort

- Réactivité
- Bruit
- Durabilité
- Rétroéclairage
- Ergonomie

Claviers spécialisés

- Claviers gaming
- Claviers ergonomiques
- Claviers compacts (60%, TKL…)
- Claviers programmables
- Claviers professionnels (montage, dev…)

Marché et constructeurs

- Logitech
- Corsair
- Razer
- SteelSeries
- Microsoft

Usages concrets

- Bureautique
- Gaming
- Programmation
- Rédaction intensive
- Accessibilité (handicap, ergonomie)

Consignes de contenu par page

Page d'accueil – Présentation du clavier

- Définition claire et reformulée
- Rôle dans un ordinateur
- Exemples d'utilisation
- Différents types de claviers (aperçu rapide)
- Présentation des pages du site

Page Fonctionnement

- Comment une touche envoie une information
- Explication simple de la matrice de clavier
- Notion de keycode
- Interaction avec le système d'exploitation

👉 Exemple concret : "Que se passe-t-il quand tu appuies sur la touche A ?"

Page Technologies

CLAVIER À MEMBRANE

- fonctionnement
- avantages :
  - silencieux
  - peu cher
- inconvénients :
  - moins précis
  - moins durable

CLAVIER MÉCANIQUE

- fonctionnement (switch)
- types de switches :
  - linéaire
  - tactile
  - clicky

- avantages :
  - précision
  - durabilité
- inconvénients :
  - bruit
  - prix

AUTRES TYPES (SECTION OBLIGATOIRE COURTE)

- chiclet (ordinateurs portables)
- optique (gaming)
- low profile

Page Critères techniques

Critères à expliquer :

- type de clavier (membrane / mécanique)
- ergonomie
- disposition (AZERTY…)
- connectivité
- bruit
- durabilité
- fonctionnalités (RGB, macros…)

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- “mécanique = toujours meilleur”
- “plus cher = meilleur”
- “RGB = performance”

Page Claviers spécialisés

- Claviers gaming (réactivité, RGB, macros)
- Claviers ergonomiques (forme, confort)
- Claviers compacts (gain de place)
- Claviers programmables (touches personnalisées)

👉 Pour chaque type :

- description
- avantages
- usage concret

Page Marché

- Présentation des principaux constructeurs :
  - Logitech
  - Corsair
  - Razer
  - SteelSeries
  - Microsoft
- Différence :
  - grand public vs gaming vs pro
- Évolution :
  - montée des claviers mécaniques
  - personnalisation
  - sans fil performant

Page Comparatif

Comparer au minimum 3 modèles :

- 1 clavier bureautique (membrane)
- 1 clavier mécanique
- 1 clavier spécialisé (ergonomique ou gaming)

TABLEAU COMPARATIF :

- type
- connectivité
- ergonomie
- fonctionnalités
- prix

ANALYSE :

- avantages / inconvénients
- usage recommandé
- choix personnel justifié

Points d'attention pédagogiques

- Ne pas réduire le clavier à "juste écrire"
- Bien distinguer les technologies
- Relier chaque type à un usage concret
- Ne pas se limiter au design (RGB, esthétique)
- Expliquer le confort et l'ergonomie (important en usage réel)

Site complet: Souris

Souris, trackball et dispositifs équivalents

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment un dispositif de pointage permet d'interagir avec un ordinateur, comprendre les différentes technologies (souris, trackball, pavé tactile…), et choisir un dispositif adapté selon un usage (bureautique, gaming, précision, accessibilité…).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Dispositif de pointage
- Souris
- Déplacement du curseur
- Interaction graphique
- Interface homme-machine

Fonctionnement

- Capteur optique / laser
- Détection du mouvement
- DPI (sensibilité)
- Boutons (clic gauche, droit, molette)
- Signal envoyé au système

Types de dispositifs

- Souris classique
- Souris gaming
- Trackball
- Pavé tactile (touchpad)
- Stylet (mention)
- Souris verticale

Technologies

- Optique
- Laser
- Sans fil / filaire
- Bluetooth / dongle USB
- Polling rate (fréquence de rafraîchissement)

Performances

- DPI (sensibilité)
- Précision
- Latence
- Réactivité
- Accélération

Ergonomie

- Forme (droitiers / gauchers)

- Prise en main (palm, claw, fingertip)
- Confort
- Fatigue
- Troubles musculo-squelettiques (TMS)

Connectivité

- USB
- Bluetooth
- Sans fil (2.4 GHz)
- Batterie / pile

Marché et constructeurs

- Logitech
- Razer
- SteelSeries
- Corsair
- Microsoft
- ...

Usages concrets

- Bureautique
- Gaming
- Graphisme / précision
- Navigation web
- Accessibilité

Consignes de contenu par page

Page d'accueil – Présentation du dispositif de pointage

- Définition claire et reformulée
- Rôle dans un ordinateur
- Exemples d'utilisation
- Présentation des différents types (aperçu)
- Présentation des pages du site

Page Fonctionnement

- Comment une souris détecte le mouvement
- Explication du capteur optique / laser
- Notion de DPI (sensibilité)
- Interaction avec le système (curseur)

👉 Exemple concret : "déplacer le curseur et cliquer sur une icône"

Page Types de dispositifs

SOURIS CLASSIQUE

- description
- usage courant

SOURIS GAMING

- haute précision
- DPI élevé
- boutons supplémentaires

TRACKBALL

- fonctionnement (boule fixe)

- avantages :
  - précision
  - moins de mouvement du bras
- inconvénients :
  - prise en main

PAVÉ TACTILE (TOUCHPAD)

- fonctionnement
- gestes (multitouch)

AUTRES (SECTION OBLIGATOIRE COURTE)

- stylet (tablette graphique)
- souris verticale
- dispositifs spécialisés (accessibilité)

▸ Page Critères techniques

Critères à expliquer :

- DPI
- latence
- type de capteur
- connectivité
- ergonomie
- autonomie

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- “plus de DPI = toujours mieux”
- “sans fil = moins performant”
- “toutes les souris se valent”

▸ Page Ergonomie & usages

- Importance du confort
- Différentes prises en main
- Souris verticales (prévention TMS)
- Trackball (cas spécifiques)
- Choix selon usage :
  - bureautique
  - gaming
  - précision

▸ Page Marché

- Présentation des principaux constructeurs :
  - Logitech
  - Razer
  - SteelSeries
  - Corsair
  - Microsoft
- Différence :
  - bureautique vs gaming
- Évolution :
  - sans fil performant
  - capteurs plus précis
  - ergonomie améliorée

Page Comparatif

Comparer au minimum 3 modèles :

- 1 souris bureautique
- 1 souris gaming
- 1 dispositif alternatif (trackball ou verticale)

TABLEAU COMPARATIF :

- type
- DPI
- connectivité
- ergonomie
- autonomie
- prix

ANALYSE :

- avantages / inconvénients
- usage recommandé
- choix personnel justifié

Points d'attention pédagogiques

- Ne pas réduire la souris à un simple déplacement du curseur
- Bien comprendre les notions de précision et de latence
- Relier ergonomie ↔ santé (important)
- Ne pas se focaliser uniquement sur le gaming
- Comparer en fonction de l'usage réel, pas uniquement des chiffres

Site complet: Tablettes graphiques

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment une tablette graphique permet d'interagir avec un ordinateur pour dessiner ou écrire, comprendre les différentes technologies (avec ou sans écran), et choisir une tablette adaptée selon un usage (dessin, retouche photo, prise de notes, design…).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Tablette graphique
- Périphérique d'entrée
- Dessin numérique
- Stylet (pen)
- Interaction précise
- Remplacement de la souris (cas spécifique)

Fonctionnement

- Surface active
- Détection du stylet
- Position (coordonnées)
- Pression (niveaux de pression)
- Inclinaison (tilt)
- Transmission vers le logiciel

Technologies

- Tablette sans écran (surface + écran PC)
- Tablette avec écran (display tablet)
- Tablette autonome (type tablette tactile)
- Stylet actif (sans batterie / avec batterie)

Performances

- Niveaux de pression (ex : 4096, 8192)
- Résolution (LPI)
- Taux de rafraîchissement (report rate)
- Latence
- Précision

Ergonomie

- Taille de la surface active
- Position de la main
- Confort d'utilisation
- Raccourcis (touches programmables)
- Écran (pour les modèles avec affichage)

Connectivité

- USB
- Bluetooth
- Sans fil
- Compatibilité OS

Compatibilité logicielle

- Logiciels de dessin (Photoshop, Krita…)
- Logiciels de retouche
- Logiciels 3D (mention)
- Prise de notes

Marché et constructeurs

- Wacom
- XP-Pen
- Huion
- Apple (iPad + Apple Pencil)

Usages concrets

- Dessin numérique
- Illustration
- Retouche photo
- Signature numérique
- Prise de notes
- Design graphique

Consignes de contenu par page

Page d'accueil – Présentation de la tablette graphique

- Définition claire et reformulée
- Rôle dans un système informatique
- Différence avec une souris
- Exemples d'utilisation
- Présentation des pages du site

Page Fonctionnement

- Comment la tablette détecte le stylet
- Explication de la pression (plus tu appuies, plus le trait est épais)
- Notion de position et précision
- Interaction avec les logiciels

👉 Exemple concret : "dessiner un trait dans un logiciel de dessin"

Page Types de tablettes

TABLETTE SANS ÉCRAN

- fonctionnement
- avantages :
  - prix
  - simplicité
- inconvénients :
  - coordination œil-main

TABLETTE AVEC ÉCRAN

- fonctionnement
- avantages :
  - dessin direct
- inconvénients :
  - prix

- encombrement

TABLETTE AUTONOME (SECTION COURTE OBLIGATOIRE)

- ex : iPad avec stylet
- usage hybride (tablette + ordinateur)

Page Critères techniques

Critères à expliquer :

- niveaux de pression
- taille
- précision
- latence
- type de tablette (avec / sans écran)
- compatibilité logicielle

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- “plus de pression = toujours utile”
- “une tablette avec écran est toujours meilleure”
- “c’est uniquement pour les professionnels”

Page Marché

- Présentation des principaux constructeurs :
  - Wacom
  - XP-Pen
  - Huion
  - Apple
- Différence :
  - entrée de gamme vs professionnel
- Évolution :
  - démocratisation
  - amélioration des stylets
  - concurrence accrue (moins cher que Wacom)

Page Comparatif

Comparer au minimum 3 modèles :

- 1 tablette sans écran (entrée de gamme)
- 1 tablette avec écran
- 1 solution type tablette autonome (iPad ou équivalent)

TABLEAU COMPARATIF :

- type
- taille
- niveaux de pression
- connectivité
- usage
- prix

ANALYSE :

- avantages / inconvénients
- usage recommandé
- choix personnel justifié

Points d’attention pédagogiques

- Ne pas confondre tablette graphique et tablette tactile classique
- Toujours relier les caractéristiques à un usage concret
- Expliquer la notion de pression (clé du sujet)
- Ne pas se limiter au dessin (prise de notes, design…)
- Comparer intelligemment (prix vs usage réel)

Site complet: Périphériques MIDI

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment un périphérique MIDI permet de contrôler de la musique numérique, comprendre la différence entre MIDI et audio, et choisir un équipement adapté selon un usage (composition, live, production musicale…).

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- MIDI (Musical Instrument Digital Interface)
- Données musicales (notes, vélocité…)
- Contrôle d'instruments virtuels
- Différence MIDI / audio
- Interface de contrôle

Fonctionnement

- Message MIDI (note on / note off)
- Vélocité (force de frappe)
- Canal MIDI
- Horloge MIDI (tempo)
- Transmission de données (pas de son)

Types de périphériques MIDI

- Clavier maître (MIDI keyboard)
- Pads (contrôleurs de pads)
- Contrôleurs (knobs, faders)
- Contrôleurs DJ
- Interfaces MIDI

Connectivité

- USB
- MIDI DIN (5 broches)
- Bluetooth MIDI
- Interface audio + MIDI

Performances

- Latence
- Précision
- Sensibilité (touches)
- Nombre de contrôles

Compatibilité

- Logiciels (DAW : Ableton, FL Studio…)
- Systèmes d'exploitation
- Plug & Play

Marché et constructeurs

- Akai Professional

- Novation
- Native Instruments
- Arturia
- Roland

Usages concrets

- Composition musicale
- Production (MAO)
- Live performance
- DJing
- Contrôle de logiciels

Consignes de contenu par page

Page d'accueil – Présentation des périphériques MIDI

- Définition claire et reformulée
- Rôle dans un système informatique
- Différence entre MIDI et audio (important)
- Exemples d'utilisation (musique, DJ…)
- Présentation des pages du site

Page Fonctionnement

- Explication simple du MIDI :
  - envoi de notes
  - pas de son transmis
- Exemple concret :
  - "appuyer sur une touche → son produit par logiciel"
- Notion de vélocité
- Canal MIDI
- Interaction avec un logiciel (DAW)

Page Types de périphériques

CLAVIER MIDI

- description
- usage (composition)

PADS

- description
- usage (beats, samples)

CONTRÔLEURS (KNOBS / FADERS)

- description
- usage (mixage, effets)

CONTRÔLEURS DJ (SECTION COURTE)

- utilisation live
- interaction avec logiciel DJ

Page Critères techniques

Critères à expliquer :

- nombre de touches / pads
- sensibilité (vélocité)
- connectivité
- compatibilité logicielle

- latence
- nombre de contrôles

Pour chaque critère :

- définition
- exemple
- impact réel

👉 Erreurs fréquentes :

- “le MIDI produit du son”
- “plus de touches = toujours mieux”
- “USB = toujours suffisant”

▸ Page Marché

- Présentation des principaux constructeurs :
  - Akai Professional
  - Novation
  - Native Instruments
  - Arturia
  - Roland
- Différence :
  - débutant vs professionnel
- Évolution :
  - intégration avec logiciels
  - contrôleurs hybrides
  - production à domicile

▸ Page Comparatif

Comparer au minimum 3 modèles de :

- clavier MIDI simple
- contrôleur à pads
- contrôleur plus avancé

TABLEAU COMPARATIF :

- type
- nombre de contrôles
- connectivité
- compatibilité
- usage
- prix

ANALYSE :

- avantages / inconvénients
- usage recommandé
- choix personnel justifié

Points d’attention pédagogiques

- Bien comprendre que le MIDI ne transporte pas de son
- Toujours relier au logiciel utilisé (DAW)
- Expliquer avec des exemples concrets (appuyer sur une touche)
- Ne pas entrer dans des détails trop techniques (protocole avancé)
- Mettre en avant les usages réels (création musicale)

Site complet: Streaming Vidéo

Objectif spécifique du composant

À la fin du travail, tu dois être capable d'expliquer comment fonctionne une installation de streaming complète, comprendre le rôle des différents éléments matériels et logiciels, distinguer les principales plateformes et protocoles utilisés, et proposer une configuration adaptée selon un usage concret (gaming, événementiel, interview, cours en direct, podcast vidéo, concert, conférence…). (support.google.com)

Notions essentielles à aborder (mots-clés directeurs)

Ces mots-clés doivent apparaître explicitement ou implicitement dans ton site.

Rôle et principe général

- Streaming
- Diffusion en direct
- Captation vidéo
- Captation audio
- Encodage
- Scène
- Source
- Diffusion en ligne
- Flux vidéo
- Régie

Chaîne technique du streaming

- Caméra
- Micro
- Carte d'acquisition
- Mélangeur audio
- Carte son / interface audio
- Logiciel de streaming
- Encodage vidéo
- Encodage audio
- Débit montant (upload)
- Plateforme de diffusion

Matériel vidéo

- Webcam
- Caméra hybride / reflex / caméscope
- HDMI
- SDI (mention simple)
- Carte d'acquisition USB
- Carte d'acquisition PCIe
- Capture console / PC
- Éclairage 3 points
- Trépied

Matériel audio

- Micro dynamique
- Micro statique / condensateur
- Interface audio
- Carte son
- Casque
- In-ear monitoring
- Mixage audio

- Filtre anti-pop
- Bras articulé

Contrôle et production

- Stream Deck
- Raccourcis
- Changement de scènes
- Overlays
- Alertes
- Widgets
- Titres
- Transition
- Monitoring

Logiciels

- OBS Studio
- Streamlabs Desktop
- Sources
- Scènes
- Plugins
- Widgets
- Enregistrement local
- Diffusion simultanée

OBS Studio est un logiciel gratuit et extensible, organisé autour des scènes et des sources. Streamlabs Desktop met davantage l'accent sur l'intégration d'overlays, d'alertes et de widgets prêts à l'emploi. (obsproject.com)

Plateformes

- Twitch
- YouTube Live
- Facebook Live (mention possible)
- Plateformes spécialisées
- Audience
- Chat en direct
- Modération

YouTube recommande l'usage de **RTMPS** pour sécuriser l'ingest live. Twitch utilise **RTMP** pour envoyer le flux vers ses serveurs d'ingest. (support.google.com)

Protocoles et transport

- RTMP
- RTMPS
- Ingest server
- Bitrate
- H.264
- H.265 (mention simple)
- HLS (mention simple)
- Résolution
- Framerate

YouTube prend en charge RTMPS et mentionne aussi HLS dans certains cas, notamment pour certains workflows avancés. Twitch documente l'envoi du flux vidéo vers ses serveurs via RTMP et publie aussi des endpoints d'ingest. (support.google.com)

Usages concrets

- Streaming gaming
- Podcast vidéo
- Interview
- Live musical
- Captation d'événement
- Formation en direct

- Streaming multi-caméra
- Streaming mobile

Consignes de contenu par page

▸ Page d'accueil – Présentation du streaming

- Définition claire et reformulée du streaming
- Différence entre enregistrement vidéo et diffusion en direct
- Présentation simple de la chaîne complète : caméra + son + logiciel + plateforme
- Exemples d'usages réels : gaming, cours, conférence, concert, émission
- Présentation des pages du site

▸ Page Fonctionnement global

- Expliquer le trajet complet du signal : image + son → capture → traitement → encodage → envoi → plateforme → spectateur
- Distinguer clairement :
  - captation
  - traitement
  - encodage
  - diffusion
- Expliquer le rôle du PC de streaming
- Expliquer la notion de débit montant et les débits minimums et conseillés
- Expliquer simplement ce qu'est un flux live

👉 Exemple concret obligatoire : "Que se passe-t-il entre la caméra du streamer et l'écran du spectateur ?"

▸ Page Matériel vidéo

- Présenter les différents types de caméras :
  - webcam
  - appareil photo / hybride
  - caméscope
  - caméra d'action (mention possible)
- Expliquer le rôle des cartes d'acquisition :
  - USB
  - PCIe
- Expliquer quand une carte d'acquisition est utile
- Présenter les câbles et connectiques utiles :
  - HDMI
  - USB
  - éventuellement SDI en mention courte
- Expliquer l'importance du cadrage et de la stabilité

ÉCLAIRAGE — SECTION OBLIGATOIRE

Présenter le principe de l'éclairage **3 points** :

- key light
- fill light
- back light

Pour chaque lumière :

- rôle
- emplacement général
- effet recherché

👉 Il faut montrer qu'un bon streaming ne dépend pas seulement de la caméra, mais aussi énormément de la lumière.

Page Matériel audio

- Expliquer pourquoi le son est souvent encore plus important que l'image
- Présenter :
  - micros USB
  - micros XLR
  - interfaces audio / cartes son
  - casques
  - in-ear monitoring
- Distinguer :
  - micro dynamique
  - micro statique / condensateur
- Expliquer à quoi sert une interface audio
- Expliquer le monitoring :
  - casque classique
  - in-ear
- Relier chaque matériel à un usage concret

👉 Exemple concret attendu :

- streamer gaming seul
- interview à deux
- live musical
- podcast vidéo

Page Logiciels de streaming

- Présenter OBS Studio
- Présenter Streamlabs Desktop
- Expliquer la logique :
  - scènes
  - sources
  - transitions
  - overlays
  - alertes
- Montrer que OBS Studio est très modulaire et extensible via plugins et scripts, tandis que Streamlabs Desktop met davantage l'accent sur les outils intégrés pour les créateurs. (obsproject.com)
- Expliquer le rôle d'un Stream Deck ou équivalent
- Montrer comment on contrôle un live en pratique

👉 Erreurs fréquentes à signaler :

- "OBS fait tout tout seul"
- "le logiciel suffit, peu importe le matériel"
- "un beau stream = juste des overlays"

Page Plateformes et protocoles

- Présenter les principales plateformes :
  - Twitch
  - YouTube Live
  - éventuellement d'autres plateformes en mention
- Expliquer le principe d'une plateforme de streaming
- Expliquer ce qu'est un serveur d'ingest
- Expliquer les protocoles :
  - RTMP
  - RTMPS
- Expliquer pourquoi RTMPS est préférable quand il est disponible
- Distinguer :

- protocole de transport
- plateforme
- logiciel d'envoi

YouTube recommande explicitement RTMPS pour le chiffrement du flux. Twitch indique que le flux vidéo est envoyé à sa plateforme via RTMP. (support.google.com)

Page Critères techniques

Critères à expliquer :

- qualité vidéo (720p, 1080p, etc.)
- framerate (30 fps / 60 fps)
- bitrate
- qualité audio
- latence
- stabilité réseau
- nombre de caméras
- qualité de l'éclairage
- ergonomie de la régie
- simplicité d'utilisation

Pour chaque critère :

- définition
- exemple concret
- impact réel sur la qualité du stream

👉 Erreurs fréquentes :

- "plus de résolution = forcément meilleur stream"
- "une webcam 4K suffit sans bon éclairage"
- "le débit descendant Internet est le plus important"
- "le protocole RTMP produit lui-même une meilleure image"
- "le matériel le plus cher est forcément le plus adapté"

Page Marché

- Présentation des grandes familles de produits :
  - webcams
  - caméras
  - cartes d'acquisition
  - interfaces audio
  - micros
  - éclairage LED
  - contrôleurs type Stream Deck
- Présentation des principaux usages du marché :
  - créateurs de contenu
  - gamers
  - entreprises
  - écoles / formation
  - événementiel
- Évolution récente du secteur :
  - démocratisation du streaming
  - montée des productions semi-professionnelles à domicile
  - intégration d'overlays, widgets et alertes
  - importance croissante des outils de contrôle et d'automatisation (streamlabs.com)

Page Comparatif

Comparer au minimum 3 configurations ou 3 équipements selon un angle cohérent.

Exemples possibles :

- 3 webcams
- 3 cartes d'acquisition

- 3 micros
- ou 3 configurations complètes :
  - débutant
  - intermédiaire
  - avancé

TABLEAU COMPARATIF

Le tableau peut inclure :

- type de matériel
- usage
- connectique
- qualité / résolution
- points forts
- limites
- prix approximatif

ANALYSE RÉDIGÉE

- avantages et inconvénients
- usage conseillé
- cohérence de la configuration
- choix personnel justifié

Points d'attention pédagogiques

- Ne pas confondre **MIDI, audio, vidéo et streaming**
- Ne pas réduire le streaming à "une caméra et OBS"
- Toujours relier le matériel à un usage réel
- Montrer que la qualité dépend d'un ensemble : image, son, lumière, réseau, logiciel, organisation
- Bien distinguer :
  - plateforme
  - logiciel
  - protocole
  - matériel
- Expliquer simplement avant de détailler
- Éviter le catalogue marketing sans analyse
- Mettre en avant la logique de chaîne technique complète

Pistes de sous-thèmes à intégrer si besoin

- streaming gaming solo
- captation multicaméra
- podcast vidéo
- stream éducatif
- live musical
- régie légère pour petit événement
- monitoring audio avec casque ou in-ear
- usage d'un Stream Deck pour automatiser les transitions et actions
- importance des overlays, alertes et widgets dans les logiciels orientés créateurs (streamlabs.com)

HTML - L'approche Mobile First

> *L'approche **Mobile-First** (ou 'd'abord pour le mobile) est une méthodologie de conception et de développement web qui priorise les appareils mobiles dans la création de sites web ou d'applications. Avec l'augmentation constante de l'utilisation des smartphones pour naviguer sur Internet, adopter cette approche est devenu essentiel pour créer des expériences utilisateur optimales. Cet article explore le concept de Mobile-First, ses avantages, et comment le mettre en pratique efficacement.*

Qu'est-ce que l'Approche Mobile-First ?

L'approche Mobile-First consiste à concevoir d'abord un site web ou une application pour les petits écrans des appareils mobiles. Une fois que la version mobile est prête, les fonctionnalités et les mises en page sont progressivement enrichies pour s'adapter aux écrans plus grands, comme les tablettes et les ordinateurs.

En termes de développement, cela signifie :

1. Commencer par un design simple et épuré adapté aux petits écrans.
2. Ajouter des fonctionnalités et des styles supplémentaires en utilisant des **media queries** pour cibler les écrans plus grands.

Pourquoi Adopter l'Approche Mobile-First ?

Prédominance du Mobile

Selon les statistiques, une grande majorité des utilisateurs accèdent au web via des appareils mobiles. En adoptant une approche Mobile-First, vous garantissez une expérience utilisateur optimale pour la majorité de vos visiteurs.

Performance

Les appareils mobiles ont souvent des connexions Internet plus lentes et des ressources limitées (comme la mémoire ou le processeur). Concevoir en priorité pour ces contraintes permet de créer des sites légers et rapides, bénéfiques pour tous les appareils.

Accessibilité et Simplicité

Concevoir pour les petits écrans oblige à se concentrer sur l'essentiel : les éléments de base, les fonctionnalités clés, et une interface utilisateur claire. Cela améliore l'accessibilité et évite les designs surchargés.

Meilleur Référencement (SEO)

Google favorise les sites optimisés pour les mobiles dans ses classements, grâce à son indexation "Mobile-First". Un site conçu pour le mobile est donc un atout majeur pour votre visibilité en ligne.

Comment Mettre en Œuvre l'Approche Mobile-First ?

Planification et Priorisation

Avant même de coder, définissez les éléments essentiels de votre site ou application :

- Quels sont les objectifs principaux ?
- Quels contenus doivent absolument apparaître sur un petit écran ?
- Quelles fonctionnalités doivent être accessibles immédiatement ?

Concevoir pour les Petits Écrans

Commencez par une mise en page simple adaptée aux écrans étroits. Par exemple :

- Utilisez une colonne unique pour afficher les contenus.
- Optez pour des boutons suffisamment grands pour être facilement cliqués avec un doigt.
- Limitez les images et les éléments graphiques volumineux.

Utilisation de Media Queries

Ajoutez progressivement des styles et des fonctionnalités pour les écrans plus grands en utilisant les **media queries**. Voici un exemple de code CSS :

```css
/* Styles pour les appareils mobiles */
body {
  font-size: 16px;
  padding: 10px;
}

/* Styles pour les écrans de tablettes (≥768px) */
@media (min-width: 768px) {
  body {
    font-size: 18px;
    padding: 20px;
  }
}

/* Styles pour les écrans d'ordinateurs (≥1024px) */
@media (min-width: 1024px) {
  body {
    font-size: 20px;
    padding: 30px;
  }
}
```

Les média-queries ne font pas partie de la matière vue en classe. Il est important de savoir qu'elles sont à la base de l'approche mobile first, mais pas de comprendre, dans le cadre de ce cours, comment elles fonctionnent. Nous utilisons **Bootstrap** en classe afin d'explorer et de profiter de l'approche mobile first sans la complexité des media-queries.

Testez sur Différents Appareils

Testez régulièrement votre site sur des appareils mobiles, tablettes et ordinateurs pour vous assurer que le contenu s'adapte correctement à chaque taille d'écran.

Optimisez les Performances

- Minimisez les fichiers CSS et JavaScript.
- Utilisez des images optimisées pour le web (formats comme WebP ou compressés en JPEG/PNG).
- Chargez les ressources uniquement lorsque nécessaire (lazy loading).

Exemple Pratique : Création d'une Page Mobile-First avec Bootstrap

Bootstrap 5 est conçu pour être Mobile-First, ce qui facilite grandement la mise en œuvre de cette approche.

Exemple de Code HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Approche Mobile-First</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container">
    <header class="text-center my-3">
      <h1>Bienvenue</h1>
      <p>Ce site est conçu avec une approche Mobile-First.</p>
    </header>

    <div class="row">
      <div class="col-12 col-md-6 mb-3">
        <div class="p-3 bg-light">Section 1</div>
      </div>
      <div class="col-12 col-md-6 mb-3">
        <div class="p-3 bg-light">Section 2</div>
      </div>
    </div>
  </div>
</body>
</html>
```

▸ **Résultat**

- Sur les petits écrans (mobile) : Les deux sections s'affichent l'une sous l'autre.
- Sur les écrans moyens et grands (tablette et ordinateur) : Les deux sections s'affichent côte à côte grâce aux classes de grille Bootstrap.

Comparaison avec l'Approche Desktop-First

MOBILE-FIRST	DESKTOP-FIRST
Priorise les petits écrans	Priorise les grands écrans
Ajout de styles pour les écrans plus grands	Suppression de styles pour les petits écrans
Optimisé pour les performances	Peut être moins performant sur mobile
Convient mieux à l'utilisation moderne	Convient aux projets plus anciens ou complexes

Bonnes Pratiques

1. **Concentrez-vous sur l'essentiel** : Ne surchargez pas la version mobile avec des éléments inutiles.
2. **Gardez une hiérarchie claire** : Le contenu le plus important doit être visible en premier.
3. **Testez souvent** : Assurez-vous que votre site s'affiche correctement sur des appareils réels.
4. **Pensez accessibilité** : Les boutons, liens, et champs doivent être utilisables facilement sur des écrans tactiles.

Conclusion

L'approche Mobile-First est devenue un standard incontournable dans la conception web moderne. En priorisant les appareils mobiles, vous créez des expériences utilisateur plus accessibles, plus performantes et mieux adaptées à la réalité des utilisateurs. Que vous soyez concepteur ou développeur, adopter cette approche est un investissement pour l'avenir de vos projets web.

Le Box Model

> *Dans cet article, nous allons plonger dans le monde fascinant du modèle de boîte en HTML. Le modèle de boîte est un concept qui joue un rôle essentiel dans la mise en page de toutes les pages web que tu vois en ligne. Alors, prépare-toi à plonger dans le monde des boîtes, des marges et des paddings !*

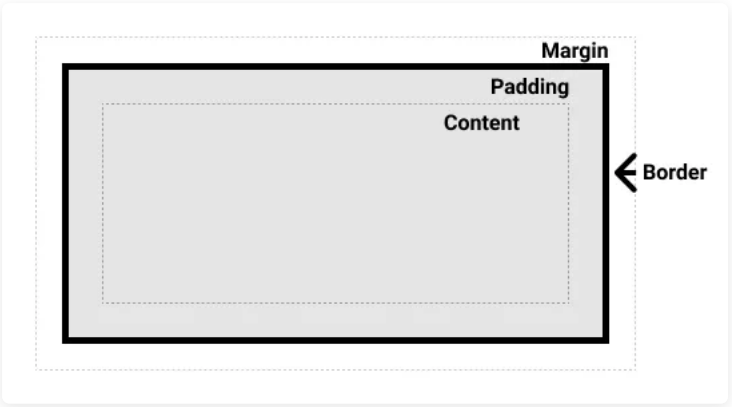
Qu'est-ce que le Modèle de Boîte ?

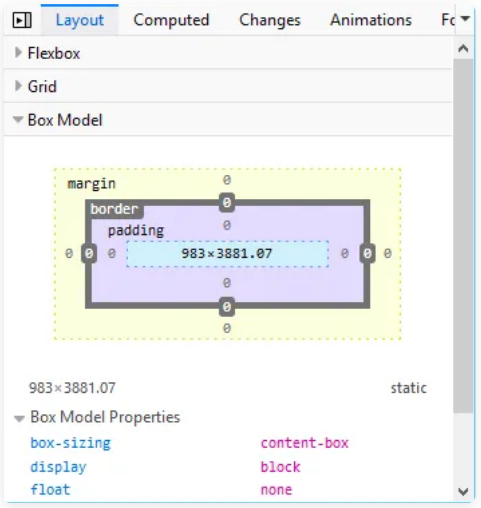
En CSS, tous les élémments sont entourés par une boite (box). Cette boite n'est pas directement visible pour les visiteurs de ta page web ni dans ton code HTML, mais elle a un impact important sur la façon dont les éléments sont affichés.

Comme il s'agit bien d'une question d'affichage (et pas de structure), tout ce qui concerne le box model est géré en CSS.

Commençons par découvrir c'est qu'est le modèle de boite, ensuite nous verrons les 2 types de boites principaux et comment les utiliser.

Le **modèle de boîte** (GB ***box model***) en HTML se réfère à la façon dont chaque élément HTML est représenté comme une boîte rectangulaire. Chaque boîte a quatre composantes principales :





1. **Le contenu (Content)** : C'est la partie intérieure de la boîte qui contient le texte, les images ou tout autre élément que tu places à l'intérieur de la balise HTML (la partie bleue, entourée par des pointillés).
2. **La bordure (Border)** : La bordure est une ligne qui entoure le contenu de la boîte. Elle peut être solide, pointillée, ou d'autres styles en fonction de ce que tu définis (en gris).
3. **Le padding (Padding)** : Le padding est l'espace entre la bordure et le contenu de la boîte. Cela permet de définir une séparation entre le contenu et la bordure. Elle agit comme une marge ***intérieure*** (en mauve)
4. **La marge (Margin)** : La marge entoure la boîte et sépare cette boîte des autres éléments de la page. Elle crée de l'espace autour de l'élément, à l'extérieur (en jaune).

Les Éléments "Block" et "Inline" en HTML et CSS

Les éléments de type "block" et "inline" sont deux concepts essentiels pour comprendre comment les éléments sont disposés et affichés sur une page web.

▸ Les Éléments "Block"

Les éléments "block" sont des éléments HTML qui **occupent toute la largeur disponible** et **commencent sur une nouvelle ligne**. Cela signifie que chaque élément "block" commence sur une nouvelle ligne et s'étend horizontalement pour occuper toute la largeur de son conteneur.

Exemples courants d'éléments "block" :

- `<div>`
- `<p>`
- `<h1>`, `<h2>`, `<h3>`, etc.
- `<ul>`, `<ol>`, `<li>`
- `<form>`
- `<header>`, `<footer>`
- Et bien d'autres !

Lorsque tu utilises des éléments "block", ils s'empilent verticalement les uns sur les autres, créant une structure de page bien définie.

On peut spécifier/changer la **hauteur** et la **largeur** d'un élément de type block.

Les éléments block acceptent les **marges** et **padding** en haut, en bas, à gauche et à droite,.

Les Éléments "Inline"

Les éléments "inline", en revanche, ne commencent pas sur une nouvelle ligne et **occupent seulement la largeur nécessaire pour afficher leur contenu**. Ils permettent donc de placer plusieurs éléments **côte à côte sur la même ligne**.

Exemples courants d'éléments "inline" :

- `<span>`
- `<a>`
- `<strong>`, `<em>`
- `<img>`
- `<button>`
- Et bien d'autres !

L'utilisation d'éléments "inline" est idéale pour intégrer du texte ou des éléments en ligne dans un paragraphe ou une phrase, sans interrompre la mise en page.

Il n'est pas possible de changer la hauteur/largeur d'un élément inline.

Les éléments inline n'appliquent **pas de marges et de padding verticaux**.

Comment les centrer horizontalement?

Pour centrer un élément `block`, il faut lui appliquer une largeur (par ex 80%) et des marges horizontales `auto` :

```
.container {
    width: 80%;
    margin: auto;
    background-color: #eee;
}
```

```
<div class="container">
    <span>Ce div est centré sur la page. Mais son contenu est centré à gauche.</span>
</div>
```

Les éléments `inline`, quant à eux, ne peuvent pas se centrer tout seuls puisqu'ils occupent tout juste la place dont ils ont besoin. Pour centrer un/des élément(s) `inline`, il faut demander à leur parent de les centrer en appliquant la propriété css `text-align` :

```
.container {

    text-align: center;

    background-color: #eee;
}
```

```
<div class="container">
    <span>Texte centré dans un div</span>
</div>
```

Bonnes Pratiques

Il est important de comprendre la différence entre les éléments "block" et "inline" pour une mise en page web cohérente. Voici quelques bonnes pratiques :

- Utilise les éléments "block" pour structurer les grandes sections de ta page, comme les en-têtes, les paragraphes et les listes.
- Utilise les éléments "inline" pour mettre en forme du texte ou pour incorporer de petits éléments dans le flux du texte.
- N'hésite pas à utiliser des balises HTML sémantiques pour une structure web claire et accessible.

Conclusion

En comprenant la différence entre les éléments "block" et "inline", tu seras en mesure de créer des mises en page web efficaces et bien structurées. Ces concepts de base sont essentiels pour toute personne qui souhaite devenir un développeur web compétent.

N'hésite pas à pratiquer en créant tes propres pages web et en utilisant à bon escient les éléments "block" et "inline". Plus tu pratiques, plus tu deviens fort dans l'art de la mise en page web.

Amuse-toi bien à explorer le monde du développement web, et à bientôt pour de nouvelles découvertes ! 😊

Explorer le Box Model

> *introduction*

Instructions

A l'aide de l'inspecteur de votre navigateur, identifiez les paramètres du box modèle appliqués aux élements de cette page. Repérez ceux pour lesquels on a spécifié un style, et identifiez ceux qui se sont vu appliquer un style par défaut par le navigateur.

NE CLIQUEZ PAS SUR CE LIEN

Questions

- As-tu remarqué que les couleurs ont une signification ?
- Quel est l'impact du div#content sur la mise en page ?
- Pourquoi le "align-center" du lien #danger ne fonctionne-t-il pas ?
- Pourquoi tu as cliqué sur le lien? 😉
- Quelle est la différence entre un `<div>` et un `<span>` ?
- Détaille le "box-model" appliqué par défaut au body

Questions d'interro

Série de 10 questions sur les éléments inline et block du box modèle HTML

1. **Qu'est-ce que le box modèle en HTML et CSS ?**
2. **Quelle est la différence entre un élément inline et un élément block en HTML ?**
3. **Donnez trois exemples d'éléments inline et trois exemples d'éléments block.**
4. **Comment les éléments inline et block se comportent-ils par rapport à la largeur et la hauteur ?**
5. **Quelle différence y a-t-il dans le box modèle entre un élément inline et block en termes de marges et de padding ?**
6. **Comment un élément inline et un élément block impactent-ils le flux de mise en page (flow) dans HTML ?**
7. **Peut-on appliquer des marges horizontales à un élément inline ? Pourquoi ou pourquoi pas ?**
    - Réponse attendue : Oui, les marges horizontales sont appliquées aux éléments inline, mais les marges verticales ne sont pas respectées.

Crée cette page:



Indices

- https://developer.mozilla.org/fr/docs/Learn/CSS/Building_blocks/The_box_model
- https://developer.mozilla.org/fr/docs/Tools/Page_Inspector/How_to/Examine_and_edit_the_box_model
- https://www.pierre-giraud.com/centrer-horizontalement-verticalement-element-html-css/

Exercice Bob l'éponge

> *Cet exercice te permet de mettre en application tout ce qu'on a appris en HTML et CSS jusqu'à présent et même de tester ta capacité à aller au-delà.*

Bien démarrer

1. Crée un nouveau dossier appelé « Bob » dans le dossier du cours « informatique / html / travaux » que tu as créé.
2. Dans ce dossier, crée un dossier « images » qui contiendra toutes les images de ton site.
3. Ensuite, télécharge les fichiers associés (index.html et les images). Pour ce faire fais un clic droit sur chacun des fichiers et sélectionne « enregistrer le lien sous… ». Enregistre les fichiers dans le dossier que tu viens de créer. Les images doivent aller dans le dossier « images ».
4. Renomme le fichier « index.html » en remplaçant PRENOM par ton prénom (ex : bob_Matis.html).

Le fichier HTML te permet de démarrer le projet rapidement.

Codes couleur

Voici les couleurs utilisées dans la page et leur code correspondant :

- Jaune : `#fef66b`
- Bleu foncé : `#005da4`
- Bleu clair : `#26baca` -> `#2dddf0`
- Rouge : `#ef5323`

Polices de caractères

Pour le titre principal, vous pouvez utiliser la police "**SpongeBob Font Wide**". Pour les titres de niveau 2, vous pouvez utiliser la police '**Krabby Patty**'.

Ces polices de caractères ne sont pas disponibles sur ton ordinateur. Pour pouvoir les utiliser dans ta page web, ajoute les 2 lignes suivantes directement après la balise `<style>` de ta page:

```
    @import url('http://fonts.cdnfonts.com/css/spongebob-font-condensed');
    @import url('http://fonts.cdnfonts.com/css/krabby-patty');
```

Le fichier HTML contient un contenu avec certains éléments html et css préremplis. C'est cadeau. !

Consignes

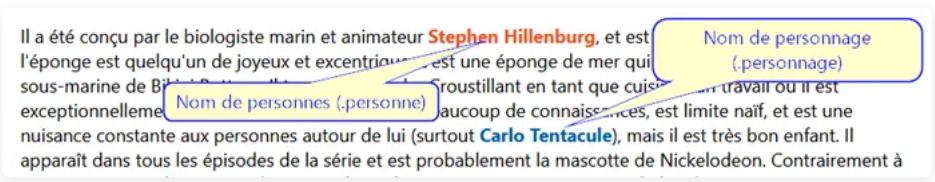


Un clic sur le logo doit ouvrir la page https://bob-leponge.fandom.com/fr/wiki/Accueil dans une nouvelle fenêtre/onglet.

Pour centrer cette image, applique-lui la class « centre ». Profites-en pour regarder comment centrer une image sur une ligne.

Le **nom des personnes** cités doit apparaître en rouge et doit être surligné en jaune quand la souris passe dessus.

Le **nom des personnages** cités doit apparaître en bleu foncé et doit être surligné en jaune quand la souris passe dessus.



Pour savoir comment surligner un texte quand la souris passe dessus... demande à Google.

La section "informations générales" peut être réalisée à l'aide d'un tableau:

Information générales	
Sexe	Mâle
Naissance	14 juillet 1986
Créature	Eponge de mer
Surnoms	L'idiot (par Carlo) Le meilleur cuisinier de Krabs Soufflé au fromage L'homme-oiseau de Bikini Bottom

Pour centrer les tables, applique-leur la déclaration CSS suivante :

```
margin : 0 auto ; /* Appliquer une marge automatique à gauche et à droite */
```

La section photos des personnages est réalisée à l'aide d'une table de 2 lignes / 5 colonnes :



Tu devrais avoir téléchargé les images dans le dossier « images », leur nom de fichier consiste en « [prénom du personne].png ».

Bonus : chaque image est un lien qui ouvre la page du personnage sur le site https://bob-leponge.fandom.com/fr/wiki/Accueil

Exercice - Les Briques

> *Cet exercice te permet de mettre en application les `div` tout en apprenant de nouvelles choses*

Bien démarrer

1. Crée un nouveau dossier appelé « `Briques` » dans le dossier du cours « informatique / html / travaux ».
2. Dans ce dossier, crée un dossier « images » qui contiendra toutes les images de ta page.
3. Ensuite, **télécharge les fichiers associés** (index.html et les images). Pour ce faire, fais un clic droit sur chacun des fichiers et sélectionne « enregistrer le lien sous… ». Enregistre les fichiers dans le dossier que tu viens de créer. Les images doivent aller dans le dossier « images ».
4. **Renomme le fichier** « index.html » en remplaçant PRENOM par ton prénom (ex : `briques-matis.html`).

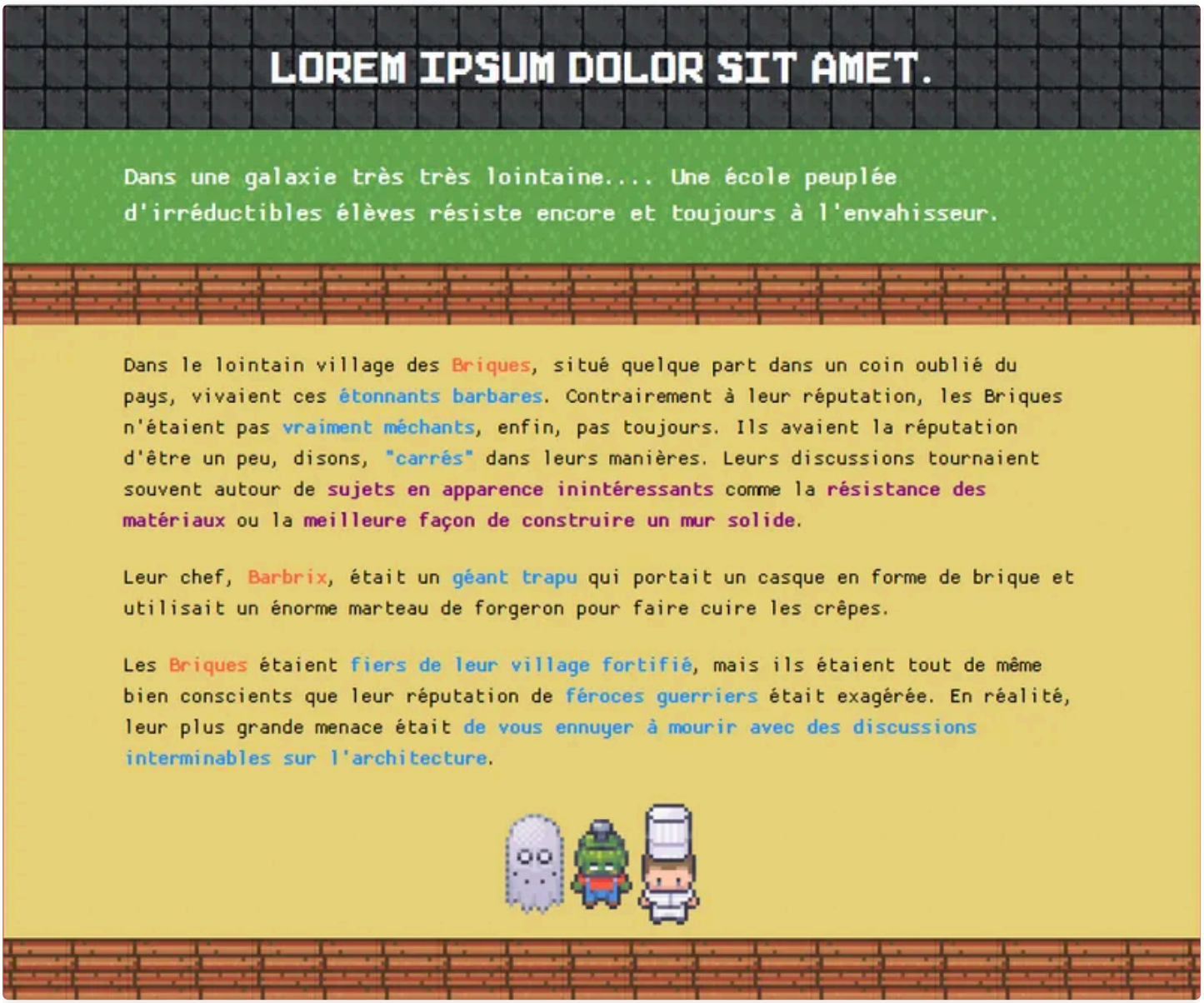
Le fichier HTML contient un contenu avec certains éléments html et css préremplis. C'est cadeau. !

Envoyer vos fichiers

Envoyez vos fichiers sur FileGator dans le dossier " `Html/Evaluations` "

Consignes

Essaie de créer une page qui ressemble au modèle suivant:



Les **noms** (personnes et peuples) cités doit apparaître en rouge.

Les **caractéristiques** citées doivent apparaître en bleu.

Les **sujets** doivent apparêtre en mauve.

Les 3 personnages doivent se trouver dans une table.

Pour centrer la table, applique la déclaration CSS suivante :

```
margin : 0 auto ; /* Appliquer une marge automatique à gauche et à droite */
```

Supprime les marges sur le `<body>` (= marges de 0px) afin d'éliminer les bordures blanches.

Codes couleur

Voici les couleurs utilisées dans la page et leur code correspondant :

- Caractéristiques: `dodgerblue`
- Noms de personnages: `tomato`
- Sujets : `purple`

Polices de caractères

Pour le titre principal, vous pouvez utiliser la police "**Pixelony**". Pour le texte du `contenu` , utilise la police '**ProggyCleanTT**'.

Pour info, ces polices de caractères ne sont pas disponibles sur ton ordinateur. Elles sont rendues disponibles sur ta page grâce à ces 2 lignes (lignes 6 et 7):

```
<link href="https://fonts.cdnfonts.com/css/pixelony" rel="stylesheet">
<link href="https://fonts.cdnfonts.com/css/proggy-clean" rel="stylesheet">
```

Réaliser les "sections"

Chaque section consiste en **2** `<div>` imbriqués (l'un à l'intérieur de l'autre):

- un `<div>` qui prend toute la largeur de la page et a une image de fond
- un `<div>` centré qui prend 80% de la largeur de la page et qui contient le texte

La couleur des sections est réalisée à l'aide d'une image de fond (pas simplement une couleur).

Appliquer une image de fond

Pour appliquer une image de fond à un élément, tu peux utiliser la **propriété CSS** suivante:

```
background: url("adresse_relative_de_l_image") repeat;
```

L'entête

La **hauteur** du div `#header` doit être **fixe**. Elle doit permettre d'afficher **exactement 3 rangées de pavés** (fichier: `pave.png`). Tu as déjà l'outil qui te permet de déterminer la taille de l'image.

L'introduction

Pour l'introduction, utilise la bonne image de fond.

Les "murs"

Ces `<div>` sont vides. Pour qu'ils apparaissent, tu dois **fixer leur hauteur en CSS**. La hauteur doit permettre d'afficher **exactement 1 rangée de briques**.

Textes à copier/coller

Dans une galaxie très très lointaine.... Une école peuplée d'irréductibles élèves résiste encore et toujours à l'envahisseur.

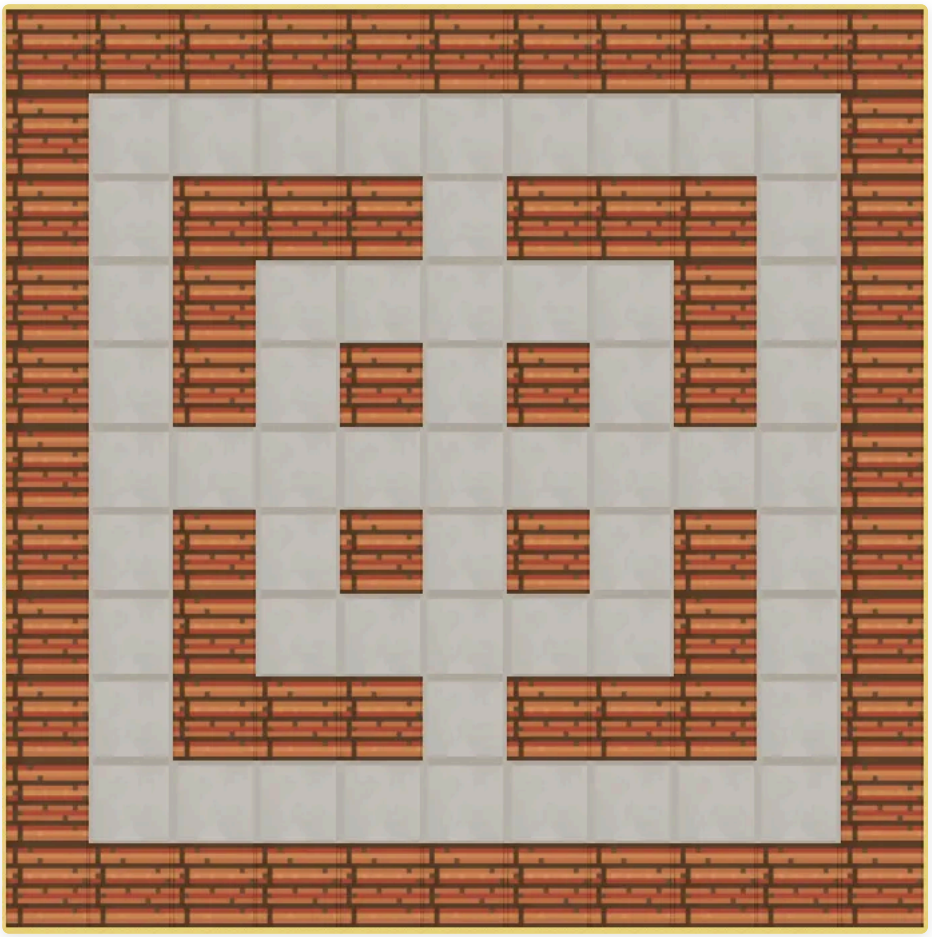
Dans le lointain village des Briques, situé quelque part dans un coin oublié du pays, vivaient ces étonnants barbares. Contrairement à leur réputation, les Briques n'étaient pas vraiment méchants, enfin, pas toujours. Ils avaient la réputation d'être un peu, disons, "carrés" dans leurs manières. Leurs discussions tournaient souvent autour de sujets en apparence inintéressants comme la résistance des matériaux ou la meilleure façon de construire un mur solide.

Leur chef, Barbrix, était un géant trapu qui portait un casque en forme de brique et utilisait un énorme marteau de forgeron pour faire cuire les crêpes.

Les Briques étaient fiers de leur village fortifié, mais ils étaient tout de même bien conscients que leur réputation de féroces guerriers était exagérée. En réalité, leur plus grande menace était de vous ennuyer à mourir avec des discussions interminables sur l'architecture.

Bonus - Mode Ninja

En bonus, essaie de réaliser le labyrinthe suivant:



Pour ce faire, utilise une table de 11 lignes et 11 colonnes.

Pour la générer rapidement, tape ce code dans VSCode: `table#labyrinthe>tr*11>td.mur*11` et appuie sur `tab` (un copier/coller ne fonctionne pas vraiment).

Ajoute les règles CSS suivantes à ta page:

```css
#labyrinthe {
    margin: 25px auto;
    border-collapse: collapse; /* Supprimer les bordures */
}

/* Cellules carrées */
#labyrinthe td {
    height: 48px;
    width: 48px;
}

.sol {
    /* Ajoute l'image de fond ici */
}
```

Comme image de fond, utilise le fichier `sol.png`.

Exercice - Tables

Cet exercice te permet de travailler les tables en HTML.

Fusionner des cellules avec COLSPAN et ROWSPAN

`colspan` permet d'étendre une cellule sur plusieurs colonnes.

`rowspan` permet d'étendre une cellule sur plusieurs lignes.

Exemples

Inspectez les tableaux ci-dessous. Les cellules avec un colspan et/ou rowspan sont encadrées en bleu.

NAME		AGE
Rasp	Poutine	48
Pair	Noail	1024

AVRIL	01	02	03	04
	05	06	07	08
	09	10	11	12

01	02	03	04	05
06	CONGÉS			07
08				09
10				11
12	13	14	15	16

Exercices

Reproduisez les tableaux ci-dessous:

A common header for three subheads			Header 3
Header 1	Header 2		
Thing A	data A1	data A2	data A3
Thing B	data B1	data B2	data B3
Thing C	data C1	data C2	data C3

Année 2018 - 2019

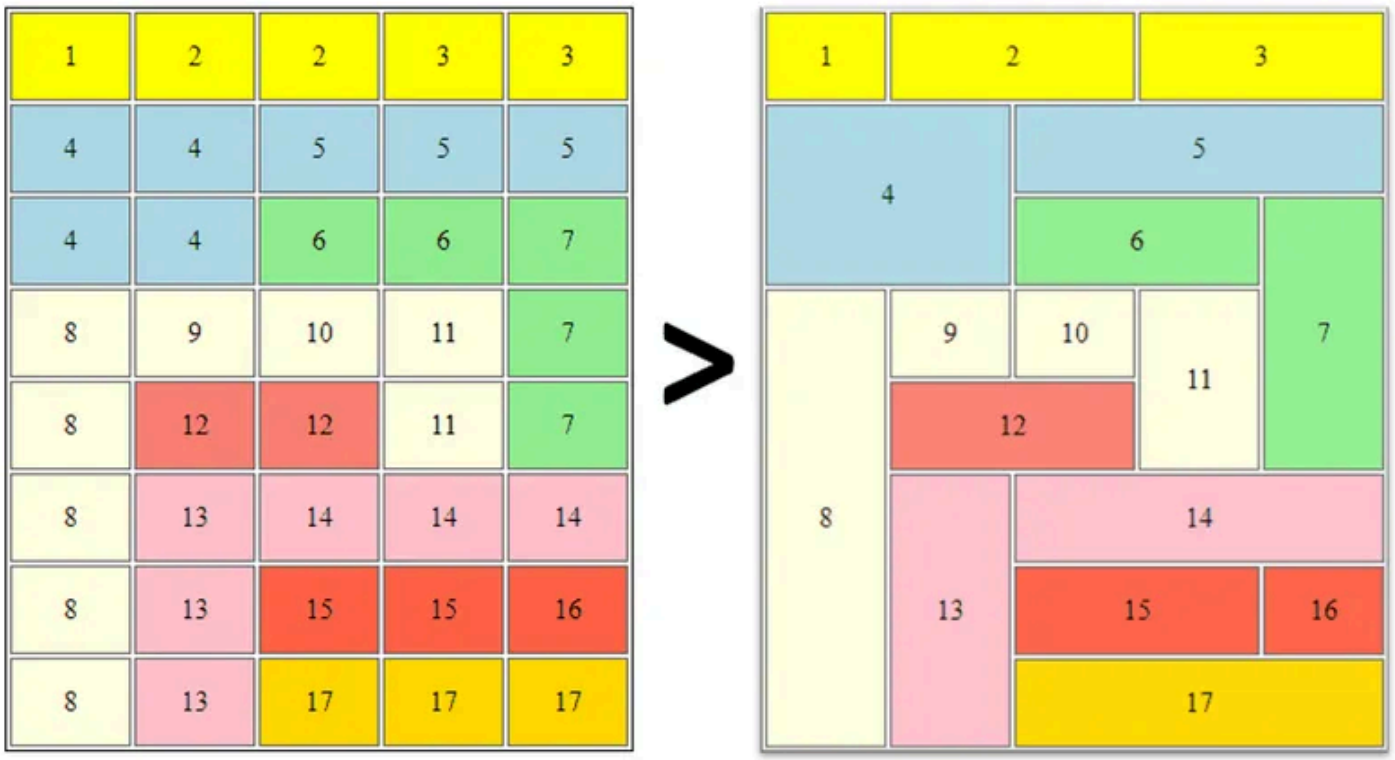
Activité	Cotisation	Inscrit	Prix
Patchwork	25 €	Oui	25 €
Peinture	33 €		0 €
Photo	45 €		0 €
Roller	58 €		0 €
Généalogie	15 €		0 €
Emaux	30 €		0 €
Frais d'adhésions			35 €
		Total	**60 €**

Une table test avec des cellules fusionnées

	Moyenne		Yeux rouges
	hauteur	poids	
Mâles	1.9	0.003	40%
Femelles	1.7	0.002	43%

Ninja Mode

Transforme la table ci-dessous comme indiqué à l'aide de `colspan` et de `rowspan`. Tu peux inspecter la table pour récupérer le code et son style.



1	2	2	3	3
4	4	5	5	5
4	4	6	6	7
8	9	10	11	7
8	12	12	11	7
8	13	14	14	14
8	13	15	15	16
8	13	17	17	17

Les formulaires HTML

> *Les formulaires en HTML constituent un élément essentiel dans la conception de sites web interactifs. Ils permettent aux utilisateurs de saisir et de soumettre des données, ce qui est crucial pour l'interaction avec le site. Dans cet article, nous allons explorer en détail les formulaires en HTML, ainsi que les différents types de champs disponibles pour la collecte de données.*

Introduction aux Formulaires HTML

Un formulaire HTML est une zone de votre page web qui contient des **champs** (ou éléments) permettant à l'utilisateur de saisir des informations.

Un champ (ou champ de formulaire) est un **élément interactif** d'un formulaire HTML qui permet à l'utilisateur de **saisir**, **sélectionner** ou **envoyer** des données.

Chaque champ correspond à une balise HTML spécifique comme `<input>`, `<select>`, `<textarea>`, etc. Il est souvent lié à un **nom (name)** qui servira d'identifiant pour récupérer la donnée côté serveur.

Lorsque l'utilisateur soumet le formulaire, les données saisies sont généralement envoyées à un serveur web pour traitement ultérieur. Les formulaires sont largement utilisés pour la collecte d'informations telles que les coordonnées, les commentaires, les préférences utilisateur, etc.



Structure de Base d'un Formulaire HTML

Voici la structure de base d'un formulaire HTML :

```html
<form action="/url-du-traitement" method="post">
    <!-- Champs du formulaire -->
</form>
```

- `action` : L'attribut `action` spécifie l'URL vers laquelle les données du formulaire seront envoyées pour traitement.
- `method` : L'attribut `method` définit la méthode HTTP utilisée pour envoyer les données du formulaire (plus d'informations dans un cours suivant).

Les Différents Types de Champs dans un Formulaire HTML

Champ de Texte






```
<input type="text" name="...">
```

Le champ de texte permet à l'utilisateur de saisir du texte sur une seule ligne.

Exemple :

```
<label for="nom">Nom :</label>
<input type="text" name="nom_complet">
```

Champ de Mot de Passe




```
<input type="password"  name="...">
```

Le champ de mot de passe masque les caractères saisis par l'utilisateur, utile pour la saisie de mots de passe sensibles.

Exemple :

```
<label for="motdepasse">Mot de passe :</label>
<input type="password" name="motdepasse">

<label for="motdepasse">Confirmez le mot de passe :</label>
<input type="password" name="confirmation">
```

Champ de Texte Multiligne




```
<textarea>
</textarea>
```

Le champ de texte multiligne permet à l'utilisateur de saisir du texte sur plusieurs lignes. Contrairement aux autres champs, la `textarea` possède une balise ouvrante et fermante.

Exemple :

```
<label for="commentaire">Votre commentaire :</label>
<textarea id="commentaire" name="commentaire" rows="4" cols="50"></textarea>
```

Case à Cocher




```
<input type="checkbox" name="...">
```

Une case à cocher (checkbox) permet à l'utilisateur de **cocher** ou **décocher** une **option binaire** : oui / non, vrai / faux, activé / désactivé, etc.

C'est un champ de type **booléen** : il peut être coché (**valeur envoyée**) ou non coché (aucune valeur envoyée).

Les cases à cocher permettent aussi à l'utilisateur de sélectionner **une ou plusieurs** options parmi une liste proposée.

Exemples :

```
<label>
  <input type="checkbox" name="conditions" value="ok"> J'accepte les conditions générales
</label>

Centres d'intérêt:
<label><input type="checkbox" name="cours[]" value="sport"></label>
<label><input type="checkbox" name="cours[]" value="musique"></label>
<label><input type="checkbox" name="cours[]" value="techno"></label>
```

Dans le cas où plusieurs valeurs peuvent être sélectionnées, il faut ajouter des crochets `[]` au champ name. Cela indique au serveur qu'il va recevoir un tableau de valeurs.

💡 Le fait d'entourer votre checkbox avec une balise `<label>` permet à l'utilisateur de cliquer sur le texte pour (dé)cocher votre case.

Bouton Radio




```
<input type="radio" name="...">
```

Les boutons radio permettent à l'utilisateur de sélectionner **une seule** option parmi plusieurs.

Exemple :

```
Genre :<br>
<label><input type="radio" name="genre" value="homme"> Homme</label>
<label><input type="radio" name="genre" value="femme"> Femme</label>
<label><input type="radio" name="genre" value="autre"> Autre</label>
```

Vous pouvez utiliser l'attribut `name` pour regrouper plusieurs options dans un même groupe. Dans l'exemple ci-dessus (genre), vous ne pouvez choisir qu'un seul genre parmi les propositions car les 3 `input` ont le même `name`.

Liste Déroulante




```
<select name="..." [multiple]>
    <option value="..."></option>
    <option value="..."></option>
    ...
</select>
```

La liste déroulante permet à l'utilisateur de sélectionner une ou plusieurs options dans un menu déroulant.

Exemple :

```
<label for="pays">Pays :</label>
<select id="pays" name="pays" multiple>
    <option value="be">Belgique</option>
    <option value="fr">France</option>
    <option value="ca">Canada</option>
</select>
```

La valeur qui se trouve entre les balises `<option>` sera affichée dans la liste déroulante. La valeur de l'attribut `value` sera envoyée au serveur.

L'attribut optionnel `multiple` permet d'indiquer que l'utilisateur peut sélectionner plusieurs éléments dans la liste. S'il n'est pas spécifié, l'utilisateur ne pourra sélectionner qu'un seul élément.

Conclusion

Les formulaires en HTML permettent d'interagir avec les utilisateurs et collecter des données sur les sites web. En utilisant les différents types de champs disponibles, vous pouvez créer des interfaces utilisateur riches et intuitives. N'hésitez pas à expérimenter avec les formulaires HTML pour répondre aux besoins spécifiques de votre site web.

Les formulaires - Partie 2

> *Les champs input HTML5 offrent une variété de fonctionnalités pour la collecte de données sur les sites web. Dans cet article, nous explorerons les différents types de champs input disponibles dans HTML5, ainsi que leurs utilisations et avantages.*

Introduction aux Champs Input HTML5

Les champs `input` sont des éléments fondamentaux des formulaires HTML, permettant aux utilisateurs de saisir et de soumettre des données. Avec l'introduction de HTML5, de nouveaux types de champs input ont été ajoutés pour améliorer l'expérience utilisateur et faciliter la validation des données côté client.

Les Principaux Types de Champs Input HTML5

1. Champ Texte (`type="text"`)

Le champ texte est utilisé pour saisir du texte sur une seule ligne. Il s'agit d'un champ polyvalent qui peut être utilisé pour la saisie de divers types de données, tels que les noms, les adresses e-mail, etc.

Exemple :

```
<input type="text" name="nom" placeholder="Entrez votre nom">
```

2. Champ Mot de Passe (`type="password"`)

Le champ mot de passe masque les caractères saisis par l'utilisateur, garantissant ainsi la confidentialité des informations sensibles telles que les mots de passe.

Exemple :

```
<input type="password" name="motdepasse" placeholder="Entrez votre mot de passe">
```

3. Champ Email (`type="email"`)

Le champ email est spécialement conçu pour la saisie d'adresses e-mail. Il valide automatiquement si l'adresse saisie correspond au format d'une adresse e-mail valide.

Exemple :

```
<input type="email" name="email" placeholder="Entrez votre adresse e-mail">
```

4. Champ Numérique (`type="number"`)

Le champ numérique restreint la saisie à des valeurs numériques et permet également de spécifier des contraintes telles que la valeur minimale, maximale et le pas d'incrémentation.

Exemple :

```
<input type="number" name="age" min="18" max="100" step="1" placeholder="Entrez votre âge">
```

5. Champ URL (`type="url"`)

Le champ URL est utilisé pour la saisie d'URLs (Uniform Resource Locators). Il valide automatiquement si l'URL saisie correspond au format d'une URL valide.

Exemple :

```
<input type="url" name="site_web" placeholder="Entrez l'URL de votre site web">
```

6. Champ Date (`type="date"` , `type="datetime"` , `type="datetime-local"` , `type="month"` , `type="week"`)

Les champs de date permettent aux utilisateurs de sélectionner des dates à partir d'un calendrier intégré. HTML5 offre plusieurs types de champs de date pour répondre à différents besoins, tels que la sélection de dates, de dates et heures, de mois, de semaines, etc.

Exemple :

```
<input type="date" name="date_naissance" placeholder="Entrez votre date de naissance">
```

7. Autres Types de Champs Input HTML5

En plus des types de champs input mentionnés ci-dessus, HTML5 propose également d'autres types de champs tels que `color` pour la sélection de couleurs, `range` pour les valeurs numériques dans une plage spécifiée, `search` pour les champs de recherche, et `tel` pour les numéros de téléphone.

Conclusion

Les champs input HTML5 offrent une gamme étendue de fonctionnalités pour la collecte de données sur les sites web modernes. En choisissant le type de champ approprié en fonction des données à collecter, vous pouvez améliorer l'expérience utilisateur et faciliter la validation des données côté client. N'hésitez pas à explorer les différents types de champs input HTML5 pour créer des formulaires interactifs et conviviaux.

Les formulaires - envoyer ses données

> *Les **formulaires HTML** permettent aux utilisateurs de saisir des données et de les envoyer à un serveur web. Ils sont utilisés pour des recherches, des inscriptions, des connexions, des commentaires, etc.*

La balise de base pour créer un formulaire est :

```
<form action="traitement.php" method="get">
  <input type="text" name="nom">
  <input type="submit" value="Envoyer">
</form>
```

Le formulaire contient :

- un **champ** (ici un champ texte nommé `nom`)
- un **bouton de soumission**
- un attribut `action` (URL de destination)
- un attribut `method` qui peut être **GET** ou **POST**

`GET` vs `POST` : quelle différence ?

	`GET`	`POST`
Transmission	Données dans l'URL	Données dans le corps de la requête
Affichage	Visible dans la barre d'adresse	Invisible
Taille max.	Limitée (~2000 caractères)	Illimitée (selon serveur)
Sécurité	Moins sécurisé (URL visible)	Plus adapté aux données sensibles
But principal	**Lire** des données	**Modifier/envoyer** des données
Effets attendus	Ne doit pas modifier les données sur le serveur	Peut entraîner une modification (ex. : base de données)
Utilisation typique	Recherche, navigation, filtres	Inscription, connexion, ajout, suppression

👉 Règle essentielle à retenir

- Une requête **GET** est censée être **sans effet secondaire** : elle sert à **obtenir des données** sans les modifier.
  Exemple : afficher un profil, rechercher un article, charger un produit.
- Une requête **POST** est utilisée pour **envoyer ou modifier** des données.
  Exemple : soumettre un formulaire d'inscription, ajouter un commentaire, modifier un mot de passe.

Cela respecte les **normes HTTP** (et les moteurs de recherche, les navigateurs, les caches web en tiennent compte).

Exemple avec `GET`

```
<form action="recherche.php" method="get">
  <input type="text" name="q" placeholder="Rechercher...">
  <input type="submit" value="Chercher">
</form>
```

Résultat dans l'URL après soumission :

```
recherche.php?q=php
```

Exemple avec `POST`

```
<form action="ajout.php" method="post">
  <input type="text" name="titre" placeholder="Titre de l'article">
  <textarea name="contenu"></textarea>
  <input type="submit" value="Publier">
</form>
```

Les données sont **envoyées discrètement au serveur** (pas visibles dans l'URL) et peuvent modifier une base de données. A noter que discrètement ne veut pas dire invisibles. Ces données sont toujours accessibles à l'aide des outils de développeurs des navigateurs.

En résumé

- Utilise `GET` pour lire ou interroger des données.

- Utilise `POST` pour transmettre des données sensibles ou **modifier** le contenu côté serveur.
- **Une requête GET ne doit jamais modifier une ressource.**

Les formulaires HTML (exercice complet)

> *Cet exercice te fait créer une page HTML5 contenant un formulaire riche, exploitant un maximum de types de champs HTML5 et leurs validations natives.*

Contexte

Tu travailles pour une organisation qui gère des événements (soirées, ateliers, tournois). On te demande de réaliser un formulaire d'inscription complet, accessible et bien structuré.

Objectifs

- Utiliser **tous les principaux types de champs** de formulaire HTML5.
- Appliquer les attributs de validation : `required`, `min`, `max`, `step`, `minlength`, `maxlength`, `placeholder`.
- Structurer un formulaire avec : `fieldset`, `legend`, `label`, `name`, `id`.
- Utiliser des éléments adaptés : `select`, `textarea`, `datalist`, `output`.

Consignes générales

- Fichier unique : `inscription.html`
- La page doit contenir :
  - un titre (`<h1>`)
  - une courte phrase de présentation
  - le formulaire
- Méthode du formulaire : `method="post"`
- Action : `action="#"` (on ne gère pas encore le serveur)
- Tous les champs doivent avoir un `label` lié au champ via `for="..."`.
- Tous les champs doivent avoir un `name` (important pour l'envoi).

Cahier des charges du formulaire

Bloc Identité

Inclure un `fieldset` avec `legend` "Identité".

Champs demandés :

- **Nom** : `type="text"`
  - `required`, `minlength="2"`, `maxlength="40"`
  - `autocomplete="family-name"`
- **Prénom** : `type="text"`
  - `required`, `minlength="2"`, `maxlength="40"`
  - `autocomplete="given-name"`
- **Pseudo (optionnel)** : `type="text"`
  - `maxlength="20"`
  - `placeholder="ex: QuizMaster42"`

Bloc Contact

`fieldset` "Contact".

- **Email** : `type="email"`
  - `required`
  - `autocomplete="email"`
- **Site web (optionnel)** : `type="url"`
  - `placeholder="https://..."`

Bloc Profil

`fieldset` "Profil".

- **Âge** : `type="number"`
  - `required` , `min="12"` , `max="99"`
- **Niveau en informatique** : `type="range"`
  - `min="0"` , `max="10"` , `step="1"`
  - Afficher la valeur choisie dans un `output` (au minimum via JS, ou sinon à défaut en texte).
- **Couleur préférée** : `type="color"`
  - valeur par défaut : `#3a86ff`

Bloc Disponibilités

`fieldset` “Disponibilités”.

- **Date de participation** : `type="date"`
  - `required`
- **Heure d’arrivée** : `type="time"`
  - `required`
- **Créneau exact (date + heure)** : `type="datetime-local"`
  - `required`

Bloc Choix de l’événement

`fieldset` “Événement”.

- **Type d’événement** : `select` (obligatoire) Options (au minimum) :
  - Soirée quiz
  - Atelier programmation
  - Tournoi gaming
  - Conférence Ajouter une option vide au début : “— Choisir —”
- **Ville** : `input` + `datalist` (obligatoire) Proposer au moins 6 villes belges (Bruxelles, Namur, Liège, Charleroi, Mons, Louvain-la-Neuve…).

Bloc Préférences

`fieldset` “Préférences”.

- **Newsletters** : `checkbox` (oui/non)
  - Valeur envoyée si cochée : `yes`
- **Régime alimentaire** : `radio` (1 choix obligatoire)
  - Omnivore
  - Végétarien
  - Vegan
  - Sans gluten

Bloc Message

`fieldset` “Message”.

- **Commentaires** : `textarea`
  - `rows="5"`
  - `maxlength="300"`
  - `placeholder="Infos utiles (besoins, matériel, contraintes, etc.)"`

Bloc Sécurité / confirmation

`fieldset` “Validation”.

- **Mot de passe** : `type="password"`
  - `required`
  - `minlength="8"`
- **Pièce jointe (optionnel)** : `type="file"`
  - accepter uniquement : `.pdf, .png, .jpg` via `accept`
- **J’accepte les conditions** : `type="checkbox"`
  - `required`

Boutons

En bas du formulaire :

- Bouton **Envoyer** : `type="submit"`
- Bouton **Réinitialiser** : `type="reset"`

Consignes supplémentaires

Le champ `Commentaires` doit avoir un `id` = `commentaires`.

Le label de la case à choser *'J'accepte les conditions'* doit avoir un `id` = `conditions`.

Ajouter des styles

Change la police de caractères du `body`.

Le formulaire doit avoir une largeur max de `1050px` et des marges automatiques (`auto`).

Change la couleur de fond de tous les `input` --> `lightyellow`.

Les légendes des fieldsets doivent s'afficher en gras.

La légende de la case à cacher "J'accepte les conditions" doit s'afficher en rouge.

Le cchamps "Commentaires" doit prendre toute la largeur disponible.

Exemple de résultat

Inscription à un événement

Complète ce formulaire pour t'inscrire à un événement. Les champs obligatoires doivent être remplis.

Identité

Nom

Prénom

Pseudo (optionnel)
ex: QuizMaster42

Contact

Email

Site web (optionnel)
https://...

Profil

Âge

Niveau en informatique (0 à 10)
5

Couleur préférée

Disponibilités

Date de participation
dd / mm / yyyy

Heure d'arrivée
-- : --

Créneau exact (date + heure)
dd / mm / yyyy -- : --

Événement

Type d'événement
— Choisir —

Ville

Préférences

Je veux recevoir les newsletters

Régime alimentaire (1 choix obligatoire) :

- Omnivore
- Végétarien
- Vegan
- Sans gluten

Message

Commentaires

Infos utiles (besoins, matériel, contraintes, etc.)

Validation

Mot de passe

J'accepte les conditions

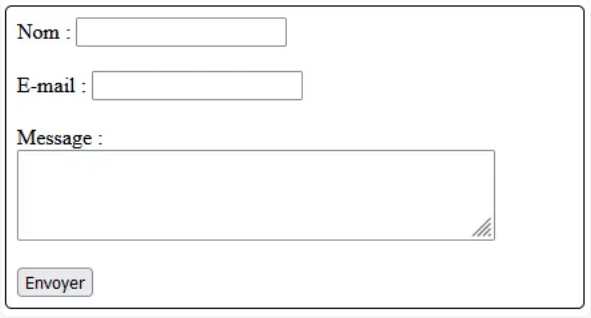
Envoyer Réinitialiser

Les formulaires HTML (exercices)

Voici des exercices de base afin d'apprendre à créer des formulaires simples en HTML

Exercice 1: Créer un formulaire de contact basique

Créez un formulaire de contact comprenant des champs pour le nom, l'e-mail et le message.



- **Action:** `https://ttrinfo.be/forms/contact.php`
- **Noms des champs:** `nom`, `email`, `message`

Exercice 2: Créer un formulaire de connexion

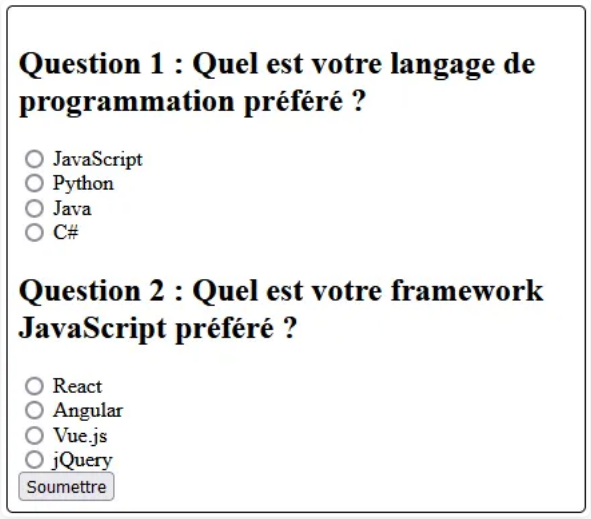
Créez un formulaire de connexion comprenant des champs pour le nom d'utilisateur et le mot de passe.



- **Action:** `https://ttrinfo.be/forms/connexion.php`
- **Noms des champs:** `utilisateur`, `mot_de_passe`

Exercice 3: Créer un formulaire de sondage à choix multiple

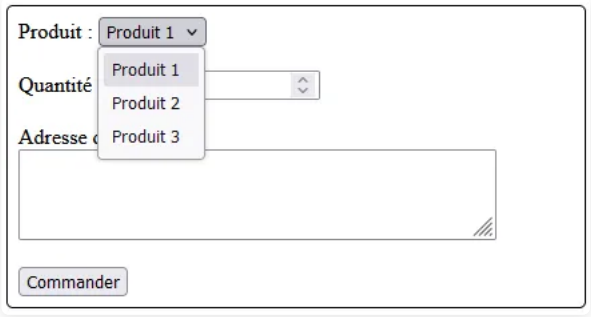
Créez un formulaire de sondage comprenant des questions à choix multiple.



- **Action:** `https://ttrinfo.be/forms/sondage.php`
- **Noms des champs:** `question1`, `question2`

Exercice 4: Créer un formulaire de commande

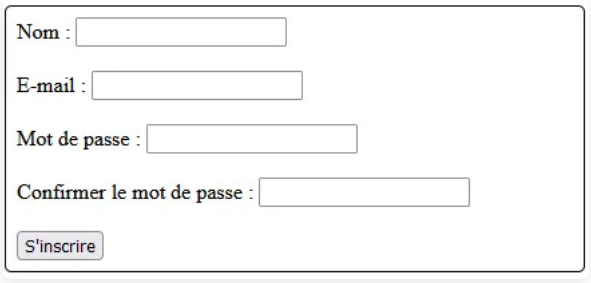
Créez un formulaire de commande pour un site de vente en ligne comprenant des champs pour le produit, la quantité et l'adresse de livraison.



- **Action:** `https://ttrinfo.be/forms/commande.php`
- **Noms des champs:** `produit`, `quantite`, `adresse`

Exercice 5: Créer un formulaire d'inscription

Créez un formulaire d'inscription comprenant des champs pour le nom, l'e-mail, le mot de passe et la confirmation du mot de passe.



- **Action:** `https://ttrinfo.be/forms/inscription.php`
- **Noms des champs:** `nom`, `email`, `mot_de_passe`, `mot_de_passe_confirmation`

Exercice 6: Créer un formulaire pour sélectionner une date

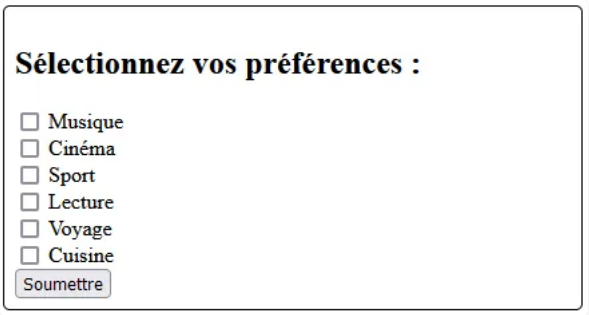
Créez un formulaire permettant à l'utilisateur de sélectionner une date.



- **Action:** `https://ttrinfo.be/forms/date.php`
- **Noms des champs:** `date`

Exercice 8: Créer un formulaire avec des cases à cocher

Créez un formulaire permettant à l'utilisateur de sélectionner ses préférences.



- **Action:** `https://ttrinfo.be/forms/cases.php`
- **Noms des champs:** `preference`

Javascript

introduction

JavaScript - Boucles for - Exercices

> *Les boucles sont des structures de contrôle essentielles en programmation, permettant d'exécuter un bloc de code plusieurs fois. En JavaScript, la boucle `for` est une des plus couramment utilisées. Cet article explore les boucles `for` en JavaScript moderne, avec des explications claires et des exemples concrets.*

Exercices sur les Boucles `for` en JavaScript

Voici 15 exercices de base sur les boucles `for` en JavaScript pour vous aider à vous familiariser avec ce concept essentiel.

Afficher des Nombres de 1 à 10

Utilisez une boucle `for` pour afficher les nombres de 1 à 10 dans la console.

Exemple :

```
for (let i = 1; i <= 10; i++) {
    console.log(i);
}
```

Afficher des Nombres Pairs de 2 à 20

Utilisez une boucle `for` pour afficher les nombres pairs de 2 à 20 dans la console.

Afficher les Éléments d'un Tableau

Déclarez un tableau `noms` contenant les valeurs ["Alice", "Bob", "Charlie"], puis utilisez une boucle `for` pour afficher chaque nom dans la console.

Somme des Nombres de 1 à 10

Utilisez une boucle `for` pour calculer et afficher la somme des nombres de 1 à 10.

Afficher une Table de Multiplication

Utilisez une boucle `for` pour afficher la table de multiplication de 5 (5, 10, 15, ..., 50).

Afficher des Nombres Impairs de 1 à 19

Utilisez une boucle `for` pour afficher les nombres impairs de 1 à 19.

Afficher les Nombres en Ordre Inverse

Utilisez une boucle `for` pour afficher les nombres de 10 à 1 dans la console.

Calculer la Factorielle d'un Nombre

Utilisez une boucle `for` pour calculer la factorielle de 5 et afficher le résultat.

La factorielle d'un nombre entier naturel, notée `n!`, est le produit de tous les entiers positifs inférieurs ou égaux à `n` (soit 1x2x3x4...x n). Par exemple, 4! = 1 x 2 x 3 x 4 = 24.

🎯 Bases de Javascript: Carnet d'objectifs

***Introduction au développement web interactif** · 4e–6e secondaire · Informatique · ~10h*

NOM	CLASSE	ANNÉE

Légende : *☐ = à valider · **Élève** = autoévaluation · **Pair** = évaluation par un(e) camarade · **Prof** = validation de l'enseignant(e)*

A. Environnement & premiers pas en JavaScript

Niveau 1 — Reproduire

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable d'ouvrir la console du navigateur (F12) et d'y exécuter du code JS.	☐	☐	☐	
Je suis capable d'intégrer du JavaScript dans une page HTML via la balise `<script>`.	☐	☐	☐	
Je suis capable d'utiliser `alert()`, `console.log()` et `document.write()` pour afficher un message.	☐	☐	☐	
Je suis capable de distinguer un script interne (inline) d'un script externe (fichier `.js`).	☐	☐	☐	

Niveau 2 — Comprendre

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je sais expliquer le rôle de JavaScript dans le trio HTML / CSS / JS.	☐	☐	☐	
Je sais choisir la méthode d'affichage adaptée selon le contexte (débogage vs utilisateur).	☐	☐	☐	

B. Variables, types et conventions

Niveau 1 — Reproduire

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable de déclarer une variable avec `let` ou `const`.	☐	☐	☐	
Je suis capable d'utiliser les types `string`, `number` et `boolean`.	☐	☐	☐	
Je suis capable d'utiliser les opérateurs arithmétiques et de concaténer des chaînes.	☐	☐	☐	
Je suis capable de recueillir une saisie utilisateur avec `prompt()` et de l'afficher.	☐	☐	☐	

Niveau 2 — Comprendre

C. Structures conditionnelles

Niveau 1 — Reproduire

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je sais expliquer la différence entre `let` et `const`, et choisir le bon mot-clé.	☐	☐	☐	
Je sais nommer mes variables en camelCase et justifier l'importance des conventions de nommage.	☐	☐	☐	
Je sais convertir une saisie texte en nombre avec `parseInt()` / `parseFloat()` si nécessaire.	☐	☐	☐	

Niveau 2 — Comprendre

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable d'écrire un bloc `if / else if / else` fonctionnel.	☐	☐	☐	
Je suis capable d'utiliser les opérateurs de comparaison (`==`, `!=`, `<`, `>`, `≤`, `≥`).	☐	☐	☐	

D. Boucles et fonctions

Niveau 1 — Reproduire

Niveau 2 — Comprendre

Niveau 3 — Généraliser / Combiner

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je sais combiner plusieurs conditions avec `&&` et `||`.	☐	☐	☐	
Je sais expliquer la différence entre `==` et `===` et utiliser `===` par défaut.	☐	☐	☐	
Je sais concevoir un algorithme de décision simple sous forme de condition imbriquée.	☐	☐	☐	

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable d'écrire une boucle `for` et une boucle `while`.	☐	☐	☐	
Je suis capable de déclarer une fonction, lui passer des paramètres et récupérer un `return`.	☐	☐	☐	

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable de concevoir une fonction qui intègre une boucle et des conditions pour résoudre un problème inédit.	☐	☐	☐	

E.

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je sais choisir entre `for` et `while` selon la situation.	☐	☐	☐	
Je sais décomposer un problème en fonctions réutilisables.	☐	☐	☐	
Je sais identifier et corriger une boucle infinie.	☐	☐	☐	

Manipulation du DOM & événements

Niveau 1 — Reproduire

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable de sélectionner un élément HTML avec `getElementById` et `querySelector`.	☐	☐	☐	
Je suis capable de modifier `innerHTML`, `textContent` et `style` d'un élément.	☐	☐	☐	
Je suis capable d'associer un événement (`click`, `keyup`) à une fonction via `addEventListener`.	☐	☐	☐	

Niveau 2 — Comprendre

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je sais expliquer ce qu'est le DOM et comment JS interagit avec la page HTML.	☐	☐	☐	
Je sais lire la valeur d'un champ `<input>` et l'utiliser dans mon script.	☐	☐	☐	

Niveau 3 — Généraliser / Combiner

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable de créer une interaction complète : saisie utilisateur → traitement JS → mise à jour de la page, sans modèle fourni.	☐	☐	☐	

F. Structures de données : tableaux, objets et JSON

Niveau 1 — Reproduire

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable de créer un tableau, y ajouter des éléments (`push`) et accéder à un index.	☐	☐	☐	
Je suis capable de créer un objet littéral avec des propriétés et d'y accéder (dot notation).	☐	☐	☐	
Je suis capable de convertir un objet en JSON (`JSON.stringify`) et l'inverse (`JSON.parse`).	☐	☐	☐	

Niveau 2 — Comprendre

Niveau 3 — Généraliser / Combiner

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je sais parcourir un tableau avec une boucle `for` ou `forEach` et afficher les résultats dans la page.	☐	☐	☐	
Je sais expliquer la différence entre un tableau et un objet, et choisir la structure adaptée.	☐	☐	☐	
Je sais expliquer l'utilité du format JSON dans les échanges entre applications web.	☐	☐	☐	

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable de stocker des données structurées (tableau d'objets) et de les afficher dynamiquement dans le DOM.	☐	☐	☐	

G. Projet intégrateur

Niveau 3 — Généraliser / Transférer

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je suis capable de concevoir et réaliser une page web interactive de A à Z (HTML + JS) sur un sujet de mon choix.	☐	☐	☐	
Je suis capable de présenter mon projet, d'expliquer mes choix techniques et de répondre aux questions.	☐	☐	☐	
Je suis capable de lire et comprendre le code d'un(e) camarade et d'y apporter une suggestion constructive.	☐	☐	☐	
Je sais identifier et corriger un bug en utilisant la console du navigateur de façon autonome.	☐	☐	☐	

H. Autonomie & transfert professionnel

Niveau 3 — Généraliser / Transférer

OBJECTIF	ÉLÈVE	PAIR	PROF	RESSOURCE
Je sais rechercher une solution technique de façon autonome (documentation MDN, StackOverflow).	☐	☐	☐	
Je sais structurer mon code de manière lisible : indentation, nommage cohérent, commentaires utiles.	☐	☐	☐	
Je sais situer JavaScript dans le paysage professionnel du développement web (front-end, frameworks, métiers).	☐	☐	☐	
Je suis capable d'adapter un script existant à un nouveau besoin sans partir de zéro.	☐	☐	☐	

JavaScript - La console développeur

> *Avant d'écrire des scripts, il faut savoir où **les exécuter** et où **lire leurs messages**. La console développeur, intégrée à tous les navigateurs, est l'outil n°1 du développeur web : essai rapide de code, affichage de variables, traque des erreurs.*

Ouvrir la console

NAVIGATEUR	RACCOURCI
Chrome / Edge	`F12` ou `Ctrl+Shift+I`
Firefox	`F12` ou `Ctrl+Shift+K`
Safari	`Cmd+Option+C` (activer d'abord le menu Développement dans les préférences)

Un panneau apparaît en bas (ou sur le côté) de la page. Plusieurs onglets : **Console**, **Éléments**, **Réseau**, **Sources**, etc. On reste sur **Console** pour commencer.

Exécuter du JavaScript à la volée

Le prompt de la console accepte du JavaScript. Tape, valide avec Entrée — c'est exécuté immédiatement dans le contexte de la page courante.

```
> 2 + 2
4

> let nom = "Alice"
undefined

> `Bonjour ${nom}`
"Bonjour Alice"

> document.title
"Mon super site"
```

C'est parfait pour :

- tester une expression rapidement,
- explorer un objet (`document` , `window` …),
- corriger une page en direct (modifier le texte, le style…) — les modifications sont perdues au rechargement, c'est juste pour expérimenter.

`console.log()` — afficher depuis ton script

C'est l'outil de débogage le plus utilisé au monde, tous langages confondus.

```
let age = 17;
let nom = "Alice";

console.log(nom);                       // "Alice"
console.log(age);                       // 17
console.log("Nom :", nom, "| Âge :", age);   // Nom : Alice | Âge : 17
console.log({ nom, age });              // { nom: "Alice", age: 17 }
```

Quelques astuces :

- Passe **plusieurs arguments** séparés par des virgules — la console les affiche les uns à la suite des autres.
- Pour visualiser un objet de façon lisible, passe-le directement : `console.log(monObjet)` affiche un arbre dépliable.

▸ `console.warn` , `console.error` , `console.table`

Variantes utiles selon le contexte :

```
console.warn("Attention, valeur étrange");   // jaune, avec icône avertissement
console.error("Quelque chose a planté");     // rouge, avec stack trace
console.table([
    { nom: "Alice", age: 16 },
    { nom: "Bob",   age: 17 }
]);                                          // tableau visuel
```

`console.table` sur un tableau d'objets est particulièrement spectaculaire — essaie une fois et tu l'utiliseras tout le temps.

Lire les erreurs

Quand ton script plante, le navigateur affiche une **erreur rouge** dans la console. Elle contient en général :

1. Le **message** d'erreur (`Uncaught ReferenceError: foo is not defined`).
2. Le **fichier** et la **ligne** où ça plante (cliquable !).
3. La **stack trace** (la chaîne d'appels qui a mené à l'erreur).

Quelques erreurs classiques :

MESSAGE	CAUSE TYPIQUE
`Uncaught ReferenceError: x is not defined`	Variable mal orthographiée ou non déclarée
`Uncaught TypeError: ... is null`	Tu utilises un élément avant qu'il existe (script avant `<body>`)
`Uncaught SyntaxError: Unexpected token`	Erreur de syntaxe : parenthèse, accolade, virgule manquante
`is not a function`	Tu appelles `()` sur quelque chose qui n'est pas une fonction (faute de frappe sur le nom)

Lire l'erreur, c'est 80% du débogage. Beaucoup de débutants paniquent à la première erreur rouge et la ferment. Au contraire : la console te dit **exactement** ce qui cloche et **où**. Lis le message, clique sur le lien de la ligne, et la moitié du problème est déjà résolu.

`alert` , `console.log` , `document.write` — lequel ?

Trois manières d'afficher quelque chose. Elles ne sont **pas équivalentes**.

MÉTHODE	POUR QUOI ?
`alert("msg")`	Message à l'**utilisateur** (boîte de dialogue bloquante). À utiliser parcimonieusement.
`console.log( ... )`	Message **pour toi**, le développeur. À utiliser sans modération.
`document.write( ... )`	Insère dans la page. **À éviter** : casse complètement la page si appelé après chargement.

Règle simple : `console.log` **pour déboguer,** `alert` **pour parler à l'utilisateur, jamais** `document.write` .

Inspecter un élément

Au-delà de la console, l'onglet **Éléments** (ou **Inspecteur** sur Firefox) montre l'arbre HTML de la page **en direct**.

- Clic droit → *Inspecter* sur n'importe quel élément de la page → l'inspecteur s'ouvre sur cet élément.
- Tu peux modifier les attributs, le texte, le CSS en temps réel (les changements sont éphémères).
- Tu peux suivre les modifications faites par ton JS : très utile pour vérifier que ton script change bien ce que tu crois.

À retenir

- `F12` ouvre la console.
- Le panneau **Console** exécute du JS dans le contexte de la page : un mini-laboratoire.
- `console.log( ... )` est ton meilleur ami pour comprendre ce qu'il se passe dans ton script. Sème-en partout pendant le débogage, retire-les avant la version finale.
- Les erreurs rouges ne sont pas un échec : elles te disent **où** et **quoi** corriger. Lis-les.
- `alert` pour l'utilisateur, `console.log` pour toi, jamais `document.write` .

JavaScript - Syntaxe et variables

> *JavaScript est un langage incontournable du Web moderne. Cette page présente l'essentiel de sa syntaxe afin de pouvoir écrire rapidement des scripts simples, lisibles et efficaces.*

Variables

En JavaScript, une variable permet de **stocker une valeur en mémoire** afin de la réutiliser ou l'afficher plus tard dans le programme.

Aujourd'hui, pour déclarer une variable, on utilise presque exclusivement `let` et `const`.

```
let age = 30;
let nom = "Jean";
const PI = 3.14;
```

- `let` : variable **modifiable**, portée limitée au bloc `{}`
- `const` : variable **non modifiable** (constante)
- `var` : ancienne syntaxe, **à éviter** (problèmes de portée et de lisibilité)

👉 **Bonne pratique** :

- utiliser `const` par défaut
- utiliser `let` uniquement si la valeur doit changer
- Tu verras `var` dans d'anciens tutoriels. Il fonctionne, mais il a des comportements imprévisibles. **On ne l'utilise plus.**

Types de données

JavaScript est un langage **à typage dynamique** : le type est déterminé automatiquement lors de l'affectation d'une valeur.

```
let nombre = 10;                // Number
let pi = 3.1415;               // Number
let prenom = "Alice";          // String
let estConnecte = true;        // Boolean
let fruits = ["pomme", "banane"]; // Array
let personne = { nom: "Jean", age: 30 }; // Object
```

Types courants : `Number`, `String`, `Boolean`

`string` — chaîne de caractères

Du texte, toujours entre guillemets (simples, doubles, ou backticks).

```
let ville = "Bruxelles";
let message = 'Bonjour tout le monde';
let phrase = `La ville est ${ville}`; // template literal
```

Les backticks (`` ` ``) permettent d'insérer des variables directement dans le texte avec `${}` — c'est très pratique.

```
let prenom = "Lucas";
let age = 16;
console.log(`Je m'appelle ${prenom} et j'ai ${age} ans.`);
// → Je m'appelle Lucas et j'ai 16 ans.
```

`number` — nombre

Entiers et décimaux, sans guillemets.

```
let temperature = -3;
let prix = 9.99;
let annee = 2025;
```

`boolean` — vrai ou faux

Exactement deux valeurs possibles : `true` ou `false`.

```
let estConnecte = true;
let aPayé = false;
```

Les booléens sont au cœur de toutes les conditions et décisions du programme.

▸ Conventions de nommage

Le code qu'on écrit est lu par d'autres humains (et par son futur soi). Un bon nommage, ça évite des heures de confusion.

CAMELCASE — LA CONVENTION JS

En JavaScript, les variables et fonctions s'écrivent en **camelCase** : la première lettre est minuscule, chaque mot suivant commence par une majuscule.

```
// ✅ Correct
let nombreDEleves = 28;
let prixUnitaire = 4.5;
let nomUtilisateur = "Amir";

// ❌ À éviter
let nombre_d_eleves = 28;  // snake_case (Python, pas JS)
let PrixUnitaire = 4.5;    // PascalCase (réservé aux classes)
let p = 4.5;               // trop court, sans sens
```

AUTRES RÈGLES

- **Nom descriptif** : `scoreJoueur` plutôt que `sj` ou `x`
- **Commence par une lettre** : pas de chiffre en début (`3prix` → ❌)
- **Pas d'accents ni d'espaces** : `nombreEtudiants` et non `nombreÉtudiants`
- **Respecte la casse** : `age` , `Age` et `AGE` sont trois variables différentes

```
// Exemple concret bien nommé
const nombreDeVies = 3;
let scoreActuel = 0;
let nomDuJoueur = prompt("Ton pseudo ?");

console.log(`${nomDuJoueur} commence avec ${nombreDeVies} vies.`);
```

> *Un bon nom de variable, c'est un commentaire gratuit. Si tu dois ajouter un commentaire pour expliquer ce que contient une variable, c'est souvent signe que son nom n'est pas assez clair.*

▸ À retenir

- `const` par défaut, `let` si la valeur change. Jamais `var` .
- Les trois types de base : `string` (texte), `number` (nombre), `boolean` (vrai/faux).
- `prompt()` renvoie toujours un `string` → pense à `parseInt()` ou `parseFloat()` si tu fais des calculs.
- Les backticks + `${}` rendent les messages beaucoup plus lisibles que la concaténation avec `+` .
- **camelCase** pour tous les noms de variables. Toujours descriptif, jamais `x` ou `temp` .

Opérateurs arithmétiques

Les opérateurs permettent de **manipuler les valeurs numériques**.

Exemples:

```
let a = 10;
let b = 3;

console.log(a + b);  // → 13  (addition)
console.log(a - b);  // → 7   (soustraction)
console.log(a * b);  // → 30  (multiplication)
console.log(a / b);  // → 3.333...  (division)
console.log(a % b);  // → 1   (modulo : reste de la division)
console.log(a ** b); // → 1000  (puissance)
```

Le **modulo** (`%`) est très utile en pratique : il permet de savoir si un nombre est pair, de créer des cycles, etc.

```
console.log(8 % 2);  // → 0  (pair)
console.log(7 % 2);  // → 1  (impair)
```

OPÉRATEUR	SYMBOLE	DESCRIPTION	EXEMPLE	RÉSULTAT
Addition	`+`	Additionne deux valeurs	`5 + 3`	`8`
Soustraction	`-`	Soustrait une valeur	`10 - 4`	`6`
Multiplication	`*`	Multiplie deux valeurs	`6 * 2`	`12`
Exposant	`**`	Exposant	`2 ** 2`	`4`
Division	`/`	Division classique	`15 / 3`	`5`
Modulo	`%`	Reste de la division entière	`10 % 3`	`1`
Incrémentation	`++`	Ajoute 1	`x++`	
Décrémentation	`--`	Retire 1	`x--`	

PEMDAS

JavaScript respecte strictement la règle PEMDAS, comme la plupart des langages de programmation.

PEMDAS définit l'ordre de priorité des opérations arithmétiques :

- Parentheses → ()
- Exposants → **
- Multiplication → *
- Division → /
- Addition → +
- Subtraction → -

Si 2 opérations ont la même priorité, elles sont exécutées de gauche à droite.

```
let resultat = 20 / 5 * 2;
console.log(resultat);
```

Résultat: `8`. Division et multiplication ont la même priorité, l'évaluation se fait de gauche à droite.

Division entière

Contrairement à Python, JavaScript ne possède pas d'opérateur de division entière. On utilise donc `Math.floor()`.

```
let resultat = Math.floor(10 / 3); // 3
```

Points-virgules

Les points-virgules `;` marquent la fin d'une instruction.

JavaScript peut parfois s'en passer, mais **il est fortement recommandé de les utiliser**.

Toujours utiliser les points-virgules pour éviter les erreurs difficiles à détecter.

```
let x = 5;
let y = 10;
```

Conclusion

Cette page couvre les **fondations indispensables** de JavaScript : variables, types, opérateurs et structure du code. Ces bases sont importantes avant d'aborder les conditions, les boucles, les fonctions et la programmation Web dynamique.

JavaScript - Exercices de base

> *Voici **10 exercices courts, ciblés et cohérents pour une introduction**, exclusivement centrés sur **la déclaration de variables** et **les calculs simples**. Ils sont progressifs, sans fonctions, sans conditions, sans boucles, et parfaitement adaptés à un **premier contact avec JavaScript**.*

Pour afficher une variable, tu peux utiliser la fonction `alert()` ou `console.log()` .

```
let resultat = 5 + 2 * 10;
alert(resultat);
console.log(resultat);
```

Exercice 1: Déclaration simple

Déclare une variable `age` et affecte-lui la valeur `16` . Affiche ensuite la valeur dans une alerte.

Exercice 2: Deux variables

Déclare deux variables `nombre1` et `nombre2` contenant des nombres (avec des valeurs de ton choix). Affiche leur somme.

Ensuite, affiche dans la console leur somme, leur différence, leur produit et leur quotient, avec un libellé clair pour chaque résultat.

Exercice 3: Calcul intermédiaire

Déclare une variable `prix` valant `12.5` et une variable `quantite` valant `4` . Calcule le prix total et stocke-le dans une variable `total` .

Exercice 3bis: Calcul intermédiaire Pareil mais en utilisant `prompt()` 😉

Exercice 4: Modification de variable

Déclare une variable `score` avec la valeur `10` . Augmente-la de `5` et affiche le résultat.

Exercice 5: Incrémentation

Déclare une variable `compteur` valant `0` . Incrémente-la deux fois et affiche sa valeur finale.

Exercice 6: Moyenne simple

Déclare trois variables `note1` , `note2` , `note3` . Calcule la moyenne et stocke-la dans une variable `moyenne` .

Exercice 7: Modulo

Déclare une variable `nombre` contenant un entier. Calcule le reste de la division par `2` et stocke le résultat dans une variable `reste` .

Exercice 8: Aire d'un rectangle

Déclare deux variables `largeur` et `hauteur` . Calcule l'aire du rectangle et stocke-la dans une variable `aire` .

Exercice 9: Conversion simple

Déclare une variable `celsius` contenant une température. Calcule la température correspondante en Fahrenheit (`F = C × 1.8 + 32`).

Exercice 10: Priorité des opérations

Déclare trois variables `a` , `b` et `c` . Calcule le résultat de l'expression suivante et stocke-le dans une variable `resultat` :

```
a + b * c
```

Exercice 11: Présentation automatique Demande à l'utilisateur son prénom et son année de naissance. Calcule son âge approximatif et affiche une phrase complète dans la console.

```
// Exemple de sortie attendue :
// "Bonjour Léa ! Tu as environ 17 ans."
```

JavaScript - La saisie utilisateur

> *Pour qu'un programme soit interactif, il doit pouvoir **lire ce que tape l'utilisateur**. `prompt()` est le moyen le plus simple — mais attention au piège classique : ce qu'il renvoie n'est jamais un nombre, même quand on tape un chiffre.*

`prompt()` — demander une valeur

`prompt()` ouvre une petite boîte de dialogue avec un champ de saisie. Quand l'utilisateur valide, la fonction renvoie ce qu'il a tapé.

```
let nom = prompt("Quel est ton prénom ?");
console.log(`Bonjour ${nom}`);
```

On peut aussi proposer une **valeur par défaut** en deuxième argument :

```
let pseudo = prompt("Ton pseudo ?", "Anonyme");
```

Si l'utilisateur clique sur **Annuler**, `prompt()` renvoie `null`. On peut tester :

```
let nom = prompt("Ton prénom ?");

if (nom === null) {
    console.log("Saisie annulée");
} else if (nom === "") {
    console.log("Saisie vide");
} else {
    console.log(`Bonjour ${nom}`);
}
```

Le piège : prompt renvoie toujours un texte

C'est **le bug** qui surprend tout le monde au début.

```
let nombre = prompt("Donne un nombre :");
// L'utilisateur tape : 5

console.log(nombre + 3);
// Affiche : "53"  (et non 8)
```

Pourquoi ? Parce que `nombre` contient **le texte** `"5"`, pas le nombre 5. Et avec l'opérateur `+`, JavaScript préfère **concaténer** quand au moins un côté est du texte : `"5" + 3` donne `"53"`.

`prompt()` **renvoie *toujours* un string**, même quand l'utilisateur tape un chiffre. Si tu veux calculer avec, tu **dois** convertir.

Convertir en nombre

Trois outils, à choisir selon le contexte.

▸ `parseInt()` — nombre entier

Lit le texte caractère par caractère jusqu'au premier qui n'est pas un chiffre. Idéal pour les âges, scores, quantités…

```
let age = parseInt(prompt("Ton âge ?"));
// "16"   → 16
// "16.7" → 16     (la partie décimale est ignorée)
// "16ab" → 16     (s'arrête au premier non-chiffre)
// "abc"  → NaN    (Not a Number)
```

▸ `parseFloat()` — nombre décimal

Pareil mais accepte le point décimal.

```
let prix = parseFloat(prompt("Prix en euros ?"));
// "9.99"  → 9.99
// "9,99"  → 9        (la virgule arrête la lecture — attention !)
// "abc"   → NaN
```

Virgule ou point ? JavaScript utilise le **point** comme séparateur décimal, comme dans la plupart des langages de programmation. Tes utilisateurs francophones taperont souvent une virgule. Tu peux convertir : `texte.replace(",", ".")` avant le `parseFloat`.

▸ `Number()` — strict

Plus exigeant : il faut que **tout le texte** soit un nombre, sinon `NaN`.

```
Number("16")      // 16
Number("16.7")    // 16.7
Number("16ab")    // NaN   (≠ parseInt, qui aurait renvoyé 16)
Number("")        // 0     (cas un peu surprenant)
```

Pour la plupart des cas pédagogiques, `parseInt` et `parseFloat` suffisent.

NaN — le résultat des conversions ratées

Quand on essaie de convertir un texte qui n'a rien de numérique, on obtient `NaN` (*Not a Number*). C'est techniquement un nombre, mais qui propage l'erreur dans tous les calculs.

```
let age = parseInt("salut");
console.log(age);          // NaN
console.log(age + 10);     // NaN
console.log(age * 2);      // NaN
```

On teste avec `isNaN()` :

```
let age = parseInt(prompt("Ton âge ?"));

if (isNaN(age)) {
    alert("Ce n'est pas un nombre valide.");
} else {
    alert(`Tu as ${age} ans.`);
}
```

Bonne pratique : **toujours vérifier** que la conversion a marché avant d'utiliser le résultat.

Pattern complet : saisie sécurisée

Le pattern classique combine prompt + conversion + validation :

```
let age = parseInt(prompt("Ton âge ?"));

if (isNaN(age) || age < 0 || age > 120) {
    alert("Âge invalide.");
} else {
    alert(`Tu as ${age} ans.`);
}
```

Si tu veux **boucler tant que la saisie n'est pas valide**, c'est typiquement le rôle d'un `while` :

```
let age;

do {
    age = parseInt(prompt("Ton âge ?"));
} while (isNaN(age) || age < 0 || age > 120);

alert(`Tu as ${age} ans.`);
```

Autres méthodes : `alert` et `confirm`

`prompt` a deux cousines plus simples.

`alert()` — afficher un message

```
alert("Bienvenue !");
```

Ne renvoie rien d'utile. Sert juste à afficher.

`confirm()` — demander oui/non

```
let veutContinuer = confirm("Veux-tu continuer ?");

if (veutContinuer) {
    console.log("OK, on continue");
} else {
    console.log("Abandon");
}
```

Renvoie `true` si OK, `false` si Annuler. Pratique pour les questions binaires.

Limites de `prompt`

`prompt()` est parfait pour apprendre, mais en pratique, on l'utilise rarement sur un vrai site :

- **Apparence figée** — impossible à styler avec du CSS.
- **Bloquant** — la page se gèle tant que l'utilisateur n'a pas répondu.
- **Mauvais pour mobile** — l'expérience est dégradée.

Sur un vrai site, on utilise des **champs** `<input>` dans un formulaire HTML. On lit leur valeur avec `document.getElementById("monChamp").value`. Mais ça, c'est l'étape suivante.

À retenir

- `prompt("question")` ouvre une boîte de saisie et renvoie ce que tape l'utilisateur.
- Le résultat est **toujours un string**, même `"42"` n'est pas le nombre 42.
- `parseInt(texte)` pour un entier, `parseFloat(texte)` pour un décimal.
- `isNaN(valeur)` vérifie si la conversion a échoué — toujours tester avant d'utiliser.
- `alert("message")` pour afficher, `confirm("question")` pour oui/non.
- `prompt` est un outil d'apprentissage. Sur un vrai site : des champs `<input>` HTML.

JavaScript - Les conditions

> *Un programme ne se contente pas d'exécuter des instructions dans l'ordre. Il doit aussi pouvoir **prendre des décisions** : l'utilisateur est-il majeur ? le score dépasse-t-il 100 ? le mot de passe est-il correct ? En JavaScript, ces décisions s'écrivent avec `if`, `else` et `else if`.*

Une condition, c'est une question

Une condition est une **question à laquelle on peut répondre par vrai ou faux** :

- L'âge est-il supérieur à 18 ?
- Le mot de passe est-il correct ?
- Le score dépasse-t-il 100 ?

Selon la réponse, le programme suit un chemin ou un autre.

`if` — exécuter du code seulement si...

La forme la plus simple : un seul bloc, exécuté **uniquement si** la condition est vraie.

```
if (condition) {
    // instructions exécutées si la condition est vraie
}
```

Exemple :

```
let age = 20;

if (age >= 18) {
    console.log("Vous êtes majeur");
}
```

Si la condition est fausse, **rien ne se passe** — le programme continue sans exécuter le bloc.

`if / else` — deux chemins, un seul emprunté

Pour exécuter **un bloc si vrai, un autre si faux** :

```
let age = 16;

if (age >= 18) {
    console.log("Accès autorisé");
} else {
    console.log("Accès refusé");
}
```

Une seule des deux branches est exécutée. Jamais les deux, jamais aucune.

```
       condition ?
       /        \
     vrai       faux
      |          |
   bloc A     bloc B
```

`if / else if / else` — plusieurs cas

Quand il y a **plus de deux possibilités**, on enchaîne des `else if` .

```
let note = 72;

if (note >= 90) {
    console.log("Excellent");
} else if (note >= 75) {
    console.log("Très bien");
} else if (note >= 50) {
    console.log("Réussi");
} else {
    console.log("Échec");
}
```

Les conditions sont testées **dans l'ordre**. Dès qu'une est vraie, son bloc s'exécute et **les suivantes ne sont plus évaluées**.

Le `else` final attrape tout ce qui n'a été capturé par aucune condition précédente. Il est optionnel.

Les opérateurs de comparaison

OPÉRATEUR	SIGNIFICATION
`>`	supérieur à
`<`	inférieur à
`≥`	supérieur ou égal
`≤`	inférieur ou égal
`===`	égal
`≠`	différent

Utilise `===` (et `≠`), pas `==` (et `!=`).

`==` compare en convertissant les types : `"5" == 5` est `true` (bizarre). `===` compare sans conversion : `"5" === 5` est `false` (logique). Prends l'habitude de `===` dès maintenant, tu t'épargneras des heures de bugs.

L'ordre des conditions compte

Quand on teste des **seuils** (notes, scores, âges…), l'ordre n'est pas neutre. JavaScript s'arrête à la première condition vraie.

```
// ❌ Ordre incorrect
let note = 80;

if (note >= 50) {
    console.log("Réussi");        // ← s'affiche ici
} else if (note >= 75) {
    console.log("Très bien");     // jamais atteint
}
```

Note = 80 est ≥ 50, donc on affiche "Réussi" et c'est fini. "Très bien" n'est jamais testé.

```
// ✅ Du plus exigeant au plus permissif
if (note >= 75) {
    console.log("Très bien");
} else if (note >= 50) {
    console.log("Réussi");
}
```

Règle pratique :

- comparaisons `≥` ou `>` → tester du **plus grand au plus petit**
- comparaisons `≤` ou `<` → tester du **plus petit au plus grand**

Exemple complet : saisie utilisateur

```
let age = prompt("Quel est votre âge ?");
age = parseInt(age);   // prompt() renvoie un string, on le convertit en nombre

if (age >= 18) {
    alert("Bienvenue !");
} else if (age >= 12) {
    alert("Accès restreint.");
} else {
    alert("Trop jeune.");
}
```

`prompt()` renvoie **toujours un texte**, même quand on tape un nombre. `parseInt()` est obligatoire avant toute comparaison numérique.

Erreurs fréquentes

Confondre `=` et `===`

```
if (age = 18) { ... }    // ❌ assigne 18 à age, condition toujours "vraie"
if (age === 18) { ... }  // ✅ compare age à 18
```

C'est la cause de bug la plus classique chez les débutants. `===` toujours, sauf si tu sais ce que tu fais.

Oublier les accolades

```
// ❌ Trompeur
if (age >= 18)
    console.log("Bienvenue");
    console.log("Tu peux entrer");   // ← TOUJOURS exécuté, indépendamment du if
```

Sans accolades, **seule la première instruction** dépend du `if`. L'indentation ne change rien — c'est un piège visuel. Mets toujours les `{ }`.

Oublier les parenthèses

```
if age >= 18 { ... }    // ❌ erreur de syntaxe
if (age >= 18) { ... }  // ✅
```

À retenir

- `if (condition) { ... }` — exécute si vrai, ignore sinon.
- `if / else` — un chemin sur deux.
- `if / else if / else` — plusieurs cas, **dans l'ordre**, seule la première condition vraie est exécutée.
- `===` toujours, jamais `==`.
- Toujours les `{ }`, même pour une seule instruction.
- Pour les seuils : ordre **décroissant** avec `≥`, **croissant** avec `≤`.

Pour combiner plusieurs conditions dans un seul test (ET, OU), voir l'article suivant.

JavaScript - Combiner plusieurs conditions

> *Une seule condition ne suffit pas toujours : un utilisateur doit être majeur **et** avoir un billet, un joueur gagne avec un gros score **ou** un bonus. Les opérateurs logiques `&&`, `||` et `!` permettent de combiner plusieurs tests dans une seule condition.*

Les trois opérateurs logiques

OPÉRATEUR	LECTURE	VRAI QUAND…
`&&`	**ET**	les **deux** conditions sont vraies
`||`	**OU**	**au moins une** condition est vraie
`!`	**NON**	la condition est **fausse** (négation)

`&&` — ET (les deux à la fois)

```
let age = 20;
let aBillet = true;

if (age >= 18 && aBillet) {
    console.log("Entrée autorisée");
}
```

Lecture : *"si l'âge est ≥ 18 **et** que la personne a un billet, alors..."*

Si **une seule** des deux conditions est fausse, le tout est faux.

```
   condition 1   ET   condition 2
       vrai            vrai
         \              /
          résultat : vrai
```

`||` — OU (au moins une)

```
let role = "admin";
let age = 17;

if (role === "admin" || age >= 18) {
    console.log("Accès autorisé");
}
```

Lecture : *"si l'utilisateur est admin **ou** qu'il est majeur..."*

Il suffit qu'**une seule** condition soit vraie. Les deux peuvent l'être aussi, ça reste vrai.

`!` — NON (négation)

`!` inverse une valeur booléenne : `!true` devient `false`, et inversement.

```
let estConnecte = false;

if (!estConnecte) {
    console.log("Veuillez vous connecter");
}
```

Lecture : *"si **non** connecté..."*

`!` se lit souvent comme *"pas"* ou *"non"* devant la variable.

Combiner plusieurs opérateurs

On peut chaîner autant d'opérateurs que nécessaire. Les **parenthèses** rendent la logique explicite :

```
let age = 20;
let aBillet = true;
let estVIP = false;

if ((age >= 18 && aBillet) || estVIP) {
    console.log("Entrée autorisée");
}
```

Lecture : *"un VIP entre toujours ; sinon il faut être majeur **et** avoir un billet."*

Conseil : dès qu'on combine `&&` et `||`, **mets des parenthèses**. Sans elles, `&&` a priorité sur `||` (comme `*` sur `+` en maths) — c'est une source classique de bugs. Les parenthèses ne coûtent rien et rendent l'intention évidente.

L'évaluation est paresseuse

JavaScript évalue les conditions **de gauche à droite** et s'arrête dès qu'il connaît le résultat :

```
if (utilisateur && utilisateur.nom === "Alice") {
    // ...
}
```

Si `utilisateur` est `null` ou `undefined`, la première partie est fausse, et **la deuxième n'est pas évaluée**. Sans cela, on aurait une erreur en lisant `.nom` sur `null`.

C'est une astuce courante : tester l'existence **avant** d'utiliser une propriété.

Exemples concrets

Contrôle d'accès

```
let role = "membre";
let estActif = true;
let estBanni = false;

if ((role === "admin" || role === "membre") && estActif && !estBanni) {
    console.log("Accès autorisé");
}
```

Validation de formulaire

```
let email = "alice@example.com";
let motDePasse = "secret123";

if (email !== "" && motDePasse.length >= 8) {
    console.log("Formulaire valide");
} else {
    console.log("Champs manquants ou trop courts");
}
```

Jeu : bonus

```
let score = 150;
let aPotion = true;
let aClef = false;

// On accède au boss si on a un gros score, OU si on a une potion ET une clef
if (score >= 200 || (aPotion && aClef)) {
    console.log("Boss débloqué !");
}
```

À retenir

- `&&` = ET — vrai si **toutes** les conditions sont vraies.

- `||` = OU — vrai si **au moins une** condition est vraie.
- `!` = NON — inverse vrai/faux.
- Quand on mélange `&&` et `||`, **toujours des parenthèses**.
- L'évaluation s'arrête dès que le résultat est connu (évaluation paresseuse). Utile pour tester l'existence d'une variable avant de l'utiliser.

JavaScript - Manipuler le DOM

> *Le DOM (*Document Object Model*) est la représentation de la page HTML que JavaScript voit. Pour rendre une page dynamique, il faut savoir **sélectionner** des éléments, **lire** leur contenu et le **modifier**. C'est la fondation de toute interactivité côté navigateur.*

C'est quoi, le DOM ?

Quand le navigateur charge une page HTML, il en construit une représentation interne : un arbre d'objets, où chaque balise devient un nœud. Cet arbre est le **DOM**.

```
      document
         |
       <html>
       /    \
  <head>   <body>
             |
           <h1>   <p>   <button>
```

JavaScript peut **lire et modifier** cet arbre — et le navigateur met à jour l'affichage en conséquence. C'est ce qui rend les pages dynamiques.

L'objet global qui te donne accès à tout ça : `document`.

Sélectionner un élément

Par identifiant — `getElementById`

L'utilisation la plus simple et la plus rapide.

```html
<p id="message">Salut !</p>
```

```js
const p = document.getElementById("message");
```

Renvoie l'élément, ou `null` si aucun élément n'a cet `id`. Un `id` doit être **unique** dans la page.

Par sélecteur CSS — `querySelector` / `querySelectorAll`

Plus flexible : on utilise la syntaxe des sélecteurs CSS.

```js
// Premier élément qui correspond
document.querySelector("#message");        // par id
document.querySelector(".surbrillance");   // par classe
document.querySelector("p");               // première balise <p>
document.querySelector("ul li.actif");     // sélecteur combiné

// Tous les éléments qui correspondent
document.querySelectorAll(".item");        // une NodeList (≈ tableau)
```

Quand utiliser quoi ?

- `getElementById` — un seul élément précis, par son id. Le plus rapide.
- `querySelector` — un seul élément, sélecteur CSS quelconque.
- `querySelectorAll` — plusieurs éléments. Renvoie une `NodeList` qu'on parcourt comme un tableau.

`querySelector` couvre 90% des cas dès qu'on dépasse les `id`.

Lire et modifier le contenu

Une fois qu'on a un élément, on peut accéder à son contenu :

```
const titre = document.querySelector("h1");

// Lire
console.log(titre.textContent);    // le texte brut
console.log(titre.innerHTML);      // le HTML interne (balises incluses)

// Modifier
titre.textContent = "Nouveau titre";
titre.innerHTML = "Titre <em>en italique</em>";
```

▸ `textContent` ou `innerHTML` ?

PROPRIÉTÉ	EFFET
`textContent`	Texte brut. Les `<` et `>` deviennent du texte affiché. **À privilégier**.
`innerHTML`	Du HTML. Les balises sont interprétées. Puissant **mais risqué** si on injecte du texte venant de l'utilisateur (faille XSS).

Règle simple : `textContent` **par défaut**. N'utilise `innerHTML` que si tu insères du HTML que **tu as écrit toi-même**, jamais du texte tapé par un utilisateur.

Modifier les attributs

Les attributs HTML (`src` , `href` , `alt` , `disabled` …) sont accessibles directement sur l'élément.

```
const img = document.querySelector("#photo");

img.src = "chat.jpg";
img.alt = "Un chat";
img.width = 300;
```

Pour les attributs personnalisés (`data-*`), on passe par `getAttribute` / `setAttribute` :

```
const carte = document.querySelector(".carte");

carte.setAttribute("data-id", 42);
console.log(carte.getAttribute("data-id"));   // "42"
```

Modifier les classes CSS

Plutôt que de toucher au style directement, on **bascule des classes** définies dans le CSS — c'est plus propre et plus maintenable.

```
const bouton = document.querySelector("#valider");

bouton.classList.add("actif");          // ajoute la classe
bouton.classList.remove("actif");       // retire la classe
bouton.classList.toggle("actif");       // ajoute si absente, retire si présente
bouton.classList.contains("actif");     // true / false
```

```
/* dans le CSS */
.actif {
    background-color: green;
    color: white;
}
```

Pour la modification ponctuelle du style en JS (couleur en direct, transition…) voir l'article suivant sur le style.

Créer et insérer des éléments

Pour ajouter dynamiquement des éléments à la page :

```
// 1. Créer
const nouveauP = document.createElement("p");
nouveauP.textContent = "Paragraphe ajouté en JS";
nouveauP.classList.add("note");

// 2. Insérer dans le DOM
document.body.appendChild(nouveauP);     // à la fin du body
```

Variantes :

```
parent.appendChild(enfant);         // ajoute à la fin
parent.prepend(enfant);              // ajoute au début
parent.insertBefore(enfant, autre);  // insère avant un autre élément
enfant.remove();                     // retire du DOM
```

▸ **Exemple : remplir une liste**

```
<ul id="fruits"></ul>
```

```
const fruits = ["pomme", "banane", "cerise"];
const liste = document.getElementById("fruits");

for (let fruit of fruits) {
    const li = document.createElement("li");
    li.textContent = fruit;
    liste.appendChild(li);
}
```

Trois fruits dans le tableau → trois `<li>` dans la page, sans toucher au HTML.

Lire la valeur d'un champ `<input>`

Pour les formulaires, ce ne sont pas `textContent` ni `innerHTML` qu'on utilise, mais la propriété `.value`.

```
<input id="nom" type="text">
<button id="valider">OK</button>
```

```
const champ = document.getElementById("nom");
const bouton = document.getElementById("valider");

bouton.addEventListener("click", () => {
    const saisie = champ.value;
    console.log(`L'utilisateur a tapé : ${saisie}`);
});
```

`.value` fonctionne pour `<input>`, `<textarea>` et `<select>`. C'est la manière propre de récupérer une saisie utilisateur sur un vrai site (plutôt que `prompt`).

Le piège : exécuter avant le chargement

```
<head>
    <script>
        const titre = document.querySelector("h1");
        titre.textContent = "Nouveau titre";  // ❌ titre est null !
    </script>
</head>
<body>
    <h1>Titre original</h1>
</body>
```

Le script s'exécute **avant** que `<h1>` n'existe dans le DOM → `querySelector` renvoie `null` → erreur en accédant à `.textContent`.

Deux solutions :

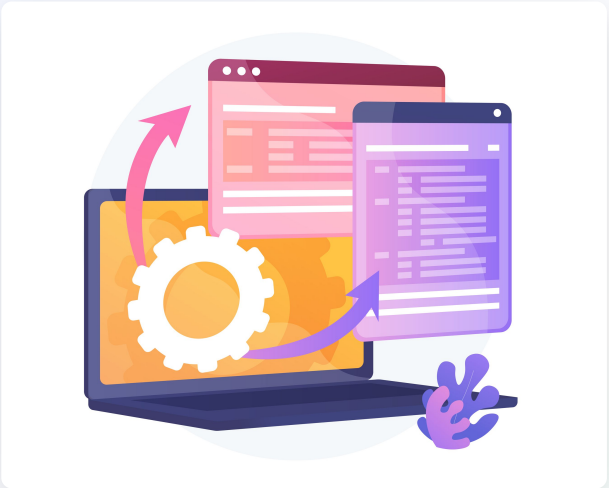
1. Mettre le `<script>` **à la fin du** `<body>`, juste avant `</body>`. C'est le réflexe simple.

2. Attendre l'événement `DOMContentLoaded` :

```
document.addEventListener("DOMContentLoaded", () => {
    const titre = document.querySelector("h1");
    titre.textContent = "Nouveau titre";
});
```

À retenir

- Le DOM = arbre d'objets que JavaScript voit, accessible via `document`.
- **Sélection** : `getElementById("id")`, `querySelector("css")`, `querySelectorAll("css")`.
- **Contenu** : `textContent` (sûr, par défaut) vs `innerHTML` (puissant, risqué).
- **Attributs** : `.src`, `.href`, `.value`… ou `getAttribute` / `setAttribute`.
- **Classes** : `classList.add` / `remove` / `toggle` — préférer les classes CSS au style en dur.
- **Création** : `createElement` + `appendChild` pour ajouter des éléments dynamiquement.
- **Champs de formulaire** : `.value` pour lire une saisie.
- **Toujours** placer le `<script>` à la fin du `<body>` (ou utiliser `DOMContentLoaded`).



SYSTÈMES D'EXPLOITATION

Fichiers et dossiers

> *Un **fichier** est un **ensemble de données** (texte, image, vidéo, programme, …) enregistré sur un support de stockage (disque dur, clé USB, carte mémoire…).*
> *Un **dossier** (ou répertoire) sert à **organiser les fichiers***

Objectifs

- Comprendre ce qu'est un fichier et un dossier.
- Savoir lire la structure d'un disque dur (arborescence).
- Faire la différence entre chemin absolu et chemin relatif.

Qu'est-ce qu'un fichier ?

Un fichier (GB ***file***) est un **ensemble de données** (texte, image, vidéo, programme, …) enregistré sur un support de stockage (disque dur, clé USB, carte mémoire…).

Chaque fichier a :

- **Un nom** (ex. `cours`)
- **Une extension** (ex. `.docx`, `.jpg`, `.mp3`) qui indique son type.

Le nom + l'extension donnent le **nom de fichier complet**.

Nom de fichier complet = [NOM.DU.FICHIER].[EXT]

Exemples :

- Nom complet: `rapport.docx` → nom de fichier = `rapport`, extension = `docx`
- Nom complet: `photo.jpg` → nom de fichier = `photo`, extension = `jpg`

Le nom de fichier est tout ce qui se trouve **AVANT le dernier point** du nom de fichier complet.

L'extension du fichier se trouve **APRÈS le dernier point** du nom du fichier. Elle indique le type de fichier et permet de déterminer le programme pour l'ouvrir. Par exemple, .docx pour un document Word ou .jpg pour une image.

Exemple:

Qu'est-ce qu'un dossier ?

Un dossier (GB ***folder***), aussi appelé répertoire (GB ***directory***) sert à **organiser les fichiers**.

- Un dossier peut contenir **des fichiers** et **d'autres dossiers**.
- On parle de **dossier parent** pour les dossiers qui contiennent d'autres dossiers.
- On parle de **dossier enfant (ou sous-dossier)** pour les dossiers qui sont à l'intérieur d'un autre.

💡 En résumé : **les dossiers contiennent des fichiers et parfois d'autres dossiers**.

Arborescence des dossiers

Les dossiers sont organisés comme un **arbre** :

- En haut, on trouve la **racine** (GB ***root***), qui est le point de départ, le dossier "le plus en haut".
- Chaque dossier peut contenir d'autres dossiers ou fichiers.

Exemple sous Windows :

```
C:\
 └── Utilisateurs
      └── Adrien
      |    └── Documents
      |    |    └── notes.txt
      |    └── Téléchargements
      └── liste.txt
```

Ici :

- `C:\` est la racine du disque C.
- `Utilisateurs` est un enfant de `C:\`.

- `Utilisateurs` a 2 enfants:
  - le dossier `Adrien`
  - le fichier `liste.txt`
- `Adrien` est l'enfant de `Utilisateurs`.
- `Adrien` a 2 dossiers enfants:
  - `Documents` est l'enfant de `Adrien`.
  - `Téléchargements` est l'enfant de `Adrien`.
- `Adrien` est donc le parent de `Documents` et `Téléchargements`
- `notes.txt` est un fichier dans `Documents`.

Les chemins d'accès

Un **chemin** indique où se trouve un fichier ou dossier.

Chemin absolu

C'est le chemin **complet** depuis la racine. Exemple :

```
C:\Utilisateurs\Adrien\Documents\notes.txt
```

Chemin relatif

C'est le chemin **par rapport à l'endroit où l'on se trouve**.

- `..` signifie « remonter d'un dossier ».

Exemple : Si je suis dans `C:\Utilisateurs\Adrien` et que je veux lire le fichier `notes.txt` qui se trouve dans `Documents`, je peux écrire :

```
type Documents\notes.txt
```

Si je veux remonter d'un cran pour afficher le fichier `liste.txt` dans `C:\Utilisateurs`, je peux écrire :

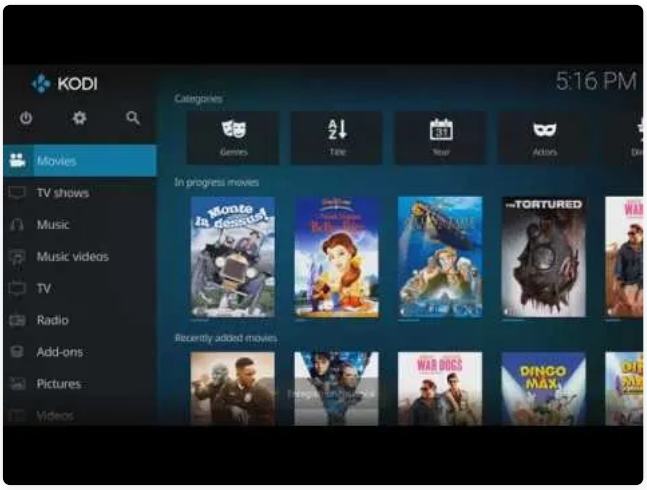
```
type ..\liste.txt
```

À retenir

- Un fichier = données enregistrées avec un nom et une extension.
- Un dossier contient des fichiers et/ou d'autres dossiers.
- Organisation en **arborescence** (racine → parents → enfants).
- **Chemin absolu** = depuis la racine.
- **Chemin relatif** = depuis l'endroit où tu es déjà (`..` pour remonter).

Organisez votre vidéothèque (2025)

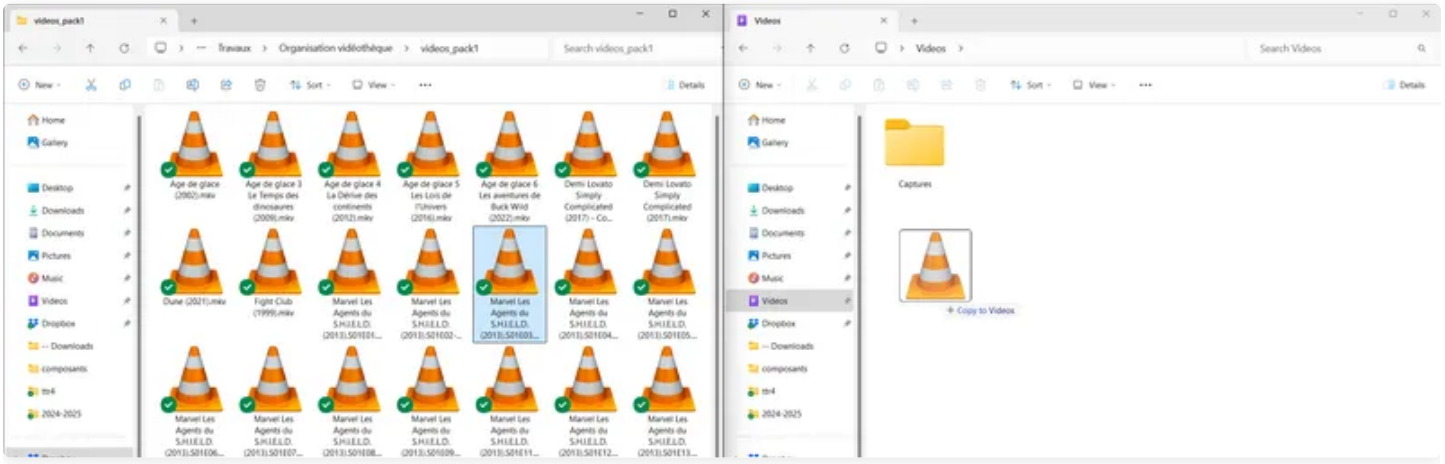
Dans cet exercice, tu vas apprendre à organiser une collection de fichiers vidéo en créant une arborescence de dossiers logique et structurée à l'aide de l'Explorateur de fichiers. Tu devras trier des fichiers, les déplacer, les renommer et les ranger de façon méthodique en suivant les consignes. Tu auras également l'occasion d'utiliser des fonctionnalités de l'Explorateur, comme le 'glisser-déposer' (drag & drop), et tu découvriras comment utiliser deux fenêtres simultanément pour travailler plus efficacement.



Objectifs d'Apprentissage

À la fin de cet exercice, tu seras capable de :

- Organiser des fichiers et des dossiers sur un disque.
- Analyser et comprendre un article technique décrivant une structure de fichiers.
- Créer des dossiers à l'aide de l'explorateur de fichiers.
- Sélectionner, copier, déplacer et renommer des dossiers et des fichiers.
- Utiliser deux fenêtres de l'explorateur de fichiers simultanément pour travailler plus efficacement.



En bonus :

- La méthode de rangement pourra t'aider à créer ton propre centre multimédia à la maison.

Instructions

Pour te permettre de suivre ta progression, utilise le `Watcher` fourni dans l'archive (voir instructions en bas de page).

1. **Télécharge ton fichier de travail sur** fichiers.ttrinfo.be
2. **Décompresse l'archive** : Une fois téléchargée, décompresse l'archive sur ton disque dans un dossier de ton choix. ⚠️ Vérifie ensuite que tu travailles dans le dossier extrait, et PAS dans l'archive.
3. **Création de l'arborescence de dossiers** : En suivant les recommandations ci-dessous, crée une arborescence de dossiers qui te permettra d'organiser ta vidéothèque. Voici quelques conseils pour la création des dossiers :
    - Tu peux créer tous les dossiers en une seule fois, puis y déplacer les fichiers.

- Ou, tu peux les créer au fur et à mesure que tu classes tes fichiers.
- Mélange les deux méthodes si tu préfères. Trouve la stratégie qui te convient le mieux !

Astuce: tu peux aussi y aller progressivement, niveau par niveau. Par exemple, tu peux d'abord déplacer tous les films dans le dossiers `Films` et toutes les séries dans le dossier `Séries` (niveau 1).

Puis pour les films, passe au niveau 2: crée les dossiers des sagas et déplaces-y les films correspondants. Crée les dossiers de chaque film et déplace le fichier correspondant.

Puis pour les séries, passe au niveau 2: crée un dossier par série et déplces-y tous les fichiers de la série. Ensuite, au 3e niveau, crée les dossiers des saisons et déplaces-y les fichiers correspondants.

🐯 Utilise deux fenêtres de l'Explorateur de fichiers : ce sera beaucoup plus simple pour glisser-déposer tes documents d'un endroit à l'autre.

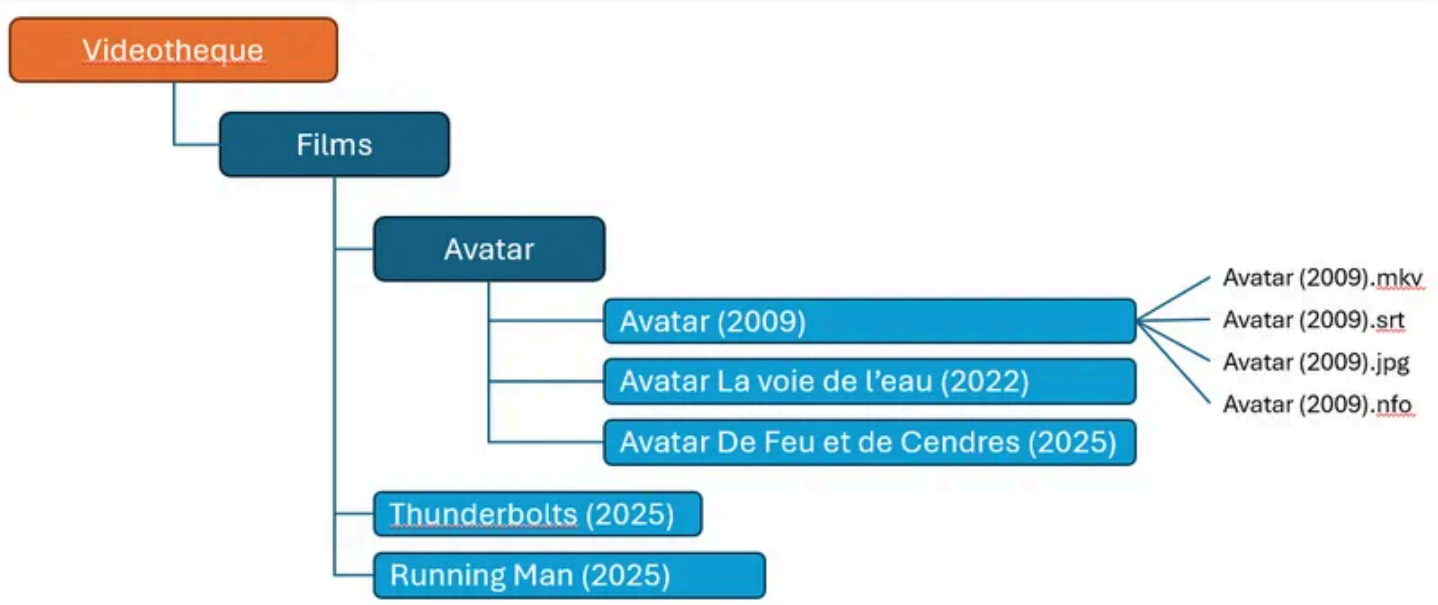
Imagine que ton ordinateur est comme ta chambre : tu as une caisse à gauche et une caisse à droite. Tu prends des objets dans la première et tu les déplaces dans la seconde. Avec deux explorateurs ouverts, c'est exactement la même chose, mais pour organiser tes fichiers.

1. **Classement des fichiers** : Utilise le "glisser-déposer" (drag & drop) pour déplacer les fichiers dans les dossiers appropriés. Pour plus de rapidité, ouvre deux fenêtres de l'Explorateur de fichiers : une pour l'arborescence de dossiers que tu crées, et l'autre pour les fichiers que tu veux trier.
2. **Besoin d'aide ?** : Consulte tes notes de cours sur l'utilisation de l'Explorateur de fichiers. N'oublie pas que ton professeur est également là pour répondre à tes questions si nécessaire.

Consignes

Ces consignes reposent sur **le bon sens et des conventions** largement acceptées dans la communauté des utilisateurs de médiathèques. L'idée principale est de séparer les différents types de médias (films, séries, concerts, etc.) dans des dossiers distincts, puis de les organiser à l'intérieur selon des critères spécifiques (par exemple, les films d'une même saga regroupés dans un dossier, les séries classées par saisons). Cette méthode te permettra non seulement de retrouver facilement tes fichiers à la main, mais aussi d'**optimiser leur reconnaissance par des logiciels comme Kodi**.

Les films



Les films doivent se trouver dans le dossier `Films` .

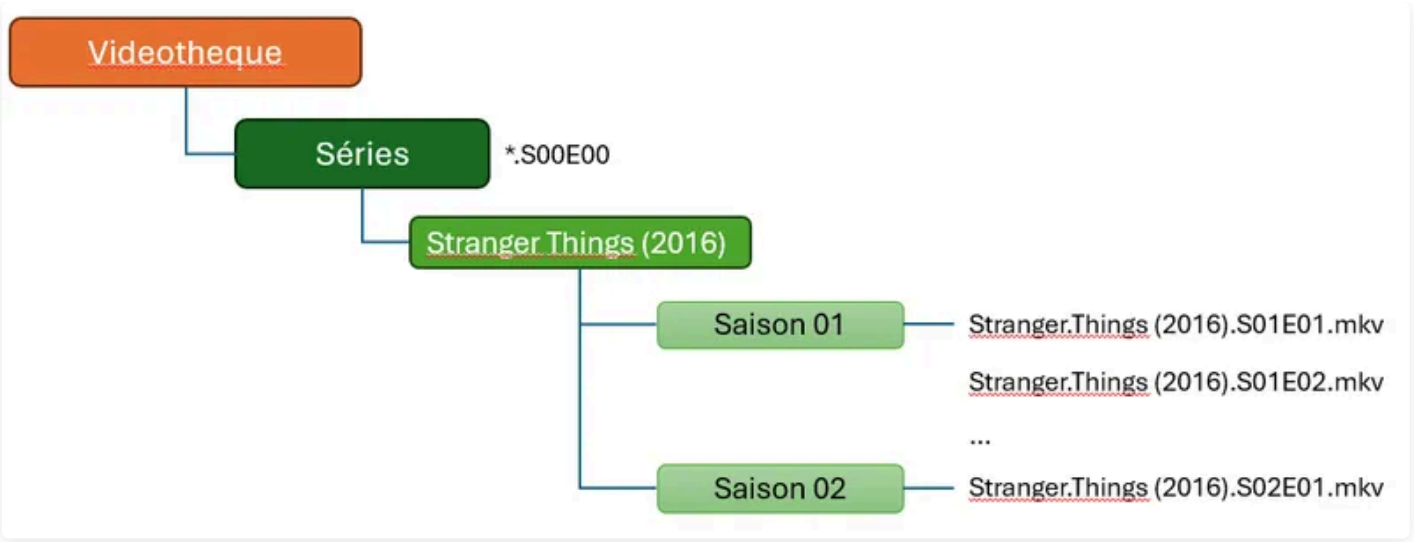
S'il n'y a qu'un seul film, place-le dans son propre dossier (nom identique à celui du film). Ex: `Thundebolts (2025).mkv` --> dossier `Thundebolts (2025)` .

S'il y a plusieurs films d'une même "série", il s'agit d'une **saga**. Dans ce cas les films doivent déplacés **dans un répertoire qui porte le nom de la saga**. Exemples de saga: Avatar, Hunger Games...

Tous les fichiers en rapport avec un même film doivent se trouver dans le même dossier.

Les séries

Tu peux identifier une série à son nom de fichier qui se termine par S00E00 --> S00 représente le numéro de saison et E00 le numéro de l'épisode dans la saison.



Chaque série doit être rangée dans son propre dossier.

Dans le dossier d'une série, il faut créer un dossier pour chaque saison, qui contient tous les fichiers des épisodes de la saison. Contrairement aux films, pas besoion de créer un dossier pour chaque épisode.

Suivi de progression

Pour suivre ta progression en temps réel, lance le programme `Watcher.exe` qui se trouve dans ton archive. Celui-ci va vérifier automatiquement si les fichiers sont à la bonne place et envoyer tes statistiques sur un serveur.

Le classement est affiché en classe.

IMPORTANT: Le classement sert à créer une petite compétition sympathique entre vous. Mais le plus important, c'est ceci : **si tu n'es pas encore en haut du classement, cela signifie simplement que tu es en train d'apprendre. Et c'est une très bonne nouvelle, parce que le but de l'activité, justement, c'est de progresser. Chaque place gagnée montre que tu avances et que tes efforts paient**. Et à la fin, ce qui compte, c'est que tu es parti·e de 0 fichiers classés. Quelle belle progression!

Et si tu as terminé l'exercice, va expliquer comment faire à quelqu'un qui a besoin d'aide!

Raccourcis claviers utiles

- `F2` : renommer un fichier
- `CTRL-SHIFT-N` : créer un nouveau dossier
- `Win- →` : déplacer la fenêtre vers la droite
- `Win- ←` : déplacer la fenêtre vers la gauche

Séquence utile pour créer le dossier d'un film/d'une série:

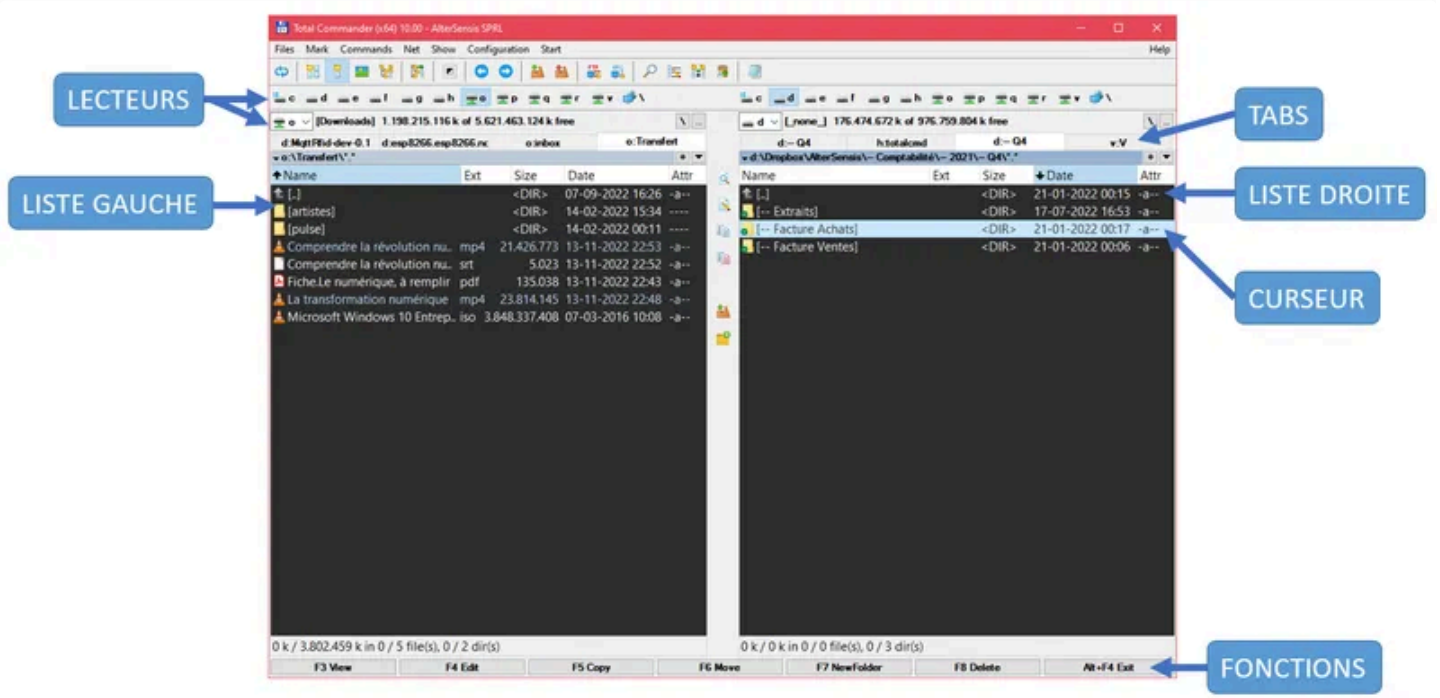
- `F2` (renomme)
- `CTRl-C` (copier le nom sélectionné, sans l'extension)
- `ESC` ou `ECHAP` ou `ENTRÉE` (qui le mode sélection)
- `CTRL-SHIFT-N` (créer un nouveau dossier)
- `CTRL-V` coller le nom du fichier
- `ENTRÉE` valider la création du dossier

Les outils Windows

Total Commander

> *Dans les métiers de l'informatique, l'efficacité et la précision sont essentielles : manipuler des fichiers, comparer des dossiers, automatiser des tâches, naviguer rapidement dans une arborescence complexe… L'explorateur de fichiers de Windows suffit pour un usage quotidien, mais il atteint vite ses limites.*

Total Commander offre une alternative professionnelle, optimisée pour la productivité, largement utilisée par les administrateurs système, techniciens et développeurs.



Qu'est-ce que Total Commander ?

Total Commander est un **gestionnaire de fichiers avancé** conçu pour Windows. Il repose sur une interface à **double panneau** (gauche/droite), permettant de visualiser simultanément deux dossiers. C'est l'héritier moderne des gestionnaires de fichiers de type "Commander", très populaires dans les environnements DOS et Linux.

Il permet notamment de :

- naviguer dans les dossiers rapidement,
- déplacer ou copier des fichiers,
- gérer les archives (ZIP, RAR, 7Z…),
- synchroniser des dossiers,
- renommer des fichiers en masse,
- se connecter à des serveurs FTP/SFTP,
- étendre les fonctionnalités par des plugins.

À quoi sert Total Commander ?

Gérer les fichiers plus efficacement

Grâce au double panneau, chaque opération (copie, déplacement, création de dossier) devient beaucoup plus rapide. Sélection → F5 → Copie. C'est plus simple et plus efficace qu'avec l'explorateur Windows.

Automatiser ou accélérer les tâches répétitives

Total Commander propose :

- renommage massif avec règles,
- sélection intelligente de fichiers,
- favoris pour sauter immédiatement d'un dossier à l'autre,
- recherche avancée avec filtres et expressions régulières.

Travailler avec des archives comme si c'étaient des dossiers

ZIP, RAR, 7z… On peut ouvrir une archive, explorer son contenu et manipuler ce qu'elle contient **sans logiciel supplémentaire**.

Synchroniser des répertoires

Idéal pour les informaticiens : on compare deux dossiers, repère les différences, et synchronise en quelques clics.

Accéder à des serveurs externes

Un module intégré permet d'accéder à :

- FTP,
- FTPS,
- SFTP (via plugin).

Pratique pour administrer des sites ou des serveurs distants.

Pourquoi Total Commander est-il supérieur à l'explorateur Windows ?

Double panneau = vitesse incomparable

L'explorateur Windows nécessite plusieurs manipulations pour passer d'un dossier à un autre ou copier un fichier. Ici, tout se fait en un geste.

Raccourcis clavier puissants

Toutes les actions sont optimisées pour le clavier :

- F3 : voir un fichier
- F4 : éditer
- F5 : copier
- F6 : déplacer
- F7 : créer un dossier
- F8 : supprimer

Pour un informaticien, cela se traduit par un **gain de productivité considérable**.

Sélection avancée des fichiers

On peut sélectionner :

- tous les .log,
- tous les fichiers > 100 Mo,
- tous les fichiers modifiés après une date précise,
- tous les fichiers correspondant à une expression régulière.

Ces fonctionnalités n'existent tout simplement pas dans l'explorateur Windows.

Comparaison et synchronisation de dossiers

Total Commander permet de comparer deux arborescences ligne par ligne, repérer les différences et synchroniser. Fonction totalement absente de l'explorateur classique.

Travail direct dans les archives

On ouvre un ZIP comme un dossier, on manipule le contenu, et on referme. Pas besoin de WinRAR, 7-Zip ou autre.

Extensions et plugins

Total Commander peut être enrichi par des plugins :

- nouveaux formats d'archives,
- outils réseau,
- visualiseurs spécialisés,
- gestionnaires de bases de données ou d'images.

Version portable

Il existe sous forme **portable**, utilisable directement sur clé USB. Idéal pour les techniciens en déplacement.

Pourquoi cet outil est-il indispensable pour les informaticiens ?

Gain de temps réel

Chaque opération est optimisée : moins de clics, moins de navigation, moins de perte de temps.

Réduction des erreurs

Grâce à la visibilité simultanée sur deux dossiers et aux outils de comparaison, les erreurs de manipulation sont fortement réduites.

Puissance pour les tâches professionnelles

Total Commander convient parfaitement pour :

- analyser des dossiers volumineux,
- nettoyer des répertoires,
- préparer des déploiements,
- gérer des fichiers logs,
- synchroniser des environnements,
- travailler avec des serveurs à distance.

Un standard dans le monde de l'IT

Depuis plus de 20 ans, Total Commander est utilisé par :

- développeurs,
- administrateurs système,
- techniciens réseau,
- équipes de support,
- enseignants en informatique.

Sa stabilité, sa rapidité et ses possibilités d'automatisation le rendent incontournable.

Conclusion

Total Commander n'est pas un simple remplacement de l'explorateur Windows : c'est un **véritable outil professionnel** conçu pour la performance et la précision. Double panneau, raccourcis, archives intégrées, synchronisation, FTP, plugins… autant de fonctionnalités qui en font un allié indispensable pour tout informaticien souhaitant travailler efficacement.

Là où l'explorateur atteint ses limites, Total Commander permet de gagner en rapidité, en organisation et en clarté.

TC - Multi-Rename Tool

Principes de fonctionnement

Multi-Rename applique un **modèle de renommage** à une sélection de fichiers ou de dossiers. La logique repose sur trois blocs :

1. **Modèle de nommage** Définition du futur nom via des variables entre crochets (`[N]` , `[C]` , `[YMD]` , etc.).
2. **Transformations** Apparence du texte : minuscules, majuscules, remplacements automatiques, etc.
3. **Substitutions et regex** Mécanismes avancés permettant de supprimer, nettoyer ou restructurer un nom existant.

L'outil permet un aperçu en temps réel : colonne « Old name » / « New name ». Aucun renommage n'est appliqué avant validation.

Variables essentielles pour construire un nom

Ces codes peuvent être combinés pour générer la structure souhaitée.

Variables liées au nom

VARIABLE	DESCRIPTION
`[N]`	Nom complet (sans extension)
`[E]`	Extension
`[N1-5]`	Sous-partie du nom (extraction par positions)
`[C]`	Compteur automatique (1, 2, 3…)
`[C:2]`	Compteur sur 2 chiffres (01, 02…)

Variables temporelles

VARIABLE	DESCRIPTION
`[YMD]`	Date du fichier au format AAAAMMJJ
`[Y-M-D]`	Date avec séparateurs
`[hms]`	Heure HHMMSS
`[h-m-s]`	Heure avec séparateurs

Exemples d'utilisation

- Renommer toutes les photos : `photo_[Y-M-D]_[C:3]` → `photo_2025-01-18_001.jpg`
- Normaliser des rapports : `rapport_[N]_[C]`
- Ajouter systématiquement un timestamp : `[N]_[YMD]_[hms]`

Zone "Replace" : remplacements simples

Ce bloc permet de nettoyer les noms :

- Replace : ` ` (espace) With : `_` → remplace tous les espaces par des underscores.
- Replace : `version` With : `v` → Clean-up de noms trop longs.

L'option **"Use Regular Expressions"** active la puissance des regex pour cas complexes :

- Find : `\s+` Replace : `_` → remplace toute séquence d'espaces multiples.
- Find : `[^A-Za-z0-9._-]` Replace : `` (vide) → supprime tous les caractères illégaux pour un import automatisé.

Case Convert : gestion des majuscules/minuscules

Options courantes :

- **Uppercase** : tout en majuscules
- **Lowercase** : tout en minuscules
- **First uppercase** : première lettre en majuscule
- **Title Case** : toutes les premières lettres en majuscule

Usage typique : normaliser un dépôt documentaire avant archivage.

Counter : numérotation contrôlée

Paramètres clés :

- **Start at** : valeur de départ
- **Step by** : incrément (exemple : +5)
- **Digits** : longueur du compteur (ex. 3 → 001)

Très utile pour des séries d'images ou de PDF séquentiels.

Extension

Champ dédié si on souhaite modifier uniquement l'extension.

Exemple : convertir des `.jpeg` en `.jpg` sans toucher au nom *Name mask :* `[N]` *Extension :* `jpg`

Exemples concrets pour informaticiens

Normalisation d'un répertoire photo téléchargé

Situation : fichiers tous différents, noms non exploitables. Objectif : tri chronologique + compteur.

Modèle : `photo_[YMD]_[C:3]` Case convert : lowercase

Résultat : `photo_20250118_001.jpg`

Créer des sous dossiers

Utilise ""

TBD

Bonnes pratiques opérationnelles

- Toujours **tester le modèle** sur quelques fichiers avant d'appliquer.
- Éviter d'utiliser plusieurs remplacements successifs si un regex simple suffit.
- Sauvegarder les modèles récurrents pour standardiser les pratiques de l'équipe.
- Effectuer le renommage sur une copie quand la perte de données serait coûteuse (exports comptables, logs).

Exercices

Télécharge les fichiers joints et essayer de les renommer (chaque série séparément) en une seule commande selon le modèle suivant:

Nom de la série (année).SxxEyy.mkv

Nom de la série: majuscule aux initiales (ex: Super Et Lois)

Structure: |Nom de la série|

- Saison xx
  - Episodes

Pour les photos: Année

- N° de mois
  - Nom original de la photo

TC - Synchronize Dirs

> *L'outil **Synchronize Dirs** de Total Commander est un instrument professionnel destiné à **comparer, analyser et synchroniser deux arborescences de fichiers**. Il est largement utilisé en administration système, support IT et sauvegarde pour détecter des écarts entre deux dossiers : fichiers manquants, modifiés, plus récents, plus anciens ou différents par le contenu. Contrairement à un simple copier–coller, Synchronize Dirs offre **une vision globale, contrôlée et réversible** du processus, avec une prévisualisation exhaustive avant toute action.*

Objectifs de l'outil

Synchronize Dirs permet de :

- comparer deux dossiers **local ↔ local**, **local ↔ réseau**, **local ↔ support externe**
- détecter les différences de **noms**, **tailles**, **dates** et **contenu**
- effectuer une **synchronisation unidirectionnelle ou bidirectionnelle**
- simuler une synchronisation **sans modifier les fichiers**
- éviter les pertes de données accidentelles

C'est un outil central pour toute stratégie de **sauvegarde raisonnée**.

Accès à l'outil

- Menu : **Commandes → Synchroniser les répertoires**

Les deux panneaux de Total Commander servent de **source (gauche)** et **destination (droite)**.

Interface générale

L'interface se décompose en quatre zones :

1. **Sélection des dossiers**
2. **Options de comparaison**
3. **Liste des différences**
4. **Actions de synchronisation**

Avant toute copie ou suppression, Total Commander affiche toujours un **état comparatif clair**.

Modes de comparaison

Comparaison par taille et date

Mode par défaut, rapide et adapté aux sauvegardes classiques.

- détecte fichiers modifiés
- sensible aux changements de date

Compare by content (comparaison par contenu)

Compare octet par octet.

- mode le plus fiable
- plus lent
- indispensable pour audit ou contrôle d'intégrité

Options essentielles

Ignore date

Ignore les différences de date de modification. → Utile lorsque les systèmes ne synchronisent pas correctement les timestamps.

Subdirs

Inclut tous les sous-dossiers. → Toujours recommandé sauf cas très spécifique.

Asymmetric

Mode **sauvegarde classique** :

- le dossier de gauche est la référence
- le dossier de droite est mis à jour pour lui ressembler

Synchronize

Mode **bidirectionnel** :

- les fichiers les plus récents sont copiés dans l'autre dossier. ⚠️ À utiliser uniquement si la logique métier est claire.

Lecture des résultats (codes couleurs)

Cette visualisation permet une **lecture immédiate de l'état du système**.

COULEUR	SIGNIFICATION
Noir	fichiers identiques
Rouge	fichier différent
Vert	fichier plus récent
Bleu	fichier absent

Prévisualisation et sécurité

Avant toute action :

les opérations proposées sont listées

- chaque fichier peut être sélectionné ou exclu
- aucune suppression n'est automatique sans validation

Il est possible de :

- décocher des fichiers
- inverser le sens de copie
- annuler l'opération

➡️ **Aucune synchronisation n'est forcée**.

Actions possibles

- Copier gauche → droite
- Copier droite → gauche
- Supprimer fichiers excédentaires
- Synchroniser automatiquement selon règles choisies

Les actions ne sont exécutées qu'après clic sur **Synchronize**.

Cas d'usage concrets

Sauvegarde incrémentielle

Dossier de travail → disque externe Mode : *Asymmetric + Compare by date & size*

Vérification de sauvegarde

Original ↔ sauvegarde Mode : *Compare by content*

Migration de poste Windows

Ancien PC ↔ nouveau PC Repérage des fichiers manquants ou divergents

Audit de données

Détection de modifications non autorisées

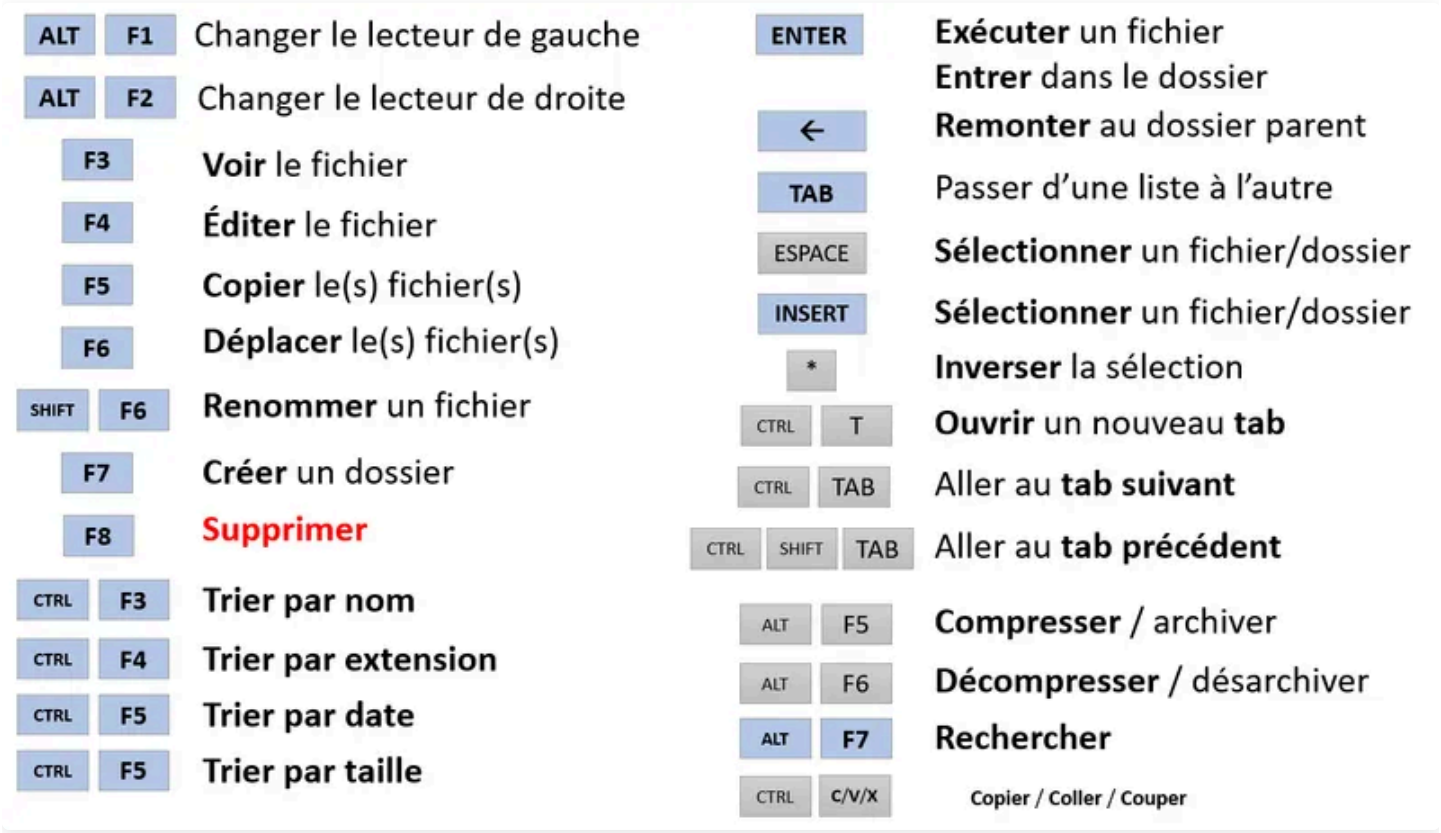
Bonnes pratiques professionnelles

- Toujours **prévisualiser** avant synchronisation
- Privilégier **Compare by content** pour données critiques
- Utiliser **Asymmetric** pour les sauvegardes
- Éviter la synchronisation bidirectionnelle en environnement multi-utilisateurs
- Tester sur une **copie** lors des premiers usages

TC - 25 raccourcis clavier essentiels

> *Toute la puissance de Total Commander vient du fait que TOUT peut être fait à l'aide de la souris.*

Voici une liste des principaux raccourcis claviers de Total Commander. Sache qu'il est possible de la personnaliser et d'ajouter tes propres raccourcis.



Opérations principales

- **F3** : Voir un fichier (visualiseur)
- **F4** : Modifier un fichier (éditeur)
- **F5** : Copier vers le panneau opposé
- **F6** : Déplacer/renommer vers le panneau opposé
- **F7** : Créer un dossier
- **F8** ou **Suppr** : Supprimer

Navigation

- **Tab** : Changer de panneau
- **Backspace** : Remonter d'un dossier (parent)
- **Alt + Flèche gauche/droite** : Historique dossier précédent/suivant
- **Ctrl + R** : Actualiser la vue
- **Ctrl + *** : Aller à la racine du disque

Sélection

- **Espace** : Sélectionner un élément + afficher taille cumulée
- **Insert** : Sélectionner et passer au suivant
- **Ctrl + A** : Tout sélectionner
- **Num / (pavé numérique)** : Inverser la sélection
- **Ctrl + B** : Afficher tous les fichiers (inclus sous-dossiers)

Recherche et filtrage

- **Alt + F7** : Recherche avancée
- **Ctrl + S** : Filtre rapide (taper pour filtrer)

Tri (indispensable)

- **Ctrl + F3** : Trier par **nom** (ordre alphabétique)
- **Ctrl + F4** : Trier par **extension**
- **Ctrl + F5** : Trier par **heure/date**
- **Ctrl + F5** : Trier par **taille**

Travail avancé

- **Ctrl + M** : Renommage en masse
- **Ctrl + Q** : Aperçu rapide dans le panneau latéral
- **Ctrl + D** : Ouvrir les dossiers favoris
- **Ctrl + F** : Gestionnaire FTP
- **Alt + F5** : Créer une archive
- **Alt + F6** : Extraire une archive



TECHNOLOGIES

3D et impression

De la modélisation 3D sur ordinateur à l'objet physique qui sort d'une imprimante : on apprend à dessiner en volume avec TinkerCad, à comprendre comment fonctionne une imprimante 3D, et on termine par la fabrication d'un petit objet.

TinkerCad — modéliser ton premier objet

> *Premier contact avec la modélisation 3D : tu fabriques un porte-clés personnalisé avec tes initiales. En le construisant, tu apprends à manipuler la vue, à empiler des solides, à percer, à aligner et à redimensionner — tout ce qu'il faut pour modéliser un objet simple.*

TinkerCad est un outil de **modélisation 3D** gratuit qui tourne dans le navigateur. Pas d'installation, pas de licence : tu te crées un compte (ou tu en utilises un fourni par le prof) et tu commences. Il est volontairement simple — pas de fonctions ultra avancées — mais il suffit largement à dessiner les objets qu'on va imprimer.

Dans ce cours, on apprend en construisant : tu vas modéliser un **porte-clés à tes initiales**. Pendant la construction, tu rencontres tous les gestes de base.

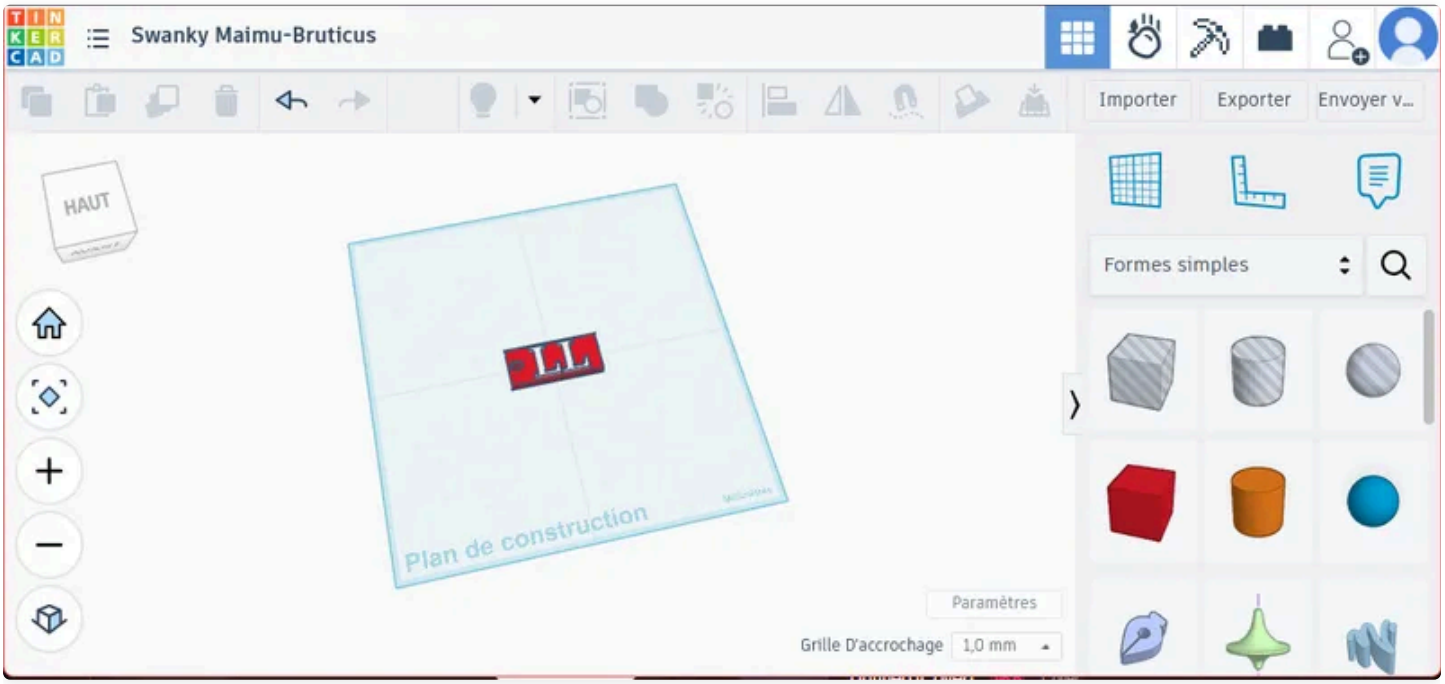
🎯 Objectifs

À la fin de ce cours, tu seras capable de :

1. Te repérer dans l'espace 3D de TinkerCad (zoomer, tourner, déplacer la vue).
2. Ajouter des **solides** et des **perçages** depuis la bibliothèque.
3. **Déplacer**, **redimensionner** et **aligner** des formes les unes par rapport aux autres.
4. Combiner solides et perçages pour creuser un objet (faire un trou, graver un texte).
5. Exporter ton modèle au format **STL** pour l'envoyer vers une imprimante 3D.

Partie 1 — L'interface et la vue 3D

Une fois connecté sur tinkercad.com, clique sur **Créer** → **Conception 3D**. TinkerCad t'ouvre un atelier vide.



Le **plan de construction** est ce grand sol quadrillé. C'est ton espace de construction, gradué en **millimètres** : un cube de 20 mm de côté sera bel et bien un cube de 2 cm une fois imprimé.

> 💡 ***Si tu as déjà vu une imprimante 3D**, pense au plan de travail comme au **plateau d'impression** : sa taille en TinkerCad correspond à la zone où l'imprimante peut déposer du plastique. Tout ce qui dépasse du plateau ne pourra pas être imprimé.*

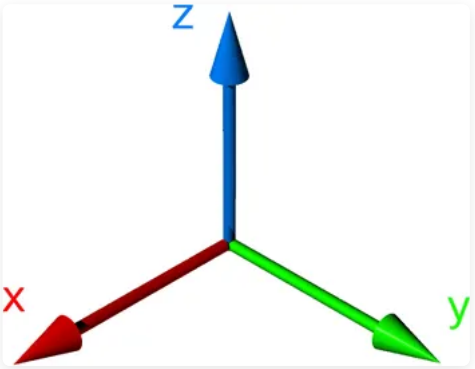
▸ Bouger la vue avec la souris

Quatre gestes à mémoriser tout de suite — c'est ce que tu vas faire le plus souvent :

> 💡 ***Si tu es perdu :** en haut à gauche, le petit cube ViewCube te permet de revenir à une vue propre (clique sur la face "Haut", "Avant", etc.). Le bouton **Maison** revient à la vue par défaut.*

GESTE SOURIS	EFFET
Clic gauche	Sélectionner / manipuler les objets (déplacer, redimensionner, lever)
Molette	Zoom avant / arrière
Clic droit maintenu + glisser	Faire tourner la vue (orbiter)
Clic molette maintenu + glisser (ou Shift + clic droit)	Déplacer la vue (sans rotation)

Les axes X, Y, Z



Trois directions :

- **X** — droite ↔ gauche
- **Y** — devant ↔ derrière
- **Z** — haut ↔ bas

Quand tu sélectionnes un objet, des petites flèches apparaissent dans ces trois directions : ce sont les axes. C'est avec elles que tu vas déplacer, redimensionner ou lever l'objet.

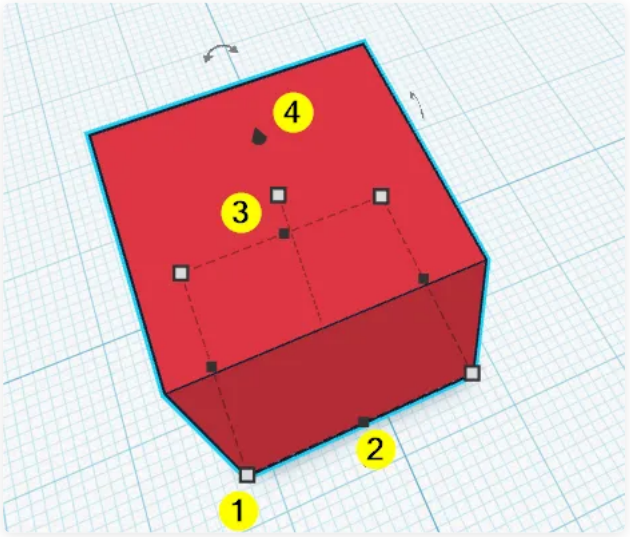
Partie 2 — Ajouter une première forme

À droite de l'écran, la **bibliothèque des formes** liste tous les solides disponibles : cube, cylindre, sphère, prisme, texte, etc.

Étape 1 : glisse un **cube rouge** depuis la bibliothèque vers le plan de travail. Tu obtiens un cube de 20 × 20 × 20 mm posé au sol.

Sélectionner, déplacer, lever

Clique sur le cube pour le sélectionner. Plusieurs éléments apparaissent autour de lui :



- des **poignées** : petits carrés blancs dans les coins [1], noirs sur les arêtes [2] et un blanc sur le dessus [3]. C'est ce qui sert à le **redimensionner** ;
- un **cône noir** pointé vers le haut [4], séparé du corps de la forme — c'est ce qui sert à le **lever** (le déplacer verticalement, sur l'axe Z). La pointe du cône devient claire au survol.

Pour déplacer le cube sur le plan de travail :

- Clique au **centre** du cube, maintiens, et glisse — il suit la souris en restant collé au sol.
- Ou utilise les **flèches du clavier** pour des déplacements précis de **1 mm** (ou **1 cm** avec Maj enfoncé).
- Tu peux fixer un seul axe de déplacement en appuyant sur la touche **Shift/Maj** en même temps.

Pendant que tu déplaces le cube, deux indications de distance s'affichent à côté de lui : une par axe du plan de travail. Le sol a en effet **deux axes** — **X** (de gauche à droite) et **Y** (de l'avant vers l'arrière). Le troisième axe, **Z**, est la hauteur — c'est celui qu'on utilise avec le cône noir pour lever l'objet. Quand tu glisses ton cube sur le sol, tu te déplaces donc en X et en Y simultanément ; avec Shift/Maj enfoncé, tu n'en bouges plus qu'un seul à la fois.

Pour lever le cube au-dessus du plan :

- Saisis le **cône noir** [4] du dessus et tire-le vers le haut. Une indication de hauteur apparaît : c'est la valeur Z.
- Pour le reposer au sol, tire-le vers le bas jusqu'à 0.

Pour changer ses dimensions : clique et tire sur une des poignées blanches ou noires (voir ci-dessous).

> ⚠️ ***Attention :*** *si tu cliques sur une* ***poignée*** *au lieu du corps de la forme, tu vas la redimensionner au lieu de la déplacer. Si ton objet se déforme alors que tu voulais juste le bouger, fais* ***Ctrl + Z*** *et recommence en cliquant bien au centre.*

Redimensionner

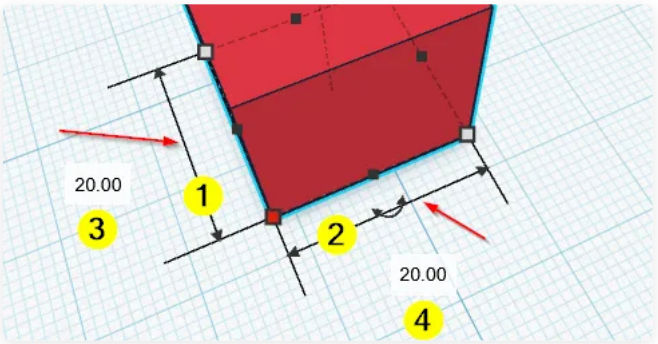
Les poignées blanches [1] et noires sur les côtés [2] étirent la forme dans une ou deux directions. Les poignées blanches **en coin** (celles du bas) permettent de redimensionner les 2 côtés qu'elles touchent en même temps. Les poignées noires permettent de changer la dimension sur un seul axe. Appuie sur **Shift/Maj** pour que les autres dimensions s'adaptent en même temps : ta forme garde alors les mêmes proportions.

Pour un porte-clés, on veut une base **rectangulaire plate**. Sélectionne ton cube, puis tape dans la case de dimension qui apparaît :

- **Largeur (X)** : 50 mm
- **Profondeur (Y)** : 20 mm
- **Hauteur (Z)** : 4 mm

Tu obtiens une plaque rectangulaire, fine et allongée. C'est la base de ton porte-clés.

Quand tu modifies une dimension, une ligne de mesure apparaît [1+2] (ou deux lignes si tu modifies deux dimensions en même temps).

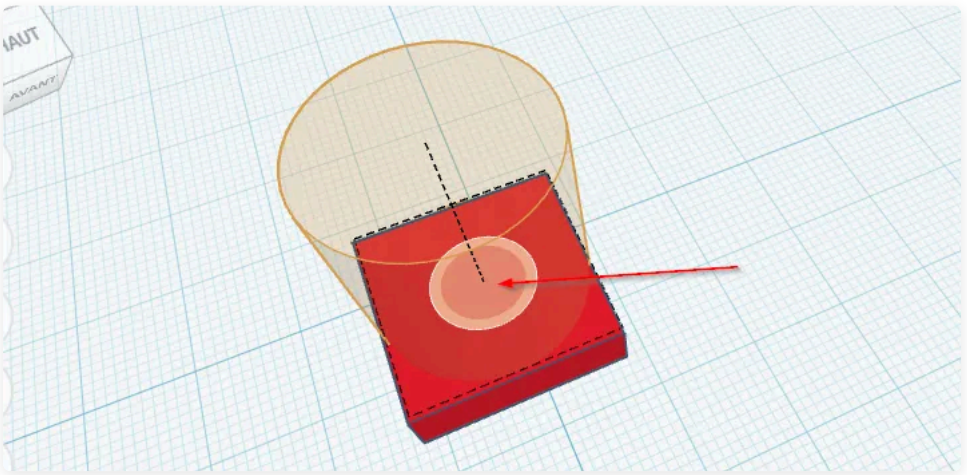


> 💡 ***Astuce :*** *tu peux toujours cliquer directement sur les chiffres affichés sous la ligne de mesure [3+4] pour entrer une valeur précise au clavier. C'est plus rapide et plus précis que de tirer à la souris.*

Partie 3 — Ajouter du texte (tes initiales)

Poser le texte directement sur la plaque

TinkerCad a une astuce très pratique : quand tu **glisses une nouvelle forme depuis la bibliothèque**, tu peux la déposer sur n'importe quelle surface d'un objet déjà présent — elle se posera dessus, pas au sol :



> ⚠️ ***C'est seulement au moment de la création*** *que ça fonctionne. Un objet déjà posé au sol ne se collera* ***pas*** *automatiquement sur un autre objet, même si tu le glisses au-dessus. Pour reposer un objet existant, il faut utiliser le* ***cône noir*** *(voir plus bas).*

1. Dans la bibliothèque, cherche la forme **Texte** (Text en anglais).
2. Glisse-la depuis la bibliothèque, **directement au-dessus de la plaque**, sans la lâcher.
3. Pendant que tu déplaces, regarde l'objet : une **ombre arrondie** sous le curseur indique la surface de dépôt. Quand l'ombre est sur le dessus de la plaque, relâche.
4. Le texte se pose à hauteur Z = **4 mm**, posé sur la plaque, et non au sol.

Régler le contenu du texte

Une zone de réglage apparaît à droite quand le texte est sélectionné :

- **Text** : tape tes initiales (ex. `LL`).
- **Police** : laisse celle par défaut, ou choisis une plus lisible.
- **Hauteur (Z)** : 4 mm (la même que la plaque).

Redimensionne ensuite ton texte pour qu'il soit un peu plus petit que la plaque, par exemple **20 × 10 × 4 mm**.

Si tu t'es trompé et que le texte est posé au sol

Pas de panique — c'est facile à corriger avec le **cône noir** :

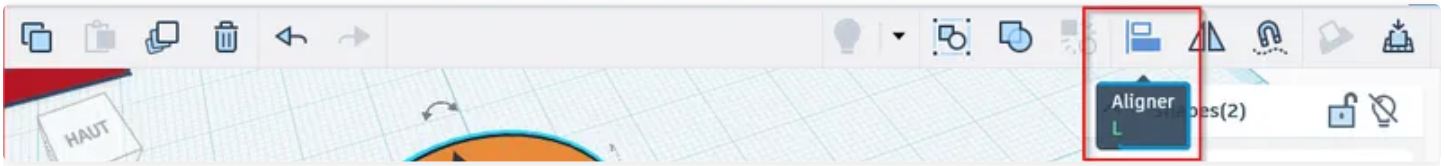
1. Sélectionne le texte.
2. Saisis le **cône noir** au-dessus de la forme (la flèche vers le haut, séparée du corps).
3. Tire-le vers le haut jusqu'à une hauteur Z de **4 mm**. Tu peux aussi cliquer sur la valeur affichée et la taper directement au clavier.

Tu as maintenant la plaque + le texte qui dépasse de 4 mm. Ton porte-clés commence à ressembler à quelque chose.

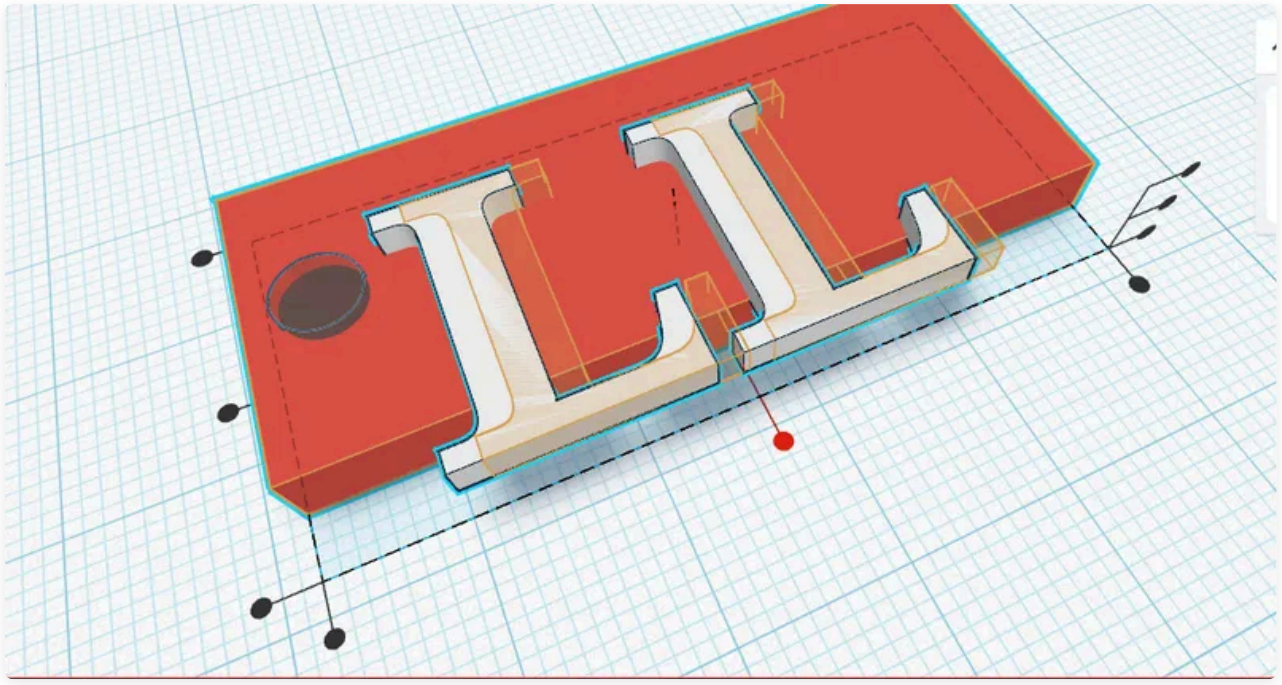
Partie 4 — Aligner deux objets

À l'œil, ton texte n'est probablement pas parfaitement centré sur la plaque. TinkerCad a un outil dédié : **Aligner**.

1. Sélectionne **les deux objets** (la plaque ET le texte). Pour ça : clique le premier, puis Shift + clic sur le second. Ou trace un rectangle de sélection autour des deux.
2. Clique sur le bouton **Aligner** en haut à droite (ou appuie sur **L**).



Des petites poignées noires apparaissent autour de la sélection : **3 par direction** (début, milieu, fin), pour chacun des axes X, Y et Z.



Choisir l'objet qui ne bouge pas

Par défaut, l'alignement utilise un **point central de la sélection** : les deux objets vont se rapprocher l'un de l'autre. Souvent, ce n'est pas ce que tu veux — tu veux que la **plaque reste à sa place** et que **le texte vienne se centrer dessus**.

Pour ça : pendant que tu es en mode Aligner, **clique sur la plaque**. Elle est désignée comme **objet de référence** (un petit indicateur visuel apparaît dessus) : c'est elle qui ne bougera plus, et les autres objets s'aligneront par rapport à elle.

Centrer le texte sur la plaque

Avec la plaque désignée comme référence :

3. Clique sur la poignée du **milieu** dans le sens de la **largeur** (axe X, de gauche à droite) pour centrer le texte horizontalement.
4. Clique sur la poignée du **milieu** dans le sens de la **profondeur** (axe Y, d'avant en arrière) pour centrer le texte de l'autre côté.

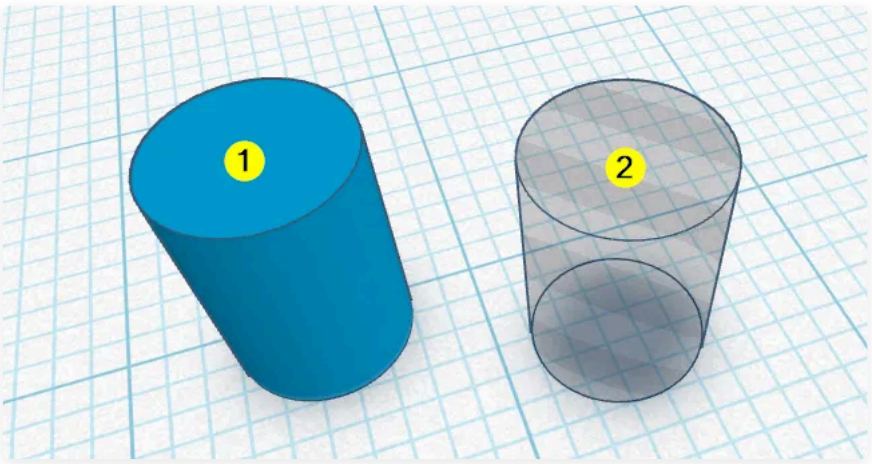
Le texte est maintenant parfaitement centré sur la plaque. Appuie sur **Échap** ou clique ailleurs pour sortir du mode Aligner.

> 🧠 ***À retenir :*** *la fonction Aligner agit sur* ***tous les objets sélectionnés en même temps****. C'est l'outil le plus utile pour avoir un travail propre — utilise-le systématiquement à la fin de chaque étape, en désignant bien l'objet de référence par un clic.*

Partie 5 — Percer : le trou pour l'anneau

Un porte-clés sans trou pour l'anneau de clés, ça ne sert à rien. C'est ici qu'on découvre une mécanique importante de TinkerCad : les **perçages**.

Solide vs perçage



Dans TinkerCad, chaque forme est soit :

TYPE	APPARENCE	EFFET
Solide [1]	Couleur pleine	Ajoute de la matière
Perçage [2]	Hachuré, semi-transparent	Enlève de la matière à ce qui se trouve dessous

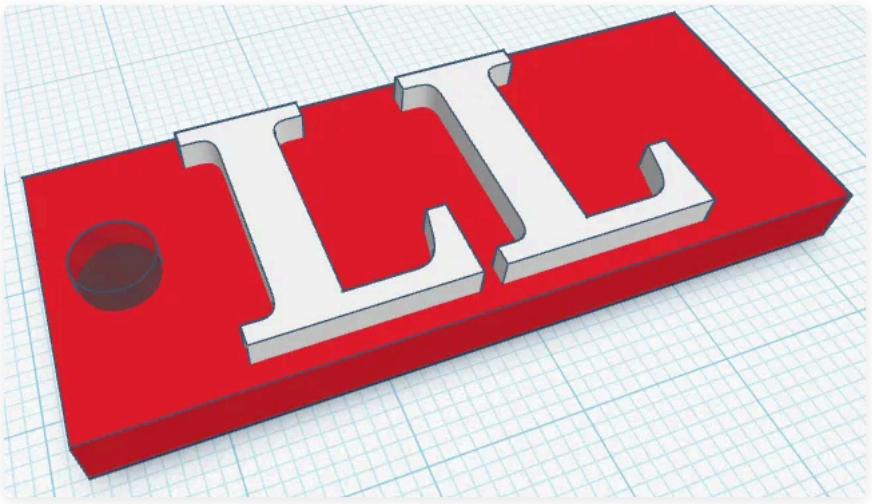
Un perçage **n'est pas un trou tout seul** : c'est une forme qui dit "tout ce que je touche, tu l'efface". Pour que le trou existe vraiment, il faut **regrouper en union** un solide et un perçage ensemble.

Créer le trou

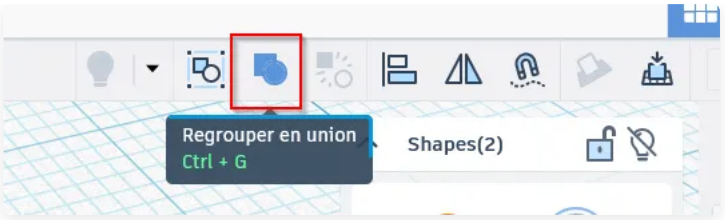
1. Glisse un **cylindre** depuis la bibliothèque.
2. Redimensionne-le : diamètre **6 mm** (donc 6 mm en X et en Y), hauteur **5 mm** (légèrement plus que la plaque pour bien la traverser).
3. Avec le cylindre sélectionné, dans le petit panneau en haut à droite, clique sur **Perçage** (l'option à côté de "Solide"). Le cylindre devient hachuré.
4. Déplace-le sur la partie **gauche** de la plaque, là où ira l'anneau.
5. Vérifie qu'il **traverse bien** la plaque de haut en bas (la base du cylindre doit être au niveau du sol, ou même en dessous).

Grouper pour faire le trou

Le trou n'apparaîtra pas tant que tu n'auras pas dit à TinkerCad de **combiner** la plaque et le perçage.



1. Sélectionne **la plaque ET le cylindre-perçage** (Shift + clic, ou rectangle de sélection).
2. Clique sur **Regrouper en union** en haut à droite (ou **Ctrl + G**).



Boum : la plaque est désormais percée. Tu peux faire tourner la vue pour vérifier que le trou la traverse vraiment.



> ⚠️ ***Si le trou n'apparaît pas :*** *c'est presque toujours parce que le cylindre est encore réglé en Solide, ou parce qu'il ne traverse pas complètement la plaque. Annule (**Ctrl + Z**), vérifie les deux points, et recommence.*

> 💡 ***Tu peux dissocier** avec **Ctrl + Maj + G** si tu t'es trompé. Tant que tu ne quittes pas TinkerCad, tu peux aussi toujours revenir en arrière (**Ctrl + Z**).*

Partie 6 — Finir le porte-clés

À ce stade, tu as :

- une plaque rectangulaire percée à gauche ;
- un texte avec tes initiales posé dessus.

Si tu veux un seul objet qui se tient ensemble, sélectionne **les deux**, puis **Regrouper en lot** (Ctrl + G). Tu obtiens un seul objet, ton porte-clés.

Variantes possibles

- **Texte gravé au lieu de relevé** : passe ton texte en *Perçage* avant de regrouper avec la plaque. Au lieu de sortir, les lettres seront creusées dedans.
- **Coins arrondis** : utilise la forme *Round Roof* ou *Cylindre* pour adoucir les angles de la plaque. Ou change le rayon dans les propriétés de la forme.
- **Forme particulière** : remplace le rectangle par une étoile, un cœur, un logo de classe…

Partie 7 — Exporter pour l'impression

Quand ton modèle te plaît, il faut le sortir au format **STL** — le format universel des fichiers à imprimer en 3D.

1. En haut à droite, clique sur **Exporter**.
2. Choisis **Tout dans la conception** (sinon TinkerCad n'exporte que ce qui est sélectionné).
3. Sélectionne le format **.STL**.
4. Le fichier se télécharge dans ton dossier *Téléchargements*.

Ce fichier `.stl`, c'est ce que ton prof (ou toi) va ouvrir dans un **slicer** pour le préparer à l'impression. On verra ça dans le cours suivant.

> ℹ️ ***Format OBJ** : TinkerCad propose aussi l'export `.obj`. Pour l'impression 3D, reste sur **STL** — c'est le standard universel et celui que tous les slicers acceptent sans broncher.*

💉 Exercices

Exercice 1 — Reproduire le porte-clés

Suis pas à pas les 7 parties ci-dessus, sans sauter d'étapes. À la fin, tu dois pouvoir créer :

- une plaque de 50 × 20 × 4 mm ;
- tes initiales en relief (ou gravées) sur le dessus ;
- un trou de 6 mm de diamètre traversant la plaque ;
- un fichier `.stl` exporté.

Exercice 2 — Un trophée empilé

Modélise un **trophée** en empilant quatre formes : un **cylindre**, un **cône** et **deux sphères**. Pour chaque pièce après la première, glisse-la **directement depuis la bibliothèque** sur la surface de la pièce précédente — utilise la technique de l'ombre arrondie pour qu'elle se pose dessus, pas au sol.

Proposition de proportions :

- **Socle** : un **cylindre** plat et large (Ø 40 mm, hauteur 8 mm), posé au sol.
- **Globe** : une **grosse sphère** (Ø 30 mm), posée au centre du socle.
- **Flamme** : un **cône** (Ø 15 mm, hauteur 25 mm), pointe vers le haut, posé sur le globe.
- **Boule finale** : une **petite sphère** (Ø 10 mm), posée sur la pointe du cône.

Utilise l'outil **Aligner** à chaque étape pour que chaque pièce soit bien centrée sur la précédente (sélectionne les deux objets, désigne le bas comme référence, puis clique sur les poignées centrales en X et en Y).

Quand tout est en place et bien aligné, sélectionne le tout et **regroupe en lot** (Ctrl + G) pour obtenir un seul objet.

Exercice 3 — Une boîte vide avec une paroi de 3 mm

Modélise une **boîte** :

- Base : un cube, 50 mm de large, 50 mm de haut.
- Intérieur creusé : un autre cube en *Perçage*, légèrement plus petit (épaisseur de mur de 3 mm) — détermine la taille correcte de ce cube toi-même.
- Aligne les deux, regroupe en union, et tu as une boîte solide avec un fond.

Exercice 4 — Un porte-stylos hexagonal

Modélise un **porte-stylos** :

- Base : un prisme à 6 côtés (forme *Polygon* avec 6 côtés), 50 mm de large, 60 mm de haut.
- Intérieur creusé : un autre prisme à 6 côtés en *Perçage*, légèrement plus petit (épaisseur de mur de 3 mm) — détermine la taille correcte toi-même.
- Aligne les deux, regroupe en union, et tu as un porte-stylos solide avec un fond.

Exercice 5 — Un dé à 6 faces

Modélise un **dé à jouer** classique de 16 × 16 × 16 mm :

- Un cube de 16 mm de côté.
- Sur chaque face, le nombre de points correspondant (1 sur la face du dessus, 6 sur celle du dessous, etc., la somme des faces opposées doit faire 7).
- Les points sont des **petites sphères en perçage** (diamètre 3 mm), creusées dans le cube.

Cet exercice te force à travailler **toutes les faces** d'un objet — pas seulement le dessus. Utilise le ViewCube pour faire pivoter la vue entre chaque face.

✍️ À retenir

- TinkerCad travaille en **millimètres** sur un plan de travail quadrillé — ce que tu vois à l'écran, c'est la taille réelle de l'objet imprimé. Le plan de travail correspond au plateau de l'imprimante.
- Gestes souris : **clic gauche** = manipuler les objets, **molette** = zoom, **clic droit** = orbiter, **Shift + clic droit** (ou clic molette) = déplacer la vue.
- Pour poser un objet sur la surface d'un autre, glisse-le **directement depuis la bibliothèque** au-dessus de cet objet — l'ombre sous le curseur montre la surface de dépôt. Un objet déjà existant ne se posera pas tout seul : il faut le lever avec le **cône noir**.
- Chaque forme est soit **Solide** (ajoute de la matière), soit un **Perçage** (enlève de la matière).
- Un perçage n'a d'effet qu'une fois **regroupé** (Ctrl + G) avec le solide qu'il doit creuser.
- L'outil **Aligner** (L) aligne plusieurs objets entre eux — utilise-le systématiquement.
- On exporte au format **STL** pour passer à l'impression.

Suite

Maintenant que tu sais modéliser un objet, il reste à comprendre **comment l'imprimer**. Dans le prochain cours, on regarde comment fonctionne une imprimante 3D, ce qu'est un slicer, et quels filaments existent pour faire le bon choix.

L'impression 3D — comment ça marche

> *Une vue d'ensemble de l'impression 3D grand public : le processus complet du modèle au filament fondu, les principales technologies d'imprimantes, et les types de filaments que tu vas croiser le plus souvent.*

Tu sais maintenant modéliser un objet dans TinkerCad. Mais entre ton fichier `.stl` et l'objet en plastique qui sort de la machine, il y a plusieurs étapes que la plupart des gens ignorent. Ce cours met les choses au clair : ce qu'il se passe vraiment, quelles machines existent, et avec quels matériaux on imprime.

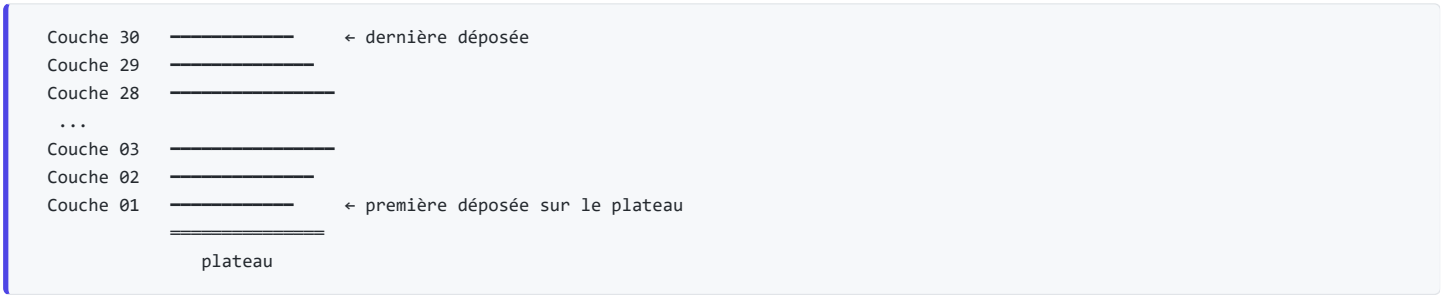
🎯 Objectifs

À la fin de ce cours, tu seras capable de :

1. Décrire les **étapes** entre un fichier 3D et un objet imprimé.
2. Expliquer ce qu'est un **slicer** et à quoi il sert.
3. Distinguer les principales **technologies** d'impression 3D grand public (FDM, résine).
4. Choisir un type de **filament** adapté à un objet donné (jouet, pièce technique, prototype…).

Partie 1 — Le principe : empiler des couches

Toutes les imprimantes 3D grand public reposent sur le même principe : **construire l'objet couche par couche, du bas vers le haut**. Au lieu d'enlever de la matière (comme une fraise ou un tour), on en ajoute petit à petit. C'est ce qu'on appelle la **fabrication additive** (GB ***additive manufacturing***).

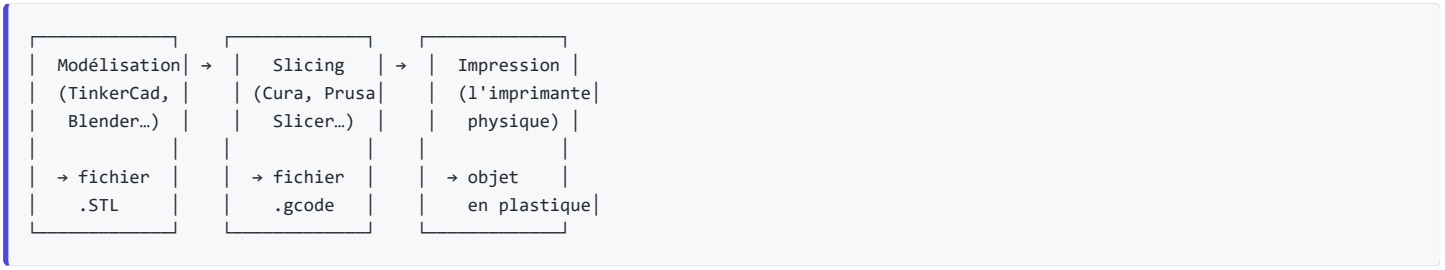


Chaque couche fait typiquement entre **0,1 mm et 0,3 mm** d'épaisseur. Plus la couche est fine, plus l'objet sera détaillé — mais plus l'impression sera longue. C'est un des compromis classiques de l'impression 3D : **qualité contre temps**.

> 💡 ***Ordre de grandeur :*** *un objet de la taille d'une balle de ping-pong (40 mm de haut) en couches de 0,2 mm = 200 couches. Si chaque couche prend 30 secondes, l'impression dure 1 h 40. Une vraie pièce technique avec des couches fines peut tourner toute la nuit.*

Partie 2 — Le processus complet

Du modèle à l'objet, il y a trois étapes :



▸ Étape 1 — Modéliser

C'est ce que tu as appris à faire dans le cours précédent : dessiner ton objet en 3D dans TinkerCad (ou Blender, Fusion 360, FreeCAD…) et l'**exporter en STL**.

Le fichier STL contient juste **la forme** de l'objet — un maillage de triangles qui décrit la surface. Aucune information sur la couleur, le matériau, ou comment l'imprimer.

▸ Étape 2 — Trancher (slicer)

Le STL ne suffit pas à piloter une imprimante. Il faut maintenant le découper en **tranches horizontales** — c'est le boulot du **slicer**.

Le slicer est un logiciel qui :

1. Prend ton fichier STL en entrée.
2. Te demande **quelle imprimante** tu utilises, **quel filament**, **quelle épaisseur de couche**, etc.
3. Découpe virtuellement l'objet en couches horizontales.
4. Calcule, pour chaque couche, **le chemin exact** que la tête d'impression doit suivre.
5. Génère un fichier **G-code** : la liste des mouvements à faire, dans l'ordre.

Les slicers grand public les plus connus :

SLICER	PARTICULARITÉ
Cura	Le plus populaire, gratuit, développé par Ultimaker
PrusaSlicer	Excellent et gratuit, optimisé pour les imprimantes Prusa mais marche avec d'autres
OrcaSlicer	Récent, dérivé de PrusaSlicer, très utilisé pour les imprimantes Bambu Lab
Bambu Studio	Le slicer maison de Bambu Lab

Dans le slicer, on règle plein de paramètres importants :

- **Épaisseur des couches** (0,1 à 0,3 mm).
- **Remplissage** (GB ***infill***) — l'intérieur de l'objet est creux et rempli d'un motif (souvent en nid d'abeille) à un taux qui va de 10 % à 100 %. Plus le taux est élevé, plus l'objet est solide… et plus l'impression est longue et consomme de filament.
- **Supports** — pour les parties en surplomb (un toit, un bras tendu). Le slicer ajoute des "échafaudages" en plastique qu'on casse à la fin.
- **Vitesse** — combien de mm/s la tête se déplace.
- **Température** — varie selon le filament.

Étape 3 — Imprimer

Une fois le G-code prêt, on l'envoie à l'imprimante (par carte SD, USB, WiFi…). La machine exécute les mouvements un par un, dépose le plastique fondu, et après quelques minutes ou quelques heures, l'objet est terminé.

> 🧠 ***À retenir :*** *un slicer ne **dessine pas** : il **traduit** ton modèle 3D en instructions machine. C'est le pont obligatoire entre le monde 3D et le monde physique. Aucune imprimante grand public ne sait lire un STL directement.*

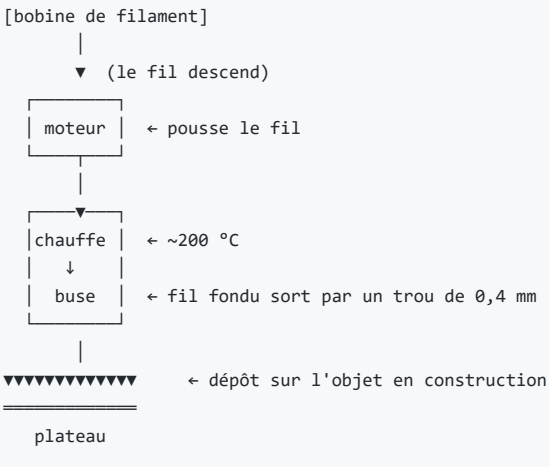
Partie 3 — Les types d'imprimantes 3D grand public

Il existe plusieurs technologies, mais deux dominent largement le grand public.

FDM — la plus répandue

FDM = *Fused Deposition Modeling* (dépôt de fil fondu). C'est la technologie de **95 %** des imprimantes 3D que tu croiseras à l'école, à la maison, au FabLab.

Principe : un **fil de plastique** est tiré depuis une bobine, poussé dans une tête chauffée (la **buse** ou GB ***nozzle***), fondu autour de 200 °C, et déposé sur le plateau couche par couche.



Avantages :

- Bon marché (machines à partir de 200-300 €, filament autour de 20 €/kg).
- Simple à utiliser et à entretenir.
- Bonne variété de filaments.

Limites :

- Les couches sont **visibles** à l'œil — surface jamais parfaitement lisse.
- Précision limitée (la buse fait 0,4 mm, on ne fera pas de bijou ultra détaillé avec).

Exemples de marques : Creality (Ender, K1), Prusa, Bambu Lab, Anycubic, Voron (à monter soi-même).

Résine (SLA / MSLA / DLP)

Une autre famille, plus chère mais plus précise.

Principe : un bac contient de la **résine liquide** photosensible. Une **lumière UV** (laser ou écran LCD) durcit la résine couche par couche. Le plateau remonte au fur et à mesure, tirant l'objet hors du bac.

Avantages :

- Précision **bien supérieure** au FDM (couches autour de 0,025 mm, soit 8× plus fines).
- Surfaces très lisses, idéales pour des figurines, des bijoux, des prototypes détaillés.

Limites :

- Plus cher : machine et résine.
- La résine liquide est **toxique** avant durcissement : gants, lunettes, ventilation obligatoires.
- Étapes de post-traitement : lavage à l'alcool, durcissement final aux UV.
- Inadapté aux grandes pièces mécaniques (résine plus cassante).

Exemples de marques : Anycubic Photon, Elegoo Mars, Phrozen, Formlabs.

Comparatif rapide

CRITÈRE	FDM	RÉSINE (SLA/MSLA)
Matériau	Fil plastique en bobine	Liquide photosensible
Précision typique	0,1–0,3 mm par couche	0,01–0,05 mm par couche
Coût machine	200–800 €	250–600 €
Coût matière (kg)	~20 €/kg	~50 €/kg
Vitesse	Moyenne à rapide	Moyenne
Sécurité	Buse chaude, mais peu de risque	Résine toxique : gants/ventilation
Idéal pour	Pièces fonctionnelles, prototypes, gros volumes	Figurines, bijoux, détails fins

À l'école, c'est presque toujours du **FDM**. C'est ce qu'on utilisera ici.

Partie 4 — Les principaux filaments FDM

Quand tu achètes une bobine, tu te retrouves face à une jungle de sigles. Voici les quatre filaments que tu vas croiser le plus souvent.

PLA — le filament de base

PLA = *Polylactic Acid* (acide polylactique). C'est le filament **par défaut** quand on débute. Fabriqué à partir d'amidon de maïs, il est en partie **biodégradable**.

- **Température d'impression :** ~200 °C.
- **Plateau :** chauffé à 50-60 °C (ou pas chauffé du tout sur certaines machines).
- **Rigidité :** bonne mais cassant — il se fend plus qu'il ne plie.
- **Tenue à la chaleur :** mauvaise — déforme à partir de 50-60 °C (donc à éviter pour un objet laissé dans une voiture en été).
- **Odeur à l'impression :** légère, plutôt agréable (un peu sucrée).

Idéal pour : prototypes, déco, jouets, objets d'usage normal à l'intérieur. **C'est ce qu'on utilise à l'école.**

PETG — le polyvalent

PETG = *Polyethylene Terephthalate Glycol*. Le grand cousin du plastique des bouteilles d'eau (PET) avec un G ajouté qui le rend imprimable.

- **Température d'impression :** ~230 °C.
- **Rigidité :** souple, moins cassant que le PLA.
- **Tenue à la chaleur :** meilleure que le PLA (jusqu'à ~70-80 °C).
- **Résistance à l'humidité :** bonne — utilisable en extérieur.

Idéal pour : pièces qui prennent des chocs (clip de fixation, bras articulé), contenants alimentaires (sous certaines conditions), objets exposés au soleil.

ABS — le technique

ABS = *Acrylonitrile Butadiene Styrene*. Le plastique des Lego, du tableau de bord de ta voiture, des coques d'électroménager.

- **Température d'impression :** ~240-260 °C.
- **Plateau :** chauffé à 100 °C minimum, **enceinte fermée** recommandée.
- **Rigidité :** très solide et résistant aux chocs.
- **Tenue à la chaleur :** excellente (jusqu'à ~100 °C).
- **Inconvénient majeur : odeur très désagréable et nocive** à l'impression. À utiliser uniquement dans une pièce ventilée ou avec une imprimante en enceinte fermée.

Idéal pour : pièces mécaniques, pièces exposées à la chaleur, pièces qui doivent durer.

TPU — le souple

TPU = *Thermoplastic Polyurethane*. Un filament **flexible** — l'objet imprimé se plie comme du caoutchouc.

- **Température d'impression :** ~220 °C.
- **Souplesse :** variable selon le grade (de la semelle de chaussure au pneu).
- **Difficulté :** délicat à imprimer (le fil souple a tendance à s'emmêler dans l'extrudeur).

Idéal pour : coques de téléphone, joints, semelles, jouets antichoc, pneus de petits véhicules.

Comparatif rapide

> 🧠 ***À retenir :*** *à l'école et pour 95 % de tes premiers projets, c'est **PLA**. Il pardonne tout, sent presque rien, et donne de beaux résultats. Tu changeras de filament quand un projet précis le demandera.*

FILAMENT	TEMPÉRATURE	SOLIDITÉ	CHALEUR	TOXICITÉ ODEUR	NIVEAU
PLA	200 °C	Cassant	Faible	Faible	Débutant
PETG	230 °C	Souple	Bonne	Faible	Intermédiaire
ABS	250 °C	Très bon	Excellent	Forte	Avancé
TPU	220 °C	Flexible	Moyenne	Faible	Avancé

Partie 5 — Quelques pièges classiques

Avant de te lancer, sache que l'impression 3D **rate parfois**. C'est normal. Les ratés les plus fréquents :

- **L'objet ne tient pas au plateau** → la première couche s'arrache. Cause : plateau pas propre, mal nivelé, ou trop loin de la buse. Solution : on règle le **nivellement** (manuel ou automatique) et on nettoie le plateau à l'alcool.
- **Spaghetti monstrueux** → l'imprimante continue d'extruder dans le vide après que l'objet s'est décollé. Toujours rester à proximité au début d'une impression.
- **Décalage de couches** → toutes les couches au-dessus d'un certain niveau sont décalées de quelques mm. Cause : un choc, ou une courroie détendue.
- **Surplombs ratés** (GB ***overhangs***) → des parties en l'air s'affaissent. Solution : ajouter des **supports** dans le slicer, ou orienter l'objet autrement avant d'imprimer.
- **Filament humide** → bruits crépitants à l'impression, surface granuleuse. Le filament absorbe l'humidité de l'air. Solution : garder les bobines dans une boîte fermée avec un sachet déshydratant.

Aucun de ces problèmes n'est dramatique : c'est l'apprentissage classique de la machine.

✍️ À retenir

- L'impression 3D grand public dépose la matière **couche par couche**, du bas vers le haut.
- Trois étapes : **modéliser** (TinkerCad → STL), **trancher** (slicer → G-code), **imprimer**.
- Le **slicer** est obligatoire : il traduit un modèle 3D en mouvements machine.
- **FDM** (fil fondu) domine le grand public — bon marché, polyvalent, surfaces visibles en couches.
- **Résine** (SLA/MSLA) est plus précise mais plus chère et plus dangereuse à manipuler.
- Les filaments courants : **PLA** (débutant, déco), **PETG** (extérieur, robuste), **ABS** (technique, chaleur), **TPU** (flexible).
- À l'école : **PLA**, presque toujours.

Suite

Tu as les concepts, tu sais modéliser et tu sais ce que fait une imprimante. Il est temps de tout mettre en œuvre : dans le prochain travail, tu vas **concevoir et faire imprimer ton premier objet**, dans un format compact qu'on peut produire en classe.

Projet : ton premier objet imprimé

> *Tu modélises et fais imprimer un petit objet de ton choix, qui doit tenir dans un cube de 4 × 4 × 4 cm. Objectif : aller du brouillon papier à l'objet physique, en passant par TinkerCad et le slicer.*

Tu as appris à modéliser dans TinkerCad et tu comprends comment fonctionne une imprimante 3D. Place à la pratique : tu vas **concevoir, modéliser et imprimer** un petit objet — quelque chose d'utile, de joli, ou de décoratif, mais qui rentre dans des dimensions imposées.

🎯 Objectifs

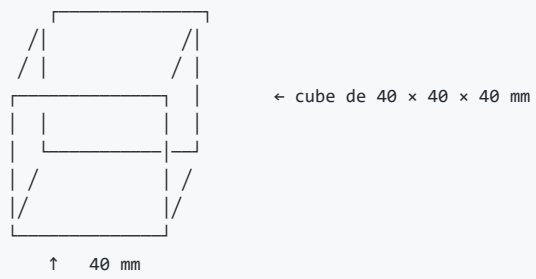
À la fin de ce projet, tu seras capable de :

1. **Concevoir** un objet en partant d'une idée (croquis papier, mesures, contraintes).
2. **Modéliser** cet objet dans TinkerCad en respectant des dimensions précises.
3. **Préparer** ton fichier pour l'impression (export STL, vérification).
4. **Présenter** ton objet imprimé et expliquer tes choix de conception.

📋 Contraintes du projet

Dimensions

Ton objet doit tenir **entièrement** dans un cube imaginaire de **4 × 4 × 4 cm** (donc 40 × 40 × 40 mm dans TinkerCad).



Pas de pièce qui dépasse, même un peu. La raison est très concrète : on imprime plusieurs projets en même temps, et chacun doit rester dans son carré.

Temps d'impression

Pour que tout le monde passe, ton objet doit **s'imprimer en moins d'une heure** sur l'imprimante de la classe. Le slicer t'annonce ce temps quand tu ouvres ton fichier — il faudra le vérifier avant la mise en file.

Si l'estimation dépasse l'heure, tu peux jouer sur :

- **Baisser le remplissage** (GB ***infill***) — par exemple 10 % au lieu de 20 %.
- **Augmenter l'épaisseur de couches** — passer de 0,2 mm à 0,25 ou 0,3 mm.
- **Simplifier ton modèle** — supprimer un détail qui prend du temps sans rien apporter.

Filament

Filament imposé : **PLA**. Tu choisis la couleur parmi ce qui est en stock le jour de l'impression.

🧠 Quelques pistes d'objets

Tu choisis librement, mais voici des idées pour démarrer si tu manques d'inspiration :

> 💡 ***Astuce :*** *un bon projet de débutant a **peu de surplombs** (peu de parties en l'air qui auraient besoin de supports). Plus la pièce ressemble à une "pyramide vue d'en haut" (large en bas, plus fin en haut), plus elle s'imprime sans soucis.*

IDÉE	POURQUOI C'EST INTÉRESSANT
Porte-clés personnalisé	Tu reprends le cours TinkerCad et tu le personnalises.
Pion de jeu de société	Cylindre, sphère, perçage — toutes les bases.
Mini-pot pour cactus	Cylindre + perçage intérieur — utile et joli.
Support pour téléphone (mini)	Plus difficile : géométrie en équilibre, angles.
Pièce de réparation	Un clip cassé, un crochet, une cale — résoudre un vrai problème.
Dé personnalisé	6 faces gravées au choix (initiales, logos, symboles).
Médaillon ou pendentif	Plat, joli, traversé par un anneau.
Logo en relief	Le logo de ta série, de l'école, d'un groupe — utilisable comme aimant de frigo.
Range-câbles	Petit clip qui se fixe sur un bureau pour tenir un câble USB.

📝 Étapes du projet

Étape 1 — Croquis papier (10 min)

Avant de toucher à TinkerCad, dessine ton idée **sur papier**.

- Trois vues : **dessus**, **face**, **côté**.
- Note les **dimensions principales** (en mm) à côté de chaque côté important.
- Vérifie que tout rentre dans 40 × 40 × 40 mm.

Cette étape semble inutile — elle ne l'est pas. Modéliser sans plan, c'est passer 30 minutes à bricoler des cubes en se demandant à quoi on voulait arriver. Avec un croquis devant les yeux, la modélisation prend 10 minutes.

Étape 2 — Modélisation dans TinkerCad

Crée ton modèle en suivant ton croquis. Pendant le travail :

- Active la **règle** dans TinkerCad (icône Règle, en haut à droite) pour vérifier tes dimensions à tout moment.
- Utilise systématiquement l'outil **Aligner** pour que rien ne soit "à peu près" centré.
- Regroupe les éléments par étapes — pas tout d'un coup à la fin.

Quand tu penses avoir fini, **fais tourner ta vue** sous tous les angles. Cherche les détails qui ne vont pas : une face qui dépasse, un trou bouché, un texte mal aligné…

Étape 3 — Export et vérification

1. Sélectionne **tout** ton objet (Ctrl + A).
2. **Exporter** → STL → "Tout dans la conception".
3. Nomme le fichier : `<TonNom>-<NomObjet>.stl` (ex. `lambrechts-porte-cles.stl`).

Avant de le rendre, fais une **dernière vérification** : ouvre le STL dans le slicer de la classe (Cura, PrusaSlicer, ou autre).

- Le **temps d'impression** est-il < 1 heure ?
- L'objet **tient-il bien** sur le plateau dans le bon sens ?
- Y a-t-il besoin de **supports** ?

Si quelque chose cloche, retourne dans TinkerCad et corrige.

Étape 4 — Impression

Une fois la file d'attente arrivée à ton tour, le prof lance l'impression. Pendant ce temps, tu observes — c'est instructif. Regarde particulièrement :

- La **première couche** : c'est elle qui décide si toute l'impression va réussir.
- Les **mouvements de la tête** sur les premiers étages : comment la pièce prend forme.

Étape 5 — Présentation

Quand ton objet est imprimé et nettoyé (supports cassés s'il y en a, fil parasite enlevé), tu le présentes en classe en répondant à ces questions :

1. **À quoi sert ton objet ?** (ou : qu'est-ce qu'il représente ?)
2. **Quelle a été la partie la plus difficile à modéliser ?** Comment tu l'as résolue ?
3. **Qu'est-ce que tu changerais** si tu devais le refaire ?
4. **Combien de temps** a duré l'impression ? Avec quelle épaisseur de couche et quel remplissage ?

✅ Critères de réussite

Ton projet est validé si :

- L'objet **tient** dans 40 × 40 × 40 mm.
- L'impression **prend moins d'une heure**.
- Le fichier STL est **propre** (un seul objet bien défini, pas de bouts en l'air ou de faces qui se chevauchent bizarrement).
- L'objet imprimé est **identifiable** — on reconnaît clairement ce que tu voulais faire.
- Tu peux **expliquer tes choix** lors de la présentation.

Grille d'évaluation détaillée

🏆 Pour aller plus loin

Si tu as fini avant les autres :

- **Personnaliser** : ajoute du texte, un logo, un détail qui te ressemble.
- **Imprimer en plusieurs couleurs** : prévois deux pièces séparées (par exemple un anneau et un texte) qu'on peut imprimer dans deux couleurs et emboîter.

CRITÈRE	POINTS
Croquis papier clair, avec cotes	/3
Respect des dimensions (40 × 40 × 40 mm max)	/3
Qualité de la modélisation (alignement, propreté)	/4
Originalité / pertinence du choix d'objet	/3
Export STL correct et imprimable	/3
Présentation orale (clarté, recul sur le travail)	/4
Total	**/20**

- **Optimiser** : essaie de descendre sous **30 minutes** d'impression sans sacrifier la solidité.
- **Tester une variante** : refais ton objet avec une autre orientation sur le plateau — la qualité change-t-elle ?
- **Préparer un second objet** différent du premier (autre forme, autre usage). Le meilleur des deux sera évalué.

⚠️ Quelques rappels avant de te lancer

⚠️ ***L'imprimante chauffe à 200 °C.*** *Ne touche **jamais** la buse pendant ou après une impression. Le plateau peut aussi être chaud — attends qu'il refroidisse avant de décoller ta pièce.*

⚠️ ***Reste réaliste sur la durée.*** *Si ton premier slicing annonce 4 h, tu n'as pas une heure de marge — tu as 3 h de trop. Simplifie maintenant, pas le jour J.*

> 💡 ***Le STL ne sait pas mentir.*** *Si l'objet rate dans le slicer (faces fantômes, "non-manifold"), c'est qu'il y a un problème dans TinkerCad : deux solides qui se touchent juste à un point, un perçage qui ne traverse pas, etc. Repasse en revue les regroupements.*

Suite

Une fois ton premier objet en main, tu auras vraiment fait le tour : du dessin papier au plastique fini. Pour la suite, tu pourras t'attaquer à des objets plus complexes — assemblages de plusieurs pièces, mécaniques (charnières, engrenages), ou pièces fonctionnelles qui remplacent un objet cassé chez toi.



GESTION DE PROJETS

Introduction à la Gestion de Projet

La **gestion de projet** est l'ensemble des méthodes, outils et pratiques permettant de **planifier, organiser et piloter un projet** afin d'atteindre des objectifs précis dans un délai et avec des ressources définies. Un projet se distingue par son caractère **temporaire** (il a un début et une fin) et par sa **finalité unique** (il vise à produire un livrable spécifique, comme un logiciel, une application ou un service).

En informatique, la gestion de projet est essentielle pour **structurer le développement de logiciels, la mise en place d'infrastructures ou la conception de systèmes complexes**. Sans une gestion efficace, les projets risquent d'être mal organisés, de dépasser les délais ou le budget, ou encore de ne pas répondre aux attentes des utilisateurs. Que ce soit avec des approches **classiques** (en cascade) ou **agiles**, bien gérer un projet informatique permet d'optimiser le travail des équipes, d'anticiper les risques et d'assurer la réussite du produit final. 🚀

Le carré des contraintes

> *Au-delà du triangle d'or : le modèle à quatre sommets de Jurgen Appelo pour piloter un projet.*

Pourquoi ce cours ?

Dans n'importe quel projet en équipe, tu vas vivre la même tension : trop à faire, pas assez de temps, une équipe imparfaite, et un client (ou un prof, ou un public) qui veut quand même un résultat de qualité.

Ce cours te donne un **outil mental** pour rendre ces tensions visibles, en parler, et faire des choix lucides. Cet outil s'appelle le **carré des contraintes**, proposé par Jurgen Appelo dans *Management 3.0* (2010) en évolution du classique **triangle d'or**.

Principe fondamental

> ***Si un sommet varie dans un sens, soit les 2 sommets adjacents varient dans le même sens, soit le sommet opposé varie dans le sens inverse.***

C'est la règle unique qui régit toute la mécanique du modèle. Le reste du cours en dérive.

Objectifs d'apprentissage

À la fin de cette séquence, tu seras capable de :

1. **Identifier** les quatre sommets du carré des contraintes et donner leur équivalent en anglais et en français.
2. **Expliquer** le principe fondamental d'arbitrage : adjacents même sens / opposé sens inverse.
3. **Décomposer** la notion de ressources entre les outils et la capacité humaine (formule à quatre facteurs).
4. **Analyser** un projet en identifiant le sommet modifié et les sommets compensateurs.
5. **Argumenter** un choix d'arbitrage dans une réunion d'équipe agile en s'appuyant sur le carré.

Cinq notions-clés à retenir

#	NOTION	IDÉE CENTRALE
1	**Triangle d'or**	Le modèle classique : Scope, Time, Cost. Critiqué par Appelo car la qualité y est implicite.
2	**Carré des contraintes**	Quatre sommets explicites : **Scope, Time, Quality, Resources**.
3	**Adjacents vs opposé**	Si je modifie un sommet, soit les deux adjacents bougent dans le même sens, soit l'opposé bouge en sens inverse.
4	**Ressources = outils × humains**	Une équipe sans outils ne produit pas, des outils sans équipe non plus. La capacité humaine, elle, suit la formule `N × C × M × T` — multiplicative.
5	**Pas d'arbitrage gratuit**	Aucun projet ne maximise les quatre sommets en même temps. Le rôle du chef de projet est de rendre les arbitrages **explicites et négociés**.

1. Du triangle au carré

Le triangle d'or classique (Iron Triangle)

Depuis les années 1960, la gestion de projet se résume souvent à trois contraintes :

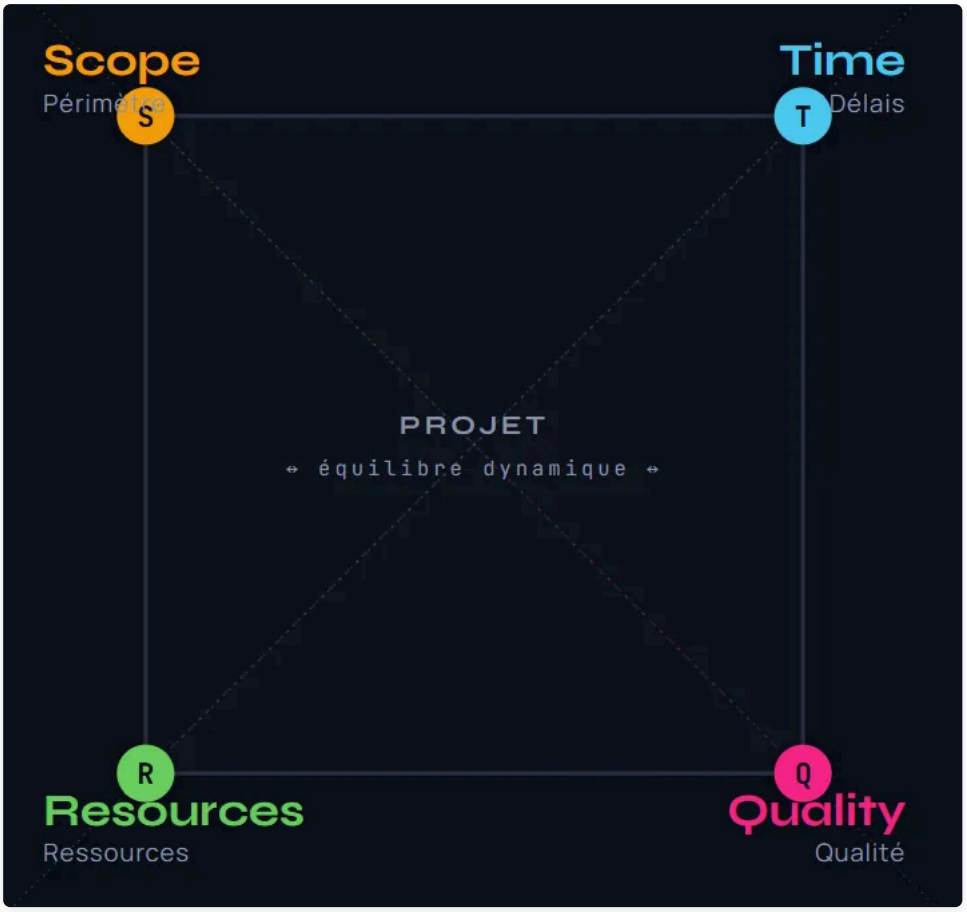
Scope (périmètre) — ce qu'on doit livrer
Time (délai) — combien de temps on a
Cost (coût) — combien ça coûte

> *« Vite, pas cher, beaucoup de fonctionnalités — choisis-en deux. »*

C'est élégant, mais Appelo trouve que **la qualité y est traitée comme un sous-produit silencieux**. Elle se dégrade quand on serre les trois autres, mais personne n'en parle ouvertement.

Le carré d'Appelo

Appelo propose d'ajouter explicitement la **Qualité** comme quatrième sommet :



Voisinages :

SOMMET	ADJACENTS (CÔTÉ DU CARRÉ)	OPPOSÉ (DIAGONALE)
Scope	Time, Resources	Quality
Time	Scope, Quality	Resources
Quality	Time, Resources	Scope
Resources	Scope, Quality	Time

Cette topologie est essentielle : c'est elle qui détermine *qui* doit bouger *quand* tu modifies un sommet.

---

2. Les quatre sommets en détail

↖ Sommet haut-gauche : Scope *(Périmètre)*

Définition. L'ensemble des fonctionnalités, livrables et exigences qui définissent **ce que** le projet doit produire. C'est la *frontière* du projet : ce qui est inclus, ce qui est exclu.

Vocabulaire.

- GB *Scope, requirements, features, deliverables*
- FR Périmètre, exigences, fonctionnalités, livrables

Exemple. Un site de réservation avec catalogue, calendrier, paiement en ligne, comptes utilisateurs, scan QR et back-office → **scope large**. Le même site avec uniquement le catalogue et le calendrier → **scope réduit**.

Piège classique. Le **scope creep** : le périmètre grossit en douce au fil du projet (« on pourrait juste ajouter… »). Sans arbitrage, c'est le délai, la qualité ou les ressources qui paient l'addition.

↗ Sommet haut-droite : Time *(Délais)*

Définition. La durée totale disponible pour livrer le projet. Inclut la **date de fin** (deadline), les **jalons** intermédiaires, et le **rythme** de travail.

Vocabulaire.

- GB *Time, schedule, deadline, milestones*
- FR Délais, temps imparti, échéance, jalons

Exemple. Une soirée d'entreprise est prévue le 14 février. Cette date **ne bouge pas**. Si le client demande une nouvelle catégorie de questions trois semaines avant, c'est le scope ou la qualité qui devra s'ajuster — jamais le délai.

Particularité. Le temps est souvent **la contrainte la plus rigide** : un salon, une rentrée scolaire, une obligation légale, un lancement marketing — ces dates sont externes au projet et ne se négocient pas.

↘ Sommet bas-droite : Quality *(Qualité)*

Définition. Le niveau d'exigence du livrable : robustesse du code, ergonomie, accessibilité, performance, sécurité, design, absence de bugs, conformité aux normes.

Vocabulaire.

- GB *Quality, robustness, usability, performance*
- FR Qualité, robustesse, ergonomie, performance

Exemple. Une app médicale doit être validée **sans bug critique** — la qualité est imposée par la loi. Un prototype présenté en interne peut tolérer des bugs mineurs : on ajuste après.

Le piège de la qualité sacrifiée. Quand on dit *« on livrera vite, on corrigera après »*, on crée de la **dette technique** : du code mal écrit qu'il faudra payer (avec intérêts) plus tard. Appelo insiste : rendre la qualité explicite, c'est la rendre négociable **en conscience**, pas par défaut.

↙ Sommet bas-gauche : Resources *(Ressources)*

Définition. Tout ce qui est mobilisé pour réaliser le projet : équipe humaine, outils, matériel, budget, locaux.

Vocabulaire.

- GB *Resources, people, tools, capacity, budget*
- FR Ressources, équipe, outils, capacité, budget

C'est le sommet **le plus mal compris** — et c'est sur lui qu'on va passer un peu plus de temps.

---

3. Zoom — Décomposer les ressources

> ⚠️ ***Note importante.*** *Ce qui suit est un **enrichissement personnel** du modèle d'Appelo, **pas la version d'origine**. Dans Management 3.0, Appelo traite les ressources de manière assez globale — équipe et budget réunis. La décomposition ci-dessous est une grille de lecture plus fine pour mieux piloter ce qu'il y a derrière la simple étiquette « ressources ».*

Deux grandes familles

Pour un projet, les ressources se regroupent en deux grandes familles :

1. **Les outils** (`O`) — tout ce qui n'est pas humain mais qui permet de produire : matériel (ordinateurs, serveurs, hardware spécialisé), logiciels (IDE, suites de design, outils de build), infrastructure (cloud, CI/CD, bases de données), locaux, licences, accès aux specs et à la documentation, budget.
2. **La capacité humaine** (`H`) — ce que l'équipe peut réellement produire, qui dépend de quatre facteurs détaillés ci-dessous.

Les deux dimensions se combinent de manière **multiplicative** :

> $R = O \times H$

Pourquoi multiplicatif ? Parce qu'une équipe top sans serveur, sans IDE, sans accès aux specs → bloquée. Et des outils flambant neufs sans personne pour les utiliser → rien ne sort. Si l'une des deux dimensions tend vers zéro, l'ensemble tend vers zéro.

La capacité humaine : quatre facteurs

> $H = N \times C \times M \times T$

Où :

- **N — Nombre de personnes** dans l'équipe (de 1 à n)
- **C — Compétence** : ce qu'elles savent faire dans le domaine du projet
- **M — Motivation** : leur envie réelle d'avancer sur *ce* projet
- **T — Temps disponible** : la part de leur temps de travail qu'elles peuvent vraiment y consacrer

⚠️ Une remarque essentielle sur les pourcentages

Dans les exemples qui suivent, on utilise des pourcentages (« compétence à 95 % », « motivation à 0 % »…). **Ces chiffres ne sont pas des mesures absolues.** Personne ne calcule scientifiquement « 78 % de motivation » ou « 42 % de compétence » — c'est impossible.

Les pourcentages sont là **uniquement pour faciliter la compréhension** : ils permettent de visualiser intuitivement les ordres de grandeur et l'effet multiplicatif. En pratique, trois niveaux suffisent souvent — *élevé / moyen / faible* — voire juste la question *« est-ce qu'il manque quelque chose ? »*.

L'important n'est **pas la précision du chiffre**. C'est de reconnaître que **tous les facteurs jouent en même temps**, et que **tout facteur très bas fait chuter l'ensemble**.

Pourquoi c'est *multiplicatif* et pas *additif*

C'est la clé du modèle. Une multiplication a une propriété particulière : **si un seul facteur est à zéro, le résultat est zéro**, peu importe les autres.

Regardons quatre cas pour s'en convaincre :

CAS 1 — LE DÉVELOPPEUR EXPERT MAIS DÉMOTIVÉ

> *1 personne · compétence très élevée ·* ***motivation nulle*** *· temps total disponible →* `H ≈ 1 × 0,95 × 0 × 1 = 0`

Le projet n'avance pas. Aucune productivité. La compétence sans motivation = zéro. C'est **la grande leçon d'Appelo** : la motivation n'est pas un « bonus » qu'on ajoute, c'est un **facteur multiplicatif**. Sans elle, tout le reste est neutralisé.

CAS 2 — L'ÉQUIPE MOTIVÉE MAIS SANS COMPÉTENCE

> *5 personnes ·* ***compétence très faible*** *· motivation maximale · temps total disponible →* `H ≈ 5 × 0,05 × 1 × 1 = 0,25`

L'enthousiasme ne remplace pas le savoir-faire. Une équipe ultra-motivée mais qui découvre une techno la veille du sprint ne livrera pas une plateforme robuste.

CAS 3 — L'EXPERT SANS TEMPS DISPONIBLE

> *1 personne · compétence maximale · motivation maximale ·* ***temps disponible très faible*** *(autres projets prioritaires) →* `H ≈ 1 × 1 × 1 × 0,05 = 0,05`

Un excellent profil à quelques % de son temps fournit la même chose qu'un débutant à quelques % : presque rien. Le temps disponible est crucial — et souvent surestimé.

CAS 4 — L'ÉQUIPE ÉQUILIBRÉE

> *5 personnes · compétence assez bonne · motivation bonne · temps partiel →* `H ≈ 5 × 0,7 × 0,8 × 0,6 ≈ 1,7`

Soit environ **17 %** d'un maximum théorique de 10 personnes à plein régime sur tous les facteurs. C'est une équipe **correcte et réaliste** : on ne travaille jamais à fond sur tous les axes en même temps.

▸ Comment évaluer ta propre équipe

Avant chaque sprint, pose-toi (en équipe) les quatre questions :

1. **N —** Sommes-nous combien à vraiment travailler dessus ? (Distinguer les présents des actifs.)
2. **C —** Maîtrisons-nous les technos nécessaires, ou faut-il apprendre en chemin ?
3. **M —** Est-ce que tout le monde *a envie* d'avancer ? Y a-t-il un sujet bloquant ?
4. **T —** Combien d'heures réelles peut-on y consacrer cette semaine ?

Et n'oublie pas la première moitié de la formule : **a-t-on les outils nécessaires ?** Sans accès au serveur de test, sans la bonne licence logiciel, sans la doc à jour, même la meilleure équipe sera bloquée.

Si l'un des facteurs (outils inclus) est très bas → c'est lui qu'il faut adresser **en priorité** avant d'ajouter du scope.

---

4. La mécanique d'arbitrage

C'est le **cœur du modèle**. Pour chaque situation, on applique le principe fondamental : **soit** les deux adjacents bougent dans le même sens (vert), **soit** l'opposé bouge en sens inverse (rouge).

▸ Situation A — Le périmètre augmente (Scope ↑)

> *Un client ajoute une fonctionnalité en cours de projet.*

Soit les deux adjacents augmentent :

- ↑ **Time** — il faut plus de temps
- ↑ **Resources** — il faut plus de ressources

Soit l'opposé diminue :

- ↓ **Quality** — la qualité baisse en compensation

▸ Situation B — Le délai se raccourcit (Time ↓)

> *La direction avance la deadline de trois semaines.*

Soit les deux adjacents diminuent :

- ↓ **Scope** — on réduit les fonctionnalités
- ↓ **Quality** — on accepte une qualité moindre

Soit l'opposé augmente :

- ↑ **Resources** — on ajoute des renforts

Situation C — Les ressources diminuent (Resources ↓)

> *Un développeur quitte l'équipe à mi-projet.*

🟢 **Soit les deux adjacents diminuent :**

- ↓ **Scope** — on coupe des features
- ↓ **Quality** — la qualité chute

🔴 **Soit l'opposé augmente :**

- ↑ **Time** — on allonge le délai

Situation D — La qualité doit monter (Quality ↑)

> *Audit qualité imposé, ou cible plus exigeante.*

🟢 **Soit les deux adjacents augmentent :**

- ↑ **Time** — il faut plus de temps pour peaufiner
- ↑ **Resources** — plus de relecture, plus de tests

🔴 **Soit l'opposé diminue :**

- ↓ **Scope** — on livre moins, mais mieux

Le rôle du chef d'équipe

À chaque demande de changement, **trois questions à se poser à voix haute** :

1. Quel sommet veut bouger, et dans quel sens ?
2. Qui sont les adjacents, qui est l'opposé ?
3. Quel scénario de compensation choisit-on : **adjacents même sens** ou **opposé sens inverse** ?

L'erreur classique : *« on garde tout pareil et on encaisse ».* Ça n'existe pas. Le carré rend l'arbitrage **explicite** au lieu de le subir en silence.

---

5. Lien avec l'écoresponsabilité numérique

Un sommet souvent oublié mais qui devrait être discuté : **toute fonctionnalité ajoutée a un coût environnemental**.

- Plus de scope → plus de code, plus de serveurs, plus de stockage, plus de bande passante consommée.
- Plus de ressources humaines → plus de déplacements, de hardware, de chauffage de bureaux.
- Plus de qualité (notamment perf) → souvent moins de gaspillage énergétique côté utilisateur.

Quand tu négocies le scope en équipe, pose la question : **cette fonctionnalité justifie-t-elle son empreinte ?** Le carré devient alors aussi un outil d'arbitrage éthique. C'est la posture **Lean & Green** de la gestion de projet moderne.

---

6. Exercices

Exercice 1 — Application du principe

Pour chaque scénario, indique les **deux scénarios de compensation possibles** selon le principe fondamental :

1. La qualité doit augmenter (audit imposé).
2. Le scope doit diminuer (le client coupe une feature).
3. Les ressources doivent augmenter (recrutement d'un dev).
4. Le délai s'allonge (livraison repoussée).

Réponse attendue par scénario :

- 🟢 **Adjacents :** quels sommets et dans quel sens ?
- 🔴 **Opposé :** quel sommet et dans quel sens ?

Exercice 2 — Diagnostic d'équipe

Évalue la capacité réelle de ton équipe agile pour la semaine prochaine. Pour chaque facteur, contente-toi d'un niveau *élevé / moyen / faible* (pas besoin de pourcentages précis) :

- **Outils** disponibles (matériel, logiciels, accès aux specs)
- **N** — nombre de personnes vraiment actives
- **C** — compétence collective sur les technos requises

- **M** — motivation
- **T** — temps réellement disponible

→ Identifie le **maillon faible** (le facteur le plus bas) et propose **une action concrète** pour le faire remonter.

Exercice 3 — Négociation

Le client demande une fonctionnalité supplémentaire à 10 jours de la livraison. Rédige un message Slack ou e-mail expliquant **les deux options possibles** (adjacents même sens / opposé sens inverse) et **ta recommandation argumentée**.

Exercice 4 — Étude de cas

Choisis un projet réel que tu connais (un projet open source, le développement d'un jeu vidéo connu, un grand chantier public…). Analyse-le à travers le carré :

- Quel sommet a été **préservé à tout prix** ?
- Quel sommet a été **sacrifié** ?
- Quel scénario d'arbitrage a été appliqué (adjacents même sens, ou opposé sens inverse) ?

Pour aller plus loin

- **Jurgen Appelo**, *Management 3.0 — Leading Agile Developers, Developing Agile Leaders*, Addison-Wesley, 2010 (chap. 1 & 4).
- **Jurgen Appelo**, *Managing for Happiness*, Wiley, 2016 (sur la motivation comme facteur multiplicatif).
- **The Mythical Man-Month**, Fred Brooks, 1975 (la fameuse loi : *« ajouter des développeurs à un projet en retard le retarde davantage »* — illustration du modèle multiplicatif).
- **The Phoenix Project**, Gene Kim et al. (sur la dette technique et l'arbitrage qualité dans un contexte DevOps).

« Le triangle vous dit ce que vous pouvez optimiser. Le carré vous rappelle ce que vous sacrifiez. »