

AI ASSISTANT CODING

Lab Assignment 1.5

Name: P.Vyshnavi

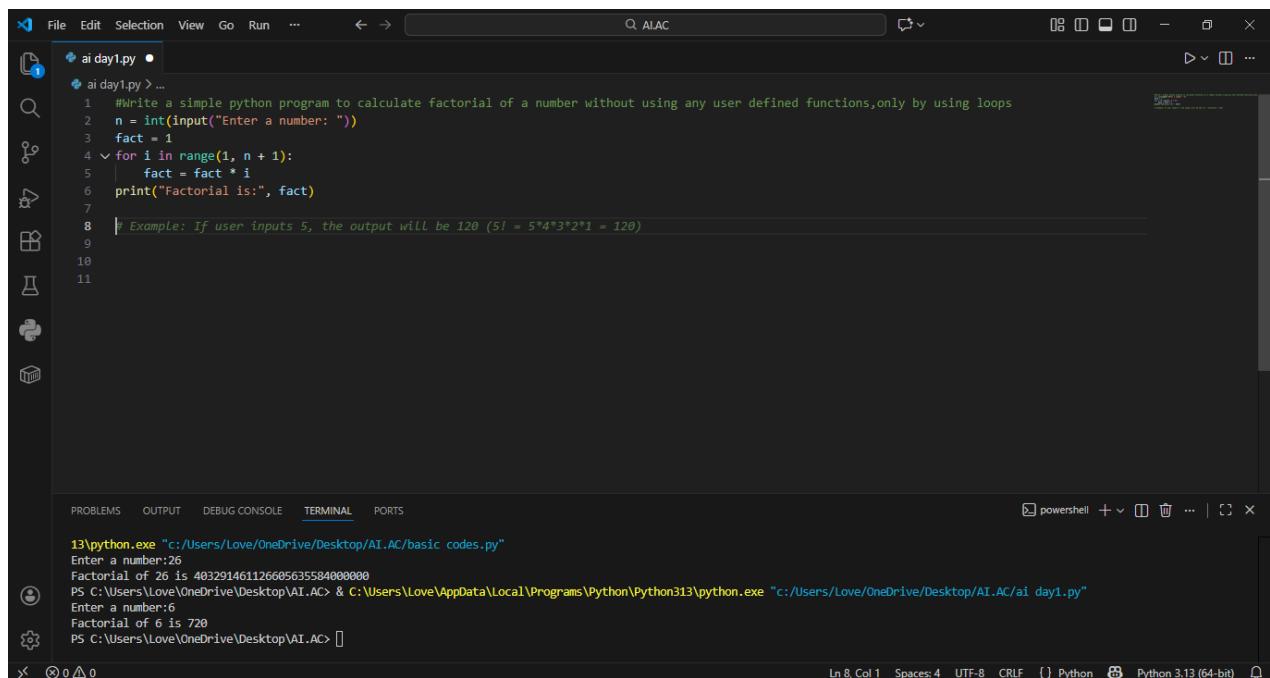
Ht.No: 2403A51L34

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:



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with icons for file operations like new, open, save, and run. The main area displays a Python script named 'ai day1.py'. The code calculates the factorial of a user-specified number using a for loop and prints the result. A note at the bottom explains the example: 'Example: If user inputs 5, the output will be 120 (5! = 5*4*3*2*1 = 120)'. Below the code editor is a terminal window showing the execution of the script. The terminal output includes the command 'python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/basic_codes.py"', user input 'Enter a number:26', the calculated factorial 'Factorial of 26 is 403291461126605635584000000', and another user input 'Enter a number:6' followed by its factorial 'Factorial of 6 is 720'. The terminal also shows the path 'PS C:/Users/Love/OneDrive/Desktop/AI.AC>'.

```
#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)

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"

The screenshot shows the VS Code interface with the file 'ai day1.py' open. The code is a simple loop-based factorial calculator. The terminal below shows the execution of the script and the output for n=6.

```
1 #Write a simple python program to calculate factorial of a number without using any user defined functions,only by using loops
2 n = int(input("Enter a number: "))
3 fact = 1
4 for i in range(1, n + 1):
5     fact = fact * i
6 print("Factorial is:", fact)
7
8 # optimize this code & simplify logic and improve readability
9
10 n = int(input("Enter a number: "))
11 if n < 0:
12     print("Factorial is not defined for negative numbers")
13 elif n == 0 or n == 1:
14     print("Factorial is: 1")
15 else:
16     factorial = 1
17     for i in range(2, n + 1):
18         factorial *= i
19     print(f"Factorial is: {factorial}")
20
21
22
23
```

```
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”

The screenshot shows the VS Code interface with the file 'ai day1.py' open. The code has been modularized into a function 'calculate_factorial'. The terminal below shows the execution of the script and the output for n=6.

```
1 ai day1.py > calculate_factorial
2
3 # create a python program to calculate factorial using a function
4 def calculate_factorial(num):
5     if num < 0:
6         return "Factorial is not defined for negative numbers"
7     if num == 0 or num == 1:
8         return 1
9     result = 1
10    for i in range(2, num + 1):
11        result *= i
12    return result
13
14 # Get input from user
15 number = int(input("Enter a number: "))
16 # Call function and display result
17 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: 9
```

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

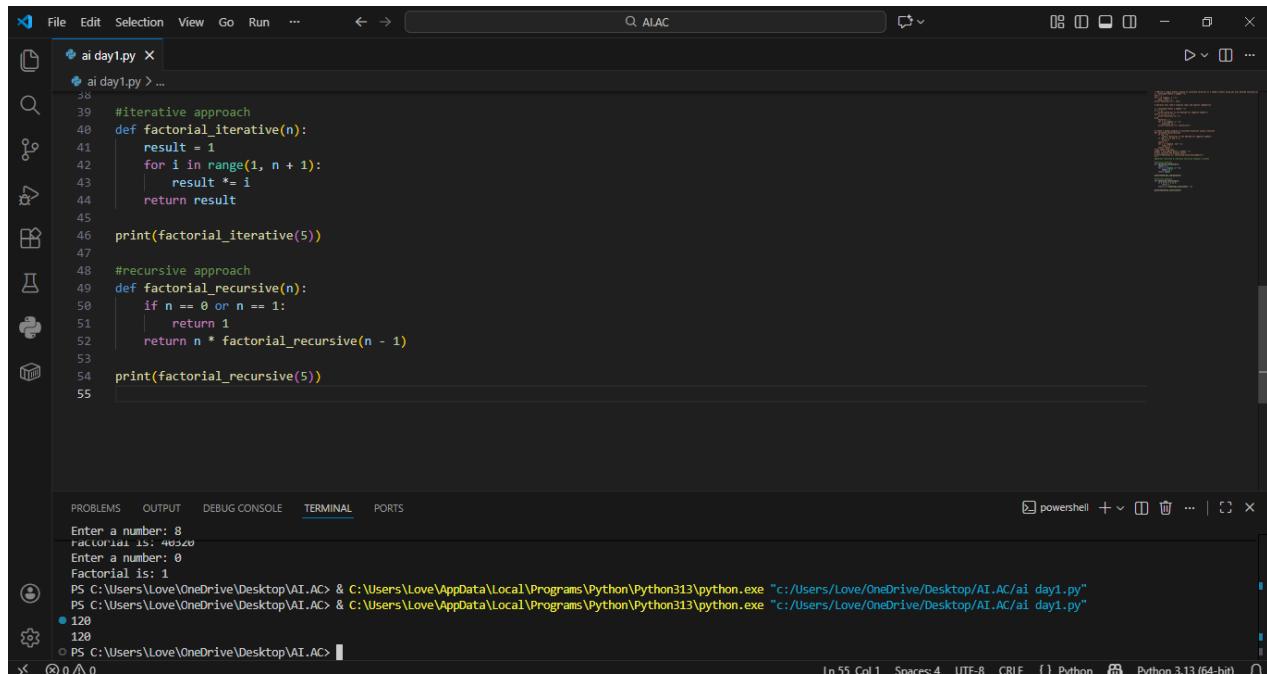
Conclusion:

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”



```
ai day1.py X
ai day1.py > ...
38  #iterative approach
39  def factorial_iterative(n):
40      result = 1
41      for i in range(1, n + 1):
42          result *= i
43      return result
44
45
46  print(factorial_iterative(5))
47
48  #recursive approach
49  def factorial_recursive(n):
50      if n == 0 or n == 1:
51          return 1
52      return n * factorial_recursive(n - 1)
53
54  print(factorial_recursive(5))
55

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell + × └─
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>
In 55 Col 1 Spaces 4 UFT-8 CR LF [ ] Python Python 3.13 (64-bit)
```

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

