Chapitre 1 : Concepts fondamentaux

I. Présentation du langage Python

Le langage de programmation Python a été créé en 1989 par *Guido van Rossum*, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée « *Monty Python's Flying Circus* » dont van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

La dernière version de Python est la version 3. Les versions successives à la 3.0 ont aboli la compatibilité descendante avec les versions 2.x qui sont désormais obsolètes et ne sont plus maintenues.

Python Software Foundation[1] est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs.

Le langage Python est gratuit, sous licence libre.

Illustration 1: Logo de Python

1 *Python Software Foundation (https://www.python.org/psf/) est une association à but non lucratif fondée le 6 mars 2001 et dédiée au langage de programmation Python. Sa mission est de promouvoir et de protéger le langage afin d'étendre la communauté d'utilisateurs. Elle s'occupe notamment du développement du langage, de la gestion des droits des marques liées à Python et de la réunion de fonds pour le financement du projet.*

II. Installation

Dans ce cours on aura le choix entre deux environnements de développement Python : *Python IDLE*[2] et *Spyder*.

II.1) Python IDLE

II.1.A) Installation sous Windows

Sous Windows, pour installer Python avec l'environnement de développement IDLE, il suffit de télécharger puis d'exécuter le fichier d'installation qui se trouve sur le site officiel : https://www.python.org/downloads/

Notons que le logiciel d'installation est disponible en deux versions:

- Version 64 bits (x64)
- Version 32 bits (x86)

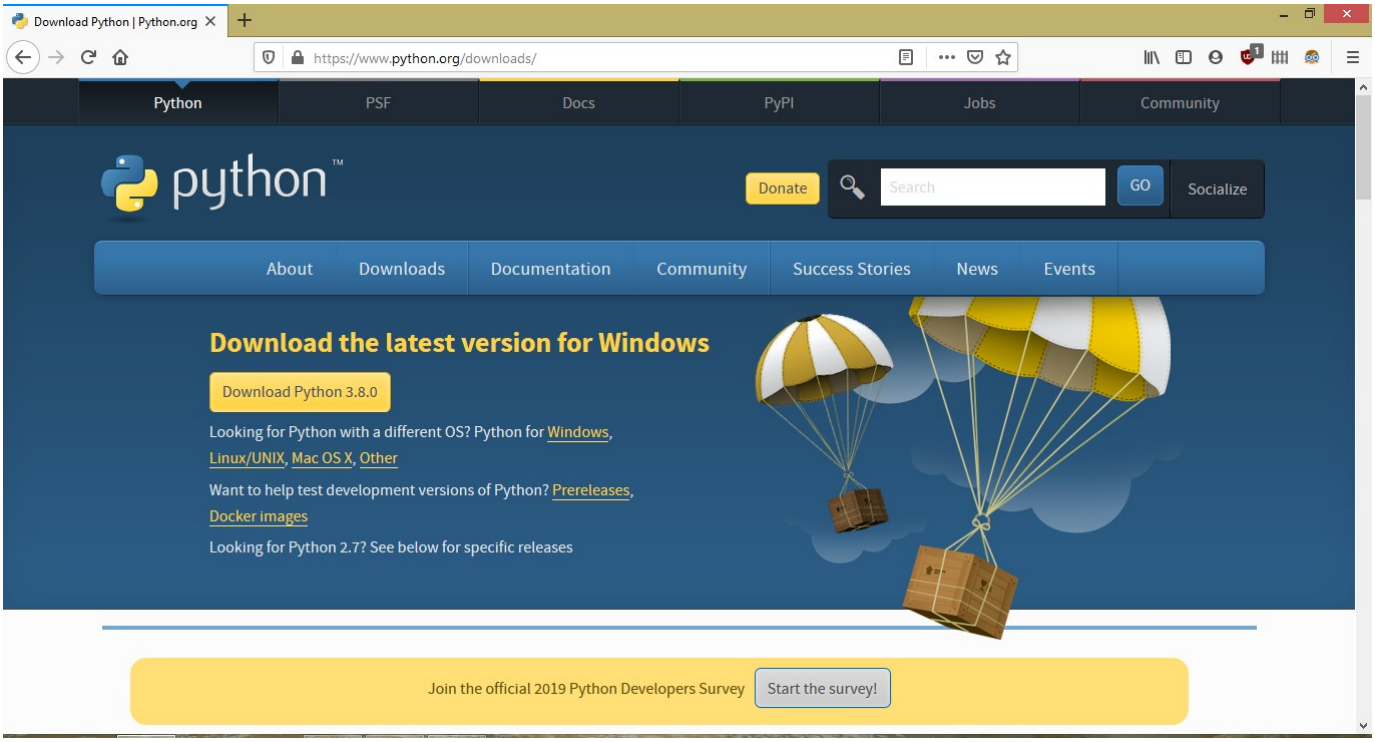

Illustration 2: Page de téléchargement du logiciel Python

2 *IDLE est l'acronyme de Integrated Development and Learning Environment*

Une fois installé, le lancement de l'IDLE se fait via : Démarrer → Programmes → Python → IDLE (Python GUI).

La figure ci-dessous présente l'interface principale du logiciel Python :

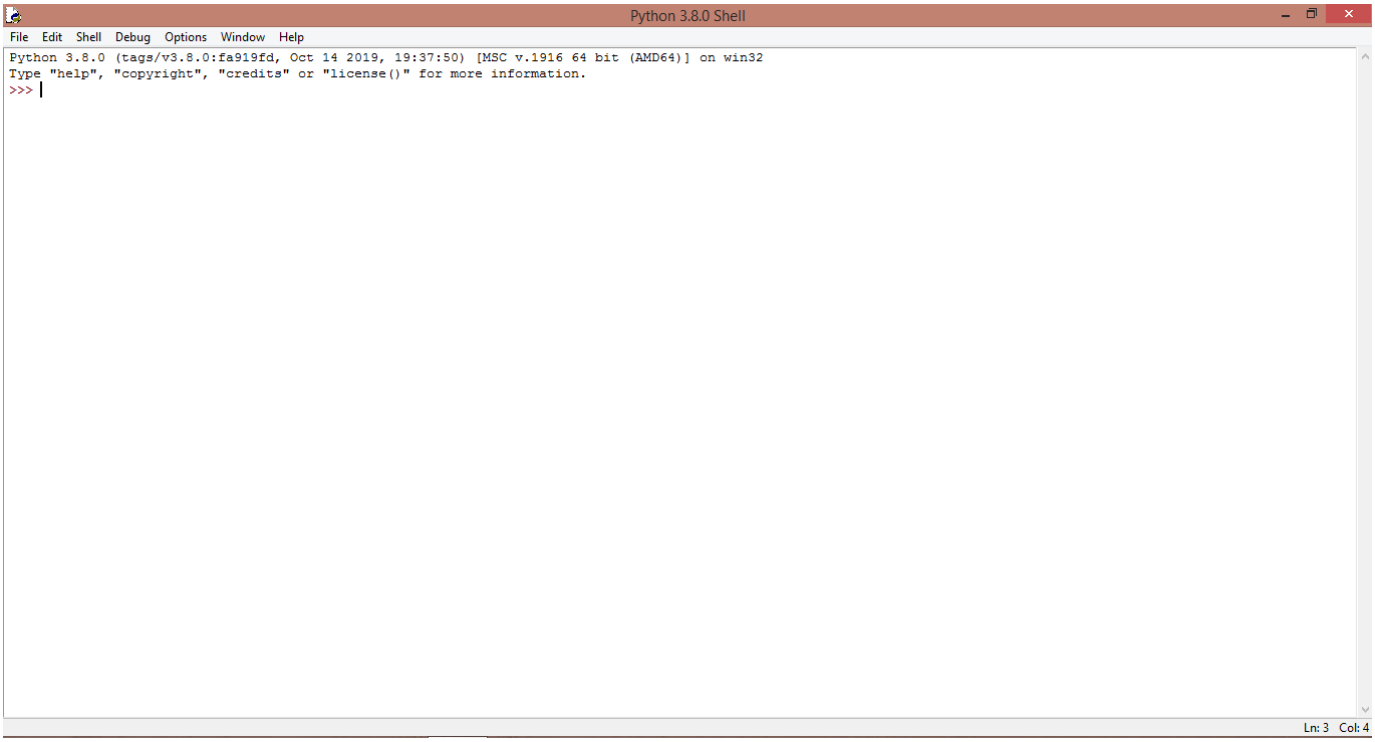

Illustration 3: L'interface d'accueil de Python IDLE

En général, un IDLE est un environnement de développement intégré *(Integrated Development Environment)* qui propose un certain nombre d'outils :

- Un éditeur de texte (pour écrire le programme) ;
- Un interpréteur (pour exécuter le programme) ;
- Un débogueur (pour tester le programme).

Il existe d'autres IDE pour Python : Eclipse/Pydev, Eric Python IDE, Spyder, etc.

II.1.B) Installation sous Linux

Dans la plupart des cas, Python est pré-installé sur la plupart des distributions Linux. Afin de vérifier la version installée, il suffit de taper la commande suivante :

```
$ python -V

Python 3.7.4
```

II.2) Spyder

L'installation d'un environnement Python complet peut être une tâche assez complexe. Tout d'abord, il faut télécharger Python et l'installer.

Par la suite, il faut télécharger un à un les packages dont on aura besoin. Parfois, le nombre de ces packages est assez élevé. Il faut tenir en compte aussi de la compatibilité entre les versions ces packages.

Spyder fait partie de la distribution Anaconda est disponible sous Windows, Linux et macOS.

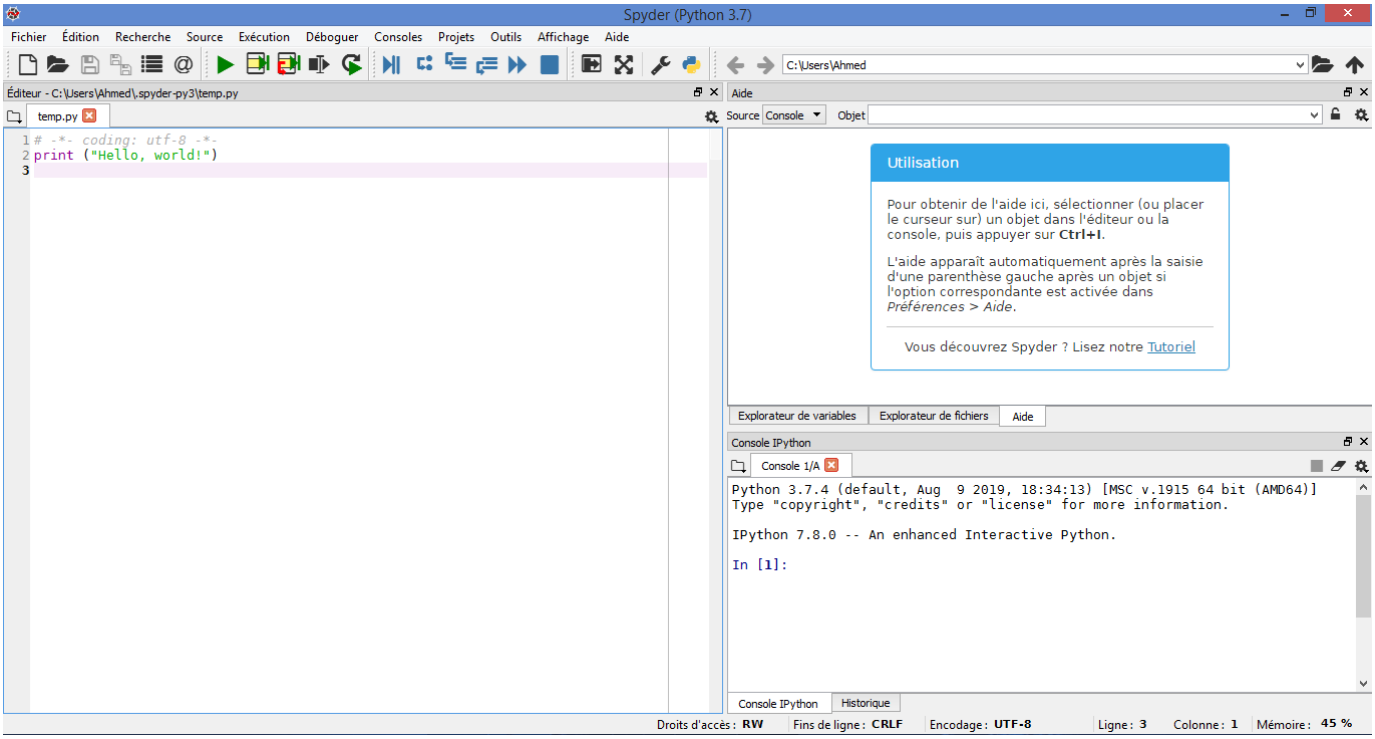

Illustration 4: L'interface d'accueil de Spyder

Anaconda est une distribution Python. Lors de son installation, Anaconda installera Python ainsi qu'une multitude de packages. Cela permet d'éviter les éventuels conflits et dysfonctionnements.

Anaconda propose aussi un outil de gestion de packages appelé *Conda*. Ce dernier permet de mettre à jour et d'installer facilement les librairies nécessaires pour le développement.

Anaconda peut être téléchargé à partir de cette adresse : https://www.anaconda.com/distribution/

NB :

L'annexe 1 présente une description plus détaillée sur le processus d'installation d'Anaconda pour Windows et Linux.

III. Caractéristiques

Python présente de nombreuses caractéristiques intéressantes :

- Il est multiplate-forme : Python est portable, non seulement sur les différentes variantes d'Unix, mais aussi sur les OS propriétaires : MacOS, BeOS, NeXTStep, MS-DOS et les différentes variantes de Windows ;
- Il est gratuit ;
- C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé ;
- La syntaxe de Python est très simple et, combinée à des types de données évolués (listes, dictionnaires, etc.), conduit à des programmes à la fois très compacts et très lisibles ;
- C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++ ;
- Il est orienté objet. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel avec un certain nombre de règles de fonctionnement et d'interactions ;
- Python intègre, comme Java ou les versions récentes de C++, un système d'exceptions, qui permettent de simplifier considérablement la gestion des erreurs ;
- Comme *Scheme* ou *SmallTalk*, Python est dynamiquement typé. Tout objet manipulable par le programmeur possède un type bien défini à l'exécution, qui n'a pas besoin d'être déclaré à l'avance ;
- Python est un langage qui évolue d'une façon continue, soutenu par une communauté d'utilisateurs dont la plupart sont des supporters du logiciel libre ;
- Il est très utilisé en bio informatique et plus généralement en analyse de données.

IV. Mode d'exécution

L'exécution d'un programme Python se fait à l'aide d'un interpréteur. Il s'agit d'un programme qui va traduire les instructions écrites en Python en langage machine, afin qu'elles puissent être exécutées directement par l'ordinateur.

Il y a deux types d'utilisation de Python :

IV.1) Le mode interactif

Dans le mode interactif, aussi appelé mode console, l'interpréteur permet d'exécuter les instructions une à une. Aussitôt une instruction encodée, il suffit d'appuyer sur la touche « ENTER » pour que l'interpréteur l'exécute.

Pour quitter le mode interactif, il suffit d'exécuter la commande *exit()*. Il s'agit d'une fonction prédéfinie permettant de quitter l'interpréteur.

IV.2) Le mode script

Dans le mode script, on doit avoir préalablement écrit toutes les instructions du programme dans un fichier texte ayant l'extension de fichier « *.py* » pour des fichiers contenant du code Python.

Une fois cela fait, l'interpréteur va lire ce fichier et exécuter son contenu, instruction par instruction. Les résultats intermédiaires des différentes instructions ne sont par contre pas affichés ; seuls les affichages explicites *(avec la fonction print, par exemple)* se produisent.

V. Applications de Python

V.1) Domaines d'application

Les domaines d'application du langage Python sont assez variées, en fait, Python peut être utilisé pour :

- Le calcul scientifique ;
- Réalisation des graphiques ;
- Le traitement du son, de la synthèse vocale ;
- Le traitement d'image ;
- La bio-informatique ;
- Les applications avec interface graphique GUI ;

- Les jeux vidéo en 2D ;
- Les applications multi touch ;
- Les applications Web ;
- Les interfaces des systèmes de gestion de base de données ;
- Les applications réseaux ;
- La communication avec des ports série RS232 ou en Bluetooth ;
- etc.

V.2) Exemples Python

- Pinterest : Site web mélangeant les concepts de réseautage social et de partage de photographies ;
- Reddit : Site web communautaire d'actualités sociales fonctionnant via le partage de signets ;
- Spotify : Service de streaming musical ;
- YouTube : Site web d'hébergement de vidéos ;
- Dropbox : Service de stockage et de partage de copies de fichiers locaux en ligne ;
- Prezi : Logiciel de présentation édité par la société hongroise éponyme ;
- Instagram : Réseau social et un service de partage de photos et de vidéos ;
- NASA : L'agence responsable de la majeure partie du programme spatial civil des États-Unis ;
- BitTorrent : Logiciel de partage de fichiers en P2P ;
- Cog: outil de génération de code ;
- ForecastWatch.com : Utilise Python pour les prévisions météo ;
- D-Link Australia : Utilise Python pour le contrôle des mises à jour des firmware de ces appareils ;
- Chez Philips, la ligne de semi-conducteurs de Fishkill fonctionne sur Python ;
- Simulation de biomolécules avec Python ;
- Miro, une application multi-plateformes de télévision par internet ;
- etc.

VI. Principes de base

VI.1) Les commandes

Une commande en Python correspond à un ordre envoyé par l'utilisateur. L'appui sur la touche « entrée » permet d'exécuter la commande saisie.

Le caractère « # » permet d'insérer des commentaires, les commentaires sont ignorés par le logiciel, ils permettent de donner des explications, commenter une commande, etc.

VI.2) Les variables

Une variable est une donnée contenant une valeur. Elle possède deux caractéristiques : un nom et une valeur.

Le nom d'une variable peut contenir des lettres, des chiffres et doit commencer par une lettre.

On évitera également d'utiliser des caractères accentués dans le nom d'une variable. Comme les espaces ne sont pas autorisés, on pourra les remplacer par un point ou un tiret bas.

De plus, le nom d'une variable ne doit pas être identique à un mot clef utilisée par Python (voir annexe 2).

Exemple :

a, rang, moyenne, variable, x1, Z3, etc.

Notons enfin que Python est sensible à la casse : ainsi « A » et « a » désignent deux variables différentes.

En Python, la déclaration d'une variable et son initialisation se font en même temps. Pour afficher la valeur d'une variable, il suffit de taper son nom.

Exemple :

```
a=6

print (a)

6
```

Il est possible d'afficher un message permettant d'initialiser une variable.

Exemple :

```
c = input("Saisir une note\n")

Saisir une note

14
```

NB :

Il faut tenir en compte que cette opération renvoie toujours une chaîne.

```
Print (type (c))

<class 'str'>

c=float (c)

print (c)

14.0
```

VI.3) Les opérateurs de base

Il existe sept opérateurs arithmétiques utilisées dans les expressions mathématiques en Python, Le tableau suivant permet de les représenter :

Description	***Notation***
Addition	+
Soustraction	-
Multiplication	*
Division	/
Division entière	//
Modulo (reste de la division entière)	%
Puissance	**

Tableau 1: Les opérateurs arithmétiques en Python

Remarque :

En Python on a :

```
print (5/3)

1.6666666666666667

print (5//3)

1

print (5%3)

2
```

VI.4) Les fonctions

VI.4.A) Principes

Une fonction est une opération réalisée par le logiciel. Une fonction se différencie par rapport à une variable par l'emploi des parenthèses après son nom. Elle peut être prédéfinie en Python ou crée par l'utilisateur.

Une fonction se caractérise par :

- Une ou plusieurs variables entre parenthèses. Ces variables sont appelées *arguments*. Il peut s'agir de n'importe quel type d'objet Python ;
- Une action : Le rôle de la fonction ;
- Résultat :Une fonction renvoie un objet Python ou rien du tout.

Exemple :

La fonction *exit()* permet de quitter la fenêtre d'exécution du logiciel.

La figure ci-dessous illustre le fonctionnement d'une fonction.

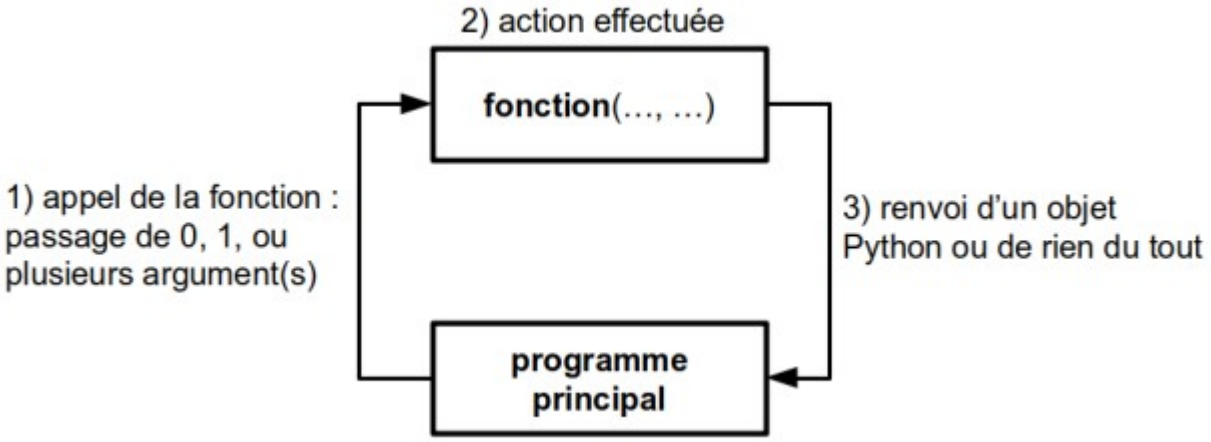

Illustration 5: Fonctionnement schématique d'une fonction

VI.4.B) Définition d'une fonction

Pour définir une fonction, Python utilise le mot-clé *def* et si on veut que celle-ci renvoie quelque chose, il faut utiliser le mot-clé *return*.

Le format général d'une fonction est :

```
def nom_fonction(liste de paramètres):

      bloc d'instructions
```

La syntaxe de *def* utilise les deux-points, un bloc d'instructions est donc attendu. L'indentation de ce bloc d'instructions *(appelé le corps de la fonction)* est obligatoire.

VI.4.C) Paramètres d'une fonction

Une fonction à besoin d'aucun ou de plusieurs paramètres ; ceux-ci doivent être placés entre parenthèses.

Exemple :

```
def carre (a):

    return a*a

a =input ("Donner une valeur")

res = carre(int (a)) #conversionn a de str → int
```

```
print (res)
```

VII. Les types des variables en Python

VII.1) Les types numériques

Il y a trois types numériques en Python :

- Le type *entier* (int) permet de représenter n'importe quel nombre entier, peu importe sa taille ;
- Le type *flottant* (float) permet de représenter des nombres comportant une partie décimale, compris entre 10^{-308} et 10^{308} (La valeur spéciale *math.inf* représente l'infini) ;
- Le type *complexe* (complex) permet de représenter des nombres complexes, où le nombre imaginaire se note $_j$.

Exemple :

```
c=2+3j

print (type (c))

<class 'complex'>
```

VII.2) Les chaînes de caractères

VII.2.A) Le type chaîne de caractères

Une *chaîne de caractères* (str) est une séquence de caractères, délimitée par des guillemets dans sa forme littérale (simple ' ou double "). On peut généralement utiliser indifféremment l'un ou l'autre type de guillemets.

VII.2.B) Opérations sur les chaînes de caractères

Python propose plusieurs fonctions permettant de gérer les chaînes de caractères parmi lesquelles on cite :

VII.2.B.a) Concaténation

La concaténation de deux chaînes se fait grâce à l'opérateur « + » qui permet de concaténer deux ou plusieurs chaînes

Exemple :

```
txt1= "Bonjour\t" #\t permet d'insérer un espace

txt2="voici un exemple Python"

print (txt1+txt2)

Bonjour Voici un exemple Python
```

VII.2.B.b) Taille d'une chaîne

La fonction *len()* permet de renvoyer la taille d'une chaîne de caractères

Exemple :

```
chaine= "On va afficher la taille d'une chaîne de caractères en
Python"

print (len (chaine))

61
```

VII.2.B.c) Découpage d'une chaîne

La fonction *.split()* découpe une chaîne de caractères en plusieurs éléments appelés champs, en utilisant comme séparateur n'importe quelle combinaison « d'espace(s) blanc(s) ».

Exemple :

```
print (chaine.split())

['On', 'va', 'afficher', 'la', 'taille', "d'une", 'chaîne',
'de', 'caractères', 'en', 'Python']
```

VII.3) Le type booléen

VII.3.A) Variables booléennes

Le type booléen se limite à deux valeurs possibles « True » ou « False »

NB :

Noter que le premier caractère est en majuscule !

Exemple :

```
a=5

b=8

r=(a<b)

print (r)

True
```

→ La variable r évalue si l'expression (a<b) est vraie ou fausse. Il se trouve qu'elle est vraie puisque 5 est inférieur à 8. Donc le résultat est True.

VII.3.B) Opérateurs booléens

En Python, on distingue 6 opérateurs booléens mentionnés ci-dessous :

Opérateur	***Rôle***
==	Test d'égalité
!=	Différence
<	inférieur
<=	Inférieur ou égal
>	Supérieur
>=	Supérieur ou égal

Tableau 2: Les opérateurs booléens en Python

VII.4) Conversion de type

Il est possible de *convertir explicitement* une donnée d'un type vers un autre, en utilisant des fonctions prédéfinies. Les fonctions *int(), float(), complex()* et *str()* permettent de convertir une donnée vers les types correspondants.

La conversion de type appelée aussi *cast* est très pratique, surtout lorsqu'on doit afficher un nombre dans une chaîne de caractères par exemple ou encore effectuer un travail sur un type donné de variables.

Exemple :

```
a=13
print (type (a))
<class 'int'>
b= complex (a)
print (b)
(13+0j)
print (type (b))
<class 'complex'>
```

VII.5) Affichage

L'instruction d'affichage d'un message en Python se fait grâce à la commande *print*.

Exemple 1 :

```
print ("Hello, world!")
Hello, world! #Texte
print(15)
15 #Numérique
print ("La valeur saisie est " +str (15))
La valeur saisie est 15
```

Exemple 2 :

```
prix=input ("Donner le prix\n")
Donner le prix
220
```

```
#Pasage type str vers float

prix = float (prix)

#Montant après remise de 5%

prix = prix * 0.95

#Affichage + conversion vers type texte

print ("le montant à payer est: " + str (prix))

le montant à payer est: 209.0
```