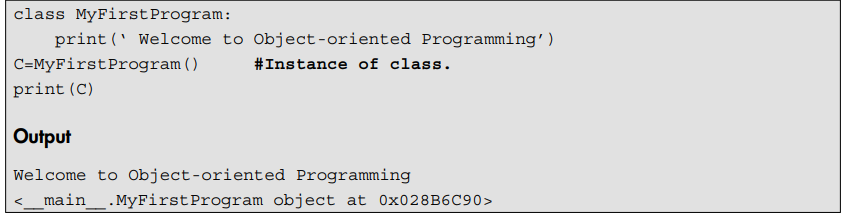
**Python Assignment No. 5**

**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 represents an entity which can be easily identified. For example, person, vehicle, fan, book etc. represent real objects. Each object has a unique identity, state and behaviour. The state of the object is also called property 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:



**Program:**



**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 last print 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**.

**Text, letter

Description automatically generated**

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

**Ans.**

**Text

Description automatically generated**

**Q4. Explain self parameter.**

**Ans.**

### The Self-parameter

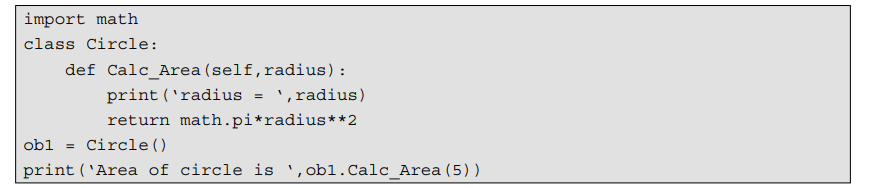
To add methods to an existing class, the first parameter for each method should be self. There is only one difference between class methods and ordinary functions. The self-parameter is used in the implementation of the method, but it is not used when the method is called. Therefore, the self-parameter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

Text, letter

Description automatically generated

**Explanation** In the above program, the Display\_Message() method takes no parameters but still 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.**

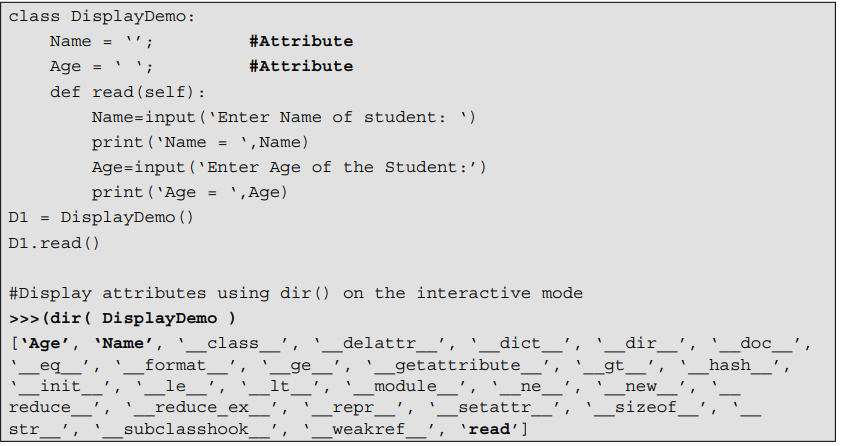
**Ans.** **Shape, rectangle

Description automatically generated with medium confidence**

**Explanation** The class with name Circle is created as shown above. The extra parameter radius is passed 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.**

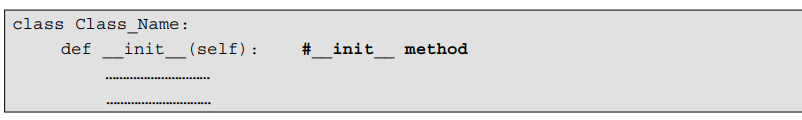
****

**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 importance of the init method is explained ahead.

The init method is known as an initialiser. It is a special method that is used to initialise the instance 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:

****

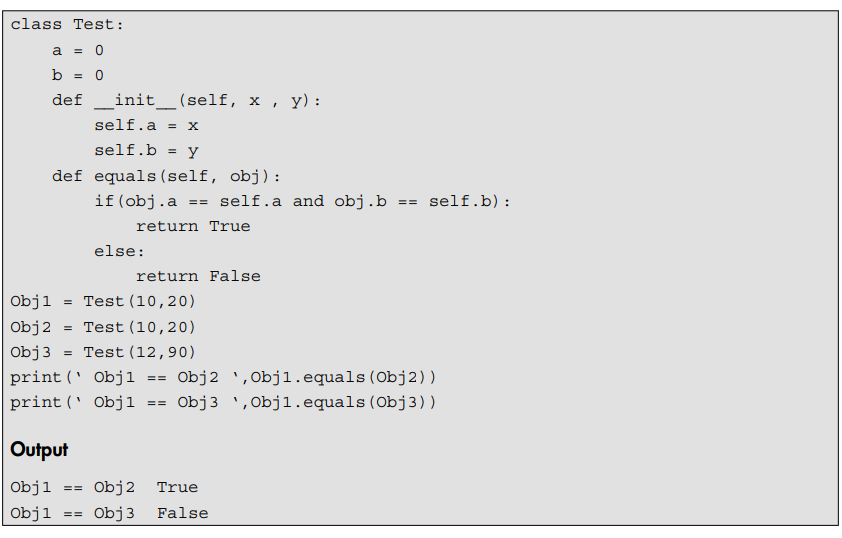
init needs to be preceded and followed by two underscores. Also init method must have self as the first argument. As self refers to the object itself, it refers to the object that invokes the method. The self-parameter within the init method automatically sets the reference for the object just created.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

**Graphical user interface, text, application

Description automatically generated**

**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.** ****

**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.

Python invokes the destructor method when the instance is about to be destroyed. It is invoked one 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 function is executed only if all the references to an instance object have been removed. Text

Description automatically generated Graphical user interface, text, application, email

Description automatically generated

**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. A new class created using an existing one is called a derived class or subclass and the existing class 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.

**Diagram

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

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

**Ans.**

The process 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 single class.
2. *Nested derivation:* The derived class can be used as the base class and a new class can be derived from it. This is possible at any extent.

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

1. ***Single inheritance:***Only one base class is used for deriving a new class. The derived class is not used as the base class.

P

Q

P is a base class. Q is a derived class. This type involves one base and one derived class. Further, no class is derived from Q.

Figure 10.3 Single inheritance

1. ***Multilevel inheritance:***When a class is derived from another derived class, the derived class acts as the base class. This is known as multilevel inheritance.

X

Z

Y

X is a base class. Y is derived from X. Z is derived from Y. Here, Y is not only a derived class but also a base class for Z. And Z can be used as a base class.

Figure 10.4 Multilevel inheritance

1. ***Multiple inheritance:***When two or more base classes are used for deriving a new class, it is called multiple Inheritance.



Z

Y

X

X and Y are base classes. Z is a derived class. Class Z inherits properties of both X and Y. Further, Z is not used as a base class.

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 of tuples 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 shown in 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 |

#### 

**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 part of 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 can use 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

1. 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.**

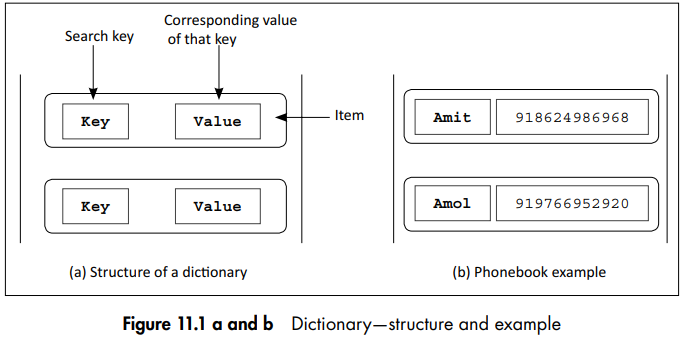
In Python, a dictionary is a collection that stores values along with keys. The sequence of such key and value pairs is separated by commas. These pairs are sometimes called entries or items. All entries are enclosed in curly brackets { and }. A colon separates a key and its value. Sometimes, items within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

****

Keys are like an index operator in a dictionary. A key can be of any type. Therefore, a dictionary maps 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 start a 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 by commas.

>>> 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 one field at a time. The third way requires all keys to string. The fourth way is good if we want to build keys and values at runtime.