1re/Tle

# SPÉCIALITÉ NSI

programmes téléchargeables

**Toutes les notions d'informatique pour commencer l'enseignement supérieur dans les meilleures conditions**

- un cours synthétique, illustré d'exemples
- des exercices corrigés
- des pistes de réflexions pour le Grand Oral
- des mémentos, des astuces pour coder plus vite et plus efficacement

ellipses

# Chapitre 1

# Bases de Python

*Cette partie devrait déjà être maîtrisée en entrant en terminale.*
La programmation est une branche de l'informatique qui sert à **créer des programmes**. Tout ce que vous avez installé sur votre ordinateur (les logiciels), sur votre smartphone android ou votre iPhone, sur votre tablette, sont des programmes : votre navigateur Internet (Internet Explorer, Firefox, Opera, etc.), votre système d'exploitation (Windows, Linux, etc.), votre lecteur MP3, votre logiciel de discussion instantanée, vos jeux vidéo, etc.

Un programme est une **séquence d'instructions dans un ordre donné**, que l'ordinateur exécute pour réaliser certaines actions. Ces instructions sont généralement assez basiques, par exemple : afficher un message sur l'écran, saisir des caractères au clavier, tracer un trait entre 2 points, additionner ou multiplier plusieurs nombres, répéter plusieurs fois la même opération (on parle de boucle)...

**Créer un programme, c'est tout simplement créer une suite d'instructions de base** qui permettra de faire ce que l'on veut. Tous les programmes sont créés ainsi : votre *Player* (VLC, Media Player Classic...) donne des instructions à l'ordinateur pour écouter de la musique ou lire des vidéos, le *chat* (WhatsApp, Telegram...) donne des instructions pour discuter avec d'autres gens sur le réseau, le système d'exploitation (Linux, Windows, MacOS) donne des instructions pour dire à l'ordinateur comment utiliser le matériel et comment fonctionner, etc.
Ces instructions sont stockées dans notre ordinateur sous la forme de bits, de petites données qui valent soit 0, soit 1. Ainsi, nos instructions ne sont rien d'autre que des suites de 0 et de 1, stockées dans notre ordinateur, et que le microprocesseur de la carte mère de l'ordinateur va interpréter comme étant des ordres à effectuer. Ces suites de 0 et 1 sont difficilement compréhensibles pour nous humains.
*Parler à l'ordinateur avec des 0 et des 1 est très dur et très long. Ce langage porte un nom c'est le langage machine.*

**Afin d'écrire des programmes compréhensibles par un être humain, on a inventé les langages de programmation.** À l'heure actuelle, depuis les débuts de l'informatique moderne dans les années 50, des milliers de langages de programmation ont été inventées.

Pour faire un programme il faut donc écrire le code source des instructions dans un langage de programmation, puis ce code subit un traitement informatique pour être traduit en langage machine compréhensible par le microprocesseur. Si le traitement est global et conduit à la création d'un exécutable, il porte le nom de compilation. Si le traitement s'effectue ligne par ligne, avec une traduction et une exécution avant de traiter la ligne suivante, on parlera de langage interprété. Le logiciel traduisant le code source d'un coup en exécutable est un compilateur. Le logiciel traduisant ligne et exécutant les lignes les unes à la suite des autres est appelé un interpréteur.

Prenons l'exemple d'un code source écrit en Python en local ou à l'aide d'un interpréteur en ligne (`replit.com` par exemple).

**Exemple**

On écrit les instructions avec un langage de programmation, et l'on appelle le fichier « code source » :

```
1 print("Bonjour Mademoiselle")
2 print("Bonjour Madame\n")
3 print("Bonjour Monsieur")
```

↓

L'interpréteur convertit chaque ligne en langage machine et l'exécute.

↓

Le processeur comprend ce qu'on lui demande dans son propre langage :

0 1 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1
1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0 0 1 0 0 0 1 0 1
puis
1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1
0 0 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0
enfin
0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1 0 1 0
0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 0 0

*Remarque* : il est préférable d'utiliser un IDE, *Integrated Development Environment* ou Environnement de Développement Intégré, c'est-à-dire un environnement permettant de taper du code et de le lancer. En effet, celui-ci contient tout ce qu'il faut pour écrire, débugger (corriger les erreurs), compiler ou interpréter le programme, et le faire fonctionner.

**Le langage Python** est apparu en 1991 pour améliorer la lisibilité du code et entraîner les étudiants à bien indenter leurs productions. Il est le successeur des langages A, B et C, ce dernier ayant été inventé une vingtaine d'années plus tôt.

Globalement, parmi les langages informatiques, on retiendra pour l'instant les grands types suivants.

- **Le langage machine :** c'est le langage plus proche de la machine, c'est une succession de 0 et de 1 qui sont traités par les composants de l'ordinateur (microprocesseur, mémoire RAM, carte mère, disque dur...)

- **L'assembleur :** *assembly language* en anglais, c'est un langage de bas niveau proche du langage machine qui peut être directement interprété par le microprocesseur de l'ordinateur tout en restant lisible par un humain. Il est reconverti en langage machine par un programme appelé assembleur. Il permet d'aller programmer au plus près du matériel avec des instructions spécifiques à l'architecture. Par exemple, il permet de contrôler directement certains éléments du microprocesseur. L'assembleur est particulièrement utile pour les tâches nécessitant une optimisation au niveau matériel ou pour les systèmes embarqués où la performance et l'efficacité sont essentielles.

- **Les langages de haut niveau :** aussi appelés langages évolués, qui se rapprochent du langage naturel et sont donc plus lisibles pour un être humain. On y distingue trois sous-familles :
  - **Les langages de script ou interprétés :** Python, BASIC, PHP, JavaScript, Ruby, Perl. Un programme écrit en langage interprété n'est pas exécuté directement par la machine : un autre programme appelé interpréteur va interpréter et exécuter les lignes les unes après les autres ;
  - **Les langages semi-interprétés :** le Java. Il faut un compilateur intermédiaire traduisant le programme non pas en « langage-machine » mais en un code intermédiaire assez analogue à de l'assembleur. Pour le Java, le compilateur traduit le code source en code appelé *ByteCode*, qui sera traduit ensuite ligne par ligne par la « machine virtuelle java » en langage machine, et exécuté ligne par ligne ;
  - **Les langage compilés :** C, C++, Pascal. Un programme écrit en langage compilé est traduit directement en langage machine, sans intermédiaire, par un programme appelé compilateur. Le fichier produit porte le nom d'« exécutable », et peut justement être exécuté (lancé).

  *Remarque* : pour les langages évolués et le langage assembleur on parle de langage de programmation.

- **Les langage de balisage :** HTML/CSS, (X)HTML/CSS, XML, LaTeX, SGML... Ces langages reposent sur ce qu'on appelle des balises avec lesquelles on peut étiqueter des données (mots, texte, etc.) pour produire du sens (ceci est un paragraphe, ceci est un titre, une citation, etc.) et pour obtenir un rendu visuel (italique, gras, couleur du texte, etc.)

  *Remarque* : les langages de balisage ou de marquage sont dits de description.

- **Les langages de requête :** SQL. Ces langages qualifient le plus souvent les langages propres aux bases de données, ils sont représentés notamment par le SQL (en anglais : *Structured Query Language* / en français : langage structuré de requêtes), qui permet de gérer une base de données, par exemple l'interroger, y insérer des données ou en supprimer d'autres, lui demander de faire ressortir des données selon des critères qu'on lui fixe. Ce langage peut s'associer avec des langages comme le PHP. Le tandem PHP/ MySQL, étudiés séparément pour l'instant, sera approfondi pour ceux qui se destinent à la création et la gestion de contenu sur Internet.

Pour l'intégralité des exercices, il est recommandé d'utiliser :

- soit l'IDE auquel le lecteur est déjà habitué, Edupython pour Python ou autre ;
- soit, et c'est même mieux, un IDE en ligne capable de conserver les données, et de comprendre différents langages. À titre d'exemple, il existe le site `replit.com` sur lequel il est possible de créer un compte pour accéder à son travail de n'importe où.

Tous les programmes de ce livre sont téléchargeables sur le site de l'Éditeur.

Il est indispensable de **bien faire tous les exercices** un par un, et de se référer aux réponses ensuite, en particulier en cas de difficulté, pour pouvoir apprendre l'informatique correctement. Cette discipline s'apprend principalement par **la pratique**.

### ⋆ Exercice 1 (Premier programme)

Créer un nouveau programme dans l'IDE Python de votre choix. Nommer le projet « Exercice 1 », par exemple, mais le fichier doit s'appeler `main.py` : c'est le fichier qui sera lancé lorsque l'on demandera à l'IDE d'interpréter le code, en général à l'aide d'un bouton *RUN*, en appuyant sur la combinaison de touches ctrl+entrée, mais parfois aussi en appuyant sur la touche F5. On placera le projet dans un dossier, par exemple « Exercices Python ».
Copier le code suivant :

```
1 print("Bonjour Mademoiselle")
2 print("Bonjour Madame\n")
3 print("Bonjour Monsieur")
```

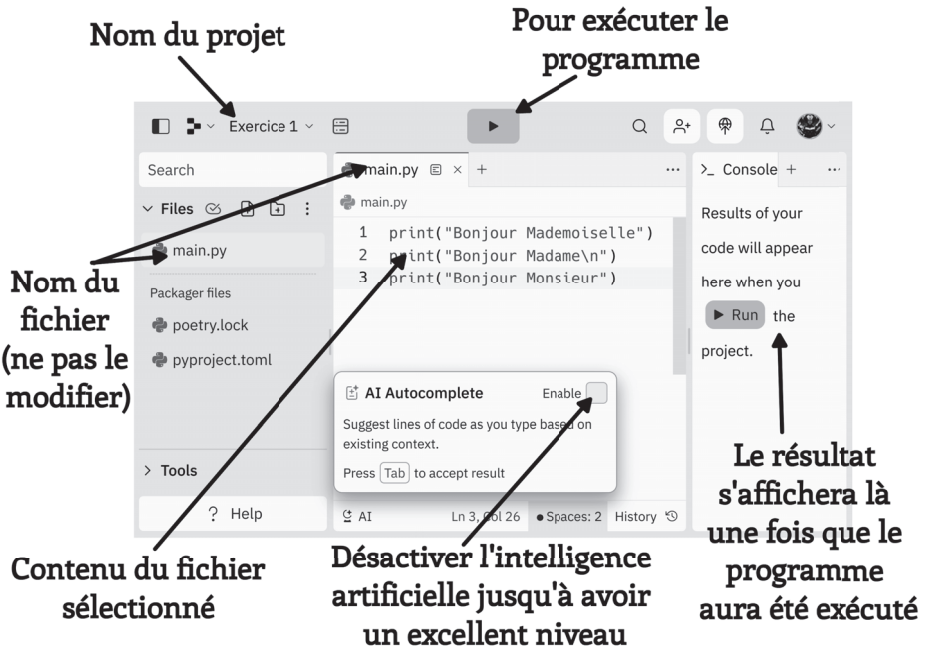
Exécuter.

**1)** Que fait ce programme ?

**2)** À quoi sert le `\n` ?

**3)** Que fait la fonction `print()` ?

Voici à quoi peut ressembler l'IDE avec le programme précédent, avant exécution :




### * Exercice 2 (Ajouter des commentaires)

Copier l'exercice 1 en exercice 2 et le nommer "Exercice 2". Le modifier comme indiqué dans les lignes suivantes :

```
"""Ce programme est un programme
test affichant trois informations à l'écran"""
print("Bonjour Mademoiselle")  # afficher bonjour Mademoiselle à l'écran et aller à la ligne
print("bonjour Monsieur \n")   # afficher bonjour Monsieur à l'écran et sauter une ligne
print("Ceci est votre premier programme")
```

Pourquoi met-on des commentaires dans un programme ?

### * Exercice 3 (Types de données)

```
print("L'ordinateur va maintenant vous demander votre âge et l'âge de l'un de vos parents \n")
ageEnfant = input("Quel est votre âge ? ")
ageParent = input("Quel est l'âge de l'un de vos parents ? ")
print("Vous avez "+ageEnfant+" ans.")
print("L'un de vos parents a "+ageParent+" ans.")

#calcul de la différence d age
differenceAge = int(ageParent) - int(ageEnfant)
print("Votre différence d'âge s'élève à "+str(differenceAge)+"
```

```
    ans.")
```

**1)** À quoi servent `int` et `str` ?

**2)** Prévoir ce qui sera affiché si l'on écrit `print("1"+"1")`, `print(1+1)`, et `print("1"+1)`. Le vérifier et le justifier.

# I) Les types de données

Les **variables** (et les constantes qui sont des données définies dont la valeur ne change jamais au cours de l'exécution du programme) sont des objets de base que les programmes manipulent constamment.
Chaque variable a un **type de données** tacitement affecté lors de son initialisation : nombre entier, nombre réel, caractère, suite de caractères, classe, tableau...
Les variables sont traditionnellement initialisées au début du programme.
Une variable est en fait une case de la mémoire vive (la RAM) qui est réservée quand on déclare la variable.
Si une variable doit changer de type, on déclare le nouveau type ; attention, ce système peut causer des erreurs très facilement si l'on ne prend aucune précaution.

**Exemple (Initialisation de variables)**

| | |
|---|---|
| `n=0` | Initialisation de `n` à `0`, type `int` |
| `n,p=1,2` | Initialisation de `n` à `1`, `p` à `2`, tous deux de type `int` |
| `n,p=1,2.0` | Initialisation de `n` à `1`, type `int`, `p` à `2.0`, type `float` |
| `s="Hello world"` | Initialisation de `s` à `"Hello world"`, type `str` |
| `n=3+2j` | Initialisation de `n` à $3+2i$, type `complex` |
| `n=["Lundi",3,"mai"]` | Initialisation de `n[0]` à `"Lundi"`, `n[1]` à `3`, `n[2]` à `"mai"`. `n` est de type `list` |

*Remarques* :

- `int` est l'abréviation d'*integer* qui signifie entier ;
- `float` signifie nombre à virgule flottante ;
- `str` est l'abréviation de *string*, une chaîne de caractères ;
- on peut changer le type d'une variable volontairement en utilisant des fonctions comme `str()` ou `int()` ;
- une liste commence son indexation à 0 et non à 1 ;
- on peut demander le type d'une variable avec la fonction `type()`.

Un élément de type `str` est une chaîne de caractères. L'ordinateur ne pouvant travailler qu'avec des nombres binaires, des nombres binaires doivent représenter des caractères. Par exemple, les caractères alphabétiques majuscules A, B, ..Z ont pour valeur décimale respective 65, 66, ..., 90 et les minuscules correspondantes a, b, ..z ont pour valeur décimale 97, 98, ..., 122. On appelle ce codage des caractères le code ASCII.

### ** Exercice 4 (Nombre de chiffres)

Le programme suivant affiche le nombre de caractères du prénom entré par l'utilisateur :

```
prenom = input("Quel est votre prénom ? ")
print("Votre prénom comporte "+str(len(prenom))+" caractères.")
```

Combien de chiffres composent le nombre $3^{1000}$ ?
*Remarque* : On écrit `3**1000` pour avoir 3 élevé à la puissance 1000 en Python.

Les opérations suivantes sur les nombres sont possibles :

```
c=a+b # c contiendra la somme du contenu des variables a et b
c=a-b # différence
c=a*b # produit
c=a/b # quotient dans la division décimale
c=a**b # puissance
c=a//b # quotient dans la division euclidienne (entiers)
c=a%b # reste dans la division euclidienne (entiers)
```

### ** Exercice 5 (Récupérer les chiffres)

En utilisant les bonnes opérations parmi les suivantes, écrire un programme qui demande un nombre à deux chiffres, puis renvoie le nombre de dizaines et le nombre d'unités qui composent ce nombre.

```
a=7
b=2
print("a=",a)
print("b=",b)

print("a+b=",a+b) # somme
print("a-b",a-b) # différence
print("a*b=",a*b) # produit
print("a/b=",a/b) # quotient dans la division décimale
print("a**b=",a**b) # puissance
print("a//b=",a//b) # quotient dans la division euclidienne (entiers)
print("a%b=",a%b) # reste dans la division euclidienne (entiers)
```

### ** Exercice 6 (Horloge)

Écrire un programme qui demande le nombre de minutes passées après minuit et renvoie l'heure.
Par exemple, pour 160, le programme doit renvoyer 2h40.

### * Exercice 7 (Code ASCII d'un caractère)

```
# Déclaration d'une variable appelée cara de type str
cara=input("Taper un caractère pour obtenir la conversion de la lettre en nombre (code ASCII)")

#Affichage du caractère et de son code.
print("Le code ASCII du caractère "+cara+" est ",ord(cara))
```

**1)** Tester avec la lettre `A`. Quel est le nombre associé ?

**2)** Tester avec la lettre `a`. Quel est le nombre associé ?

**3)** Pourquoi `A` et `a` n'ont-ils pas le même code ?

**4)** Que se passe-t-il si l'on essaye ce programme en entrant `"Aa"` ? Pourquoi ?

**5)** Pourquoi une virgule dans le `print()` ?

On utilise principalement trois bases en informatique pour écrire les nombres avec plus ou moins de chiffres, même si la base décimale (usuelle) est largement suffisante au quotidien. La base décimale est simplement écrite dans le code, tandis que la base binaire est précédée de `0b` et la base hexadécimale de `0x`.

- **Base 10 (décimale)** : la base 10 est le système numérique que nous utilisons couramment, basé sur dix chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Chaque position dans un nombre en base 10 représente une puissance de dix :