Chapitre 1

Python - Présentation et Prise en main

1.1 Python : un langage de programmation

Pour communiquer avec un ordinateur, il faut parler son langage, qui est le langage binaire, aussi appelé **langage machine**. Par exemple, pour dire « Bonjour » en langage machine, "il suffit" d'écrire :

0100001001101111011011100110101001101111011101010111010

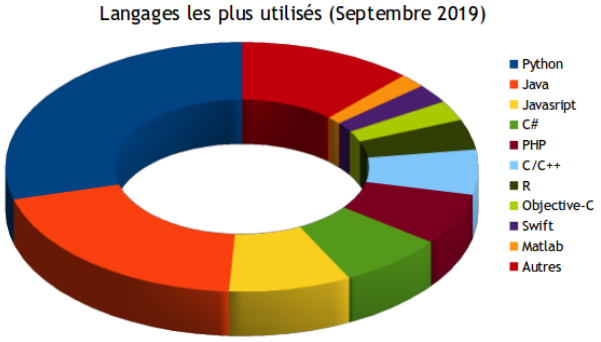
De manière évidente, cela paraît très compliqué. Avec seulement 2 "lettres" - le 0 et le 1 - il n'est pas étonnant qu'un mot aussi simple que bonjour soit si long à écrire.

Pour interagir facilement avec la machine, il convient donc de lui parler différemment, dans un langage intermédiare entre notre langue maternelle et le langage machine. Un tel langage est appelé un **langage de programmation**.

Dans un tel langage, on écrit différentes **instructions** - en **anglais** pour la plupart - en respectant une syntaxe particulère. Ces instructions sont ensuite traduites - de façon transparente - en langage machine, compréhensible par l'ordinateur.

Voici une courte liste des langages de programmation les plus utilisés :

- **Java** : développé par Sun Microsystems dans les années 90, Java est un langage de programmation populaire, utilisé pour créer des programmes et des applications mobiles (Android).
- **C/C++** : créé dans les années 70, le C est utilisé dans le développement de systèmes d'exploitation, et a posé les bases de nombreux autres langages ! Le C++ est une évolution du langage C, permettant notamment la **programmation orientée objet**.



Source : *http ://pypl.github.io/PYPL.html*

- **Objective-C** : langage de programmation généraliste utilisé par les systèmes d'exploitation et les applications Apple.
- **PHP** : utilisé en développement WEB, il permet de rendre un site Internet dynamique (gestion des utilisateurs, forums, compteur de visites...) et est notamment utilisé par WordPress ou Facebook.
- **Javascript** : également outil de développement WEB, il ne partage avec le Java que le préfixe *java*. Il est utilisé par un navigateur pour animer une page Web : menus déroulants, graphiques animés...

▷ *Et Python dans tout ça ?*

Python est un langage de programmation créé par **Guido Van Rossum**, dont la première version est sortie en 1991.
C'est aujourd'hui un langage très populaire, possédant de nombreux avantages :

- il permet une **initiation aisée** aux concepts de base de la programmation grâce à sa syntaxe relativement simple, tout en permettant la réalisation de programmes complexes, comme des jeux, des interfaces graphiques, la gestion du réseau...
- il est **multi-plateformes** : on peut aussi bien l'utiliser sous Windows, sous MacOS ou sous une distribution Linux
- il dispose d'une **importante communauté** : on trouve généralement la solution à beaucoup de problèmes sur Internet
- il est placé sous **licence libre** : l'utilisateur est libre de l'utiliser, de le modifier, de le redistribuer y compris commercialement, et tout ça gratuitement

Python est aussi un langage de programmation qualifié de **haut niveau**.
Cela signifie qu'il est « plus proche du programmeur que de la machine » : sa syntaxe est facilement compréhensible, et le programmeur ne s'occupe que du résultat final de son programme.

Dans un langage **bas niveau**, le programmeur doit gérer, en plus de son programme, toutes les ressources que ce dernier utilise : registres, adresses mémoire, instructions processeurs...

Un langage bas niveau est souvent plus difficile à utiliser qu'un langage haut niveau, mais les programmes qui en résultent sont souvent plus rapides et plus efficaces.

Logo de *Python*

Enfin, Python est un langage **interprété**. Un interpréteur traduit une à une les instructions Python en langage machine.
Contrairement à Python, le langage C (par exemple) est un langage **compilé** : un compilateur se charge de traduire l'ensemble des instructions (appelé aussi **code source**) en langage machine, dans le but d'optimiser le résultat.

On pourrait voir cela comme la traduction d'un texte dans une autre langue.
On peut traduire mot à mot, et obtenir une traduction compréhensible mais parfois mal formulée, ou traduire le texte dans sa globalité, et ainsi obtenir une traduction plus facile à lire, et donc plus rapide à comprendre.

1.2 Installation et Utilisation

Programmer en Python nécessite généralement deux choses :

1. L'**interpréteur Python** : c'est lui qui va traduire nos instructions en langage machine. Il est indispensable.
2. Un **éditeur de code** (IDE) : facultatif, mais bien utile lorsqu'il s'agira d'écrire de gros programmes. L'éditeur est semblable à un éditeur de texte, mais nous permet d'écrire du code Python et de l'interpréter.

Nous allons nous contenter dans un premier temps d'installer l'interpréteur Python.

☞ *Il existe plusieurs versions de Python : les versions 2.x, fonctionnelles mais dépassées aujourd'hui, et les versions 3.x que nous utiliserons dans ce cours.*

Installation

Sous Windows et Mac OS X

Il suffit de se rendre sur le site officiel de Python à l'adresse `https://www.python.org`.
Dans l'onglet *Download*, on choisit le système d'exploitation, puis on clique sur le lien correspondant à la version de Python choisie - on choisira la plus récente, soit la 3.7.4 en Septembre 2019 - juste en-dessous de *Stable Releases*.
On ouvre ensuite le fichier téléchargé, et on se laisse guider par l'assistant d'installation. Terminé !

Sous Linux

Python est pré-installé sur la plupart des distributions Linux. Il n'y a donc rien à faire en général.

Utilisation de Python

Nous allons commencer par utiliser Python en **mode interpréteur**, ou en **ligne de commande** en utilisant la **console Python**.

Sous Windows

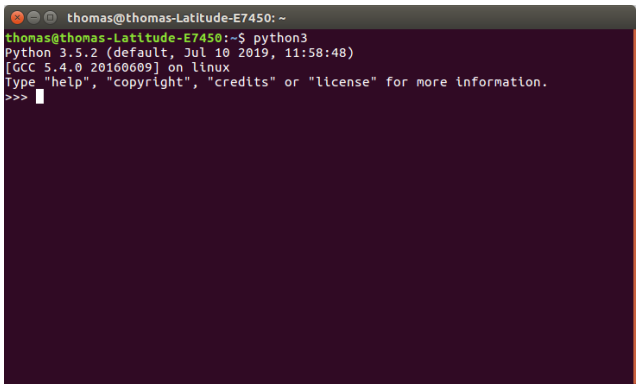
- En passant par les menus : Démarrer $\Longrightarrow$ Python 3.x $\Longrightarrow$ Python (Command Line)
- Via le raccourci Windows + R, qui permet d'ouvrir la console Windows.
  La console s'ouvre, et il suffit alors d'écrire *py* pour se retrouver dans l'interpréteur Python.

Sous Mac OS X

Il suffit d'ouvrir l'application *Terminal*, d'écrire *python3* puis de valider avec *Entrée*.

Sous Linux

Ouvrir un terminal avec le raccourci Ctrl + Alt + T, écrire *python3* puis valider.



Un terminal sous Linux (Ubuntu) donnant accès à l'interpréteur Python

1.3 Premières instructions

Dans l'interpréteur Python, il est possible de rentrer des instructions après les trois chevrons `>>>`. Pour valider ces instructions, on appuie sur la touche Entrée du clavier. L'instruction est alors traduite en langage machine par l'interpréteur puis exécutée par notre machine.

Essayons de dire bonjour à Python en écrivant *Bonjour* dans l'interpréteur :

```
>>> Bonjour
File "<stdin>", line 1
    Bonjour
           ^
SyntaxError: invalid syntax
>>>
```

Drôle de réponse, Python manque assurément de politesse. Il nous permet cependant de rentrer une commande supplémentaire en affichant à nouveau les trois chevrons. En réalité, il ne connaît ni *Bonjour*, ni *Salut*, ni *Hello*.

☞ *Comme souvent en programmation, la syntaxe et les messages d'erreurs sont en anglais. Python n'échappe pas à la règle !*

À défaut de comprendre des mots simples, Python est un formidable calculateur. Essayons quelque chose de simple :

```
>>> 1 + 2
3
```

Il nous répond, et de manière juste.

Mais Python peut faire encore mieux : il connaît les 4 opérations de base que sont l'**addition** (+), la **soustraction** (-), la **multiplication** (*) et la **division** (/).

Comme vous, il connaît les priorités de calcul, et gère les parenthèses sans problème.

```
>>> 2*4-1
7
>>> 2*(4-1)
6
>>> 1 + 1/3
1.3333333333333333
```

Dans ce dernier calcul, Python a effectué une approximation : il a tronqué le nombre $1+\frac{1}{3}=\frac{4}{3}$, qui n'est pas décimal.
On remarque aussi que comme la calculatrice, Python n'utilise pas la virgule mais le **point** comme **séparateur décimal**.

Python dispose également des 3 opérations élémentaires suivantes : le **quotient** (//) et le **reste** (%) dans une division euclidienne, ainsi que la **puissance** (**).

```
>>> 68 // 6
11
>>> 68 % 6
2
>>> 3**3
27
```

☞ *La division euclidienne de 68 par 6 donne le quotient et le reste : 68 = 6 × **11** + **2***

L'opération reste (%), également appelée **modulo**, est une opération très importante et très utilisée dans de nombreux langages de programmation. Nous en verrons des exemples plus loin dans ce cours.

Exemple. La division euclidienne de 1446 par 7 est $1446 = 7 \times 206 + 4$, d'où :

```
>>> 1446 // 7
206
>>> 1446 % 7
4
```

Exemple. Calcul de 2^{1000}, en moins de temps qu'il ne faut pour l'écrire !

```
>>> 2**1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581
27594672917553146825187145285692314043598457757469857480393456777482423098542107460506237114187795418
21530464749835819412673987675591655439460770629145711964776865421676604298316526243868372056680693376
```

Python sait travailler avec de très grands nombres ! Il ne met guère plus de temps à calculer 2^{10000}.

☞ *Attention ! Le calcul de puissances plus grandes peut demander un certain temps, et risque de bloquer l'ordinateur...*

Python sait également travailler avec des nombres binaires et héxadécimaux. Pour utiliser un nombre binaire, il suffit de le précéder par `0b`, et `0x` pour un nombre héxadécimal. Python retourne les résultats en décimal.

```
>>> 0b1100 + 0b1010
22
>>> 0b110 * 0b111
42
>>> 0xAA
170
```

1.4 QCM

Dans le QCM suivant, chaque question peut admettre **une ou plusieurs bonnes réponses**.

Questions	Réponses
1. Pour communiquer avec un ordinateur, il est nécessaire de connaître des instructions écrites en binaire.	☐ Vrai ☐ Faux
2. Python est un langage de programmation :	☐ Interprété ☐ Compilé ☐ De bas niveau ☐ De haut niveau
On répondra aux questions suivantes sans utiliser l'interpréteur Python.	
3. Que va afficher la commande suivante ? `>>> Calculer 3 * 2`	☐ `6` ☐ `5` ☐ Une erreur
4. Que va afficher la commande suivante ? `>>> (3 + 1) // 3`	☐ `1.3333333333333333` ☐ `1` ☐ Une erreur
5. Pour calculer le nombre de secondes dans une journée, on utilise la commande :	☐ `nombreSecondesJournee()` ☐ `60 * 60 * 24` ☐ `60 * 24`
6. Que va afficher la commande suivante ? `>>> 0b1100 * 0b100`	☐ `0b110000` ☐ `48` ☐ `0x30`