1. _____ represents an entity in the real world with its identity and behaviour.

a) A method

b) An object

c) A class

d) An operator

2. _____ is used to create an object.

a) class

b) constructor

c) User-defined functions

d) In-built functions

3. What will be the output of the following Python code?

```
class test:
    def __init__(self,a="Hello World"):
        self.a=a
     def display(self):
        print(self.a)
obj=test()
obj.display()
```

a) The program has an error because constructor can't have default arguments

b) Nothing is displayed

c) "Hello World" is displayed

d) The program has an error display function doesn't have parameters

4. What is setattr() used for?

a) To access the attribute of the object

b) To set an attribute

c) To check if an attribute exists or not

d) To delete an attribute

5. What is getattr() used for?

a) To access the attribute of the object

b) To delete an attribute

c) To check if an attribute exists or not

d) To set an attribute

6. What will be the output of the following Python code?

```
class change:
    def __init__(self, x, y, z):
        self.a = x + y + z
x = change(1,2,3)
y = getattr(x, 'a')
setattr(x, 'a', y+1)
print(x.a)
```

a) 6

b) 7

c) Error

d) 0

7. What will be the output of the following Python code?

```
class test:
    def __init__(self,a):
        self.a=a
    def display(self):
        print(self.a)
obj=test()
obj.display()
```

a) Runs normally, doesn't display anything

b) Displays 0, which is the automatic default value

c) Error as one argument is required while creating the object

d) Error as display function requires additional argument

8. Is the following Python code correct?

```
class A:
    def __init__(self,b):
        self.b=b
    def display(self):
          print(self.b)
obj=A("Hello")
del obj
```

a) True

b) False

9. What will be the output of the following Python code?

```
class test:
    def __init__(self):
        self.variable = 'Old'
        self.Change(self.variable)
    def Change(self, var):
        var = 'New'
obj=test()
print(obj.variable)
```

a) Error because function change can’t be called in the __init__ function

b) ‘New’ is printed

c) ‘Old’ is printed

d) Nothing is printed

10. What is Instantiation in terms of OOP terminology?

a) Deleting an instance of class

b) Modifying an instance of class

c) Copying an instance of class

d) Creating an instance of class

11. What will be the output of the following Python code?

```
class fruits:
    def __init__(self, price):
        self.price = price
obj=fruits(50)
obj.quantity=10
obj.bags=2
print(obj.quantity+len(obj.__dict__))
```

a) 12

b) 52

c) 13

d) 60

12. What will be the output of the following Python code?

```
class Demo:
    def __init__(self):
        pass
    def test(self):
        print(__name__)
obj = Demo()
obj.test()
```

a) Exception is thrown

b) __main__

c) Demo

d) test

13. The assignment of more than one function to a particular operator is _____

a) Operator over-assignment

b) Operator overriding

c) Operator overloading

d) Operator instance

14. Which of the following is not a class method?

a) Non-static

b) Static

c) Bounded

d) Unbounded

15. What will be the output of the following Python code?

```
c.test=c.test+1
k=k+1
class A:
    def __init__(self):
        self.test = 0
def main():
    Count=A()
    k=0
    for i in range(0,25):
        add(Count,k)
    print("Count.test=", Count.test)
    print("k =", k)
main()
```

a) Exception is thrown

b)

Count.test=25

k=25

c)

Count.test=25

k=0

d)

Count.test=0

k=0

16. Which of the following Python code creates an empty class?

a)

```
class A:
    return
```

b)

```
class A:
    pass
```

c)

```
class A:
```

d) It is not possible to create an empty class

17. Is the following Python code valid?

```
class B(object):
    def first(self):
        print("First method called")
    def second():
        print("Second method called")
ob = B()
B.first(ob)
```

a) It isn’t as the object declaration isn’t right

b) It isn’t as there isn’t any __init__ method for initializing class members

c) Yes, this method of calling is called unbounded method call

d) Yes, this method of calling is called bounded method call

18. What are the methods which begin and end with two underscore characters called?

a) Special methods

b) In-built methods

c) User-defined methods

d) Additional methods

19. Special methods need to be explicitly called during object creation.

a) True

b) False

20. What will be the output of the following Python code?

```
class demo():
    def __repr__(self):
        return '__repr__ built-in function called'
    def __str__(self):
        return '__str__ built-in function called'
s=demo()
s
```

a) Error

b) Nothing is printed

c) __str__ called

d) __repr__ called

21. What will be the output of the following Python code?

```
class demo():
    def __repr__(self):
        return '__repr__ built-in function called'
    def __str__(self):
        return '__str__  built-in function called'
s=demo()
print(s)
```

a) __str__ called

b) __repr__ called

c) Error

d) Nothing is printed

22. What is hasattr(obj,name) used for?

a) To access the attribute of the object

b) To delete an attribute

c) To check if an attribute exists or not

d) To set an attribute

23. What will be the output of the following Python code?

```
class stud:
    def __init__(self, roll_no, grade):
        self.roll_no = roll_no
        self.grade = grade
    def display (self):
        print("Roll no : ", self.roll_no,  ", Grade: ", self.grade)
stud1 = stud(34, 'S')
stud1.age=7
print(hasattr(stud1, 'age'))
```

a) Error as age isn't defined

b) True

c) False

d) 7

24. What is delattr(obj,name) used for?

a) To print deleted attribute

b) To delete an attribute

c) To check if an attribute is deleted or not

d) To set an attribute

25. __del__ method is used to destroy instances of a class.

a) True

b) False

26. What will be the output of the following Python code?

```
class stud:
   'Base class for all students'
   def __init__(self, roll_no, grade):
      self.roll_no = roll_no
      self.grade = grade
   def display (self):
      print("Roll no : ", self.roll_no,  ", Grade: ", self.grade)
print(student.__doc__)
```

a) Exception is thrown

b) __main__

c) Nothing is displayed

d) Base class for all students

27. What does print(Test.__name__) display (assuming Test is the name of the class)?

a) ()

b) Exception is thrown

c) Test

d) __main__

28. Which of the following best describes inheritance?

a) Ability of a class to derive members of another class as a part of its own definition

b) Means of bundling instance variables and methods in order to restrict access to certain class members

c) Focuses on variables and passing of variables to functions

d) Allows for implementation of elegant software that is well designed and easily modified

29. Which of the following statements is wrong about inheritance?

a) Protected members of a class can be inherited

b) The inheriting class is called a subclass

c) Private members of a class can be inherited and accessed

d) Inheritance is one of the features of OOP

30. What will be the output of the following Python code?

```
class Demo:
    def __new__(self):
        self.__init__(self)
        print("Demo's __new__() invoked")
    def __init__(self):
        print("Demo's __init__() invoked")
class Derived_Demo(Demo):
    def __new__(self):
        print("Derived_Demo's __new__() invoked")
    def __init__(self):
        print("Derived_Demo's __init__() invoked")
def main():
    obj1 = Derived_Demo()
    obj2 = Demo()
main()
```

a)

Derived_Demo’s __init__() invoked

Derived_Demo's __new__() invoked

Demo's __init__() invoked

Demo's __new__() invoked

b)

Derived_Demo's __new__() invoked

Demo's __init__() invoked

Demo's __new__() invoked

c)

Derived_Demo's __new__() invoked

Demo's __new__() invoked

d)

Derived_Demo's __init__() invoked

Demo's __init__() invoked

31. What will be the output of the following Python code?

```
class Test:
    def __init__(self):
        self.x = 0
class Derived_Test(Test):
    def __init__(self):
        self.y = 1
def main():
    b = Derived_Test()
    print(b.x,b.y)
main()
```

a) 0 1

b) 0 0

c) Error because class B inherits A but variable x isn't inherited

d) Error because when object is created, argument must be passed like Derived_Test(1)

32. What will be the output of the following Python code?

```
class A():
    def disp(self):
        print("A disp()")
class B(A):
    pass
obj = B()
obj.disp()
```

a) Invalid syntax for inheritance

b) Error because when object is created, argument must be passed

c) Nothing is printed

d) A disp()

33. All subclasses are a subtype in object-oriented programming.

a) True

b) False

34. When defining a subclass in Python that is meant to serve as a subtype, the subtype Python keyword is used.

a) True

b) False

35. Suppose B is a subclass of A, to invoke the __init__ method in A from B, what is the line of code you should write?

a) A.__init__(self)

b) B.__init__(self)

c) A.__init__(B)

d) B.__init__(A)

36. What will be the output of the following Python code?

```
class Test:
    def __init__(self):
        self.x = 0
class Derived_Test(Test):
    def __init__(self):
        Test.__init__(self)
        self.y = 1
def main():
    b = Derived_Test()
    print(b.x,b.y)
main()
```

a) Error because class B inherits A but variable x isn’t inherited

b) 0 0

c) 0 1

d) Error, the syntax of the invoking method is wrong

37. What will be the output of the following Python code?

```
class A:
    def __init__(self, x= 1):
        self.x = x
class der(A):
    def __init__(self,y = 2):
        super().__init__()
        self.y = y
def main():
    obj = der()
    print(obj.x, obj.y)
main()
```

a) Error, the syntax of the invoking method is wrong

b) The program runs fine but nothing is printed

c) 1 0

d) 1 2

38. What does built-in function type do in context of classes?

a) Determines the object name of any value

b) Determines the class name of any value

c) Determines class description of any value

d) Determines the file name of any value

39. Which of the following is not a type of inheritance?

a) Double-level

b) Multi-level

c) Single-level

d) Multiple

40. What does built-in function help do in context of classes?

a) Determines the object name of any value

b) Determines the class identifiers of any value

c) Determines class description of any built-in type

d) Determines class description of any user-defined built-in type

41. What will be the output of the following Python code?

```
class A:
    def one(self):
        return self.two()

    def two(self):
        return 'A'

class B(A):
    def two(self):
        return 'B'
obj1=A()
obj2=B()

print(obj1.two(),obj2.two())
```

a) A A

b) A B

c) B B

d) An exception is thrown

42. What type of inheritance is illustrated in the following Python code?

```
class A():
    pass
class B():
    pass
class C(A,B):
    pass
```

a) Multi-level inheritance

b) Multiple inheritance

c) Hierarchical inheritance

d) Single-level inheritance

43. What type of inheritance is illustrated in the following Python code?

```
class A():
    pass
class B(A):
    pass
class C(B):
    pass
```

a) Multi-level inheritance

b) Multiple inheritance

c) Hierarchical inheritance

d) Single-level inheritance

44. What does single-level inheritance mean?

a) A subclass derives from a class which in turn derives from another class

b) A single superclass inherits from multiple subclasses

c) A single subclass derives from a single superclass

d) Multiple base classes inherit a single derived class

45. What will be the output of the following Python code?

```
class A:
    def __init__(self):
        self.__i = 1
        self.j = 5

    def display(self):
        print(self.__i, self.j)

class B(A):
    def __init__(self):
        super().__init__()
        self.__i = 2
        self.j = 7
c = B()
c.display()
```

a) 2 7

b) 1 5

c) 1 7

d) 2 5

46. Which of the following statements isn't true?

a) A non-private method in a superclass can be overridden

b) A derived class is a subset of superclass

c) The value of a private variable in the superclass can be changed in the subclass

d) When invoking the constructor from a subclass, the constructor of superclass is automatically invoked

47. What will be the output of the following Python code?

```
class A:
    def __init__(self,x):
        self.x = x
    def count(self,x):
        self.x = self.x+1
class B(A):
    def __init__(self, y=0):
        A.__init__(self, 3)
        self.y = y
    def count(self):
        self.y += 1
```

```
def main():
    obj = B()
    obj.count()
    print(obj.x, obj.y)
main()
```

a) 3 0

b) 3 1

c) 0 1

d) An exception in thrown

48. What will be the output of the following Python code?

```
class A:
    pass
class B(A):
    pass
obj=B()
isinstance(obj,A)
```

a) True

b) False

c) Wrong syntax for isinstance() method

d) Invalid method for classes

49. Which of the following statements is true?

a) The __new__() method automatically invokes the __init__ method

b) The __init__ method is defined in the object class

c) The __eq(other) method is defined in the object class

d) The __repr__() method is defined in the object class

50. Method issubclass() checks if a class is a subclass of another class.

a) True

b) False

52. What will be the output of the following Python code?

```
class A:
    def __init__(self):
        self.__x = 1
class B(A):
    def display(self):
        print(self.__x)
def main():
    obj = B()
    obj.display()
main()
```

a) 1

b) 0

c) Error, invalid syntax for object declaration

d) Error, private class member can’t be accessed in a subclass

53. What will be the output of the following Python code?

```
class A:
    def __init__(self):
        self._x = 5
class B(A):
    def display(self):
        print(self._x)
def main():
    obj = B()
    obj.display()
main()
```

a) Error, invalid syntax for object declaration

b) Nothing is printed

c) 5

d) Error, private class member can’t be accessed in a subclass

54. What will be the output of the following Python code?

```
class A:
    def __init__(self,x=3):
        self._x = x
class B(A):
    def __init__(self):
        super().__init__(5)
    def display(self):
        print(self._x)
```

```
def main():
    obj = B()
    obj.display()
main()
```

a) 5

b) Error, class member x has two values

c) 3

d) Error, protected class member can’t be accessed in a subclass

55. What will be the output of the following Python code?

```
class A:
    def test1(self):
        print(" test of A called ")
class B(A):
    def test(self):
        print(" test of B called ")
class C(A):
    def test(self):
        print(" test of C called ")
class D(B,C):
    def test2(self):
        print(" test of D called ")
obj=D()
obj.test()
```

a)

test of B called

test of C called

b)

test of C called

test of B called

c) test of B called

d) Error, both the classes from which D derives has same method test()

56. What will be the output of the following Python code?

```
class A:
    def test(self):
        print("test of A called")
class B(A):
    def test(self):
        print("test of B called")
        super().test()
class C(A):
    def test(self):
        print("test of C called")
        super().test()
class D(B,C):
    def test2(self):
        print("test of D called")
obj=D()
obj.test()
```

a)

test of B called

test of C called

test of A called

b)

test of C called

test of B called

c)

test of B called

test of C called

d) Error, all the three classes from which D derives has same method test()

57. Which of the following best describes polymorphism?

a) Ability of a class to derive members of another class as a part of its own definition

b) Means of bundling instance variables and methods in order to restrict access to certain class members

c) Focuses on variables and passing of variables to functions

d) Allows for objects of different types and behaviour to be treated as the same general type

58. What is the biggest reason for the use of polymorphism?

a) It allows the programmer to think at a more abstract level

b) There is less program code to write

c) The program will have a more elegant design and will be easier to maintain and update

d) Program code takes up less space

59. What is the use of duck typing?

a) More restriction on the type values that can be passed to a given method

b) No restriction on the type values that can be passed to a given method

c) Less restriction on the type values that can be passed to a given method

d) Makes the program code smaller

60. What will be the output of the following Python code?

```
class A:
    def __str__(self):
        return '1'
class B(A):
    def __init__(self):
        super().__init__()
class C(B):
    def __init__(self):
        super().__init__()
def main():
    obj1 = B()
    obj2 = A()
    obj3 = C()
    print(obj1, obj2,obj3)
main()
```

a) 1 1 1

b) 1 2 3

c) ‘1’ ‘1’ ‘1’

d) An exception is thrown

61. What will be the output of the following Python code?

```
class Demo:
    def __init__(self):
        self.x = 1
    def change(self):
        self.x = 10
class Demo_derived(Demo):
    def change(self):
        self.x=self.x+1
        return self.x
def main():
    obj = Demo_derived()
    print(obj.change())

main()
```

a) 11

b) 2

c) 1

d) An exception is thrown

62. A class in which one or more methods are only implemented to raise an exception is called an abstract class.

a) True

b) False

63. Overriding means changing behaviour of methods of derived class methods in the base class.

a) True

b) False

64. What will be the output of the following Python code?

```
class A:
    def __repr__(self):
        return "1"
class B(A):
    def __repr__(self):
        return "2"
class C(B):
    def __repr__(self):
        return "3"
o1 = A()
o2 = B()
o3 = C()
print(obj1, obj2, obj3)
```

a) 1 1 1

b) 1 2 3

c) ‘1’ ‘1’ ‘1’

d) An exception is thrown

65. What will be the output of the following Python code?

```
class A:
    def __init__(self):
        self.multiply(15)
        print(self.i)

    def multiply(self, i):
        self.i = 4 * i;
class B(A):
    def __init__(self):
        super().__init__()

    def multiply(self, i):
        self.i = 2 * i;
obj = B()
```

a) 15

b) 60

c) An exception is thrown

d) 30

66. What will be the output of the following Python code?

```
class Demo:
    def check(self):
        return " Demo's check "
    def display(self):
        print(self.check())
class Demo_Derived(Demo):
    def check(self):
        return " Derived's check "
Demo().display()
Demo_Derived().display()
```

a) Demo’s check Derived’s check

b) Demo’s check Demo’s check

c) Derived’s check Demo’s check

d) Syntax error

67. What will be the output of the following Python code?

```
class A:
    def __init__(self):
        self.multiply(15)
    def multiply(self, i):
        self.i = 4 * i;
class B(A):
    def __init__(self):
        super().__init__()
        print(self.i)

    def multiply(self, i):
        self.i = 2 * i;
obj = B()
```

a) 15

b) 30

c) An exception is thrown

d) 60

68. What will be the output of the following Python code?

```
class Demo:
    def __check(self):
        return " Demo's check "
    def display(self):
        print(self.check())
class Demo_Derived(Demo):
    def __check(self):
        return " Derived's check "
Demo().display()
Demo_Derived().display()
```

a) Demo’s check Derived’s check

b) Demo’s check Demo’s check

c) Derived’s check Demo’s check

d) Syntax error

69. What will be the output of the following Python code?

```
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        return 1
    def __eq__(self, other):
        return self.x * self.y == other.x * other.y
obj1 = A(5, 2)
obj2 = A(2, 5)
print(obj1 == obj2)
```

a) False

b) 1

c) True

d) An exception is thrown

70. What will be the output of the following Python code?

```
class A:
    def one(self):
        return self.two()
    def two(self):
        return 'A'
class B(A):
    def two(self):
        return 'B'
obj2=B()
print(obj2.two())
```

a) A

b) An exception is thrown

c) A B

d) B

71. Which of the following statements is true?

a) A non-private method in a superclass can be overridden

b) A subclass method can be overridden by the superclass

c) A private method in a superclass can be overridden

d) Overriding isn’t possible in Python

72. Which of these is not a fundamental feature of OOP?

a) Encapsulation

b) Inheritance

c) Instantiation

d) Polymorphism

73. Which of the following is the most suitable definition for encapsulation?

a) Ability of a class to derive members of another class as a part of its own definition

b) Means of bundling instance variables and methods in order to restrict access to certain class members

c) Focuses on variables and passing of variables to functions

d) Allows for implementation of elegant software that is well designed and easily modified

74. What will be the output of the following Python code?

```
class Demo:
    def __init__(self):
        self.a = 1
        self.__b = 1

    def display(self):
        return self.__b
obj = Demo()
print(obj.a)
```

a) The program has an error because there isn't any function to return self.a

b) The program has an error because b is private and display(self) is returning a private member

c) The program runs fine and 1 is printed

d) The program has an error as you can't name a class member using __b

75. What will be the output of the following Python code?

```
class Demo:
    def __init__(self):
        self.a = 1
        self.__b = 1
    def display(self):
        return self.__b
obj = Demo()
print(obj.__b)
```

a) The program has an error because there isn't any function to return self.a

b) The program has an error because b is private and display(self) is returning a private member

c) The program has an error because b is private and hence can't be printed

d) The program runs fine and 1 is printed

76. Methods of a class that provide access to private members of the class are called as _____ and _____

a) getters/setters

b) __repr__/__str__

c) user-defined functions/in-built functions

d) __init__/__del__

77. Which of these is a private data field?

```
class Demo:
    def __init__(self):
        __a = 1
        self.__b = 1
        self.__c__ = 1
        __d__= 1
```

a) __a

b) __b

c) __c__

d) __d__

78. What will be the output of the following Python code?

```
class Demo:
    def __init__(self):
        self.a = 1
        self.__b = 1
    def get(self):
        return self.__b
obj = Demo()
print(obj.get())
```

a) The program has an error because there isn’t any function to return self.a

b) The program has an error because b is private and display(self) is returning a private member

c) The program has an error because b is private and hence can't be printed

d) The program runs fine and 1 is printed

79. What will be the output of the following Python code?

```
class Demo:
    def __init__(self):
        self.a = 1
        self.__b = 1
    def get(self):
        return self.__b
obj = Demo()
obj.a=45
print(obj.a)
```

a) The program runs properly and prints 45

b) The program has an error because the value of members of a class can't be changed from outside the class

c) The program runs properly and prints 1

d) The program has an error because the value of members outside a class can only be changed as self.a=45

80. Private members of a class cannot be accessed.

a) True

b) False

81. The purpose of name mangling is to avoid unintentional access of private class members.

a) True

b) False

82. What will be the output of the following Python code?

```
class fruits:
    def __init__(self):
        self.price = 100
        self.__bags = 5
    def display(self):
        print(self.__bags)
obj=fruits()
obj.display()
```

a) The program has an error because display() is trying to print a private class member

b) The program runs fine but nothing is printed

c) The program runs fine and 5 is printed

d) The program has an error because display() can’t be accessed

83. What will be the output of the following Python code?

```
class student:
    def __init__(self):
        self.marks = 97
        self.__cgpa = 8.7
    def display(self):
        print(self.marks)
obj=student()
print(obj._student__cgpa)
```

a) The program runs fine and 8.7 is printed

b) Error because private class members can’t be accessed

c) Error because the proper syntax for name mangling hasn’t been implemented

d) The program runs fine but nothing is printed

84. Which of the following is false about protected class members?

a) They begin with one underscore

b) They can be accessed by subclasses

c) They can be accessed by name mangling method

d) They can be accessed within a class

85. What will be the output of the following Python code?

```
class objects:
    def __init__(self):
        self.colour = None
        self._shape = "Circle"

    def display(self, s):
        self._shape = s
obj=objects()
print(obj._objects_shape)
```

a) The program runs fine because name mangling has been properly implemented

b) Error because the member shape is a protected member

c) Error because the proper syntax for name mangling hasn’t been implemented

d) Error because the member shape is a private member