Multiple-Choice Questions

Which of the following best describes the purpose of assertions in Python?

A) To ensure the code runs faster.

B) To check conditions that should hold true while running a program.

C) To print values for debugging purposes.

D) To stop program execution after every loop.

What is the main difference between black-box testing and white-box testing?

A) Black-box testing relies on understanding the implementation details, while white-box testing focuses on testing behavior from a specification perspective.

B) Black-box testing ignores the implementation, focusing on input-output behavior, while white-box testing tests based on the code's internal logic.

C) White-box testing is limited to unit tests, while black-box testing applies to integration tests.

D) There is no difference; they are used interchangeably.

Which type of test would be best for evaluating a function that performs complex mathematical calculations requiring high precision?

A) Performance tests

B) Unit tests

C) Boundary tests

D) Security tests

Coding Practice Problem:

Problem Description: You are given a function ***calculate_area(shape, dimension1, dimension2=None)***. The function calculates the area of a given shape. It supports:

- Circle: ***dimension1*** is the radius.
- Rectangle: ***dimension1*** is the length, and ***dimension2*** is the width.
- Square: ***dimension1*** is the side length.

Task:

- Write tests for this function using black-box testing. You do not know the implementation details, so you must rely solely on the input-output behavior described above.

Assumptions:

- The function will raise an exception if an invalid shape is provided.

(Implementation for the calculate_area function on the next page)

Here's the implementation for the ***`calculate_area`*** function, which can handle circles, rectangles, and squares based on the input parameters:

```
def calculate_area(shape, dimension1, dimension2=None):
    """
    Calculate the area of a circle, rectangle, or square.

    Parameters:
    - shape (str): The shape type ("circle", "rectangle", "square").
    - dimension1 (float): The first dimension (e.g., radius, length, or side length)
    - dimension2 (float, optional): The second dimension for rectangles (e.g., width

    Returns:
    - float: The calculated area.

    Raises:
    - ValueError: If the shape is not supported or inputs are invalid.
    """
    if shape == "circle":
        if dimension1 <= 0:
            raise ValueError("Radius must be positive.")
        return math.pi * dimension1 ** 2
    elif shape == "rectangle":
        if dimension1 <= 0 or (dimension2 is None or dimension2 <= 0):
            raise ValueError("Length and width must be positive.")
        return dimension1 * dimension2
    elif shape == "square":
        if dimension1 <= 0:
            raise ValueError("Side length must be positive.")
        return dimension1 ** 2
    else:
        raise ValueError("Unsupported shape type.")
```

Answers:

1. **B;**
2. **B;**
3. **B**

Example answer:

```
def test_calculate_area():

    # Test circle with radius 3 (expected area: 3.14159 * 3^2)

    assert round(calculate_area("circle", 3), 2) == 28.27, "Expected area for
    circle with radius 3 is 28.27"

    # Test rectangle with length 4 and width 5 (expected area: 4 * 5)

    assert calculate_area("rectangle", 4, 5) == 20, "Expected area for
    rectangle with length 4 and width 5 is 20"

    # Test square with side length 6 (expected area: 6^2)

    assert calculate_area("square", 6) == 36, "Expected area for square with
    side length 6 is 36"

    # Test invalid shape

    try:
        calculate_area("triangle", 3, 4)
        assert False, "Expected an exception for invalid shape"
    except ValueError:
        pass # Expected behavior

test_calculate_area()
```