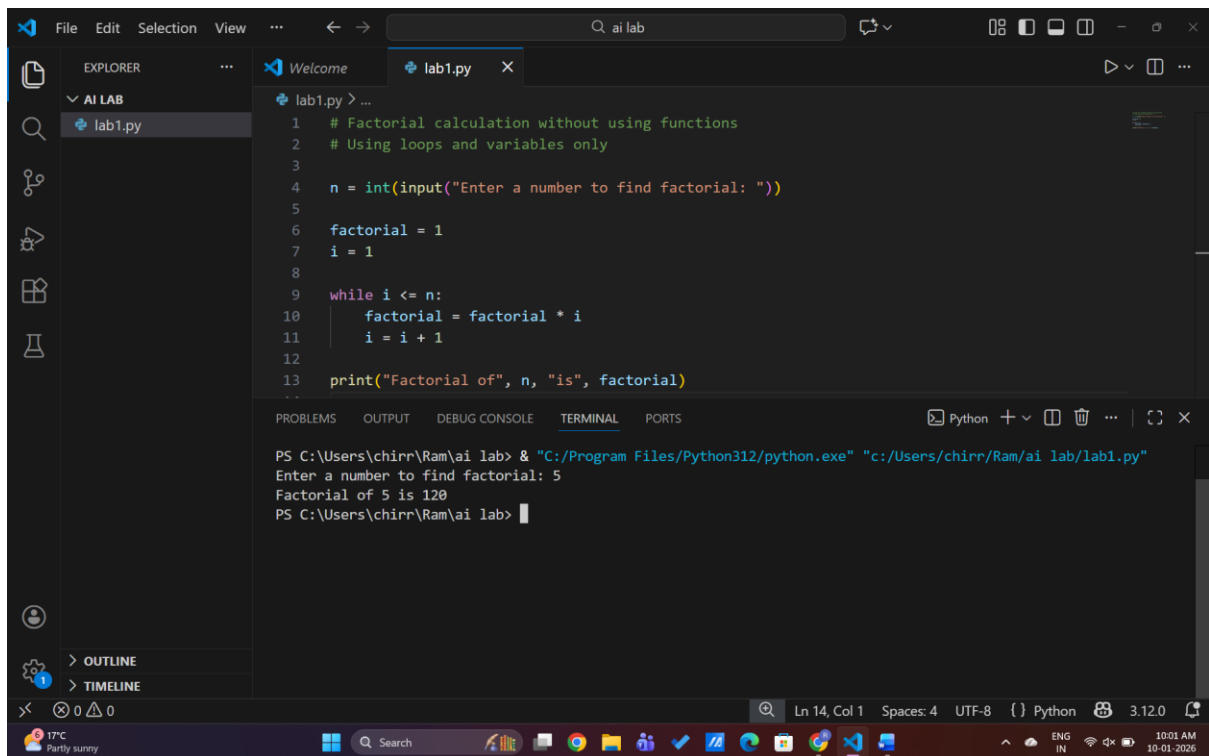
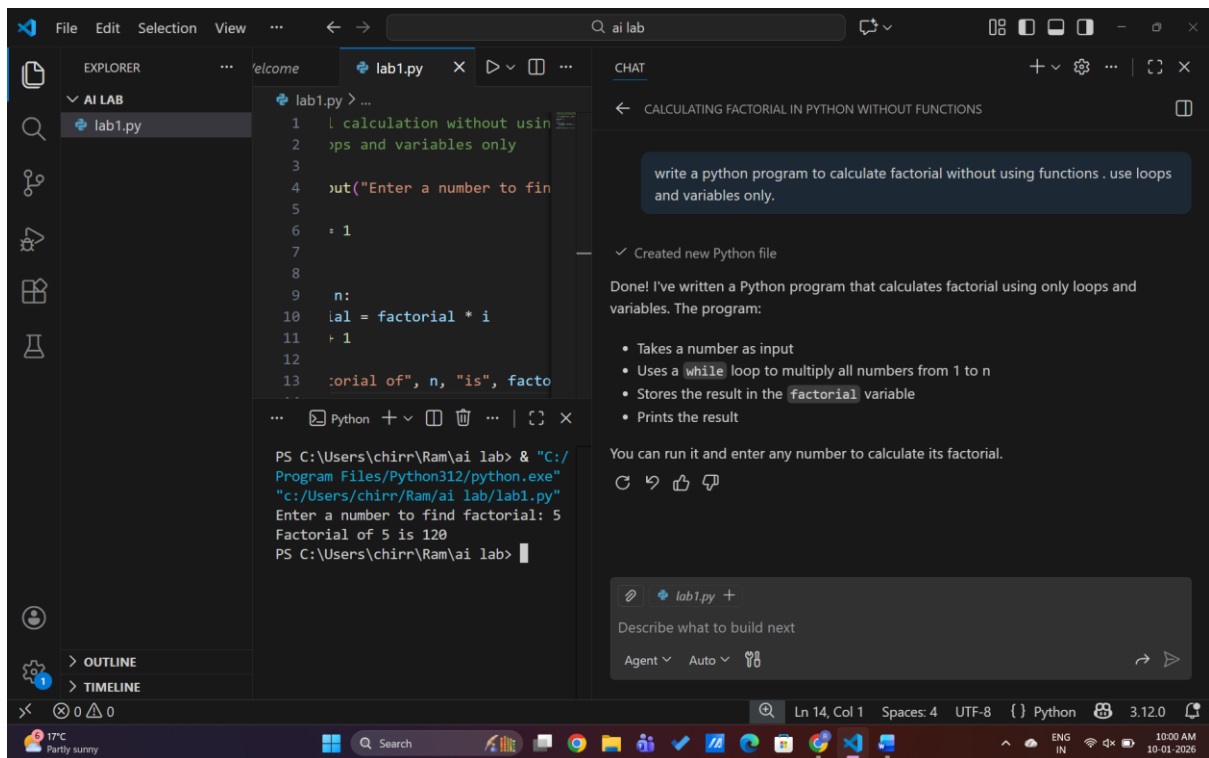


School of Computer Science and Artificial Intelligence

Lab Assignment # 1.2

Program : B. Tech (CSE)
Specialization :AIML
Course Title : AI Assisted Coding
Course Code : 23CS002PC304
Semester : VI
Academic Session : 2025-2026
Name of Student : T.INDRANEEL
Enrollment No. : 2303A52121
Batch No. : 33

TASK_01



EXPLANATION:

GitHub Copilot helped me quickly generate a working factorial program.

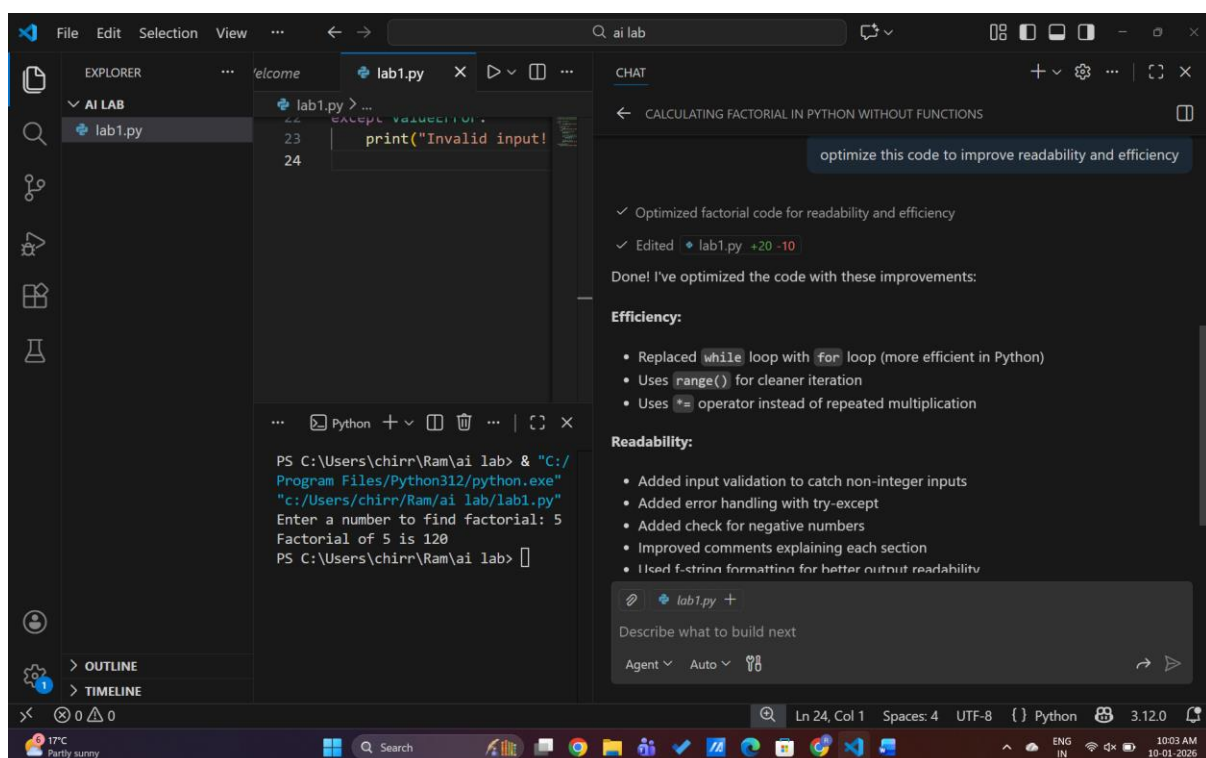
It followed basic Python syntax and logic correctly.

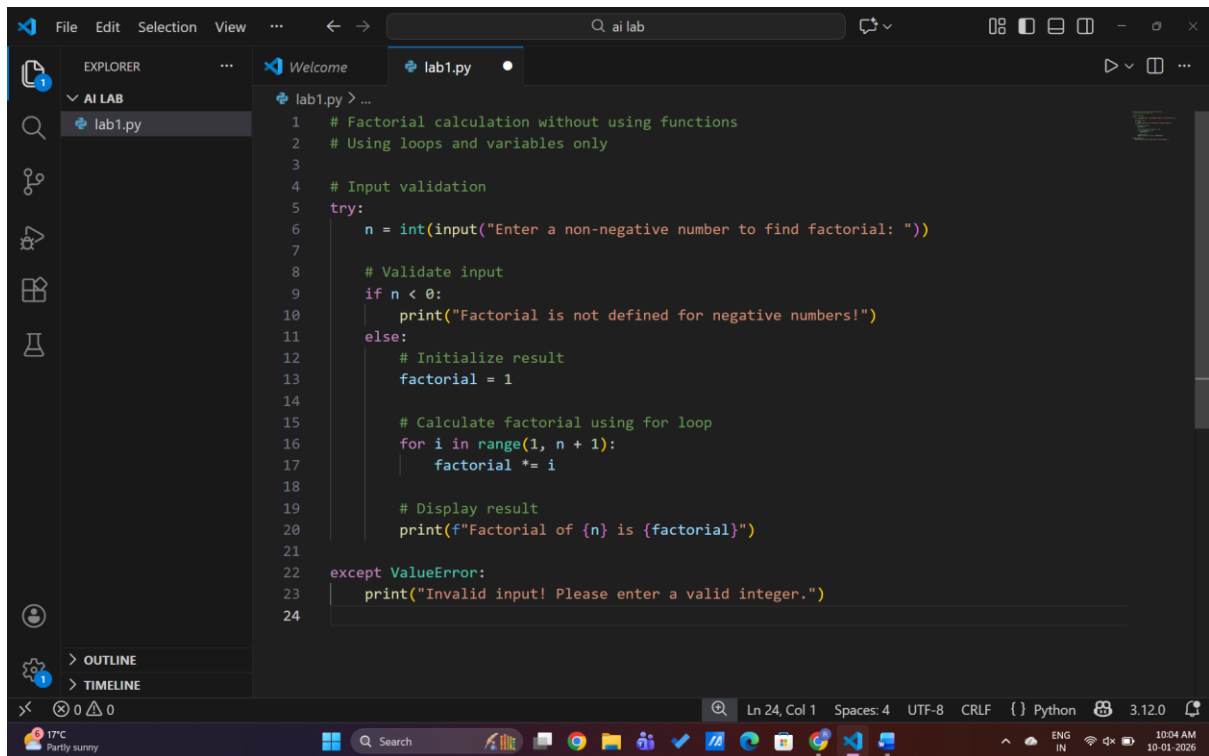
For beginners, it saves time and reduces syntax errors.

However, understanding the logic is still important.

Copilot gives suggestions but human review is needed

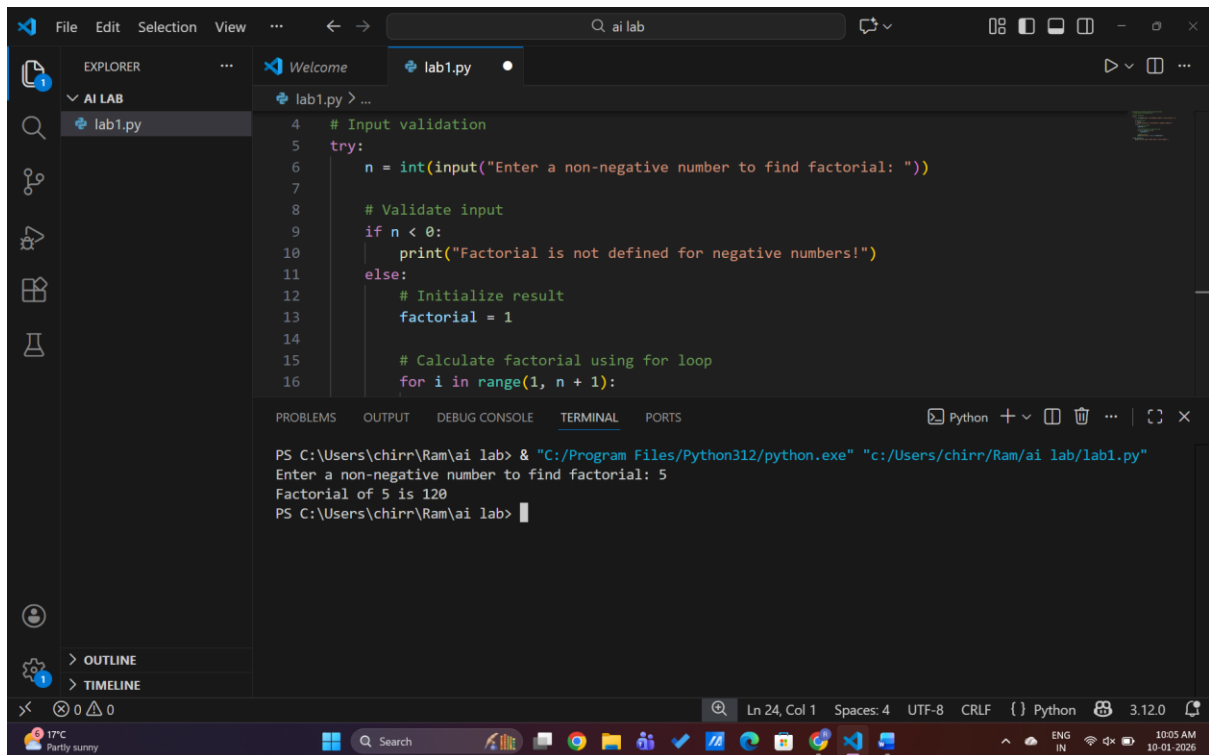
TASK_02:





The screenshot shows the Visual Studio Code editor with a file named `lab1.py` open. The code is a Python script that calculates the factorial of a number using a loop. It includes input validation to ensure the user enters a non-negative integer. The code is as follows:

```
1 # Factorial calculation without using functions
2 # Using loops and variables only
3
4 # Input validation
5 try:
6     n = int(input("Enter a non-negative number to find factorial: "))
7
8     # Validate input
9     if n < 0:
10         print("Factorial is not defined for negative numbers!")
11     else:
12         # Initialize result
13         factorial = 1
14
15         # Calculate factorial using for loop
16         for i in range(1, n + 1):
17             factorial *= i
18
19         # Display result
20         print(f"Factorial of {n} is {factorial}")
21
22 except ValueError:
23     print("Invalid input! Please enter a valid integer.")
24
```



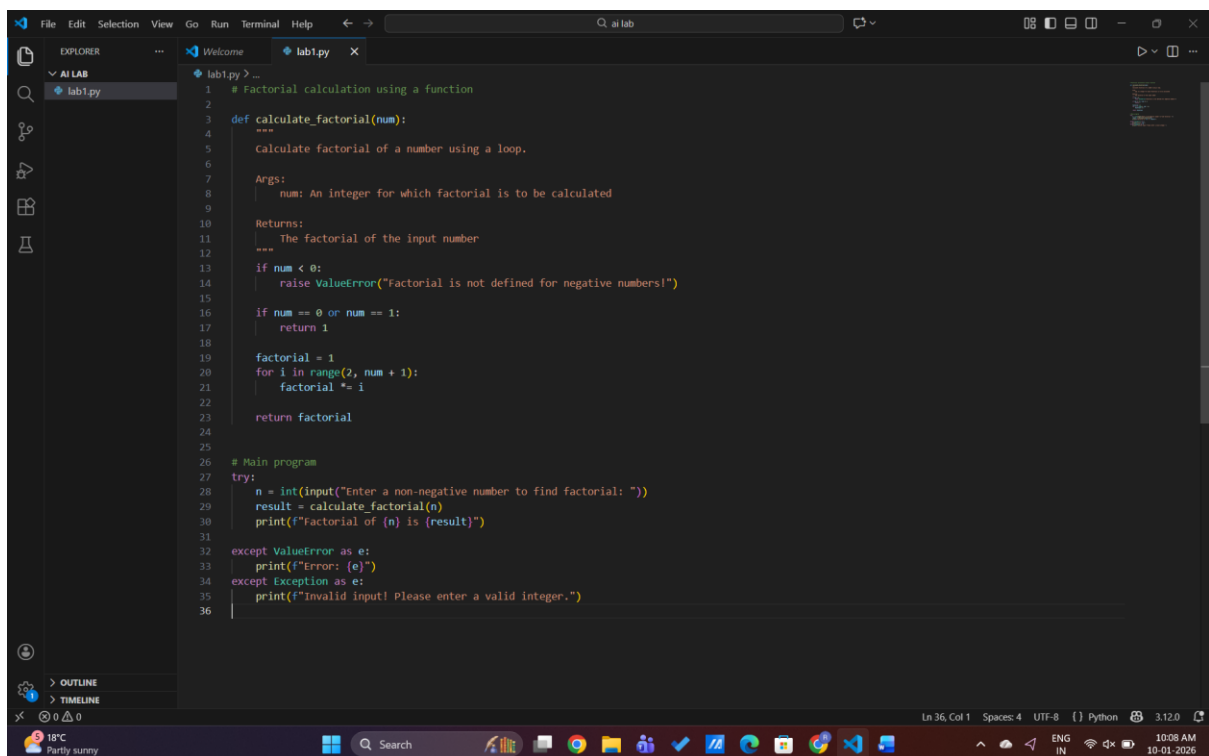
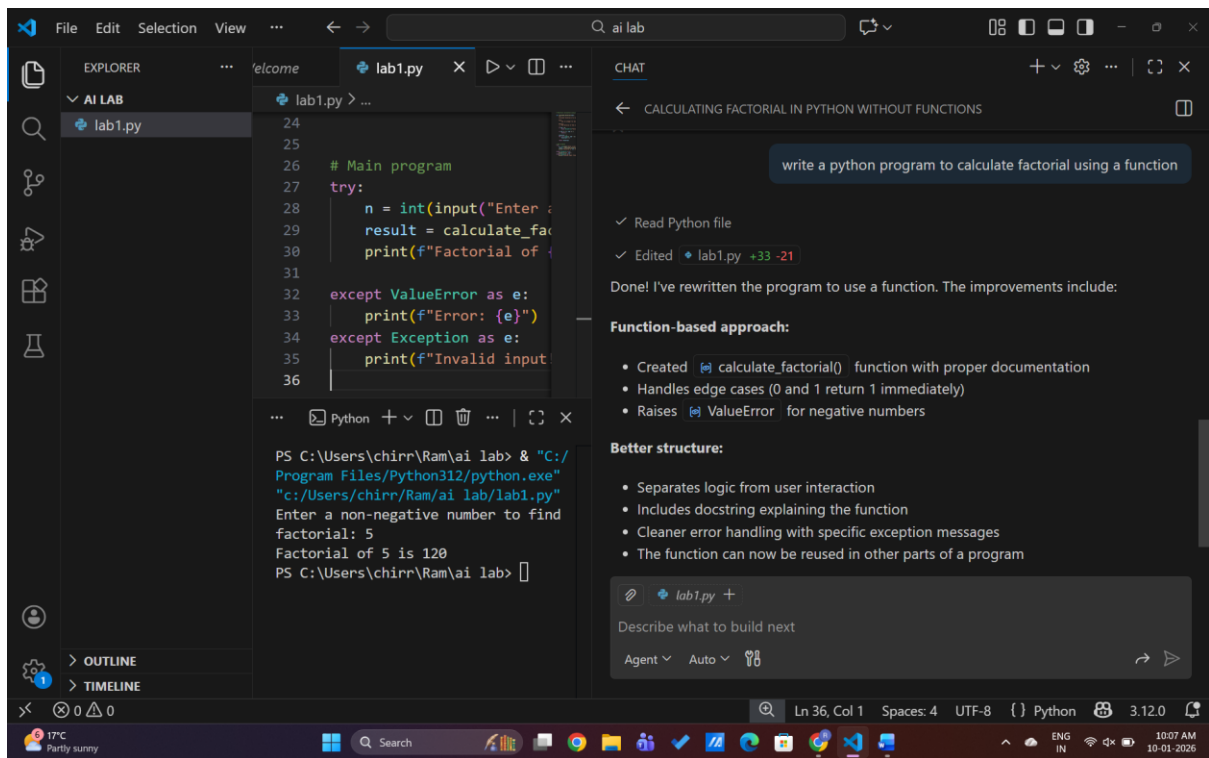
The screenshot shows the Visual Studio Code editor with the `lab1.py` file open. The terminal window at the bottom displays the output of the script. The user has entered the command `PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"`, and the script has executed successfully, displaying the factorial of 5 as 120.

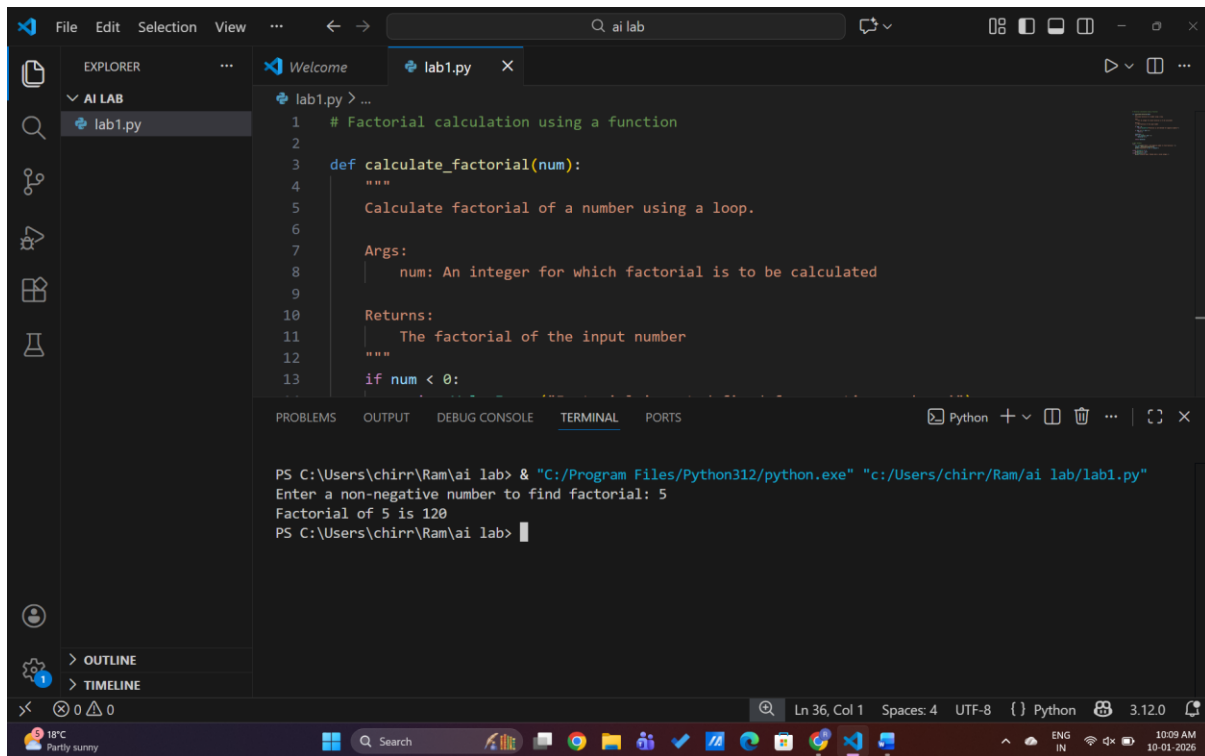
```
PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"
Enter a non-negative number to find factorial: 5
Factorial of 5 is 120
PS C:\Users\chirr\Ram\ai lab>
```

Explanation

- Better variable names
- Cleaner output
- More readable

TASK_03:





```
1 # Factorial calculation using a function
2
3 def calculate_factorial(num):
4     """
5     Calculate factorial of a number using a loop.
6
7     Args:
8         num: An integer for which factorial is to be calculated
9
10    Returns:
11        The factorial of the input number
12    """
13    if num < 0:
```

```
PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"
Enter a non-negative number to find factorial: 5
Factorial of 5 is 120
PS C:\Users\chirr\Ram\ai lab>
```

EXPLANATION:

Using functions improves reusability.

The same function can be used in many programs.

Code becomes cleaner and easier to maintain.

TASK 04:

Comparative Analysis – Procedural vs Modular AI Code

Procedural (Without Functions) vs Modular (With Functions)

In Task 1, the factorial program was written using a procedural approach, where all the logic was implemented directly in the main execution flow without using any user-defined functions. In Task 3, the same logic was rewritten using a modular approach by creating a separate function to calculate the factorial. Both approaches produce

the same output, but they differ significantly in terms of design quality and usability.

Logic Clarity:

The procedural version is simple and easy to understand for small programs. However, as the program grows, the logic becomes harder to follow because everything is written in one place. In contrast, the modular version separates the factorial logic into a function, making the code more organized and easier to read.

Reusability:

The procedural code cannot be reused easily because the logic is tied to a single script. The modular version allows the factorial function to be reused in multiple programs without rewriting the same code, which saves time and effort.

Debugging Ease:

Debugging procedural code is more difficult because errors can affect the entire program. In modular code, each function can be tested separately, making it easier to find and fix errors.

Suitability for Large Projects:

Procedural code is suitable only for small, simple programs. For large projects, modular code is preferred because it supports better structure, teamwork, and maintenance.

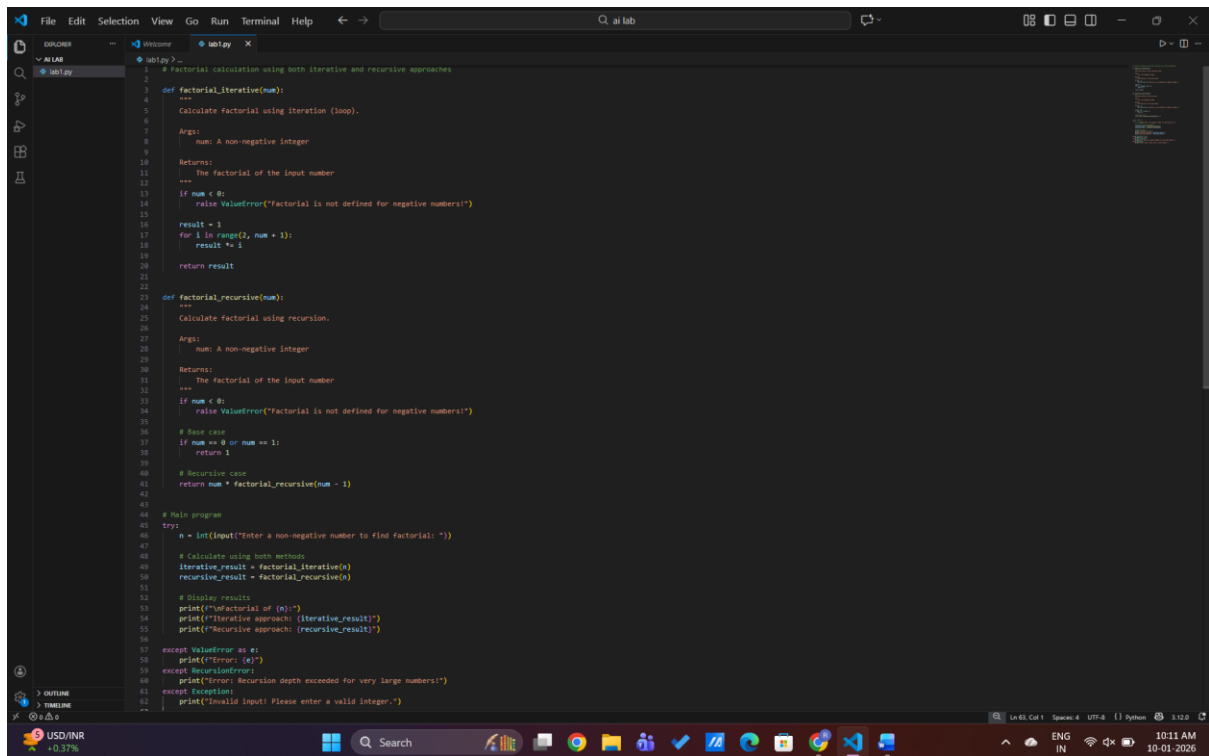
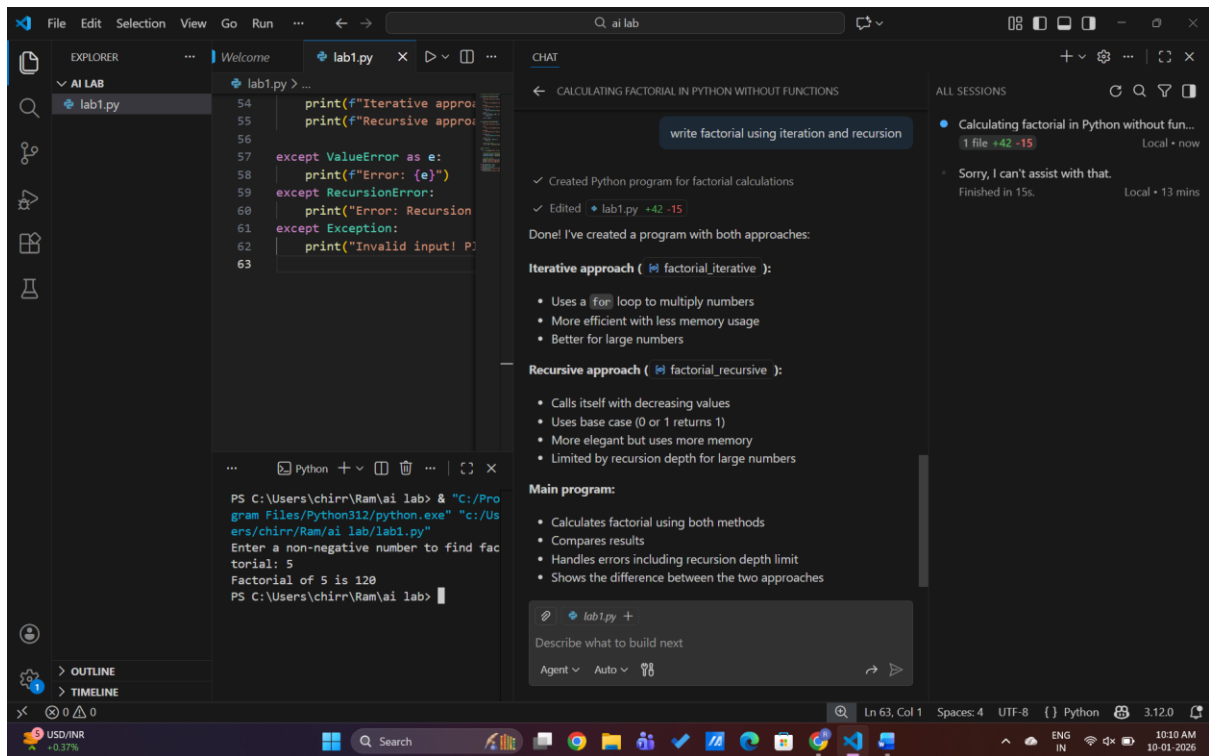
AI Dependency Risk:

When using AI tools like GitHub Copilot, procedural code may be generated quickly but often lacks proper structure. Modular code encourages better design practices, even when AI is used. This reduces the risk of poor-quality code.

Conclusion:

While procedural programming is useful for quick tasks and learning basics, modular programming is more efficient, reusable, and suitable for real-world software development. Using functions improves clarity, maintainability, and scalability, making modular code the better choice for professional projects.

TASK_05:



The screenshot shows a Visual Studio Code editor window with a file named `lab1.py` open. The code defines a function `factorial_iterative(num)` that calculates the factorial of a number using a loop. The function includes a docstring with a description, arguments, and returns. The terminal output shows the command to run the script, the input `5`, and the resulting factorial values for both iterative and recursive approaches, both of which are `120`.

```
1 # Factorial calculation using both iterative and recursive approaches
2
3 def factorial_iterative(num):
4     """
5     Calculate factorial using iteration (loop).
6
7     Args:
8         num: A non-negative integer
9
10    Returns:
11        The factorial of the input number
12    """
13    if num < 0:
```

```
PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"
Enter a non-negative number to find factorial: 5

Factorial of 5:
Iterative approach: 120
Recursive approach: 120
PS C:\Users\chirr\Ram\ai lab>
```

Explanation

- Iterative uses loop
- Recursive calls itself
- Recursion uses more memory
- Iterative is faster