OOP - Polymorphism in Python

1. What is Polymorphism?

Polymorphism means 'many forms'. It allows the same method, function, or operator to behave differently based on the object or data it is acting upon.

2. Types of Polymorphism in Python

a. Compile-Time Polymorphism (Static)

In Python, compile-time polymorphism is achieved through:

- Operator overloading
- Default arguments

Python doesn't support traditional method overloading like Java.

Example: Operator Overloading

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

    def __add__(self, other):
        return Number(self.value + other.value)

    def __str__(self):
        return str(self.value)

n1 = Number(10)
n2 = Number(20)
print(n1 + n2)  # Output: 30
```

Example: Default Arguments (Simulated Overloading)

OOP - Polymorphism in Python

```
def greet(name=None):
    if name:
        print(f"Hello, {name}")
    else:
        print("Hello!")

greet()          # Hello!
greet("Akash")   # Hello, Akash
```

b. Run-Time Polymorphism (Dynamic)

Achieved using method overriding in inheritance. The method that gets executed is determined at runtime based on the object's class.

Example: Method Overriding

```
class Animal:
    def sound(self):
        print("Some generic sound")

class Dog(Animal):
    def sound(self):
        print("Bark")

class Cat(Animal):
    def sound(self):
        print("Meow")

def make_sound(animal):
    animal.sound()
```

```
make_sound(Dog())   # Bark
make_sound(Cat())   # Meow
```

3. Duck Typing (Python Specific)

Duck typing is a concept where the object's suitability is determined by the presence of a method or behavior rather than the actual type of the object.

Example: Duck Typing

```
class Bird:
    def fly(self):
        print("Flying with wings")

class Plane:
    def fly(self):
        print("Flying with engines")

def take_off(flyer):
    flyer.fly()

take_off(Bird())   # Flying with wings
take_off(Plane())  # Flying with engines
```

4. Summary: Compile-Time vs Run-Time Polymorphism

Compile-Time Polymorphism:

- Resolved before program runs.
- Achieved using operator overloading and default args.
- Example: __add__ method

OOP - Polymorphism in Python

Run-Time Polymorphism:

- Resolved while program runs.
- Achieved using method overriding.
- Example: sound() in Dog/Cat

Duck Typing:

- No need for inheritance, just method presence.