Chapter-1

Topic: Classes and Objects

1. Introduction

2. Classes and objects

    Defining classes

    Creating classes

    Data abstraction and hiding through classes

3. Class method and self argument

4. The _ _ init _ _ () method (the class constructor)

5. Class variables and object variables

6. The _ _ Del _ _ () method

7. Other special methods

8. Public and private data members

9. Private methods

10. Calling a class method from another class method

11. Built-in functions to check, get, set and delete class attributes

12. Built-in class attributes

13. Garbage collections (destroying objects)

14. Class methods

15. Static methods

1. Introduction

- **Attribute:** Data items that makes up an instance
- **Class:** A user-defined prototype for an object that defines a set of attributes (class variables and instance variables) and methods that are accessed via dot notation
- **Class Variable:** A variable defined within a class that is shared by all instances of a class
- **Data Member:** A variable (class variable or instance variable) defined within the class that holds data associated with a class and its objects
- **Instance:** Object of a class
- **Instance Variable:** A variable that is defined inside a class method and belongs only to the current instance of the class
- **Instantiation:** The process of creating an instance of a class
- **Method:** Function defined in a class definition and is invoked on instances of that class
- **Namespace:** A mapping from names to objects in such a way that there is no relation between names in different namespaces
- **Object:** Instance and objects are used interchangeably
- **Object Oriented Language:** A language that supports object oriented features like classes, inheritance, operator overloading.
- **Object Oriented Programming:** A style of programming in which data and operations that manipulate it are together encapsulated inside single entity called class

2. Classes and Objects

- Classes and Objects are the two main aspects of object oriented programming
- Class is the basic building block in Python
- A class creates a new type and object is an instance (or variable) of the class
- Classes provides a blueprint or a template using which objects are created
- In python, everything is an object or instance of some class

For Example:

- All integers variables that we define in our program are actually instances of class INT
- All string variables are objects of class string
  I, e, we can find out the type of any object using the type () function

2.1 Defining Classes

Python has a very simple syntax of defining a class.

Syntax:

```
class class_name:
   <Statement-1>
   <Statement-2>
   ......
   ......
   ......
   <Statement-N>
```

Hint: A class can be defined in a function or with an if statement.

From the above syntax we can see that class definition, it starts with a keyword "class" followed by "class_name" and a colon (:).

The statement in the definition can be any of these following i,e,

  - Sequential instructions
  - Decision control statements,
  - Loop statements
  - Function definitions
- Variables defined in a class are called class variables
- Functions defined inside a class are called class methods
- Class methods and class variables are called class members
- The class members can be accessed through class objects
- Class methods have access to all the data contained in the instance of the object
- Class definitions can appear anywhere in a program, but they are usually written near the beginning of the program, i,e, after the import statement, note that when a class definition is entered a new namespace is created and used as the local scope. Therefore all assignments to local variables go into this new namespace.

Hint: A class creates a new local namespace where all its attributes (data and functions) are defined.

2.2 Creating Objects

- Once a class is defined the next job is to create an object (or instance) of that class.
- The object can then access class variables and class methods using the dot operator (.).

The syntax to create an object is given as follows

```
object_name = class_name
```

Program to access class variables using class object

```
class ABC:

   var = 10      # class variable

obj = ABC()

print(obj.var)           # class variable is accessed using class object
```

Output

```
10
```

Program Explanation:

- In the program we defined a class ABC which has a variable var having a value of 10.
- The object of the class is created and used to access the class variable using dot operator.

2.3 Data Abstraction and Hiding through Classes

- Data Abstraction refers to the process by which data and function are defined in such a way that only essential details are provided to the outside world and the implementation details are hidden.
- In Python, classes provide methods to the outside world to provide the functionality of the object or to manipulate the objects' data.
- Any entity outside the world does not know about the implementation details of the class or that method.

- Data Encapsulation, also called data hiding, organizes the data and methods into a structure that prevents data access by any function (or method) that is not specified in the class. This ensures the integrity of the data contained in the object.

Encapsulation defines different access levels for data variables and member functions of the class, these access levels specifies the access rights for example:

- Any data or function with access level "public" can be accessed by any function belonging to any class, this is the lowest level of data protection.
- Any data or function with access level "private" can be accessed only by the class in which it is declared, this is the highest level of data protection.

In python, private variable are prefixed with a double underscore (_ _),

For example: _ _ var is a private variable of the class

3. Class Method and "SELF" Argument

Class is a template / blueprint for real world entities

Example: Mobile_Phone is an object, This Mobile_Phone object has some Properties and Behavior.

- Properties are: color, cost, battery life
- Behaviors are: make-calls, play-game, watch-videos

So basically class combine these Properties and Behavior under one templates known as class.

Example-1: Consider the program which has one class variable and one class method.

From the below example first identify which is variable, which is method, how to access variable, how to access method, operator used to invoke method

# Program to access class members using the class object

```
class ABC():

   var = 10

   def display(self):

      print("In Class Method")
```

```
obj = ABC()

print(obj.var)

obj.display()
```

Output

```
10

In Class Method
```

- Class methods (or functions defined in the class), Class methods must have the first argument named as "self" and this is the first argument that is added to the beginning of the parameter list.
- For the "self" you will not pass a value to “self “parameter when you call the method
- Python provides its value automatically
- The “self” argument refers no arguments, it should be defined to accept the "self"

Thus, the class methods uses "self", they require an object or instance of the class to be used. For this reason they are often referred to as "instance methods".

So you have observed that the class method accepts no values but still has "self" as an argument and both the class members are accessed through the object of the class.

Example-2 Create a class with the object called Mobile_Phone, declare three variables and some values to it. Define a methods called make_call, play_game and create an object or instance of the class Mobile_Phone and then invoke the methods make_call and play_game using the created object or instance.

```
class Mobile_Phone():

      var1 = 100

   var2 = 200

   var3 = 300

# make_call is a method name and "self" this self method belongs to the instance
which invokes it
```

```
    def make_call(self):

        print("Making a phone call")

# play_game is a method name and "self" this self method belongs to the instance
which invokes it

    def play_game(self):

        print("Playing a game")

# p1 is an object or created the instance of a Class Mobile_Phone

p1 = Mobile_Phone ()

# accessing the variables

print(p1.var1)

print(p1.var2)

print(p1.var3)

# accessing the Method or if you want to invoke the defined methods (make_call,
play_game) from the object (p1)

p1.make_call()

p1.play_game()
```

Output:

```
100

200

300

Making a phone call

Playing a game
```

Hint:

P1 is a name of an instance, . Is an operator, Make_call is a name of a method

Key Points to remember:

- The statement inside the class definition must be properly indented.
- A class that has no other statements should have a pass statement at least.
- Class methods or functions that begins with double underscore (_ _) are special functions with a predefined and a special meaning.