

AI ASSISTED CODING

ASSIGNMENT-10.4

NAME:shaik faheem

ROLL NO:2503A51L39

BATCH: 24BTCAICSB20

Task Description#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()
```

Expected Outcome#1: Students need to submit corrected code with comments.

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

Corrections made in the code:

The loop for i in range(1, n): is incorrect. It should be range(1, n + 1) to include n itself.

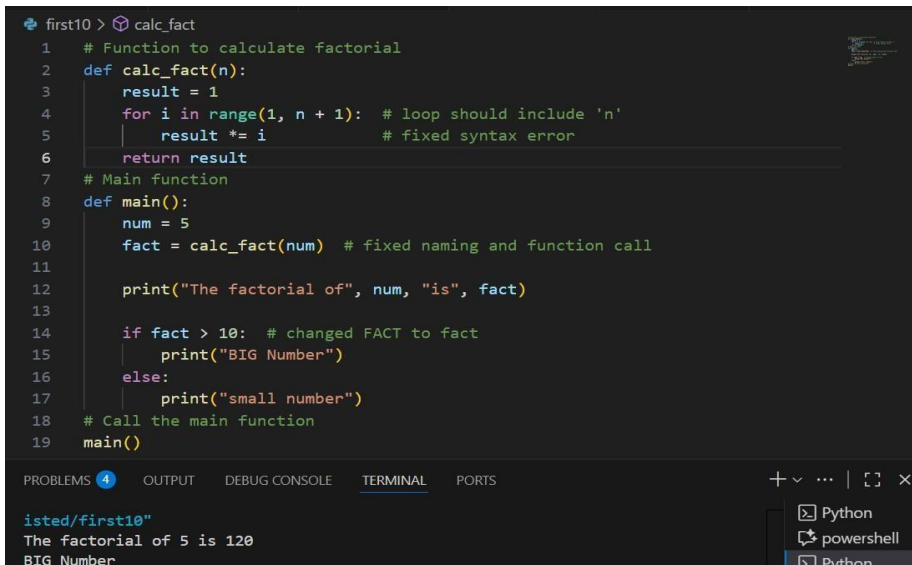
Variable x = 0 in calcFact () is never used.

In main (), num is defined, but passed to calcFact () as num, yet the function is called before num is declared as a parameter.

The variable t = 10 is declared but unused.

The if FACT > 10: is valid, but t being declared above has no effect.

Use proper naming conventions (e.g., `fact` instead of `FACT`). Ensure consistent indentation and spacing for readability.



```
first10 > calc_fact
1 # Function to calculate factorial
2 def calc_fact(n):
3     result = 1
4     for i in range(1, n + 1): # loop should include 'n'
5         result *= i          # fixed syntax error
6     return result
7
8 # Main function
9 def main():
10     num = 5
11     fact = calc_fact(num) # fixed naming and function call
12
13     print("The factorial of", num, "is", fact)
14
15     if fact > 10: # changed FACT to fact
16         print("BIG Number")
17     else:
18         print("small number")
19
20 # Call the main function
21 main()

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
istd/first10"
The factorial of 5 is 120
BIG Number
```

Task Description#2 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.

```
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()
```

Expected Output#2: Clean format python code with much readability.

PROMPT: improve variable names, add comments, and apply PEP8 formatting (cleaned up) for the above python code

```

second10.py > ...
1  # Function to generate Fibonacci series up to 'n' terms
2  def generate_fibonacci_series(n_terms):
3      first = 0
4      second = 1
5      count = 2 # Starts from 2 since 0 and 1 are already added
6      series = [first, second]
7      # Generate the rest of the series
8      while count < n_terms:
9          next_number = first + second
10         series.append(next_number)
11         first = second
12         second = next_number
13         count += 1
14     return series
15 # Main function
16 def main():
17     number_of_terms = 10
18     fibonacci_series = generate_fibonacci_series(number_of_terms)
19     print("Fibonacci series till", number_of_terms, "terms:", fibonacci_series)
20 # Execute main
21 main()
22

```

```

python313/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]

```

Key Improvements:

- Replacing `a`, `b`, `c`, `z` with `first`, `second`, `count`, `series` makes the logic self-explanatory.
- Function name `generate_fibonacci_series` tells you exactly what the function does.
- Original used `while c <= xx`, which actually **generates `xx + 1` terms**, but your corrected code uses `while count < n_terms`, which properly limits the list to **exactly `n_terms` values**.
- Each logical block now has a comment — helps learners or future readers quickly understand what the code is doing.
- PEP8 style: consistent spacing, indentation, and line length.
- Your code looks professional and beginner-friendly now.

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

MANUAL CODE:

```
third10man.py > ...
1  def add(a, b):
2      """
3      Add two numbers.
4      args
5      a : The first number.
6      b : The second number.
7      Returns The sum of a and b.
8      """
9      return a + b
10 def subtract(a, b):
11     """
12     Subtract one number from another.
13     args
14     a : The number to subtract from.
15     b : The number to subtract.
16     Returns The result of a - b.
17     """
18     return a - b
19 def multiply(a, b):
20     """
21     Multiply two numbers.
22     args
23     a : The first number.
24     b : The second number.
25     Returns The product of a and b.
26     """
27     return a * b
28 def divide(a, b):
29     """
30     Divide one number by another.
31     args
32     a : The numerator.
33     b : The denominator.
34     Returns The result of a / b.
35     Raises ZeroDivisionError
36     |   If b is zero.
37     """
38     if b == 0:
39         raise ZeroDivisionError("Cannot divide by zero.")
40     return a / b
41
```

```

third10ai.py > _
1  """
2  basic_calculator.py
3  A simple calculator module that provides basic arithmetic operations:
4  addition, subtraction, multiplication, and division.
5  This module is intended for demonstration or educational use,
6  showcasing the use of modular functions and NumPy-style docstrings.
7  Functions
8  """
9
10 - add(a, b)
11 - subtract(a, b)
12 - multiply(a, b)
13 - divide(a, b)
14 """
15
16 def add(a, b):
17     """
18     Add two numbers.
19     Parameters
20     """
21     """
22     a : int or float
23     | The first number.
24     b : int or float
25     | The second number.
26     Returns
27     """
28     """
29     int or float
30     | The sum of a and b.
31     """
32     return a + b
33
34 def subtract(a, b):
35     """
36     Subtract one number from another.
37     Parameters
38     """
39     """
40     a : int or float
41     | The number to subtract from.
42     b : int or float
43     | The number to subtract.
44     Returns
45     """
46     """
47     int or float
48     | The result of a - b.
49     """
50     return a - b
51
52 def multiply(a, b):
53     """
54     Multiply two numbers.
55     Parameters
56     """
57     """
58     a : int or float
59     | The first number.
60     b : int or float
61     | The second number.
62     Returns
63     """
64     """
65     int or float
66     | The product of a and b.
67     """
68     return a * b
69
70 def divide(a, b):
71     """
72     Divide one number by another.
73     Parameters
74     """
75     """
76     a : int or float
77     | The numerator.
78     b : int or float
79     | The denominator.
80     Returns
81     """
82     """
83     float
84     | The result of a / b.
85     Raises
86     """
87     """
88     ZeroDivisionError
89     | If the denominator is zero.
90     """
91     if b == 0:
92         raise ZeroDivisionError("Cannot divide by zero.")
93     return a / b
94

```

OBSERVATION:

- **AI-assisted docstrings** offer:
 - Professional structure
 - Better readability
 - Tool compatibility
 - Reusability in teams, APIs, and documentation websites
- **Manual comments** are a good starting point for beginners, but lack structure and formal style, which may be insufficient for large or shared codebases.

The **manual comments** provide basic understanding but lack structure, standard formatting, and detailed type information. In contrast, the **AI-assisted docstrings** follow the **NumPy documentation standard**, offering clearer structure, better readability, and compatibility with professional tools. This makes them more suitable for collaborative and scalable projects.