TP n°2 : instructions conditionnelles, boucles

PTSI Lycée Eiffel

27 septembre 2017

1 Familiarisation avec l'environnement de développement

1.1 Quelques types de données plus complexes

Vous avez déjà manipulé lors du premier TP quelques variables de types simples numériques, habituellement entiers (int) ou flottants (float). Nous aurons l'occasion au cours de l'année de manipuler d'autres types de données, que nous allons introduire dans ce TP pour se familiariser avec avant de les aborder plus en détail en cours dans quelques semaines.
Tapez les instructions suivantes :

- x='ptsi'
- type(x)

Le type de données str (string : chaîne de caractères) sera l'un de ceux que nous utiliserons le plus puisqu'il sera indispensable à chaque fois que nous aurons à travailler avec du texte. Vous pouvez taper les commandes suivantes en tentant de répondre aux questions entre parenthèses :

- x[2]
- x[0] (que représente de façon générale x[i] ?)
- len(x) (à quoi sert la fonction len ?)
- y=x+'power'
- y (qu'effectue l'opération d'addition sur des chaînes de caractères ?)
- z='24'
- type(z)
- z+24 (expliquez ce qui se passe)
- z+z
- int(z)+24
- w='Les TP de Python c'est trop chouette' (expliquez à nouveau ce qui se passe)
- w="Les TP de Python c'est trop chouette"

Un autre type de données vraiment plus complexe est celui de liste (qui regroupe en fait plusieurs variables pouvant éventuellement être de types différents), pour lequel nous allons également donner quelques commandes basiques.

- t=[1,2,3,4]
- type(t)
- len(t)
- t[2]
- t.append(12)
- t (que fait la commande append ?)
- t.remove(3)
- t (que fait la commande remove ?)
- t.reverse()
- t (que fait la commande reverse ?)
- t.sort()

- t (que fait la commande sort ?)

Question subsidiaire : quel est à votre avis la meilleure façon de procéder pour faire ce que fait la commande sort ? On répondra partiellement à cette question loin d'être simple dans un futur TD.

1.2 Quelques erreurs en Python

Voici une petite liste d'erreurs classiques que vous risquez de croiser en programmant en Python. Repérez le type d'erreur annoncé par Python, et chercher à comprendre précisément où se situe l'erreur.

- a=b
- 3/0

Le petit programme suivant est à taper dans l'éditeur et non dans la console :

- if a=2 print('bonjour')

Tester le même programme avec un == à la place du =

1.3 Tracé de courbes avec Python

Pour tracer des courbes rudimentaires, nous allons faire appel à un environnement rudimentaire qui nécessite l'installation sur votre Python d'un module nommé matplotlib. Le principe des graphiques réalisés par ce module est simplement de relier entre eux des points dans le plan dont on donne les abscisses et les ordonnées sous forme de deux listes de nombres. Un exemple de petite programme faisant intervenir ce module :

```
import matplotlib.pyplot as plt
abscisse=[0,1,2,3,4,5]
ordonnee=[2 ?4,7,5,12,8]
plt.plot(abscisse,ordonnée)
plt.xlabel('variable x')
plt.ylabel('image f(x)')
plt.title('Quelle superbe courbe !')
plt.show()
```

Essayez de comprendre et de commenter le rôle de chacune des lignes de ce programme.

2 Premiers programmes ; premières boucles

2.1 Quelques exercices

Quelques exercices pour s'échauffer ne nécessitant théoriquement pas de connaissances plus spécifiques que ce qu'on a déjà abordé lors du premier TP.

Exercice :

1. Écrire un programme qui demande son nom à l'utilisateur, puis affiche **Bonjour machin** (en remplaçant machin par le nom saisi).
2. Écrire un programme qui demande à l'utilisateur un nombre d'heures, un nombre de minutes et un nombre de secondes, et qui convertit le tout en secondes.
3. Écrire un programme qui fait exactement le contraire du précédent (on demande un nombre de secondes et on convertit en heures, minutes et secondes).

Testez désormais le petit programme suivant :

- a=input('Saisissez un premier nombre :')

- b=input('Saisissez un deuxième nombre :')
- if a>b :
- print(a)
- else :
- print(b)

Que fait ce programme ? Modifiez-le pour qu'il calcule et affiche le plus grand parmi trois nombres saisis par l'utilisateur.

Écrivez maintenant vous-même un programme effectuant la résolution d'équations du second degré.

2.2 Premières boucles FOR

L'instruction for permet d'effectuer des tâches répétitives en utilisant une seule instruction. Son fonctionnement diffère légèrement de celui des instructions équivalentes dans d'autres langages que Python, nous allons simplement voir quelques exemples basiques d'utilisation, commencez par comprendre comment fonctionne la commande range à l'aide des exemples suivants :

- range(3,7)
- range(7)
- range(2,15,3)

Effectuez maintenant le tout petit programme suivant, et expliquez ce qui se passe :

```
for i in range(3,7) :
    print(3*i)
```

Essayez à présent d'écrire un programme calculant et affichant la somme des carrées de tous les nombres entiers inférieurs ou égaux à 100. Même question ensuite mais en ne gardant que la somme des carrés des entiers impairs.