TP 1 : Découverte Python

1 Présentation de Python

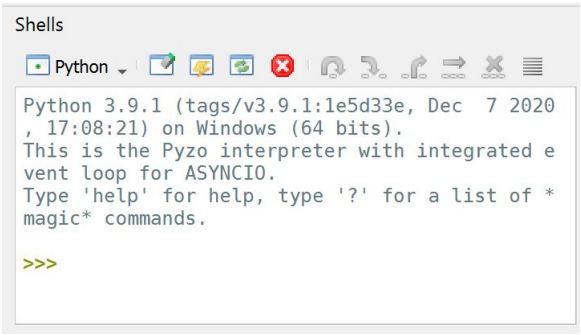
Python est un langage de programmation permettant d'écrire des séquences d'instructions, de résoudre des problèmes en fabriquant des fonctions et de la programmation orientée objet.

Ce langage fut inventé par G. van Rossum en 1989 et publié en 1991. On peut le télécharger sur le site https ://www.python.org et l'installer sur n'importe quel système d'exploitation.
C'est l'un des langages informatiques le plus utilisé dans le monde. On le retrouve dans les universités, dans l'enseignement général et dans de nombreuses industries. Youtube a été implémenté en `Python`. Il possède plusieurs avantages :

— une syntaxe simple et claire ;
— une évolution constante ;
— une certaine puissance ;
— des milliers de bibliothèques disponibles gratuitement s'adaptant à de nombreux domaines (nous en détaillerons quelques unes.

Tout au long de l'année, nous allons utiliser `Python` en étroite corrélation avec le cours de mathématiques.
On va détailler un peu les éléments qui nous permettent d'interagir avec l'ordinateur. Les *screenshots* suivant ont été faits sur l'éditeur `Pyzo` :

- **La console (ou shell).**



La console est une interface interactive que l'on peut utiliser comme calculatrice. C'est ici que l'on lira l'ensemble des résultats de nos programmes.

- **L'éditeur de text.**

C'est un espace dans lequel on peut écrire des programmes longs qui pourront être enregistrés et exécutés dans la console.

Exercice 1

Ouvrir l'éditeur et taper `print('Hello World !')`. Appuyer sur F5 pour exécuter votre programme.

2 Les variables

Dans un programme, une variable sert à désigner une zone mémoire de l'ordinateur. On y stocke une information (un nombre, une matrice, une chaîne de caractères...), on peut accéder à cette information quand

on le souhaite ou bien la changer. Elle est notée en général à l'aide d'une chaîne de caractère (lettres, caractères numériques). `Python` fait la différence entre les lettres minuscules et majuscules.
Pour introduire une variable, on utilise l'affectation. Cette opération s'effectue en utilisant la commande « = ». La valeur située à droite de ce symbole est affectée à la variable dont le nom est écrit à gauche. Par exemple, la commande `a=3` crée une variable appelée `a` qui contient la valeur 3.

Remarque : Attention, le signe égal en `Python` n'a pas le même sens qu'en mathématiques. La commande `a=3` se lit : « on range la valeur 3 dans la variable `a` ».

Il est possible de faire des affectations simultanées : `a,b=3,2` affecte la valeur 3 à `a` et 2 à `b`.

Exercice 2

Taper les commandes suivantes dans la console :

Console Python

```
>>> x=7
>>> y=x
>>> x=11
```

Que valent les variables x et y ? Expliquer.

Exercice 3

Taper les commandes suivantes dans la console :

Console Python

```
>>> x=-2
>>> y=x
>>> x=x+1
```

Que valent les variables x et y ? Expliquer.

Exercice 4

Dans l'éditeur, taper les commandes suivantes :

Python

```
1 x=2
2 y=3
3 x=y
4 y=x
```

Exécuter ce programme. Quelles sont les valeurs de x et y ? Expliquer.

3 Les types de variable

Le type d'une variable est la nature de son contenu. Voici la liste des types que nous allons rencontrer (nous préciserons la manière dont `Python` désigne ces types) :

- les entiers, désignés par `int` ;
- les nombres à virgules (pour représenter les nombres réels), désignés par `float` ;
- les booléens, désignés par `bool` (voir plus loin) ;
- les listes, désignées par `list` (voir un TP ultérieur) ;
- les chaînes de caractères, désignés par `str` ;
- les tableaux, désignés par `array`.

Remarque : En `Python`, les variables ont un type, mais le programmeur n'est pas obligé de préciser ce type. La fonction `type()` permet de connaître le type d'une variable.

Exercice 5

Taper les commandes suivantes dans la console :

Console Python

```
>>> x='bonjour'
>>> y=1
>>> z=2.0
>>> t='2.0'
>>> u=False
```

Quels sont les types des variables ci-dessus ? Expliquer.

Exercice 6

Taper successivement les commandes suivantes dans la console :

Console Python

```
>>> x=1.0
>>> type(x)
>>> x=int(x)
>>> type(x)
>>> y=1.2
>>> x=int(y)
>>> y
```

Expliquer les résultats obtenus.

4 Les opérations sur les nombres

Voici les commandes pour effectuer les opérations sur les variables de type `int` ou `float` :

- `a + b` addition.
- `a − b` soustraction.
- `a * b` multiplication.
- `a/b` division.
- `a **b` puissance a^b.

Remarque : Il est tout à fait possible d'effectuer des opérations directement avec des nombres.

Exercice 7

Taper les commandes suivantes dans la console :

Console Python

```
>>> 3+2
>>> 3*2
>>> 3**2
>>> 17/5
>>> 17.0/5.0
```

Expliquer.

Exercice 8

Créer deux variables `a` et `b` contenant toutes deux la valeur 3. Calculer le résultat de la division de `a` par `b`. On enregistrera le résultat de cette division dans une variable `c`. Déterminer le type des variables `a`, `b` et `c`. Que remarque t-on ?

Remarque : Le résultat d'une division est toujours de type `float`. Il est possible de convertir une variable `a` de type `float` (qui contient un entier) en une variable de type `int` en utilisant la commande : `int(a)`.

5 Importer des modules

Par défaut, `Python` ne connaît pas toutes les constantes ou les fonctions usuelles. Pour les utiliser, il faut au préalable importer des modules (un fichier qui contient des commandes, fonctions,...). Il y a plusieurs manières

d'importer un module, pour vous les présenter, on va utiliser le module `math` qui contient notamment les fonctions usuelles, les constantes e et π et une fonction `factorial` :

- **Première méthode :** on utilise la commande :

  ```
  import module
  ```

  On utilise alors les fonctions de ce module en le précisant devant chaque fonction comme ceci :

  ```
  module.fonction()
  ```

  Exemple :

  Console Python
  ```
  >>> import math
  >>> pi
  >>> math.pi
  >>> math.sqrt(16)
  >>> math.exp(5)
  ```

- **Deuxième méthode :** on utilise la commande :

  ```
  import module as mdl
  ```

  Cette méthode permet de ne pas avoir à taper le nom du module en entier pour appeler les fonctions :

  ```
  mdl.fonction()
  ```

  Exemple :

  Console Python
  ```
  >>> import math as m
  >>> m.e
  >>> m.log(m.exp(5))
  ```

- **Troisième méthode :** on peut importer uniquement une fonction du module :

  ```
  from module import fonction
  ```

  On utilise alors la fonction importée en utilisant directement son nom :

  ```
  fonction()
  ```

  Exemple :

  Console Python
  ```
  >>> exp(0)
  >>> from math import exp
  >>> exp(0)
  ```

- **Quatrième méthode :** on peut importer toutes les fonctions du module :

  ```
  from module import *
  ```

  On utilise les fonctions importées avec leur nom :

  ```
  fonction()
  ```

  Exemple :

  Console Python
  ```
  >>> from math import *
  >>> pi
  >>> sqrt(4)
  >>> log(1)
  >>> exp(log(6))
  ```

6 Les booléens

Les booléens sont un type de variables qui ne prennent que deux valeurs : `True` ou `False`. Ils servent essentiellement à représenter le résultats d'expressions logiques qui sont soit vraie, soit fausse.

Exemple :

Il est possible de faire des opérations sur les booléens, voici les trois qu'il faut connaître : on considère les deux booléens `A` et `B`

- `and` : l'expression `A and B` renvoie la valeur `True` si les booléens si `A` et `B` contiennent la valeur `True`, sinon, elle renvoie `False`.
- `or` : l'expression `A or B` renvoie la valeur `True` si au moins un des booléens contient la valeur `True`, sinon, elle renvoie `False`.
- `not` : l'expression `not A` renvoie la négation du booléen `A`.

Les booléens que nous croiserons viendront généralement des tests logiques suivants (a et b désignent des variables de type `int` ou `float`) :

- $a > b$ renvoie `True` si a est strictement supérieur à b, `False` sinon.
- $a < b$ renvoie `True` si a est strictement inférieur à b, `False` sinon.
- $a >= b$ renvoie `True` si a est supérieur ou égal à b, `False` sinon.
- $a <= b$ renvoie `True` si a est inférieur ou égal à b, `False` sinon.
- $a == b$ renvoie `True` si a est égal à b, `False` sinon.
- $a != b$ renvoie `True` si a est différent à b, `False` sinon.

Exercice 9

Taper les commandes suivantes dans la console :

Console Python

```
>>> import math
>>> 5==math.sqrt(25)
>>> math.sqrt(3)**2==3
```

Expliquer les résultats obtenus.

7 Les entrées/sorties

Dans un langage de programmation, on appelle entrées/sorties les commandes qui interrompent l'exécution du programme pour communiquer avec l'utilisateur. Une première commande que nous allons utiliser est la fonction `input` qui attend que l'utilisateur tape quelque chose au clavier et qui prend la valeur de la **chaîne de caractère** correspondante. On l'utilise souvent sous la forme :

Console Python

```
>>> a=input()
```

La valeur tapée sera alors affectée à la variable appelée `a`.

ATTENTION : *Cette instruction n'affiche rien dans la console, cela ne signifie pas que ça ne marche pas !*

La valeur donnée à `a` est une chaîne de caractère, si on souhaite récupérer une valeur numérique pour la réutiliser dans des calculs, on écrira `int(input())` pour obtenir un entier ou `float(input())` pour un nombre à virgule.

On peut mettre une chaîne de caractère entre parenthèse après la fonction `input`, ce qui permet d'afficher un message avant que l'utilisateur n'entre la valeur.

Une seconde commande, `print`, affiche dans la console les expressions qui lui sont données en argument. Elle peut afficher des chaînes de caractères mais aussi le contenu de variables.

Exemple :

Python
```
x=int(input('Entrer un nombre :'))
x=x+1
x=x**2*2
print(x)
```

Exercice 10

Écrire un programme qui affecte une valeur quelconque à une variable `a`, une autre à une valeur `b` et qui échange les valeurs contenues dans les deux variables. *On supposera que l'on ne connaît pas les valeurs affectées aux différentes variables.*

8 Les fonctions

Une fonction est une suite d'instructions qui dépend de paramètres.
La commande suivante :

```
def f(x):
        return ...
```

permet de créer la fonction, nommée `f`, qui à chaque variable x associe la quantité située après `return`. Il se peut que le résultat nécessite plusieurs instructions (des conditions, des boucles itératives), elles sont alors écrites avant avec une indentation par rapport au `def`.
Une fonction peut posséder plusieurs variables, l'entête devient alors : **def** `f`$(x_1,x_2,\ldots,x_n)$:

Remarques :

- La première ligne définit le nom de la fonction, les variables d'entrée.
- Le nom de la fonction ne doit contenir ni accent ni espace.
- Il est essentiel de bien conserver l'indentation, c'est elle qui permet à `Python` de différentier le corps de la fonction avec le reste.
- Il ne faut pas oublier les deux points de la fin de la première ligne !

Exemple :
Définir la fonction suivante appelée `fonctionCarree` :

Python
```
def fonctionCarre(x):
        return x**2
```

Exécuter la fonction avec F5.
Dans la console, taper les instructions suivantes :

Console Python
```
>>> fonctionCarree(6)
>>> a=5
>>> a=fonctionCarree(a)
>>> a
>>> x=3
>>> fonctionCarree(7)
>>> x
```

Exercice 11

Écrire une fonction qui prend pour argument trois réels qui renvoie leur somme.

Exercice 12

Écrire une fonction qui prend en argument un distance en kilomètre et un temps en heure, et qui renvoie la vitesse en km/h.

Exercice 13

Écrire une fonction qui prend pour argument deux réels strictement positifs correspondant aux longueurs de deux côtés d'un triangle rectangle et qui renvoie la longueur de l'hypoténuse.

Exercice 14

On considère la suite géométrique $(u_n)_{n\in\mathbb{N}}$ de raison 0.3 et de premier terme $u_0 = 2$.

1) Écrire une fonction prenant pour argument un entier naturel n et qui renvoie la valeur du terme u_n.

2) Conjecturer la limite de la suite.

Exercice 15

Écrire une fonction qui prend pour argument trois réels a, b et c et qui renvoie la valeur du discriminant associé au trinôme $ax^2 + bx + c$.