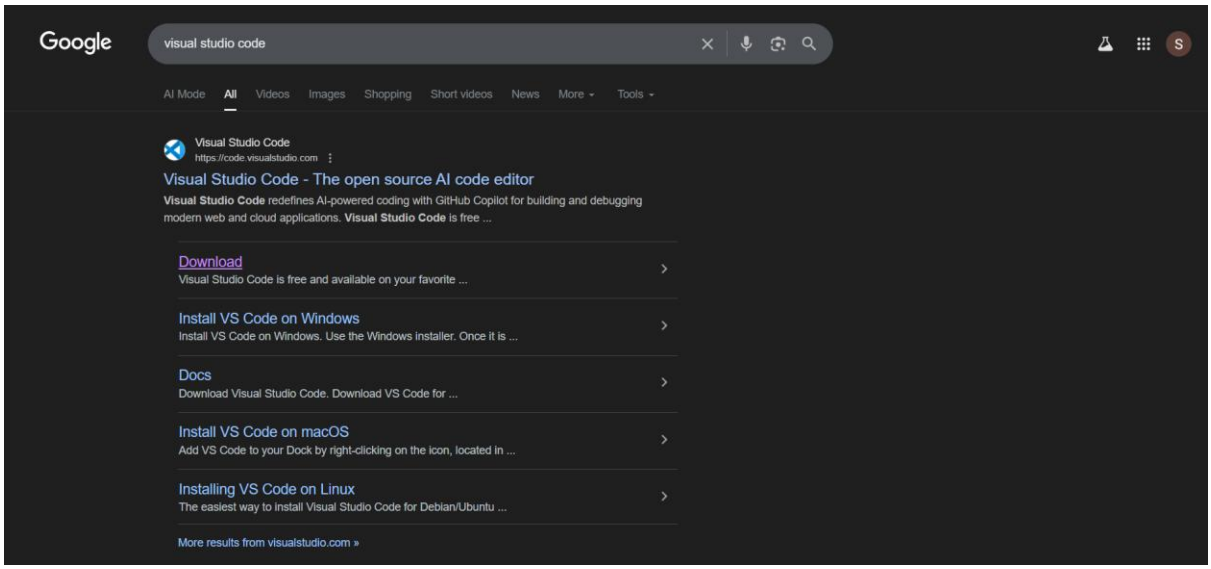
SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment 1.2

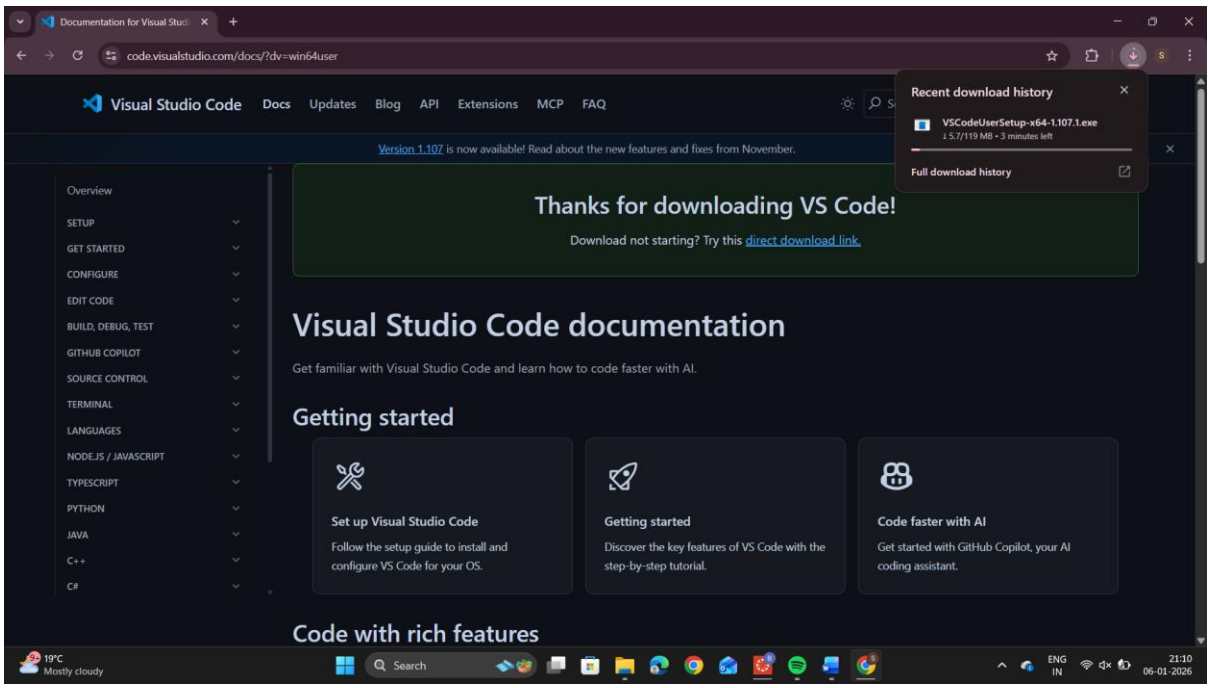
Program	**: B. Tech (CSE)**
Course Title	**: AI Assisted Coding**
Course Code	**: 23CS002PC304**
Semester	**: III**
Academic Session	**: 2025-2026**
Name of Student	**: Sruthi**
Enrollment No.	**: 2403A51L10**
Batch No.	**: 51**
Date	**: 06-10-2026**

Search the Visual Studio Code in the browser. Then click on the Download.

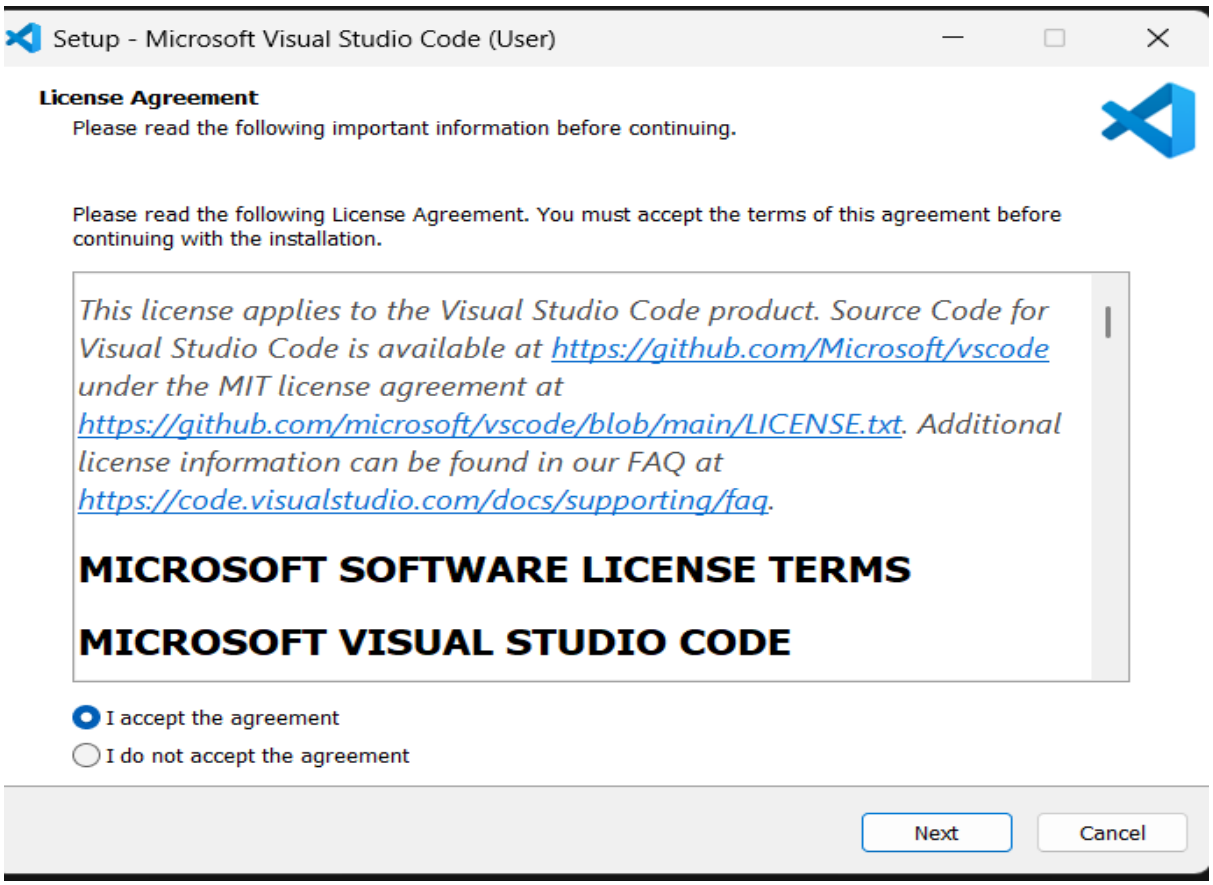


Click on the OS your desktop or laptop

A download .exe file you can see.

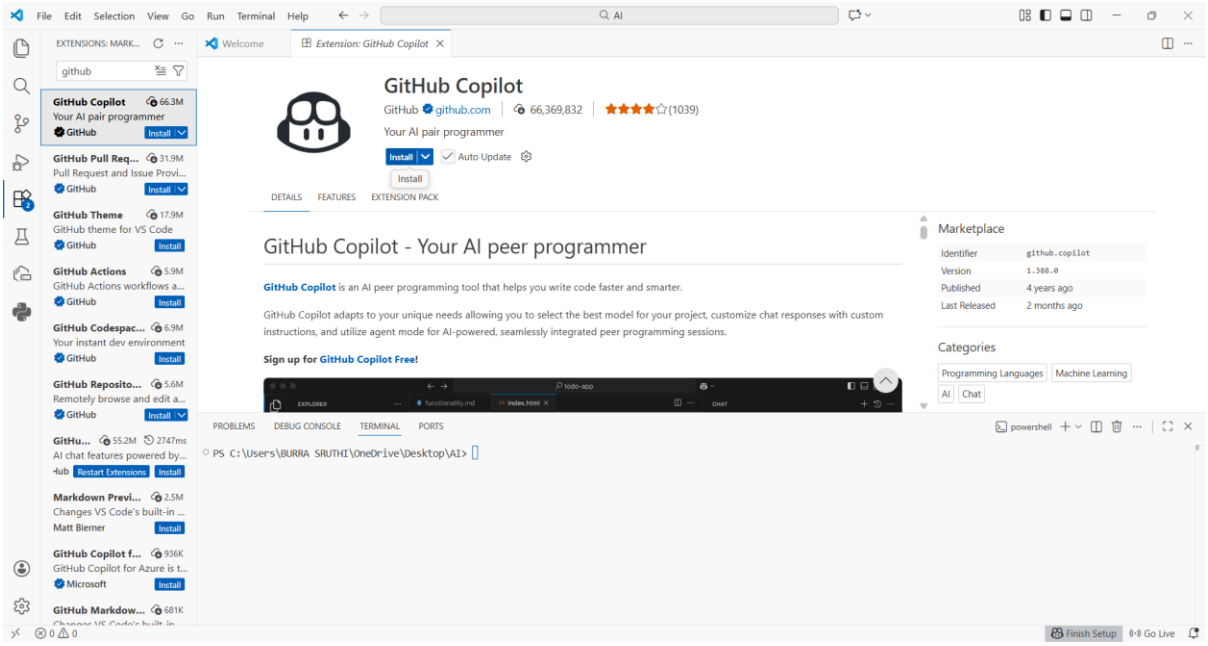


After the downloading the .exe file completed. Click on the .exe file then you will see the below window . Then click on the accept the agreement click next.

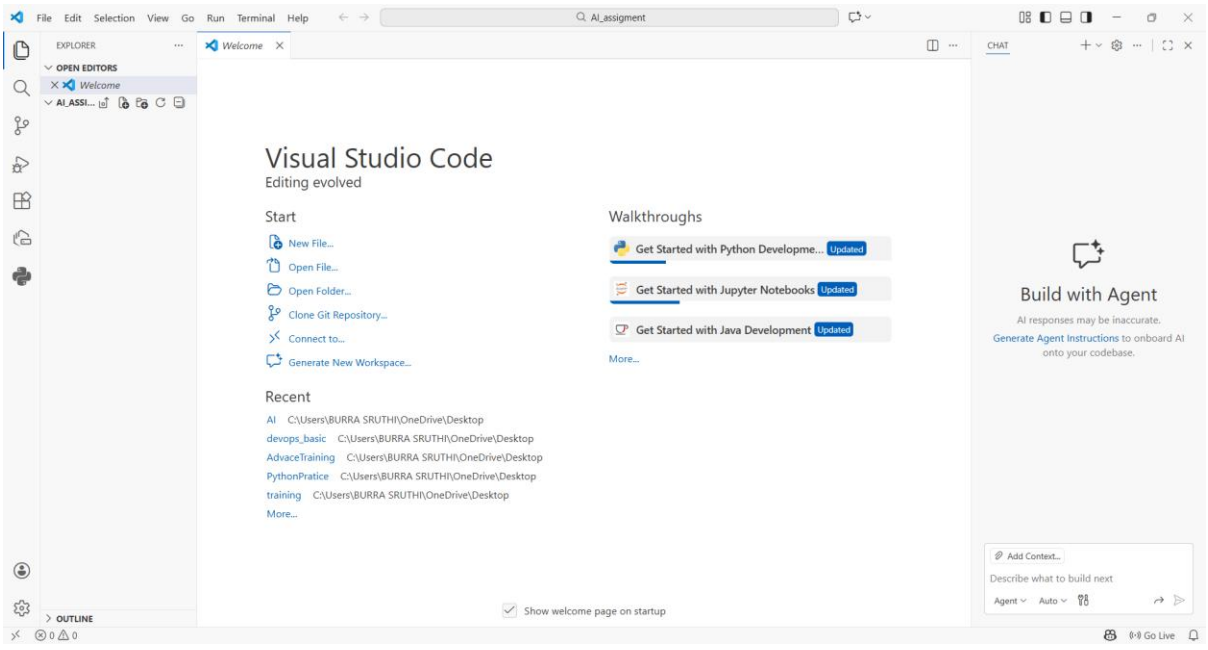


After that click next and then click on install. After the completion of download. Open the vs code

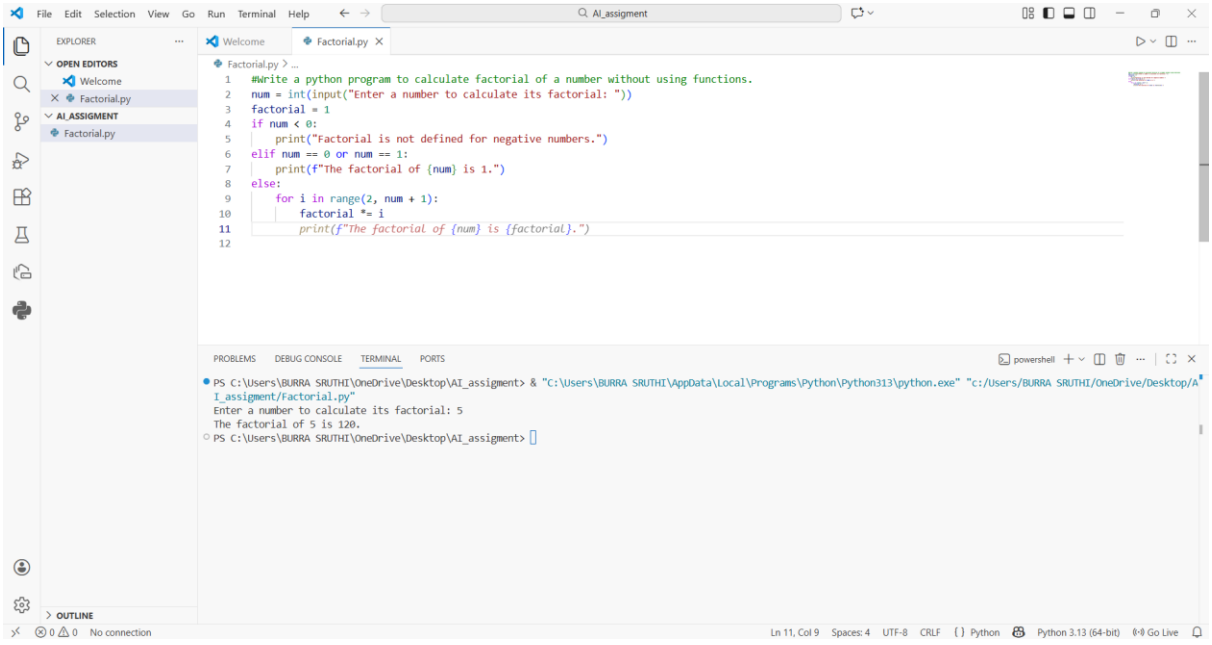
Click on the extension then search for GitHub Copilot. Click on the install



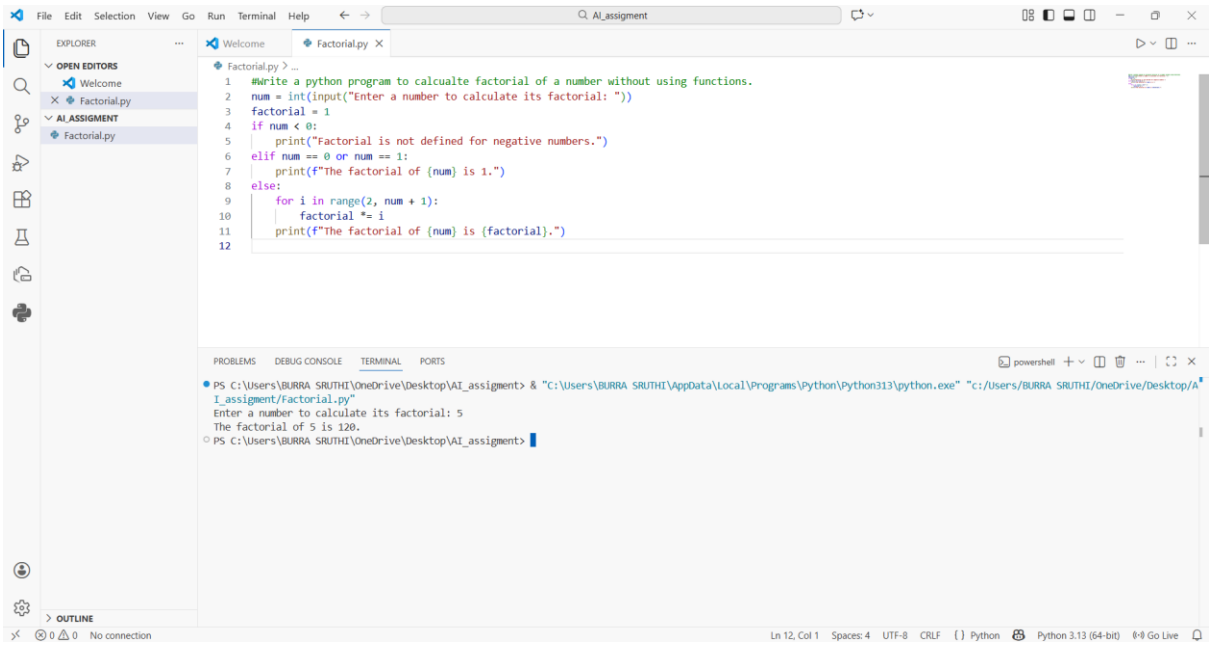
After the installation completed . sign in into the github copilot using the account of github. Create a new folder on desktop open it in the visual studio.



Create a new python file
And give prompt as **#Write a python program to calculate factorial of a number without using functions.**
The AI gives Inline suggestion like which is in the below window.

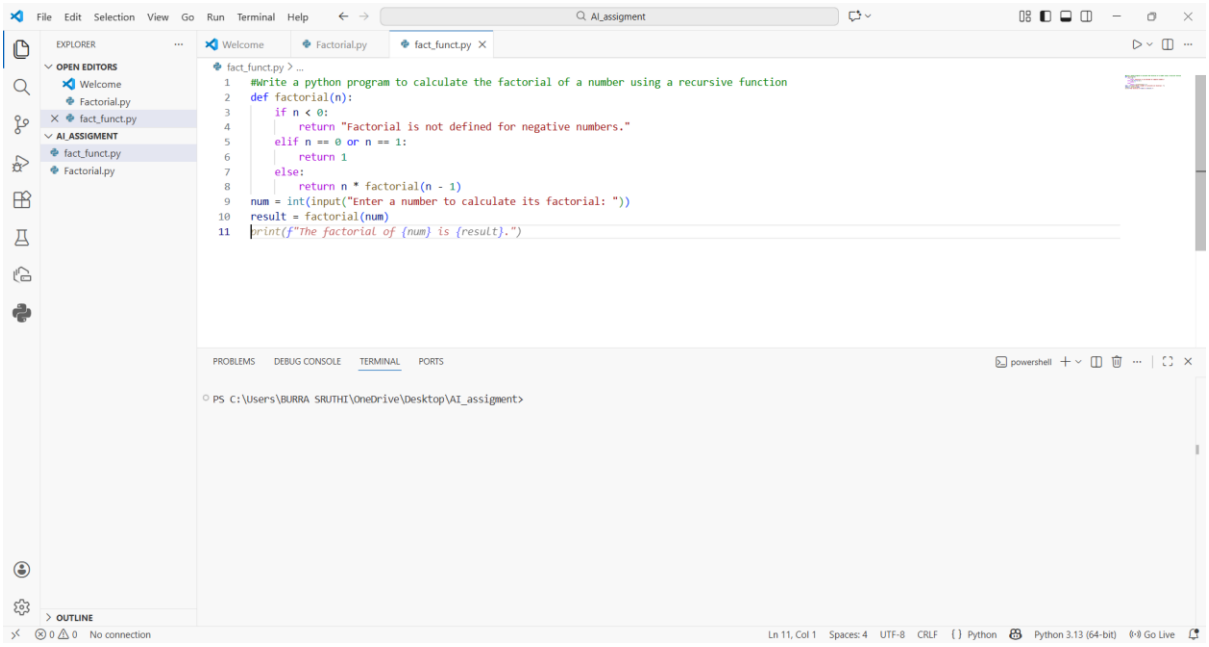


Below window shows the total code and the output of the code

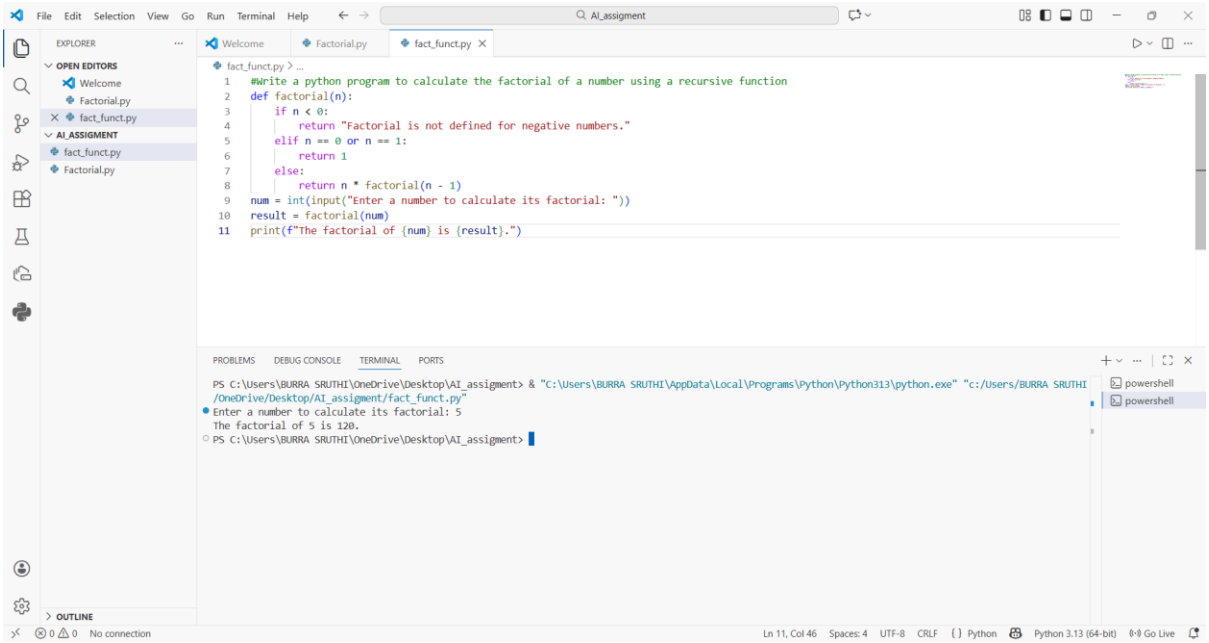


SR UNIVERSITY

Create a new python file
Give the prompt as **#Write a python program to calculate the factorial of a number using a recursive function**
The github copilot give inline suggestions after you enter the prompt.
The copilot gives the suggestions like in the below window



The input and output of the factorial of a number.



SR UNIVERSITY

Comparative Analysis Table

Criteria	Without Functions	With Recursive Functions
Logic Clarity	Logic mixed with input/output	Logic separated and input separated
Reusability	Cannot reuse logic easily	Function can be reused anywhere
Debugging Ease	Harder to isolate errors	Easy to test function independently
Suitability for Large Projects	Not suitable	Highly suitable
AI Dependency Risk	High (copy-paste dependent)	Lower (structured and controlled)

Execution Flow Explanation:

Without function:

1. The program prompts the user to enter a number.
2. It checks if the number is negative, zero, or one, and handles those cases accordingly.
3. For numbers greater than one, it initializes a variable 'factorial' to 1.
4. It then uses a for loop to iterate from 2 to the entered number and multiplies 'factorial' by each integer in that range.
5. Finally, it prints the calculated factorial.

With Recursive Functions:

1. The program defines a recursive function 'factorial' that takes an integer 'n' as input.
2. Inside the function, it checks if 'n' is negative, zero, or one, and returns appropriate values for those cases.
3. For numbers greater than one, the function calls itself with 'n-1 and multiplies the result by 'n', effectively calculating the factorial recursively.
4. The program then prompts the user to enter a number and calls the 'factorial' function with that number.
5. Finally, it prints the calculated factorial.

Comparison Analysis

Aspect	Without function	With Recursive function
Readability	Very clear and straightforward	Very elegant and mathematically expressive
Stack Usage	Uses constant memory	Uses call stack for each function call
Performance	Faster due to no function call overhead	Slightly slower due to recursion overhead
Memory Usage	Low	Higher
Ease of Debugging	Easier	Harder due to nested calls

SR UNIVERSITY

When Recursion is Not Recommended

Recursion should be avoided when:

1. **Input size is very large**

   Can cause stack overflow

2. **Problem has a simple loop-based solution**

   Iteration is more efficient

3. **Performance is critical**

   Recursive calls add overhead

4. **Language has limited recursion depth**

   Python has a recursion limit (~1000)