## **What are classes and objects in Python?**

A python is an object-oriented programming language. Unlike procedure-oriented programming, where the main emphasis is on functions, object-oriented programming stress on objects.

Object is simply a collection of data (variables) and methods (functions) that act on those data. And, the class is a blueprint for the object.

We can think of the class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

As, many houses can be made from a description, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called **instantiation**.

## **Defining a Class in Python**

Like function definitions begin with the keyword [def](https://www.programiz.com/python-programming/keyword-list#def), in Python, we define a class using the keyword [class](https://www.programiz.com/python-programming/keyword-list#class).

The first string is called docstring and has a brief description about the class. Although not mandatory, this is recommended.

Here is a simple class definition.

1. class MyNewClass:
2. '''This is a docstring. I have created a new class'''
3. pass

A class creates a new local [namespace](https://www.programiz.com/python-programming/namespace) where all its attributes are defined. Attributes may be data or functions.

There are also special attributes in it that begin with double underscores (\_\_). For example, \_\_doc\_\_ gives us the docstring of that class.

As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

class MyClass:

"This is my second class"

a = 10

def func(self):

print('Hello')

# Output: 10

print(MyClass.a)

# Output: <function MyClass.func at 0x0000000003079BF8>

print(MyClass.func)

# Output: 'This is my second class'

print(MyClass.\_\_doc\_\_)

When you run the program, the output will be:

10

<function 0x7feaa932eae8="" at="" myclass.func="">

This is my second class

## **Creating an Object in Python**

We saw that the class object could be used to access different attributes.

It can also be used to create new object instances (instantiation) of that class. The procedure to create an object is similar to a [function](https://www.programiz.com/python-programming/function) call.

1. >>> ob = MyClass()

This will create a new instance object named ob. We can access attributes of objects using the object name prefix.

Attributes may be data or method. Method of an object are corresponding functions of that class. Any function object that is a class attribute defines a method for objects of that class.

This means to say since MyClass.func is a function object (attribute of class), ob.func will be a method object.

# **Accessor and Mutator in Python**

Since it's a good idea to keep internal data of an object **private,** we often need methods in the class interface to allow the user of objects to modify or **access** the internally stored data, in a controlled way.

A method defined within a class can either be an **Accessor** or a **Mutator** method. An accessor method is a function that returns a copy of an internal variable or computed value. A common practice is to name these with the word get. A **mutator** method is a function that modifies the value of an internal data variable in some way. The simplest form of mutator function is one that sets a variable directly to a new value. A common practice is to name these with the word set.

#### 

#### **Example**

**class** MyClass():

**def** \_\_init\_\_(self):

self.\_\_my\_attr = 3

**def** set\_my\_attr(self,val):

self.\_\_my\_attr = val

**def** get\_my\_attr(self):

**return** self.\_\_my\_attr

obj1 = MyClass()

**print** (obj1.get\_my\_attr())

obj1.set\_my\_attr(7)

**print** (obj1.get\_my\_attr())

In the example above,

The methods obj1.get\_my\_attr() is **Accessor** methods since it doesn’t alter the object an in any sense but only pulls the relevant information. But obj1.set\_my\_attr(7) is a **mutator** method since it effectively changes the object to a new one.