Multiple-Choice Questions (1 Mark Each)

1. Which of the following lines of code will result in a float data type?

 a) x = 10 b) y = "5.0" c) z = 7 / 2 d) a = 15 % 4

Correct Answer: c) z = 7 / 2

Explanation: the "/" operator performs true division, which always returns a float, even if the result is a whole number. The other options result in an integer, a string, and an integer respectively.

2. What is the output of the following code snippet?

```
a = 20
b = 3
result = a // b + 5
print(result)
```

a) 11 b)11.0 c) 10.0 d) 12

Correct Answer: a) 11

Explanation: The "//" operator performs floor division, which truncates the result to an integer. 20 // 3 is 6. The addition 6 + 5 results in 11. The final output is an integer.

Descriptive Questions (5-7 Marks Each)

1. Explain the difference between a **return** statement and a **print** statement within a Python function. Provide a code example for each to illustrate your explanation.

Answer:

- A **return** statement is used to exit a function and return a value to the caller. The returned value can be stored in a variable or used in further calculations. A function can only return a single value (though that value can be a complex object like a list or tuple). Once a return statement is executed, the function's execution stops.
- A **print** statement is a built-in function that displays output to the console. It does not return any value from the function. Its sole purpose is to show information to the user.

Example:

```python
# Using a return statement
def add_numbers(x, y):
  return x + y

sum_result = add_numbers(10, 5)
print(f"The sum is: {sum_result}") # Output: The sum is: 15

# Using a print statement
def greet(name):
  print(f"Hello, {name}!")

greet("Alice") # Output: Hello, Alice!
# No value is returned, so you can't assign the result
```

2. Write a recursive Python function to calculate the factorial of a non-negative integer. Explain how the recursion works for a given example, such as factorial(4).

Answer:

- **Function:**

```python
def factorial(n):
  # Base case: Factorial of 0 is 1
  if n == 0:
    return 1
  # Recursive case: n! = n * (n-1)!
  else:
    return n * factorial(n - 1)
```

Explanation for factorial(4):

- factorial(4) is called. Since n is not 0, it returns 4 * factorial(3).
- factorial(3) is called. It returns 3 * factorial(2).
- factorial(2) is called. It returns 2 * factorial(1).
- factorial(1) is called. It returns 1 * factorial(0).
- factorial(0) is called. This is the **base case**, so it returns 1.

- The function calls begin to return. factorial(1) receives 1 and returns 1 * 1 = 1.
- factorial(2) receives 1 and returns 2 * 1 = 2.
- factorial(3) receives 2 and returns 3 * 2 = 6.
- Finally, factorial(4) receives 6 and returns 4 * 6 = 24.
- The final result, 24, is returned to the original caller.