

Chap. 1 Introduction à la programmation Python

Thème 1

Environnement Python

Sommaire du chapitre 1

- ▷ **Environnement Python**
- ▷ Types de base
- ▷ Variables et affectation
- ▷ Expressions
- ▷ Instructions
- ▷ Fonctions
- ▷ Erreurs et « bugs »

Le langage de programmation Python a été développé par le hollandais Guido Van Rossum en 1990. Très utilisé dans l'analyse de données, le développement web ou encore le *deep learning*, Python est un langage de programmation extrêmement souple et simple d'utilisation, qui permet de nombreuses applications.



Python est un langage **multiplateforme**, c'est-à-dire qu'il est disponible sous diverses architectures (compatible PC, tablettes, smartphones, ordinateur Raspberry Pi, ...) et différents systèmes d'exploitation (Windows, Linux, Mac OS, Android...).

Python est un langage **multiparadigme**, c'est-à-dire qu'il offre la possibilité de différentes approches de programmation (appelées **paradigmes**), comme par exemple la programmation **impérative** (instruction par instruction), la programmation **orientée-objet** (avec des objets munis d'attributs et de méthodes propres) ou encore la programmation **fonctionnelle** (avec des fonctions).

Le langage Python est gratuit, sous **licence libre**. Actuellement, Python en est à sa version 3.

Remarques

Le langage Python a très largement contribué au développement de l'**intelligence artificielle**, auparavant réservé à un public expert. Le module **keras** a par exemple démocratisé le *deep learning*, une technique d'apprentissage très performante, auprès des étudiants, des chercheurs et des start-ups.

I Un langage interprété

Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Python est un langage de programmation **interprété**, c'est-à-dire que les instructions qui sont envoyées au logiciel, que l'on appelle « **interpréteur** », sont « transcrites » en langage machine au fur et à mesure de la lecture du code. D'autres langages (comme le C ou le C++) sont appelés « langages **compilés** » car, avant de pouvoir les exécuter, un logiciel spécialisé se charge de transformer le code du programme en langage machine. On appelle cette étape la « **compilation** ». À chaque modification du code, il faut rappeler une étape de compilation.

Les avantages d'un langage interprété sont la simplicité (on ne passe pas par une étape de compilation avant d'exécuter son programme) et la portabilité (un langage tel que Python est censé fonctionner aussi bien sous Windows que sous Linux ou Mac OS, et on ne devrait avoir à effectuer aucun changement dans le code pour le passer d'un système à l'autre). Cela ne veut pas dire que les langages compilés ne sont pas portables, loin de là ! Mais on doit utiliser des compilateurs différents et, d'un système à l'autre, certaines instructions ne sont pas compatibles, voire se comportent différemment.

En contrepartie, un langage compilé se révélera bien plus rapide qu'un langage interprété (la traduction à la volée de votre programme ralentit l'exécution), bien que cette différence tende à se faire de moins en moins sentir au fil des améliorations.

Pour cette année, il vous faudra installer Python sur le système d'exploitation que vous utilisez pour que l'ordinateur puisse comprendre votre code.



II Installation

Sous Windows ou Mac OS X, 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.

Sous Linux, Python est pré-installé sur la plupart des distributions Linux.

III IDLE

IDLE est un environnement de développement intégré (IDE en anglais : Integrated Development Environment) pour Python.

IDLE 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 de nombreux autres environnement de développement (framework) : EduPython, PyCharm, Spyder, Jupyter Notebook. L'avantage de ces framework c'est qu'ils sont dédiés à Python et que l'ajout de bibliothèques (modules) est facile. On peut aussi simplement utiliser un éditeur de texte comme Visual Studio Code, Notepad++, Atom.

IV Anaconda Navigator



L'installation d'un environnement Python complet peut-être une vraie galère. Il faut commencer par télécharger Python et l'installer. Puis, télécharger un à un les packages dont on a besoin. Parfois, le nombre de ces librairies peut-être grand.

De plus, il faut s'assurer de la compatibilité entre les versions des différentes packages qu'on a à télécharger. Bref, ce n'est pas amusant !

Anaconda est une **distribution Python**. A son installation, Anaconda installera Python ainsi qu'une multitude de packages (voir liste de packages anaconda), évitant ainsi des problèmes d'incompatibilités entre les différents packages.

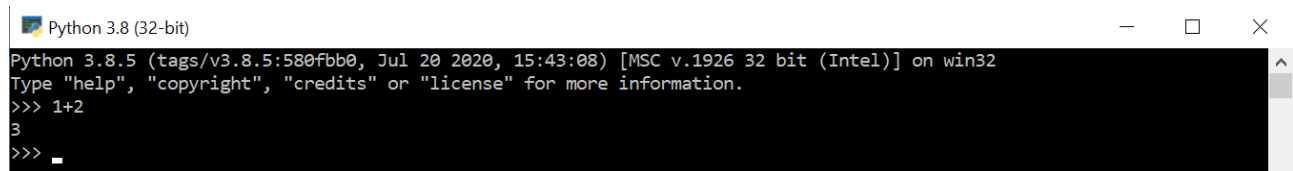
Anaconda propose un outil de gestion de packages appelé Conda, qui permettra de mettre à jour et d'installer aisément les librairies dont on aura besoin pour nos développements.

V L'interpréteur Python

Le langage de programmation Python permet d'interagir avec la machine à l'aide d'un programme appelé l'**interpréteur Python** que l'on peut utiliser de deux façons différentes :

V.a Le mode interactif

Le mode interactif consiste à dialoguer avec l'interpréteur, qui à travers l'invite de commande Python, se présentant par les trois chevrons `>>>`, attend les instructions à exécuter. Par exemple :

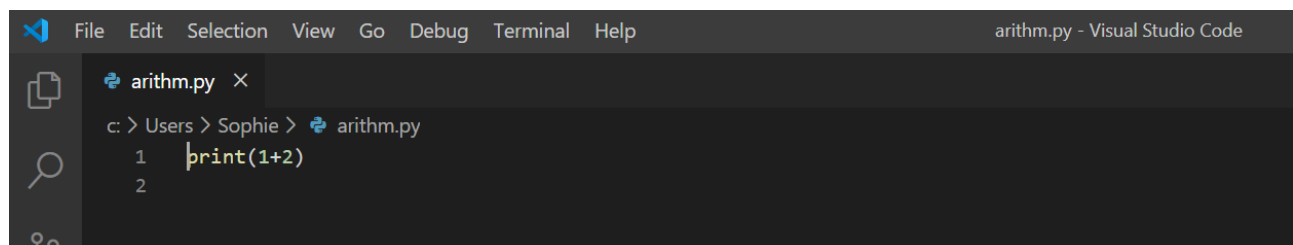


```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+2
3
>>> _
```

V.b Le mode programme

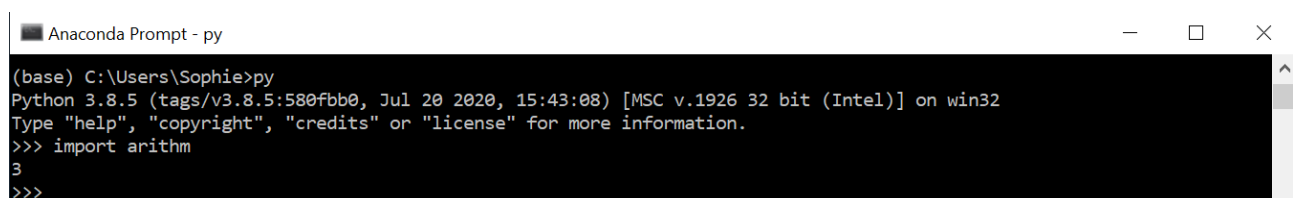
Le mode programme consiste à écrire une suite d'instructions dans un fichier et à les faire exécuter par l'interpréteur. Cette suite d'instructions s'appelle un **programme** ou encore un **code source**. On évite ainsi de ressaisir à chaque fois les instructions dans le mode interactif.

Par exemple, dans un fichier portant le nom *arithm.py*, on peut écrire l'instruction suivante :



```
File Edit Selection View Go Debug Terminal Help
arithm.py - Visual Studio Code
c: > Users > Sophie > arithm.py
1 print(1+2)
2
```

Puis, on peut exécuter ce programme par l'interpréteur Python, ce qui affiche 3 à l'écran :

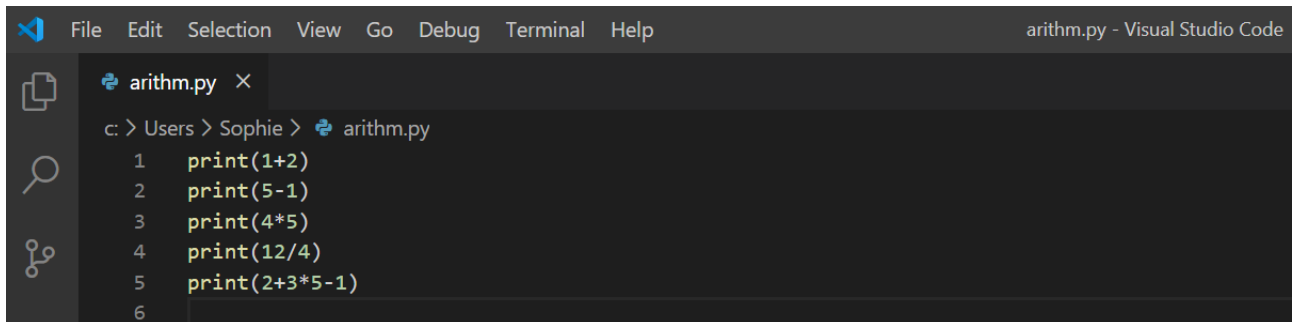


```
Anaconda Prompt - py
(base) C:\Users\Sophie>py
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import arithm
3
>>>
```

Remarque : En mode programme, les résultats des expressions calculées ne sont plus affichés à l'écran. Il faut utiliser pour cela une instruction explicite d'affichage. En Python, elle s'appelle

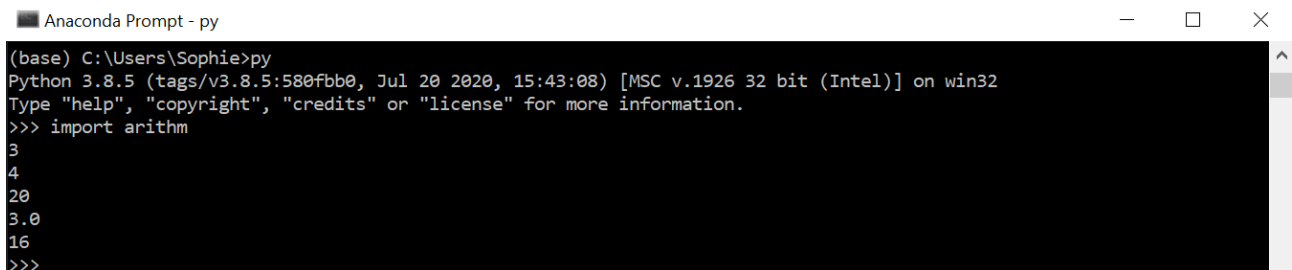
```
print.
```

Ou encore :



```
File Edit Selection View Go Debug Terminal Help arithm.py - Visual Studio Code
c: > Users > Sophie > arithm.py
1 print(1+2)
2 print(5-1)
3 print(4*5)
4 print(12/4)
5 print(2+3*5-1)
6
```

et



```
Anaconda Prompt - py
(base) C:\Users\Sophie>py
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import arithm
3
4
20
3.0
16
>>>
```

Remarque : L'instruction `print` n'est pas limitée à l'affichage des nombres. On peut également afficher un message, écrit entre guillemets. Par exemple, l'instruction `print("Salut tout le monde ...")` affiche le message *Salut tout le monde ...* à l'écran. Le texte écrit entre guillemets est appelé une **chaîne de caractères**. Les caractères accentués sont autorisés, tout comme les caractères provenant d'autres langues et plus généralement tous les caractères Unicode. On note que les guillemets englobants ne sont pas affichés.

Quelle est la différence entre l'instruction `print(1+3)` et l'instruction `print("1+3")` ? Que se passe-t-il ?

Remarques

Cette vidéo présente les deux modes d'utilisation de Python à savoir le mode interactif et le mode programme via le logiciel Visual Studio Code.