Python Assignment No. 5

Q1. How to define a class. Write a program to create a simple class and print message "Welcome to Object Oriented Programming" and print the address of the instance of class.

Ans. As discussed above, class is another name for type in Python. A class may contain data in the form of fields. Fields are also called attributes and coded in the form of procedures known as methods. Finally, a programmer has to create an object of its own class, where the object representsan entity which can be easily identified. For example, person, vehicle, fan, book etc. represent realobjects. Each object has a unique identity, state and behaviour. The state of the object is also calledproperty or attribute. For example, a circular object has a data field radius which is a property that characterises a circle. The syntax to define a class in Python is given as follows:

```
Class Class_Name:
    Initializer
    attributes
    methods()
    Statement(s)
```

Program:

```
class MyFirstProgram:
    print(' Welcome to Object-oriented Programming')
C=MyFirstProgram()      #Instance of class.
print(C)
```

Output

```
Welcome to Object-oriented Programming
<__main__.MyFirstProgram object at 0x028B6C90>
```

Explanation Name of the class is MyFirstProgram. The instance, i.e. 'C' of the class is created. Inside the class, the print statement is used to display the welcome message. Additionally, the lastprint statement is used to display the address of the computer's memory where the object 'C' is stored.

Q2. Write a program to access attributes of a class for defining area of rectangle.

Ans.

```
class Rectangle:
    length=0;          #Attribute  length
    breadth=0;         #Attribute  breadth
R1 = Rectangle ()      #Instance of a class
print(R1.length)       #Access attribute length
print(R1.breadth)      #Access attribute breadth
```

Output

```
0
0
```

Q3. Write a program to calculate area of rectangle by assigning the value to the attributes of rectangle, i.e. length and breadth.

Ans.

```
class Rectangle:
    length=0;          #Attribute  length
    breadth=0;         #Attribute  breadth
R1 = Rectangle ()      #Instance of a class
print('Initial values of Attribute')
print('Length   = ',R1.length)      #Access attribute length
print('Breadth  = ',R1.breadth)     #Access attribute breadth
print('Area of Rectangle = ',R1.length * R1.breadth )
R1.length   =  20   #Assign value to attribute length
R1.breadth  =  30   #Assign  value to attribute breadth
print('After reassigning the value of attributes')
print('Length = ',R1.length )
print('Breadth = ',R1.breadth )
print('Area of Rectangle is ',R1.length * R1.breadth)
```

Output

```
Initial values of Attribute
Length   =  0
Breadth  =  0
Area of Rectangle =  0
After reassigning the value of attributes
Length =  20
Breadth =  30
Area of Rectangle is  600
```

Q4. Explain self parameter.

Ans.

The Self-parameter

To add methods to an existing class, the first parameter for each method should be self. There isonly one difference between class methods and ordinary functions. The self-parameter is used in the implementation of the method, but it is not used when the method is called. Therefore, the self-parameter references the object itself. Program illustrates the self-parameter and addition of methods to an existing class.

```
class MethodDemo:
    def Display_Message(self):
        print('Welcome to Python Programming')
ob1 = MethodDemo()      #Instance of a class
ob1.Display_Message()   #Calling Method
```

Output

```
Welcome to Python Programming
```

Explanation In the above program, the Display_Message()method takes no parameters butstill the method has the self-parameter in the function definition. Therefore, the self-parameter refers to the current object itself. Finally, the method is called and the message is displayed.

Q5. Write a program to create a class named circle. Pass the parameter radius to the method named Calc_Area() and calculate area of circle.

Ans.

```
import math
class Circle:
    def Calc_Area(self,radius):
        print('radius = ',radius)
        return math.pi*radius**2
ob1 = Circle()
print('Area of circle is ',ob1.Calc_Area(5))
```

Output

```
radius =  5
Area of circle is  78.53981633974483
```

Explanation The class with name Circle is created as shown above. The extra parameter radius ispassed to a method defined inside the class Calc_Area(). The instance ob1 of a class is created and used to call the method of the existing class. Even though the method Calc_Area() contains two parameters, viz. self and radius, only one parameter should be passed, viz. the radius of the circle while calling the method.

Q6. Write a program to display attributes in a given class.

Ans.

```
class DisplayDemo:
    Name = '';          #Attribute
    Age = ' ';          #Attribute
    def read(self):
        Name=input('Enter Name of student: ')
        print('Name = ',Name)
        Age=input('Enter Age of the Student:')
        print('Age = ',Age)
D1 = DisplayDemo()
D1.read()

#Display attributes using dir() on the interactive mode
>>>(dir( DisplayDemo )
['Age', 'Name', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__',
'__init__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__
reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__
str__', '__subclasshook__', '__weakref__', 'read']
```

Q7. Explain __init__ method(constructor). Write a simple program using init method.

Ans.

There are many inbuilt methods in Python. Each method has its own significance. The importanceof the_init method is explained ahead.

The_init method is known as an initialiser. It is a special method that is used to initialise theinstance variable of an object. This method runs as soon as an object of a class is instantiated. The syntax of adding_init method to a class is given as follows:

```
class Class_Name:
    def __init__(self):    #__init__ method
        ……
        ……
```

init needs to be preceded and followed by two underscores. Also_init method must have self as the first argument. As self refers to the object itself, it refers to the object that invokes the method. The self-parameter within the_init_ method automatically sets the reference forthe object just created. The_init method can also have positional and/or keyword arguments.

```
class Circle:
    def __init__(self,pi):
        self.pi = pi
    def calc_area(self,radius):
        return self.pi*radius**2
C1=Circle(3.14)
print(' The area of Circle is ',C1.calc_area(5))
```

Output

```
The area of Circle is 78.5
```

Explanation In the above program we have created a class named Circle. The class contains two different methods, viz. one is__init_ method and another calc_area() to calculate the area of a circle. Notice that in the above program we do not explicitly call the_init method. We have created an instance of the class Circle, i.e. C1. While creating the instance of the class we have passed the arguments following the class name to initialise the instance variable of an object.

Q8. Write a program to pass object as a parameter to a method.

Ans.

```
class Test:
    a = 0
    b = 0
    def __init__(self, x , y):
        self.a = x
        self.b = y
    def equals(self, obj):
        if(obj.a == self.a and obj.b == self.b):
            return True
        else:
            return False
Obj1 = Test(10,20)
Obj2 = Test(10,20)
Obj3 = Test(12,90)
print(' Obj1 == Obj2 ',Obj1.equals(Obj2))
print(' Obj1 == Obj3 ',Obj1.equals(Obj3))
```

Output

```
Obj1 == Obj2  True
Obj1 == Obj3  False
```

Q9. Explain __del__ (destructor) method. Write a program to illustrate the use of __del__ method.

Ans.

Like other object-oriented programming languages, Python also has a destructor. The method
__del__ denotes the destructor and the syntax to define destructor is.

```
def __del__(self)
    block
```

Python invokes the destructor method when the instance is about to be destroyed. It is invokedone per instance. The self refers to the instance on which the __del__() method is invoked. In other words, Python manages garbage collection of objects by reference counting. This functionis executed only if all the references to an instance object have been removed.

```
class Destructor_Demo:
    def __init__(self):      #Constructor
        print('Welcome')
    def __del__(self):       #Destructor
        print('Destructor Executed Successfully')
Ob1=Destructor_Demo()        #Instantiation
Ob2 = Ob1
Ob3 = Ob1    #Object Ob2 and Ob3 refers to same object
print(' Id of Ob1 = ',id(Ob1))
print(' Id of Ob2 = ',id(Ob2))
print(' Id of Ob3 = ',id(Ob3))
```

```
del Ob2     #Remove reference Ob2
del Ob1     #Remove reference Ob1
del Ob3     #Remove reference Ob3
```

Output

```
Welcome
 Id of Ob1 =  47364272
 Id of Ob2 =  47364272
 Id of Ob3 =  47364272
Destructor Executed Successfully
```

Explanation In the above example, we have used constructor **__init__** and destructor **__del__** functions. Initially we have instantiated the object **Ob1** and then assigned aliases **Ob2** and **Ob3** to it. The inbuilt function **id()** is used to confirm that all the three aliases reference to the same object. Finally, all the aliases are removed using the del statement.

Q10. Explain Inheritance in details. Write a simple program on inheritance.

Ans. Inheritance is one of the most useful and essential characteristics of object-oriented programming.The existing classes are the main components of inheritance. New classes are created from the existing ones. The properties of the existing classes are simply extended to the new classes. Anew class created using an existing one is called a derived class or subclass and the existingclass is called a base class or super class. An example of inheritance is shown in Figure 10.2. The relationship between base and derived class is known as kind of relationship. A programmer can define new attributes, i.e. (member variables) and functions in a derived class.

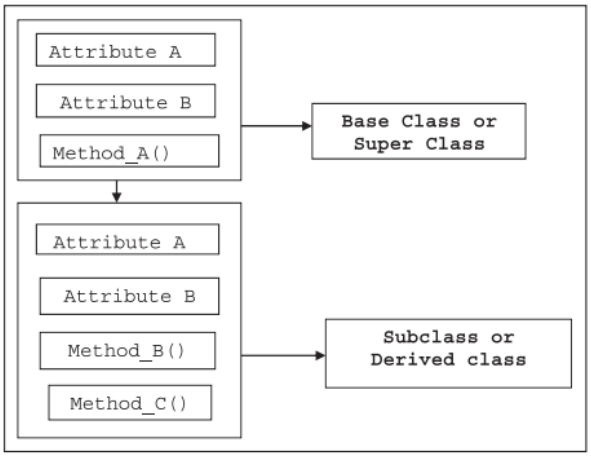

Figure 10.2 Simple example of inheritance

```
class A:
    print('Hello I am in Base Class')
class B(A):
    print('Wow!! Great ! I am Derived class')
ob2 = A()  #Instance of class B
```

Output

```
Hello I am in Base Class
Wow!! Great! I am Derived class
```

Q11. Explain types of inheritance in details with suitable block diagram.

Ans.

Theprocess of inheritance can be simple or complex according to the following:

1. *Number of base classes:* A programmer can use one or more base classes to derive a singleclass.
2. *Nested derivation:* The derived class can be used as the base class and a new class can bederived from it. This is possible at any extent.

Inheritance can be classified as: (i) single inheritance, (ii) multilevel inheritance and (iii) multipleinheritance. Each of these has been described in detail as follows:

(i) ***Single inheritance:*** Only one base class is used for deriving a new class. The derived class is notused as the base class.

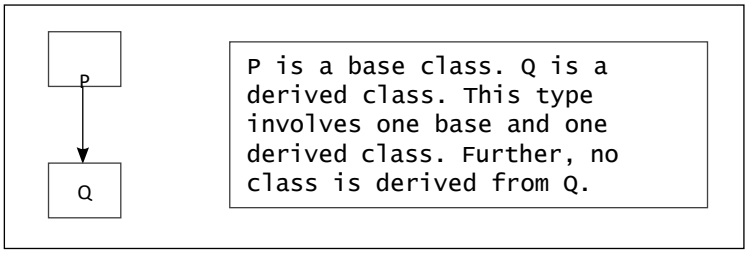

Figure 10.3 Single inheritance

(ii) ***Multilevel inheritance:*** When a class is derived from another derived class, the derived classacts as the base class. This is known as multilevel inheritance.

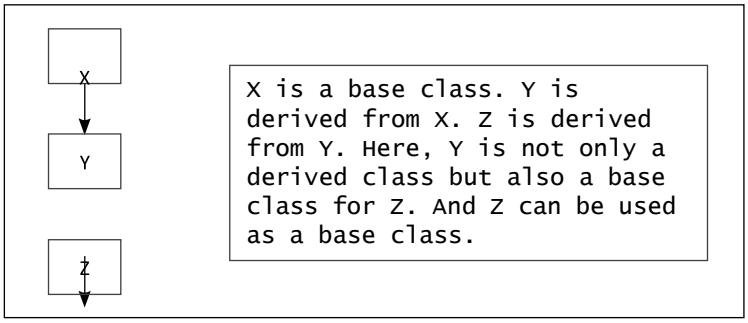

Figure 10.4 Multilevel inheritance

(iii) ***Multiple inheritance:*** When two or more base classes are used for deriving a new class, it iscalled multiple Inheritance.

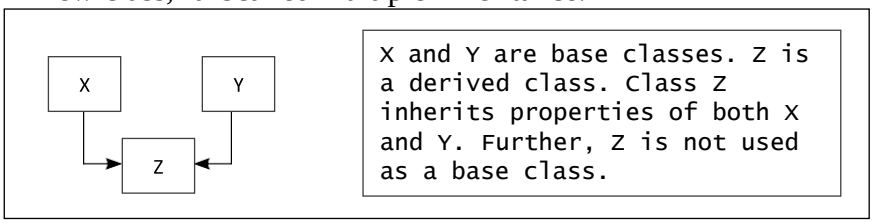

Figure 10.5 Multiple inheritance

Q12. Explain Tuple. Describe inbuilt function for tuple along with example.

Ans.

Tuples work exactly like lists. A tuple contains a sequence of items of many types. The elements oftuples are fixed. Once a tuple has been created, we cannot add or delete elements, or even shuffle their order. Hence, tuples are immutable. This means that once created, they cannot be changed. Since tuples are immutable, their length is also fixed. A new tuple must be created to grow or shrink an earlier one.

Inbuilt Functions for Tuples

Python provides various inbuilt functions that can be used with tuples. Some of these are shownin Table 11.1.

Table 11.1 Inbuilt functions that can be used with tuples

Inbuilt Functions	*Meaning*
len()	Returns the number of elements in a tuple
max()	Returns the element with the greatest value
min()	Returns the element with the smallest value
sum()	Returns the sum of all the elements of a tuple
index(x)	Returns the index of element x
count(x)	Returns the number of occurrences of element x

Example

```
>>> t1=("APPLE")
>>> len(t1)  #Return the length of tuple t1
5
>>> max(t1)  #Return Element from tuple with Maximum Value
'P'
>>> min(t1)  #Return Element from tuple with Minimum Value
'A'
>>> t1.index('A')
0
>>> t1.count('P')
2
```

Q13. Define Python Set. How to create set(creating empty set, creating set of 4 elements).

Ans.

A set is an unordered collection of unique elements without duplicates. A set is mutable. Hence, we can easily add or remove elements from a set. The set data structure in Python is used to support mathematical set operations.

Creating Sets

A programmer can create a set by enclosing the elements inside a pair of curly brackets {}. The elements within a set can be separated using commas. We can also create a set using the inbuilt set()function, or from an existing list or tuple.

Examples

>>>S1 =set() # Creates an empty Set

>>>S1 # Print Set S1

set()

>>> type(S1) # Check type of S1

<class 'set'>

>>> S1={10,20,30,40} # Create set of 4 elements

>>> S1 # Print Set S1

{40, 10, 20, 30}

```
>>> S2=[1,2,3,2,5]          # Create List
>>> S2                      # Print List
[1, 2, 3, 2, 5]
>>> S3=set(S2)              # Convert List S2 to Set
>>> S3                      #Print S3 (Removes duplicate from the List)
{1, 2, 3, 5}
```

```
>>> S4=(1,2,3,4)            # Create Tuple
>>> S5=set(S4)              # Convert Tuple to Set
>>> S5                      # Print S5
{1, 2, 3, 4}
```

Q14. Explain Set operations in details.

Ans.

In mathematics or everyday applications we often use various set operations, such as union(), intersection(), difference()and symmetric_difference(). All these methods are partof the set class.

- ***The union()Method***

The union of two sets A and B is a set of elements which are in A, in B or in both A and B. We canuse the union method or the |operator to perform this operation.

Example

```
>>> S1={1,2,3,4}
>>> S2={2,4,5,6}
>>> S1.union(S2)
{1, 2, 3, 4, 5, 6}
```

```
>>>S1 | S2
{1, 2, 3, 4, 5, 6}
```

- ***The intersection()Method***

The intersection of two sets A and B is a set which contains all the elements of A that also belong to B. In short, intersection is a set which contains elements that appear in

both sets. We can use intersection methods or the & operator to perform this operation.

Example

```
>>> S1={1,2,3,4}

>>> S2={3,4,5,6}

>>> S1.intersection(S2)

{3, 4}

>>> S1 & S2

{3, 4}
```

- ***The difference()Method***

The difference between two sets A and B is a set which contains the elements in set A but not in set
A. We can use the difference method or the – operator to perform the difference operation.

Example

```
>>> A={1,2,3,4}

>>> B={3,4,5,6}

>>> A.difference(B)

{1, 2}

>>>>>> A-B

{1, 2}
```

- ***The symmetric_difference()***

The symmetric difference is a set which contains elements from the either set but not in both sets. We can use symmetric_difference method or the ^ (exclusive) operator to perform this operation.

Example

```
>>> S1={1,2,3,4}

>>> S2={3,4,5,6}

>>> S1.symmetric_difference(S2)

{1, 2, 5, 6}

>>> S1^S2

{1, 2, 5, 6}
```

Q15. Explain basics of dictionaries along with suitable diagram.

Ans.

In Python, a dictionary is a collection that stores values along with keys. The sequence of such keyand value pairs is separated by commas. These pairs are sometimes called entries or items. All entries are enclosed in curly brackets { and }. A colon separates a key and its value. Sometimes, items within dictionaries are also called associative arrays because they associate a key with a value.

Simple examples of dictionaries are given as follows:

Phonebook - {“Amit”:“918624986968”, “Amol”:“919766962920”}

Country Code Information - {“India”:“+91”,“USA”:“+1”,“Singapore”: “+65”}

The structure of a dictionary is shown in Figure 11.1a. The above phonebook example is illustrated in Figure 11.1b.

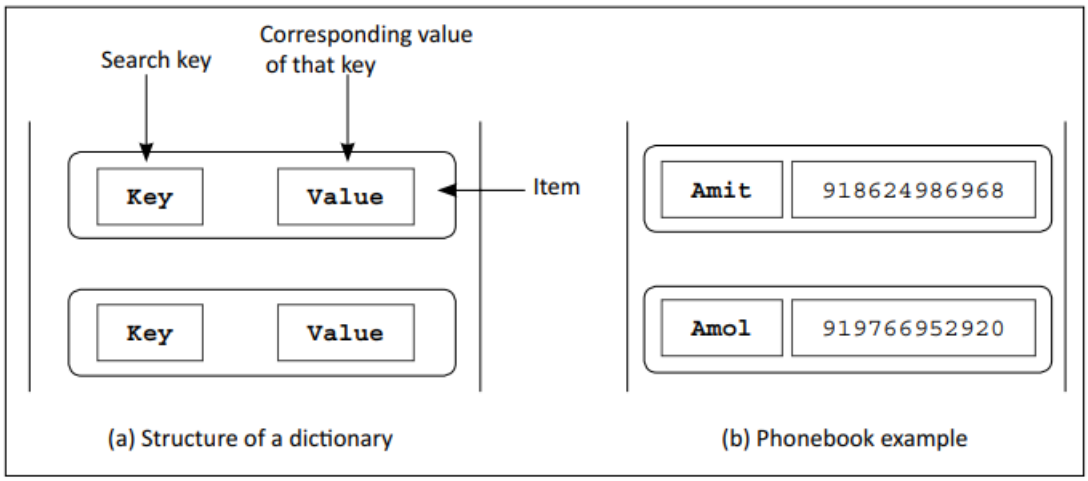

Figure 11.1 a and b Dictionary—structure and example

Keys are like an index operator in a dictionary. A key can be of any type. Therefore, a dictionarymaps a set of objects, i.e. keys to another set of objects, i.e. values. It is a mapping of unique keys to values, i.e. each key is mapped to one value. Also, dictionaries do not contain any duplicate keys.

Q16. Describe 4 different ways to create dictionaries.

Ans. We can create a dictionary by enclosing the items inside a pair of curly brackets {}. One way to starta dictionary is to create an empty dictionary first and then add items to it.

Creating an Empty Dictionary

Example

```
>>>D1 = {}                # Create Empty Dictionary
>>>D1                     # Print Empty Dictionary
{}
>>> type(D1)              # Check the type of D1
<class 'dict'>
```

Creating a Dictionary with Two Items

To create a dictionary of two items, the items should be in the form of key:value and separated bycommas.

```
>>> P={"Amit":"918624986968", "Amol":"919766962920"}
>>> P                #Display P
{'Amit': '918624986968', 'Amol': '919766962920'}
```

Creating Dictionaries in Four Ways:

Example

```
#Way 1:
>>>D1={'Name':'Sachin','Age':40}
>>> D1
{'Name': 'Sachin', 'Age': 40}

#Way 2:
>>> D2={}
>>> D2['Name']='Sachin'
>>> D2['Age']=40
>>> D2
{'Name': 'Sachin', 'Age': 40}

#Way 3:
>>> D3=dict(Name='Sachin',Age=40)
>>> D3
{'Name': 'Sachin', 'Age': 40}
```

#Way 4:

>>> dict([('name','Sachin'),('age',40)])

{'age': 40, 'name': 'Sachin'}

Explanation

In the above example, we have created dictionaries in four different ways. We can select the first way if we know all the contents of a dictionary in advance. The second way, if we want to add onefield at a time. The third way requires all keys to string. The fourth way is good if we want to buildkeys and values at runtime.