

**School of Computer Science and Artificial Intelligence****Lab Assignment # 9.2**

---

<b>Program</b>	: B. Tech (CSE)
<b>Specialization</b>	: -
<b>Course Title</b>	: AI Assisted Coding
<b>Course Code</b>	: 23CS002PC304
<b>Semester</b>	: II
<b>Academic Session</b>	: 2025-2026
<b>Name of Student</b>	: Mohammed Hashir Sayam Hussain
<b>Enrollment No.</b>	: 2403A51L27
<b>Batch No.</b>	: 51
<b>Date</b>	: 06/02/26

---

**Submission Starts here****Screenshots:**

Task Description -1 (Documentation – Function Summary Generation)

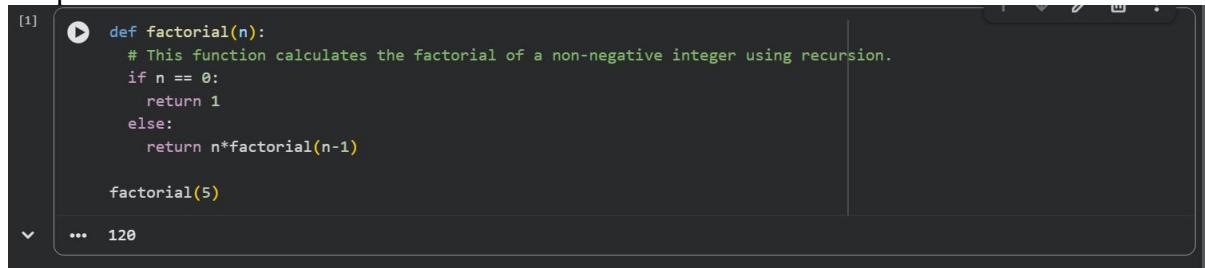
Code used:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
factorial(5)
```

Prompt:

Add a short summary describing the purpose of the function in the code

Output:



```
[1] def factorial(n):
    # This function calculates the factorial of a non-negative integer using recursion.
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)

    factorial(5)
    ...
    ... 120
```

A screenshot of a Jupyter Notebook cell. The cell number is [1]. The code defines a recursive factorial function. It includes a docstring explaining that it calculates the factorial of a non-negative integer using recursion. The function checks if n is 0, in which case it returns 1. Otherwise, it returns n multiplied by the result of calling itself with n-1. Below the definition, there is a call to factorial(5). To the right of the call, the output is shown as a large ellipsis (...), indicating the full output of the calculation.

Task Description -2 (Documentation – Logical Explanation for Conditions and Loops)

Code Used:

```
def numbers(n):
    if n > 0:
        result = "Positive"
    elif n < 0:
        result = "Negative"
    else:
        result = "Zero"
    return result
numbers(3)
```

Prompt:

explain only decision-making logic and loop behavior inside the code

Output:

```
[7] def numbers(n):
    # Decision-making logic: Determines if 'n' is positive, negative, or zero.
    # No loops are present in this function; execution flows through conditional branches.
    # Check if 'n' is positive
    if n > 0:
        result = "Positive"
    # If not positive, check if 'n' is negative
    elif n < 0:
        result = "Negative"
    # If neither positive nor negative, 'n' must be zero
    else:
        result = "Zero"
    return result
numbers(3)

... 'Positive'
```

### Task Description -3 (Documentation – File-Level Overview)

Code Used:

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

def square_list(numbers):
    return [n * n for n in numbers]

def main():
    print(factorial(5))
    print(square_list([1, 2, 3, 4]))

main()
```

Prompt:

Generate a high-level overview summarizing the overall purpose and functionality of the Code. add the summary at the beginning

Output:

```
[8] 0s
# This code defines two utility functions: `factorial` to calculate the factorial of a number,
# and `square_list` to square each element in a list. It then demonstrates their usage in a `main` function.

def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

def square_list(numbers):
    return [n * n for n in numbers]

def main():
    print(factorial(5))
    print(square_list([1, 2, 3, 4]))

main()
...
120
[1, 4, 9, 16]
```

#### Task Description -4 (Documentation – Refine Existing Documentation)

Code Used:

```
def calculate_discount(price, discount):
    # calculate
    if price > 0:
        # apply discount
        final_price = price - (price * discount / 100)
        return final_price
    else:
        # wrong price
        return 0
```

Prompt:

Rewrite the documentation to improve clarity and consistency.

Output:

```
[10] 0s
def calculate_discount(price, discount):
    """
    Calculates the final price after applying a discount.

    Args:
        price (float or int): The original price of the item.
        discount (float or int): The discount percentage to apply (e.g., 10 for 10%).

    Returns:
        float: The final price after applying the discount. Returns 0 if the original price is not positive.
    """
    if price > 0:
        # Apply the discount percentage to the original price
        final_price = price - (price * discount / 100)
        return final_price
    else:
        # Return 0 if the price is invalid (non-positive)
        return 0
```

## Task Description -5 (Documentation – Prompt Detail Impact Study)

Code Used:

```
def reverse_string(text):  
    return text[::-1]
```

Prompt - 1:

Write a short documentation comment explaining what this Python function does.

Output - 1:

```
[12] ✓ 0s ⏪ def reverse_string(text):  
    """  
        Reverses a given string.  
  
    Args:  
        text (str): The input string to be reversed.  
  
    Returns:  
        str: The reversed string.  
    """  
    return text[::-1]
```

Prompt - 2:

Write a clear and detailed documentation comment for the given Python function.

Output - 2:

```
[13] ✓ 0s ⏪ def reverse_string(text):  
    """  
        Reverses the given input string.  
  
        This function takes a string as input and returns a new string  
        with the characters in reverse order. It utilizes Python's string slicing  
        feature to achieve this efficiently.  
  
    Args:  
        text (str): The input string that needs to be reversed.  
  
    Returns:  
        str: A new string containing the reversed characters of the input string.  
  
    Examples:  
        >>> reverse_string("hello")  
        'olleh'  
        >>> reverse_string("Python")  
        'nohtyP'  
        >>> reverse_string("12345")  
        '54321'  
    """  
    return text[::-1]
```