Python: Afficher un menu de sélection

Un menu dans une console, c'est l'application la plus fréquente du combo *liste + boucle for*. On part d'un menu écrit à la main, on le remplace par un menu généré depuis une liste, puis on lui ajoute des numéros automatiques avec `enumerate()`. Et à la fin, l'utilisateur peut vraiment choisir.

5GMS 5TTR 6TTR 3TTR Exploration

*Presque toutes les applications console montrent un menu à un moment ou l'autre : un jeu (« Nouvelle partie / Charger / Quitter »), une appli de quiz (« Choisis ta catégorie »), un script utilitaire (« Que veux-tu faire ? »). Ce cours montre la **bonne façon** de l'écrire — et surtout, comment passer du « ça marche pour 3 options » au « ça marche pour 30 ».*

🎯 Objectifs

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

1. Écrire un menu console avec un titre, des options et un choix utilisateur.
2. Reconnaître pourquoi le menu « écrit à la main » devient ingérable au-delà de 3-4 options.
3. Générer un menu **depuis une liste**, avec une boucle `for`.
4. Ajouter une **numérotation automatique** grâce à `enumerate()`.
5. Récupérer le choix de l'utilisateur et l'utiliser pour accéder à l'option sélectionnée.

Partie 1 — Version 1 : le menu écrit à la main

Imagine que tu codes un mini-zoo et que tu dois afficher la liste des animaux disponibles pour que l'utilisateur en choisisse un. Première approche, naïve, mais qui marche :

```
print("Choisis un animal :")
print("Chat")
print("Lapin")
print("Capibara")
```

Résultat :

```
Choisis un animal :
Chat
Lapin
Capibara
```

Ça marche pour 3 animaux. Mais regarde ce qui cloche déjà :

- **Pas de numéros** : comment l'utilisateur exprime son choix ? Il doit taper le mot exact ? Avec ou sans accent ?
- **Pas évolutif** : si on ajoute un animal, il faut ajouter une ligne `print`.
- **Couplage fort** : le menu est figé dans le code, on ne peut pas le générer depuis un fichier ou une saisie.

> 🧠 **LE SYMPTÔME** : plusieurs `print` qui se ressemblent comme deux gouttes d'eau. C'est exactement le signal qui dit *« remplace-moi par une boucle ! »*.

Partie 2 — Version 2 : le menu via une liste + `for`

Première amélioration : ranger les animaux dans une **liste** et les afficher avec une boucle.

```python
# La liste des animaux du zoo
animaux = ["Chat", "Lapin", "Capibara"]

print("Choisis un animal :")

# La boucle affiche un animal par ligne
for animal in animaux:
    print(animal)
```

Résultat identique à la version 1 :

```
Choisis un animal :
Chat
Lapin
Capibara
```

À première vue, c'est plus long que la version 1. **Mais** :

- **Pour ajouter un animal**, il suffit de l'ajouter à la liste : `animaux = ["Chat", "Lapin", "Capibara", "Lion"]`. La boucle l'affichera tout seul.
- **Pour 30 animaux**, ce code fait toujours 5 lignes.
- **Si la liste vient d'ailleurs** (un fichier, une base de données, une saisie utilisateur), seul `animaux = …` change. Le menu reste le même.

> 💡 **CONVENTION** rappelée du cours précédent : la liste est au **pluriel** (`animaux`), la variable de boucle au **singulier** (`animal`). Tu peux lire le code à voix haute : *« pour chaque animal dans animaux, affiche-le ».*

C'est mieux, mais l'utilisateur ne sait toujours pas quoi taper pour choisir. Il faut un **numéro** devant chaque option.

Partie 3 — Version 3 : numéroter avec `enumerate()`

Ce qu'on veut afficher maintenant :

```
Choisis un animal :
1. Chat
2. Lapin
3. Capibara
```

Pour avoir un numéro qui s'incrémente automatiquement à chaque tour, Python fournit la fonction `enumerate()`. Elle prend une liste et te donne, à chaque tour, **deux valeurs** : l'index et l'élément.

```python
animaux = ["Chat", "Lapin", "Capibara"]

print("Choisis un animal :")

for (i, animal) in enumerate(animaux):
    print(f"{i}. {animal}")
```

Résultat :

```
Choisis un animal :
0. Chat
1. Lapin
2. Capibara
```

Tu remarques quelque chose : la numérotation commence à **0**. C'est parce que Python compte à partir de 0 (l'index d'un élément dans une liste). Pour un humain, on préfère commencer à 1.

Décortiquons `enumerate()`

```python
for (i, animal) in enumerate(animaux):
#    └──────┘    └──────────┘
#  deux variables    enumerate produit
#  par tour          des paires (index, élément)
```

À chaque tour de boucle, `enumerate` donne une **paire** :

Tour	`i`	`animal`
1	0	`"Chat"`
2	1	`"Lapin"`
3	2	`"Capibara"`

> ℹ️ **SANS `ENUMERATE()`** tu aurais dû gérer un compteur à la main : `i = 0; for animal in animaux: print(i, animal); i += 1`. C'est plus long, plus fragile, et personne ne fait ça en Python — `enumerate` est l'outil dédié.

Commencer à 1 au lieu de 0

Deux possibilités :

Solution A — Ajouter 1 dans le `print`

```
for (i, animal) in enumerate(animaux):
    print(f"{i + 1}. {animal}")
```

Solution B — Le paramètre `start` de `enumerate()`

```
for (i, animal) in enumerate(animaux, start=1):
    print(f"{i}. {animal}")
```

Les deux donnent le même résultat. La solution B est légèrement plus propre — `start=1` exprime clairement l'intention.

```
Choisis un animal :
1. Chat
2. Lapin
3. Capibara
```

Partie 4 — Récupérer le choix de l'utilisateur

Le menu est joli, mais à quoi sert-il si l'utilisateur ne peut pas répondre ? Ajoutons la **saisie**.

```
animaux = ["Chat", "Lapin", "Capibara"]

print("Choisis un animal :")
```

```
4
5  for (i, animal) in enumerate(animaux, start=1):
6      print(f"{i}. {animal}")
7
8  # Récupère la réponse, convertit en entier
9  choix = int(input("Ton choix ? "))
10
11 # Retrouve l'animal correspondant (attention : -1 car les listes commencent à 0)
12 animal_choisi = animaux[choix - 1]
13
14 print(f"Tu as choisi : {animal_choisi}")
```

Déroulement à l'exécution :

```
1  Choisis un animal :
2  1. Chat
3  2. Lapin
4  3. Capibara
5  Ton choix ? 2
6  Tu as choisi : Lapin
```

> ⚠️ **POURQUOI `choix - 1` ?** L'utilisateur tape `2` pour le deuxième animal. Mais dans la liste, le deuxième animal est à l'index `1` (les listes commencent à 0). D'où le `-1`. Ne te trompe pas dans ce petit décalage, c'est la source d'erreur n°1 sur les menus.

Cas problématique : que se passe-t-il si l'utilisateur tape n'importe quoi ?

- Il tape `0` → `animaux[-1]` → renvoie le **dernier** animal (`Capibara`). Comportement bizarre mais pas plantage.
- Il tape `99` → `animaux[98]` → **plantage** (`IndexError`).
- Il tape `chien` → `int("chien")` → **plantage** (`ValueError`).

Pour gérer ces cas proprement, il faut ajouter une **validation** — c'est typiquement le rôle d'une boucle `while` qui redemande tant que la réponse est invalide. On voit ça dans le cours sur les boucles `while` avec validation.

Partie 5 — Le programme complet, commenté

Voici la version finale, avec commentaires pédagogiques :

```
1  # --- Données du menu (modifie cette liste pour changer le menu) ---
2  animaux = ["Chat", "Lapin", "Capibara", "Lion", "Dragon de Komodo"]
```

```python

# --- Affichage du menu ---
print("=== ZOO DU LYCÉE ===")
print("Choisis un animal à observer :")
print()  # ligne vide

for (i, animal) in enumerate(animaux, start=1):
    print(f"  {i}. {animal}")

print()

# --- Saisie du choix ---
choix = int(input("Ton choix ? "))

# --- Traitement du choix ---
animal_choisi = animaux[choix - 1]
print(f"\n→ Tu pars observer un {animal_choisi}. Bonne visite !")
```

Exécution :

```
=== ZOO DU LYCÉE ===
Choisis un animal à observer :

  1. Chat
  2. Lapin
  3. Capibara
  4. Lion
  5. Dragon de Komodo

Ton choix ? 3

→ Tu pars observer un Capibara. Bonne visite !
```

Note que pour **ajouter un animal**, il suffit de l'ajouter à la liste `animaux`. Tout le reste du programme s'adapte tout seul. **C'est ça, l'élégance d'un menu géré par boucle.**

🧪 Exercices

Exercice guidé — Le menu du restaurant

Énoncé. Écris un programme qui :

1. Affiche le menu d'un restaurant avec ces plats : `["Lasagnes", "Salade César", "Burger", "Pizza margherita", "Tarte aux fruits"]`
2. Numérote chaque plat à partir de 1.
3. Demande à l'utilisateur son choix.
4. Affiche un message du type : `→ Commande validée : Pizza margherita.`

Étape 1 — Réfléchir au plan

Ce qu'il faut :

- Une **liste** des plats.
- Une **boucle for + enumerate** pour afficher avec numéros.
- Un **input** pour le choix.
- Un accès à la liste avec `[choix - 1]` pour récupérer le plat.

Étape 2 — Le squelette

```
plats = ["Lasagnes", "Salade César", "Burger", "Pizza margherita", "Tarte aux fruits"]

print("Menu du restaurant :")
# TODO : boucle d'affichage

# TODO : saisie du choix

# TODO : affichage de la commande
```

Étape 3 — La boucle d'affichage

```
for (i, plat) in enumerate(plats, start=1):
    print(f"  {i}. {plat}")
```

(Convention pluriel/singulier : `plats` → `plat`.)

Étape 4 — La saisie

```
choix = int(input("\nQuel plat veux-tu ? "))
```

Étape 5 — La validation finale

```
plat_choisi = plats[choix - 1]
print(f"→ Commande validée : {plat_choisi}.")
```

Étape 6 — Le programme complet

```
plats = ["Lasagnes", "Salade César", "Burger", "Pizza margherita", "Tarte aux fruits"]

print("Menu du restaurant :")

for (i, plat) in enumerate(plats, start=1):
    print(f"  {i}. {plat}")

choix = int(input("\nQuel plat veux-tu ? "))
plat_choisi = plats[choix - 1]

print(f"→ Commande validée : {plat_choisi}.")
```

Exécution :

```
Menu du restaurant :
  1. Lasagnes
  2. Salade César
  3. Burger
  4. Pizza margherita
  5. Tarte aux fruits

Quel plat veux-tu ? 4
→ Commande validée : Pizza margherita.
```

Exercice 2 — Le menu d'un jeu

Écris un menu de jeu vidéo avec les options :

```
options = ["Nouvelle partie", "Charger une partie", "Options", "Crédits", "Quitter"]
```

Le menu doit ressembler à ça :

```
╔═══════════════════╗
║   MENU PRINCIPAL  ║
╚═══════════════════╝
  1. Nouvelle partie
  2. Charger une partie
  3. Options
  4. Crédits
  5. Quitter

Ton choix > 1
Tu as choisi : Nouvelle partie
```

Exercice 3 — Menu de classes du lycée

À partir de la liste :

```
classes = ["3TTR", "4TTR", "5TTR", "6TTR", "5TQ", "6TQ", "3GMS", "4GMS", "5GMS", "6GMS"]
```

Écris un programme qui demande à l'élève sa classe et affiche le message :

```
Classes disponibles :
  1. 3TTR
  2. 4TTR
  ...
  10. 6GMS

Ta classe ? 5
→ Bienvenue en 5TTR !
```

Exercice 4 — Bonus : éviter la duplication

© 2025 Mr. Lambrechts https://v2.ttrinfo.be 8 | 9

Tu veux pouvoir afficher **plusieurs menus différents** dans ton programme (menu principal, sous-menu options, etc.) — sans copier-coller la logique d'affichage. Écris une fonction `afficher_menu(titre, options)` qui prend un titre et une liste, et fait l'affichage. Indice : c'est juste la mise en commun de tout ce qu'on a vu, dans un `def`. (Si tu n'as pas encore vu les fonctions, garde cet exercice pour plus tard.)

✍️ À retenir

- Un menu console se construit avec **trois briques** : une liste d'options, une boucle `for + enumerate`, un `input`.
- `enumerate(liste)` donne à chaque tour une paire `(index, élément)` — utile dès qu'il faut un numéro devant les éléments.
- `enumerate(liste, start=1)` commence la numérotation à 1 — plus naturel pour un humain.
- Pour récupérer l'option choisie : `options[choix - 1]` (attention au `-1` puisque les listes commencent à 0).
- Pour ajouter ou retirer une option, on **modifie la liste** — rien d'autre ne change.
- La gestion des erreurs de saisie (lettres au lieu de numéros, numéro hors limites) demande une boucle `while` avec validation — sujet du cours suivant.

Suite

Le menu fonctionne, mais il **plante** si l'utilisateur tape n'importe quoi. Pour gérer ça proprement — redemander tant que la saisie est invalide — il nous faut une autre famille de boucles : **les boucles** `while`. On y vient.