**Chapitre 2 : Variables et types de base**

Ce chapitre m’a permis de bien comprendre les éléments fondamentaux du langage Python, notamment comment sont gérées les **variables** et les **types de données de base**. Ce sont des bases indispensables pour tout calcul scientifique.

**Les variables**

En Python, une variable est simplement une **référence vers un objet**. Il suffit de faire une affectation comme x = 3 pour la créer. Il n’y a **pas besoin de spécifier le type** à l’avance, Python le devine automatiquement. J’ai aussi appris qu’on pouvait faire des affectations multiples comme a = b = c = 1.

Les noms de variables peuvent contenir lettres, chiffres, et underscore \_ mais ne doivent **pas commencer par un chiffre**. Et attention, les noms sont **sensibles à la casse** (Nom ≠ nom).

**Types numériques**

Python gère plusieurs types de nombres :

* int : les entiers (comme ℤ), sans limite de taille autre que la mémoire de l’ordi
* float : nombres à virgule (approximation de ℝ), représentés en mémoire en virgule flottante
* complex : les nombres complexes (ℂ), très utiles en calcul scientifique

**Les entiers**

Simple à manipuler. Les opérations comme +, -, \* rendent un int. La division / donne un float, mais // donne un entier.

**Les flottants**

Ils sont souvent approximatifs à cause de la manière dont ils sont stockés en mémoire. Exemple classique :

python

CopierModifier

0.4 - 0.3 == 0.1 # False !

J’ai appris qu’il fallait être très prudent avec les comparaisons de float. Il existe aussi des constantes spéciales comme inf (infini) et nan (not a number). On peut les tester avec math.isinf() ou math.isnan().

**Détails sur les floats**

Python utilise 64 bits pour représenter un float (float64). Le plus petit nombre qu’on peut représenterest très proche de zéro, mais il y a une **zone vide autour de zéro** qu’on appelle "gap" des flottants. Très utile à savoir pour éviter des bugs subtils.

**Les nombres complexes**

En Python, un nombre complexe s’écrit avec j, comme z = 3 + 2j. On peut accéder à sa partie réelle avec .real et sa partie imaginaire avec .imag. Ils sont aussi très pratiques à manipuler, avec des fonctions comme z.conjugate().

**Les booléens**

Les booléens (True et False) sont essentiels pour les tests logiques. On les obtient avec des comparaisons (a > b, etc.). Python utilise les mots and, or, et not pour les opérateurs logiques.

J’ai aussi vu que **presque tout peut être converti en booléen** : les listes vides, 0, les chaînes vides sont considérés comme False, tout le reste est True.

Petit piège : bool([1, 2]) marche, mais bool([1, 2, 3]) dans un contexte de condition provoque une erreur si c’est un array NumPy.

**Les chaînes de caractères (strings)**

Les strings sont délimitées par ' ou ", et pour du texte sur plusieurs lignes on utilise """.

Très pratique : on peut faire de la concaténation ('Hello ' + 'world'), de la répétition ('Hi' \* 3), des découpes (s[:3], s[-1], etc.) et aussi **du formatage**.

J’ai appris à utiliser les f-strings :

python

CopierModifier

name = "Aminata"

print(f"Bonjour {name}")

Et aussi l’ancienne méthode avec format(). On peut contrôler la précision d’un nombre :

python

CopierModifier

f"{pi:.2f}" # 3.14

**Ce que je retiens**

Ce chapitre m’a montré à quel point Python est **souple** mais qu’il faut aussi être **attentif aux détails**, notamment avec les flottants et les conversions booléennes. Ces notions sont simples mais très puissantes, et je sens qu’elles vont beaucoup m’aider dans les chapitres suivants, surtout en calcul scientifique.