AI ASSISTANT CODING

Lab Assignment 1.5

Name: P.Vyshnavi

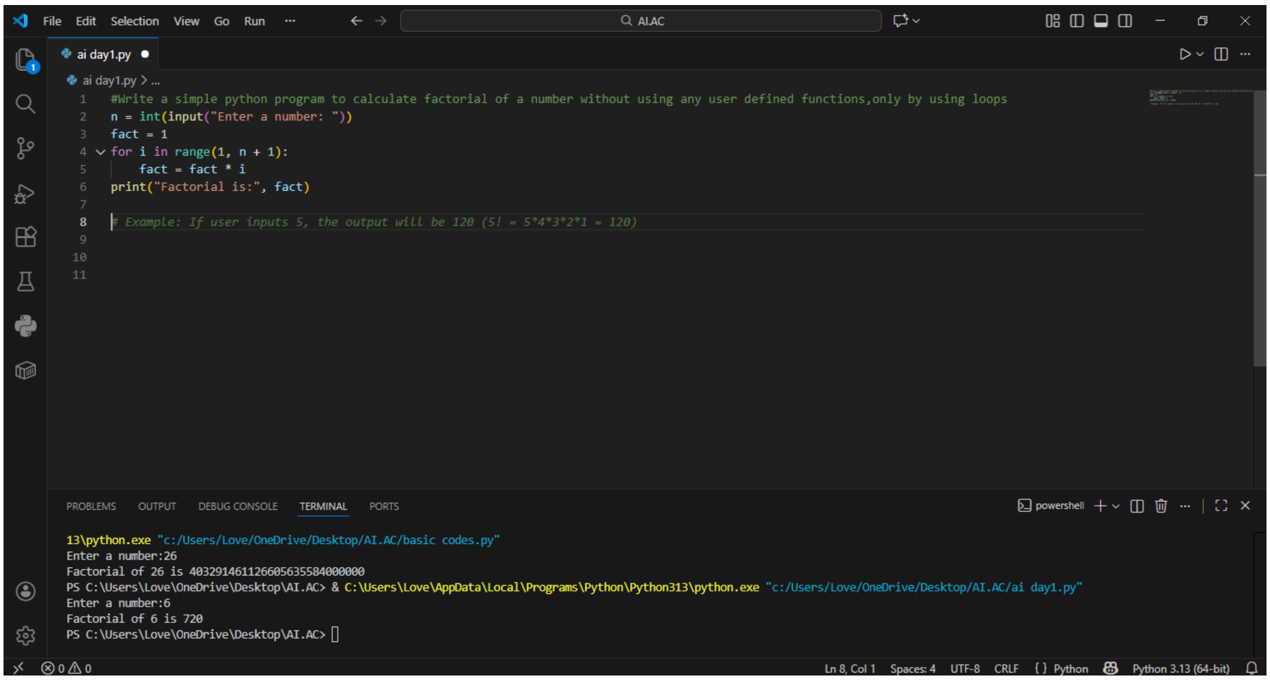
Ht.No: 2403A51L34

Batch : 52

Task 1: AI-Generated Logic Without Modularization (Greatest of three numbers)

Prompt Used:" Write a python program to find the greatest of three numbers with comments"

Task 2: AI Code Optimization & Cleanup Original Code:

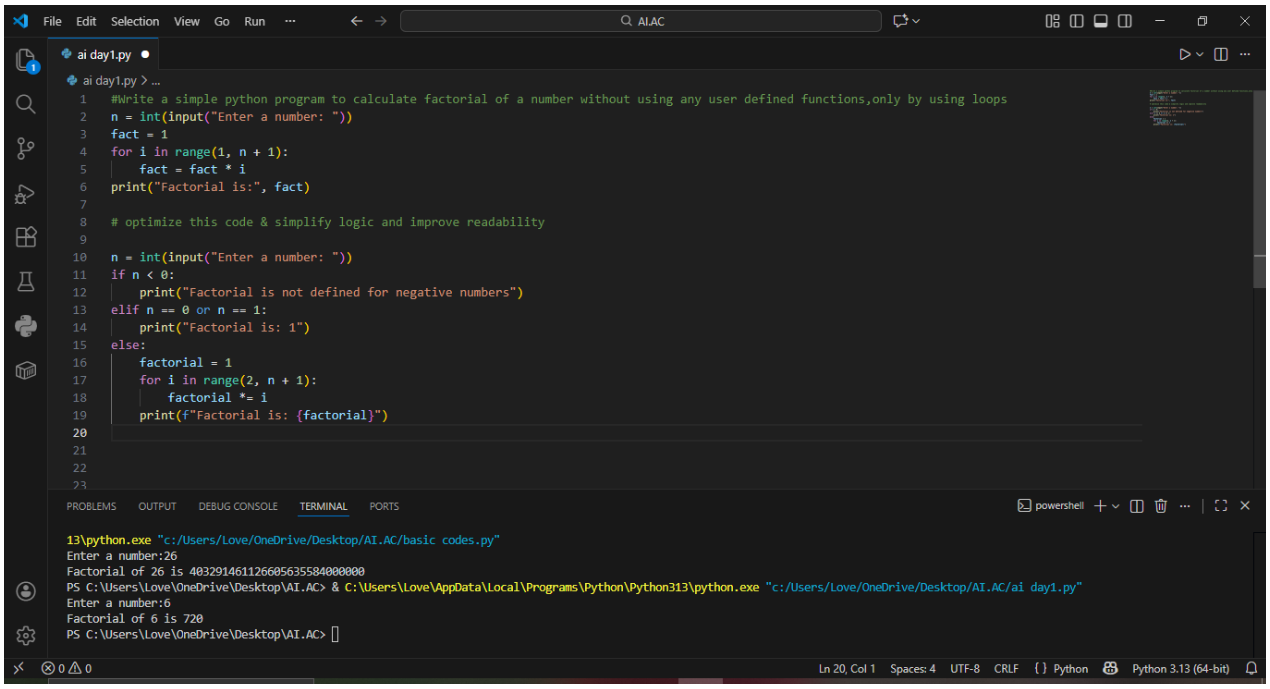


```python
#Write a simple python program to calculate factorial of a number without using any user defined functions,only by using loops
n = int(input("Enter a number: "))
fact = 1
for i in range(1, n + 1):
    fact = fact * i
print("Factorial is:", fact)

# Example: If user inputs 5, the output will be 120 (5! = 5*4*3*2*1 = 120)
```

```
13\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/basic codes.py"
Enter a number:26
Factorial of 26 is 403291461126605635584000000
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/ai day1.py"
Enter a number:6
Factorial of 6 is 720
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

Prompt Used: "optimize this code & simplify logic and improve readability"



```python
#Write a simple python program to calculate factorial of a number without using any user defined functions,only by using loops
n = int(input("Enter a number: "))
fact = 1
for i in range(1, n + 1):
    fact = fact * i
print("Factorial is:", fact)

# optimize this code & simplify logic and improve readability

n = int(input("Enter a number: "))
if n < 0:
    print("Factorial is not defined for negative numbers")
elif n == 0 or n == 1:
    print("Factorial is: 1")
else:
    factorial = 1
    for i in range(2, n + 1):
        factorial *= i
    print(f"Factorial is: {factorial}")
```

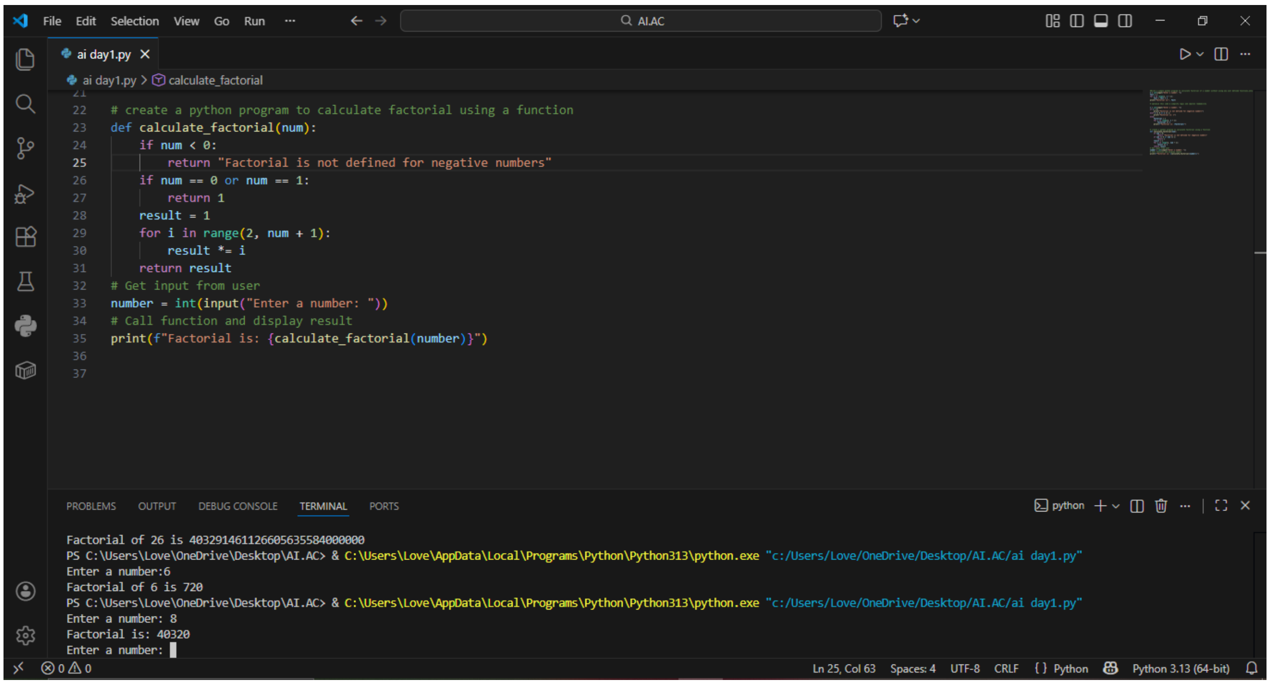
```
13\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/basic codes.py"
Enter a number:26
Factorial of 26 is 403291461126605635584000000
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/ai day1.py"
Enter a number:6
Factorial of 6 is 720
PS C:\Users\Love\OneDrive\Desktop\AI.AC> 
```

Code is cleaner and easier to maintain

The optimized version improves clarity, maintainability, and readability without affecting performance.

Task 3: Modular Design Using AI Assistance (Factorial with Functions)

Prompt Used: " Create a python program to calculate factorial using a function"



```python
# create a python program to calculate factorial using a function
def calculate_factorial(num):
    if num < 0:
        return "Factorial is not defined for negative numbers"
    if num == 0 or num == 1:
        return 1
    result = 1
    for i in range(2, num + 1):
        result *= i
    return result
# Get input from user
number = int(input("Enter a number: "))
# Call function and display result
print(f"Factorial is: {calculate_factorial(number)}")
```

```
Factorial of 26 is 403291461126605635584000000
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/ai day1.py"
Enter a number:6
Factorial of 6 is 720
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/ai day1.py"
Enter a number: 8
Factorial is: 40320
Enter a number: 
```

Using functions improves reusability because the same logic can be called multiple times.
It also improves readability and debugging.
Modular code is easier to maintain in large projects.

Task 4: Comparative Analysis

Procedural vs Modular AI Code

Criteria	**Without Function**	**With Function**
Logic Clarity	Moderate	High
Reusability	No	Yes
Debugging	Difficult	Easy
Large Projects	Not suitable	Highly suitable
AI Dependency Risk	Higher	Lower

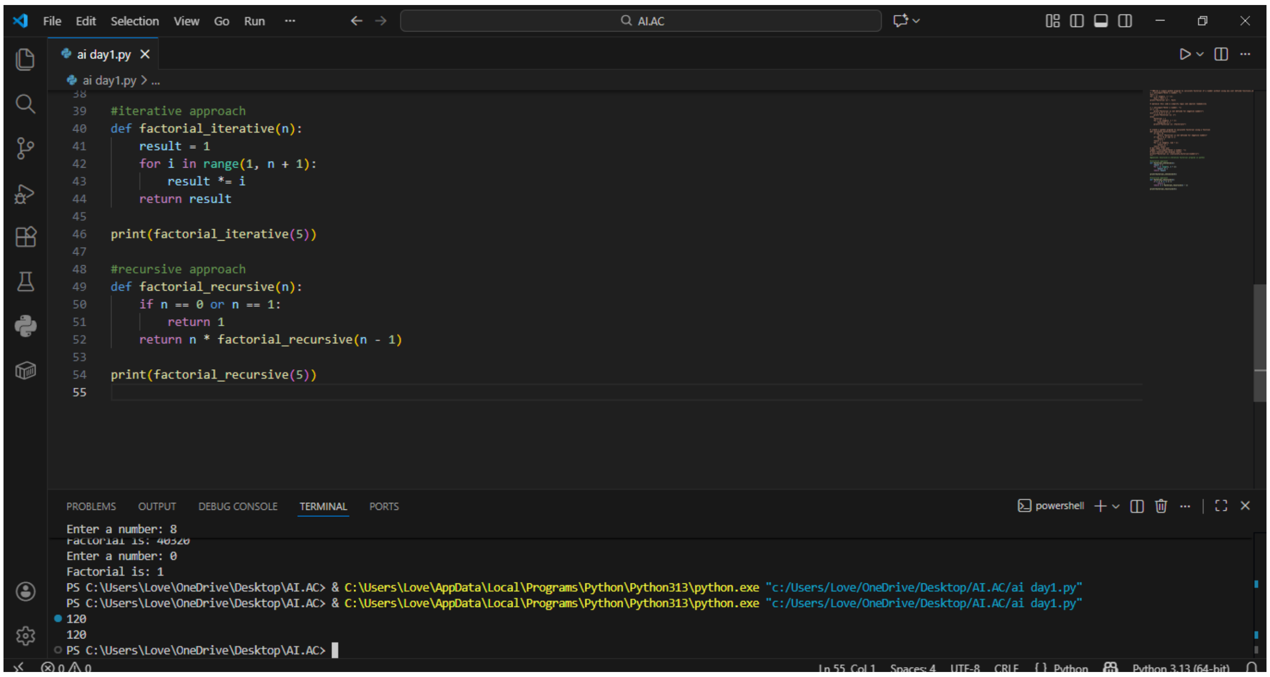
Conclusion:

Function-based design is more scalable and suitable for real-world applications.

Procedural code is only suitable for small scripts

Task 5: Iterative vs Recursive AI Code

Prompt Used: “Generate iterative and recursive factorial programs in Python”



```python
#iterative approach
def factorial_iterative(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

print(factorial_iterative(5))

#recursive approach
def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial_recursive(n - 1)

print(factorial_recursive(5))
```

```
Enter a number: 8
Factorial is: 40320
Enter a number: 0
Factorial is: 1
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/ai day1.py"
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/ai day1.py"
120
120
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

Execution Flow Explanation

- Iterative version uses loops
- Recursive version uses function calls
- Recursive calls stack memory

Comparison:

Aspect	**Iterative**	**Recursive**
Readability	Easy	Moderate
Stack Usage	No	Yes
Performance	Better	Slightly slower
Recommendation	Preferred	Avoid for large inputs