Chapitre 1

Introduction

Ce chapitre s'intéresse tout d'abord à l'installation du langage *Python* et à la réalisation d'un premier programme avec des instructions dont le sens est intuitif. Les derniers paragraphes présentent de nombreuses extensions disponibles sur Internet. Elles rendent le langage *Python* très attractif dans des domaines variés. Ces extensions témoignent que ce langage emporte l'adhésion de nombreux informaticiens qui en retour assurent sa pérennité. Il permet de relier facilement différents éléments, différentes applications. C'est une des raisons de son succès.

1.1 Ordinateur et langages

Il est rare aujourd'hui de ne pas avoir déjà entendu ou lu les termes informatiques définis ci-dessous. C'est un rapide rappel.

1.1.1 L'ordinateur

On peut considérer simplement qu'un ordinateur est composé de trois ensembles : le microprocesseur, la mémoire, les périphériques. Cette description n'a pas varié en cinquante ans depuis qu'un scientifique du nom de von Neumann l'a imaginée.

Le microprocesseur est le cœur de l'ordinateur, il suit les instructions qu'on lui donne et ne peut travailler qu'avec un très petit nombre d'informations. Sa vitesse se mesure en GigaHertz (GHz) qui correspondent au nombre d'opérations qu'il est capable d'effectuer en une seconde et en nombre de cœurs qui détermine le nombre d'opérations en parallèle qu'il est capable d'exécuter. On lui adjoint une mémoire avec laquelle il échange sans arrêt des données. Sa capacité se mesure en octets (kilo-octets, mégaoctets, gigaoctets ou leurs abréviations Ko, Mo, Go [1]). Ces échanges entre processeur et mémoire sont rapides.

Les périphériques regroupent tout le reste (écran, clavier, souris, disque dur, imprimante...). Ils sont principalement de deux types : les périphériques de stockages (disque dur, DVD) et ceux qui nous permettent de dialoguer avec l'ordinateur, que ce soit pour afficher, sonoriser (écran, enceintes) ou pour recevoir (souris, clavier, webcam, micro...).

1. Un kilo-octets équivaut à $1024 = 2^{10}$ octets, un Mo à 1024 Ko et un Go à 1024 Mo.

1.1.2 Termes informatiques

Certains termes reviendront fréquemment dans le livre. Ils sont souvent utilisés comme si tout le monde les connaissait comme la notion d'algorithme qui paraît plutôt abstraite si on se réfère à sa définition dans le dictionnaire. Dans la pratique, on confond souvent algorithme avec programme informatique.

> **Définition 1.1 : algorithme**
> Un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver avec certitude, en un nombre fini d'étapes, à un certain résultat et cela, indépendamment des données.

Leur écriture est indépendante du langage choisi, qu'il soit écrit en *Basic*, en *Pascal*, en *C*, en *Perl*, en *PHP*, en *Python*, en français, un algorithme reste le même. Pour les algorithmes simples (un tri par exemple), le passage d'un langage à l'autre consiste souvent à traduire mot-à-mot, ce qui favorise l'apprentissage d'un nouveau langage informatique lorsqu'on en connaît déjà un. Les différences entre ces langages résident dans leur spécialisation, le *Visual Basic* permet de piloter les applications *Microsoft Office*, le *PHP*, le *JavaScript* sont dédiés à la programmation Internet. Le langage *Python* n'est pas spécialisé. Il est souvent moins efficace que chaque langage appliqué à son domaine de prédilection mais il peut éviter l'apprentissage d'une syntaxe différente.

> **Définition 1.2 : programme**
> Un programme informatique est une suite d'instructions ou séquence d'instructions. C'est la réalisation informatique d'un ou plusieurs algorithmes. Il dépend du langage.

> **Définition 1.3 : compilateur et compilation**
> Le compilateur est un programme qui traduit un code écrit dans un langage de programmation en langage dit "machine", compréhensible par l'ordinateur. La compilation est le fait de traduire un programme afin que l'ordinateur le comprenne.

> **Définition 1.4 : langage interprété**
> Un langage interprété est converti en instructions propres à la machine au fur et à mesure de son exécution.

Le langage *Python* n'est pas un langage compilé car un programme *Python* n'est pas traduit en langage machine, il est un langage interprété. Entre son écriture et son exécution, il n'y a pas d'étape intermédiaire telle que la compilation et on peut ainsi tester un programme plus rapidement même si son exécution est alors plus lente.

> **Définition 1.5 : mot-clé**
> Un mot-clé est une composante du langage et fait partie de sa grammaire qui comprend également les opérateurs numériques.

La table 3.1 (page 58) regroupe les mots-clés du langage *Python*. Elle contient peu de

mots-clés : son concepteur s'est attaché à créer un langage objet avec la grammaire la plus simple possible.

> **Définition 1.6 : instruction**
> Ce terme est assez vague et dépend en général du langage. On peut considérer qu'une instruction est une expression syntaxiquement correcte pour un langage donné.

Les instructions sont très courtes et tiennent sur une ligne mais si on n'en prend qu'une partie, elle perd son sens. Ce serait comme considérer une phrase avec un verbe transitif mais sans son complément d'objet direct.

1.2 Présentation du langage *Python*

1.2.1 Histoire résumée

Python est un langage objet interprété de haut niveau, il a été créé au début des années quatre-vingt-dix par Guido van Rossum. Entre 1995 et 2001, Rossum a changé plusieurs fois de travail tout en continuant l'élaboration du langage *Python*. En 2001, la PSF (Python Software Foundation) est créée. Il s'agit d'une organisation à but non lucratif détenant les droits de propriété intellectuelle de *Python*. Il est depuis distribué sous forme de logiciel libre. *Python* est couvert par sa propre licence [2]. Toutes les versions depuis la 2.0.1 sont compatibles avec la licence GPL [3]. Selon Wikipedia [4], Rossum travaillerait maintenant pour l'entreprise *Google* qui propose une plate-forme de développement de sites web en *Python* [5] et qui héberge de nombreux projets en langage *Python* via son projet *Google Code* [6].

1.2.2 Le langage en quelques points

On distingue plusieurs classes parmi les langages informatiques selon la syntaxe qu'ils proposent ou les possibilités qu'ils offrent. *Python* est un langage : interprété, orienté objet, de haut niveau, modulaire, à syntaxe positionnelle, au typage dynamique.

Le langage *Python* est dit *interprété* car il est directement exécuté sans passer par une phase de compilation qui traduit le programme en langage machine, comme c'est le cas pour le langage *C*. En quelque sorte, il fonctionne autant comme une calculatrice que comme un langage de programmation. Afin d'accélérer l'exécution d'un programme *Python*, il est traduit dans un langage intermédiaire [7] qui est ensuite interprété par une machine virtuelle *Python*. Ce mécanisme est semblable à celui propre au langage *Java*.

Le langage est orienté objet car il intègre le concept de classe ou d'objet. Un objet

2. Voir le site http://www.python.org/psf/license/, cette licence autorise les usages commerciaux.
3. Ou GNU Public Licence (voir le site http://www.gnu.org/copyleft/gpl.html).
4. http://fr.wikipedia.org/wiki/Python_(langage)
5. http://code.google.com/appengine/
6. http://code.google.com/
7. Celui-ci est appelé *bytecode*. Ce mot ne signifie pas grand chose excepté que ce *bytecode* est seulement compréhensible par la machine virtuelle *Python*.

regroupe un ensemble de données et un ensemble de fonctionnalités attachées à ces données. Ce concept relie la description des données et les algorithmes qui leur sont appliqués comme un tout indissociable. Un objet est en quelque sorte une entité autonome avec laquelle on peut communiquer via une interface.

On considère que le langage *Python* est de haut niveau car il propose des fonctionnalités avancées et automatiques comme le *garbage collecting*. Cette tâche correspond à la destruction automatique des objets lorsqu'ils ne sont plus utilisés. Cette fonctionnalité est proposée par la plupart des langages interprétés ou encore *Java* mais pas *C++*. Il propose également des structures de données complexes éloignées des types numériques standards telles que des dictionnaires.

Le langage *Python* est modulaire. La définition du langage est très succincte et autour de ce noyau concis, de nombreuses librairies ou modules ont été développées. *Python* est assez intuitif. Etre à l'aise avec ce langage revient à connaître tout autant sa syntaxe que les nombreux modules disponibles. Comme il est assez simple de construire un pont entre *Python* et *C++*, de nombreux projets open source antérieurs à *Python* sont maintenant accessibles dans ce langage et sous toutes les plateformes.

Le langage *Python* est à syntaxe positionnelle en ce sens que l'indentation fait partie du langage. Le point virgule permet de séparer les instructions en langage *C*, l'accolade permet de commencer un bloc d'instructions. En *Python*, seule l'indentation permet de marquer le début et la fin d'un tel bloc, ce procédé consiste à décaler les lignes vers la droite pour signifier qu'elles appartiennent au même bloc d'instructions et qu'elles doivent être exécutées ensemble. Cette contrainte rend les programmes *Python* souvent plus faciles à lire.

Le langage *Python* est à typage dynamique. Le type d'une variable, d'une donnée est définie lors de l'exécution du programme. Chaque information est désignée par un identificateur qui est manipulé tout au long du programme. Le fait que telle ou telle opération soit valide est vérifiée au moment où elle doit être réalisée et non au moment où le programme est traduit en langage machine, c'est-à-dire compilé.

1.2.3 Avantages et inconvénients du langage *Python*

Alors qu'il y a quelques années, le langage *C* puis *C++* s'imposaient souvent comme langage de programmation, il existe dorénavant une profusion de langages (*Java*, *JavaScript*, *PHP*, *Visual Basic*, *C#*, *Perl*, *PHP*, ...). Il est souvent possible de transposer les mêmes algorithmes d'un langage à un autre. Le choix approprié est bien souvent celui qui offre la plus grande simplicité lors de la mise en œuvre d'un programme même si cette simplicité s'acquiert au détriment de la vitesse d'exécution. Le langage *PHP* est par exemple très utilisé pour la conception de sites Internet car il propose beaucoup de fonctions très utiles dans ce contexte. Dans ce domaine, le langage *Python* est une solution alternative intéressante.

Comme la plupart des langages, le langage *Python* est tout d'abord portable puisqu'un même programme peut être exécuté sur un grand nombre de systèmes d'exploitation comme *Linux*, *Microsoft Windows*, *Mac OS X*... *Python* possède également l'avantage d'être entièrement gratuit tout en proposant la possibilité de pouvoir réaliser des applications commerciales à l'aide de ce langage.

Si le langage *C* reste le langage de prédilection pour l'implémentation d'algorithmes complexes et gourmands en temps de calcul ou en capacités de stockage, un langage tel que *Python* suffit dans la plupart des cas. De plus, lorsque ce dernier ne convient pas, il offre toujours la possibilité, pour une grande exigence de rapidité, d'intégrer un code écrit dans un autre langage tel que le *C/C++*[8] ou *Java*, et ce, d'une manière assez simple. La question du choix du langage pour l'ensemble d'un projet est souvent superflue, il est courant aujourd'hui d'utiliser plusieurs langages et de les assembler. Le langage *C* reste incontournable pour concevoir des applications rapides, en revanche, il est de plus en plus fréquent d'"habiller" un programme avec une interface graphique programmée dans un langage tel que *Python*.

En résumé, l'utilisation du langage *Python* n'est pas restreinte à un domaine : elle comprend le calcul scientifique, les interfaces graphiques, la programmation Internet. Sa gratuité, la richesse des extensions disponibles sur Internet le rendent séduisant dans des environnements universitaire et professionnel. Le langage est vivant et l'intérêt qu'on lui porte ne décroît pas. C'est un critère important lorsqu'on choisit un outil parmi la myriade de projets open source. De plus, il existe différentes solutions pour améliorer son principal point faible qui est la vitesse d'exécution.

1.3 Installation du langage *Python*

1.3.1 Installation du langage *Python*

Python a l'avantage d'être disponible sur de nombreuses plates-formes comme *Microsoft Windows*, *Linux* ou *Mac OS X*. L'installation sous *Windows* est simple. Le langage *Python* est déjà intégré aux systèmes d'exploitation *Linux* et *Mac OS X*.

Sous *Microsoft Windows*, il suffit d'exécuter le fichier *python-2.6.5.msi*[9] ou tout autre version plus récente. En règle générale, il est conseillé de télécharger la dernière version stable du langage, celle avec laquelle le plus grande nombre d'extensions seront compatibles, en particulier toutes celles dont vous avez besoin. Les options d'installation choisies sont celles par défaut, le répertoire d'installation est par défaut C:/Python26. A la fin de cette installation apparaît un menu supplémentaire dans le menu *Démarrer* (ou *Start*) de *Microsoft Windows* comme le montre la figure 1.1. Ce menu contient les intitulés suivant :

IDLE (Python GUI)	éditeur de texte, pour programmer
Module Docs	pour rechercher des informations dans la documentation
Python (command line)	ligne de commande *Python*
Python Manuals	documentation à propos du langage *Python*
Uninstall Python	pour désinstaller *Python*

La documentation (de langue anglaise[10]) décrit en détail le langage *Python*, elle inclut également un tutoriel[11] qui permet de le découvrir. La ligne de commande (voir

8. Ce point est d'ailleurs abordé au paragraphe 6.5, page 158.
9. Cette version est disponible à l'adresse http://www.python.org/download/. C'est la version utilisée pour développer les exemples présents dans ce livre.
10. http://www.python.org/doc/
11. La requête *python français* de nombreux sites de documentation sur *Python* en français.

Figure 1.1 : *Menu ajouté à Microsoft Windows.*

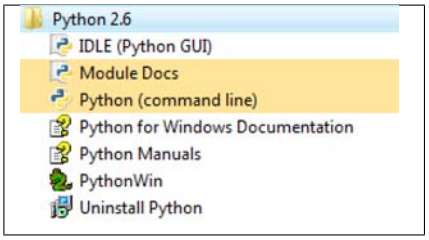

figure 1.2) permet d'exécuter des instructions en langage *Python*. Elle est pratique pour effectuer des calculs mais il est nécessaire d'utiliser un éditeur de texte pour écrire un programme, de le sauvegarder, et de ne l'exécuter qu'une fois terminé au lieu que chaque ligne de celui-ci ne soit interprétée immédiatement après qu'elle a été écrite comme c'est le cas pour une ligne de commande.

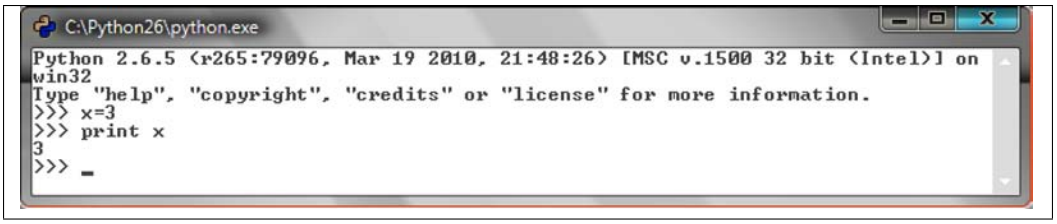

Figure 1.2 : *Ligne de commande, la première ligne affecte la valeur 3 à la variable* `x`*, la seconde ligne l'affiche.*

1.3.2 Particularité de *Mac OS X* et *Linux*

Le langage *Python* est déjà présent sur les ordinateurs d'*Apple* si ceux-ci sont équipés du système d'exploitation *Mac OS X*. Il est également présent dans les distributions *Linux*. L'installation de *Python* est parfois rendue nécessaire si on désire utiliser la dernière version du langage. L'installation des extensions du langage est similaire à celle d'autres extensions *Linux*.

Il ne faut pas oublier de vérifier la version du langage *Python* installée sur l'ordinateur *Linux* et *Mac OS X*. Il suffit pour cela de taper la ligne `help()` pour en prendre connaissance. Cela signifie que toutes les extensions qui doivent être installées doivent l'être pour cette version qu'il est néanmoins possible de mettre à jour [12].

1.3.3 Utilisation de l'éditeur de texte

La figure 1.1 montre le menu installé par *Python* dans le menu "Démarrer" de *Microsoft Windows*. En choisissant l'intitulé "IDLE (Python GUI)", on active la fenêtre de commande de *Python*(voir figure 1.3). Les instructions sont interprétées

12. voir la page http://www.python.org/download/mac/ ou http://www.python.org/download/linux/

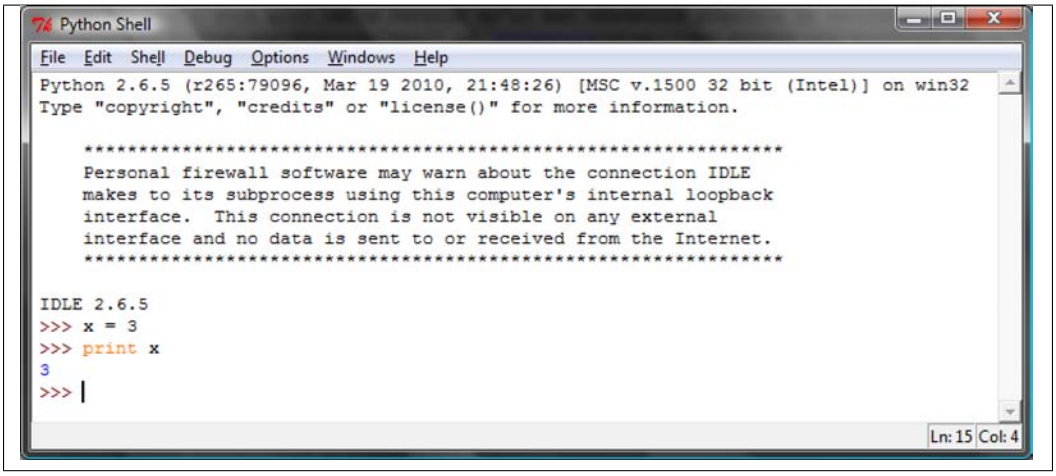

Figure 1.3 : *Fenêtre de commande fournie avec le langage Python.*

au fur et à mesure qu'elles sont tapées au clavier. Après chaque ligne, cette fenêtre de commande conserve la mémoire de tout ce qui a été exécuté. Par exemple :

```
>>> x = 3
>>> y = 6
>>> z = x * y
>>> print z
18
>>>
```

Après l'exécution de ces quelques lignes, les variables [13] `x`, `y`, `z` existent toujours. La ligne de commande ressemble à une calculatrice améliorée. Pour effacer toutes les variables créées, il suffit de redémarrer l'interpréteur par l'intermédiaire du menu `Shell->Restart Shell`. Les trois variables précédentes auront disparu.

Il n'est pas possible de conserver le texte qui a été saisi au clavier, il est seulement possible de rappeler une instruction déjà exécutée par l'intermédiaire de la combinaison de touches `ALT + p`, pressée une ou plusieurs fois. La combinaison `ALT + n` permet de revenir à l'instruction suivante. Pour écrire un programme et ainsi conserver toutes les instructions, il faut d'actionner le menu `File->New Window` qui ouvre une seconde fenêtre qui fonctionne comme un éditeur de texte (voir figure 1.4).

Après que le programme a été saisi, le menu `Run→Run Module` exécute le programme. Il demande au préalable s'il faut enregistrer le programme et réinitialise l'interpréteur *Python* pour effacer les traces des exécutions précédentes. Le résultat apparaît dans la première fenêtre (celle de la figure 1.3). La pression des touches "Ctrl + C" permet d'arrêter le programme avant qu'il n'arrive à sa fin.

Remarque 1.7 : fenêtre intempestive
Si on veut se débarrasser de cette fenêtre intempestive qui demande confirma-

13. Les variables sont définies au chapitre 2. En résumé, on peut considérer une variable comme une lettre, un mot qui désigne une information. Dans cet exemple, les trois variables désignent des informations numériques qu'on peut manipuler comme si elles étaient effectivement des nombres. Les variables supprimées, les informations qu'elles désignaient ne plus accessibles et sont perdues également.

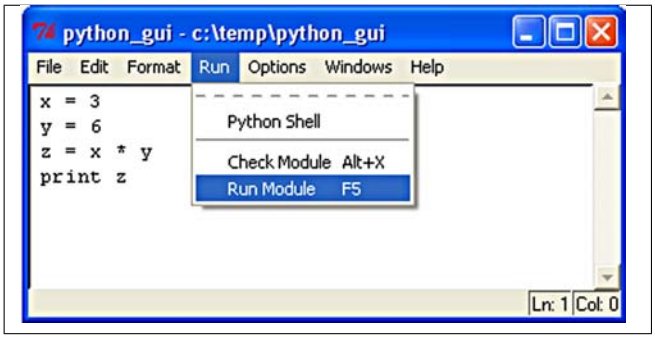

Figure 1.4 : *Fenêtre de programme, le menu* `Run→Run Module` *permet de lancer l'exécution du programme. L'interpréteur Python est réinitialisé au début de l'exécution et ne conserve aucune trace des travaux précédents.*

tion pour enregistrer les dernières modifications, il suffit d'aller dans le menu `Options->Configure IDLE` et de choisir `No prompt` sur la quatrième ligne dans la rubrique `General` (voir figure 1.5). Ce désagrément apparaît sur la plupart des éditeurs de texte et se résout de la même manière.

Cette description succincte permet néanmoins de réaliser puis d'exécuter des programmes. Les autres fonctionnalités sont celles d'un éditeur de texte classique, notamment la touche `F1` qui débouche sur l'aide associée au langage *Python*. Il est possible d'ouvrir autant de fenêtres qu'il y a de fichiers à modifier simultanément.

Néanmoins, il est préférable d'utiliser d'autres éditeurs plus riches comme celui proposé au paragraphe 1.4.1. Ils proposent des fonctions d'édition plus évoluées que l'éditeur installé avec *Python*.

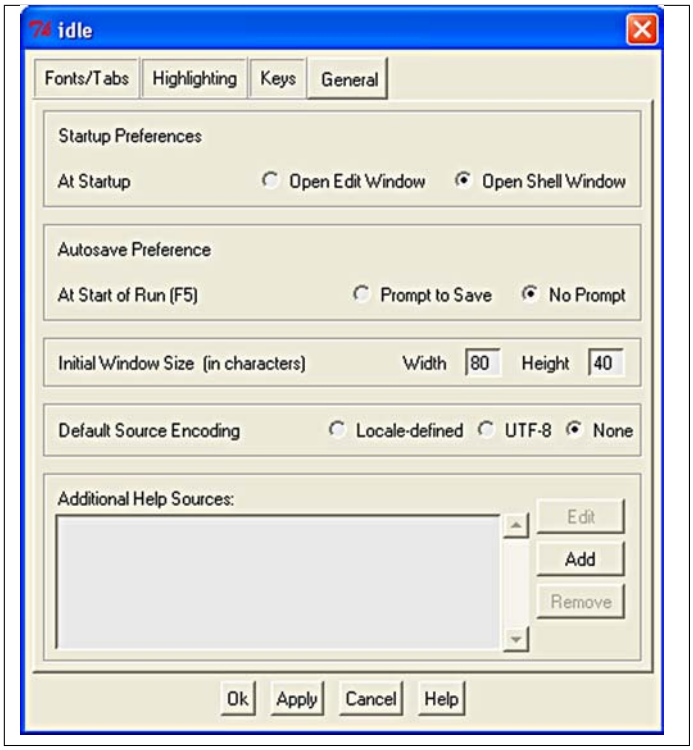

Figure 1.5 : *Cliquer sur* `General` *puis sur* `Noprompt` *pour se débarrasser de la fenêtre intempestive qui demande à l'utilisateur de confirmer la sauvegarde des dernières modifications. D'autres éditeurs de texte se comportent de la même manière et disposent d'une option similaire.*