Mini-projet : le jeu du pendu

Le but de ce TP est d'utiliser les connaissances acquises en Python pour élaborer un jeu du pendu. Libre à vous pour y arriver de vous aider un peu ou beaucoup de la démarche détaillée ci-dessous. Vous pouvez bien entendu utiliser le document sur le langage Python placé sur le site de la classe, l'aide de l'environnement Pyzo ou chercher de l'aide sur internet.

Vous n'oublierez pas de commenter votre programme et de le tester au fur et à mesure.

1 Programme de base

1.1 Introduction

Le jeu du pendu consiste à deviner un mot secret, lettre par lettre, en un nombre maximal d'essais. On va décomposer le programme en fonctions élémentaires, que l'on va étudier séparément, afin de simplifier l'écriture de celui-ci. Le projet sera rédigé dans un fichier nommé `pendu.py` par exemple.

1.2 Variables et constantes

1. Définir une constante `ERREUR_MAX` correspondant au nombre maximal d'erreurs permises.
2. Initialiser les variables contenant le mot secret (par exemple : "victoria"), la liste des lettres déjà essayées ainsi que le nombre d'erreurs commises.

1.3 État du jeu

Écrire une fonction `mot_trouve` qui retourne **True** si le joueur a réussi à trouver le mot secret et **False** sinon. On se base pour cela sur la liste des lettres déjà essayées que l'on compare au mot secret.

1.4 Entrée des essais de lettres par le joueur

Demander au joueur, avec la fonction `entree_lettre()`, de deviner une des lettres du mot secret et de la rentrer au clavier. Celle-ci sera lue grâce `input`. Afin de rendre le programme robuste et de parer à la plupart des éventualités, on passera la lettre entrée en minuscule à l'aide de la méthode `lower` et on vérifiera que ce qui est rentré est bien non vide, bien une lettre et qu'il n'y en a qu'une. Sinon, afficher un message informatif et relancer l'invitation.

1.5 Suivi du jeu

Écrire une fonction `affichage_resultat()` qui affiche :

- les lettres déjà essayées par le joueur et qui ne sont pas dans le mot secret,
- l'état actuel du mot recherché, en remplaçant par un tiret "-" les lettres pas encore trouvées,
- le nombre d'essais restants. Par exemple : si la liste des lettres trouvées est vide, alors l'expression `affichage_resultat()` devra afficher `--------`, si la liste est `[a,v,m,d,t]`, alors on doit afficher `v--t---a`. On pourra utiliser les méhodes `append` et `join` des chaine de caractères et/ou du parcours de chaine (slicing).

1.6 Programme principal

Il s'agit maintenant d'écrire le programme principal. Vous pourriez déjà commencer par une petite introduction affichant le nom du jeu, le nombre de lettres du mot secret et le nombre d'essais dont on dispose. On finira celle-ci par un `input` afin que le jeu attende que le joueur ait lu et appuie sur la touche <Entrée> avant de continuer.

Ensuite, tant que le joueur n'a pas atteint le nombre maximal d'essais, l'inviter à choisir une lettre : si la lettre est dans le mot secret, on peut afficher un message de félicitation, sinon un message de regrets. Enfin, si le nombre d'essai est supérieur ou égal à `ERREUR_MAX`, afficher que le joueur a perdu et le narguer avec le mot secret qu'il n'a pas réussi à déchiffrer.

2 Améliorations

2.1 Une autre pour la route ?

Écrire une fonction `autre_partie` qui demande au joueur s'il souhaite rejouer, récupère la réponse et retourne **True** ou **False** suivant la réponse.

2.2 Changement du mot secret

Pour que le jeu puisse être rejoué avec intérêt, changer le mot secret à chaque fois que l'on relance le jeu.

1. On pourra définir une liste de mots secrets et on choisira aléatoirement dans celle-ci grâce à la fonction `randint` du module `random`. Ne pas oublier de tenir à jour la liste des mots secrets déjà utilisés pour qu'ils ne soient pas re-proposés.
2. Autre façon de faire : afin de rendre le jeu portable sur plusieurs ordinateurs, écrire la listes des mots secrets dans un fichier texte **mots.txt**, enregistré dans le même répertoire que `pendu.py`, chaque mot secret occupant une ligne du fichier. Étudier comment écrire ce fichier, le charger dans une liste du programme en utilisant le module **os**, les fonctions `open`, `readlines` et `close`.

2.3 Fin rapide

Permettre au joueur de donner le mot complet s'il l'a deviné avant la fin du jeu. Par exemple, on peut demander au joueur de rentrer une lettre ou le mot "devin" pour essayer de voir s'il a trouvé le bon mot secret. Ne pas oublier de gérer le cas où le joueur se serait trompé, par exemple en lui enlevant 2 essais ...