

# Avant-propos

L'idée générale de ce livre est de procéder avant tout par l'exemple et la pratique. Aussi les concepts comme la syntaxe sont généralement introduits par un petit script commenté avant d'être définis proprement. De plus, chaque section contient de nombreux exercices dont les corrigés sont donnés en fin d'ouvrage. Le code est séparé du corps du texte par un cadre sombre :

```
print('Hello World')
```

Le résultat de son exécution est quant à lui précédé de trois chevrons (>>>) pour reproduire l'apparence qui vous sera bientôt familière de la console. Notez que les espaces ne sont plus explicités comme ils l'étaient dans le code lui-même.

```
>>>  
Hello World
```



Ce livre a été conçu pour être accessible à toute personne n'ayant jamais programmé, c'est pourquoi certaines étapes initiales sont très détaillées. Cependant, nous avons aussi pensé aux étudiant-e-s ayant déjà une expérience d'un autre langage comme C, PHP, Java, R ou encore Scilab. Pour ceux et celles-ci, nous avons ici et là ajouté quelques encarts (dans un cadre gris clair, avec un symbole d'engrenage, comme ce paragraphe) essayant d'expliquer les différences entre le langage auquel ils et elles sont habitué-e-s, et le fonctionnement de Python. Les lecteurs et lectrices débutant la programmation peuvent sereinement ignorer tous ces blocs précédés d'un engrenage.

Les exercices ne nécessitent pas pour être résolus de connaissance mathématique au-delà d'un baccalauréat scientifique ; en revanche il est probable que ceux des tous premiers chapitres intéresseront davantage des étudiant-e-s ayant un peu de familiarité

avec l'arithmétique et la combinatoire. Au fur et à mesure de votre progression, vous verrez les mathématiques disparaître car les objets informatiques manipulés commencent à être suffisamment intéressants en eux-mêmes.

Il y a deux catégories d'exercices : les exercices d'application directe, dispersés au fil du cours, et les problèmes plus difficiles, récapitulés à la fin de chaque section. Ceci étant, la difficulté des exercices d'application va un peu croissante avec le livre, tenant compte du fait que vous progressez – essentiellement, vous êtes capables de manipuler un code de taille plus importante.

Les chapitres 1 à 3 contiennent la base que doit connaître toute personne souhaitant acquérir rapidement des bases en Python, quelle que soit sa motivation. Les chapitres 4 à 6 sont plutôt destinés à ceux et celles aspirant à un métier dans l'informatique, car les principes qui y sont développés restent pertinents quel que soit le langage. Les chapitres 7 et 8 peuvent intéresser tout le monde, mais seront vitaux pour les futur data scientists. Enfin, les deux derniers chapitres introduisent l'objet, notion fondamentale pour l'informatique de gestion.

# Table des matières

<b>Avant-propos .....</b>	<b>3</b>
<b>1 Découvrir le langage .....</b>	<b>9</b>
1.1 Installer et exécuter Python .....	9
1.2 Utiliser Python comme calculatrice .....	11
1.3 Structures de contrôle .....	15
1.4 Résumé .....	20
1.5 Exercices plus avancés .....	20
<b>2 Une première structure : la liste .....</b>	<b>23</b>
2.1 Définition .....	23
2.2 Opérations sur les listes .....	27
2.3 La mémoire et les listes .....	33
2.4 Résumé .....	35
2.5 Exercices plus avancés .....	36
<b>3 Programmer avec des fonctions .....</b>	<b>37</b>
3.1 Définir et utiliser des fonctions .....	37
3.2 Utiliser les fonctions des bibliothèques usuelles .....	42
3.3 Quelques aspects un peu plus subtils des fonctions .....	45
3.4 Résumé .....	50
3.5 Exercices plus avancés .....	51

<b>4 Un peu d'algorithmique .....</b>	<b>53</b>
4.1 Algorithmes récursifs .....	53
4.2 Programmation dynamique .....	57
4.3 Exemple détaillé : trouver le plus court chemin .....	59
4.4 Résumé .....	63
4.5 Exercices plus avancés .....	64
<b>5 Maîtriser le temps de calcul .....</b>	<b>67</b>
5.1 Quand l'optimisation devient nécessaire .....	67
5.2 La complexité au pire des cas .....	72
5.3 Méthodes de calcul .....	75
5.4 Résumé .....	79
5.5 Exercices plus avancés .....	80
<b>6 Structures de données .....</b>	<b>81</b>
6.1 Quelques alternatives à la liste .....	81
6.2 Le mapping .....	86
6.3 Bien choisir sa structure .....	88
6.4 Résumé .....	91
6.5 Exercices plus avancés .....	92
<b>7 Analyser du texte .....</b>	<b>95</b>
7.1 Conversion de chaînes .....	95
7.2 Rechercher et remplacer .....	96
7.3 Expressions rationnelles .....	99
7.4 Résumé .....	105
7.5 Exercices plus avancés .....	106

<b>8 Importer et exporter des données .....</b>	<b>109</b>
8.1 Manipulation de fichiers .....	109
8.2 Bases de données .....	115
8.3 Résumé .....	120
8.4 Exercices plus avancés .....	120
<b>9 Créer des Classes en Python .....</b>	<b>123</b>
9.1 Vers l'objet .....	123
9.2 Classes, constructeurs et méthodes .....	125
9.3 Séparation interface et contenu .....	127
9.4 Composer des objets, déléguer des méthodes .....	130
9.5 Résumé .....	136
9.6 Exercices plus avancés .....	137
<b>10 Programmation orientée objet .....</b>	<b>139</b>
10.1 L'encapsulation .....	139
10.2 L'héritage .....	143
10.3 Tout est objet .....	145
10.4 Résumé .....	149
10.5 Exercices plus avancés .....	150
<b>11 Corrigés des exercices .....</b>	<b>151</b>
11.1 Chapitre 1 .....	151
11.2 Chapitre 2 .....	159
11.3 Chapitre 3 .....	167
11.4 Chapitre 4 .....	175
11.5 Chapitre 5 .....	180

11.6 Chapitre 6 .....	188
11.7 Chapitre 7 .....	197
11.8 Chapitre 8 .....	204
11.9 Chapitre 9 .....	221
11.10 Chapitre 10 .....	231
<b>Bibliographie .....</b>	<b>245</b>
<b>Index.....</b>	<b>246</b>

# Chapitre 1

## Découvrir le langage

### 1.1 Installer et exécuter Python

Nous supposons que vous souhaitez commencer à programmer au plus vite sans vous préoccuper trop avant des questions d’environnement, c’est pourquoi nous commençons par quelques instructions qui pourront vous sembler un peu directives. Lorsque la question de choisir l’environnement de travail qui vous convient le mieux vous paraîtra pertinente, vous trouverez facilement sur le web un comparatif des différentes possibilités.

#### Avec un IDE

Si vous travaillez sur une machine pour laquelle vous disposez de droits d’administrateur, par exemple votre ordinateur personnel, nous vous recommandons d’installer successivement un interpréteur Python et un environnement de travail intégré (IDE).

Le premier est le minimum absolu pour pouvoir exécuter un script Python. Vous pouvez le télécharger sur le site de la *Python Software Foundation*. Les exemples de ce livre ont été rédigés avec la version 3.7, mais nous vous recommandons d’installer la version stable la plus récente – il n’y aura pas de problème de rétrocompatibilité pour les versions 3.x entre elles.

Il est théoriquement possible de travailler avec le seul interpréteur et une console. Toutefois pour des raisons de confort nous recommandons d’installer également un IDE compatible avec votre système d’exploitation tel que *Thonny*, *Pyzo*, *PyCharm* ou encore *IDLE* – ce dernier étant fourni avec la distribution Python, mais pas nécessairement le plus facile à prendre en main. Vous disposerez ainsi d’une interface unique dans laquelle écrire votre code et lire les résultats, ainsi que d’un grand nombre d’autres fonctionnalités que vous découvrirez au fur et à mesure de votre progression.

Alternativement au fait d’installer sur votre machine l’interpréteur et l’IDE, vous avez la possibilité d’utiliser une interface en ligne, comme `pythonanywhere.com`, qui vous offre les fonctionnalités d’un IDE à distance.

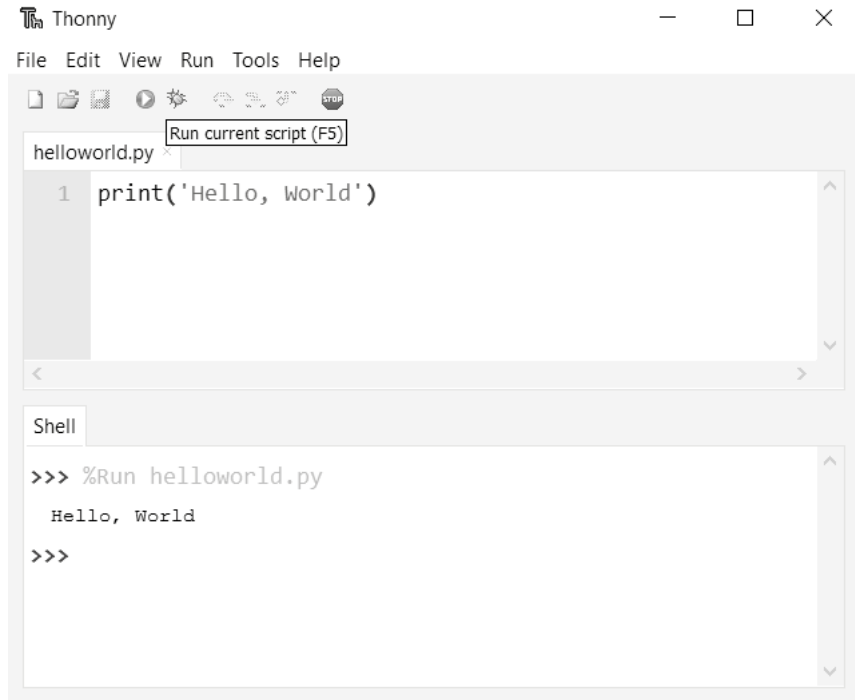


FIGURE 1.1 – Exemple d’interface (Thonny). Le code est écrit dans l’éditeur intégré et le résultat affiché dans la console (Shell) juste en-dessous.

Dans tout ce livre, nous ne ferons appel qu’à des bibliothèques fournies dans la distribution de base, aussi nous n’aurons pas à nous préoccuper d’installer des modules complémentaires. Si vos centres d’intérêts vous amènent à avoir besoin de bibliothèques externes, par exemple pour le calcul scientifique, vous pourrez vous référer à la documentation de *pip*.

## Sans IDE

Si vous ne souhaitez pas ou ne pouvez pas installer d’IDE, par exemple parce que vous travaillez dans une salle informatique de votre université, c’est un tout petit peu plus compliqué, mais pas insurmontable. Il faut cependant au minimum qu’un interpréteur Python soit installé sur la machine <sup>1</sup>.

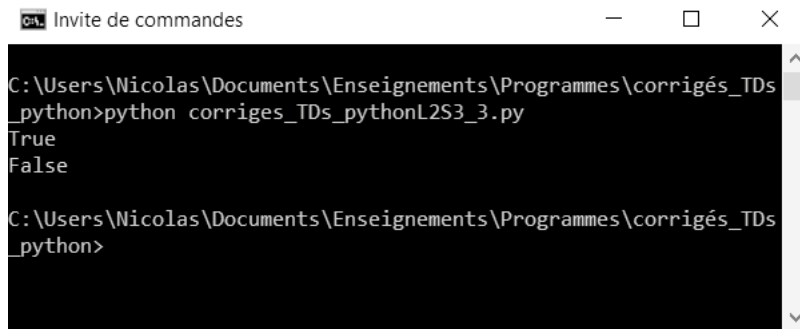
Vous écrirez votre code dans un éditeur de texte quelconque, de préférence proposant une fonctionnalité de coloration syntaxique et ne générant pas de problème d’interprétation des caractères spéciaux. *Notepad++* sous Windows, *textwrangler*

---

1. Sachez en dernier recours que toute distribution Linux intègre un interpréteur Python, pas nécessairement la version la plus à jour cependant (typiquement, s’il s’agit d’une version de Python 2.x et non 3.x il vous faudra modifier un peu la syntaxe utilisée dans ce livre).



sous Mac OS ou encore *scribes*, *gedit* sous Linux sont de bons exemples. Vous enregistrerez ensuite votre fichier sous un nom de fichier portant l'extension `.py`. Ceci fait, vous ouvrirez un terminal et écrirez simplement `python xxx.py` où `xxx` est le nom de votre fichier, qui sera alors exécuté. Le retour sera affiché directement dans le terminal.

A screenshot of a Windows terminal window titled "Invite de commandes". The terminal shows the following text:

```
C:\Users\Nicolas\Documents\Enseignements\Programmes\corrigés_TDs_python>python corrigés_TDs_pythonL2S3_3.py
True
False

C:\Users\Nicolas\Documents\Enseignements\Programmes\corrigés_TDs_python>
```

FIGURE 1.2 – Exemple de terminal (Windows).

## 1.2 Utiliser Python comme calculatrice

Même dans l'hypothèse où vous n'avez aucune notion d'algorithmique, vous êtes certainement familier-ère avec le fonctionnement d'une calculatrice. Le moins qu'on puisse attendre de Python est qu'il vous permette d'en simuler une. Cela nous fournit l'occasion de notre premier script :

```
x = 3798
y = 48518
z = y * x
print(z)
```

Lequel produit dans la console le résultat suivant :

```
>>>
184271364
```

Les lignes une et deux affectent la valeur à deux variables, la troisième en effectue la multiplication, la quatrième utilise la fonction `print()` pour provoquer un affichage dans la console. Le même résultat aurait pu être obtenu en écrivant directement `print(3798 * 48518)`.



Si vous avez déjà programmé dans d'autres langages, vous remarquerez que la déclaration de variable est simplifiée à l'extrême : pas de mention de type ni d'allocation de mémoire, pas même de mot-clef indiquant la déclaration. Cela ne signifie pas pour autant que Python n'est pas un langage typé. Simplement, le type est résolu à partir du contexte.

```
x = 3798
y = 48518
z = y * x
print(type(z))
s = 'Hello_World'
print(type(s))
```

```
>>>
<class 'int'>
<class 'str'>
```

Remarquez que Python attribue aux variables des types (ici `int` pour entier et `str` pour chaîne de caractères), lesquels peuvent être retrouvés à partir de la fonction `type()`. Ces types conditionnent le comportement des fonctions, opérateurs, etc. qui leur sont appliqués. Nous aurons amplement l'occasion de revenir sur cette notion au fur et à mesure de notre progression, mais voyons tout de suite un exemple très simple qui illustre combien il est important d'avoir cette idée de type en tête quand on programme.

```
x, y, z, t = 2, 1/3, '2', '1/3'
print(type(x), type(y), type(z), type(t))
print(x+y)
print(z+t)
```

```
>>>
<class 'int'> <class 'float'> <class 'str'> <class 'str'>
2.3333333333333335
21/3
```

À la ligne 3, nous effectuons une addition entre un entier et un décimal (`float`) et nous obtenons le résultat attendu – à part l’arrondi un peu bizarre. À la ligne 4, nous utilisons le même opérateur `+`, mais ici entre deux chaînes de caractères : le résultat obtenu est non pas une addition mais une concaténation. Si vous faites l’expérience, vous remarquerez enfin que tenter `x+z` provoque un affichage d’erreur :

```
>>>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Retenons donc de tout ceci une chose : un opérateur fonctionne différemment, voire pas du tout, selon le type des objets auxquels on l’applique. Il est donc important de distinguer le type des objets manipulés, par exemple l’entier `3` de la chaîne de caractères `'3'`.



Contrairement à certains autres langages comme PHP qui effectueraient une conversion implicite, Python explique qu’il ne sait pas comment interpréter cet opérateur entre deux objets pour lesquels le comportement prévu est radicalement différent : il préfère lancer une exception plutôt que de risquer la propagation d’un comportement non prévu. Toutefois lorsqu’un type est inclus dans un autre (`int` dans `float`), il effectue une conversion vers le type le plus général.

Pour le moment, nous n’avons utilisé que les opérations élémentaires, mais il est certain que Python peut également gérer des fonctions mathématiques plus complexes.

```
from math import *
print(sqrt(2), log(7.12), factorial(6))
```

```
>>>
1.4142135623730951 1.9629077254238845 720
```

Pas de surprise dans le comportement des fonctions racine carrée (`sqrt`), logarithme (`log`) et factorielle (`factorial`). Notez cependant la première ligne qui sert à importer ces fonctions depuis la librairie `math`, sans quoi le code renverrait un message d’erreur. Nous verrons un peu plus loin le fonctionnement des librairies, mais l’idée ici est simple : seules quelques fonctions sont immédiatement disponibles pour Python, les autres doivent être explicitement importées depuis une librairie afin que l’interpréteur sache exactement ce qu’il doit faire.