Polymorphism in Python

Understanding Polymorphism in Object-Oriented Programming

What is Polymorphism?

- Polymorphism means 'many shapes' and is a key concept in Object-Oriented Programming.
- - Allows methods, functions, or operators to behave differently based on input.
- - Promotes flexibility and reusability in code.

Types of Polymorphism in Python

- 1. Polymorphism with Functions and Methods
- 2. Polymorphism with Operators (Operator Overloading)
- 3. Polymorphism with Inheritance (Method Overriding)
- 4. Polymorphism with Built-in Functions (e.g., len())

Examples of Polymorphism

- 1. Function/Method Polymorphism:
-    - Example: Different animal classes implementing a 'speak' method differently.
- 2. Operator Overloading:
-    - Example: '+' used for both numbers and strings.
- 3. Inheritance:
-    - Example: Subclasses overriding parent class methods.

Advantages of Polymorphism

- - Increases code readability and flexibility.
- - Promotes code reusability.
- - Reduces complexity by allowing a unified interface for different data types.

```
class Animal:
    def speak(self):
        raise NotImplementedError("Subclass must implement abstract method")
# Dog inherited from animal class
class Dog(Animal):
    def speak(self):
        return "Woof!"
```

```
• # Cat inherited from animal class
• class Cat(Animal):
•     def speak(self):
•         return "Meow!"

• # Cow inherited from animal class
• class Cow(Animal):
•     def speak(self):
•         return "Moo!"
• # Using Polymorphism
• animals = [Dog(), Cat(), Cow()]
• for animal in animals:
•     print( animal. speak())
```

Thanks