Python - Cours 1

Introduction, notion de fonctions, instructions `if`, boucles `for` et `while`

I L'environnement Thonny

A l'ouverture de Thonny, plusieurs fenêtres peuvent apparaître. L'affichage peut être changé dans l'onglet "Affichage" ; mais par défaut, l'onglet utilisé pour écrire du script (l'éditeur de texte) est toujours visible. On affichera également l'onglet "Console" qui permet essentiellement deux choses :

1. Utiliser Python comme une calculatrice.
2. Interagir avec le programme exécuté via des commandes d'entrées et sorties.

Les lignes de la console commencent par >>> alors que les lignes de codes sont numérotées.

II Variables & opérations

En programmation, une variable est une case mémoire dans laquelle on stocke une information. En Python, cette information peut être de plusieurs types :

- chaîne de caractères : `str` (string)
- nombre entier : `int` (integer)
- nombre flottant (nombre à virgule) : `float`
- booléen (`True` ou `False`) : `bool`
- liste : `list`

> ☞ Pour info...
> On peut demander le type d'une variable avec la commande `type()`.

> ★Subtile...★
> En informatique, un nombre flottant n'est pas un décimal... Et certains décimaux (exemple 0,1) sont donnés en valeur approchée. Pour mieux comprendre cela, on pourra s'intéresser à la façon dont les nombres sont écrits en binaire et en écriture flottante...

Attention au nom des variables :

- Python distingue les minuscules des majuscules : la variable *x* n'est pas la même que la variable X.
- Le nom d'une variable doit toujours débuter par une lettre.

L'affectation des variables se fait avec le signe "=" (inutile de les déclarer au préalable).
Opérations sur les nombres (`int` ou `float`) :

+	-	*	/	**
addition	soustraction	multiplication	division	puissance

> Petite remarque
> On écrit 2.7 et pas 2,7 (écriture anglo-saxonne des décimaux) !

Tests logiques :

==	<	<=	>	>=	!=

> Important !
> On se souviendra du "==" pour tester l'égalité de deux quantités (qui peuvent également être du type `str`).

Tester les lignes suivantes une par une, directement dans la console :

```
>>> a=1     #Affecte 1 à la variable a
>>> a=a+1     #Calcule a+1 et affecte cette valeur à a (incrémentation)
>>> a+=1     #Incrémente a de 1
>>> b,s=4,"Truc"     #Affecte simultanément 4 à b et le texte "Truc" à s
>>> a==b     #Teste l'égalité a=b
>>> a<b
>>> a>=b
>>> a==b and a<b
>>> a==b or a<b
>>> not a==b
>>> t="truc"
>>> s==t
>>> u="Truc"
>>> s==u
```

> Utile !
> • On utilise "#" pour commenter une ligne : le texte écrit après ce symbole n'est pas pris en compte dans l'exécution du code.
> • On utilise la commande `del()` pour effacer le contenu d'une ou plusieurs variables.

> Petite remarque
> Les `and` et `or` ont exactement les mêmes significations que les connecteurs logiques et et ou en mathématiques.

III Entrées et sorties

Les commandes `input()` et `print()` permettent d'interagir avec la console. Du texte peut être indiqué (entre " " ou ' ') pour rendre la communication plus fluide.

Exécuter les lignes suivantes et commenter :

Python 1 – Entrées & sorties 1

```
x=input("Entrer une valeur de x : ")
y=input("Entrer une valeur de y : ")      # Demande de saisir une valeur de x
print(x)          # Affiche la valeur de la variable x
print("x")        # Affiche le texte "x"
print(x+y)
```

Commentaires :

Par défaut, une variable saisie après un `input` est du type `str`. Si l'on souhaite qu'elle soit un entier ou un flottant, on écrit `x=int(input("..."))` ou `x=float(input("..."))`.

Exécuter les lignes suivantes en prenant différentes valeurs de x (entier, décimal, rationnel) :

Python 2 – Entrées & sorties 2

```
x=int(input("Entrer une valeur de x : "))        #Convertit l'entrée en float
y=int(input("Entrer une valeur de y : "))
print(x+y)
```

Commentaires :

Hors programme...

La commande `eval()` permet d'exécuter des chaînes de caractères comme des instructions... Donc on pourrait utiliser `x=eval(input("x=?"))` si on souhaite saisir un rationnel comme valeur de x...

Exécuter les lignes suivantes :

Python 3 – Entrées & sorties 3

```
prenom=input("Quel est ton prénom ?")
print(prenom)

#Concaténation :
print("Bonjour",prenom)
print('Bonjour',prenom)
print("Bonjour",prenom,", bienvenue !")
print("Bonjour "+prenom+", bienvenue !") #Saisir les espaces dans la chaîne de texte
```

IV Les bibliothèques

Au fur et à mesure de l'année, nous serons amenés à utiliser certaines commandes, existantes en Python, mais qui ne sont implémentées qu'en faisant appel à certaines bibliothèques les contenant. Pour importer L'ensemble d'une bibliothèque et la renommer avec un raccourci :

```
import ... as ...
```

Voici les bibliothèques que nous utiliserons le plus souvent :

- `numpy` : contient entre autres les fonctions mathématiques usuelles ; mais également des fonctions qui seront utiles pour effectuer des calculs sur les listes et les matrices.
- `numpy.linalg` : quand nous aurons à manipuler les matrices et systèmes linéaires.
- `numpy.random` : pour simuler des expériences aléatoires.

- `matplotlib.pyplot` : pour des représentations graphiques de fonctions, de suites et de séries statistiques.
- `pandas` : pour manipuler et déterminer des indicateurs sur des séries statistiques.

Nous ne nous attarderons pas davantage ici ; nous rappellerons les différentes bibliothèques ainsi que les fonctions nécessaires en cas de besoin.

V Les fonctions

En programmation, une fonction est une suite d'instructions, dépendant éventuellement d'un ou plusieurs paramètres, que l'on nomme afin de pouvoir l'exécuter lors de son appel.

Petite remarque
Comme pour les variables, on essaiera de nommer les fonctions de façon relativement explicite et claire. Par exemple, ne pas hésiter à nommer `moyenne` une fonction qui renvoie la moyenne de plusieurs nombres.

Python 4 – Fonction avec `print`

```
def bonjour(prenom):
    print("Bonjour",prenom,"bienvenue !")
```

Dans cet exemple, la fonction `bonjour()` ne *renvoie* rien, elle ne fait qu'*afficher*... Voyons la différence entre ces deux notions avec un exemple de fonction en mathématiques.

Exécuter les lignes suivantes :

Python 5 – `print` & `return`

```
def f(x):
    return x**2+x-1

def g(x):
    if x==0:
        y=-1
    else:
        y=3
    return y
```

puis, dans la console :

```
>>> f(0)
>>> g(0)
>>> f(0)==g(0)
```

Commentaires :

Important !
Si l'on souhaite que `fonction(paramètre)` prenne une certaine valeur, alors il est nécessaire d'utiliser `return`.

À retenir...
Remarquons également qu'il n'y a pas d'instruction pour marquer la fin de la fonction. Ce sera également le cas pour les instructions `if`, `for`, `while`... Tout est dans l'**indentation** !

On retiendra au passage la syntaxe :

```
def nomdefonction(param1,param2,...):
    instructions
```

Comme en mathématiques, la ou les paramètres d'une fonction Python sont des variables *locales*. Si l'on souhaite utiliser une variable en dehors de la fonction, il est nécessaire qu'elle soit *globale*.

VI Instruction `if`

Cette commande porte bien son nom ; et voici sa structure ainsi que la syntaxe :

```
if condition1 :
    instructions1
elif condition2 :
    instructions2
...
else :
    instructions
```

Il peut y avoir un nombre quelconque de parties `elif` ; et la partie `else` est facultative.

Petite remarque
`elif` est l'abréviation de "`else if`", qui signifie "ou, si...".

Python 6 – Fonction définie par morceaux

```
def f(x):
    if x<0:
        y=2*x+3
    elif x<2:
        y=3-x
    else:
        y=x*x-3
    return y
```

Commentaires :

VII Boucles `for` et `while`

- La boucle `for` permet de parcourir les éléments d'une séquence (liste [1] ou chaîne de caractères par exemple), de donner aux variables les différentes valeurs parcourues et d'exécuter des instructions en fonction. C'est une boucle bornée.
- La boucle `while` est une boucle non bornée. Elle permet l'exécution d'instructions tant qu'une condition est vérifiée.

[1] Nous reviendrons en détails sur les listes lors du prochain cours.

VII.1 Boucle `for`

Python 7 – `for`

```
L=[11,12,13,14,15]
for i in L:
    i=i**2
    print(i)
```

Commentaires :

Python 8 – `for`

```
for k in "truc":
    print(k)
```

Commentaires :

Python 9 – `for`

```
for n in range(10):
    x=7*n
    print(x)
```

Commentaires :

À RETENIR...

La commande `range(n)` permet d'itérer sur les valeurs entières de 0 à $n-1$ (il y a 10 valeurs, en partant de 0). Plus généralement, la commande `range(a,b)` permet d'itérer sur les valeurs entières de a (inclus) à b (exclu) ; et la commande `range(a,b,p)` permet d'itérer sur les entiers de a (inclus) à b (exclu) avec un pas de p (pouvant être négatif).

On retient la syntaxe :

```
for k in sequence:
    instructions
```

VII.2 Boucle `while`

Python 10 – `while`

```
n=0
while 2**n<1000:
    n=n+1
print(n)
```

Petite remarque

La ligne `n=n+1` peut être remplacée par `n+=1`.

Que se passe-t-il si l'on remplace `2**n<1000` par `2**n>1000` ? Et si l'on oublie d'**incrémenter** n ?

Commentaires :

●○○○ Exercice 1 - Échange

Écrire un programme qui permet d'échanger les valeurs de deux variables.

●○○○ Exercice 2 - Fonction mathématique

On considère la fonction $f : x \longmapsto \begin{cases} \frac{1}{x} & \text{si } x < -1 \\ x^2 & \text{si } x \in [-1;0[\\ 2 & \text{si } x = 0 \\ e^x & \text{si } x > 0 \end{cases}$.

Définir une fonction `Python` qui prend en argument d'entrée un réel x et qui renvoie $f(x)$.

●○○○ Exercice 3 - Discriminant

Écrire une fonction prenant en arguments d'entrée les coefficients a, b, c du polynôme $ax^2 + bx + c$ et renvoyant ses éventuelles racines.

●○○○ Exercice 4 - Moyenne

1. Écrire une fonction qui prend en arguments trois notes et renvoie la moyenne (coefficients égaux à 1) de ces trois notes.
2. Écrire une fonction qui demande de saisir trois notes ainsi que les coefficients associés et qui renvoie la moyenne pondérée de ces trois notes.

●○○○ Exercice 5 - Passage, rattrapage ou redoublement ?

On définit des seuils de passage dans la classe supérieure selon les critères suivantes :

- Si la moyenne est inférieure ou égale à 6 : redoublement.
- Si la moyenne est dans]6;9] : rattrapage.
- Si la moyenne est strictement supérieure à 9 : passage.

Écrire un programme qui demande à l'utilisateur son prénom et sa moyenne, puis affiche la situation dans laquelle il se situe.

> **Petite remarque**
> On peut combiner les exercices 4 et 5 pour faire un programme qui demande les notes, et renvoie le résultat de passage.

●●○○ Exercice 6 - Table de multiplication

Écrire un programme qui demande un nombre à l'utilisateur et renvoie la table de multiplication complète de ce nombre selon l'affichage suivant :

```
Quelle table souhaites-tu ? 7
Table de multiplication de 7
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
```

●●○○ Exercice 7 - Compteur de lettre

Écrire un programme qui demande une lettre et un mot à l'utilisateur ; et qui affiche le nombre d'occurrences de cette lettre dans le mot choisi.

●●○○ Exercice 8 - Somme

1. Écrire un programme qui calcule et affiche $1 + 2^4 + 3^4 + ... + n^4$ pour une valeur de n demandée à l'utilisateur.
2. Écrire une fonction qui prend en argument d'entrée un entier naturel n et renvoie la valeur de $1 + 2^4 + 3^4 + ... + n^4$ en sortie.
3. Écrire un programme qui permet d'afficher le plus petit entier naturel n tel que $1 + 2^4 + 3^4 + ... + n^4$ dépasse 10^9.

●●○○ Exercice 9 - Factorielle

Écrire une fonction qui prend en argument d'entrée un entier naturel n, et renvoie la valeur de $n!$ en sortie.