Python Projet

Nassib Abdallah

November 2019

Votre travail consiste à développer un fichier "main.py" contenant le programme principal et un fichier **".py"** pour chacune des parties composant le projet.

1 Première Partie : Jeu du Pendu

Ce jeu consiste à deviner toutes les lettres qui doivent composer un mot, éventuellement avec un nombre limité de tentatives et des thèmes prédéfinis. Chaque fois que le joueur devine une lettre, celle-ci s'affiche. Sinon, le dessin d'un pendu commence à apparaître...

Votre travail dans cette partie consiste à écrire dans le fichier "jeu_pendu.py" les fonctions qui permettent à un utilisateur de jouer au jeu du pendu. Pour ce faire, vous devrez coder les fonctions suivantes :

1. afficherPendu:

   Cette fonction permet d'afficher le Pendu et contient donc toutes les formes du Pendu. Elle prend comme paramètre la liste des lettres incorrectes entrées par l'utilisateur et renvoie le Pendu qui correspond au nombre de fausses tentatives.

2. choisirMot:

   Cette fonction a pour tâche de choisir au hasard un mot dans un fichier .txt et de le renvoyer.

   Elle doit donc effectuer les tâches suivantes :

   (a) Lire un fichier ".txt" qui contient des mots.

   (b) Stocker dans une liste, les mots ayant un minimum de trois lettres.

   (c) Retourne un mot choisi au hasard dans la liste pour le jeu.

3. getLettre : Cette fonction ne prend pas de paramètre d'entrée, elle oblige l'utilisateur à entrer une lettre et renvoie cette lettre en minuscules.

4. verifLettre:

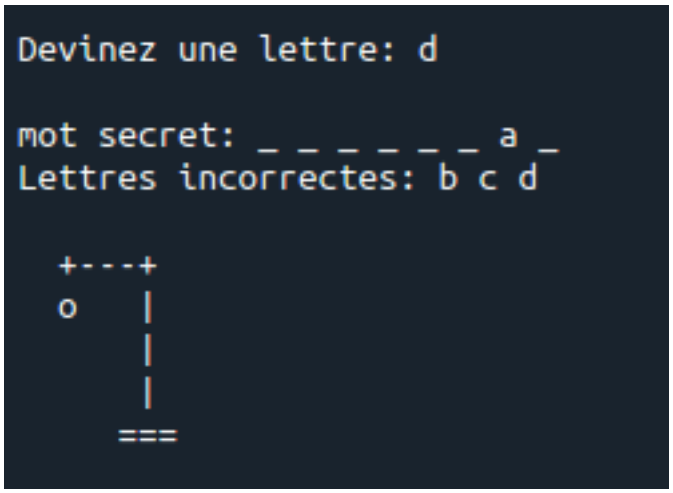

Figure 1: exemple d'execution

cette fonction prend en entrée la lettre saisie par l'utilisateur, le mot à deviner, une liste contenant les lettres incorrectes et une liste contenant les lettres devinées.

La fonction vérifie si la lettre se trouve dans le mot. Si c'est le cas, elle ajoute la lettre à la liste des lettres correctes à l'endroit où elle se trouve dans le mot secret. Sinon, elle ajoute la lettre à la liste des lettres incorrectes.

La fonction renvoie des listes actualisées de lettres devinées et incorrectes.

5. jouerPendu: Cette fonction ne prend pas de paramètres d'entrée. Elle initialise les variables du jeu, implémente les fonctions afin de permettre à l'utilisateur de jouer un jeu.

Une fois le travail effectué, importez et implémentez vos fonctions dans le fichier principal "main.py" puis écrivez le script pour simuler un jeu avec une option permettant de laisser l'utilisateur jouer à nouveau tant qu'il n'a pas entré un mot-clé "STOP".

Remarque : n'oubliez pas de tester chaque fonction séparément pour vous assurer qu'elle renvoie la valeur attendue.

2 Deuxième Partie : Fonctions Récursives

2.1 Travail à réaliser

Dans cette partie, vous allez écrire dans un fichier "Recursives.py", les fonctions récursives suivantes :

1. la fonction factorielle **recursiveFact(n)** qui calcule la factorielle d'un entier.

   Note : Le factoriel de l'entier n, est le produit de tous les entiers positifs inférieurs ou égaux à n. La factorielle est désignée par n !

2. La fonction **sum(L)** qui calcule la somme de n entiers dans une liste.

3. la fonction **multiply(n,m)** qui prend deux nombres et les multiplie ensemble de façon récursive.

4. la fonction **puissance(x, n)** qui calcule x^n en se basant sur la définition suivante :

   (a) $x^0 = 1$

   (b) $x^n = (x^2)^{\frac{n}{2}}$ si n est pair

   (c) $x^n = x(x^{n-1})$ si n est impair

5. la fonction de fibonacci **Fibo(n)** , qui calcule le terme de rang n de la séquence de Fibonacci.

   Par définition, les deux premiers nombres de la séquence de Fibonacci sont 0 et 1, et chaque nombre suivant est la somme des deux précédents. La séquence est créée de la manière suivante : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

   Cette fonction peut être définie comme suit : f(n) = f(n-1) + f(n-2) avec des valeurs fixes f(0) = 0 f(1) = 1

6. la fonction **premierRecursive(n)** pour vérifier si un nombre n est premier (vous devez vérifier si "n" est divisible par tout nombre inférieur à "n")

3 Troisième Partie : Programmation Fonctionnelle

Dans cette partie, vous allez écrire dans un fichier "functions_LMFL.py", les fonctions qui correspondent aux trois sous-parties suivantes :

3.1 Fonctions en arguments de fonctions : Fonction Lambda

1. écrire la fonction **sumLabda(x,y)** qui renvoie la somme de ses deux arguments.

2. écrire la fonction **isDivisible(x,n)** qui prend deux paramètres x,n et retourne une valeur booléenne. Vrai si x est divisible par n et Faux sinon.

NOTE : VOUS DEVEZ UTILISER LES FONCTIONS LAMBDA.

3.2 Fonctions d'ordre supérieur

3.2.1 Les fonctions *map*

map() est une fonction qui prend deux arguments : le premier argument est le nom d'une fonction et le second est une séquence (par exemple une liste). map() applique la fonction à tous les éléments de la séquence.

1. écrire une fonction qui renvoie les éléments au carré dans une liste en utilisant la fonction map et lambda.
2. écrire une fonction qui additionne les éléments de deux listes en utilisant la fonction map et lambda.
3. écrire une fonction qui calcule les valeurs de f(x) de x allant de 1 à 10 pour les deux fonctions : f(x) = 2,5*x^2 + 2 et f(x) = cos(x).

3.2.2 Les fonctions *filter*

filter() est une fonction qui requiert deux arguments : le premier est une fonction qui teste chaque élément du second argument qui est une liste itérable pour voir si elle vérifie la condition ou non (elle renvoie une valeur booléenne pour chaque élément de la liste en entrée).

1. écrire une fonction qui utilise la fonction filter() pour filtrer une liste (en paramètre) pour ne laisser que des nombres négatifs.
2. écrire une fonction qui, en utilisant la fonction filter(), filtre les nombres pairs d'une liste L (en paramètre) pour ne laisser que les nombres impairs dans une nouvelle liste.
3. écrire une fonction qui utilise à la fois les fonctions map() et filter() afin d'ajouter 2000 aux valeurs inférieures à 8000 dans la liste L (en paramètre).

3.2.3 Listes en compréhension

Listes en compréhension sont une manière élégante de définir et de créer des listes basées sur des listes existantes. Les listes sont généralement plus compactes et plus rapides que les fonctions et boucles de création de listes normales.

1. écrire une fonction qui utilise une liste en compréhension pour construire une liste à partir des carrés de chaque élément d'une liste (en paramètre), si le carré est supérieur à 50.
2. écrire une fonction qui, à l'aide d'une liste de compréhension, renvoie une nouvelle liste de la liste "nombres" (en paramètre), qui ne contient que les nombres positifs de la liste sous forme d'entiers.

3. écrire une fonction qui prend un dictionnaire en paramètre et utilise la liste en compréhension pour retourner une liste de noms de véhicules dont le poids est inférieur à 5000 kilogrammes. Dans la même liste de compréhension, mettez les noms clés en majuscules.
   Le dictionnaire donné est composé de véhicules et de leur poids en kilogrammes. dict={ "Sedan" : 1500, "SUV" : 2000, "Pickup" : 2500, "Minivan" : 1600, "Fourgon" : 2400, "Semi" : 13600, "Bicyclette" : 7, "Moto" : 110}

4 Code principal

Rassemblez tous vos tests en un seul code principal qui respecte l'algorithme ci-dessous :

- Le programme doit afficher un menu contenant un numéro pour chaque partie de cet énoncé suivi du titre de la partie. Exemple :
  1- Jeu du Pendu
  2- Fonction Récursive
  3- Programmation Fonctionnelle
  4-Quitter

- L'utilisateur choisit la partie qu'il veut tester en entrant son numéro. Si le numéro saisi n'est pas dans le menu, un message d'erreur s'affiche *"le numéro saisi n'existe pas, veuillez saisir un numéro valide"*.

- Dans chaque partie, le programme affiche une description et montre toutes les fonctions qu'elle contient. De plus, il doit permettre à l'utilisateur de tester la fonction de son choix.

5 Rendu du projet

Envoyez votre travail sous la forme d'une archive(.zip) à l'adresse nassib.abdallah@univ-angers.fr.
Le nom de cette archive sera NOM_Prenom_Projet.zip (exemple ABDALLAH_Nassib_Projet.zip).
Cette archive contiendra un répertoire dont le nom sera NOM_Prenom_Projet.
Le contenu de ce répertoire sera organisé ainsi :

- Les noms des fichiers/dossiers seront en minuscules et sans caractère spéciaux (-,+,é,è,etc.).

- un fichier lisezmoi.txt **(format txt simple, pas de .docx ou .odt)** avec votre nom et prénom et indiquant tout point particulier mettant en valeur votre travail et sur lequel vous voulez attirer l'attention du correcteur.