1. What is OOP:

Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOPs Concepts:

- ✓ Class
- ✓ Objects
- ✓ Data Abstraction
- ✓ Encapsulation
- ✓ Inheritance
- ✓ Polymorphism
- ✓ Dynamic Binding
- ✓ Message Passing

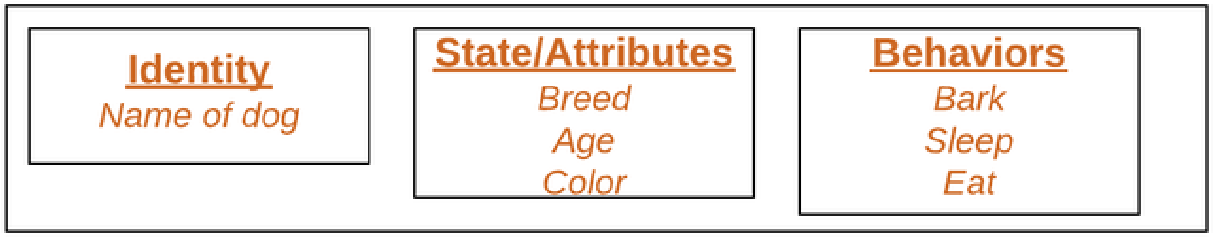
2. What is a Class?

A **Class** is a blueprint or template for creating objects in OOP. It defines the structure (attributes) and behaviors (methods) that the objects created from the class will have. For example, a class Car might have attributes like color, model, and methods like drive() or stop().

3. What is Object:

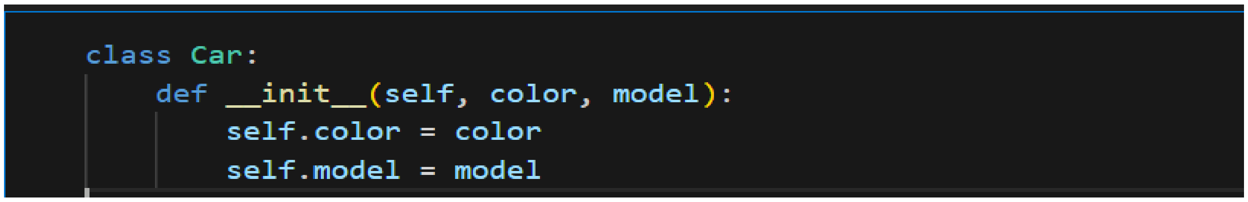
It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behavior. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

For example "Dog" is a real-life Object, which has some characteristics like color, Breed, Bark, Sleep, and Eats.



4. What is a Constructor?

A **Constructor** is a special method in a class that gets called automatically when an object is created. In Python, the constructor is the __init__() method. It is used to initialize the object's attributes and set up the object's initial state.

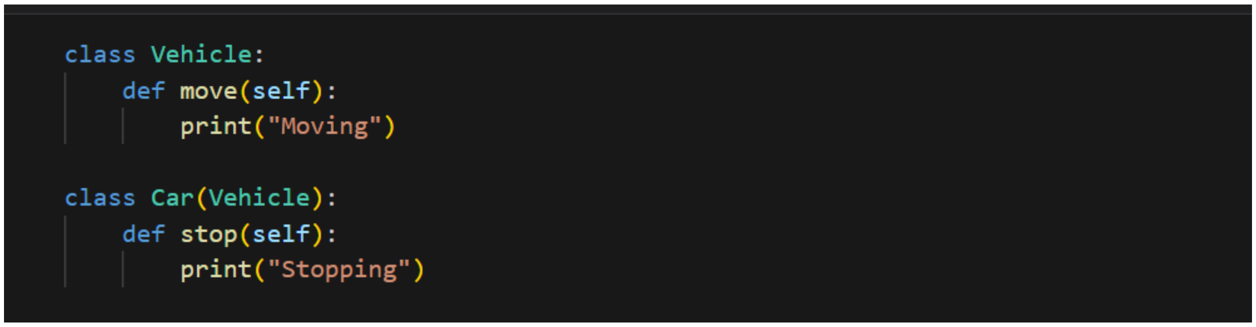


```
class Car:
    def __init__(self, color, model):
        self.color = color
        self.model = model
```

5. What is Inheritance?

Inheritance is an OOP concept where one class (child or derived class) inherits the properties and methods of another class (parent or base class). This allows code reusability and the ability to extend or modify behaviors of the parent class without changing it.

Example:

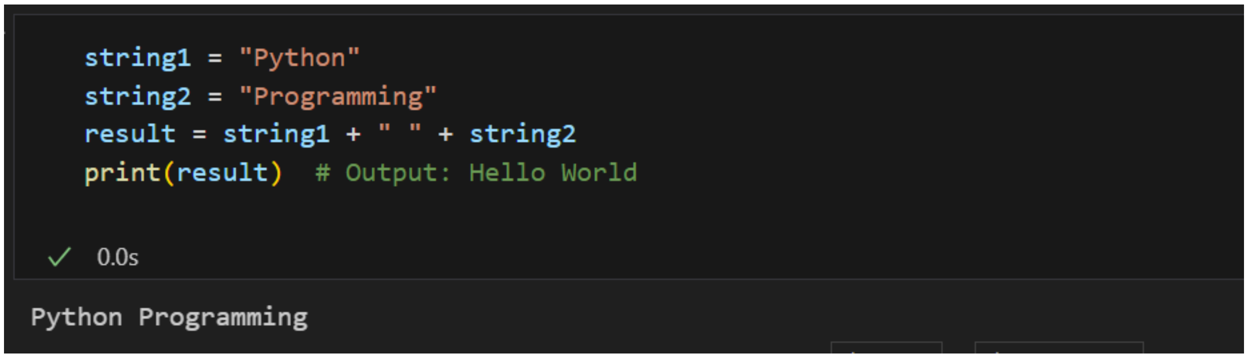


```
class Vehicle:
    def move(self):
        print("Moving")

class Car(Vehicle):
    def stop(self):
        print("Stopping")
```

6. What is Concatenation?

Concatenation refers to combining two or more strings into a single string. In Python, concatenation is often done using the + operator.



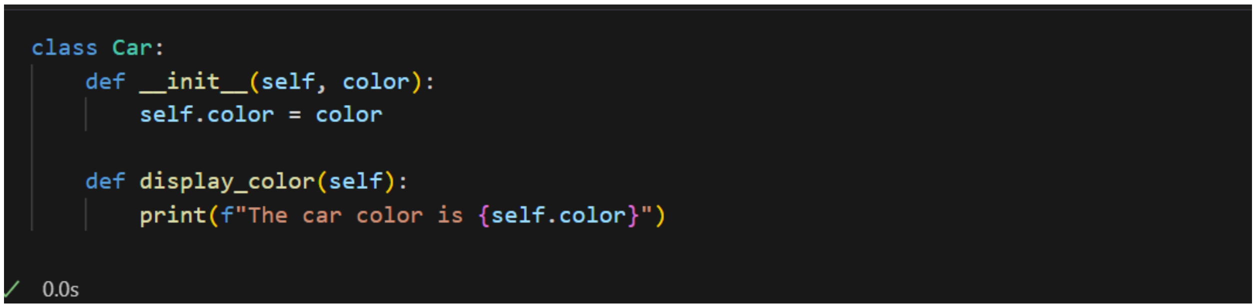
```
string1 = "Python"
string2 = "Programming"
result = string1 + " " + string2
print(result)  # Output: Hello World
```

✓ 0.0s

```
Python Programming
```

7. What is the self keyword in a Class?

The self keyword in Python refers to the instance of the class. It allows access to the attributes and methods of the class inside its methods. Every method in a class must have self as its first parameter to reference the object that calls the method.



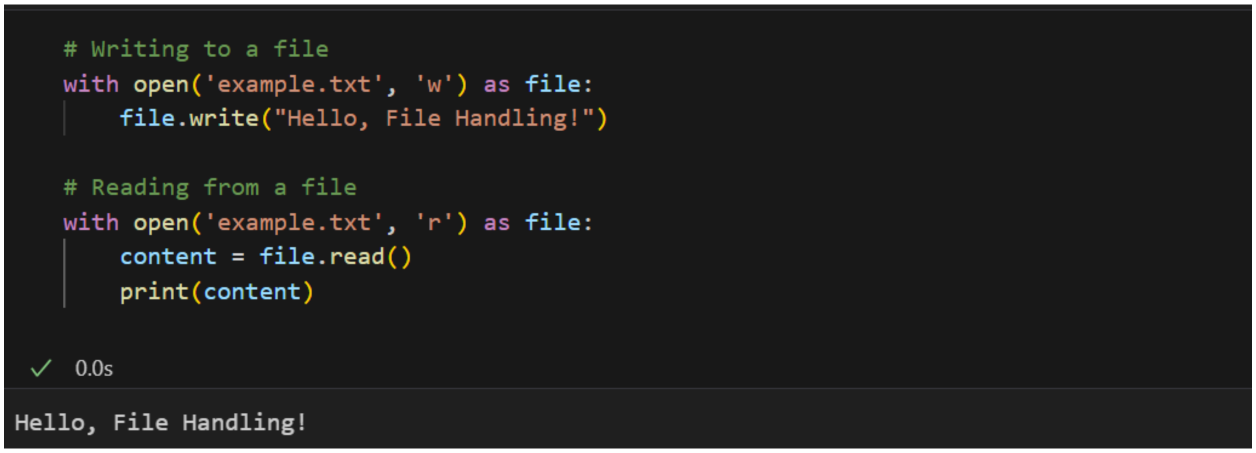
```
class Car:
    def __init__(self, color):
        self.color = color

    def display_color(self):
        print(f"The car color is {self.color}")
```

✓ 0.0s

8. What is File Handling?

File Handling refers to the process of opening, reading, writing, and closing files in a program. In Python, file handling is done using built-in functions like open(), read(), write(), and close().



```
# Writing to a file
with open('example.txt', 'w') as file:
    file.write("Hello, File Handling!")

# Reading from a file
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

✓ 0.0s

```
Hello, File Handling!
```

9. What is NumPy?

NumPy (Numerical Python) is a powerful library in Python for numerical computing. It provides support for multi-dimensional arrays (like matrices) and mathematical operations on arrays. It is commonly used for scientific and engineering computations, data manipulation, and machine learning.

NumPy (short for Numerical Python) is a foundational Python library used for efficient numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a vast collection of high-level mathematical functions to perform operations on these arrays. NumPy is widely used for scientific and technical computing due to its ability to handle large datasets, perform linear algebra, Fourier transforms, and generate random numbers. Its array structure allows for efficient memory management and fast computation, making it essential in fields like data science, machine learning, and engineering.