Python OOP

16-dars

Polymorphism (Koʻp shakllilik)

Polymorphism (koʻp shakllilik) — bu dasturiy tushuncha boʻlib, u orqali bir xil nomdagi metodlar yoki funksiya chaqirilganda turli xil obyektlar har xil tarzda javob qaytarishi mumkin. Polymorphism orqali bir xil metod yoki funksiyaga turli classlarda har xil xatti-harakatlar berish mumkin, ya'ni turli shakllarda bir xil metoddan foydalanish imkonini beradi.

Polymorphismning asosiy g'oyasi shundaki, u dasturning moslashuvchanligi va kengaytirilishini osonlashtiradi. Dasturda umumiy funksionallikni alohida harakatlar bilan birlashtiradi.

Polymorphism'ning ikki turi

Compile-time polymorphism (statik polymorphism) — funksiyalarning bir nechta ko‘rinishlari (overloading) yordamida amalga oshiriladi, ammo Python’da bu funksiyalarni overload qilish qo‘llanilmaydi.

Run-time polymorphism (dinamik polymorphism) — bir xil metodni meros oluvchi classlarda har xil qilib implement qilish bilan amalga oshiriladi. Python asosan run-time polymorphism bilan ishlaydi.

Polymorphism va Meros Olish

Polymorphism meros olish bilan chambarchas bog‘liq. Bitta ota classdagi metod farzand classlarda turli xil usullar bilan implement qilinadi. Bu hodisa **method overriding** deb ataladi, ya'ni farzand class ota classdagi metodni qayta yozadi.

Bu yerda `speak()` metodi barcha classlarda mavjud, lekin har bir class metodni o‘ziga xos tarzda amalga oshiradi. Shu sababli, polymorphism yordamida bir xil metod turli classlarda turlicha natijalar beradi.

```
class Animal:
    def speak(self):
        return "Animal makes a sound."

class Dog(Animal):
    def speak(self):
        return "Dog barks."

class Cat(Animal):
    def speak(self):
        return "Cat meows."

# Polymorphic behavior
animals = [Dog(), Cat(), Animal()]

for animal in animals:
    print(animal.speak())

# Output:
# Dog barks.
# Cat meows.
# Animal makes a sound.
```

Polymorphism Funksiyalar bilan

Polymorphism faqat class metodlariga emas, balki funksiyalarda ham amalga oshirilishi mumkin. Har xil turdagi obyektlar bir xil nomli funksiyalar orqali turli xatti-harakatlarni amalga oshirishi mumkin.

Bu misolda `animal_sound()` funksiyasi `Dog` va `Cat` obyektlarini qabul qiladi va ularning har biri o‘ziga xos `speak()` metodini bajaradi.

```
class Dog:
    def speak(self):
        return "Dog barks."

class Cat:
    def speak(self):
        return "Cat meows."

def animal_sound(animal):
    print(animal.speak())

# Turli obyektlar uchun bitta funksiya
dog = Dog()
cat = Cat()

animal_sound(dog)   # Output: Dog barks.
animal_sound(cat)   # Output: Cat meows.
```

Polymorphism va Universal Funksiyalar

Polymorphismdan foydalanib bir xil nomli funksiya har xil turdagi obyektlarni qabul qilishi va ularning turiga qarab turli xatti-harakatlarni bajarishi mumkin. Bu funksiyalarning bir xil ko‘rinishda ishlashi, lekin turli obyektlar bilan moslashishi mumkinligini ko‘rsatadi.

Bu yerda `area()` metodi ikkita har xil classlarda turlicha yozilgan, lekin ular bir xil nomga ega va bir xil funksiya orqali chaqirilmoqda. Har bir obyektning `area()` metodini chaqirish orqali u o‘ziga xos xatti-harakatni bajaradi.

```python
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius

shapes = [Rectangle(4, 5), Circle(3)]

for shape in shapes:
    print(f"Area: {shape.area()}")
```

Polymorphismning Afzalliklari

Kodni soddalashtirish: Turli obyektlar bir xil interfeys orqali ishlash imkoniyatiga ega bo‘ladi, bu esa kodni soddalashtiradi va tuzilmani yaxshilaydi.

Moslashuvchanlik: Dastur kodini qayta yozmasdan yangi classlar qo‘shib, dastur imkoniyatlarini kengaytirish imkonini beradi.

Kengaytirish imkoniyati: Polymorphism yordamida yangi obyektlar qo‘shish va ularning metodlarini xuddi avvalgi obyektlar kabi ishlatish mumkin.

Vazifa

Mashq 1:class `Shape` yarating. Unda `area()` metodi bo‘lsin. `Rectangle`, `Square`, va `Circle` classlarini bu classdan meros qilib oling va ularda `area()` metodini implement qiling.

Mashq 2: `Person` classidan foydalanib, `Employee` va `Manager` classlarini yarating. Ular turli xil `work()` metodiga ega bo‘lsin va polymorphism orqali bir xil funksiyada ularning xatti-harakatlarini amalga oshiring.