AI ASSISTED CODING LAB 10.2

Lab 10: Code Review and Quality: Using AI to improve code quality and readability

ENROLLMENT NO: 2403A52006

NAME: MOHAMMED MUSTAFA

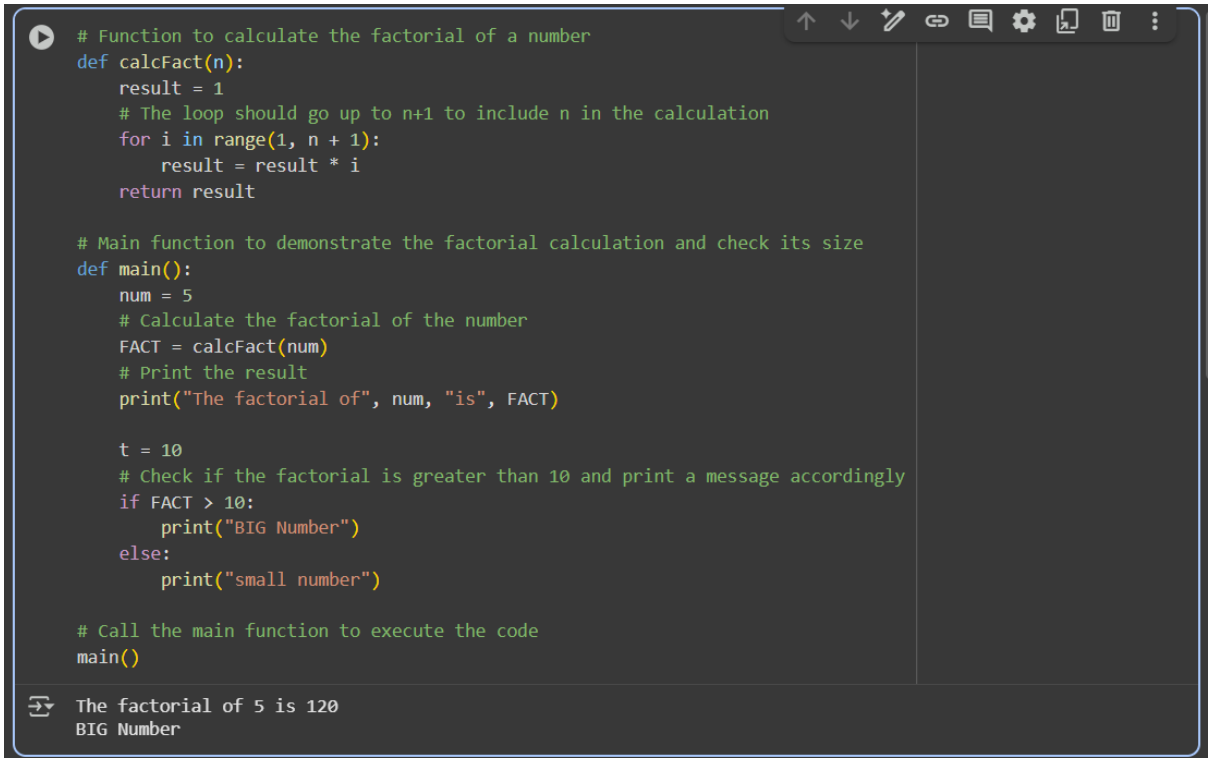
COURSE: CSM(AI/ML)

BATCH: 01

DATE: 7-10-2025

Task Description1: AI-Assisted Code Review (Basic Errors)
• Write python program as shown below.
• Use an AI assistant to review and suggest corrections

Used Prompt: In the given code check for corrections, add comment lines and output the corrected code.



```python
# Function to calculate the factorial of a number
def calcFact(n):
    result = 1
    # The loop should go up to n+1 to include n in the calculation
    for i in range(1, n + 1):
        result = result * i
    return result

# Main function to demonstrate the factorial calculation and check its size
def main():
    num = 5
    # Calculate the factorial of the number
    FACT = calcFact(num)
    # Print the result
    print("The factorial of", num, "is", FACT)

    t = 10
    # Check if the factorial is greater than 10 and print a message accordingly
    if FACT > 10:
        print("BIG Number")
    else:
        print("small number")

# Call the main function to execute the code
main()
```

```
The factorial of 5 is 120
BIG Number
```

This is the corrected code using multiple comment lines and it has been made much more readable for a beginner.

Task Description2: Automatic Inline Comments

• Write the Python code for Fibonacci as shown below and execute.

• Ask AI to improve variable names, add comments, and apply PEP8 formatting
(cleaned up).

• Students evaluate which suggestions improve readability most. One

Used Prompt: In the given code improve variable names, add comments, and apply PEP8 formatting and make
it more readable.

```python
# Function to generate Fibonacci series up to a given number
def generate_fibonacci(limit):
    # Initialize the first two numbers of the series
    first_num = 0
    second_num = 1
    # Initialize the next number in the series
    next_num = 2

    # Initialize the list to store the Fibonacci series
    fibonacci_series = [first_num, second_num]

    # Generate the series up to the limit
    while next_num <= limit:
        # Calculate the next Fibonacci number
        sum_of_previous_two = first_num + second_num
        # Append the calculated number to the series
        fibonacci_series.append(sum_of_previous_two)
        # Update the first and second numbers for the next iteration
        first_num = second_num
        second_num = sum_of_previous_two
        # Increment the next number counter
        next_num = next_num + 1

    return fibonacci_series
```

```python
# Main function to demonstrate the Fibonacci series generation
def main():
    # Set the limit for the Fibonacci series
    limit_number = 10
    # Generate the Fibonacci series
    fib_series = generate_fibonacci(limit_number)
    # Print the generated series
    print("Fibonacci series up to", limit_number, ":", fib_series)

# Call the main function to execute the code
main()
```

```
Fibonacci series up to 10 : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Now, this code has been cleaned up. Variable names are changed, comments added and PEP8 formatting added

Task Description3:

• Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply,
divide).
• Incorporate manual docstring in code with NumPy Style
• Use AI assistance to generate a module-level docstring + individual function
docstrings.
• Compare the AI-generated docstring with your manually written one

Used Prompt:

Write a python script with 4 functions (calculator: add, subtract, multiply, divide) incorporate manual docstring in code with NumPy style. Take the input from the user.

```python
def add(a, b):
    """
    Adds two numbers.

    Parameters
    ----------
    a : float or int
        The first number.
    b : float or int
        The second number.

    Returns
    -------
    float or int
        The sum of the two numbers.
    """
    return a + b
```

```python
def subtract(a, b):
    """
    Subtracts the second number from the first.

    Parameters
    ----------
    a : float or int
        The first number.
    b : float or int
        The second number.

    Returns
    -------
    float or int
        The difference between the two numbers.
    """
    return a - b

def multiply(a, b):
    """
    Multiplies two numbers.

    Parameters
    ----------
    a : float or int
        The first number.
    b : float or int
        The second number.
```

```
        Returns
        -------
        float or int
            The product of the two numbers.
        """
        return a * b

    def divide(a, b):
        """
        Divides the first number by the second.

        Parameters
        ----------
        a : float or int
            The first number (dividend).
        b : float or int
            The second number (divisor).

        Returns
        -------
        float or int
            The quotient of the division.

        Raises
        ------
        ZeroDivisionError
            If the divisor (b) is zero.
        """
```

```
        if b == 0:
            raise ZeroDivisionError("Cannot divide by zero")
        return a / b

    # Take input from the user
    try:
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))

        # Perform calculations and display results
        print(f"{num1} + {num2} = {add(num1, num2)}")
        print(f"{num1} - {num2} = {subtract(num1, num2)}")
        print(f"{num1} * {num2} = {multiply(num1, num2)}")
        print(f"{num1} / {num2} = {divide(num1, num2)}")

    except ValueError:
        print("Invalid input. Please enter numbers only.")
    except ZeroDivisionError:
        print("Error: Cannot divide by zero.")
    except Exception as e:
        print(f"An error occurred: {e}")
```

```
Enter the first number: 5
Enter the second number: 49
5.0 + 49.0 = 54.0
5.0 - 49.0 = -44.0
5.0 * 49.0 = 245.0
5.0 / 49.0 = 0.10204081632653061
```