Name: Moutam Bhavani
Enroll no : 2403A54084

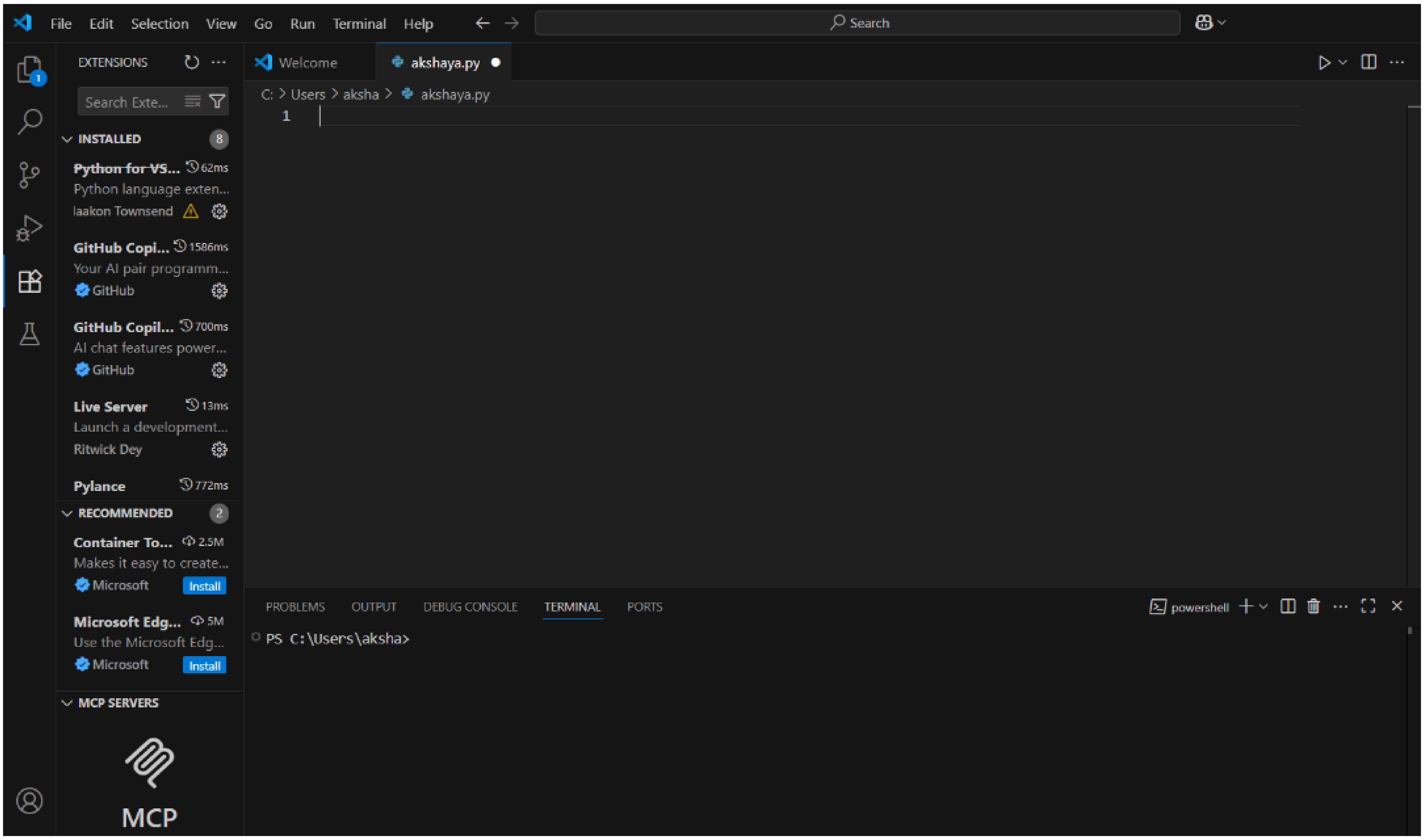
Batch : (DS)batch-3

Task 0

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

Expected Output

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

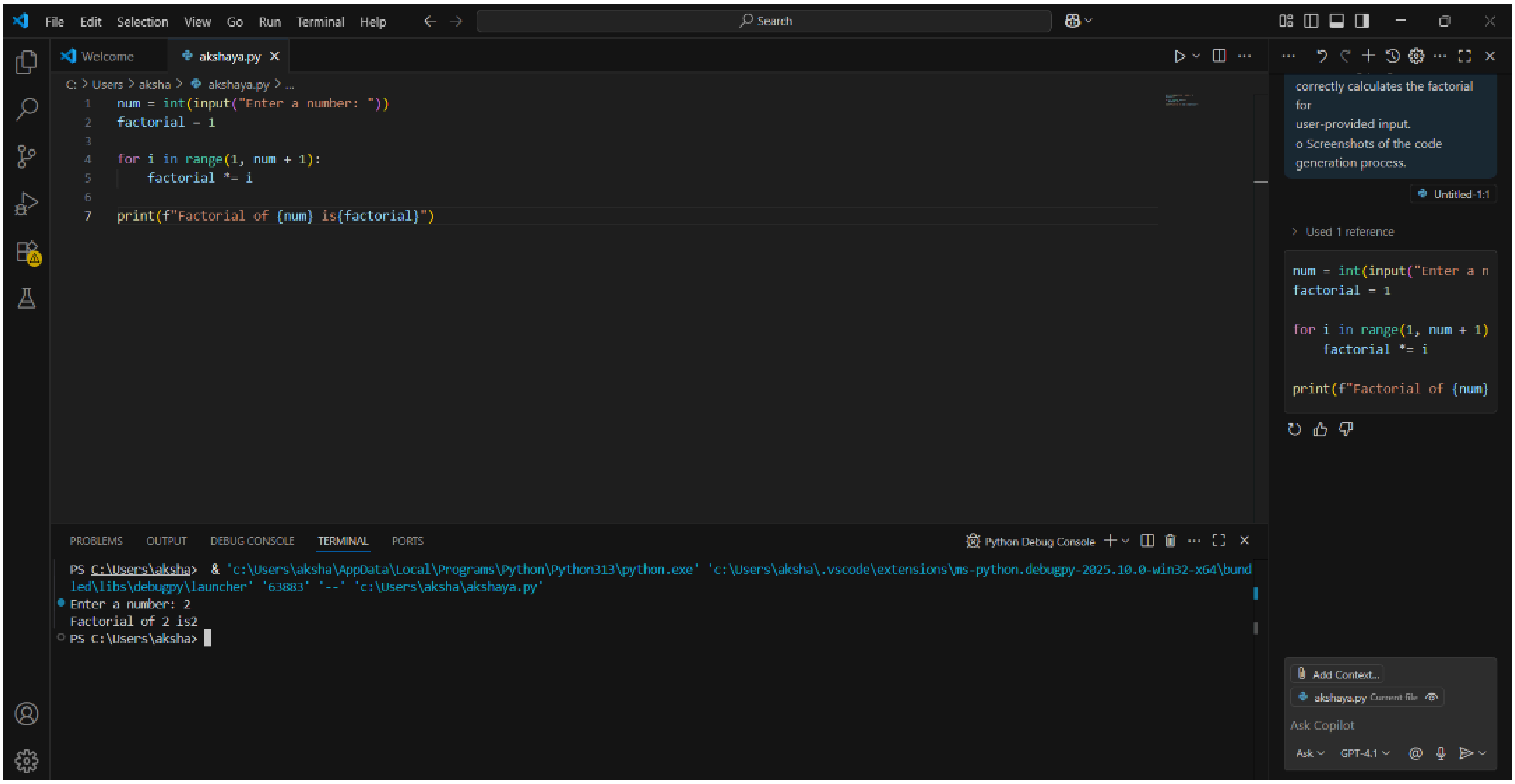


```
PS C:\Users\aksha>
```

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).
- Expected Output:
  - A working program that correctly calculates the factorial for user-provided input.
  - Screenshots of the code generation process.

prompt : generate a python program which takes a number from the user and give the factorial as the output ,without using any functions



```python
num = int(input("Enter a number: "))
factorial = 1

for i in range(1, num + 1):
    factorial *= i

print(f"Factorial of {num} is{factorial}")
```

```
PS C:\Users\aksha>  & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bund
led\libs\debugpy\launcher' '63883' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 2
Factorial of 2 is2
PS C:\Users\aksha>
```

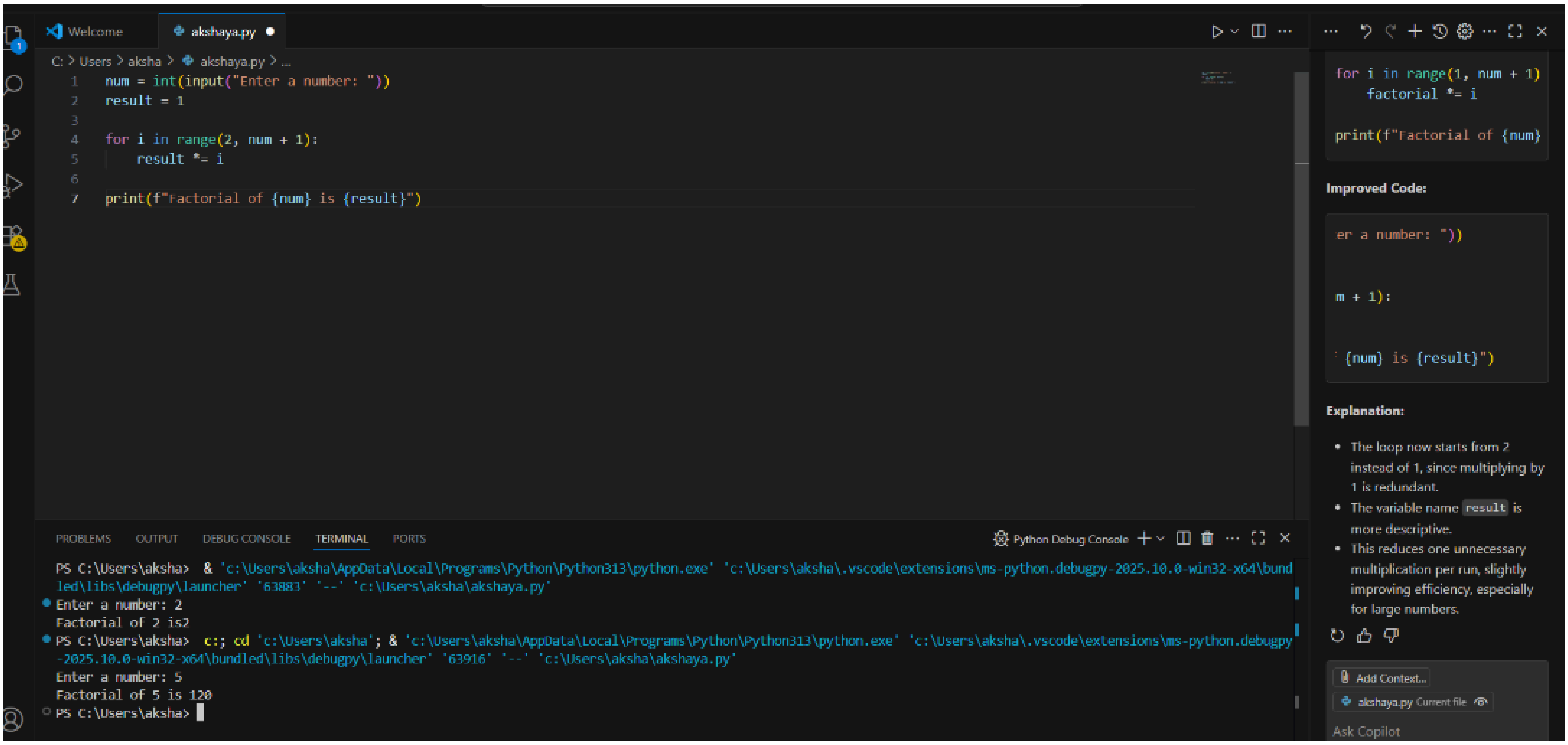
correctly calculates the factorial
for
user-provided input.
o Screenshots of the code
generation process.

```python
num = int(input("Enter a n
factorial = 1

for i in range(1, num + 1)
    factorial *= i

print(f"Factorial of {num}
```

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).
- Expected Output:
  - Original and improved versions of the code.
  - Explanation of how the improvements enhance performance.

prompt : improve the efficiency from the previous code by removing unnecessary variables,optimizing loops etc...



```python
num = int(input("Enter a number: "))
result = 1

for i in range(2, num + 1):
    result *= i

print(f"Factorial of {num} is {result}")
```

```
PS C:\Users\aksha>  & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63883' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 2
Factorial of 2 is2
PS C:\Users\aksha>  c:; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63916' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\aksha>
```

```
for i in range(1, num + 1)
    factorial *= i

print(f"Factorial of {num}
```

Improved Code:

```
er a number: "))


m + 1):


 {num} is {result}")
```

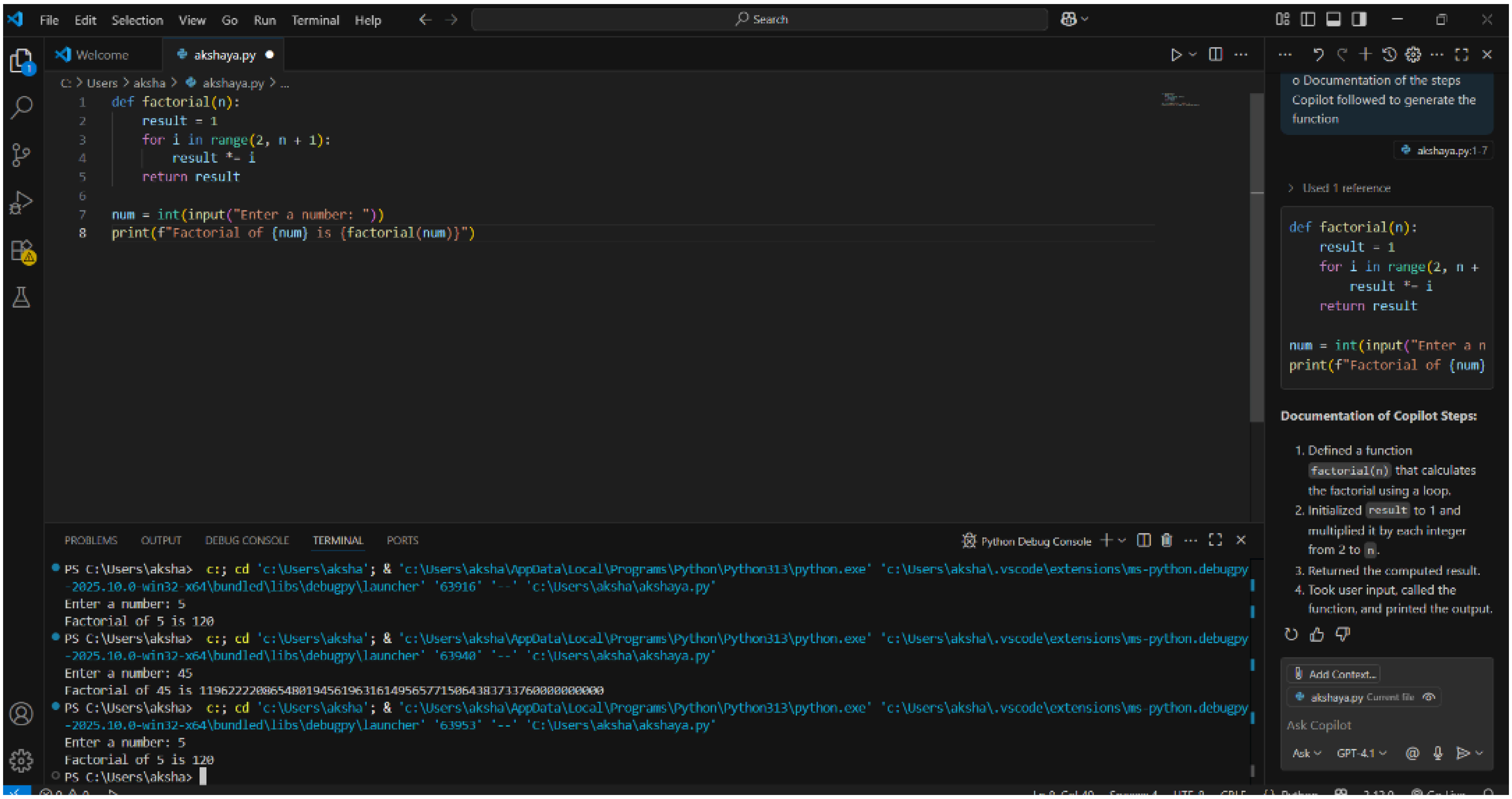
Explanation:

- The loop now starts from 2 instead of 1, since multiplying by 1 is redundant.
- The variable name `result` is more descriptive.
- This reduces one unnecessary multiplication per run, slightly improving efficiency, especially for large numbers.

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.
- Expected Output:
  - Correctly working factorial function with sample outputs.
  - Documentation of the steps Copilot followed to generate the function.

Prompt :generate a python program which take the input from the user and give factorial of it by taking an user defined function



```python
def factorial(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

num = int(input("Enter a number: "))
print(f"Factorial of {num} is {factorial(num)}")
```

```
PS C:\Users\aksha>  c:; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63916' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\aksha>  c:; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63940' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 45
Factorial of 45 is 119622220865480194561963161495657715064383733760000000000
PS C:\Users\aksha>  c:; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63953' '--' 'C:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\aksha>
```

o Documentation of the steps Copilot followed to generate the function

akshaya.py:1-7

Used 1 reference

```python
def factorial(n):
    result = 1
    for i in range(2, n +
        result *= i
    return result

num = int(input("Enter a n
print(f"Factorial of {num}
```

Documentation of Copilot Steps:

1. Defined a function `factorial(n)` that calculates the factorial using a loop.
2. Initialized `result` to 1 and multiplied it by each integer from 2 to `n`.
3. Returned the computed result.
4. Took user input, called the function, and printed the output.

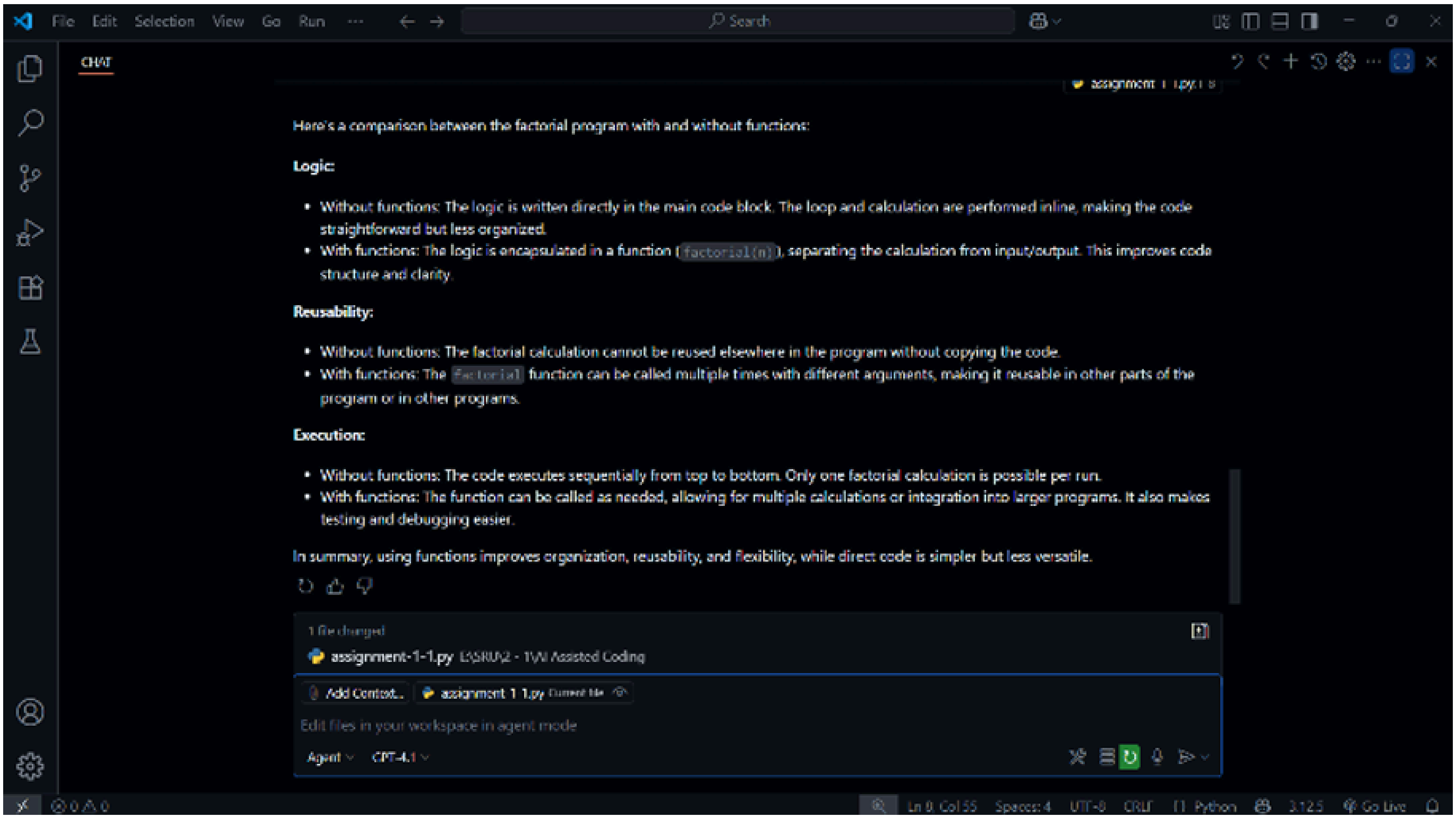
Task 4: Comparative Analysis – With vs Without Functions

- Description:
  Differentiate between the Copilot-generated factorial program with functions and without functions in terms of logic, reusability, and execution.
- Expected Output:

A comparison table or short report explaining the differences

Prompt:

- Differentiate between program with functions and without functions in terms of logic, reusability, and execution.


File Edit Selection View Go Run ···
Search
CHAT
assignment-1-1.py
1 file changed
assignment-1-1.py E:\SRU\2 - 1\AI Assisted Coding
Add Context...
assignment-1-1.py Current file
Edit files in your workspace in agent mode
Agent
GPT-4.1
Ln 9, Col 55 Spaces: 4 UTF-8 CRLF {} Python 3.12.5 Go Live

Here's a comparison between the factorial program with and without functions:

Logic:

- Without functions: The logic is written directly in the main code block. The loop and calculation are performed inline, making the code straightforward but less organized.
- With functions: The logic is encapsulated in a function (`factorial(n)`), separating the calculation from input/output. This improves code structure and clarity.

Reusability:

- Without functions: The factorial calculation cannot be reused elsewhere in the program without copying the code.
- With functions: The `factorial` function can be called multiple times with different arguments, making it reusable in other parts of the program or in other programs.

Execution:

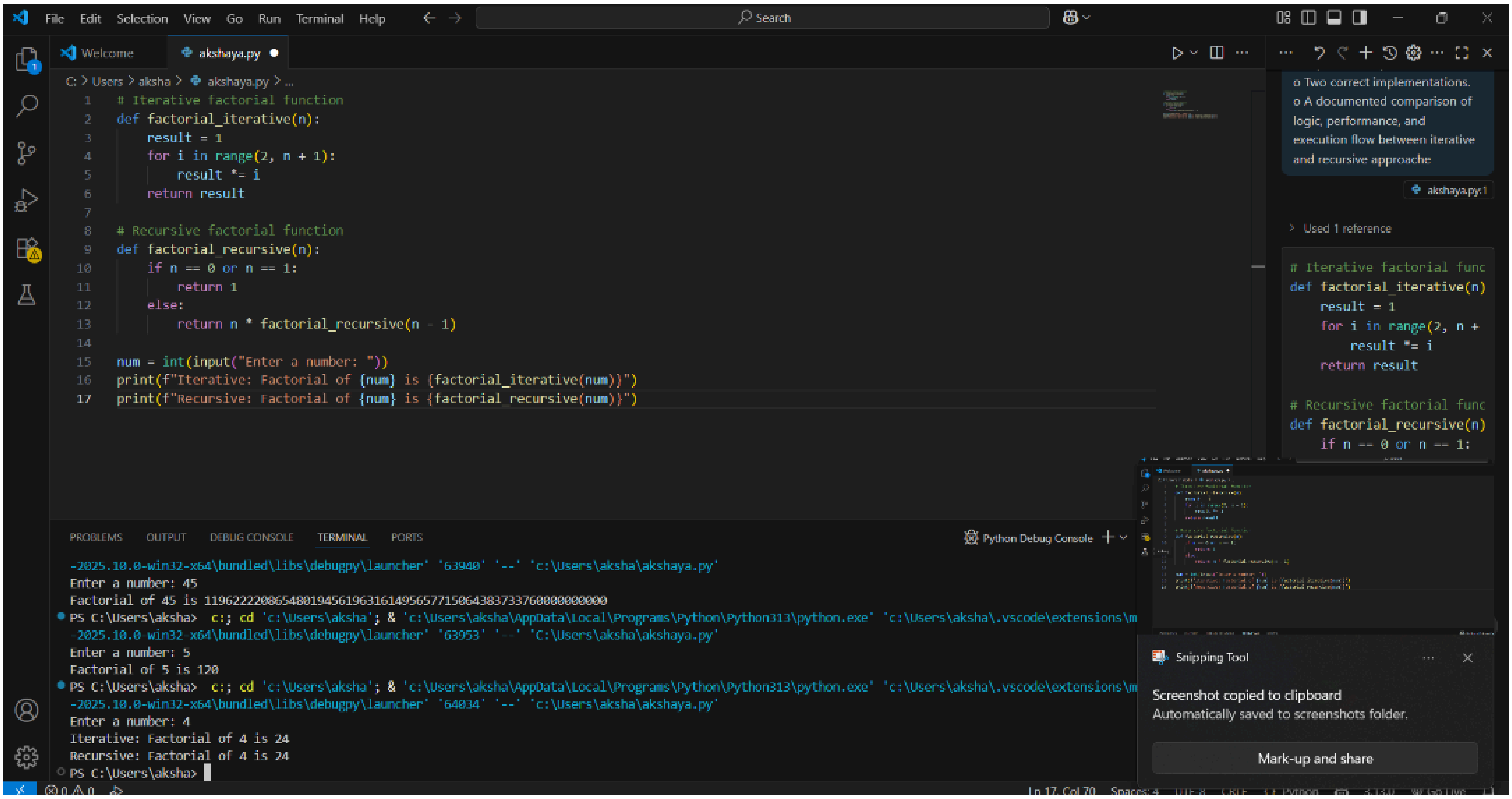
- Without functions: The code executes sequentially from top to bottom. Only one factorial calculation is possible per run.
- With functions: The function can be called as needed, allowing for multiple calculations or integration into larger programs. It also makes testing and debugging easier.

In summary, using functions improves organization, reusability, and flexibility, while direct code is simpler but less versatile.

Task 5: Iterative vs Recursive Factorial

- Description:
  Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function.
- Expected Output:
  - Two correct implementations.
  - A documented comparison of logic, performance, and execution flow between iterative and recursive approaches.

- Prompt : generate both iterative and recursive versions of the factorial function



File Edit Selection View Go Run Terminal Help
Search
Welcome
akshaya.py
C: > Users > aksha > akshaya.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
akshaya.py:1
Used 1 reference
Snipping Tool
Mark-up and share
Ln 17, Col 70

```python
# Iterative factorial function
def factorial_iterative(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# Recursive factorial function
def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial_recursive(n - 1)

num = int(input("Enter a number: "))
print(f"Iterative: Factorial of {num} is {factorial_iterative(num)}")
print(f"Recursive: Factorial of {num} is {factorial_recursive(num)}")
```

```
-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63940' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 45
Factorial of 45 is 119622220865480194561963161495657715064383733760000000000
PS C:\Users\aksha>  c:; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\m
-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63953' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\aksha>  c:; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\m
-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '64034' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 4
Iterative: Factorial of 4 is 24
Recursive: Factorial of 4 is 24
PS C:\Users\aksha> 
```

o Two correct implementations.
o A documented comparison of logic, performance, and execution flow between iterative and recursive approache

```python
# Iterative factorial func
def factorial_iterative(n)
    result = 1
    for i in range(2, n +
        result *= i
    return result

# Recursive factorial func
def factorial_recursive(n)
    if n == 0 or n == 1:
```

Screenshot copied to clipboard
Automatically saved to screenshots folder.