Python OOP Concepts & Examples

September 21, 2023

In Python we are going to look at • Class, Object, Inheritance and Constructor with Example.
• Polymorphism in python with Example

Sure, I can provide you with explanations and examples for classes, objects, inheritance, constructors, and polymorphism in Python.

Class and Object:

A class is a blueprint for creating objects, and objects are instances of classes. Classes define the structure and behavior of objects. Here's an example:

python Copy code

```python
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        print(f"{self.name} is barking!")

# Creating objects (instances) of the Dog class
dog1 = Dog("Buddy", "Golden Retriever")
dog2 = Dog("Charlie", "Poodle")

# Accessing object attributes and methods
print(f"{dog1.name} is a {dog1.breed}.")
```

```
dog2.bark()
```

In this example, we define a Dog class with attributes (`name` and `breed`) and a method (`bark`). We then create two Dog objects (`dog1` and `dog2`) and access their attributes and methods.

Inheritance:

Inheritance is a way to create a new class by inheriting properties and methods from an existing class. The existing class is called the parent or base class, and the new class is called the child or derived class. Here's an example:

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return f"{self.name} says Woof!"

class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow!"

# Creating objects of child classes
dog = Dog("Buddy")
cat = Cat("Whiskers")

# Calling the speak method of each object
print(dog.speak())
print(cat.speak())
```

In this example, we have a base class `Animal` with a `speak` method. The Dog and `Cat` classes inherit from `Animal` and override the `speak` method to provide their own implementation.

Constructor:

In Python, a constructor is a special method named `__init__` that gets called automatically when an object is created from a class. It is used to initialize object attributes. We've already seen constructors in the examples above.

Polymorphism:

Polymorphism allows objects of different classes to be treated as objects of a common base class. It enables flexibility in your code by allowing you to use objects interchangeably. Here's an example using the `speak` method from the previous example:

python Copy code

```python
def animal_speak(animal):
    print(animal.speak())

# Creating objects of child classes
dog = Dog("Buddy")
cat = Cat("Whiskers")

# Using the same function to make different objects speak
animal_speak(dog)  # Calls Dog's speak method
animal_speak(cat)  # Calls Cat's speak method
```

In this example, the `animal_speak` function takes an argument of type `Animal` (the base class), and we can pass objects of both Dog and `Cat` classes to it. This demonstrates polymorphism, as the same function can work with objects of different classes as long as they share a common interface (in this case, the `speak` method).
These are the basic concepts of classes, objects, inheritance, constructors, and polymorphism in Python. They are fundamental to object-oriented programming in the language.

Continue this conversation