Mon Petit Python

Coding Club by Epitech

3 juillet 2017

Table des matières

1 Introduction

Bonjour, je suis heureux de vous accompagner aujourd'hui pour votre apprentissage !

Python est un langage de programmation, dont la première version est sortie en 1991 sur Macintosh. Son créateur, Guido van Rossum, travaillait à cette époque au Centrum voor Wiskunde en Informatica aux Pays-Bas. En 2001 est lancée la Python Software Foundation, une organisation à but non lucratif détentrice des droits de propriété intellectuelle du Python. Ce langage a été baptisé ainsi en hommage à la troupe de comiques les « Monty Python ».

Le Python est très utilisé de nos jours, en raison de à sa rapidité d'exécution et de sa facilité d'apprentissage. Il existe deux versions majeures de Python utilisée : Python 2 et Python 3. Aujourd'hui, nous allons utiliser la version la plus récente et la plus recommandée : Python 3.

2 Premiers pas sur Linux

Comme vous pouvez le voir, nous sommes ici sur le système d'exploitation Linux. Si vous n'êtes pas familier avec ce système, voici les commandes basiques à utiliser dans un terminal de commandes. Elles vous seront utiles tout au long du cours :

- ls : permet de lire le contenu du dossier dans lequel vous vous trouvez

- cd nomdudossier : permet de se déplacer dans le dossier choisi

- emacs -nw nomdufichier : permet de créer et d'éditer un fichier

Voilà, avec ses trois commandes, vous êtes prêt à vous lancer !

Remarque : sur emacs, il faut faire la manipulation de touches ctrl+X ctrl+S pour sauvegarder et la manipulation de touches ctrl+X ctrl+C pour quitter.

3 Mon Premier Programme

Bien, nous allons commencer par créer un simple programme d'affichage. Pour ce faire, il faut d'abord créer le fichier "helloword.py" à l'aide de la commande emacs donnée précédemment. Bien, une fois sur l'éditeur de texte, écrivez la chose suivante :

```
1 print("Hello World")
```

puis sauvegardez et quittez emacs. Bien, maintenant que votre programme est prêt, il ne reste plus qu'à l'exécuter. Pour ce faire, veuillez faire la commande suivante :

```
python3 helloworld.py
```

. Et voilà, vous avez créé et exécuté votre premier programme. Mais qu'avons-nous fait exactement ? Eh bien, nous avons fait appel dans notre programme à la fonction print, capable d'afficher sur la sortie standard une chaîne de caractère entrée en paramètre (ici : "Hello Word").

Remarque : Les chaînes de caractères doivent tout le temps être placées entre "".

4 On est parti

4.1 Les variables

1) Les variables

Maintenant que vous pouvez afficher du texte, nous allons voir comment conserver des données. Pour ce faire, nous utilisons des espaces mémoire appelés variables. On peut voir une variable comme une boîte dans laquelle on range des données pour les réutiliser par la suite. On peut assigner une valeur à une variable avec le symbole '='.

```
1 A = 5
2 b = 5 + 5
3 str = "Bonjour"
```

Ici, 'a' prend la valeur 5, b prend la valeur 10 et str prend la valeur de la chaîne "Bonjour".

4.2 Les Tableaux

Un tableau est une variable contenant un ensemble de données. Chaque donnée est une à une placée dans les cases du tableau. Pour ensuite prendre une donnée, il faut utiliser l'index de la case comme montré dans l'exemple suivant :

```
1 tab = [5, “”toto]
2 print(tab[0])
```

Remarque : L'index de la première case du tableau est à 0 et non à 1. Par exemple, pour atteindre la première case, on doit faire tab[0], pour le second tab[1].

Il est donc possible de faire un tableau dans un tableau, ce qu'on appelle un tableau à double entrée, dans ce cas.

```
1 tab = [[5, “”tata], [7, “”toto]]
2 print(tab[0][0])
3 print(tab[1][0])
```

Remarque : Du coup, pour atteindre ces données, il faut faire l'index que l'on veut sur le petit tableau par rapport au grand. Par exemple, pour atteindre ici le '7', il faut faire tab[1][0]. Pour atteindre "tata", il faut faire tab[0][1].

Bien, nous allons maintenant faire un mini exercice. Le but est simple, vous devez afficher les caractères des tableaux suivants : tab1 = ["100", "Bonjour", 10], tab2 = [30, "Hello World", "20"] en alternant entre les deux tableaux.

```
>>>python3 helloworld.py
100
30
”Bonjour”
”Hello World”
10
20
```

.

4.3 Les Conditions

Bien, maintenant que vous savez manipuler des variables, nous allons voir ce qu'est une condition. Une condition permet de comparer des données entre elles pour ensuite exécuter un code précis en fonction du résultat.

```
1 if (a == 5) :
2           print“(a = ”5)
3 elif (a == 6) :
4           print“(a = ”6)
5 else :
6           print“(’nest pas égale à 5 ou ”6)
```

Ici, vous pouvez voir les trois mots clés utilisés pour les boucles : - le "if" : première comparaison à exécuter si elle est réalisée. Dans ce cas, le code écrit dans la ligne en dessous sera exécuté. On peut le traduire par "si"

- le "elif" : Deuxième comparaison à exécuter si la première n'est pas réalisée. On peut le traduire par "sinon si"

- le "else" : partie du code à exécuter si ni la comparaison du "if" ou du "elif" ne sont réalisés. On peut le traduire par "sinon"

Il existe plusieurs symboles de comparaison. Ici, vous avez pu voir le symbole de la comparaison "==" permettant de voir si deux variables sont égales. Voici les autres :

- != : vérifie si les deux variables ne sont pas égales

- > : vérifie si la première variable est supérieure à la seconde

- < : vérifie si la première variable est inférieure à la seconde

- >= : vérifie si la première variable est supérieure ou égale à la seconde

- <= : vérifie si la première variable est inférieure ou égale à la seconde

Remarque : Attention, il ne faut pas confondre le symbole "=" permettant d'assigner une valeur à une variable et le symbole "==" permettant de faire une comparaison.

Bien, maintenant, nous allons faire un mini exercice : Le but est simple, vous devez faire un programme qui demande à l'utilisateur un nombre. Vous devrez alors dire si le nombre est supérieur, inférieur ou égale à 50.

```
>>> pyhton3 condition.py
veuillez entrer un nombre : 60
le nombre est supérieur à 50

>>> pyhton3 condition.py
veuillez entrer un nombre : 35
le nombre est inférieur à 50

>>> pyhton3 condition.py
veuillez entrer un nombre : 50
le nombre est égale à 50
```

Remarque : pour cet exercice, vous devez utiliser la fonction "input". Voici un petit exemple de l'utilisation de la fonction :

```
1 var = input"(veuillez entrer un nombre :")
2 print(var);
```

```
>>>python3 input.py
veuillez entrer un nombre : 70
70
```

4.4 Les boucles

Bien, après les conditions, nous nous attaquons aux boucles. Une boucle permet de répéter une partie de code un certain nombre de fois.

La boucle la plus utilisée (et celle que vous utiliserez aujourd'hui) est la boucle while. La boucle while se répète tant que la condition donnée n'est pas remplie.

```
1 i = 0
2 while (I < 10) :
3           print(I)
4           I = i + 1;
```

Ici, on va boucler dans le while jusqu'à ce que i soit inférieur à 10.

Nous allons maintenant faire un petit exercice sur les boules. Vous devez afficher une à une toutes les données présentes dans un tableau.

Par exemple avec le tableau : tata = [1, 2, 4, 10, 15]

```
>>> pyhton3 boucle.py
1
2
4
10
15
```

4.5 Les fonctions

Bien, jusqu'à maintenant, nous écrivions toutes nos lignes de code en vrac dans le fichier. Il est donc maintenant temps de tout réorganiser dans ce que l'on appelle des fonctions. Une fonction permet d'exécuter certaines lignes de code en évitant les répétitions.

Voici un exemple simple de fonction :

```
1  def fonction1 (nb) :
2          return (nb + 5)
3
4  def fonction2 (nb) :
5          return (nb + 1)
6
7  if __name__ == '__main__' :
8          a = 5
9          b = 10
10         res1 = fonction1(b)
11         print(res1);
12         res2 = fonction2(a)
13         print(res2)
```

Pour commencer à vous expliquer, un programme en python commence par une fonction main. On peut la trouver facilement dans un code grâce à la ligne "main".

Comme vous pouvez le voir, cette fonction main fait appel aux fonctions 1 et 2 définies plus haut grâce au mot clef def. Pour faire simple, quand la fonction main fait appel à une des autres fonctions, alors le code appelé est exécuté jusqu'à :

-ce que la fonction finisse les lignes de commande à exécuter

-ou que l'on tombe sur le mot clé "return".

Le mot clé "return", lui, permet de renvoyer une variable à la fonction qui a appelé celle dans laquelle nous sommes. Par exemple, ici la fonction 1 renvoie la variable nb + 5 à la fonction main. Et voilà, je vous ai appris tout ce qu'il y a à savoir sur le python pour réussir l'exercice du jour. Maintenant, c'est à vous de jouer !!!

5 Morpion

Bien, maintenant que vous connaissez les bases du python, nous allons vous faire faire un petit jeu : le morpion.

Le morpion est un jeu dans lequel deux joueurs inscrivent à tour de rôle leur symbole (souvent des O et des X) dans une grille composée de 3 colonnes et 3 lignes. Le vainqueur est le joueur qui réussit en premier à aligner 3 de ses symboles que ce soit en diagonale, verticalement ou horizontalement. Si la grille est plaine sans qu'aucun joueur n'aligne ses 3 pions, la partie se finit en match nul.

```
>>> morpion.py
   |     |
_____________
   |     |
_____________
   |     |
au tour du joueur 1 : 2
   |  0  |
_____________
   |     |
_____________
   |     |
au tour du joueur 2 : 5
   |  0  |
_____________
   |  X  |
_____________
   |     |
au tour du joueur 1 : 1
 0 |  0  |
_____________
   |  X  |
_____________
   |     |

(_____________________________________)

au tour du joueur 2 : 7
 0 |  0  | X
_____________
 0 |  X  |
_____________
 X |     |
Félicitation, le joueur 2 à gagner !!!
```