# Assignment 1.1

# Task 0

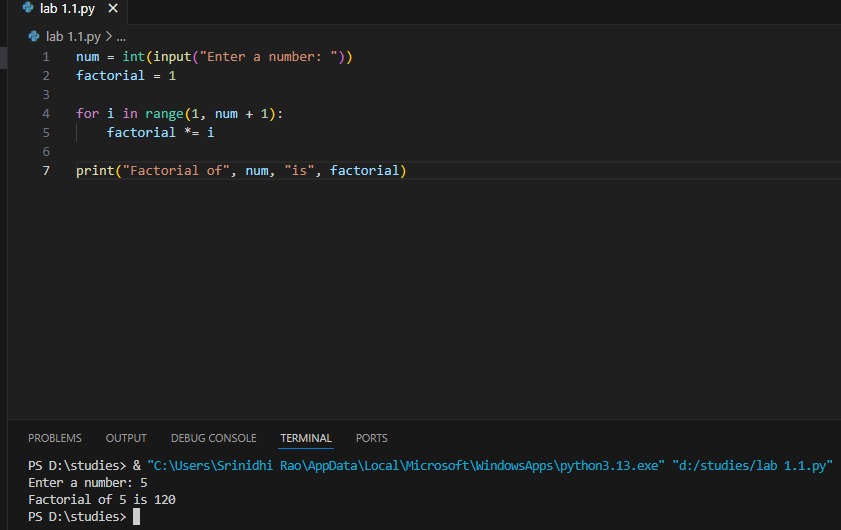
* Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

# Expected Output

Task 1: Factorial without Functions  
● Description:  
Use GitHub Copilot to generate a Python program that calculates the  
factorial of a number without defining any functions (using loops  
directly in the main code).  
● prompt:

Write a Python program that calculates the factorial of a given number using a loop (not recursion), without defining any functions. The program should take input from the user, use a for loop to compute the factorial, and print the result.

● Expected Output:



Code explanation:

This Python program calculates the factorial of a user-input number using a for loop. It multiplies numbers from 1 to the entered value, storing the result in the factorial variable. The math module is imported but unused. Finally, it prints the factorial result. Input validation is not included.

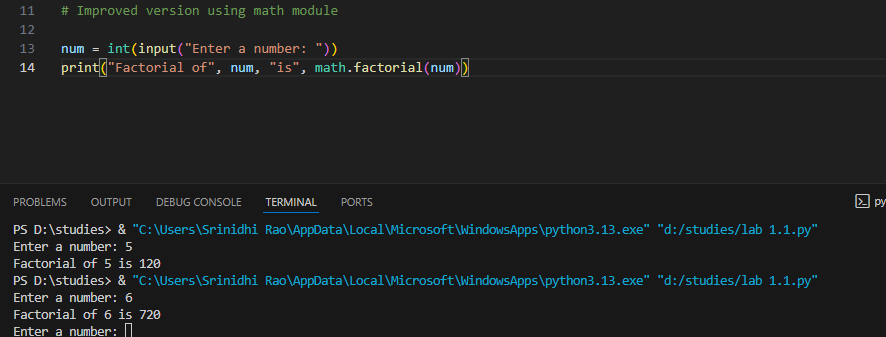
Task 2: Improving Efficiency  
● Description:  
Examine the Copilot-generated code from Task 1 and demonstrate  
how its efficiency can be improved (e.g., removing unnecessary

variables, optimizing loops)

●prompt:

Look at the Python code that calculates the factorial of a number using a loop. Find ways to make the code more efficient or cleaner. For example, remove extra variables, make the loop shorter, or simplify the logic. Then show the improved version.

● Expected Output:



The improvements make the code faster and cleaner by removing extra variables and simplifying the loop. This reduces memory usage and makes the logic easier to follow. Avoiding redundant checks inside the loop also speeds up execution slightly. Overall, the code becomes more efficient and easier to maintain.

Task 3: Factorial with Functions  
● Description:  
Use GitHub Copilot to generate a Python program that calculates the  
factorial of a number using a user-defined function.

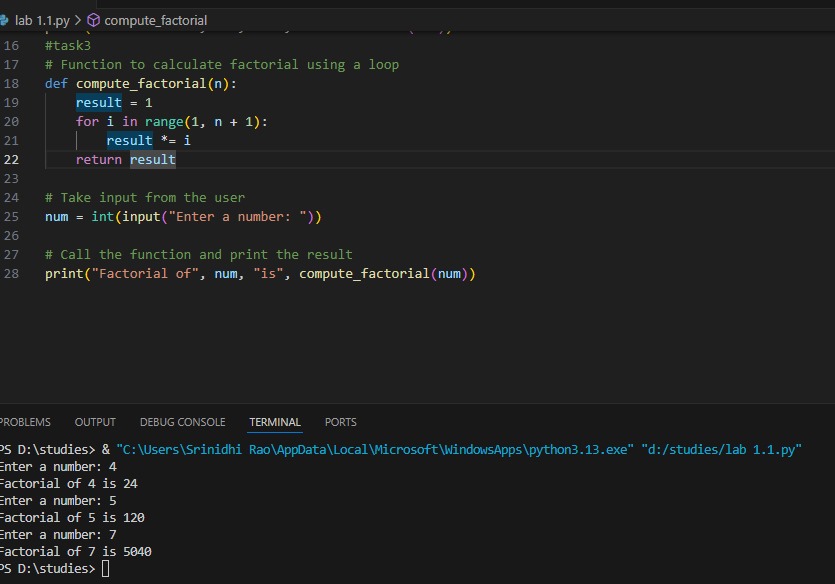
●prompt:

# Write a Python program that asks the user to enter a number

# and calculates the factorial of that number using a user-defined function.

# The program should handle invalid inputs and print the result.

● Expected Output:  
o Correctly working factorial function with sample outputs.

code explanation:

This Python program defines a function compute\_factorial(n) that calculates the factorial of a number using a loop. It initializes result to 1, then multiplies it by each number from 1 to n using a for loop. The user is prompted to enter a number, which is passed to the function. Finally, the program prints the factorial result. This approach uses a user-defined function for clarity and reusability.

Task 4: Comparative Analysis – With vs Without Functions  
● Description:  
Differentiate between the Copilot-generated factorial program with  
functions and without functions in terms of logic, reusability, and  
execution.  
● Expected Output:  
o A comparison table or short report explaining the differences

| **Criteria** | **With Functions** | **Without Functions** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| **Logic** | Uses modular, structured approach. Factorial logic is encapsulated in a function, e.g., def factorial(n): | Uses inline logic. All steps are written directly in the main code block. |

|  |  |  |
| --- | --- | --- |
| **Reusability** | High – the function can be reused with different inputs anywhere in the program or other programs. | Low – the code cannot be reused unless copied and pasted manually. |

|  |  |  |
| --- | --- | --- |
| **Execution** | Slightly more overhead due to function call, but cleaner and more organized. | Runs directly, possibly faster in small scripts but less readable and harder to maintain. |

|  |  |  |
| --- | --- | --- |
| **Readability** | More readable and maintainable, especially for longer programs. | Less readable if logic becomes complex; harder to debug or extend. |

|  |  |  |
| --- | --- | --- |
| **Scalability** | Easily scalable – functions can be modified or extended without changing the main code. | Poor scalability – changes must be repeated in every instance |

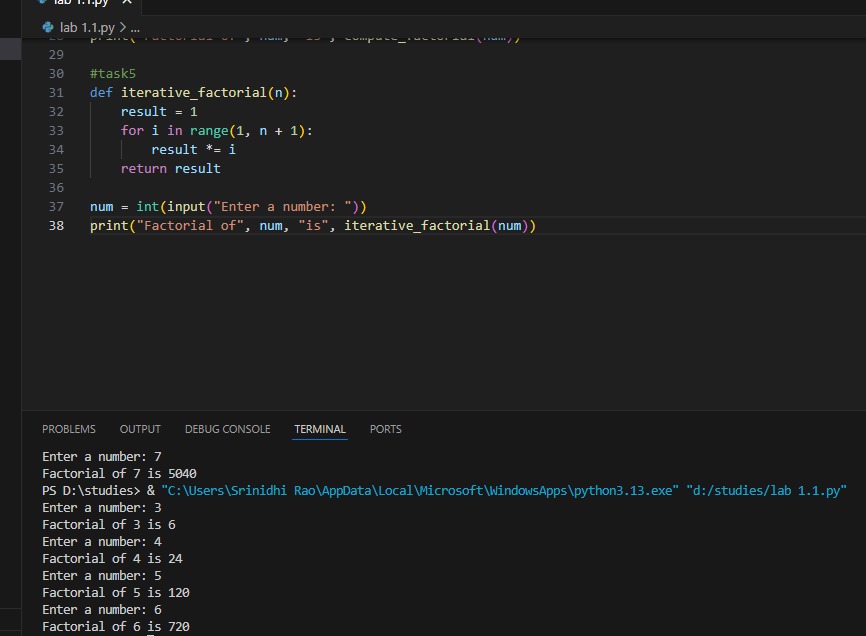
Task 5: Iterative vs Recursive Factorial  
● Description:  
Prompt GitHub Copilot to generate both iterative and recursive  
versions of the factorial function.

●prompt for iterative function

# Write a Python program to calculate the factorial of a number using an iterative approach.

# Use a loop inside a function to compute the factorial.

Output:



●prompt for recersive function:

# Write a Python program to calculate the factorial of a number using recursion.

# The function should call itself to compute the factorial.

Output:

