Unit III

CHAPTER 10

PYTHON CLASSES AND OBJECTS

Learning Objectives

After the completion of this chapter, the student is able to

- Understand the fundamental concepts of Object Oriented Programming like: Classes, Objects, Constructor and Destructor.
- Gain the knowledge of creating classes and objects in Python.
- Create classes with Constructors.
- Write complex programs in Python using classes.

10.1 Introduction

Python is an Object Oriented Programming language. Classes and Objects are the key features of Object Oriented Programming. Theoretical concepts of classes and objects are very similar to that of C++. But, creation and implementation of classes and objects is very simple in Python compared to C++.

Class is the main building block in Python. Object is a collection of data and function that act on those data. Class is a template for the object. According to the concept of Object Oriented Programming, objects are also called as instances of a class. In Python, everything is an object. For example, all integer variables that we use in our program is an object of class int. Similarly all string variables are also object of class string.

10.2 Defining classes

In Python, a class is defined by using the keyword class. Every class has a unique name followed by a colon (:).

Syntax:

```
class class_name:
    statement_1
    statement_2
    ……..
    ……..
    statement_n
```

Where, statement in a class definition may be a variable declaration, decision control, loop or even a function definition. Variables defined inside a class are called as "Class Variable" and functions are called as "Methods". Class variable and methods are together known as members of the class. The class members should be accessed through objects or instance of class. A class can be defined anywhere in a Python program.

Example: Program to define a class

```
class Sample:
        x, y = 10, 20          #  class variables
```

In the above code, name of the class is Sample and it has two variables x and y having the initial value 10 and 20 respectively. To access the values defined inside the class, you need an object or instance of the class.

10.3 Creating Objects

Once a class is created, next you should create an object or instance of that class. The process of creating object is called as "Class Instantiation".

Syntax:

Object_name = class_name( )

Note that the class instantiation uses function notation ie. class_name with

10.4 Accessing Class Members

Any class member ie. class variable or method (function) can be accessed by using object with a dot (.) operator.

Syntax:

Object_name . class_member

Example : Program to define a class and access its member variables

```
class Sample:
        x, y = 10, 20          #class variables
S=Sample( )          # class instantiation
print("Value of x = ", S.x)
print("Value of y = ", S.y)
print("Value of x and y = ", S.x+S.y)
```

Output :

```
Value of x =  10
Value of y =  20
Value of x and y =  30
```

In the above code, the name of the class is Sample. Inside the class, we have assigned the variables **x** and **y** with initial value **10** and **20** respectively. These two variables are called as class variables or member variables of the class. In class instantiation process, we have created an object **S** to access the members of the class. The first two print statements simply print the value of class variable **x** and **y** and the last print statement add the two values and print the result.

10.5 Class Methods

Python class function or Method is very similar to ordinary function with a small difference that, the class method must have the first argument named as **self**. No need to pass a value for this argument when we call the method. Python provides its value automatically. Even if a method takes no arguments, it should be defined with the first argument called **self**. If a method is defined to accept only one argument it will take it as two arguments ie. self and the defined argument.

When you declare class variable within class, methods must be prefixed by the class name and dot operator.

Note

The statements defined inside the class must be properly indented.

Example 10.2: Program to find total and average marks using class

```
class Student:
    mark1, mark2, mark3 = 45, 91, 71          #class variable

    def process(self):                        #class method
        sum = Student.mark1 + Student.mark2 + Student.mark3
        avg = sum/3
        print("Total Marks = ", sum)
        print("Average Marks = ", avg)
        return

S=Student()
S.process()
```

In the above program, after defining the class, an object S is created. The statement S.process( ), calls the function to get the required output.

Note that, we have declared three variables mark1, mark2 and mark3 with the values 45, 91, 71 respectively. We have defined a method named **process** with **self** argument, which means, we are not going to pass any value to that method. First, the process method adds the values of the class variables, stores the result in the variable sum, finds the average and displays the result.

Thus the above code will show the following output.

Output

Total Marks = 207

Average Marks = 69.0

Example : program to check and print if the given number is odd or even using class

```
class Odd_Even:
    def check(self, num):
        if num%2==0:
        print(num," is Even number")
    else:
        print(num," is Odd number")
n=Odd_Even()
x = int(input("Enter a value: "))
n.check(x)
```

When you execute this program, Python accepts the value entered by the user and passes it to the method check through object.

Output 1

```
Enter a value: 4
4  is Even number
```

Output 2

```
Enter a value: 5
5  is Odd number
```

10.6 Constructor and Destructor in Python

Constructor is the special function that is automatically executed when an object of a class is created. In Python, there is a special function called **"init"** which act as a Constructor. It must begin and end with double underscore. This function will act as an ordinary function; but only difference is, it is executed automatically when the object is created. This constructor function can be defined with or without arguments. This method is used to initialize the class variables.

General format of __init__ method (Constructor function)

```
def __init__(self, [args ……..]):
        <statements>
```

Example : Program to illustrate Constructor

```
class Sample:
        def __init__(self, num):
                print("Constructor of class Sample...")
                self.num=num
                print("The value is :", num)
S=Sample(10)
```

The above class "Sample", has only a constructor with one argument named as num. When the constructor gets executed, first the print statement, prints the "Constructor of class Sample….", then, the passing value to the constructor is assigned to self.num and finally it prints the value passed along with the given string.

The above constructor gets executed automatically, when an object **S** is created with actual parameter 10. Thus, the Python display the following output.

Constructor of class Sample...

The value is : 10

Class variable defined within constructor keep count of number of objects created with the class.

Example : Program to illustrate class variable to keep count of number of objects created.

```
class Sample:
        num=0
        def __init__(self, var):
                Sample.num+=1
                self.var=var
                print("The object value is = ", var)
                print("The count of object created = ", Sample.num)

S1=Sample(15)
S2=Sample(35)
S3=Sample(45)
```

In the above program, class variable **num** is shared by all three objects of the class Sample. It is initialized to zero and each time an object is created, the num is incremented by 1. Since, the variable shared by all objects, change made to num by one object is reflected in other objects as well. Thus the above program produces the output given below.

Output

```
The object value is          =    15
The count of object created  =    1
The object value is          =    35
The count of object created  =    2
The object value is          =    45
The count of object created  =    3
```

***Note:* class variable is similar to static type in C++**

Destructor is also a special method gets executed automatically when an object exit from the scope. It is just opposite to constructor. In Python, __del__( ) method is used as destructor.

Example : Program to illustrate about the __del__( ) method

```
class Sample:
        num=0
        def __init__(self, var):
                Sample.num+=1
                self.var=var
                print("The object value is = ", var)
                print("The value of class variable is= ", Sample.num)
        def __del__(self):
                Sample.num-=1
                print("Object with value %d is exit from the scope"%self.var)
S1=Sample(15)
S2=Sample(35)
S3=Sample(45)
```

10.7 Public and Private Data Members

The variables which are defined inside the class is public by default. These variables can be accessed anywhere in the program using dot operator.

A variable prefixed with double underscore becomes private in nature. These variables can be accessed only within the class.

Example : Program to illustrate private and public variables

```
class Sample:
        def __init__(self, n1, n2):
                self.n1=n1
                self.__n2=n2
        def display(self):
                print("Class variable 1 = ", self.n1)
                print("Class variable 2 = ", self.__n2)
S=Sample(12, 14)
S.display()
print("Value 1 = ", S.n1)
print("Value 2 = ", S.__n2)
```

In the above program, there are two class variables n1 and n2 are declared. The variable n1 is a public variable and n2 is a private variable. The display( ) member method is defined to show the values passed to these two variables.

The print statements defined within class will successfully display the values of n1 and n2, even though the class variable n2 is private. Because, in this case, n2 is called by a method defined inside the class. But, when we try to access the value of n2 from outside the class Python throws an error. Because, private variable cannot be accessed from outside the class.

Output

```
Class variable 1 =  12
Class variable 2 =  14
Value 1 =  12

Traceback (most recent call last):
        File "D:/Python/Class-Test-04.py", line 12, in <module>
                print("Value 2 = ", S.__n2)
AttributeError: 'Sample' object has no attribute '__n2'
```

10.8 Sample Programs to illustrate classes and objects

Program 1: Write a program to calculate area and circumference of a circle

```
class Circle:
        pi=3.14
def __init__(self,radius):
        self.radius=radius
def area(self):
        return Circle.pi*(self.radius**2)
def circumference(self):
        return 2*Circle.pi*self.radius
r=int(input("Enter Radius: "))
C=Circle(r)
print("The Area =",C.area())
print("The Circumference =", C.circumference())
```

Output:

```
Enter Radius: 5
The Area = 78.5
The Circumference = 31.400000000000002
```

Program 2: write a menu driven program that keeps record of books available in you school library

```
class Library:
        def __init__(self):
                self.bookname=""
                self.author=""

        def getdata(self):
                self.bookname = input("Enter Name of the Book: ")
                self.author = input("Enter Author of the Book: ")

        def display(self):
                print("Name of the Book: ",self.bookname)
                print("Author of the Book: ",self.author)
                print("\n")
```

```
book=[] #empty list
ch = 'y'
while(ch=='y'):
        print("1. Add New Book \n 2.Display Books")
        resp = int(input("Enter your choice : "))

if(resp==1):
        L=Library()
        L.getdata()
        book.append(L)
elif(resp==2):
        for x in book:
                x.display()

else:
        print("Invalid input....")

ch = input("Do you want continue....")
```

Output:

1. Add New Book

2.Display Books

Enter your choice : 1

Enter Name of the Book: Programming in C++

Enter Author of the Book: K. Kannan

Do you want continue....y

1. Add New Book

2.Display Books

Enter your choice : 1

Enter Name of the Book: Learn Python

Enter Author of the Book: V.G.Ramakrishnan

Do you want continue....y

1. Add New Book

2.Display Books

Enter your choice : 1

Enter Name of the Book: Advanced Python

Enter Author of the Book: Dr. Vidhya

Do you want continue....y

1. Add New Book

2.Display Books

Enter your choice : 1

Enter Name of the Book: Working with OpenOffice

Enter Author of the Book: N.V.Gowrisankar

Do you want continue....y

1. Add New Book

2.Display Books

Enter your choice : 1

Enter Name of the Book: Data Structure

Enter Author of the Book: K.Lenin

Do you want continue....y

1. Add New Book

2.Display Books

Enter your choice : 1

Enter Name of the Book: An Introduction to Database System

Enter Author of the Book: R.Sreenivasan

Do you want continue....y

1. Add New Book

2.Display Books

```
Enter your choice : 2
Name of the Book:  Programming in C++
Author of the Book:  K. Kannan

Name of the Book:  Learn Python
Author of the Book:  V.G.Ramakrishnan

Name of the Book:  Advanced Python
Author of the Book:  Dr. Vidhya

Name of the Book:  Working with OpenOffice
Author of the Book:  N.V.Gowrisankar

Name of the Book:  Data Structure
Author of the Book:  K.Lenin

Name of the Book:  An Introduction to Database System
Author of the Book:  R.Sreenivasan

Do you want continue....n
```

Program 3: Write a program to accept a string and print the number of uppercase, lowercase, vowels, consonants and spaces in the given string

```
class String:
    def __init__(self):
        self.uppercase=0
        self.lowercase=0
        self.vowels=0
        self.consonants=0
        self.spaces=0
        self.string=""
    def getstr(self):
        self.string=str(input("Enter a String: "))
```

```
def count_upper(self):
        for ch in self.string:
                if (ch.isupper()):
                        self.uppercase+=1

def count_lower(self):
        for ch in self.string:
                if (ch.islower()):
                        self.lowercase+=1

def count_vowels(self):
        for ch in self.string:
                if (ch in ('A', 'a', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')):
                        self.vowels+=1

def count_consonants(self):
        v=('A', 'a', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')
        for ch in self.string:
                if ch not in v and ch.isalpha():
                        self.consonants+=1
def count_space(self):
        for ch in self.string:
                if (ch==" "):
                        self.spaces+=1
def execute(self):
        self.count_upper()
        self.count_lower()
        self.count_vowels()
        self.count_consonants()
        self.count_space()
def display(self):
        print("The given string contains...")
        print("%d Uppercase letters"%self.uppercase)
        print("%d Lowercase letters"%self.lowercase)
        print("%d Vowels"%self.vowels)
        print("%d Consonants"%self.consonants)
        print("%d Spaces"%self.spaces)
```

```
S = String()
S.getstr()
S.execute()
S.display()
```

Output

```
Enter a String:Welcome To Learn Computer Science
The given string contains...
5 Uppercase letters
24 Lowercase letters
12 Vowels
17 Consonants
3 Spaces
```

Program 4: Write a program to store product and its cost price. Display all the available products and prompt to enter quantity of all the products. Finally generate a bill which displays the total amount to be paid

```
class MyStore:
    __prod_code=[]
    __prod_name=[]
    __cost_price=[]
    __prod_quant=[]

def getdata(self):
    self.p = int(input("Enter no. of products you need to store: "))
    for x in range(self.p):
        self.__prod_code.append(int(input("Enter Product Code: ")))
        self.__prod_name.append(str(input("Enter Product Name: ")))
        self.__cost_price.append(int(input("Enter Cost price: ")))
```

```
def display(self):
        print("Stock in Stores")
        print("-------------------------------------------------------")
        print("Product Code \t Product Name \t Cost Price")
        print("-------------------------------------------------------")
        for x in range(self.p):
            print(self.__prod_code[x], "\t\t", self.__prod_name[x], "\t\t", self.__cost_
                                                                        price[x])
        print("-------------------------------------------------------")

def print_bill(self):
        total_price = 0
        for x in range(self.p):
            q=int(input("Enter the quantify for the product code %d : "%self.__
                                                                    prod_code[x]))
self.__prod_quant.append(q)
total_price = total_price +self.__cost_price[x]*self.__prod_quant[x]

        print("          Invoice Receipt                ")
        print("-------------------------------------------------------------------------")
        print("Product Code\t Product Name\t Cost Price\t Quantity \t Total Amount")
        print("-------------------------------------------------------------------------")
        for x in range(self.p):
            print(self.__prod_code[x], "\t\t", self.__prod_name[x], "\t\t",
                self.__cost_price[x], "\t\t", self.__prod_quant[x], "\t\t",
                self.__prod_quant[x]*self.__cost_price[x])
        print("-------------------------------------------------------------------------")
        print("                                    Total Amount = ", total_price)
```

```
S=MyStore()
S.getdata()
S.display()
S.print_bill()
```

Output:

```
Enter no. of products you need to store: 5
Enter Product Code: 101
Enter Product Name: Product-A
Enter Cost price: 25
Enter Product Code: 201
Enter Product Name: Product-B
Enter Cost price: 35
Enter Product Code: 301
Enter Product Name: Product-C
Enter Cost price: 35
Enter Product Code: 401
Enter Product Name: Product-D
Enter Cost price: 50
Enter Product Code: 501
Enter Product Name: Product-E
Enter Cost price: 120
```

Stock in Stores

Product Code	Product Name	Cost Price
101	Product-A	25
201	Product-B	35
301	Product-C	35
401	Product-D	50
501	Product-E	120

```
Enter the quantify for the product code 101 : 10
Enter the quantify for the product code 201 : 15
Enter the quantify for the product code 301 : 10
Enter the quantify for the product code 401 : 20
Enter the quantify for the product code 501 : 10

                    Invoice Receipt
```

Product Code	Product Name	Cost Price	Quantity	Total Amount
101	Product-A	25	10	250
201	Product-B	35	15	525
301	Product-C	35	10	350
401	Product-D	50	20	1000
501	Product-E	120	10	1200

```
                                        Total Amount = 3325
```

Points to remember

- Python is an Object Oriented Programming language.
- Classes and Objects are the key features of Object Oriented Programming.
- In Python, a class is defined by using the keyword class.
- Variables defined inside a class is called as “Class Variable” and function are called as “Methods”.
- The process of creating object is called as “Class Instantiation”.
- Constructor is the special function that is automatically executed when an object of a class is created.
- In Python, there is a special function called “init” is used as Constructor.
- Destructor is also a special method gets execution automatically when an object exits from the scope.
- In Python, __del__( ) method is used as destructor.
- A variable prefixed with double underscore is becomes private in nature.

Hands on Experience

1. Write a program using class to store name and marks of students in list and print total marks.

2. Write a program using class to accept three sides of a triangle and print its area.

3. Write a menu driven program to read, display, add and subtract two distances.

6MVJH

Evaluation

Part - I

Choose the best answer **(1 Marks)**

1. Which of the following are the key features of an Object Oriented Programming language?
   (a) Constructor and Classes (b) Constructor and Object
   (c) Classes and Objects (d) Constructor and Destructor
2. Functions defined inside a class:
   (a) Functions (b) Module
   (c) Methods (d) section
3. Class members are accessed through which operator?
   (a) & (b) .
   (c) # (d) %
4. Which of the following method is automatically executed when an object is created?
   (a) __object__( ) (b) __del__( )
   (c) __func__( ) (d) __init__( )
5. A private class variable is prefixed with
   (a) __ (b) &&
   (c) ## (d) **
6. Which of the following method is used as destructor?
   (a) __init__( ) (b) __dest__( )
   (c) __rem__( ) (d) __del__( )

7. Which of the following class declaration is correct?

(a) class class_name (b) class class_name<>

(c) class class_name: (d) class class_name[ ]

8. Which of the following is the output of the following program?

```
class Student:
        def __init__(self, name):
                self.name=name
                print (self.name)
S=Student("Tamil")
```

(a) Error (b) Tamil

(c) name (d) self

9. Which of the following is the private class variable?

(a) __num (b) ##num

(c) $$num (d) &&num

10. The process of creating an object is called as:

(a) Constructor (b) Destructor

(c) Initialize (d) Instantiation

Part -II

Answer the following questions (2 Marks)

1. What is class?
2. What is instantiation?
3. What is the output of the following program?

```
class Sample:
        __num=10
        def disp(self):
                print(self.__num)
S=Sample()
S.disp()
print(S.__num)
```

4. How will you create constructor in Python?
5. What is the purpose of Destructor?

Part -III

Answer the following questions (3 Marks)

1. What are class members? How do you define it?
2. Write a class with two private class variables and print the sum using a method.
3. Find the error in the following program to get the given output?

```
class Fruits:
        def __init__(self, f1, f2):
                self.f1=f1
                self.f2=f2
        def display(self):
                print("Fruit 1 = %s, Fruit 2 = %s" %(self.f1, self.f2))
F = Fruits ('Apple', 'Mango')
del F.display
F.display()
```

Output

```
        Fruit 1 = Apple,   Fruit 2 = Mango
```

4. What is the output of the following program?

```
class Greeting:
        def __init__(self, name):
                self.__name = name
        def display(self):
                print("Good Morning ", self.__name)
obj=Greeting('Bindu Madhavan')
obj.display()
```

5. How to define constructor and destructor in Python?

Part -IV

Answer the following questions (5 Marks)

1. Write a menu driven program to add or delete stationary items. You should use dictionary to store items and the brand.

References

1. *https://docs.python.org/3/tutorial/index.html*
2. *https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list*
3. *Python programming using problem solving approach – Reema Thareja – Oxford University press.*
4. *Python Crash Course – Eric Matthes – No starch press, San Francisco.*