# 1 Qu'est-ce qu'un langage de programmation ?

Programmer consiste à demander à l'ordinateur d'effectuer des actions, comme par exemple *Faire le calcul 3 + 5*. Comme on l'a déjà vu, un ordinateur ne peut stocker et travailler qu'en langage binaire (informations codées par des suites de 0 et de 1). Il est inconcevable d'écrire directement un programme en langage binaire, car ce serait extrêmement fastidieux !

Pour résoudre ce problème, on écrit les programmes dans des langages évolués, *i.e.*, compréhensibles par le programmeur comme, par exemple, le langage `C` ou le langage `Python` qui sont des *langages évolués de haut niveau*. En informatique, un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un alphabet, d'un vocabulaire et de règles de grammaire.

Comme l'ordinateur ne travaille qu'en *langage binaire ou machine*, il ne comprend pas et ne peut exécuter directement un langage évolué. Il faut donc traduire ce programme écrit en langage évolué en langage binaire ou en langage machine : c'est le rôle de la *compilation* ou de l'*interprétation*. Les programmes réalisant cette traduction automatique s'appellent des *compilateurs* ou des *interpréteurs*. Le programme produit par un compilateur s'appelle un *exécutable* : il pourra ensuite être directement exécuté par un ordinateur. Au contraire, l'interpréteur réalise cette traduction « à la volée » (il ne produit pas d'exécutable).

Pour un programme écrit, par exemple, en langage `C`, on utilise un *compilateur* `C`. On dit alors que le langage `C` est un *langage compilé*. En revanche, pour un programme écrit en `Python`, on utilise un *interpréteur* `Python` qui traduit le le code source ligne par ligne en langage machine au moment même de l'exécution du programme. On dit dans ce cas que la langage `Python` est un *langage interprété*. En d'autres termes, un *compilateur* traduit l'ensemble du programme avant qu'il puisse être exécuté et fournit un programme binaire à la suite de ce travail. En revanche, un *interpréteur* traduit une ligne et le code machine correspondant est exécuté ; puis il traduit ensuite la ligne suivante qui peut alors être exécutée, etc. En particulier, le code binaire associé n'est jamais disponible car jamais enregistré.

# 2 Un peu d'histoire

Entre 1842 et 1843, une jeune comtesse du nom d'Ada Lovelace traduisait le mémoire d'un mathématicien italien du nom de Luigi Menabrea sur la machine analytique proposée par Charles Babbage. À cette traduction, la jeune comtesse avait ajouté ses propres notes dont l'une décrivait de façon détaillée une séquence progressive d'opérations pour résoudre certains problèmes mathématiques. Le premier programme était né.

Dans les années 1950, les trois premiers langages de programmation modernes ont été conçus :

- `FORTRAN`, un traducteur de formules (FORmula TRANslator),
- `LISP`, spécialisé dans le traitement des listes (LISt Processor),
- `COBOL`, spécialisé dans la programmation d'application de gestion (COmmon Business Oriented Langage).

En 1972, le `C` fait son apparition. Créé par le regretté Denis Ritchie, ce langage a servi à coder le système Unix.

Le langage `Python` est un langage *interprété* qui permet (sans l'imposer) une approche modulaire et orientée objet de la programmation. Il est développé depuis 1991 par Guido van Rossum (université d'Amsterdam) et de nombreux contributeurs bénévoles. Il a été nommé ainsi en référence à la série télévisée *Monthy Python's Flying Circus*.

Les principales caractéristiques de `Python` sont les suivantes :

- Il est *open source*. Libre et gratuit, il est supporté, développé et utilisé par une large communauté : 300 000 utilisateurs et plus de 500 000 téléchargements par an. Il peut également être utiliser sans restriction dans des projets commerciaux.
- Il est *portable* non seulement sur les différentes variantes `Unix`, mais aussi sur les `OS` propriétaires : `Mac OS`, `Windows`, etc...
- Il possède *une syntaxe très simple* et, combinée à des types de données évolués (listes, dictionnaires...), conduit à des programmes à la fois très compacts et très lisibles [1].
- Il gère ses ressources (mémoire, descripteurs de fichiers, etc...) sans intervention du programmeur par un mécanisme de *comptage de référence*.
- Il n'y a pas de *pointeurs explicites* comme dans la plupart des langages de programmation.
- Il est (optionnellement) *multi-threadé*, c'est-à-dire qu'il est capable d'exécuter efficacement plusieurs tâches simultanément.
- Il est *orienté objet* : Il supporte *l'héritage multiple* et *la surcharge des opérateurs*. Dans son modèle objet, et en reprenant la terminologie de `C++`, toutes les méthodes sont virtuelles.
- Il intègre, comme `Java` ou les versions récentes de `C++`, un système *d'exceptions*, qui permettent de simplifier considérablement la gestion des erreurs.
- Il est *dynamique* (l'interpréteur peut évaluer des chaînes de caractères représentant des expressions ou des instructions), *orthogonal* (un petit nombre de concepts suffit à engendrer des constructions très riches), *réflectif* (il supporte la métaprogrammation, par exemple la capacité pour un objet de se rajouter ou de s'enlever des attributs ou des méthodes, ou même de changer de classe en cours d'exécution) et *introspectif* (un grand nombre d'outils de développement, comme le debugger ou le profiler, sont implantés en `Python` lui-même).
- Il est *dynamiquement typé* : tout objet manipulable par le programmeur possède un type bien défini à l'exécution, qui n'a pas besoin d'être déclaré à l'avance.
- Il est *extensible* : on peut facilement l'interfacer avec des bibliothèques `C` existantes et on peut aussi s'en servir comme d'un langage d'extension pour des systèmes logiciels complexes.
- Sa *bibliothèque standard* et les *paquetages contribués*, donnent accès à une grande variété de services : calcul scientifique, chaînes de caractères et expressions régulières, services `UNIX` standards (fichiers, pipes, signaux, sockets, threads, etc...), protocoles Internet (Web, News, `FTP`, `CGI`, `HTML`, etc...), persistance et bases de données, interfaces graphiques, etc...

Le langage `Python` continue à évoluer, soutenu par une grande communauté d'utilisateurs, dont la plupart sont des supporters du logiciel libre. Parallèlement à l'interpréteur principal, écrit en `C` et maintenu par le créateur du langage, un second interpréteur, écrit en `Java`, est en cours de développement.

Actuellement, les programmes peuvent être écrits en `Python2` ou `Python3`. Les versions les plus récentes sont `Python3.8` (stable) et `Python3.9` (en développement). On renvoie au site officiel de `Python` [2] pour plus de détails.

**Toutes les instructions décrites dans ce document sont en `Python3`.**

1. A fonctionnalités égales, un programme `Python` (abondamment commenté et présenté selon les canons standards) est souvent de 3 à 5 fois plus court qu'un programme `C` ou `C++` (ou même `Java`) équivalent, ce qui représente en général un temps de développement de 5 à 10 fois plus court et une facilité de maintenance largement accrue.
2. https://www.python.org/doc/

# 3 Prise en main de `Python`

Pour apprendre la programmation Python, rien de telle que la pratique en tapant des lignes de codes et en observant ce qu'elles font. Ainsi, brique après brique, on arrive à construire des programmes de plus en plus complexes.

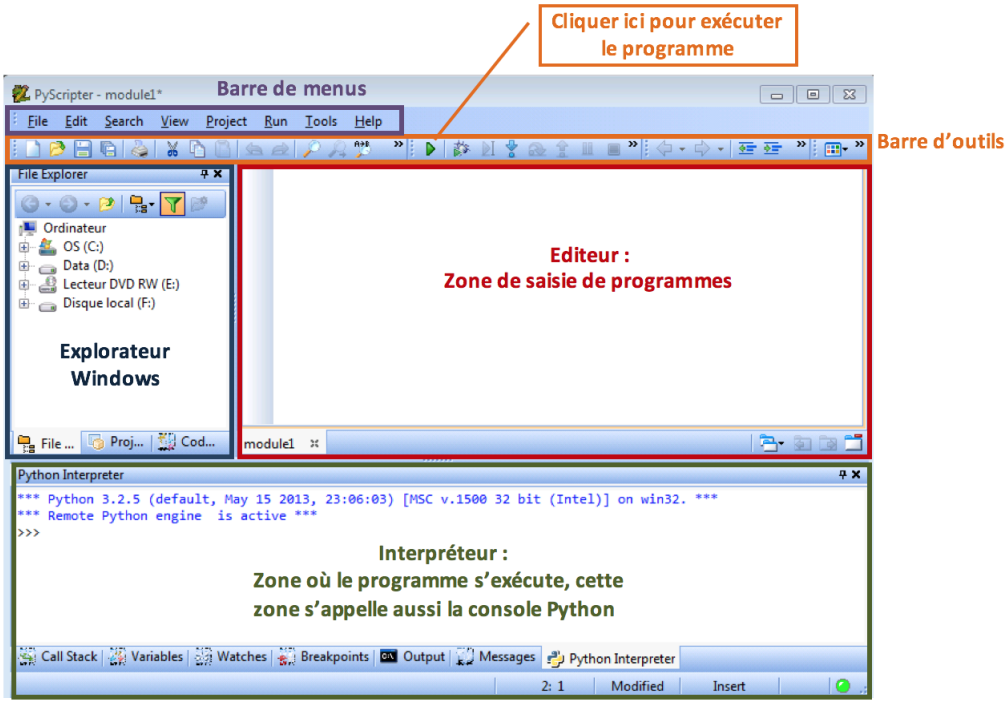
Bien sûr, pour taper ces lignes de codes, il est nécessairement d'avoir à disposition un IDE (Environnement de Développement Intégré). Il en existe de nombreux : `IDLE`, `Spider` (pour la distribution `Anaconda`), `EduPython`, `Basthon` (en ligne), `Edupyter`, etc.

Pour l'enseignement de spécialité NSI, nous avons choisi de programmer avec le logiciel `EduPython` et avec l'application `Capytale`.

## 3.1 `EduPython`

Le logiciel `EduPython` fonctionne uniquement sous `Windows` et peut être téléchargé ici.


Après avoir lancé le logiciel, il se présente de la façon suivante :




La fenêtre graphique du logiciel `EduPython` est divisée en plusieurs zones :

- une barre de menu en haut,
- une barre d'outils placée juste en dessous de la barre de menu,
- une zone pour l'explorateur de fichiers de `Windows`,
- une zone d'édition de programmes,
- une console d'exécution.

Nous utiliserons ce logiciel un peu plus tard dans l'année pour réaliser des programmes plus conséquents. Pour débuter dans l'apprentissage des différentes notions de programmation, nous utiliserons plutôt l'application `Capytale`

## 3.2 `Capytale`

L'application `Capytale` est liée à l'IDE `Basthon` et est disponible directement dans l'ENT. Nous manipulerons des Notebooks qui permettent de taper à la fois du texte classique et des programmes `Python` qui pourront être exécutés en direct sans aucune installation préalable. Pour une prise en main de cette application, commençons par réaliser le Notebook *NSI Première Partie 1 Chapitre 1 Prise en main Notebook Jupyter 1/2* de prise en main.

Pour aller plus loin, vous pouvez également réaliser le Notebook *NSI Première Partie 1 Chapitre 1 Prise en main Notebook Jupyter 2/2*.