1. Which of the following command is used to open a file “c:\temp.txt” in read-mode only?

A. infile = open(“c:\temp.txt”, “r”)

**B. infile = open(“c:\\temp.txt”, “r”)**

C. infile = open(file = “c:\temp.txt”, “r+”)

D. infile = open(file = “c:\\temp.txt”, “r+”)

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

try:

if '1' != 1:

raise "someError"

else:

print("someError has not occurred")

except "someError":

print ("someError has occurred")

a) someError has occurred

b) someError has not occurred

**c) invalid code**

d) none of the mentioned

Explanation: A new exception class must inherit from a BaseException. There is no such inheritance here.

1. Which type of inheritance is being used in the code given below?

class A:

pass

class B:

pass

class C(A, B):

pass

1. Inheritance
2. **Multilevel Inheritance**
3. Double Inheritance
4. None of these.

4. Which of the following allows us to read individual lines from the file?

1. **readline()**
2. readLine()
3. readfile()
4. readOneLine()

5. 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

Explanation: The super().\_\_init\_\_() in the subclasses has been properly invoked and none of other subclasses return any other value. Hence 1 is returned each time the object is created and printed.

6.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

Explanation: The derived class B overrides base class A.

7.In the following Python code, which function is the decorator?

def mk(x):

def mk1():

print("Decorated")

x()

return mk1

def mk2():

print("Ordinary")

p = mk(mk2)

p()

a) p()

**b) mk()**

c) mk1()

d) mk2()

Explanation: In the code shown above, the function mk() is the decorator. The function which is getting decorated is mk2(). The return function is given the name p().

8.The two snippets of the following Python codes are equivalent.(they are just demo functions. Use decorator concept.)

CODE 1

@f

def f1():

print(“Hello”)

CODE 2

def f1():

print(“Hello”)

f1 = f(f1)

**a) True**

b) False

Explanation: The @ symbol can be used as an alternate way to specify a function that needs to be decorated. The output of the codes shown above is the same. Hence they are equivalent. Therefore this statement is true.

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

class A:

@staticmethod

def a(x):

print(x)

A.a(100)

a) Error

b) Warning

**c) 100**

d) No output

Explanation: The code shown above demonstrates rebinding using a static method. This can be done with or without a decorator. The output of this code will be 100.

10.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(),end="")

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

Explanation: The method check is private so it can’t be accessed by the derived class. Execute the code in the Python shell.