**Python Classes and Objects:**

In Python, classes and objects are fundamental concepts of object-oriented programming (OOP).And Classes are blueprints for creating objects, while objects are instances of classes. They allow us to model real-world entities with attributes (data) and methods (functions).

## Defining Classes in Python:

A class defined using the class keyword, followed by the class name. In class, you can define multiple attributes and methods.

**Syntax:**

class ClassName:

# Attributes

attribute1 = value1

attribute2 = value2

# Methods

def method1(self, parameters):

# Method body

pass

def method2(self, parameters):

# Method body

Pass

## **Object Definition**

In Python, an object is an instance of a class, which is a user-defined data structure holding its own data members and member functions. Objects are created by instantiating a class and can access the attributes and methods defined within that class.

**Syntax:**

obj = ClassName()

print(obj.attribute)

**Components of an Object:**

1. State:
   * The state of an object is represented by its attributes, which define its properties or characteristics. These attributes hold the current values associated with the object.
2. Behavior:
   * The behavior of an object is defined by its methods, which represent the actions or operations that the object can perform. Methods define how an object interacts with itself and other objects.
3. Identity:
   * Each object has a unique identity, which distinguishes it from other objects. This identity enables objects to interact with each other and maintain their individuality.

**Declaring Class Objects (Instantiation):**

When an object of a class is created, it is said to be instantiated. This process involves creating a new instance of the class, which inherits the attributes and methods defined in the class blueprint. However, each instance has its own unique state, determined by the specific values assigned to its attributes.

**Explanation:**

* A class serves as a blueprint for creating objects in Python.
* Objects are created by calling the class name followed by parentheses, similar to calling a function, which invokes the class's constructor method (\_\_init\_\_).
* The obj variable holds a reference to the newly created object.
* Attributes are the variables that belong to a class and define its properties.
* Attributes are accessed using the dot (.) operator, where obj.attribute accesses the value of the attribute attribute belonging to the object obj.

**Key Points on Python Classes:**

* Classes are created using the class keyword.
* Attributes are public and can be accessed using the dot (.) operator.
* The class encapsulates both data (attributes) and behavior (methods).
* Objects instantiated from a class inherit its attributes and methods.

classes and objects in Python allow for the creation of user-defined data structures with associated behaviors, enabling modular and reusable code design.

Objects in Python provide a powerful mechanism for modeling real-world entities and implementing their behavior in code. By instantiating classes, developers can create multiple objects, each with its own unique state and behavior, enabling modular and flexible software design.

**Example Code:**

class Car:

# Class attribute

wheels = 4

# Constructor method

def \_\_init\_\_(self, make, model, year):

# Instance attributes

self.make = make

self.model = model

self.year = year

# Method to display car information

def display\_info(self):

print(f"{self.year} {self.make} {self.model}, {self.wheels} wheels")

# Creating objects of the Car class

car1 = Car("Toyota", "Camry", 2020)

car2 = Car("Ford", "Mustang", 2018)

# Accessing object attributes and methods

car1.display\_info()

car2.display\_info()

**Explanation of Code:**

* We define a class named Car, which represents a car entity.
* Inside the class, we have a class attribute wheels set to 4, representing the number of wheels of a car.
* The \_\_init\_\_ method serves as the constructor, which initializes instance attributes (make, model, year) when a new object is created.
* The display\_info method displays information about the car, including its make, model, year, and number of wheels.
* We create two objects (car1 and car2) of the Car class with different attributes.
* We then call the display\_info method on each object to print the car information.

**Conclusion:**

Classes and objects are essential concepts in Python that facilitate object-oriented programming. Classes act as blueprints for creating objects, allowing us to model real-world entities with attributes and methods. By defining classes and creating objects, we can organize our code more effectively and create reusable components.