BACHELOR OF COMPUTER APPLICATIONS

SEMESTER *5*

DCA3104

PYTHON PROGRAMMING

Unit 6

OOPS Through Python

Table of Contents

1. INTRODUCTION

In previous units, you have learned about the basic programming and syntax of Python. You can now make simple programs that can do basic arithmetic operations and build upon those basics. However, most real-life problems cannot be solved using functions and modules alone; what you need is an object. Python is an object-oriented language, though using OOP is not necessary as you start learning the language.

OOP is an essential part of a programming language that lets us manage the complexities we face in real-life situations. With each year, we are surpassing our technological management, including more complex features in our machine. Take software applications as an example. Back in time, these used to be very straightforward and served only one purpose. However, with time, their functionality has grown. Now, you expect that an application will have various features in it. While programming, these features need to be managed, so their complexities do not affect the user's experience.

Like to understand the working of a machine, it has to be broken into smaller and more manageable parts; similarly, it is better to take basic components during programming and then build the program from down. Objects are the basic blocks of programming that enable a programmer to handle real-life situations. They act as building blocks that come together to create a building. Before diving into OOP through Python, you first need to understand the components of OOP and how it works.

- Objects: Everything that we see around ourselves is an object. A book, a table, a fan, everything is an object that is further classified and given a name for our better understanding. In software development and programming language, this concept of object is used to simplify understanding the problem at hand. Any entity that has some sort of physical characteristics is classified as an object. For a programming language, an object has a definite state, shows a unique identity, and displays a behaviour.
- Classes: If each object in the world is an object, then there will be millions of objects that one has to consider. However, the management of such a large number can become complex and inefficient. Thus, as in human life, we have classified similar objects in one

group. In a programming language, we classify the objects that show the same behaviour or have the same state in a class.

- Benefits of OOP: Object-oriented programming has the foremost benefit of being essential to consider and compute real-life situations. It lets the programmer conceptualize the real-world problem in terms of a programming language.Through OOP, you can reuse the class you once created. It reduces redundancy and increases the efficiency of programming and readability. You do not have to change the entire algorithm if you want to change one program component. OOP gives you the freedom to make changes while leaving the other parts unaffected.

1.1 Learning Objectives

After studying this chapter, you will be able to:

- *Describe how to use classes and objects in Python.*
- *Understand object references and object identity.*
- *Describe how to create methods in Python.*
- *Explain the concept of encapsulation, polymorphism, and inheritance.*

2. CLASS IN PYTHON

To understand the term class, take an analogy like cats, jaguars, tigers, cheetah, and more. All these animals come under the cat family. On the other hand, dogs, wolves, foxes, and such come under the dog family. You can consider them subclasses that come under the larger class of animals. It is the concept of classes that are considered in Python as well.

STUDY NOTE

Classes in Python are usually names using singulars with the first letter in uppercase.

In Python, a class is defined as an object constructor. It acts as a blueprint to create objects. Class in Python is different from what we understand from it using real-life situations because its definition also integrates the supported methods. All instances, or objects, of a class follow a method. Thus, all the subclasses of that class will follow the method as well.

Creation of Class in Python

To create a class in Python, you will use the class keyword The name of the class immediately follows the statement. An example is shown below.

Example:

```
class ClassName:

    a = 6
```

- Here, a is an attribute of the class. Note that the class's name starts with a capital letter, as is the convention. The class contains a documentation string that is practical to obtain through ClassName._doc_.
- This documentation string, also known as docstring, is the first string inside the class. It has a small details of the class, and including a docstring is not necessary. However, it is advised to add it.

STUDY NOTE

In Java or C, curly braces are used to show that a code belongs to a certain function instead of indentation like in Python.

After class creation, an associated new class object is created quickly. This class object has the same name as the class. With the class object's help, you can access different class attributes and instantiate new objects. As you can see, the class does not inherit anything from any other class. Later in this chapter, we will study the inheritance property.

If you create a class with no attribute, the interpreter will give an error. Thus, you either have to put an attribute inside the class or write the keyword “pass”. This keyword tells the interpreter that it can move on to the next command line.

Also, note that everything inside the class is indented. In Python, indentation is an essential part of the syntax. Being indented means that the command or code is associated with the class. In the future, you will notice, indentation is used in the case of "if", "for", and other such cases.

SELF-ASSESSMENT QUESTIONS - 1

1. A class name can be any word written in any syntax as desired by the user. [True/False]

2. _____ is an object constructor in a programming language.

3. _____ is the first string mentioned inside the class.

4. To indicate that a code belongs to a class defined above it, one has to use _____ in Python.

5. Pick a valid class name?
   A. Animals
   B. Classes
   C. IAmAClass
   D. @ClassName!

3. OBJECT IN PYTHON

We have covered the fact that everything in Python is treated as an object. Variables, functions, lists, tuples, dictionaries, sets, etc., every entity defined in Python becomes objects. When you define an integer, it belongs to an integer class, just like a lion belongs to the animal class. Every object in Python has its own class. In a class, you define data structures that can hold data members. These data members, such as variables and constructs, are accessed by an object of the class.

STUDY NOTE

To access the class function, we use the name of the object along with the dot operator.

In order to check the type of user-defined classes and all objects, the function isinstance(object, type) is used.

How to create an object in Python

The method of constructing an object in a class is referred instantiation, and an object is called an instant. The syntax used to define an object inside a class is given below.

ObjectName = ClassName (<required arguements>)

Example Program: Here is a code that displays the name and grade of the student.

```
class Student:
        name = “Ishan”
        grade = 8
stu1 = Student()
print(stu1.name)
print(stu1.grade)
#The output of the program will be as follows:
Ishan
8
```

- Here, the object is stu1, and with the dot operator, we accessed the student's name and grade.

SELF-ASSESSMENT QUESTIONS - 2

6. _____ is used to access the class function with an object.

7. The process of creating an object with a class is called _____.

8. Which is the correct syntax to use an object of a class?
   A. classname_objectname
   B. classname.attribute
   C. classname = attribute
   D. None of the above

9. isinstance is used to check the type of a user-defined class. **[True/False]**

10. A class name can begin with a lowercase letter. **[True/False]**

4. OBJECT REFERENCES

Imagine an arrow. It is your object, your variable name that you created in the memory. The arrow, once released, will find a place to lodge, which will be its memory location. In Python's terms, whenever you proceed with the variable assignment, that is, assigning a value to a variable, what you are doing is forming an equation. You create the left side of the equation by declaring a variable. This variable gets saved in the memory with the address, say, 1800. Thus, the address 1800 refers to the variables that you created. In Python, everything, including the variable, is an object; therefore, this 1800 becomes your object reference that the variable's name will represent.

STUDY NOTE

Using the built-in function type() can return the type of the object.

The object reference is the indication of the memory location of the object placement. Take an example below to understand the concept deeper.

Example:

```
a = 5

b = 6

c = 8

a = b

b = c

print (a, b)

#The output of the program will be:

6, 8
```

- The value of a, as assigned to the user, should be 5. Similarly, the value of b should be 6. However, when you wrote 'a = b' and 'b = c', what you essentially did is commanded that a is b, and b is c. Thus, the location of variable 'a' became the location of variable 'b', and

the location of 'c' became that of 'b'. Changing the object reference also changed the value stored in that variable.

An object in Python exists as long as it has an object reference. If the number of object references drops to one, then that object is no longer accessible. Thus, we say that the lifetime of the object that began with its creation ended. In the future, we can store other data in this empty memory.

SELF-ASSESSMENT QUESTIONS - 3

11. The location in the memory where the object is stored is:
    A. Reference
    B. Object reference
    C. All of the above
    D. None of the above

12. An object with no object references is _____.

13. If an object is defined in Python as a = 6, then the object reference of 6 is _____.

14. There can be a data in the memory with no object reference. **[True/False]**

15. The address of a data in memory is returned in the form of an integer. **[True/False]**

5. OBJECT IDENTITY

Every object in Python has a unique identity. The interpreter ensures that no two objects have the same identity if they both have overlapping lifetimes. When there is no object reference for an object, only then its identity becomes available to be assigned to any other object. Python has a built-in function, id(), to return an object's identity in the form of an integer. This integer represents the location in the memory where the object has been stored.

STUDY NOTE

Unique objects having a value between the range of -5 to 256 will have the same object identity. This is because, at the startup, the interpreter creates an object in this range and then as the program continues, it reuses them. -

Example:

```
n = 60

id(n)

7811108
```

Activity I

Create a program that declares two objects. Assign the object reference of one object to the other (for example, if you have objects a and b, you can simply write a = b). Compare the object identity of the objects. Now, create two objects that store the same value. What are their object identities? Discuss the results.

SELF-ASSESSMENT QUESTIONS - 4

16. In Python no two objects have same identity. **[True/False]**
17. The function id() in Python, returns integer value representing:
    A. Identity of the object
    B. the location in the memory where the object has been stored.
    C. All of the above
    D. None of the above
18. When there is no _____ for an object, only then its identity becomes available to be assigned to any other object.

6. CREATING METHODS IN PYTHON

Before you begin to create Python methods, you will first have to grasp the concept of function. A function is a set of commands or parameters which are closed under a parenthesis. They perform a specific action and return a value. Functions are great tools to simplify and shorten the length of the code. If you find that you have to perform an action repeatedly in a program, then instead of writing the code again and again, all you have to do is write it once in a function. Now, whenever you need to perform a function, you will have to call the function. It will return the value so that your code works impeccably and has a few codes as possible.

STUDY NOTE

A Python method is similar to functions but it is applicable on objects.

Python has various built-in functions that perform the functions that are commonly used across various programs. Take print () for an example. It displays the result that you want to print on the output screen. How does it do it? The commands required to display a result have already been coded in the print () function. Hence, a user does not have to write it again in every program. Similarly, functions like len() and many others are available in the library of Python.

Users can also define and create functions in Python. You will have to use the keywords “def” to define a function. You can add parameters and commands inside the function and then call the function anywhere in the program.

Example: Here is an example of a function defined in Python.

```
def hello (num):
        return “Hello,” + name
```

- The code creates a function named "hello". It prints the string "hello" with the name entered by the user. Notice that the second line of command is indented to tell the interpreter that it is associated with the function. You can call this function in the program, as is shown below.

```
print hello (“World”)

#The output of this will be: Hello, World.
```

Now that you have understood the concept of functions let's move on to methods. You must have come across the word methods in the previous units, especially while defining various data types such as lists, sets, tuples, dictionaries, etc. For example, by using the method append (), to the end of a list you can append an object. Or using the method remove () deletes an object from the sequence data types. As you have already used methods in programs, you must have a vague idea about the term's meaning.

A method is a process through which you represent a class's behaviour the objects associated with the class. In Python, a method works similarly as a function. However, a method must be called on an object.

You can refer to previous units to get the list of methods along with their description in Python.

How to Create a Method?

A method needs to be created inside a class. The example provided below explains the syntax needed to be used to create a method.

Example:

```
class Hello:
  def __init__(self):
     pass
  def printhello(self, name):
     print(f"Hello, {name}")
     return name
```

```
myname = Hello()

myname.printhello('Ayush')
```

- Here, the method defined is printhello (). It has parameters and returns a value.

What is __init__ ()? If you are familiar with other object-oriented languages such as C++, you might know that a special function is used to initiate the class attributes. This function is called constructor and usually has the same name as the class. In Python, the method __init__ is used for the same purpose. The parameter in this is self, as it calls to the class itself. The keyword "pass" is used as the method cannot be kept empty.

Activity II

Find the error in the code given below.

```
class Bird:
        def __init__ (self, name, age)
        self.name = name
        self.age = age
        def sing (self, song):
                return "sings". format (self, name, song)
        def chirps(self):
                return "is chirping". format (self, name)
sparrow = Bird ( "Sparrow", 3)
print(sparrow.sing ("Happy"))
print(sparrow.chirps())
```

SELF-ASSESSMENT QUESTIONS - 5

19. A method can be created anywhere in the program. **[True/False]**

20. Functions are a set of commands that have parameters and return a value. [True/False]

21. To define a method, you need to use the keyword def. **[True or False]**

22. ___init___ () is used to initialize the class attributes. **[True or False]**

23. Which of the following is a valid name of a method in Python?
    A. AMethod
    B. @This_is_a_method
    C. methodname
    D. !Method!

7. IMPLEMENTATION OF ENCAPSULATION

Through encapsulation, you can wrap data and methods in one unit. Thus, other users cannot access the elements of the unit or change them. The method is beneficial to avoid any accidental changes in the data. Encapsulation allows change in an object's variable only through the object's method. Such variables are called private variables.

Encapsulation is easily exemplified by class. It encases all of the elements as well as the data. such as member functions, variables, etc., and does not allow any changes.

Protected members

The members of a class are not available to access from the exterior of the class. Accessibility is permitted within the class or its subclasses only. In Python, a data member is protected by prefixing it with a single underscore "_".

Private members

Private members are secured, but these cannot be accessed even by the super or parent class's subclasses, and a double underscore "__" is used as a prefix to inform the interpreter that a data member is private. Note, remember that the init() method is preceded and followed by a double underscore.

Here is a sample code that signifies protected and private members in a Python program.

STUDY NOTE

A class is a programming construct that contains all of the variables and methods specified within it.

Sample code:

```
class MyClass:
    def __init__(self):
        self.a = "This is my class"
        self._b = "This is a protected member"
        self.__c = "This is a private member"


class ChildClass(MyClass):
    def __init__(self):
        super().__init__()
        print(self._b)


obj1 = MyClass()
print(obj1.a)
```

Try to run the program and find the output that you get.

SELF-ASSESSMENT QUESTIONS - 6

24. Private members of a class cannot be accessed. **[True or False]**

25. Protected class members begin with an underscore. **[True or False]**

26. Protected class members can be accessed within a class. **[True or False]**

27. _____ is used to define a private class member.

28. The method to wrap data into one unit:

    A. Encapsulation
    B. Polymorphism
    C. Inheritance
    D. None of the above

8. IMPLEMENTATION OF POLYMORPHISM

Assume a situation where you have to calculate the price of an item. The price is stored in the form of a list, a tuple, or even a dictionary. Depending upon the type of instance, you will have to put conditions in your program to retrieve the value.

However, is it always possible to know the kind of data type used? What will happen if the data type is changed for some reason? You will have to change the entire code as well. Changing the code, again and again, is neither productive nor uses the intensive features that Python provides

STUDY NOTE

Since Python allows various classes to provide methods with identical name, we may use the principle of polymorphism when constructing class methods.

You can let the object decide the operation and carry it out by itself in such a situation. The object can retrieve its value on its own. You will not have to make any corrections or change the data type again. It is possible through polymorphism.

Polymorphism means the occurrence of a state in different forms. Through polymorphism, you can express an entity in various forms and scenarios. Take the example of + sign in Python. for integers, it is used to perform arithmetic addition. When used between two strings, it performs concatenation. Thus, + operator in Python can be used in different forms, depending upon the syntax.

Similarly, functions like len(), del, etc., can be used with various data types. When using the function len(), you do not know whether the data type whose length you can calculate is an integer or a string. The function will return the length of the data type, regardless of its nature. Here is an example.

Example:

```
print (len (“Python”))
print (len ([9, 10, 11]))
#The output of the program will be:
6
3
```

When it comes to classes, polymorphism comes especially handy as, through it, Python permits the usage of methods of the same name across various classes in a program. You can also define polymorphic functions. The functions will perform the action for which the commands are provided without knowing the object's data type or other attributes.

Example:

```
def add (x, y, z = 0):
        return x + y + z
print (add (2, 3))
print (add (4, 5, 6))
Output is shown below:
5
15
```

Similarly, through polymorphism, you can apply two distinct class types in the same manner. Another essential function of polymorphism is overriding. When a child class obtains a parent class's properties through inheritance, it is practical to alter a method in the inherited child class. The process involves the reimplementation of the method in child class. This method is called method overriding. All methods of the parent class, except private methods, can be overridden through polymorphism. However, the opposite of this is not possible. You cannot override a method in the subclass through the super or parent class.

SELF-ASSESSMENT QUESTIONS - 7

29. Why is polymorphism used?
    A. There is less code to write.
    B. Program code takes less space
    C. The programmer can think on an abstract level
    D. The code will be elegant and easy to maintain
30. Overriding means changing the behaviour of methods of derived class methods in the parent class. **[True or False]**
31. It is possible to override a _____ private method.

9. IMPLEMENTATION OF INHERITANCE

Inheritance is another essential property of object-oriented programming. Through this process, one class can acquire the properties of a different class. The process ensures that a code is reused so that it decreases redundancy. Through inheritance, you can effectively recreate various real-life situations where one class has similar base class properties. Through inheritance, you can make changes in the base class's properties in the child class without the base class alteration.

STUDY NOTE

Inheritance is transitive in nature. That is, if a subclass inherits the properties from the base class, then all the subclasses of that subclass will also inherit those properties. Here is a sample code to show you how inheritance works.

class Employee:

```
  def __init__(self, name):
     self.name = name
     self.ID = None
   def getName(self):
     return self.name
   def getID(self):
     return self.ID
   def checkemployee(self):
     return False
class Department(Employee):
  def checkemployee(self):
```

```
    return True

emp = Employee("Ayush")

emp.ID = "145"

 7print(emp.getName(), emp.getID(), emp.checkemployee())
```

For the child class to identify its parent class, it needs to mention the parent class in its definition. This method is subclassing or calling the constructor of the parent class. The syntax followed for this is as follows.

```
class subclass_name (superclass_name)
```

Types of Inheritance

The types of inheritance are divided based on the number of child and parent classes.

- **Single Inheritance:** The attributes of a single parent class are passed on to the child class.
- **Multiple Inheritance:** When a derived class gets its properties to form more than one superclass, it is called multiple inheritances. The subclass derives all the properties of the parent classes.
- **Multilevel Inheritance:** When another generation of classes further inherits the parent and child class features, it is classed as a multilevel inheritance.
- **Hierarchical Inheritance:** When more than one derived class receive the properties from one individual parent class, then it is the case of hierarchical inheritance.
- **Hybrid Inheritance:** Hybrid inheritance occurs when various types of inheritances are used together.

SELF-ASSESSMENT QUESTIONS - 8

32. Which of the following is a wrong statement about inheritance in Python?
    A. The inheriting class is referred as a subclass.
    B. Private members from a class can be acquired.
    C. Inheritance is a feature of OOP
    D. Protected members of a class can be inherited.

33. Which of the following is not a category of inheritance?
    A. Double-level
    B. Multiple
    C. Single-level
    D. Multi-level

34. All subclasses are a subtype in object-oriented programming. **[True or False]**

35. When a subclass is defined in Python that serves a subtype, the keyword used is a subtype. **[True or False]**

36. _____ is when different types of inheritance are used at the same time.

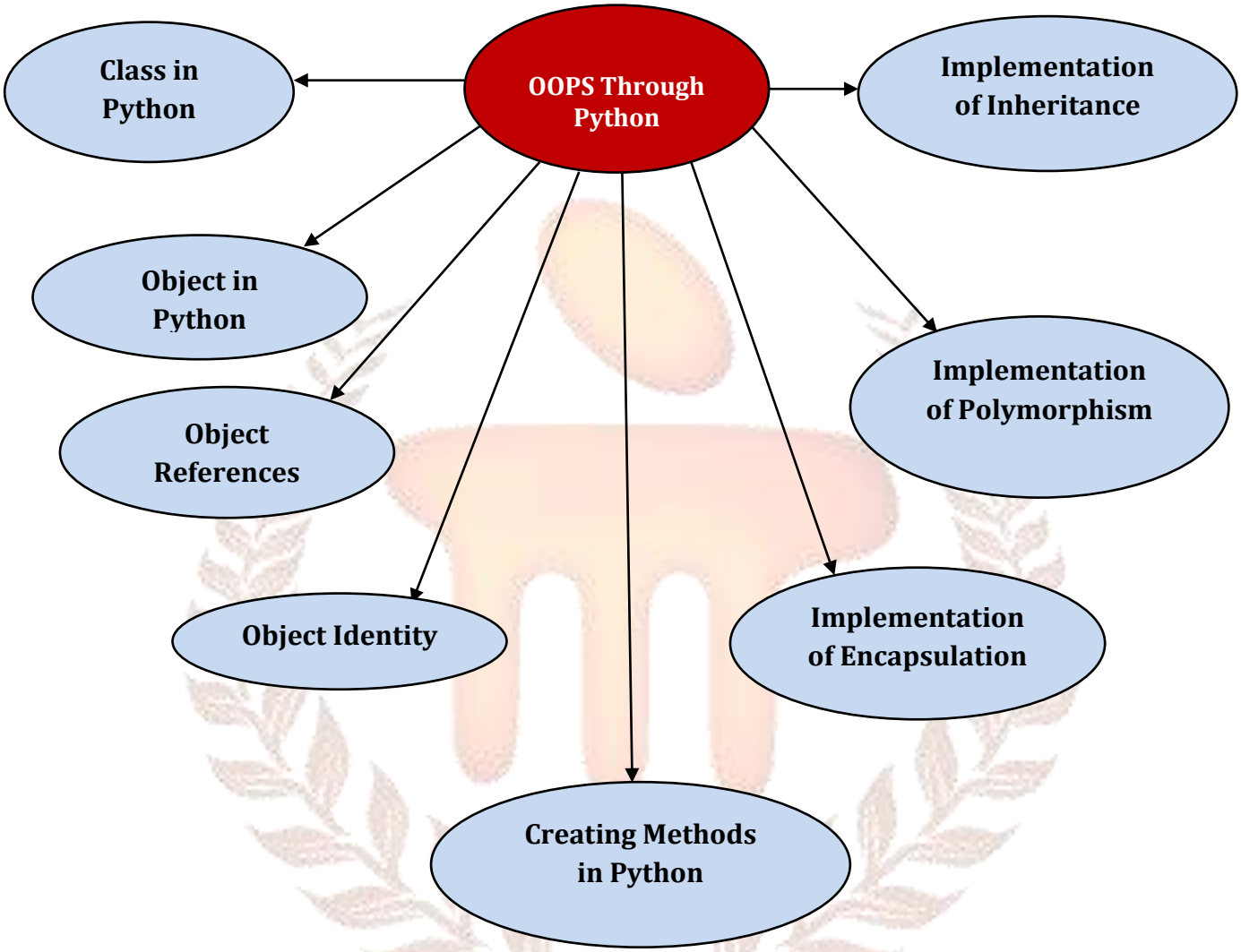

Fig 1: Conceptual Map

10. SUMMARY

- Python is an example of object-oriented language. All in the code is considered as an object, such as variables, classes, functions, etc.
- Class defines the attributes of an object. It is a code template used for creating objects.
- Object in Python is a collection of data or variables and methods that are associated with that data.
- Object reference represents the memory address where the object is stored. It provides a unique identity to the object.
- Object identity is used to identify an object used in a program uniquely. The id() function is used to find the identity of any object. It returns the address in the memory where the object is stored.
- A method in Python is a function that belongs to an object. They are defined under a class with the keyword def.
- Python library provides various methods for various objects such as lists, tuples, sets, dictionaries, strings, etc.
- Encapsulation is when the data and functions that operate on that data are packed into a single component. The object's elements cannot be accessed or altered by someone else.
- A protected member can only be accessed by the subclasses or child classes of the parent class.
- A private data member in a class cannot be accessed by the superclass's subclasses or child classes.
- Through polymorphism, one can create class methods having the same name in different classes. In polymorphism, the child class and parent class can have identical names.
- The inheritance function allows you to create a class that acquire the methods, properties, and attributes of another class. The child class is tagged as the derived class since it inherits the parent class properties.
- There are many different inheritance categories like single inheritance, multiple inheritance, multilevel inheritance, and hierarchical inheritance.

- When two or more inheritances are used at the same time, it is hybrid inheritance.

11. GLOSSARY

Object: It is a unique data structure that is defined with the help of its class. It includes data members as well as methods.

Class: It's an object prototype that's been defined by the user. It specifies the class's attributes.

Data member: It is a class variable that carries the data related to a class and its objects.

Class variable: It is a variable that all the instances in the class share. They are defined outside of the methods of class but exist inside the class.

Instance: An instance of a class is a single object of that class.

Instantiation: Construction of a class instance.

Method: Specific function attributable to a class or object.

Instance variable: The variable that belongs to the class's instance where it is defined.

Function overloading: When one function is assigned with more than one behaviour.

Operator overloading: When an operator is assigned more than one function.

Inheritance: The method of transferring the attributes of one class to other classes that are obtained from it.

Encapsulation: It enables the user to use an object without the knowledge of how it is constructed.

Polymorphism: It enables the user to call an object's method without the knowledge of the class or type of the object.

12. CASE STUDY

Automation of Library Function in University

A University is planning to convert its library and all its resources to be available online. The system of borrowing books is to be automated for a smoother function of the library. The system created should be such that the students can access the library's database even if they are not on the campus.

For this, a team has been set up. The team has to create light, well-structured, and easy to navigate software for the users. They have been given a time limit to convert one section of the library online through this software.

The team decided to keep the application code as short as possible to increase the project's efficiency. The application is created such that if one enters the unique code given to each book, then all other information regarding the book will appear, including its title, author, publication house, and publishing year. The application should add books to the database, view a book, and delete a book from the records.

Source-

Discussion Questions:

1. What classes should be defined to create the application? Can you find the inherited classes and their objects?
2. Write and execute the code required for the application.

13. TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Q1. Develop a program to construct a class and display the namespace of the class.

Q2. Write a program to form an instance of a class and print the instance's namespace.

Q3. Define a function in Python that gives a square of a number.

Q4. Create a class in Python names Student. Print the type of class.

Q5. What is inheritance?

LONG ANSWER QUESTIONS

Q1. Formulate a Python program that translate an integer to a roman numeral.

Q2. Formulate a Python program that derives all the unique subsets of the given set. Take the input set to be [2,7,8].

Q3. Formulate a Python program that discloses the cube of a given number using pow(x,n).

Q4. Formulate a Python program to exhibit the class name of an object as output.

Q5. Explain the concept of encapsulation.

14. ANSWERS

SELF ASSESSMENT QUESTIONS

1. False
2. Class
3. Docstring
4. Indentation
5. A, B, C
6. Dot operator
7. Instantiation
8. B. classname.attribute
9. True
10. False
11. Object reference
12. no longer valid
13. a
14. False
15. True
16. True
17. C. All of the above
18. B. Object reference
19. False
20. True
21. True
22. True
23. A and C
24. False
25. True
26. True
27. Double underscore
28. A. Encapsulation

29. D. The code will be elegant and easy to maintain.
30. False
31. Private
32. Option B
33. A. Double-level
34. False
35. False
36. Hybrid

TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Answer 1:

```
class dis_solution:
  def sub_sets(self, sset):
    return self.subsetsRecur([], sorted(sset))
  def subsetsRecur(self, current, sset):
    if sset:
      return self.subsetsRecur(current, sset[1:]) + self.subsetsRecur(current + [sset[0]], sset[1:])
    return [current]
for name in dis_solution.__dict__:
  print(name)
```

Answer 2:

```
class MyClass:
  def __init__(self, x, y):
```

```
    self.x = x
    self.y = y
instance = MyClass(10, 20)
print(instance.__dict__)
```

Answer 3:

```
def square(x):
return x*x
print(square(4))
```

Answer 4:

```
class Student:
  pass
print(type(Student))
```

Answer 5: Inheritance is the process through which the attributes of one class are moved to other classes constructed from it. One class can derive its properties from one or multiple parent classes. Similarly, one superclass can have one of the multiple subclasses.

LONG ANSWER QUESTIONS

```
from collections import OrderDict
def write_roman (num):
        roman = OrderDict()
        roman [1000] = "M"
        roman [900] = "CM"
        roman [500] = "D"
```

```
        roman [400] = “CD”

        roman [100] = “C”

        roman [90] = “XC”

        roman [50] = “L”

        roman [40] = “XL”

        roman [10] = “X”

        roman [5] = “V”

        roman [4] = “IV”

        roman [1] = “I”

def romam_num(num)”

        for r in roman.keys():

                x, y = divmod(num, r)

                yield roman[r] * x

                num -= (r * x)

                if num <= 0:

                        break

return “”.join([a for a in roman_num(num)])

num = 30

print write_roman(num)
```

Answer 2:

```
class SetSolution:
  def sub_sets(self, sset):
    return self.subsetsRecur([], sorted(sset))
  def subsetsRecur(self, current, sset):
    if sset:
      return self.subsetsRecur(current, sset[1:]) + self.subsetsRecur(current + [sset[0]], sset[1:])
    return [current]
print(SetSolution().sub_sets([2, 7, 8]))
```

Answer 3:

```
class pow_solution:
  def pow (self, x, n):
    if x==0 or x==1 or n==1:
      return x
    if x==-1:
      if n%2 ==0:
        return 1
      else:
        return -1
    if n==0:
```

```
        return 1
    if n<0:
        return 1/self.pow(x,-n)
    val = self.pow(x,n//2)
    if n%2 ==0:
        return val*val
    return val*val*x
print(pow_solution().pow(2, 3));
```

Answer 4:

```
import itertools
x = itertools.cycle('WXYZ')
print(type(x).__name__)
```

Answer 5:

Provision for controlling access to methods and variables is available and applicable in an object-oriented python program. This provision eliminates the modification of data by accident, and this process is known as encapsulation. One can define a class's data members as private or protected to ensure that other users cannot access the data. It is also beneficial to protect the data or data members against any accidental changes.

15. SUGGESTED BOOKS AND E-REFERENCES

BOOKS:

- Dusty Phillips (2010) Python 3 Object Oriented Programming. 1st edn. Packt Publishing Limited.
- Allen Downey, Jeffrey Elkner, Chris Meyers (2015) Learning with Python. 1st edn. Dreamtech Press.
- Booch (2009) Object-Oriented Analysis and Design with Applications, 3rd edn. Pearson Education India.
- Sebastian Raschka (2017) Python Machine Learning. 2nd edn. Packt Publishing.

REFERENCES:

- Object-Oriented Programming in Python, viewed on 27 March 2021

  <https://realpython.com/python3-object-oriented-programming/#what-is-object-oriented-programming-oop>

- Python Object Oriented Programming, viewed on 27 March 2021

  <https://www.programiz.com/python-programming/object-oriented-programming>

- Learn OOP in Python Short Course, viewed on 27 March 2021

  <https://www.youtube.com/playlist?list=PLQ1vEpscLn-vrc-5Nynkr7AEfko0k4tEr>