Classes and Objects in Python

Python is an object-oriented programming language that offers classes, which are a potent tool for writing reusable code. To describe objects with shared characteristics and behaviours, classes are utilised.

Classes in Python:

In Python, a class is a user-defined data type that contains both the data itself and the methods that may be used to manipulate it. In a sense, classes serve as a template to create objects. They provide the characteristics and operations that the objects will employ.

Creating Classes in Python

In Python, a class can be created by using the keyword class, followed by the class name. The syntax to create a class is given below.

Syntax

```
class ClassName:
    #statement_suite
```

Code:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def greet(self):
        print("Hello, my name is " + self.name)
```

Objects in Python:

An object is a particular instance of a class with unique characteristics and functions. After a class has been established, you may make objects based on it. By using the class constructor, you may create an object of a class in Python. The object's attributes are initialised in the constructor, which is a special procedure with the name __init__.

Syntax:

```
# Declare an object of a class
object_name = Class_Name(arguments)
```

Example:

Code:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def greet(self):
        print("Hello, my name is " + self.name)

# Create a new instance of the Person class and assign it to the variable person1
person1 = Person("Ayan", 25)
person1.greet()
```

The self-parameter

The self-parameter refers to the current instance of the class and accesses the class variables. We can use anything instead of self, but it must be the first parameter of any function which belongs to the class.

Class and Instance Variables

All instances of a class exchange class variables. They function independently of any class methods and may be accessed through the use of the class name. Here's an illustration:

Code:

```
class Person:
    count = 0   # This is a class variable

    def __init__(self, name, age):
        self.name = name    # This is an instance variable
        self.age = age
        Person.count += 1   # Accessing the class variable using the name of the class
person1 = Person("Ayan", 25)
person2 = Person("Bobby", 30)
print(Person.count)
```

Code:

```
class Person:
    def __init__(self, name, age):
        self.name = name    # This is an instance variable
        self.age = age
person1 = Person("Ayan", 25)
person2 = Person("Bobby", 30)
print(person1.name)
print(person2.age)
```