Projet - MasterMind

Le but de ce projet est de reproduire le fonctionnement du jeu du MasterMind.
Pour comprendre le fonctionnement du jeu : cliquer ici.

Résumé du jeu :

Un joueur choisit une combinaison de 4 ou 5 couleurs placées à un certain endroit.
Le but de l'autre joueur est de deviner en 10 coups maximum la combinaison de couleurs choisie par l'autre joueur.

A chaque tours, le joueur qui cherche la bonne combinaison propose une combinaison et l'autre joueur lui indique le nombre de couleurs bien placées et si les couleurs proposées sont présentes dans la combinaison choisie.

Ce projet peut être développé de plusieurs manières mais on va s'interesser ici à le réaliser de manière simple, en utilisant des listes python. On considère aussi que l'autre joueur est l'ordinateur (ce programme).

La création de la liste à chercher

Pour le but de ce projet, on décide que la combinaison à chercher est définie aléatoirement.
Les couleurs seront définies comme des chaînes de caractère de cette manière : 'bleu' correspond à la couleur bleue.
On se munit des couleurs suivantes :
$bleu$, $rouge$, $vert$, $blanc$, $jaune$ et $orange$
D'autres pourront être rajoutées mais cela augmente la difficulté pour gagner.

1. Créer la liste des couleurs du jeu comme variable globale.

Pour se faire, on va utiliser le module **random** et la méthode **sample**. Cette méthode prend en paramètre une liste et un entier n et renvoie un tirage de n éléments de la liste en paramètre **sans remise**.

2. Créer la fonction `creer_combinaison_solution` qui prend en paramètre une liste **l** et un entier **n** et renvoie un tirage de n éléments de la liste l sans remise.

Les indications pour deviner

A chaque tour, l'autre joueur donne l'indication à celui qui devine le nombre de couleurs bien placées et le nombre de couleurs qui sont bien présentes dans sa combinaison.
Cette partie s'interesse à ces indications.

3. Créer la fonction `nombre_couleurs_présentes` qui prend en paramètre deux listes : une qui correspond à une proposition et l'autre à la solution et renvoie le nombre de couleurs de la proposition présentes dans la solution.

4. Créer la fonction `nombre_couleurs_bien_placees` qui prend en paramètre deux listes : une qui correspind à une propositio net l'autre à la solution et renvoie le nombre de couleurs de la propositions qui sont bien placées dans la solution.

5. Créer la fonction `gagnant` qui prend en paramètre une liste qui correspond à une proposition, une autre correspondant à la solution et renvoie **True** si la liste proposée est la solution, **False** sinon.

La boucle de jeu

On a ici le coeur de notre jeu : la création de la liste solution de manière aléatoire et les vérifications des indications.
Il ne nous manque donc que le fonctionnement du jeu : la boucle de jeu.
Cette boucle de jeu sera conçue par une fonction `jouer` qui sera complétée petit à petit à la suite des questions et fera appel à diverses fonctions.

6. On veut réaliser un bel affichage pour le lancement du jeu. Créer la fonction `premier_affichage` qui ne prend pas de paramètres et affiche dans la console ce rendu :

```
############### MasterMind ###############
# Jeu réalisé par : Nom Prénom           #
##########################################
```

7. Créer la fonction `afficher_gagnant` et qui réalise l'affichage suivant :

```
Vous avez gagné.
La combinaison était bien '/combinaison gagnante/'
```

8. Créer la fonction `choix_joueur`. Cette fonction doit répondre à cet affichage et doit renvoyer la combinaison proposée par le joueur.

```
Couleurs possibles '/couleurs de la question 1/'
Proposer une combinaison de 4 couleurs :
#Choix du joueur#
Vous avez proposé '/combinaison du joueur/'
```

9. Créer la fonction `verifications` qui prend en paramètre une combinaison proposée par le joueur et la combinaison solution et affiche dans la console le nombre de couleurs présentes et le nombre de couleurs bien placées. Elle proposera un affichage ludique similaire à la question 6.
10. Grâce aux fonctions précédentes, compléter la fonction jouer.

En voici le pseudo-code incomplet:

```
    fonction jeu():

    liste_solution <- creer_combinaison_solution
    gagne <- ?
    nombre_tours <- ?
    tant que nombre_tours < ??? ou gagne est ??? : Faire
        proposition <- choix du joueur
        afficher les verifications
        gagne <- gagnant
        si gagne est ??? :
            affichage du message gagnant
        sinon :
            nombre_tours <- ???
```