Coder un Mastermind en Python

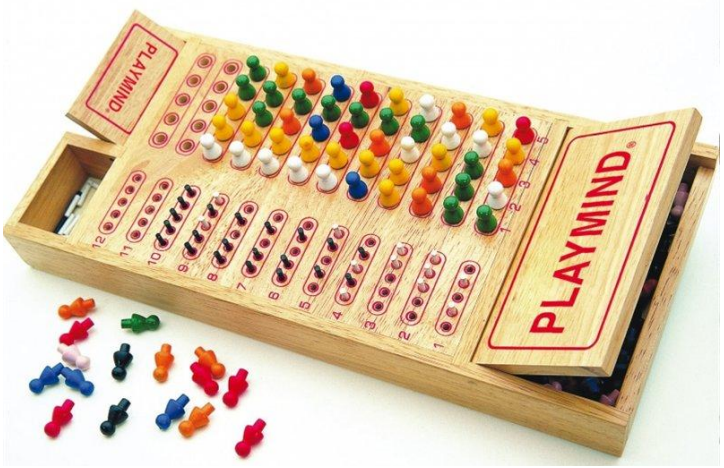
Niveau Débutant • 13-16 ans • 1h30

1. Présentation du jeu

Le **Mastermind** est un jeu consistant à trouver une suite de couleurs en devinant chacune des couleurs qui la composent. Bien que traditionnellement avec des couleurs, nous allons aujourd'hui programmer une version digitale de ce jeu en **Python**, en remplaçant les couleurs par des chiffres.

Certains éléments du code du **Mastermind** ont été perdus… À vous de compléter le code du jeu pour qu'il fonctionne de nouveau !

Dans ce sujet, vous apprendrez à programmer des boucles, des conditions en **Python** et le fonctionnement global d'un jeu.



2. Outils & Ressources

2.1 Ressources

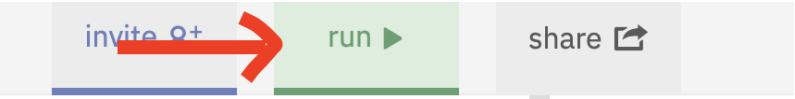
Pour commencer, rendez-vous sur https://repl.it/@EmmaEpitech/PythonMastermind.

L'écran est découpé en plusieurs parties : la partie fichier, à gauche : c'est ici que vous coderez le jeu. Et la partie rendu, à droite : c'est ici que vous verrez votre progression et pourrez tester votre jeu.

2.2 Prise en main de la plateforme

Sur la partie gauche de l'écran, tu peux voir une liste de fichiers **Python**.
Aujourd'hui, tu n'auras besoin de modifier que le fichier "**main.py**". Tu peux toutefois aller voir ce qu'il se passe dans les autres fichiers si tu le souhaites.

Pour lancer ton code appuie sur la flèche verte "**RUN**" en haut de l'écran.



3. Découverte du jeu

3.1 Initialiser le Mastermind

La première étape pour ce jeu de **Mastermind** est l'initialisation. Lors de cette étape, un nombre aléatoire sera déterminé, entre 1000 et 9999, et les variables du jeu seront initialisées, comme la variable contenant le nombre à deviner par exemple.

Pour cela nous allons initialiser 3 variables. Ajoute les lignes de code suivantes au début de ton fichier.

```
try_count = 0
user_guess = None
code_to_guess = str(random.randint(1000, 9999))
```

3.2 La boucle de jeu

À présent, nous allons créer la boucle de jeu. À chaque tour, nous devons vérifier si le code a été trouvé.

La variable **code_to_guess**, correspond au code à deviner, c'est la solution.
Et la variable **user_guess** contient un code que l'utilisateur pense est le bon.

Ajoute la ligne suivante pour créer la boucle de jeu :

```
while code_to_guess != user_guess:
```

A chaque tour de boucle, nous allons demander à l'utilisateur d'entrer un code, puis incrémenter le compteur de 1. Ajoute les lignes suivantes à l'intérieur de la boucle :

```
    user_guess = input("Devine les 4 chiffres: ")
    if len(user_guess) != 4:
        continue
    try_count += 1
```

Lance le jeu pour voir ce qu'il se passe !

3.3 Tour de jeu : vérifier le code entré par l'utilisateur

Nous allons maintenant vérifier les correspondances entre le code entré par l'utilisateur avec le code à deviner.

Ajoute les lignes suivantes pour créer une nouvelle variable, puis parcours chacun des chiffres entrés par l'utilisateur pour vérifier s'il est valide ou non :

```
corrected_code = ""
for i in range(0, 4):
    if user_guess[i] == code_to_guess[i]:
        corrected_code += user_guess[i]
    else:
        corrected_code += "#"
```

Les variables user_guess et code_to_guess fonctionnent comme un tableau. Si la valeur contenue est la suite de chiffres 5634 par exemple, il est possible d'accéder à chaque chiffre séparément selon le tableau suivant:

user_guess[0]	user_guess[1]	user_guess[2]	user_guess[3]
= 5	= 6	= 3	= 4

Lance le jeu pour voir ce qu'il se passe !

3.4 Déterminer la fin de la partie

À présent, nous allons vérifier si l'utilisateur a entré le bon code, et déterminer si la partie est terminée ou si elle doit continuer.

A la suite de la boucle, ajoute les lignes suivantes pour determiner si l'utilisateur à entré trop de mauvais code et si la partie est donc perdue :

```
if try_count >= MAX_TRY:
    print("Perdu :(")
    break
```

Ajoute ensuite la condition suivante pour déterminer si l'utilisateur a entré un mauvais code, et peut continuer à jouer. Pour cela, nous allons réutiliser la variable corrected_code créée précédemment.

```
if code_to_guess != user_guess:
    print("Bons numéros: " + corrected_code)
    print("Essaye à nouveau !")
```

Enfin, ajoute une dernière condition pour féliciter l'utilisateur lorsque le bon code a été trouvé ! Ajoute la ligne suivante à la fin du programme, **en dehors de la condition**.

```
if try_count < MAX_TRY:
    print("Bravo ! Tu as trouvé le code !")
```

*Appuie sur **RUN** et essaye de jouer !*

Bonne chance à toi !

4. Bonus

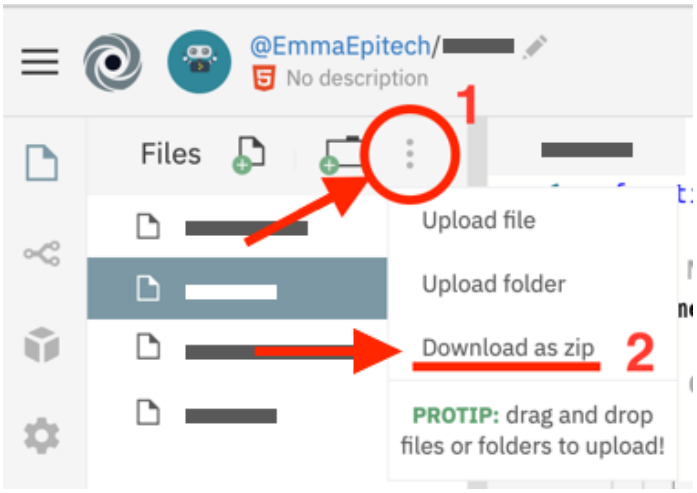
Si tu as fini toutes les étapes précédentes, tout d'abord **Bravo** !

Nous avons préparé quelques étapes bonus que tu peux faire pour améliorer le jeu ou le rendre plus difficile.

- Fait en sorte que le code à deviner fasse plus de 4 chiffres.
- Fait en sorte que le code à deviner puisse contenir des lettres.
- Afficher le code qu'il fallait trouver lorsque le joueur a perdu.
- Proposer à l'utilisateur de rejouer une nouvelle partie.

5. Sauvegarde ton programme

Tu peux enregistrer ton jeu sur ton ordinateur pour le modifier ou y jouer plus tard. Pour cela, clique sur “**Download as zip**”.



6. Quelques liens utiles

Pour apprendre le Python :

➔ https://www.w3schools.com/python

Pour refaire l’exercice :

➔ https://repl.it/@EmmaEpitech/PythonMastermind

Pour voir nos autres exercices :

➔ https://repl.it/@EmmaEpitech

Pour plus d’informations sur nos activités :

➔ https://www.e-mma.org