SOMMAIRE

Le langage de programmation Python a été créé dans les années 1990 par Guido van Rossum, aux Pays-Bas. Il dérive du langage C, qui faisait suite au langage ABC. La dernière version de Python est la version 3 (3.12 en 2024 à l'écriture de ce livre). La version 2 étant devenue obsolète, il est conseillé de ne pas en faire usage.

C'est un langage aujourd'hui très utilisé pour ses nombreux avantages :

- C'est un langage Open Source qui est donc gratuit et fonctionne sur de nombreux systèmes d'exploitation.
- Il est facile à apprendre puisque c'est un langage de haut niveau, qui demande peu de connaissances du fonctionnement d'un ordinateur.
- C'est un langage interprété qui, quoique plus lent par rapport aux langages compilés comme C ou Ocaml, offre plus de flexibilité pour la mise au point du code. Il est assez simple à débugger. Par ailleurs, un fichier Python pourra être lu sur tout interpréteur Python.
- De nombreux modules très performants ont été mis au point dans de nombreux domaines (traitement de données, calcul scientifique, mise au point et gestion de sites internet, conception d'infographies...). C'est donc un langage « passe-partout ». Ceci compense dans bien des cas ses performances moindres par rapport à d'autres langages plus bas niveau.

Installation

Le plus simple pour les grand débutants est d'utiliser EduPython, éditeur de texte utilisé au lycée qui, au moment de l'installation, installe un interpréteur Python, et tous les modules les plus classiques (tous ceux présents dans ce livre). Attention, sur le site d'EduPython, ce logiciel n'est disponible que pour le système d'exploitation Windows. Il existe des tutoriels sur internet pour l'installer sur d'autres systèmes.

Une autre possibilité est d'installer la distribution Anaconda (gratuite, qui installe tout d'un coup, disponible sur le site https ://www.anaconda.com/download) ou Miniconda (en version réduite qui prend moins de place mais qui ne contient pas le navigateur qui peut faciliter l'installation de modules) qui contient déjà les bibliothèques les plus utilisées et conda, un gestionnaire de paquets pour en installer d'autres. Cette distribution propose quelques éditeurs de texte comme VScodium par exemple, qui proposent une aide syntaxique (couleurs, etc.) pour le programmeur, mais qui n'ont pas l'avantage de la console. De plus, le navigateur Anaconda met à votre disposition, dans la partie learning, toutes les documentations des modules installés.

Si vous ne souhaitez pas utiliser ces méthodes, parce que vous êtes un expert du Terminal ou que vous n'avez pas besoin de nombreux modules, vous pouvez télécharger directement l'interpréteur Python correspondant à votre système d'exploitation sur le site `https://www.python.org/downloads/release`. Vous

le téléchargerez avec IDLE qui est un éditeur de texte, et le gestionnaire de fichiers pip qui permet d'installer des modules via une ligne de commande sur le Terminal. Si vous ne souhaitez pas travailler avec IDLE, vous pouvez installer un autre éditeur de texte.

Quoi qu'il arrive, à tout moment, si vous avez des questions, la documentation Python est très complète et pourra probablement y répondre : `https://www.python.org/doc/`. De la même façon, si la question est relative à une bibliothèque, chacune a son propre site internet avec la documentation correspondante, souvent très détaillée avec des exemples.

Fonctionnement

Une fois Python installé, vous ouvrez votre éditeur de texte et deux options s'offrent à vous : soit vous créez un fichier Python (d'extension ".py"), soit vous ouvrez un fichier Python (que vous aurez téléchargé sur le site de l'éditeur par exemple). Vous avez alors deux parties : votre fichier ".py" et la console qui en fonction de l'éditeur sera déjà ouverte en bas de la fenêtre ou s'ouvrira dans une autre fenêtre lors de l'exécution de votre fichier.

Vous pouvez tout taper dans votre fichier puis l'exécuter. Si vous voulez afficher quelque chose dans la console, il faudra utiliser la commande `print()` en mettant dans les parenthèses ce que vous voulez afficher. Ou alors vous tapez directement dans la console la variable ou le calcul que vous voulez afficher et appuyez sur entrée. La console affichera alors ce que vous attendez. Tout ce que vous tapez dans le fichier ne sera connu de la console que lorsque le fichier sera exécuté. Il faudra donc l'exécuter à chaque modification avant les nouveaux tests.

En revanche, tout ce qui est tapé dans la console n'est pas conservé en mémoire. Ainsi, pour tester une fonction sur un cas simple, la console peut être utile. Mais dès lors que vous avez plusieurs lignes de codes, mieux vaut prendre la bonne habitude de l'enregistrer régulièrement dans un fichier.

Enfin, il est possible d'exécuter les fichiers Python directement à partir du terminal, avec la commande `python3 nom.py`. Cet usage n'est pas utile quand on débute et est plutôt réservé à une utilisation avancée de la programmation.

Python est un langage que l'on appelle de programmation orientée objet (POO). Cela explique les deux notations de fonctions `f(a,x)`, et `a.f(x)` associée à une méthode (ou fonction d'une classe d'objets). Nous ne parlerons pas dans ce livre de ce type de programmation avancé.

Python est un langage « *batteries included* ». Cela signifie qu'au départ un grand nombre de fonctionnalités sont disponibles. Les types et fonctions disponibles sans importation d'autres modules sont dits natifs.

Modules et bibliothèques

Un module est un programme Python dans lequel se trouve des fonctions, des constantes, qui peuvent nous être très utiles. Par exemple, le module `math` contient la valeur (approchée) du nombre π, du nombre e, mais aussi de nombreuses fonctions comme la fonction cosinus ou la fonction exponentielle. Le module `matplotlib.pyplot` contient de nombreuses fonctions permettant d'obtenir de jolis graphiques dans des contextes très variés. Plus généralement, les modules permettent de partager des programmes déjà conçus, testés et optimisés pour certaines tâches.
Remarquez que le nom du dernier module est double. En fait `matplotlib` est un très gros module, que l'on appellera **bibliothèque**, dans lequel se trouve d'autres modules, comme ici le module `pyplot`.
Pour utiliser un module, il y a plusieurs possibilités :

```
1 import nom_du_module as alias
```

On importe l'ensemble du module `nom_du_module`. On peut utiliser, mais ce n'est pas obligatoire, un alias qui permettra de raccourcir les commandes ensuite. Une fonction `f` de ce module pourra alors être utilisée en écrivant `nom_du_module.f` ou plus simplement `alias.f`.

```
1 from nom_du_module import *
```

Cette commande permet à nouveau de charger l'ensemble du module. Par ailleurs, pour utiliser une fonction `f` de ce module, on peut directement écrire `f`. Pour autant, il ne s'agit pas d'une pratique recommandable. Il est fréquent que deux modules différents comportent des fonctions nommées de la même façon (mais qui ne font pas forcément la même chose). Dans ce cas, il y aurait un conflit entre les deux, et Python utilisera l'une ou l'autre de façon arbitraire.

```
1 from nom_du_module import f
```

Ici, cela permet de ne charger qu'une seule fonction d'un module, et non son entièreté, ce qui peut être avantageux dans le cas de modules volumineux.

Si le module n'a pas été installé au préalable, Python renverra une erreur :

```
ModuleNotFoundError: No module named 'nom_du_module'
```