**CLASSES AND OBJECTS**

Q1. What is an abstract class in Python?

a) A class that cannot be instantiated

b) A class with only private methods

c) A class with only public methods

d) A class with no methods

#### Q2. How do you define an abstract class in Python?

a) By using the abstract keyword

b) By using the abstract class statement

c) By importing the abc module and using @abstractmethod decorator

d) By using the virtual keyword

#### Q3. Which OOP principle allows a class to have multiple methods with the same name but different parameters?

a) Inheritance

b) Polymorphism

c) Encapsulation

d) Abstraction

#### Q4. Which method is called when an object is deleted?

a) \_\_delete\_\_

b) \_\_remove\_\_

c) \_\_del\_\_

d) \_\_destroy\_\_

#### Q5. What does the isinstance() function do in Python?

a) Checks if an object is an instance of a specific class

b) Checks if an object is a subclass of a specific class

c) Checks if two objects are equal

d) Checks if two objects are the same instance

#### Q6. What does the following Python code represent?

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def display\_info(self):

print("Name:", self.name)

print("Age:", self.age)

student1 = Student("John", 20)

#### student1.display\_info()

a) Definition of a Student class with attributes name and age

b) Creation of a Student object student1 with name “John” and age 20

c) Displaying information about the student

d) All of the above

#### Q7. What does the following Python code represent?

class Shape:

def \_\_init\_\_(self, sides):

self.sides = sides

def display\_sides(self):

print("Number of sides:", self.sides)

class Triangle(Shape):

def \_\_init\_\_(self, side1, side2, side3):

super().\_\_init\_\_(3)

self.side1 = side1

self.side2 = side2

self.side3 = side3

def display\_area(self):

s = (self.side1 + self.side2 + self.side3) / 2

area = (s \* (s - self.side1) \* (s - self.side2) \* (s - self.side3)) \*\* 0.5

print("Area of the triangle:", area)

triangle1 = Triangle(3, 4, 5)

triangle1.display\_sides()

triangle1.display\_area()

a) Number of sides: 3 Area of the triangle: 6.0

b) Number of sides: 3 Area of the triangle: 7.5

c) Number of sides: 4 Area of the triangle: 6.0

d) Number of sides: 4 Area of the triangle: 7.5

#### Q8. What is method overriding in OOP?

a) Creating a new method with the same name as a method in the superclass

b) Deleting a method from a superclass

c) Hiding a method in the superclass

d) Replacing a method in the superclass with a new implementation in the subclass

Q9. 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

Q10.  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