**Chapitre 1 : Premiers pas avec Python**

Dans ce premier chapitre, j’ai appris les bases de Python, surtout pour le calcul scientifique. Il s'agit d'une introduction pratique pour ceux qui débutent, comme moi, mais aussi pour ceux qui connaissent déjà un autre langage.

**1. Installation et environnement**

Avant de coder, il faut bien s’équiper. Le livre recommande Python version 3.7 ou plus, avec des modules comme NumPy, SciPy, matplotlib (pour les calculs et les graphes), ainsi qu’un bon éditeur comme **Spyder**. Le plus simple est d’installer **Anaconda**, qui regroupe tout ça.

**2. Utilisation de Spyder et IPython**

Spyder est un environnement très pratique : on y trouve un éditeur à gauche, une console IPython en bas, et une aide à droite. IPython est une version améliorée du shell Python, très utile pour tester rapidement du code.

On peut aussi exécuter des fichiers .py avec run nom\_du\_fichier.py dans IPython. Pour les débutants comme moi, c’est super pratique.

**3. Types de données de base**

Python est simple mais puissant. J’ai découvert les types suivants :

* **Nombres** : entiers, réels, complexes.
* **Chaînes de caractères** (strings) : avec guillemets simples ou doubles.
* **Listes** : très utilisées, on peut y mettre différents types, même d'autres listes.
* **Booléens** : True ou False, utilisés avec des comparaisons.

Les opérations sont intuitives : +, \*, ==, <, etc. Et la fonction print() permet d’afficher les résultats.

**4. Boucles et conditions**

La boucle for permet de répéter une action plusieurs fois, par exemple sur une liste ou avec range(n). On peut aussi utiliser break pour sortir de la boucle, et else si la boucle va jusqu’au bout.

Les conditions if, else, elif permettent d’exécuter des actions différentes selon les cas. Cela rend le programme plus intelligent.

**5. Fonctions**

Avec le mot-clé def, on peut créer des fonctions pour réutiliser du code. Par exemple :

python

CopierModifier

def f(x):

return 2 \* x + 1

On peut ensuite appeler f(2) et ça renvoie 5. Très utile pour organiser son code.

**6. Scripts et modules**

Un fichier .py avec plusieurs lignes de code est appelé un script. On peut créer ses propres fonctions, les mettre dans un fichier, et les réutiliser avec import. C’est comme créer sa propre boîte à outils.

**7. Interpréteur Python**

L’interpréteur lit et exécute le code ligne par ligne. Les erreurs de syntaxe sont détectées avant l’exécution. Mais certaines erreurs, appelées erreurs d’exécution (runtime errors), n’apparaissent que quand on exécute une fonction.

**Conclusion**

Ce chapitre m’a permis de poser les bases de Python. Maintenant, je comprends mieux comment installer les outils, créer des programmes simples, utiliser des fonctions, des conditions et des boucles. Ce n’est que le début, mais c’est déjà très encourageant.