

## AI ASSISTED CODING

Name: P.Swaran Raj  
Batch: 37

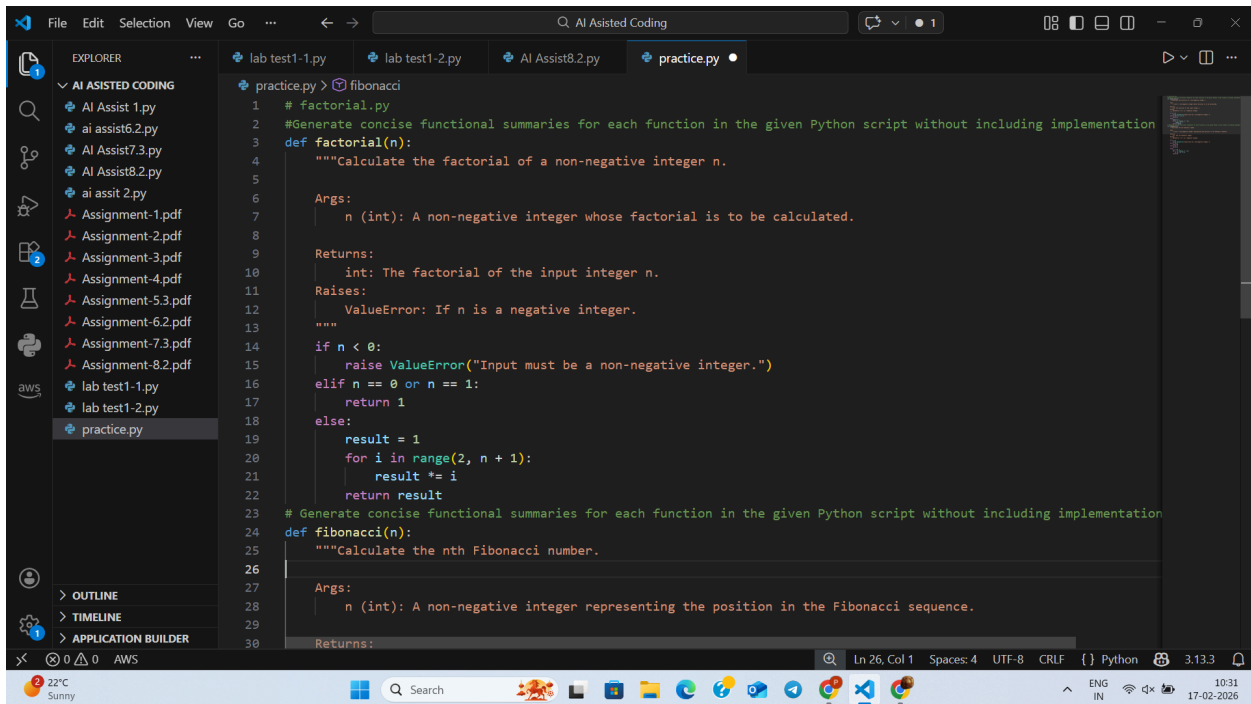
Roll.No: 2303A52247  
Assignment- 9.2

### Task Description -1 (Documentation – Function Summary Generation)

Use AI to generate concise functional summaries for each Python function in a given script.

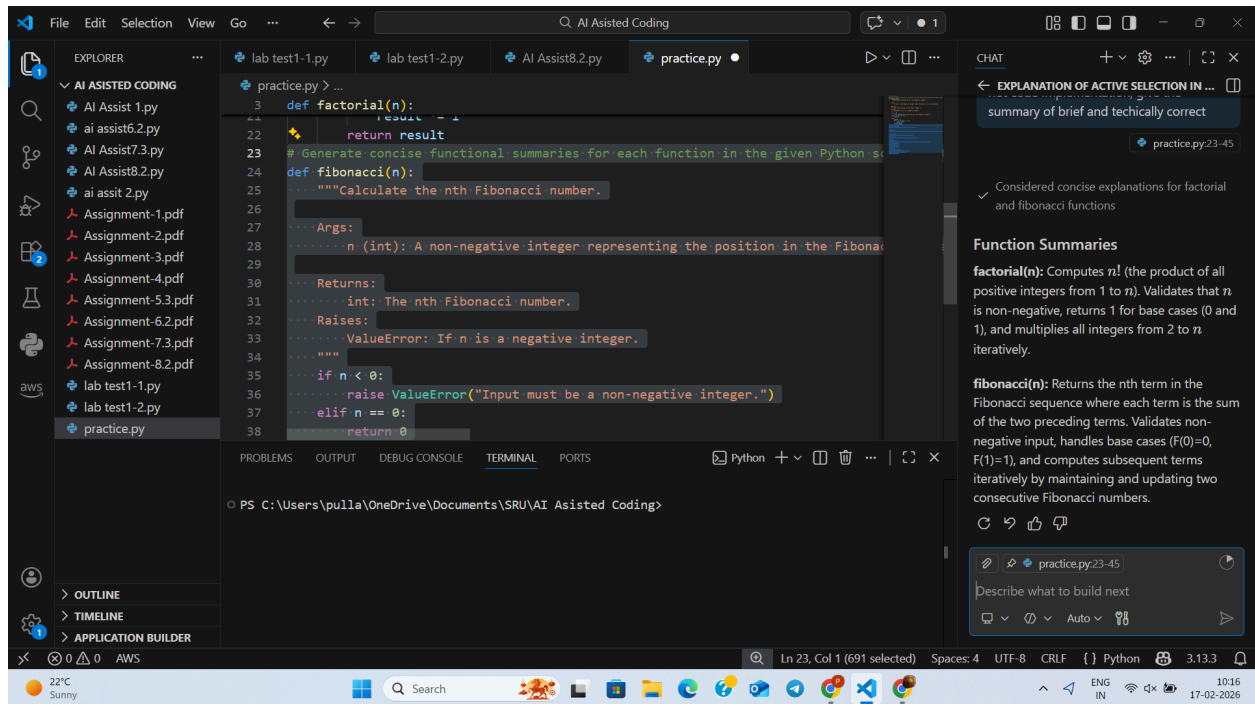
#### Prompt:

**“#Generate concise functional summaries for each function in the given Python script without including implementation details.”**



The screenshot shows a Visual Studio Code editor window with the 'AI ASSISTED CODING' extension active. The Explorer sidebar on the left lists several files, including 'practice.py'. The main editor area displays the code for 'practice.py', which contains two functions: 'factorial' and 'fibonacci'. The 'factorial' function has a docstring that has been generated by the AI, including an 'Args' section for 'n' and a 'Returns' section for 'int'. The 'fibonacci' function also has a docstring generated by the AI, including an 'Args' section for 'n' and a 'Returns' section. The code is as follows:

```
1 # factorial.py
2 #Generate concise functional summaries for each function in the given Python script without including implementation
3 def factorial(n):
4     """Calculate the factorial of a non-negative integer n.
5
6     Args:
7         n (int): A non-negative integer whose factorial is to be calculated.
8
9     Returns:
10         int: The factorial of the input integer n.
11
12     Raises:
13         ValueError: If n is a negative integer.
14     """
15     if n < 0:
16         raise ValueError("Input must be a non-negative integer.")
17     elif n == 0 or n == 1:
18         return 1
19     else:
20         result = 1
21         for i in range(2, n + 1):
22             result *= i
23         return result
24
25 # Generate concise functional summaries for each function in the given Python script without including implementation
26 def fibonacci(n):
27     """Calculate the nth Fibonacci number.
28
29     Args:
30         n (int): A non-negative integer representing the position in the Fibonacci sequence.
31
32     Returns:
```



## Function Summaries

**factorial(n):** Computes  $n!$  (the product of all positive integers from 1 to  $n$ ). Validates that  $n$  is non-negative, returns 1 for base cases (0 and 1), and multiplies all integers from 2 to  $n$  iteratively.

**fibonacci(n):** Returns the  $n$ th term in the Fibonacci sequence where each term is the sum of the two preceding terms. Validates non-negative input, handles base cases ( $F(0)=0$ ,  $F(1)=1$ ), and computes subsequent terms iteratively by maintaining and updating two consecutive Fibonacci numbers.

## Observation:

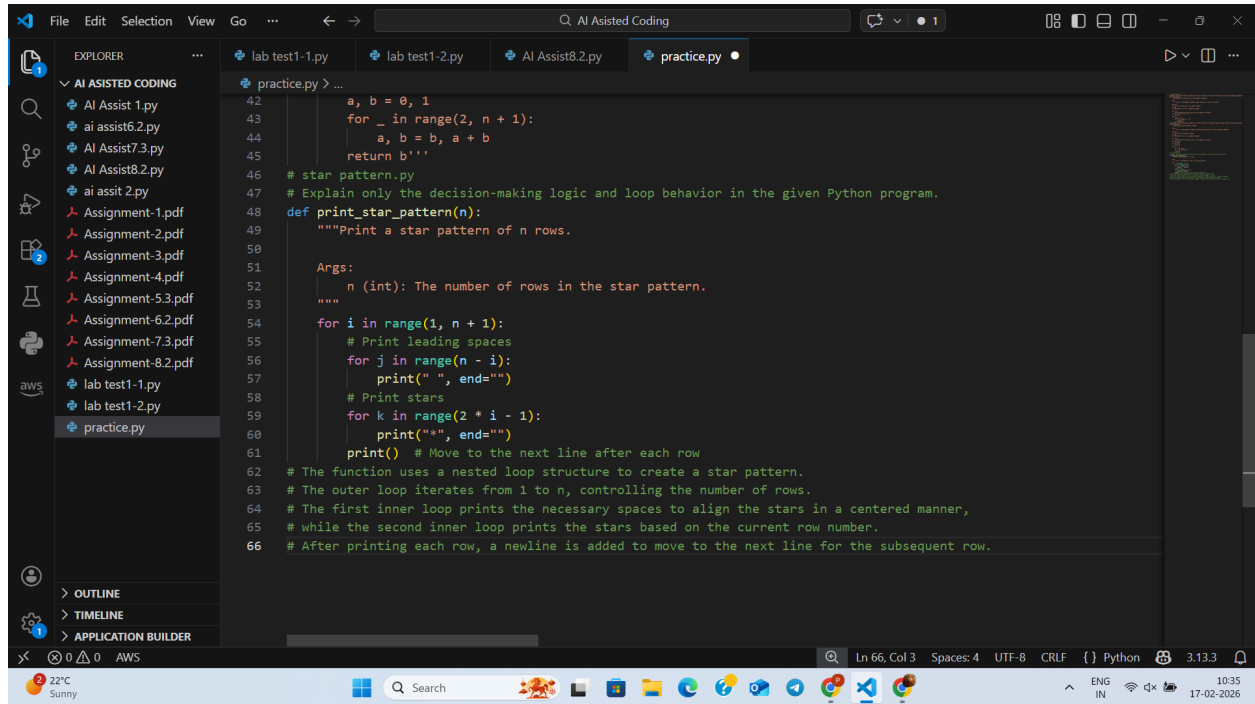
AI successfully generated brief and technically accurate summaries. The documentation improved readability and understanding of function roles.

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

Use AI to document the logic behind conditional statements and loops in a Python program.

## Prompt:

“#Generate concise functional summaries for each function in the given Python script without including implementation details.”



The screenshot shows the Visual Studio Code interface with a Python file named `practice.py` open. The file contains a Fibonacci sequence generator and a star pattern printer. An AI-generated summary is provided for the `print_star_pattern` function.

```
42 a, b = 0, 1
43 for _ in range(2, n + 1):
44     a, b = b, a + b
45     return b'''
46 # star pattern.py
47 # Explain only the decision-making logic and loop behavior in the given Python program.
48 def print_star_pattern(n):
49     """Print a star pattern of n rows.
50
51     Args:
52         n (int): The number of rows in the star pattern.
53     """
54     for i in range(1, n + 1):
55         # Print leading spaces
56         for j in range(n - i):
57             print(" ", end="")
58         # Print stars
59         for k in range(2 * i - 1):
60             print("*", end="")
61         print() # Move to the next line after each row
62 # The function uses a nested loop structure to create a star pattern.
63 # The outer loop iterates from 1 to n, controlling the number of rows.
64 # The first inner loop prints the necessary spaces to align the stars in a centered manner,
65 # while the second inner loop prints the stars based on the current row number.
66 # After printing each row, a newline is added to move to the next line for the subsequent row.
```

```

PS C:\Users\pulla\OneDrive\Documents\SRU\AI Asisted Coding> Python -m pydoc practice
Help on module practice:
Help on module practice:

NAME
    practice

DESCRIPTION
    # factorial.py
    #Generate concise functional summaries for each function in the given Python script without including implementation details.
    def factorial(n):
        """Calculate the factorial of a non-negative integer n.

        Args:
            n (int): A non-negative integer whose factorial is to be calculated.

        Returns:
            int: The factorial of the input integer n.

        Raises:
            ValueError: If n is a negative integer.
        """
        if n < 0:
            raise ValueError("Input must be a non-negative integer.")
        elif n == 0 or n == 1:
            return 1
        else:
            result = 1
            for i in range(2, n + 1):
                result *= i
            return result
    # Generate concise functional summaries for each function in the given Python script without including implementation details.
    def fibonacci(n):
        """Calculate the nth Fibonacci number.

        Args:
            n (int): A non-negative integer representing the position in the Fibonacci sequence.

        Returns:
            int: The nth Fibonacci number.

        Raises:
            ValueError: If n is a negative integer.
        """
        if n < 0:
            raise ValueError("Input must be a non-negative integer.")
        elif n == 0:
            return 0
        elif n == 1:
            return 1
        else:
            a, b = 0, 1
            for _ in range(2, n + 1):
                a, b = b, a + b
            return b

FUNCTIONS
    print_star_pattern(n)
        Print a star pattern of n rows.

        Args:
            n (int): The number of rows in the star pattern.

```

## Summary: Star Pattern Function

The `print_star_pattern(n)` function creates a centered star pyramid with `n` rows using three nested loops:

1. Outer loop (rows): Iterates from 1 to `n`, controlling the total number of rows.
2. First inner loop (spaces): Prints `n - i` leading spaces to center each row, decreasing as rows increase.

3. Second inner loop (stars): Prints  $2 * i - 1$  stars per row, expanding from 1 to  $2n - 1$  stars.

After each row completes, a newline is printed to move to the next line. The result is a centered, expanding star pyramid pattern.

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

Use AI to generate a high-level overview describing the functionality of an entire Python file.

#### Prompt:

**“#Generate a high-level overview describing the functionality of the entire Python file”**

```
"""This Python file contains three functions: factorial, fibonacci, and print_star_pattern.
The factorial function calculates the factorial of a non-negative integer,
while the fibonacci function computes the nth Fibonacci number.
Both functions include error handling for invalid input.
The print_star_pattern function generates a star pattern based on the specified number of rows,
utilizing nested loops to create a visually appealing output.
Each function is well-documented with clear descriptions of their purpose,
arguments, return values, and potential exceptions."""
```

#### Generated Documentation:

A brief module-level overview was added at the top of the file summarizing its purpose and core functionality.

#### Observation:

The file-level overview helps readers quickly understand the script before reading detailed code.

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

Use AI to improve clarity and consistency of existing documentation in Python code.

#### Prompt:

## “#Rewrite existing comments to improve clarity and consistency without changing technical meaning.”

```
# Rewrite existing comments to improve clarity and consistency without changing technical meaning.
def factorial(n):
    """Calculate the factorial of a non-negative integer n.

    Args:
        n (int): A non-negative integer whose factorial is to be calculated.

    Returns:
        int: The factorial of the input integer n.

    Raises:
        ValueError: If n is a negative integer.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
def fibonacci(n):
    """Calculate the nth Fibonacci number.
```

```
    """Calculate the nth Fibonacci number.

    Args:
        n (int): A non-negative integer representing the position in the Fibonacci sequence.

    Returns:
        int: The nth Fibonacci number.

    Raises:
        ValueError: If n is a negative integer.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")
    elif n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        a, b = 0, 1
        for _ in range(2, n + 1):
            a, b = b, a + b
        return b
def print_star_pattern(n):
```

```
def print_star_pattern(n):
    """Print a star pattern of n rows.

    Args:
        n (int): The number of rows in the star pattern.
    """
    for i in range(1, n + 1):
        # Print leading spaces to center the stars
        for j in range(n - i):
            print(" ", end="")
        # Print stars based on the current row number
        for k in range(2 * i - 1):
            print("*", end="")
        print() # Move to the next line after each row
```

### Generated Documentation:

AI refined unclear comments into structured and consistent explanations.

### Observation:

Improved documentation enhanced readability while preserving original logic.

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

Study the impact of prompt detail on AI-generated documentation quality.

### Prompt Used (1):

“#Document this Python function.”

```

# Document this Python function.
def calculate(a, b, operation):
    """Calculate the result of a binary operation on two numbers.

    Args:
        a (float): The first operand.
        b (float): The second operand.
        operation (str): The operation to perform. Must be one of "add", "sub", "mul", or "div".

    Returns:
        float: The result of the operation.

    Raises:
        ValueError: If the operation is not one of the supported operations.
        ZeroDivisionError: If attempting to divide by zero.
    """
    if operation == "add":
        return a + b
    elif operation == "sub":
        return a - b
    elif operation == "mul":
        return a * b
    elif operation == "div":

```

## Prompt Used (2):

“#Generate structured documentation including purpose, parameters, return value, and usage example.”



```
# Generate structured documentation including purpose, parameters, return value, and usage example.
def calculate(a, b, operation):
    """Calculate the result of a binary operation on two numbers.

    Args:
        a (float): The first operand.
        b (float): The second operand.
        operation (str): The operation to perform. Must be one of "add", "sub", "mul", or "div".

    Returns:
        float: The result of the operation.

    Raises:
        ValueError: If the operation is not one of the supported operations.
        ZeroDivisionError: If attempting to divide by zero.

    Example:
        >>> calculate(10, 5, 'add')
        15
        >>> calculate(10, 5, 'sub')
        5
        >>> calculate(10, 5, 'mul')
        50
        >>> calculate(10, 5, 'div')
        2.0
    """
```

```
        >>> calculate(10, 5, 'add')
        15
        >>> calculate(10, 5, 'sub')
        5
        >>> calculate(10, 5, 'mul')
        50
        >>> calculate(10, 5, 'div')
        2.0
    """
    if operation == "add":
        return a + b
    elif operation == "sub":
        return a - b
    elif operation == "mul":
        return a * b
    elif operation == "div":
        if b == 0:
            raise ZeroDivisionError("Cannot divide by zero.")
        return a / b
    else:
        raise ValueError("Unsupported operation. Use 'add', 'sub', 'mul', or 'div'.")
```

## Generated Documentation:

The brief prompt produced short comments.

The detailed prompt produced structured and comprehensive documentation.

**Observation:**

Detailed prompts resulted in clearer and more complete documentation.

Prompt specificity directly affects documentation quality.

**Conclusion:**

This lab demonstrates that AI-assisted documentation improves code readability, and detailed prompts significantly enhance documentation quality and clarity.