AI ASSISTED CODING

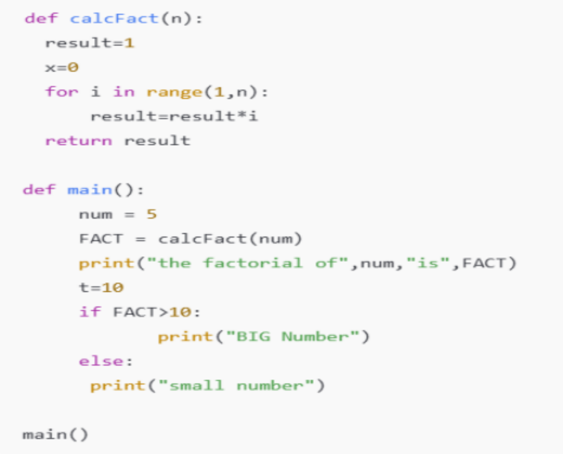
ASSIGNMENT-10.4

E.RAVALI

2503A51L18

TASK-1: AI-Assisted Code Review (Basic Errors)

- Write python program as shown below.
- Use an AI assistant to review and suggest corrections



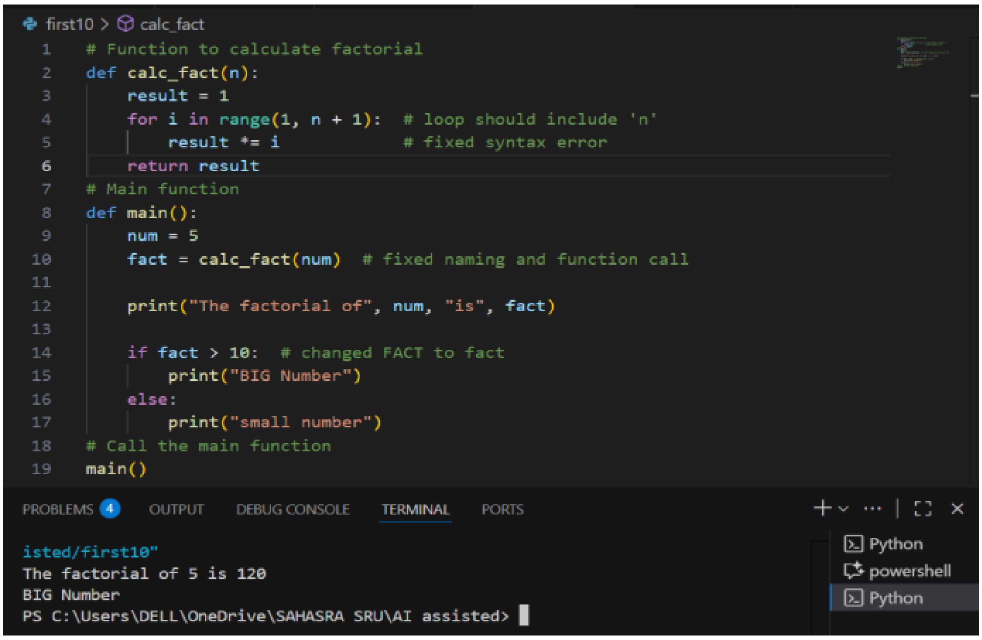
```
def calcFact(n):
  result=1
  x=0
  for i in range(1,n):
      result=result*i
  return result

def main():
     num = 5
     FACT = calcFact(num)
     print("the factorial of",num,"is",FACT)
     t=10
     if FACT>10:
            print("BIG Number")
     else:
      print("small number")

main()
```

PROMPT:

- Review the above code and suggest corrections. rewrite the correct code with comments.



```
# Function to calculate factorial
def calc_fact(n):
    result = 1
    for i in range(1, n + 1):  # loop should include 'n'
        result *= i            # fixed syntax error
    return result
# Main function
def main():
    num = 5
    fact = calc_fact(num)  # fixed naming and function call

    print("The factorial of", num, "is", fact)

    if fact > 10:  # changed FACT to fact
        print("BIG Number")
    else:
        print("small number")
# Call the main function
main()
```

```
isted/first10"
The factorial of 5 is 120
BIG Number
PS C:\Users\DELL\OneDrive\SAHASRA SRU\AI assisted> 
```

TASK-2 :Automatic Inline Comments

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

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

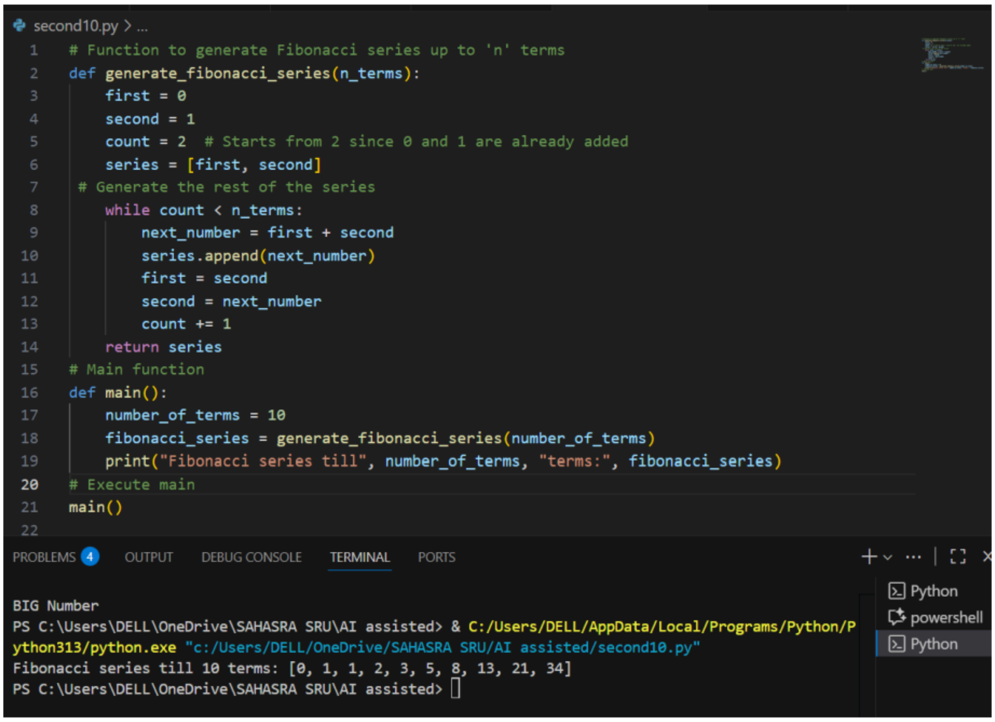
```
def f1(xX):
   a=0
   b=1
   c=2
   Zz=[a,b]
   while c<=xX:
      d=a+b
      Zz.append(d)
      a=b
      b=d
      c=c+1
   return Zz

def m():
     NN=10
     ans=f1(NN)
     print("fib series till",NN,":",ans)

m()
```

PROMPT:

- USE AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up) for the above python code.



```
# Function to generate Fibonacci series up to 'n' terms
def generate_fibonacci_series(n_terms):
    first = 0
    second = 1
    count = 2  # Starts from 2 since 0 and 1 are already added
    series = [first, second]
 # Generate the rest of the series
    while count < n_terms:
        next_number = first + second
        series.append(next_number)
        first = second
        second = next_number
        count += 1
    return series
# Main function
def main():
    number_of_terms = 10
    fibonacci_series = generate_fibonacci_series(number_of_terms)
    print("Fibonacci series till", number_of_terms, "terms:", fibonacci_series)
# Execute main
main()
```

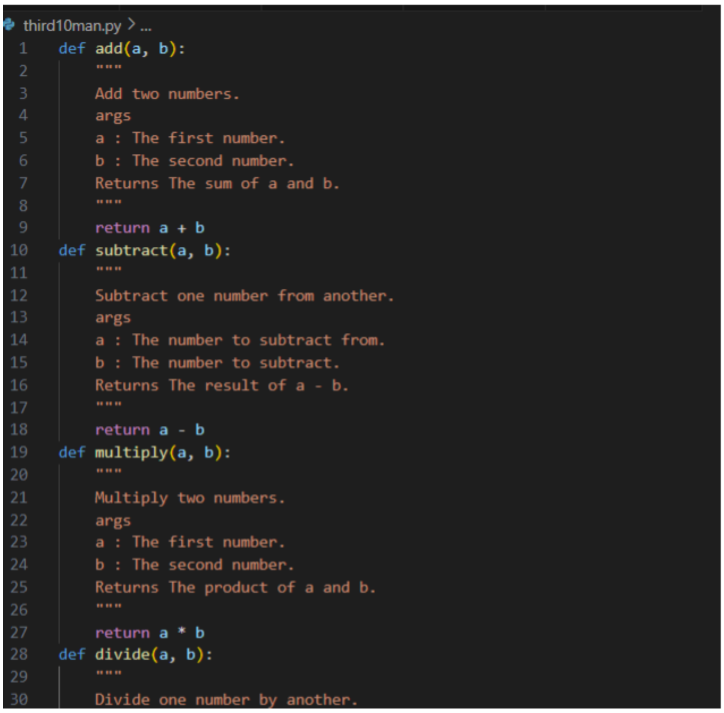
```
BIG Number
PS C:\Users\DELL\OneDrive\SAHASRA SRU\AI assisted> & C:/Users/DELL/AppData/Local/Programs/Python/P
ython313/python.exe "c:/Users/DELL/OneDrive/SAHASRA SRU/AI assisted/second10.py"
Fibonacci series till 10 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
PS C:\Users\DELL\OneDrive\SAHASRA SRU\AI assisted> 
```

Task Description#3

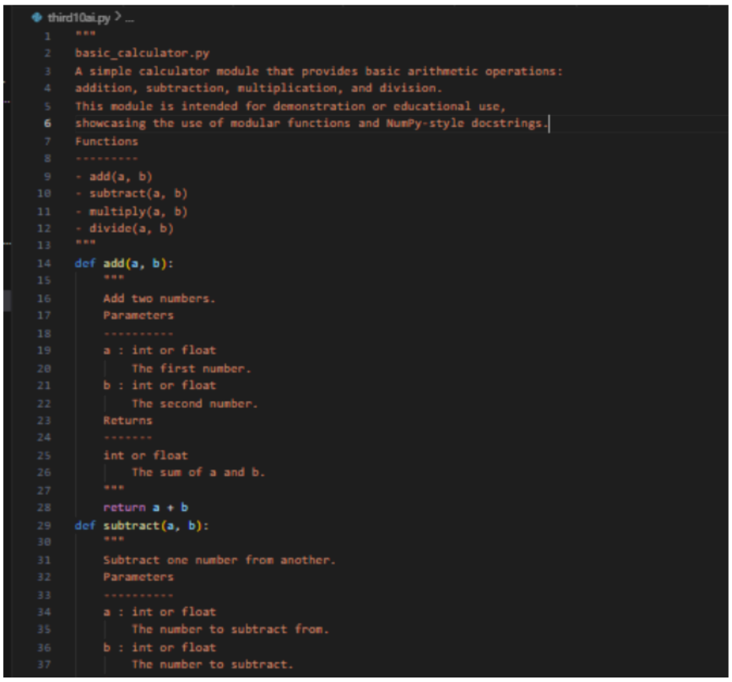
- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual docstring in code with NumPy Style

PROMPT:

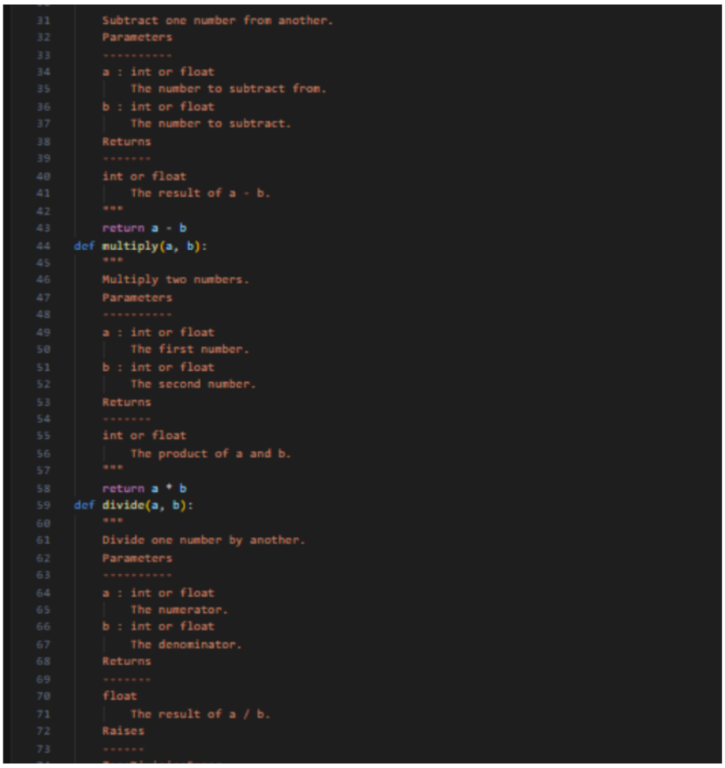
- Use AI assistance to generate a module-level docstring + individual function docstrings.
- Compare the AI-generated docstring with your manually written one.



```python
def add(a, b):
    """
    Add two numbers.
    args
    a : The first number.
    b : The second number.
    Returns The sum of a and b.
    """
    return a + b
def subtract(a, b):
    """
    Subtract one number from another.
    args
    a : The number to subtract from.
    b : The number to subtract.
    Returns The result of a - b.
    """
    return a - b
def multiply(a, b):
    """
    Multiply two numbers.
    args
    a : The first number.
    b : The second number.
    Returns The product of a and b.
    """
    return a * b
def divide(a, b):
    """
    Divide one number by another.
```



```python
"""
basic_calculator.py
A simple calculator module that provides basic arithmetic operations:
addition, subtraction, multiplication, and division.
This module is intended for demonstration or educational use,
showcasing the use of modular functions and NumPy-style docstrings.
Functions
---------
- add(a, b)
- subtract(a, b)
- multiply(a, b)
- divide(a, b)
"""
def add(a, b):
    """
    Add two numbers.
    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.
    Returns
    -------
    int or float
        The sum of a and b.
    """
    return a + b
def subtract(a, b):
    """
    Subtract one number from another.
    Parameters
    ----------
    a : int or float
        The number to subtract from.
    b : int or float
        The number to subtract.
```



```
    Subtract one number from another.
    Parameters
    ----------
    a : int or float
        The number to subtract from.
    b : int or float
        The number to subtract.
    Returns
    -------
    int or float
        The result of a - b.
    """
    return a - b
def multiply(a, b):
    """
    Multiply two numbers.
    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.
    Returns
    -------
    int or float
        The product of a and b.
    """
    return a * b
def divide(a, b):
    """
    Divide one number by another.
    Parameters
    ----------
    a : int or float
        The numerator.
    b : int or float
        The denominator.
    Returns
    -------
    float
        The result of a / b.
    Raises
    ------
```

OBSERVATION:

AI-assisted docstrings offer:

- Professional structure
- Better readability
- Tool compatibility
- Reusability in teams, APIs, and documentation websites