**1. Method Overloading: (Design time polymorphism)**

* **Definition**: Method overloading refers to the ability to define multiple methods in a class with the same name but different signatures (number or type of parameters).

class Calculator:

def add(self, a, b=0, c=0):

return a + b + c

calc = Calculator()

print(calc.add(5)) # Output: 5

print(calc.add(5, 10)) # Output: 15

print(calc.add(5, 10, 15)) # Output: 30

In the example, add() is overloaded in practice by using default values, allowing it to work with different numbers of arguments.

**Example (using \*args)**:

class Calculator:

def add(self, \*args):

return sum(args)

calc = Calculator()

print(calc.add(5)) # Output: 5

print(calc.add(5, 10)) # Output: 15

print(calc.add(5, 10, 15)) # Output: 30

In this case, \*args is used to handle any number of arguments.

**2. Method Overriding: (Run time polymorphism)**

* **Definition**: Method overriding occurs when a child class provides a specific implementation of a method that is already defined in its base class. When we call an overridden method, the subclass’s method is used rather than the parent class’s method.

class Animal:

def sound(self):

return "Some generic sound"

class Dog(Animal):

def sound(self):

return "Bark"

class Cat(Animal):

def sound(self):

return "Meow"

dog = Dog()

cat = Cat()

print(dog.sound()) # Output: Bark

print(cat.sound()) # Output: Meow

In this case, both Dog and Cat override the sound() method of the Animal class.

**Differences between Overloading and Overriding:**

* **Overloading** is about defining multiple methods with the same name but different parameters within the same class (handled through argument tricks in Python).
* **Overriding** is about redefining a method from the base class in a child class, providing a specific implementation in the child class.