
CodroidHub Summer Training

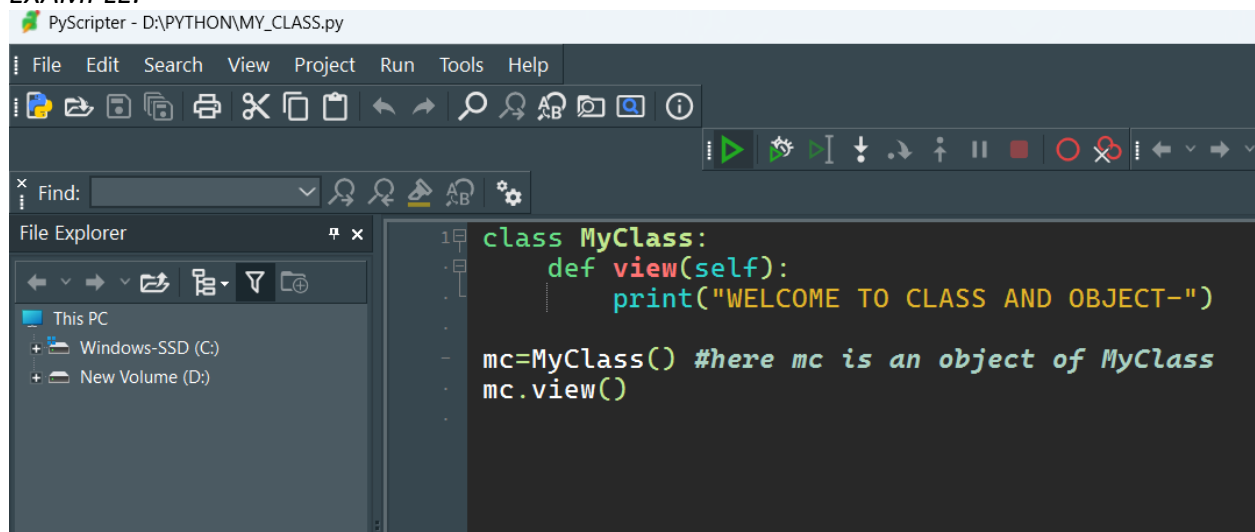
Title:-

CLASS AND OBJECT in Python

CLASS AND OBJECT IN PYTHON

A **class** is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it to maintain its state. Class instances can also have methods (defined by their class) for modifying their state.

EXAMPLE:



The screenshot shows the PyScripter IDE interface. The title bar indicates the file path is D:\PYTHON\MY_CLASS.py. The menu bar includes File, Edit, Search, View, Project, Run, Tools, and Help. The toolbar contains various icons for file operations, editing, and running code. The File Explorer on the left shows the project structure with 'This PC', 'Windows-SSD (C:)', and 'New Volume (D:)'. The main editor window displays the following Python code:

```
1 class MyClass:
2     def view(self):
3         print("WELCOME TO CLASS AND OBJECT-")
4
5 mc=MyClass() #here mc is an object of MyClass
6 mc.view()
```

OUTPUT:

```
Python Interpreter

>>>
*** Remote Interpreter Reinitialized ***
WELCOME TO CLASS AND OBJECT-
>>>
```

Call Stack Variables Watches Breakpoints Output Messages Python Interpreter

Object of Python Class

In [Python programming](#) an Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values. It's not an idea anymore, it's an actual dog, like a dog of breed pug who's seven years old. You can have many dogs to create many different instances, but without the class as a guide, you would be lost, not knowing what information is required.

An object consists of:

- **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
- **Behaviour:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

EXAMPLE OF CONSTRUCTOR:

```
PyScripter - D:\PYTHON\CONSTRUCTOR.py

File Edit Search View Project Run Tools Help

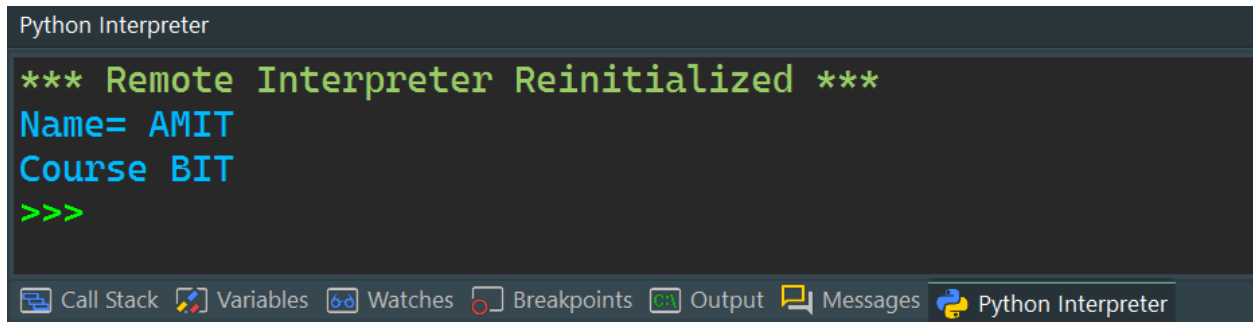
class Constructor:
    def __init__(self, name, course): #CONSTRUCTOR
        self.name = name
        self.course = course
    def view(self):
        print("Name=", self.name)
        print("Course", self.course)

co = Constructor("AMIT", "BIT") #here mc is an object of MyClass
co.view()
```

OUTPUT:

```
Python Interpreter

*** Remote Interpreter Reinitialized ***
Name= AMIT
Course BIT
>>>
```

A screenshot of a Python Interpreter window. The window has a dark background. The title bar at the top says "Python Interpreter". The main area displays the following text: "*** Remote Interpreter Reinitialized ***" in green, "Name= AMIT" in blue, "Course BIT" in blue, and ">>>" in green. At the bottom, there is a toolbar with icons and labels for "Call Stack", "Variables", "Watches", "Breakpoints", "Output", "Messages", and "Python Interpreter".