OOP in Python (Part 1)

Object-Oriented Programming (OOP) in Python is a way of writing code that focuses on organizing data (attributes) and actions (methods/functions) together into objects. These objects are instances of classes, which act as blueprints defining how the objects should be structured and what they can do.

Python'da Nesne Yönelimli Programlama (OOP)

Python'da Nesne Yönelimli Programlama (OOP) verileri (nitelikler) ve eylemleri (metod/fonksiyonlar) nesneler içinde bir araya getirmeye odaklanan bir kod yazma yöntemidir. Bu nesneler, sınıfların örnekleri olup, nesnelerin nasıl yapılandırılması gerektiğini ve ne yapabileceklerini tanımlayan şablonlar olarak işlev görür.

In [ ]:

```python
# Define a Class Called Cat
class Cat:
    # This is the constructor method that initializes the object with some
    def __init__(self, name, age):
        self.name = name  # Defines the attribute for the cat's name
        self.age = age    # Defines the attribute for the cat's age

    # Method to make the cat meow
    def meow(self):
        print(f"{self.name} says: Meow Meow!")

    # Method to display the cat's age
    def display_age(self):
        print(f"{self.name} is {self.age} years old.")

# Create an instance (object) of the Cat class
Cat1 = Cat("Whiskers", 4)
# Use the methods of the Cat object
Cat1.meow()          # The cat meows
Cat1.display_age()   # Displays the cat's age
```

```
Whiskers says: Meow Meow!
Whiskers is 4 years old.
```

We define a class Cat with two methods (init, meow) and two attributes (name, age). The init method is a special constructor method that is automatically called when a new object is created. It initializes the object's attributes with the provided values. We create an instance of the Cat class called cat1, passing the name 'Whiskers' and age 3 as arguments. Then, we use the meow method to make cat1 meow and the display_age method to show its age.

In [ ]:

```python
# Define a class called 'Car'
class Car:
    # This is the constructor method that initializes the object with some
    def __init__(self, make, model, year):
        self.make = make  # Attribute to store the car's make
        self.model = model  # Attribute to store the car's model
        self.year = year  # Attribute to store the car's manufacturing yea
        self.speed = 0  # Attribute to store the car's current speed

    # Method to start the car
    def start(self):
        print("Car started.")

    # Method to accelerate the car
    def accelerate(self, mph):
        self.speed += mph  # Increasing the speed by the provided miles pe
        print(f"Accelerating. Current speed: {self.speed} mph")

    # Method to brake the car
    def brake(self, mph):
        self.speed -= mph  # Decreasing the speed by the provided miles pe
        print(f"Braking. Current speed: {self.speed} mph")

    # Method to display car information
    def display_info(self):
        print(f"Make: {self.make}, Model: {self.model}, Year: {self.year},

# Create instances (objects) of the car class
car1 = Car("Toyota", "Camry", 2022)  # Creating an instance for a Toyota C
car2 = Car("Tesla", "Model 3", 2023)  # Creating an instance for a Tesla M
```

In [ ]:

```python
# Use the methods of the objects
car1.start()  # Starting the car represented by car1
car1.accelerate(30)  # Accelerating the car by 30 mph
car1.accelerate(20)  # Accelerating the car by additional 20 mph
car1.brake(5)  # Applying brakes to reduce the speed by 5 mph

car2.start()  # Starting the car represented by car2
car2.accelerate(30)  # Accelerating the car by 30 mph
car2.accelerate(80)  # Accelerating the car by additional 80 mph
car2.brake(5)  # Applying brakes to reduce the speed by 5 mph

# Display information about the cars
car1.display_info()  # Displaying information about car1
car2.display_info()  # Displaying information about car2
```

```
Car started.
Accelerating. Current speed: 30 mph
Accelerating. Current speed: 50 mph
Braking. Current speed: 45 mph
Car started.
Accelerating. Current speed: 30 mph
Accelerating. Current speed: 110 mph
Braking. Current speed: 105 mph
Make: Toyota, Model: Camry, Year: 2022, Speed: 45 mph
Make: Tesla, Model: Model 3, Year: 2023, Speed: 105 mph
```

we define a class Car with four methods (init, start, accelerate, and brake) and an attribute speed. The init method is a special constructor method that is called when a new object is created. It initializes the object's attributes with the provided values.

We create two instances of the Car class, car1 and car2, and then use the methods to perform actions like starting, accelerating, and braking the cars. The display_info method shows the details of each car, including the current speed.

That's the essence of OOP in Python. Classes are used to create objects, and those objects can have attributes and methods that define their behavior and properties. This approach allows for code reusability, modularity, and a more organized structure.

Onur_Gumus