Python MiniProjet

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 Premiere Partie : Jeu du Morpion

Jeu du morpion :
Le morpion est un jeu de réflexion à deux joueurs. Il se joue sur une grille carrée de 3x3 cases. Les joueurs doivent remplir chacun à leur tour une case de la grille avec le symbole qui leur est attribué : O (rond) ou X (croix). Le gagnant est celui qui aligne en premier trois de ses symboles, horizontalement, verticalement ou en diagonale.

écrire dans le fichier **"morpion.py"** les fonctions suivantes:

1. la fonction **initialiserGrille** qui retourne une grille initialisée. La taille de la grille sera définie par une variable TAILLE_GRILLE fixée à 3 au début du script.

2. la fonction **afficherGrille** qui prend une grille en paramètre et l'affiche selon le modèle dans la figure 1.

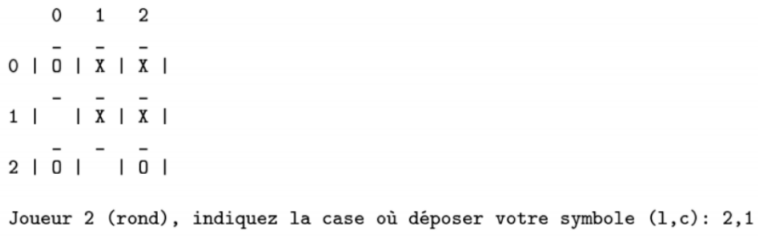

Figure 1: Exemple de grille

3. la fonction **grillePleine** qui prend une grille en paramètre et retourne un booléen qui vaut *True* si la grille est pleine .

4. la fonction **estLibre** qui prend en paramètres une ligne, une colonne, une grille et retourne un booléen qui indique si la case en disponible.

5. la fonction **placerPion** qui prend en paramètres le type de symbole à déposer, une ligne, une colonne, une grille et qui retourne une grille modifiée.

6. la fonction **coupGagnant** qui prend en paramètres les coordonnées d'une case ainsi qu'une grille et retourne un booléen indiquant si le coup est gagnant.

Une fois le travail réalisé, implémenter vos fonctions dans le fichier principal **main.py** puis écrivez le programme permettant de simuler une partie de jeu.

2 Deuxieme Partie : Fonctions Récursives

Ecrire dans un fichier **"Recursives.py"**, les fonctions récursives suivantes :

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

  1. $x^0 = 1$
  2. $x^n = (x^2)^{\frac{n}{2}}$ si n est pair
  3. $x^n = x(x^{n-1})$ si n est impair

- une fonction récursive **palindrome(s)** qui renvoie $True$ si la chaine de caractères s est un palindrome, $False$ sinon.
  NB : un palindrome est une chaine de caractères qui peut se lire indifféremment dans les deux sens.

- une fonction récursive **nombres_impairs(l)** qui renvoie une liste constituée des nombres impairs contenus dans la liste numérique l.

- une fonction récursive **trier_insertion(l)** qui a le principe de trier les entiers dans la liste l selon l'algorithme suivant:
  Pour trier n entiers rangés dans une liste, il suffit de trier les n-1 premiers entiers puis d'insérer le nème à sa place dans les n-1 déjà triés. Ecrire la fonction auxiliaire *inserer* qui sera également récursive.

Implementer et tester chacune de ces fonctions dans votre fichier **"main.py"**.

3 Troisième Partie : Programmation Fonctionnelle

3.1 Fonctions en arguments

Ecrire une fonction récursive *zero(f, eps, a, b)* qui renvoie le zéro "z" d'une fonction continue f strictement monotone entre a et b, en effectuant une recherche

dichotomique.
Comme paramètres de la fonction *zero*, on a aussi deux réels a et b vérifiant $a < z < b$, et un réel positif *eps* qui vérifiera pour finir $|f(z)| < eps$.

Tester votre fonction *zero* avec la fonction "cos" puis avec la fonction $f(x) = 2x^2–4$ dans votre fichier **"main.py"**.

aide:adbmal snoitcnof sel zesilitU

3.2 Fonctions d'ordre supérieur

- Ecrire une fonction récursive **longueurs_chaines(l)** qui renvoie, étant donnée une liste de chaînes de caractères l, la liste des longueurs respectives de ces chaînes.
- Ecrire une fonction **longeurs_chaines_map(l)** qui utilise la fonction *map* qui renvoie la liste des longueurs respectives d'une liste de chaine de caracteres.
- Ecrire une fonction qui, en utilisant la fonction $filter$, renvoie, étant donnée une liste de chaînes de caractères, la liste des chaînes de longueur supérieure à 3.
- En utilisant les listes en compréhension, écrire une instruction dans une fonction qui renvoie la liste des éléments définis par x^2 sachant que $x \in N$ et $x < 20$ et x pair.
- Ecrire une fonction récursive $intersection(l, m)$ qui renvoie la liste des éléments communs aux deux listes l et m.

Implementer et tester chacune de ces fonctions dans votre fichier **"main.py"**.

4 Quatrième Partie : Arbre Binaire

Un arbre est une structure constituée de noeuds, qui peuvent avoir des enfants (qui sont d'autres noeuds). Deux types de noeuds ont un statut particulier : les noeuds qui n'ont aucun enfant, qu'on appelle des feuilles, et un noeud qui n'est l'enfant d'aucun autre noeud, qu'on appelle la racine.

Ecrire dans un fichier **"ArbreBinaire.py"**, les fonctions suivantes :

1. Soit un arbre binaire ordonné dont chaque nœud X contient une valeur entière telle que :
   valeurs du sous-arbre gauche de $X < valeur(X) < valeurs$ du sous-arbre droit de X.
   Ecrire une fonction récursive qui :
   (a) indique si une valeur donnée se trouve dans l'arbre (renvoie True ou False)
   (b) insère une valeur donnée (renvoie un arbre reconstruit)

2. une fonction récursive qui calcule la valeur arithmétique associée à un arbre binaire dont les feuilles sont des entiers strictement positifs et les nœuds internes sont des symboles additifs ou multiplicatifs.

3. Ecrire une fonction récursive qui calcule la valeur booléenne d'une formule en *et*, *ou*, *non*.
   NB : on connaît les règles élémentaires du calcul booléen (True et True donne True, True ou False donne True, non False donne True, etc.).
   Un exemple de formule en représentation Python :
   ['et', ['ou', [False, [], []], ['non', [False, [], []], []]], ['et', ['ou', [False, [], []],[True, [], []]],[True, [], []]]]
   L'expression ci-dessus aura pour résultat True.
   NB : dans un arbre 'non' on ignorera le fils droit (initialisé à [])

Tester dans votre fichier **"main.py"**,toutes les fonctions que vous avez codées dans cette partie.

5 Code principal

Regrouper l'ensemble de vos testes 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 :
  0- Morpion
  1- Fonction Récursive
  2- Programmation Fonctionnelle
  3- Arbre Binaire
  4-Quitter

- L'uilisateur choisit la partie qu'il souhaite tester en saisissant son numéro. Si le numero entré ne figure pas dans le menu, affichez un message d'erreur *"le numéro que vous avez saisie n'existe pas, veuillez saisir un numéro valide"*.

- Dans chacune des parties, le programme affiche une description et affiche l'ensemble des fonctions qu'elle contient. De plus, elle doit permettre à l'utilisateur de pouvoir tester la fonction de son choix.

6 Rendu du projet

Envoyez votre travail sous la forme d'une archive à l'adresse nassib.abdallah@uco.fr. Le nom de cette archive sera vos-noms-projet.zip. Cette archive contiendra un répertoire dont le nom sera vos-noms-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, surtout 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.