Indications pour le projet Mastermind

Fonction `genere_combinaison_secrete`

En faisant appel à `help(random)`, on a pu voir que la fonction `sample()` permet d'obtenir un échantillon de `k=n` éléments d'une liste `population`. On peut donc utiliser `sample(liste, k=n)` pour obtenir directement une liste de n éléments différents choisis par les éléments de la liste.

Fonction `saisir_combinaison`

On peut décider de saisir les couleurs tout d'un coup (avec `input()`), sous la forme d'une seule chaîne de caractères de 4 caractères.

Il s'agit alors de récupérer la liste formée des caractères séparément. C'est exactement ce que renvoie `list(chaine)`.

Fonction `combi_to_str`

On peut initialiser une chaîne de caractères vide, et lui ajouter un par un les éléments de la combinaison.

Fonction `nb_occurrences`

Aide 1. On peut initialiser un compteur à zéro, et lui ajouter un à chaque fois que l'on rencontre l'élement en parcourant la liste.

Aide 2. On n'a pas besoin des indices, donc le plus simple est de parcourir la liste `combinaison` par éléments.

Fonction `trouve`

On a besoin des indices, donc le plus simple est de parcourir la liste `combinaison` en parcourant les indices. On peut retourner le bon indice dès qu'on le trouve. Ou retourner l'indice `-1` après n'avoir rien trouvé.

Fonction `evalue_proposition`

1. On commence par faire une copie de la proposition : `copie_proposition = proposition[:]`
2. et une copie de la combinaison secrète...
3. On fait un premier parcours de la proposition, et on compare les pions à ceux de la combinaison secrète pour savoir s'ils sont à la bonne place. S'ils le sont, on change l'élément correspondant dans les copies par le caractère choisi au début du programme (`#`). En changeant également l'élément correspondant dans la combinaison secrète, on s'assure qu'on ne l'utilisera plus pour compter les mal placés.
4. On fait un deuxième parcours de la proposition, en sautant ceux qui sont déjà comptés comme "bien placés". On cherche alors si le pion est présent quelque part dans la combinaison. Vous pouvez utiliser votre fonction `trouve()` et récupérer l'indice correspondant. Si on le trouve, on change alors l'élément correspondant dans les copies par le caractère choisi au début du programme (`X`).
5. On doit renvoyer une copie de la proposition, contenant des caractères `#`, `X` et `0`. Il faut donc refaire un dernier parcours pour remplacer les couleurs absentes par le caractère choisi (`0`), avant de retourner le résultat.

Fonction `manche_de_mastermind`

1. L'ordinateur doit commencer par générer et sauvegarder une combinaison secrète.
2. Le joueur va réessayer de faire des propositions jusqu'à ce qu'il gagne ou jusqu'à ce qu'il perde. On va donc se lancer dans une boucle qui continuera tant qu'il n'a pas gagné, et tant que le nombre d'essais maximmal n'est pas atteint. Ces deux conditions doivent pouvoir s'exprimer donc il faut peut-être définir des variables qui permettent de les exprimer.
3. À chaque tour de boucle, l'ordinateur va demander à l'utilisateur de saisir une combinaison, puis va évaluer cette combinaison avec la fonction que vous avez créée. Il va falloir afficher la réponse, et évaluer si le jeu est terminé ou pas. Si la manche n'est pas terminée, la boucle reprend, et on fait un tour de jeu supplémentaire.