TD1Bis: Modules Python

PC0, L2 Informatique UBO

26 janvier 2022

But : fonction de fonction, liste, utilisation de module, exercices d'algorithmique
Avant de faire les exercices de la section 4, exécuter les codes des sections 1 à 3.

1 Complément sur les fonctions

Une fonction peut être en paramètre d'une autre fonction. Voici un exemple :

```
def f(x):
    return 2*x+1
def g(x):
    return x//2
def h(fonc, x):
    return fonc(x)
print(h(f,3)) #-> 7
print(h(g,4)) #-> 2
```

2 Utilisation des listes et référence objet

On définit une liste de valeurs en utilisant la notation "[]". Par exemple

```
l1 = [2, 6, 10, 13 ,0]
print (l1[0]) #->2 l'index commence à 0
l1[0]=8       #modification de la valeur du 1er element de la liste l1
print(l1) #->[8, 6, 10, 13, 0]
```

On peut modifier un élément de la liste et copier aussi une liste, mais attention à la copie :

```
l1 = [2, 6, 10, 13 ,0]
l2=l1    #copie de l1, même objet
l3=l1[:] #copie de l1 indépendante= objet différent
l1[0]=8
print(l1)  #->[8, 6, 10, 13, 0]
print(l2) #l2 est aussi modifié car c'est le même objet que l1
           #->[8, 6, 10, 13, 0]
print(l3) #l3 est une copie indépendante de l1, l3 n'est pas modifiée
           #->[2, 6, 10, 13, 0]
print("valeurs identiques l1 et l2?", l1==l2, "objets identiques?", l1 is l2) #True True
print("valeurs identiques l1 et l1[:]?", l1==l1[:], "objets identiques?", l1 is l1[:])#True False
```

3 Espace de nom, module

Un **module** est un fichier script Python permettant de définir des éléments de programme réutilisables. Ce mécanisme permet d'élaborer efficacement des bibliothèques de fonctions ou de classes.

Avantages des modules :
— réutilisation du code ;
— la documentation et les tests peuvent être intégrés au module ;
— réalisation de services ou de données partagés ;
— partition de l'espace de noms du système.
Les modules sont définis dans des fichiers dont l'extension est en *.py* (le module machin est dans le fichier machin.py)

L'instruction *import* charge et exécute le module indiqué s'il n'est pas déjà chargé. L'ensemble des définitions contenues dans ce module deviennent alors disponibles : variables globales, fonctions, classes. Vérifier que vous avez bien accès au module *numpy* avec le code suivant :

```
import numpy as np
np.sin(np.pi/2)
```

On peut détailler les fonctions à importer du module choisi avec $from$ .. *import* ...

```
from numpy import sin,cos
np.sin(np.pi/2)
```

Il est conseillé d'importer dans l'ordre :
— les modules de la bibliothèque standard
— les modules des bibliothèques tierces
— Les modules personnels
les bibliothèques NumPy et matplotlib sont utilisées pour le calcul scientifique et la visualisation.

3.1 La bibliothèque NumPy

Le module numpy est la boîte à outils indispensable pour faire du calcul scientifique avec Python. Pour modéliser les vecteurs, matrices et, plus généralement, les tableaux à n dimensions, numpy fournit le type ndarray. On utilisera les fonctions mathématiques disponibles.
Pour plus de détail sur les fonctions importées, voir
`https://courspython.com/fonctions-mathematiques-numpy.html`
`https://docs.scipy.org/doc/numpy/reference/routines.math.html`

Pour connaitre la liste des objets exportés de numpy, utiliser

```
np.*?
```

Après avoir importé le module comme suit :

```
import numpy as np
```

3.2 La bibliothèque matplotlib

Cette bibliothèque permet toutes sortes de représentations de graphe 2D.
Quelques exemples vous sont ici donnés. Vous pouvez tester ces exemples en exécutant les codes ci-dessous :

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-10, 10, 200)
y = np.sin(np.pi * x)/(np.pi * x)
plt.plot(x, y)
plt.show()
```

ou encore

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-2, 2, 500)
plt.plot(x, np.around(x,0), label="around(x,0)")
plt.plot(x, np.trunc(x), label="trunc(x)")
plt.legend()
plt.show()
```

Pour plus de détail, voir les exemples disponibles sous
`https://courspython.com/introduction-courbes.html#`

PyLab permet d'utiliser de manière plus aisée les bibliothèques NumPy (http ://www.numpy.org/) et matplotlib (https ://matplotlib.org/) pour de la programmation scientifique avec Python.

Exemple avec PyLab

```
import pylab
pylab.figure(1) #crée la figure 1
pylab.plot([1,2,3,4], [1,7,3,5]) #dessine sur la figure 1
pylab.show() #affiche
```

3.3 Charger ses propres modules

Mettre les fonctions et les classes dans vos propres modules dans des fichiers ayant l'extension *.py*. Le chemin d'accès aux modules est disponible sous jupyter avec :

```
import os
import sys
print(sys.path)
```

On peut rajouter un accès aux fichiers se trouvant dans un répertoire.

```
sys.path.insert(0, os.path.abspath('H:\\L2Python\TD1'))
```

4 Exercices

4.1 Fonction de fonction

On veut définir les valeurs des fonctions *sinus*, *cosinus*, *tangente* pour les valeurs allant de 0 à $2 * pi$.

1. Définir à partir de la liste $l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]$, la liste lp des différentes valeurs allant de 0 à $2 * pi$ espacée de $pi/6$. Prendre en compte la remarque de la section 2 pour ne pas modifier la valeurs de la liste l.
2. Définir une fonction qui calcule le *cosinus* des différentes valeurs de lp. Cette fonction retourne une liste de valeurs. Visualiser dans une figure, les différentes valeurs pour chaque valeur de lp.
3. Définir une fonction fon qui prend en paramètres une fonction et une liste et qui permet d'appliquer la fonction en paramètre sur la liste passée en paramètre. Utiliser cette fonction pour recalculer le *cosinus*, le *sinus* et *tangente* de la liste de valeurs définies dans lp. Visualiser les valeurs sur une figure.

4.2 Dessins géométriques

Dans cet exercice, **on n'utilise pas de classe Point** pour l'ensemble des questions. Un point est modélisé sous forme de liste de deux valeurs correspondant à l'abscisse et à l'ordonnée du point.

1. Définir une fonction qui prend les quatre points formant les coins du carré et qui affiche les quatre segments définissant le carré.
2. Définir une nouvelle fonction qui permet d'afficher un segment de droite à partir de deux points et avec une couleur au choix.
3. Définir une fonction pour afficher un carré en rouge et un triangle en bleu au-dessus du carré.
4. A partir d'un point et d'une longueur, définir les différents points associés au coins du carré. Définir la fonctions qui renvoie la liste des points du carré puis afficher le carré qui part de l'origine (origine =[0,0])et qui est de longueur 15.
5. Définir un module regroupant vos différentes fonctions, et utiliser ce module pour afficher un nombre de carrés et triangles de votre choix.

Importer les différents modules nécessaires pour l'affichage.

Variante avec la classe Point : utiliser la classe Point (ou Coordinate) vu en cours à la place des listes pour réécrire les fonctions des questions précédentes.

4.3 Tri à bulles

Définir une liste de valeurs puis les trier dans l'ordre croissant en utilisant l'algorithme du tri à bulles. Compter le nombre de permutations effectuées.