Initiation à Python

LP ESSIG

2013 – 2014

1 Introduction

Le but de ce TP est d'écrire quelques fonctions qui vont permettre d'automatiser un jeu de mastermind.

Dans un premier temps vous devrez écrire des fonctions qui vont automatiser des sous-parties simples du jeu puis vous les associerez pour écrire un programme capable de gérer des parties de mastermind.

Le principe du jeu repose sur l'utilisation de la fonction `randint` disponible dans le module `random` ; prenez quelques temps pour comprendre son fonctionnement avant de vous lancer.

2 Quelques fonctions

Écrivez une fonction `compare` qui prend en paramètre deux nombres ou deux chaînes de caractères (à votre appréciation, mais souvenez-vous en pour la suite). Cette fonction doit retourner une liste de booléens qui indiquent si chaque chiffre (ou caractère) est identique dans chacun des paramètres.

Écrivez une fonction `resultat` qui prend en paramètre une liste de booléens et qui renvoie une chaîne de caractères de la même taille que cette liste. Une valeur `True` dans la liste correspond au caractère « O » dans la chaîne sinon il s'agit du caractère « X ».

En utilisant la fonction `resultat`, écrivez une fonction `resultat_trie` qui place les « O » avant les « X » (ou inversement).

À l'aide de la fonction `input` et du test `isdecimal` sur les chaînes de caractères, écrire une fonction `moins_de_dix_mille` qui renvoie un entier (ou une chaîne de caractères, *cf.* première fonction). Cet entier doit être demandé à l'utilisateur et ne pas dépasser 9999.

Écrire une fonction `initialiser` qui prend un booléen en paramètre. Ce booléen représente un niveau de difficulté et va indiquer si, oui ou non, la partie est une partie difficile.

Cette fonction va retourner un couple (tuple de 2 éléments) contenant le nombre de tentatives possibles pour ce niveau de difficulté et une valeur aléatoire entre 1000 et 9999.

Vous pouvez faire en sorte de retourner un nombre entre 0 et 9999 mais cela implique de faire attention au nombre de chiffres/caractères lorsque vous utilisez la fonction `compare`.

3 Une partie complète

Écrire une fonction `partie_mastermind` en réutilisant l'ensemble des fonctions que vous venez d'écrire. Cette fonction prend en paramètre un booléen représentant le niveau de difficulté et effectue les actions suivantes :

— initialisation de variables en fonction de la difficulté ;
— tant qu'il reste des tentatives :
  — demande d'un nombre inférieur à 10000 à l'utilisateur ;
  — comparaison entre ce nombre et le nombre aléatoire ;
  — affichage du résultat en fonction de la difficulté ;
  — arrêt si le nombre aléatoire a été trouvé.
— affichage d'un message de victoire ou de défaite.

Testez cette fonction en l'appelant avec le paramètre `True` ou `False` pour vérifier que tout fonctionne convenablement.

Écrivez une fonction `mastermind` dont le rôle est de demander à l'utilisateur quel type de partie il veut jouer puis d'attendre la fin de la partie pour en relancer une. Tant que l'utilisateur indique une partie normale (valeur de « 1 », par exemple) ou une partie difficile (valeur de « 2 »), la fonction `partie_mastermind` est appelée avec le bon paramètre. Sinon la fonction `mastermind` s'arrête.

Ajoutez les lignes de code suivantes en fin de fichier :

```
if __name__ == '__main__':
  mastermind()
```

Vous pourrez ainsi tester votre code de 2 manières :

1. En important votre fichier comme un module puis en exécutant la fonction `mastermind` dans un shell.
2. En exécutant votre fichier depuis la ligne de commande avec un appel de la forme `python nom_de_fichier.py`

4 Aller plus loin

Une façon d'améliorer ce programme est de rajouter un mode de jeu facile.

Modifiez les fonctions `mastermind`, `initialiser` et `partie_mastermind` pour tenir compte de ce nouveau mode de jeu (les booléens ne suffisent plus).

Modifiez la fonction `compare` : elle place toujours `True` dans la liste si deux chiffres à la même position sont les mêmes mais s'ils sont différent alors une recherche est effectuée dans le nombre entier. Si le chiffre est présent à une position différente, la valeur `None` est placée dans la liste sinon on y place la valeur `False`.

Écrivez une fonction `resultat_easy` qui transforme les `True` en « O », les `False` en « X » et les `None` en « ~ » et qui est utilisée à la place de `resultat` en mode facile.