🧩 Défis Python – Chaînes de caractères

Voici une série d'exercices/défis plus complets sur la manipulation des chaines de caractères.

Exploration

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.