**Chapitre 5 : Calcul Symbolique avec SymPy**

Ce chapitre m’a permis de découvrir une facette très puissante de Python : le **calcul symbolique**, c’est-à-dire la manipulation d'expressions mathématiques comme des objets, un peu comme on le ferait à la main mais en utilisant l’ordinateur. Le module **SymPy** est dédié à cela. Contrairement à NumPy qui travaille sur des nombres (et donc donne des résultats numériques), SymPy peut traiter les symboles, simplifier des équations, faire des dérivées, des intégrales, résoudre des équations algébriques, et même afficher des formules mathématiques joliment.

**1. Déclaration de symboles**

La base du calcul symbolique, c’est la notion de **symbole**. On crée des symboles avec symbols() :

python

CopierModifier

from sympy import symbols

x, y = symbols('x y')

Une fois qu'on a défini ces variables symboliques, on peut créer des expressions mathématiques comme x + 2\*y, et SymPy les gère comme des objets manipulables.

**2. Manipulation d’expressions**

Ce que j’ai apprécié, c’est qu’on peut vraiment *jouer* avec les expressions : les simplifier (simplify()), les développer (expand()), les factoriser (factor()), ou encore les évaluer pour une valeur donnée (subs()).

Par exemple :

python

CopierModifier

expr = (x + 1)\*\*2

developpee = expr.expand()

On peut même substituer des valeurs :

python

CopierModifier

expr.subs(x, 3)

**3. Calcul différentiel**

SymPy peut dériver n’importe quelle expression symbolique :

python

CopierModifier

from sympy import diff

diff(x\*\*2 + 2\*x + 1, x)

Ce qui est génial, c’est qu’on n’a pas besoin de connaître la formule par cœur — SymPy le fait pour nous.

**4. Calcul intégral**

Tout comme pour les dérivées, on peut calculer des **intégrales** :

python

CopierModifier

from sympy import integrate

integrate(x\*\*2, x)

Et même des intégrales définies :

python

CopierModifier

integrate(x\*\*2, (x, 0, 2))

**5. Résolution d’équations**

Une autre fonctionnalité très utile : on peut **résoudre des équations** algébriques ou différentielles.

Par exemple :

python

CopierModifier

from sympy import solve

solve(x\*\*2 - 4, x)

Cela donne les racines de l’équation. Très pratique pour éviter les erreurs de calcul à la main.

**6. Affichage mathématique**

SymPy peut aussi afficher les résultats en **notation mathématique lisible**, ce qui est parfait pour les rapports ou les interfaces :

python

CopierModifier

from sympy import pretty, pprint

pprint(expr)

Et même produire du LaTeX si on veut intégrer les résultats dans un document scientifique.

**Ce que je retiens**

Ce chapitre m’a vraiment montré que Python ne sert pas uniquement à manipuler des données numériques ou faire des boucles. Avec SymPy, on peut faire du **calcul symbolique avancé**, comme on le ferait dans un logiciel de mathématiques comme Mathematica ou Maple, mais directement dans un script Python.

Les choses que j’ai retenues :

* La différence entre le calcul numérique (avec NumPy) et symbolique (avec SymPy).
* La puissance des fonctions comme diff, integrate, solve, expand, simplify, etc.
* La facilité avec laquelle on peut créer et transformer des expressions mathématiques.
* L’intérêt du rendu LaTeX et mathématique lisible.

Ce chapitre est essentiel si on veut faire du calcul formel en Python, que ce soit pour des études mathématiques, des modèles symboliques, ou simplement pour automatiser des démonstrations.