Python-4

April 20, 2019

1 OOP

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, class is a blueprint for the object.

We can think of 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

1.1 Class

Like function definitions begin with the keyword def, in Python, we define a class using the keyword 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.

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

1.2 Instance

A specific object from defined class.

1.3 Instanciation

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 call.

1.4 Data Members

1.4.1 Methods

Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

1.4.2 Attributes

class variables
instance variables

1.5 Inheritance

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.

class BaseClass: Body of base class class DerivedClass(BaseClass): Body of derived class

1.6 Operator overloading

Python operators work for built-in classes. But same operator behaves differently with different types. For example, the + operator will, perform arithmetic addition on two numbers, merge two lists and concatenate two strings.

This feature in Python, that allows same operator to have different meaning according to the context is called operator overloading.

To overload the + sign, we will need to implement **add**() function in the class. With great power comes great responsibility. We can do whatever we like, inside this function. But it is sensible to return a Point object of the coordinate sum.

1.7 Encapsulation

Using OOP in Python, we can restrict access to methods and variables. This prevent data from direct modification which is called encapsulation. In Python, we denote private attribute using underscore as prefix i.e single " _ " or double " __".

1.8 Constructors

Class functions that begins with double underscore (__) are called special functions as they have special meaning.

Of one particular interest is the **init**() function. This special function gets called whenever a new object of that class is instantiated.

This type of function is also called constructors in Object Oriented Programming (OOP). We normally use it to initialize all the variables.