on termine en beauté avec un concept très puissant : la **surcharge d'opérateurs** ! 🚀

**💥 Concept #16 — Surcharge d’opérateurs (\_\_add\_\_, \_\_str\_\_, etc.)**

**🧠 1. Définition**

La **surcharge d’opérateurs** te permet de définir ce que certains opérateurs comme +, -, ==, \* et bien d’autres font quand tu les utilises avec tes objets.  
Cela rend ton code plus **expressif** et **intuitif**.

Exemple : Si tu as une classe Point, tu peux faire point1 + point2 pour additionner les coordonnées.

**📘 Exemple simple avec + (additionner deux objets) :**

python

CopyEdit

class Point:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def \_\_add\_\_(self, other):

return Point(self.x + other.x, self.y + other.y)

def \_\_str\_\_(self):

return f"Point({self.x}, {self.y})"

point1 = Point(2, 3)

point2 = Point(4, 5)

resultat = point1 + point2

print(resultat) # Affiche : Point(6, 8)

**✨ Ce que fait chaque méthode spéciale :**

* \_\_add\_\_ : définit l'addition de deux objets.
* \_\_str\_\_ : définit la représentation textuelle d'un objet, ce qui est utilisé par print().

**🧩 Surcharge d’autres opérateurs populaires :**

**🔹 \_\_eq\_\_ — pour l’égalité (==) :**

python

CopyEdit

class Point:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def \_\_eq\_\_(self, other):

return self.x == other.x and self.y == other.y

Tu peux comparer tes objets avec ==.

**🔹 \_\_lt\_\_ — pour les comparaisons (<, >) :**

python

CopyEdit

class Point:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def \_\_lt\_\_(self, other):

return (self.x + self.y) < (other.x + other.y)

Tu peux comparer tes objets avec des opérateurs < ou >.

**🔹 \_\_mul\_\_ — pour la multiplication (\*) :**

python

CopyEdit

class Point:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def \_\_mul\_\_(self, other):

return Point(self.x \* other, self.y \* other)

Tu peux multiplier tes objets avec un nombre.