Editorial-W9A1: Python Basics and OOP

Question 1

You want to get some numbers from a user, as you are performing a study on finding most chosen numbers at random. Which of the following code snippets correctly accomplishes this?
(MSQ)

Options:

1.

```
numbers = input("Enter numbers separated by spaces: ").split()
numbers = [int(x) for x in numbers]
```

2.

```
numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
```

3.

```
numbers = [int(x) for x in input("Enter numbers separated by spaces: ").split()]
```

4.

```
numbers = input("Enter numbers separated by spaces: ").split()
numbers = list(map(int, numbers))
```

Answer 1, 2, 3, 4
Explanation: All options correctly convert space-separated input into a list of integers.

For Question 2 & 3

You are developing a geometry toolkit for an educational app that calculates areas of different shapes. To ensure that each shape class provides its own logic for computing area, you decide to implement an abstract class Shape with an abstract method area(). For example, the Circle class must implement the area() method based on its radius.

```
from abc import ABC, abstractmethod
import math

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass
```

```
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

circle = Circle(5)
print(circle.area())
```

Question 2(NAT)

What will be the output of the code above?
A) 78.53
B) 25
C) The code will raise an error because area() is not implemented in Circle.
D) The code will raise an error because Shape is not a subclass of ABC.

Answer: A) 78.53

Explanation: The Circle class implements the area() method using the formula π * radius^2, and with radius = 5, the output is the area of the circle.

Question 3(MSQ)

What is the purpose of the abstractmethod decorator in the Shape class?

A) To make the area() method optional for subclasses. B) To prevent the Shape class from being instantiated directly.
C) To specify (or enforce) that all subclasses must implement area()
D) To allow the Shape class to be instantiated without providing an area() method.

Answer: B) and C)

- The abstractmethod decorator ensures that the Shape class cannot be instantiated directly, as the area() method is abstract and needs to be implemented by subclasses.
- The abstractmethod decorator defines that any subclass must implement the area() method.

Question 4

You are building a multilingual greeting system for a customer support chatbot. The function should greet the user in a specified language, but if no language is provided, it should default to Hindi. Similarly, if the user's name isn't known, it should default to "Guest". To do this, you want to define a function with default parameters for language and name.

Which of the following is a valid function definition that supports this?

- A) def greet(lang = "Hindi", name = "Guest"):
- B) def greet(name = "Guest", lang):
- C) def greet("English" = lang, "Guest" = name):
- D) def greet("Guest" == name, lang):

Correct Answer: A

Explanation:

- In Python, default parameter values are assigned using the assignment operator.
- Parameters with default values must come after those without defaults in the function definition.
- Options C and D are syntactically incorrect, and option B violates the rule about the order of parameters.
- Option A correctly defines a function greet with 2 default parameters. If the function is called without providing a value for the name parameter, it will default to "Guest".

Question 5

You are designing a plugin system for a game where each plugin adds a special power to the player. The outer() function generates a new plugin based on a power value (x) and returns a function that can apply this power to any input (y). You want to ensure that the function returned by outer() correctly retains the value of x even after the outer function has finished executing.

What is the output of the following code?

```
def outer(x):
    def inner(y):
        return x + y
    return inner

closure = outer(11)
print(closure(4))
```

- A) 15

- B) 10
- C) 5
- D) Error

Correct Answer: A
Explanation: This code demonstrates a closure. The outer() function returns the inner() function, which remembers the value of x. When we call outer(11), it returns inner with x=11. Then calling closure(4) is equivalent to inner(4), which returns 11 + 4 = 15.

Question 6

You are combining features from two classes — one logs messages to a database (B) and the other to a file (C). Both inherit from a common base class A. You create a new class D that inherits from both B and C, and you want to see which version of the foo() method gets called.

```
class A:
    def foo(self):
        print("A's foo()")

class B(A):
    def foo(self):
        print("B's foo()")

class C(A):
    def foo(self):
        print("C's foo()")

class D(B, C):
    pass

d = D()
```

d.foo()

What will be the output of the code above?

A) A's foo()

B) B's foo()

C) C's foo()

D) The program will raise an error because D does not implement foo().

Answer: B

Explanation:

- Python uses the Method Resolution Order (MRO) to determine the order in which methods are called.
- Since D inherits from B and C (i.e., class D(B, C)), and B appears first in the inheritance list, D will call B's foo() method first.
- Even though both B and C have a foo() method, the MRO determines that B's implementation should be used.

Question 7

A programmer is implementing a selection sort algorithm to sort an array. After which of the following pass (least pass of all such passes that satisfy this condition) does the array ([49, 16, 12, 92]) become sorted using selection sort?

A) 1st pass

B) 2nd pass

C) 3rd pass

D) 4th pass

Correct Answer: A

Explanation: After 1st pass itself the array will become sorted as minimum element will get swapped by element at 1st position of array.

Question 8

You are working as a programmer and using bubble sort to sort an array.

Question: What will be the state of the array ([37, 31, 12, 90]), after the 2nd pass of bubble sort?

A) [31, 37, 12, 90]

B) [12, 31, 37, 90]

C) [12, 37, 31, 90]

D) [31, 12, 37, 90]

Correct Answer: B

Explanation: After the first pass, the largest element (90) is at the end. After the second pass, the next largest element (37) is in its correct position.

Question 9

A user is designing a math library with a class Calculator that allows chaining of methods for addition, subtraction, and multiplication. Identify the output of the following code:

```
class Calculator:
    def __init__(self, value=0):
        self.value = value

    def add(self, number):
        self.value += number
        return  self

    def subtract(self, number):
        self.value -= number
        return  self

    def multiply(self, number):
        self.value *= number
        return  self

calc = Calculator()
result = calc.add(10).subtract(2).multiply(3).value
print(result)
```

(a) 24

(b) 30

(c) 36

(d) Error

Correct Option: (a)
Explanation: The add method adds 10 to the initial value of 0. The subtract method subtracts 2 from the result (10). The multiply method multiplies the result (8) by 3, yielding 24.

Question 10

A tech company tracks laptops sold by using a class Laptop. What will this code output?

```
class Laptop:
    brand = "Generic"
    def __init__(self, model):
        self.model = model



l1 = Laptop("XPS")

l2 = Laptop("MacBook")

print(l1.brand, l1.model)
```

(a) Generic XPS

(b) XPS Generic

(c) Error

(d) None

Correct Option: (a)

Explanation: brand is a class attribute shared across all instances, while model is an instance attribute unique to each object. The output correctly shows "Generic XPS."

Question 11

A pet store uses Python to represent animals. Both Dog and Cat classes have a method sound. What will be printed?

```
class Dog:

        def sound(self):

                return  "Bark"



class Cat:

        def sound(self):

                return  "Meow"



def animal_sound(animal):

                print(animal.sound())



d = Dog()

c = Cat()

animal_sound(d)

animal_sound(c)
```

(a) Bark Meow

(b) Meow Bark

(c) Error

(d) None

Correct Option: (a)

Explanation: Both classes have a sound method. The animal_sound function accepts any object with a sound method, demonstrating polymorphism.

Question 12

A bank maintains account balances securely. The Account class uses private attributes to store balance information. What will be the output of the following?

```
class Account:

    def __init__(self, balance):

        self.__balance = balance



    def get_balance(self):

        return  self.__balance



account = Account(1000)

print(account.get_balance())
```

(a) 1000

(b) Error

(c) None

(d) __balance

Correct Option: (a)

Explanation: The attribute __balance is private, but the get_balance method allows access. The print statement calls this method, correctly outputting 1000.

Question 13

A library tracks borrowed books. The base class Library has a method info. The child class Member overrides this method. What will be printed?

```
class Library:

    def info(self):

        return  "This is a library."



class Member(Library):

    def info(self):

        return  "This is a member."



m = Member()
```

```
print(m.info())
```

(a) This is a library.

(b) This is a member.

(c) Error

(d) None

Correct Option: (b)

Explanation: The Member class overrides the info method from Library. The m.info() call uses the overridden method in Member, printing "This is a member."

Question 14

Ramesh wants to calculate the area of rectangles. He writes a Python class Rectangle with a method calculate_area. What will the following code print?

```
class Rectangle:

    def __init__(self, length, width):

        self.length = length

        self.width = width



    def calculate_area(self):

        return  self.length * self.width
```

```
rect = Rectangle(5, 3)

print(rect.calculate_area())
```

(a) 8

(b) 15

(c) 10

(d) Error

Correct Option: (b)

Explanation: The calculate_area method multiplies length and width, returning 5 * 3 = 15. The print statement correctly displays this result.

Question 15

A zoo is building a system to categorize animals. A Bird class is created as a parent class, and a Parrot class inherits it. What will this code output?

```
class Bird:

    def __init__(self, species):

        self.species = species


class Parrot(Bird):

    def __init__(self, species, name):

        super().__init__(species)
```

```
        self.name = name


parrot = Parrot("Macaw", "Rio")
print(parrot.species, parrot.name)
```

(a) Macaw Rio

(b) Rio Macaw

(c) Error

(d) None

Correct Option: (a)

Explanation: The Parrot class calls the Bird class constructor using super(). The attributes species and name are initialized and accessed correctly.