Python OOPs: Class, Object, Inheritance and Constructor with Example

Till now we have been learning procedural programming now we will switch into object-oriented programming. For understanding Object-oriented programming you must understand these simple concepts.

* Class
* Object
* Inheritance
* Constructors

## Class

A class can be thought of as blueprint about what you are going to develop. Classes provide you with a way to gather the same type of function and attributes together. For example, every dog has the following attributes.

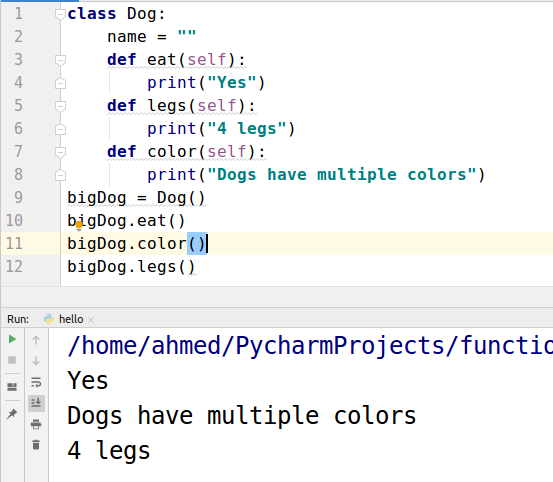
* Eat
* Sleep
* Bark
* Colour

A class will help us gathers all these attributes in one place.

Let’s have a look at how to define Class.

|  |
| --- |
| class Dog:  name = ""  def eat(self):  print("Yes")  def legs(self):  print("4 legs")  def color(self):  print("Dogs have multiple colors") bigDog = Dog() bigDog.eat() bigDog.color() bigDog.legs() |

The following will be the output.



## Objects

|  |
| --- |
| bigDog = Dog() |

The above syntax is used to create an object. By creating an object it means that you are creating a variable that can be used to access all the methods or variables that are defined in the class.

## Use of ‘Self’ in Class functions.

The first argument of the function in the class is always “self”. This variable is used to access the variable that is defined in the class. You can use any keyword other than “self” but “self” word is conventional. Every Python programmer uses “self”.

## Inheritance

This term is self-explanatory. In inheritance, a derived class is formed using the base class.

The derived class has all the features of the base class. For example, As we know there are many types of Dogs. We don’t need to create a separate class for every dog type. As we created a ‘Dog’ class above that can work as a base class. We can derive a new class of specialized dog type i.e “bulldog”.

|  |
| --- |
| class Dog:  name = ""   def eat(self):  print("Yes")   def legs(self):  print("4 legs")   def color(self):  print("Dogs have multiple colors")   class bulldog(Dog):  name = "bulldog"   def work(self):  print("I can't do work but I can play with you")   mydog = bulldog() print(mydog.name) mydog.legs() mydog.work() |



We didn’t define any function named ‘legs’ inside ‘bulldog’ class. It automatically inherited it from the base class named ‘dog’.

## Constructor

The constructor is a special type of functions that are used to auto define some features whenever an object is created. For example, you own a room and you want that whenever you enter that room lights are turned on automatically. Just like that whenever an object has created the constructor automatically initializes some variables.

|  |
| --- |
| class Dog:   def \_\_init\_\_(self, name):  self.name = name   def printName(self):  print("my dog name is ", self.name)  myDog = Dog("bulldog") myDog.printName() |

The constructor functions are defined by special name \_\_init\_\_. We are passing the name as we create the object.

