Decouverte de Python

June 8, 2015

1 Découverte de Python

Dans ce cours autour de la programmation Python et de l'algorithmie, nous verons beaucoup de petits morceaux de programmes.

Il est vivement recommander de ne pas faire **que** les questions poser mais aussi d'**expérimenter** par soi même.

1.1 Base de Python et Spyder

Dans les encadrés, on peut lancer des commandes Python et écrire des petits programmes.

La touche **F5** ou l'icône en triangle vert permet d'executer tout le programme écrit dans la partie de droite.

```
In [1]: print('Boujour')
```

```
Boujour
```

- Modifier la commande précédente pour qu'elle n'affiche plus bonjour mais votre nom puis executer cette commande.

Il est possible d'afficher plusieurs messages:

```
In [2]: print("bonjour")
        print('monde')
```

```
bonjour
monde
```

- Modifier le programme précédent pour qu'il affiche:

  Bonjour

  tous

  le

  monde

Les commandes commençant par un dièze, ne seront pas executées. On appelle cela des commentaires. Nous les utiliserons pour expliquer ce que l'on fait.

```
In [7]: # la commande suivante affiche Bonjour
        print('bonjour')
        # Quand on veut afficher quelque chose, on utilisera la commande print()
```

```
bonjour
```

1.2 Les variables

Les variables sont des *enveloppes* où l'on stockera des nombres, des mots, des phrases ou des choses plus compliqués.

```
In [4]: un_nombre = 10
```

Ainsi dans *l'enveloppe* appelée **un_nombre** on y met le nombre 10.
C'est ce que l'on appelle **assigner** une **valeur** à une variable ou **stocker** une valeur dans une variable.

```
In [5]: un_nombre = 10
        print(un_nombre)
```

```
10
```

- Modifier le programme précédent pour stocker 2 dans **un_nombre** puis l'arrficher.

Pour stocker des mots ou des phrases, il faut les mettre entre guillemets

```
In [6]: un_mot = "Coucou"
        une_phrase = "ceci est une phrase"
```

- Modifier le programme précédent pour changer le contenu des variables puis ajouter deux lignes pour afficher ce que contiennent ces variables (avec la commande **print**).

1.3 Opérations

Un ordinateur ne sais **que** calculer. On peut donc lui faire faire toutes les opérations classiques

```
In [13]: print(2 + 3)
         print(4*5 + 1)
```

```
5
21
```

Ces simples calculs ne sont pas très interessants. La programmation devient réellement efficace quand on travail non plus avec des nombres que l'on définit par avance mais avec des nombres stockés dans des variables.

```
In [17]: # On définit 3 variables
         a = 2
         b = 10
         c = 100
```

```
In [18]: # On peut afficher des calculs avec elles
         print(a * b)
         print(a * b + c)
         print(a / 3 + c * 0.5)
```

```
20
120
50.666666666666664
```

```
In [20]: # On peut aussi stocker le résultat d'un calcul dans une variables
         d = a + b
         print(d)
         print(a)
         a = a + 1
         print(a)
```

```
13
3
4
```

- Modifier les valeurs initiales de a, b et c puis executer les 3 blocks.
- Écrire un programme avec une variable **age** qui affiche

  Bonjour, j'ai **age** ans.

1.4 Exercice avec les suites

Nous allons commencer à voir comment utiliser un programme pour calculer les permiers termes d'une suite.

Soit u_n la suite définie par

$$u_0 = 100$$

et

$$u_{n+1} = 0.4u_n + 10$$

On veut calculer u_3

```
In [22]: # Premiere valeur de la suite (initialisation): u0
         u = 100
         # On lui applique la relation de récurrence pour avoir la deuxième valeur u1
         u = 0.4*u+10
         # On recommence jusqu'à atteindre u3
         u = 0.4*u+10
         u = 0.4*u+10
         # On affiche la valeur obtenue
         print(u)
```

```
22.0
```

- Modifier le programme précédent pour calculer u_5
- Rreprendre le programme précédent pour calculer v_5 pour la suite v_n définie par

$$v_0 = 10$$

et

$$u_{n+1} = 5u_n - 3$$

1.5 Boucles

Le problème c'est que si l'on veut calculer u_{100}, il faudra recopier un grand nombre de fois la formule de récurence ce qui n'est pas très astucieux...

Pour éviter d'avoir à recopier plusieurs fois la même chose. Quand il a des répétitions à faire, nous utilisons des **boucles**.

Pour faire des boucles on utilise l'instruction *for*

```
In [7]: for i in range(10):
            print("bonjour")
```

```
bonjour
bonjour
bonjour
bonjour
bonjour
bonjour
```

```
bonjour
bonjour
bonjour
bonjour
```

La fonction **range** va enumérer les nombres de 0 jusqu'à la valeur entre parenthèses.

- Transformer le programme pour qu'il repète 20 fois le mot bonjour
- Remplacer "bonjour" par i pour voir les valeurs prises par i

On remarquera qu'à la fin de la ligne **for**, il y a deux points puis que ce qui doit être répété est indenté (avec la touche **tab**).

Dans l'exemple suivant nous allons faire plusieurs choses dans la boucle.

```
In [8]: un_mot = "bonjour"
        un_nombre = 2
        for i in range(10):
            print(i)
            print(un_mot)
            un_nombre = un_nombre + 1
        print("fin de la boucle")
        print(un_nombre)
```

```
0
bonjour
1
bonjour
2
bonjour
3
bonjour
4
bonjour
5
bonjour
6
bonjour
7
bonjour
8
bonjour
9
bonjour
fin de la boucle
12
```

- Recopier le programme précédent puis l'adapter pour calculer u_{100}
- Écrire un programme qui calcule le n-ième terme de la suite (w_n) définie par

$$w_0 = 3$$

et

$$w_{n+1} = -2w_n + 2$$

- Écrire un programme qui calcule le n-ième terme de la suite (t_n) définie par

$$t_0 = 1$$

et

$$t_{n+1} = t_n^n + 1$$

- Écrire un programme qui calcule la somme des n premiers termes de la suite (w_n)

In [ ]: