Leçon d'informatique : Introduction à Python 3

S. Benlhajlahsen - PCSI$_1$

Sommaire

I Présentation

I.A Qu'est-ce que l'informatique ?

La Société Informatique de France (SIF) propose la définition suivante.

> **À retenir :**
> L'informatique est la science et la technique de la représentation de l'information d'origine artificielle ou naturelle, ainsi que des processus algorithmiques de collecte, stockage, analyse, transformation et communication de cette information, exprimés dans des langages formels ou des langues naturelles et effectués par des machines ou des êtres humains, seuls ou collectivement.

I.B Information et ordinateur

En Anglais, ordinateur se dit *computer*, et le verbe *to compute* veut dire calculer. Il s'agit donc d'un dispositif qui reçoit en entrée des paramètres, sur lesquels il effectue des opérations, et il renvoie le résultat de ce calcul. Cela dit, un boulier chinois en faisait autant il y a des millénaires. Un ordinateur est un peu plus que cela :

- Il permet l'**automatisation** de calculs, en particulier de multiples répétitions d'un même type de calcul. Par exemple, quand vous retouchez une photo pour appliquer un effet de flou, le même petit calcul est répété sur chaque pixel de l'image. Il y a des millions de pixels dans une seule photo, donc hors de question de le faire à la main !
- Un ordinateur est fabriqué pour effectuer un nombre limité d'opérations différentes, mais on peut automatiser la combinaison de ces opérations de base pour effectuer des calculs plus complexes. Un ordinateur est donc **polyvalent**. Ainsi, l'effet de flou sur photo a été étudié, décomposé en étapes élémentaires dont l'enchaînement a été automatisé pour le confort de l'utilisateur.

> **À retenir :**
> L'automatisation des tâches se fait à l'aide d'instructions que l'on écrit à l'avance et que l'ordinateur exécute au moment où la tâche est accomplie. Cela s'appelle la **programmation**.

I.C Programmation

Écrire les instructions qui constituent un programme suppose :

- d'avoir un vocabulaire pour préciser les opérations à effectuer. On appelle cela des **mots-clefs** [1] : ils ont un sens précis et sont réservés à l'usage correspondant ;
- d'avoir une grammaire pour savoir comment agencer ces mots clefs de manière à constituer une phrase (une instruction) qui a un sens. On appelle cela les règles de **syntaxe**.

Attention !

Il faudra être rigoureux sur la synthaxe. Une erreur de synthaxe peut bloquer le fonctionnement de votre programme.

1. *keywords* en anglais.

II Coder l'information

II.A Exprimer une quantité d'information

L'unité élémentaire d'information est le **bit** (*binary digit*, chiffre binaire). Il n'a donc que deux valeurs possibles, 0 ou 1. Un groupement de $8 = 2^3$ bits s'appelle un **octet** (*byte* en Anglais), de symbole B [2]. On peut définir des multiples de l'octets comme pour n'importe quelle unité (par des puissances de 10), mais comme en informatique on travaille toujours en base 2, on peut aussi les définir par des puissances de 2. Remarquant que $2^{10} = 1024$:

Unité (base 2)	Valeur	Unité (base 10)	Valeur
kibi-octet (kiB)	1024 B = 2^{10} B	kilo-octet (kB)	1000 B
mébi-octet (MiB)	1024 kiB = 2^{20} B	méga-octet (MB)	1000 kB = 10^6 B
gibi-octet (GiB)	1024 MiB = 2^{30} B	giga-octet (GB)	1000 MB = 10^9 B
tébi-octet (TiB)	1024 GiB = 2^{40} B	téra-octet (TB)	1000 GB = 10^{12} B

Enfin, attention aux confusions de la langue courante. Par écrit les deux systèmes de notation sont globalement respectés (en tout cas dans le monde professionnel), à l'oral on a tendance à utiliser les appellations de la base 10 même quand on parle de la base 2. Donc prêtez attention au contexte !

II.B Codage des valeurs

La manière dont est traduite une valeur en bits dépend de la nature de la variable, son **type**. Le nombre de bits alloués gouverne le nombre maximal de valeurs différentes qui peuvent être utilisées. Les valeurs les plus courantes sont :

Espace alloué	Nombre de valeurs différentes
8 bits	$2^8 = 256$
10 bits	$2^{10} = 1024$
12 bits	$2^{12} = 4096$
16 bits	$2^{16} = 65\,536$
24 bits	$2^{24} = 16\,777\,216$
32 bits	$2^{32} = 4\,294\,967\,296$
64 bits	$2^{64} = 18\,446\,744\,073\,709\,551\,616$

Voici quelques utilisations typiques :

- 8 bits : codage d'une couleur dans les formats d'images et de vidéos (JPEG, Blu-ray…) ;
- 10 bits : code d'une couleur dans certains formats d'images professionnels ;
- 16 bits : codage des sons dans les formats audio grand public (CD, MP3…) ;
- 24 bits : codage des sons dans les formats audio professionnels (mastering) ;
- 32 bits : codage des entiers dans la plupart des langages de programmation ;
- 64 bits : codage des flottants dans la plupart des langages de programmation.

Les types simples les plus utilisés sont les entiers, les flottants et les caractères.

II.C Entiers

En première approche, on peut dire que les entiers sont stockés sous forme binaire dans l'espace mémoire alloué : l'entier n est représenté sur p bits par la suite $(a_{p-1}, a_{p-2}, \ldots, a_1, a_0)$ de valeurs 0 ou 1 telle que [3]

$$n = \sum_{k=0}^{p-1} a_k 2^k$$

Il reste le problème de la représentation des entiers négatifs, n ous y reviendront dans un chapitre ultérieur. Cela limite les entiers à un intervalle de taille 2^p. Cependant, en théorie, Python est capable d'utiliser les entiers sans limite de valeur.

2. En France, on trouve souvent la notation o à la place de B, mais elle n'est pas reconnue internationalement.
3. C'est la représentation **big endian** ou **petitboutienne**, d'autres ordres des indices existent.

II.D Flottants

La notion mathématique de réel n'a pas d'équivalent en informatique. En effet, un nombre réel nécessite souvent un processus infini pour être défini (suite de ses décimales, suite qui tend vers le réel, etc) qu'on ne peut pas contenir dans une mémoire finie. On devra se contenter d'approximations. Les valeurs approchant les nombres réels sont appelés **nombres flottants** pour rappeler que leur virgule "flotte", leur forme est semblable à la notation scientifique utilisée dans les sciences, par exemple $6{,}02 \times 10^{23}$. Si on suppose que l'on dispose de 8 chiffres significatifs le nombre précédent s'écrit $\frac{60200000}{10^7} \times 10^{30}$, il peut être représenté par les entiers 60200000 et 30.
En Python, ces valeurs ont le type `float`. On les écrit avec un **point à la place de notre virgule** ; on peut aussi utiliser un exposant `e` qui signifie puissance 10 ; `898547e-5` signifie `8.98547` Leur codage en mémoire utilise deux entiers [4], comme l'exemple ci-dessus que nous étudierons dans un chapitre ultérieur.

II.E Caractères

De même chaque caractère est associé à un nombre entier que l'on peut coder dans la mémoire. Il faut donc une table de correspondance entre les caractères et leurs représentations entières. Cela s'appelle un encodage.

ASCII L'encodage ASCII [a] est le plus ancien encodage encore utilisé, l'ASCII 7 bits, ne contient que les caractères utilisés en Anglais. Il contient $128 = 2^7$ caractères, donc chaque caractère peut être représenté par un entier sur 7 bits. En pratique, l'unité élémentaire d'information en mémoire étant l'octet, un caractère se voit donc plutôt allouer 8 bits, le 8e bit étant alors laissé à 0. On y trouve les 26 lettres de l'alphabet latin en majuscules et minuscules, les principaux symboles de ponctuation et les chiffres arabes. Quelques exemples :

Caractère	Code	Caractère	Code
A	65	0	48
B	66	1	49
a	97	(	40

a. ASCII veut dire American Standard Code for Information Interchange : Code américain normalisé pour l'échange d'information.

Remarque Il y a aussi des "caractères spéciaux" qui n'ont pas d'apparence mais sont utiles par exemple pour structurer une chaîne de caractères :

Caractère	Code	Représentation en Python
Nouvelle ligne	10	`\n`
Tabulation	9	`\t`

Remarque Par contre, l'ASCII 7 bits ne contient pas de caractères accentués, de lettres grecques, etc [a]. Du coup, dans la deuxième moitié du 20ème siècle, chaque pays a utilisé le 8e bit pour étendre l'ASCII et y ajouter les caractères nécessaires pour sa ou ses langues. Mais ces efforts n'ont pas été normalisés, conduisant à de multiples encodages variant selon les pays et les types d'ordinateurs, générant un casse-tête d'interopérabilité

a. Ces limitations se retrouvent encore aujourd'hui, par exemple, dans certains communications par internet : il est bien connu qu'utiliser des lettres accentuées dans des mails ou des noms de fichier quand on communique avec des gens d'autres pays peut parfois créer des problèmes

L'Unicode Il a fallu attendre les années 2000 pour voir se concrétiser un standard international recouvrant la grande majorité des besoins de toutes les langues : l'**Unicode**. De nos jours, l'Unicode est l'encodage par défaut de la majorité des ordinateurs usuels. Tous les langages de programmation modernes savent manipuler des caractères en Unicode [a]. Dans la variante **UTF-8** de l'Unicode (la plus répandue), l'espace mémoire alloué à un caractère est variable, allant de 1 à 4 octets.

a. Python, depuis sa version 3, travaille nativement en Unicode.

III Pratique de la programmation

III.A Code source et langage machine

Développer, c'est écrire un programme. Le plus souvent, votre travail sera d'**implémenter** un algorithme : traduire sa construction logique dans le langage choisi. Ce que vous écrivez alors s'appelle le **code source**.

4. mantisse et exposant

Cependant, l'ordinateur ne peut pas comprendre un code source ! En effet, par construction, il ne peut comprendre qu'un langage constitué d'une longue suite de 0 et de 1 (binaire) : le **langage machine**. Il va donc falloir convertir le code source en langage machine. Il y a deux grandes approches :

- la **compilation** : la conversion doit être faite manuellement avant l'exécution ; On génère alors un fichier intermédiaire appelé fichier binaire ou **bytecode** ;
- l'**interprétation** : la conversion est faite « à la volée » (automatiquement) au moment de l'exécution.

III.B Evolution des langages

Les langages ajoutent une nouvelle interface entre les idées du programmeur et leur exécution par l'ordinateur. Voici quelques étapes de l'apparition des nouveaux langages.

- Années 50 : langages spécialisés FORTRAN, LISP, COBOL
- 1958 : définition de ALGOL, qui donnera l'impulsion des langages universels
- 1967 : simula-67 premier langage orienté objet
- 1972 : PROLOG, programmation logique
- 1973 : ML, programmation fonctionnelle (évolution de lisp)

Remarque C'est le langage C qui a le plus inspiré les langages utilisés ensuite : c'est un langage impératif de bas niveau qui se traduit très facilement en langage machine tout en offrant un système de types. Les langages successifs améliorent en terme de puissance d'expression et de facilité d'écriture les langages antérieurs. Les avantages des langages modernes sont :

- la lisibilité du code ;
- sa concision ;
- son indépendance par rapport au processeur.

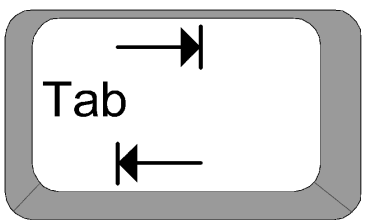
III.C Propriétés de Python

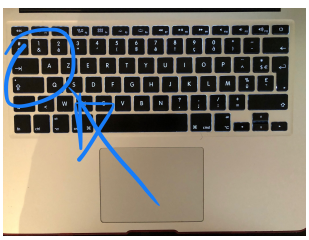
Python est : Nous allons apprendre les concepts importants de la programmation avec le langage Python.

1. un langage Open Source, disponible gratuitement sur la grande majorité des architectures ;
2. qui s'appuie sur une importante communauté de développeurs ;
3. il contient de nombreux outils sous forme de modules qui sont intégrés (*Batteries Included*) ;
4. plusieurs milliers de packages supplémentaires sont disponibles dans tous les domaines ;
5. C'est un langage moderne, rapide dans le cycle écriture-test (interprété) et simple (indentation significative).
6. l'interpréteur Python est écrit en C, de même que certaines bibliothèques (d'autres sont écrites en Fortran) ;
7. On peut lui reprocher parfois une lenteur lors de l'exécution de programmes importants[a]. Lors du développement de projets industriels on peut être amené à ré-écrire le code (ou une partie du code) dans un autre langage.
8. Cependant, c'est un langage bien adapté à l'écriture de projets, il est devenu le standard dans les laboratoires pour écrire simplement des programmes personnalisés.

a. Les lanages compilés sous souvent plus rapides.

Vocabulaire : L'**indentation** consiste en l'ajout de tabulations ou d'espaces dans un fichier, pour une meilleure lecture et compréhension du code. Elle est obtenue à partir de la touche de symbole





Par exemple, dans le programme de la figure 1, la deuxième ligne a subie une indentation. La troisième ligne a été désindentée par rapport à la deuxième. Dans beaucoup d'éditeurs de texte, cette indentation correspond à 4 espaces

insécables.

```
1 def f(x):
2   return x
3 print('f(2) vaut', f(2))
```

FIGURE 1 – Exemple de programme avec une indentation.

IV Un premier programme

Comment « faire du Python » ? Pour écrire des instructions en Python, il y trois solutions :

- écrire les programmes dans des fichiers du type `fichier.py` puis les exécuter à travers un logiciel ;
- exécuter le programme par l'intermédiaire d'un terminal ;
- utiliser un `notebook` comme *jupyter*. Les fichiers seront alors enregistré avec du type `fichier.ipynb`.

Remarque : Dans les trois situations, on peut utiliser des solutions en ligne ou installer des logiciels sur sa machine.

IV.A Installation

Pour commencer à écrire des programmes, vous pouvez :

- installer les paquets correspondants au système explotation de la machine depuis le site officiel : https://www.python.org ou la page de la distribution anaconda qui est recommandée : https://www.anaconda.com ;
- utiliser un compilateur en ligne.

IV.B Environnement de développement

Pour programmer de manière efficace, il faut :

- un éditeur de texte ;
- un compilateur/interpréteur ;
- un debugger qui aidera à trouver les erreurs dans le programme.

Utilisation d'un IDE Un environnement de développement ou IDE (pour *integrated development environment*) regroupe l'ensemble et offre des raccourcis qui facilitent l'édition du programme.

Exemple de Thonny Thonny est le logiciel que l'on utilisera cette année pour éditer et executer des programmes (voir figure 2). Il se compose en partie d'une fenêtre d'édition dans le lequel on éditer son programme. et d'une fenêtre d'interpréteur/shell qui affichera certaines sorties du programme.

Conseil Dans un premier temps, on peut utiliser des solutions en ligne. Dans un second temps, il est vivement conseiller fortement d'installer très vite Thonny [a] sur votre machine. C'est par la pratique personnelle que vous vous préparerez le mieux au concours.

a. ou une autre solution comme anaconda.

IV.C Les premières règles de syntaxe

Syntaxe

On peut ajouter librement des espaces dans une ligne pour améliorer la lisibilité **sauf en début de ligne**.

Les deux instructions ci-dessous sont équivalentes.

```
1 a=2
2 a = 2
```

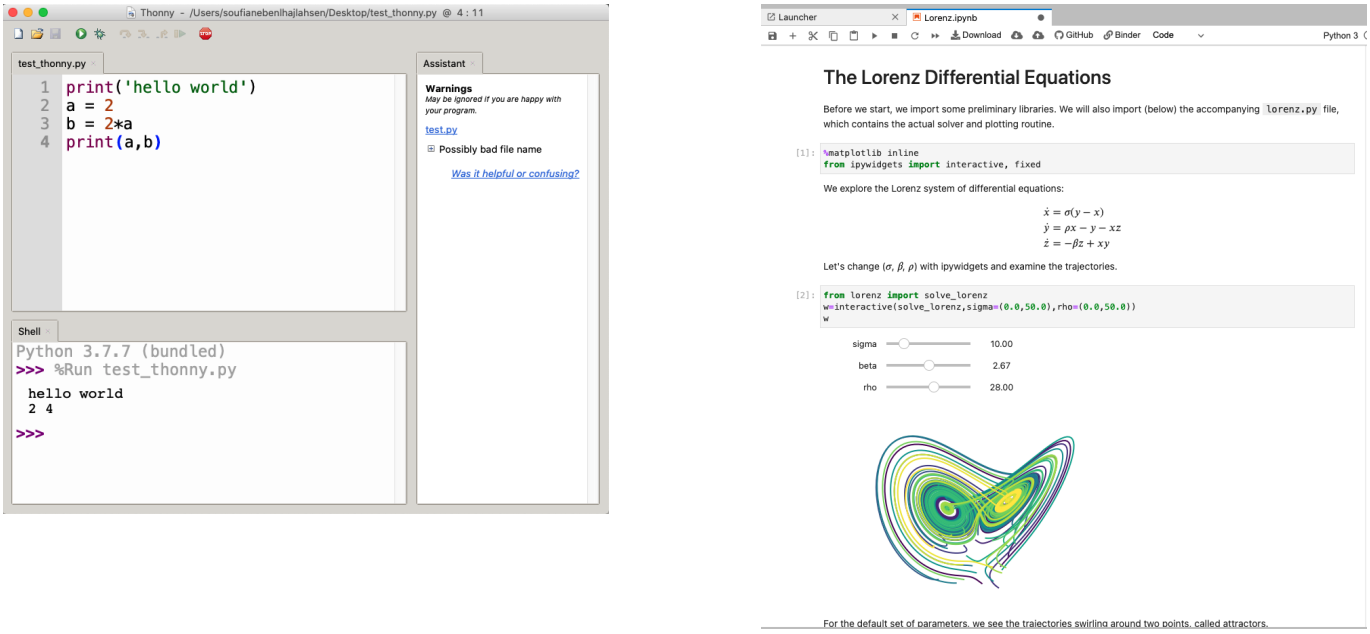


FIGURE 2 – À gauche, fenêtre graphique du logiciel thonny. À droite, un *notebook jupyter*.

Syntaxe

La fin d'une instruction est repérée par le passage à la ligne suiante.

Ce code ci-dessous constitué de deux instructions est correct :

```
a=2
print(a)
```

mais le suivant ne l'est pas :

```
a=2 print(a)
```

Syntaxe

On peut librement laisser des lignes vides entre les lignes d'instruction pour aérer le code.

```
a=2

print(a)
```

Délimiteurs

Les parenthèses `()`, les accolades `{}` et les crochets `[]` sont des délimiteurs. Les trois types de délimiteurs ne sont pas interchangeables en Python. Ils ont chacun des rôles précis.

- Les parenthèses ont leur rôle habituel pour le calcul `x*(x+1)` ou les fonctions `fonction(x)` ;
- les crochets servent pour l'extraction des éléments d'une liste ou d'une chaînes de caractère ;
- les accolades seront peu utilisées, ils servent pour les dictionnaires ou les ensembles.

Syntaxe Sur une ligne, tout ce qui se trouve après un dièse `#` est **commenté**, c'est-à-dire ignoré par Python.

```
3 a=2
4 # cette ligne est un commentaire !
5 print(a)
```
