Operator Overloading in Python

What is Operator Overloading?

- Python allows you to **define custom behavior for standard operators** (like +, -, *, etc.) for your own classes.
- This is called **operator overloading**.
- **Why?** So that operators can work with **user-defined objects** just like they work with built-in types.

Solved Example 1: Overloading + operator

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    # Overload the + operator
    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

    def display(self):
        print(f"({self.x}, {self.y})")

p1 = Point(2, 3)
p2 = Point(4, 5)
p3 = p1 + p2   # Calls p1.__add__(p2)
p3.display()
```

Output:

(6, 8)

Explanation:

- __add__ is a **special method** in Python to define +.
- When we do p1 + p2, Python internally calls p1.__add__(p2).
- Result is a **new Point object** with summed coordinates.

Other Common Operators You Can Overload

Operator	Special Method
+	__add__
-	__sub__
*	__mul__
/	__truediv__

Operator Overloading in Python

Operator	Special Method
==	__eq__
<	__lt__
>	__gt__

Solved Example 2: Overloading * operator for a custom class

```
class Number:
    def __init__(self, value):
        self.value = value

    # Overload the * operator
    def __mul__(self, other):
        return Number(self.value * other.value)

    def display(self):
        print(self.value)

# Create two numbers
n1 = Number(5)
n2 = Number(10)

# Multiply using * operator
n3 = n1 * n2  # Calls n1.__mul__(n2)
n3.display()
```

Output:
50

Explanation:
- The * operator is customized to **multiply the values** of two Number objects.
- n1 * n2 returns a **new Number object** with the product.

Key Points
- Enables custom behavior for operators with objects.
- Uses special methods (also called magic methods) like __add__, __sub__, __mul__.
- Helps make user-defined objects behave like built-in types.
- Supports polymorphism in object-oriented programming.

Common Operators and Their Methods

Operator Overloading in Python

Operator	Method
+	__add__
-	__sub__
*	__mul__
/	__truediv__
==	__eq__
<	__lt__
>	__gt__