TP1 RAPPELS SUR LES LISTES INDEXEES

OBJECTIFS :

Revoir les notions de bases : entrées, sorties variables, structure de contrôles

Revoir les notions plus complexes de fonctions, de listes indexées

Définir de bonnes pratiques.

1. Création de listes indexées simples

A l'aide de Pyzo2015 ouvrir un fichier nommé : votre_nom_revision_liste.py que vous sauverez dans votre répertoire personnel au fur et à mesure de l'avancée du TP.

a. Importer une bibliothèque (voir Annexe)

Importer la fonction nommée deepcopy de la bibliothèque copy :

```
from copy import deepcopy
```

on évite d'écrire `from copy import *` qui permet d'importer toutes les fonctions proposées dans la bibliothèque copy. Cela permet d'éviter le risque d'avoir un conflit entre une fonction de la bibliothèque copy qui aurait le même nom qu'une de celles que l'on crée.

b. Créer une liste vide

La liste vide s'appellera listeVide. Le nom de la variable indique son contenu et se compose de deux mots avec le deuxième qui commence par une majuscule.

c. Créer deux listes liste01, liste02 composées de cinq 0, de deux façons différentes

Faites afficher le contenu de ces listes dans la console et dans le programme.

d. Ajouter à liste01 la valeur 1 puis 2 puis 3

Vérifier le contenu de liste01 à chaque extension

e. Tranche de liste

Il est possible de sélectionner une partie d'une liste en utilisant un indiçage construit sur le modèle `[m:n+1]` pour récupérer tous les éléments, du é**m**ième au é**n**ième (de l'élément `m` inclus à l'élément `n+1` exclu). On obtient ainsi une tranche de liste.

Donner la commande qui permet de récupérer tous les 0 de liste01 (modifiée comme précédemment) dans la console dans le programme.

Donner la commande qui permet de récupérer tous les nombres différents de 0 de liste01 (modifiée comme précédemment) dans la console dans le programme.

Donner la commande qui permet de récupérer [0, 1, 2] ?

Qu'obtient-on avec liste01[-1] ?

TP1 RAPPELS SUR LES LISTES INDEXEES

f. Listes par compréhension

Une liste en compréhension L a la syntaxe minimale suivante : **[expr for x in t]**

où

- la paire de crochets, les mots-clefs *for* et *in* sont obligatoires
- t est, par exemple, une liste (voir plus bas pour d'autres possibilités)
- x est l'élément courant qui parcourt la liste t ; x est appelé variable de contrôle de la liste en compréhension
- expr est une expression qui dépend en général de x et dont la valeur est placée dans L

Si t est un conteneur (une liste, une chaîne, etc), la liste en compréhension L avec la syntaxe ci-dessus a toujours même nombre d'éléments que le conteneur t.

Exemple :

```
liste10 = [10*i for i in range(10)]
```

Entourer expr , x et le contenu du in . Que doit-on obtenir ?

Sur le même modèle créer **liste0123** qui contient tous les entiers de 0 à 20 compris

Ajouter une condition sur ces valeurs derrière le range pour créer une liste **listePair** qui contient que les valeurs paires.

g. Copies de listes

Copie simple

Créer une nouvelle liste : listeCopieSimple et utiliser l'opérateur d'affectation pour y copier le contenu de liste01

Copie en profondeur

Créer une nouvelle liste : listeCopieProfonde grâce au code suivant :

```
35 #copie en profondeur
36 listeCopieProfonde = deepcopy(liste01)
```

Ajouter la valeur 36 à liste01 puis faites afficher la composition de liste01 , listeCopieSimple et listeCopieProfonde. Que constatez-vous ?

TP1 RAPPELS SUR LES LISTES INDEXEES

2. Travail sur les listes

a. Création d'une bibliothèque de fonctions (voir Annexe)

Vous allez créer dans le même répertoire que celui du fichier votre_nom_revision_liste.py , le fichier lycee.py.

On utilisera liste01modifiée pour tester les fonctions.

b. Création de la fonction « parcours_par_index »

Créer la fonction « parcours_par_index » qui prend comme argument une liste et affiche les valeurs ligne par ligne. Vous garderez la même structure pour la suite (commentaire mise en forme dans le print)

```
1  #parcours
2  #bonnes pratique
3  def parcours_par_index(l):
4      '''
5      affiche les élements de la liste l
6      parcours par index
7      '''
8      for i in range(len(l)):
9          print(f"index {i} , valeur {l[i]}")
```

Modifier le import

Appeler la fonction dans votre programme principal

Vérifier son fonctionnement

c. Création de la fonction « parcours_par_valeur»

Même chose mais en parcourant les différents éléments de la liste . On désire ici afficher aussi la valeur des indices.

d. Création de la fonction « recherche_une_valeur »

Renvoie Vrai ou faux suivant que la valeur passée en argument est présente dans la liste (on n'utilise pas le in)

e. Création de la fonction « recherche_un_nb_valeur »

Si la valeur passée en argument est présente dans la liste, cette fonction renvoie le nombre de de répétition dans cette liste (nombre d'occurrences). Si cette valeur n'est pas présente dans la liste, la fonction retourne false.

.

f. Création de la fonction « recherche_minimum»

Renvoie la valeur du minimum et le premier indice s'il existe plusieurs minimum de même valeur.

g. Création de la fonction « recherche_maximum»

Idem maximum

3. Création de listes de listes

- Créer la liste matriceX55 qui est une liste de 5 listes de 5 par compréhension :

```
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]
, [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

- Créer la liste matriceX0123 qui est une liste de 5 listes de 5 par compréhension

```
[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10,
11, 12, 13, 14], [15, 16, 17, 18, 19],
[20, 21, 22, 23, 24]]
```

On accède aux différents éléments par un double indiçage ex matriceXX55 [2][3]. On utilise aussi ce double indiçage pour affecter si besoin une valeur à la bonne place.

- Créer la fonction matriceDouble qui multiplie par 2 les valeurs de matriceX0123.
- Créer la fonction matriceDiagonale qui renvoie une liste des éléments se trouvant aux indices i X i : [0,6,12,18,24]

TP1 ANNEXES

1. Bibliothèques en Python

Il existe plusieurs manières d'importer une bibliothèque. Imaginons que l'on ait créé le fichier test.py dont le code est le suivant :

```
print "Hello world"

def f(x):
  return x ** 2

def g(x):
  return x ** 0.5

variable = 5
```

On suppose que le fichier test.py se trouve dans le répertoire courant. Voici plusieurs manières de l'importer comme une bibliothèque.

Syntaxe	**Accès à f**	**Accès à g**	**Accès à variable**
import test	test.f	test.g	test.variable
import test as t	t.f	t.g	t.variable
from test import *	f	g	variable
from test import g	**Impossible**	**g**	**Impossible**
from test import g as h	Impossible	h	Impossible

Remarque importante : une fois une bibliothèque chargée, toute modification de son code-source est sans effet. Il faut utiliser reload(test) pour forcer le rechargement