Python Classes

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 for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

Some points on Python class:

Classes are created by keyword class.

Attributes are the variables that belong to a class.

Attributes are always public and can be accessed using the dot (.) operator. Eg.: Myclass.Myattribute

Class Definition Syntax:

class ClassName: # Statement-1 . . . # Statement-N

Example:

```
# demonstrate defining
# a class

class Dog:
    pass
```

Class Objects

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.

Behavior: 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.

Declaring an object –

```
# Python3 program to
# demonstrate instantiating
# a class


class Dog:

    # A simple class
    # attribute
    attr1 = "mammal"
    attr2 = "dog"

    # A sample method
    def fun(self):
        print("I'm a", self.attr1)
        print("I'm a", self.attr2)

# Driver code
# Object instantiation
Rodger = Dog()

# Accessing class attributes
# and method through objects
print(Rodger.attr1)
Rodger.fun()
print(Rodger)
```

```
mammal
I'm a mammal
I'm a dog
<__main__.Dog object at 0x00000247D8CAD128>
```

init method

The **init** method is similar to constructors in C++ and Java. Constructors are used to initializing the object’s state. Like methods, a constructor also contains a collection of statements(i.e. instructions) that are executed at the time of Object creation. It runs as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.

```
# A Sample class with init method
class Person:

    # init method or constructor
    def __init__(self, name):
        self.name = name

    # Sample Method
    def say_hi(self):
        print('Hello, my name is', self.name)

p = Person(' Girraj ')
p.say_hi()
```

```
Hello, my name is  Girraj
```

Class and Instance Variables

Instance variables are for data, unique to each instance and class variables are for attributes and methods shared by all instances of the class. Instance variables are variables whose value is assigned inside a constructor or method with self whereas class variables are variables whose value is assigned in the class.

```
# Python3 program to show that the variables with a value
# assigned in the class declaration, are class variables and
# variables inside methods and constructors are instance
# variables.

# Class for Dog
class Dog:

    # Class Variable
    animal = 'dog'

    # The init method or constructor
    def __init__(self, breed, color):

        # Instance Variable
        self.breed = breed
        self.color = color

# Objects of Dog class
Rodger = Dog("Pug", "brown")
Buzo = Dog("Bulldog", "black")
a = Dog("Boxcer ","black & brown")

print('Rodger details:')
print('Rodger is a', Rodger.animal)
print('Breed: ', Rodger.breed)
print('Color: ', Rodger.color)

print('\nBuzo details:')
print('Buzo is a', Buzo.animal)
print('Breed: ', Buzo.breed)
print('Color: ', Buzo.color)

print('\na details:')
print('a is a', a.animal)
print('Breed: ', a.breed)
```

```python
print('Color: ', a.color)

# Class variables can be accessed using class
# name also
print("\nAccessing class variable using class name")
print(Dog.animal)
```

```
Rodger details:
Rodger is a dog
Breed:  Pug
Color:  brown

Buzo details:
Buzo is a dog
Breed:  Bulldog
Color:  black

a details:
a is a dog
Breed:  Boxcer
Color:  black & brown

Accessing class variable using class name
dog
```

```python
# Python3 program to show that we can create
# instance variables inside methods

# Class for Dog
class Dog:

    # Class Variable
    animal = 'dog'

    # The init method or constructor
    def __init__(self, breed):

        # Instance Variable
        self.breed = breed

    # Adds an instance variable
    def setColor(self, color):
        self.color = color

    # Retrieves instance variable
    def getColor(self):
        return self.color

# Driver Code
Rodger = Dog("pug")
```

```
Rodger.setColor(" Brown ")
print(Rodger.getColor())
```

```
 Brown
```

Python Inheritance

Inheritance enables us to define a class that takes all the functionality from a parent class and allows us to add more. In this tutorial, you will learn to use inheritance in Python.

Inheritance in Python

Inheritance is a powerful feature in object oriented programming.

It refers to defining a new class with little or no modification to an existing class. The new class is called derived (or child) class and the one from which it inherits is called the base (or parent) class.

Python Inheritance Syntax

class BaseClass:

```
      Body of base class
```

class DerivedClass(BaseClass):

```
      Body of derived class
```

Derived class inherits features from the base class where new features can be added to it. This results in re-usability of code.

Example:

```
class Polygon:
    def __init__(self, no_of_sides):
        self.n = no_of_sides
        self.sides = [0 for i in range(no_of_sides)]

    def inputSides(self):
        self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i
in range(self.n)]

    def dispSides(self):
        for i in range(self.n):
            print("Side",i+1,"is",self.sides[i])

class Triangle(Polygon):
    def __init__(self):
        Polygon.__init__(self,3)

    def findArea(self):
        a, b, c = self.sides
        # calculate the semi-perimeter
        s = (a + b + c) / 2
        area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
        print('The area of the triangle is %0.2f' %area)
```

```
t = Triangle()
```

```
t.inputSides()
```

```
Enter side 1 : 5
Enter side 2 : 6
Enter side 3 : 9
```

```
t.dispSides()
```

```
Side 1 is 5.0
Side 2 is 6.0
Side 3 is 9.0
```

```
t.findArea()
```

```
The area of the triangle is 14.14
```