SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 1.5

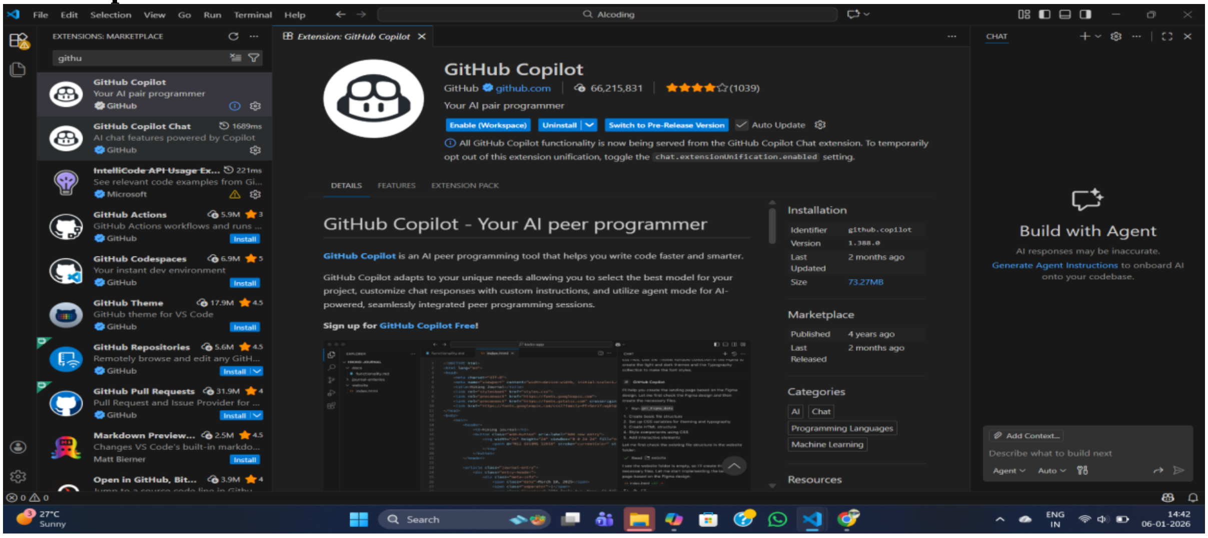
Program	**: B. Tech (CSE)**
Specialization	**:**
Course Title	**: AI Assisted coding**
Course Code	**: 23CS201PC302**
Semester	**: II**
Academic Session	**: 2025-2026**
Name of Student	**:P Abhinav**
Enrollment No.	**: 2403A51L38**
Batch No.	**: 52**
Date	**:10-01-2026**

Submission Starts here

OUTPUT :

SCREENSHOTS:

Task 0: Install and configure GitHub Copilot in VS Code. Take screenshots of each step.



Extension: GitHub Copilot

GitHub Copilot

GitHub github.com | 66,215,831 | (1039)

Your AI pair programmer

Enable (Workspace) | Uninstall | Switch to Pre-Release Version | Auto Update

All GitHub Copilot functionality is now being served from the GitHub Copilot Chat extension. To temporarily opt out of this extension unification, toggle the `chat.extensionUnification.enabled` setting.

GitHub Copilot - Your AI peer programmer

GitHub Copilot is an AI peer programming tool that helps you write code faster and smarter.

GitHub Copilot adapts to your unique needs allowing you to select the best model for your project, customize chat responses with custom instructions, and utilize agent mode for AI-powered, seamlessly integrated peer programming sessions.

Sign up for GitHub Copilot Free!

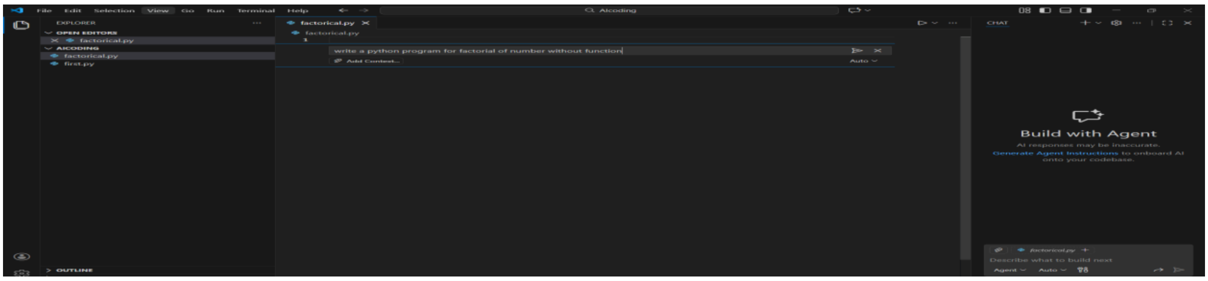
Installation	
Identifier	github.copilot
Version	1.388.0
Last Updated	2 months ago
Size	73.27MB

Marketplace	
Published	4 years ago
Last Released	2 months ago

Categories: AI, Chat, Programming Languages, Machine Learning

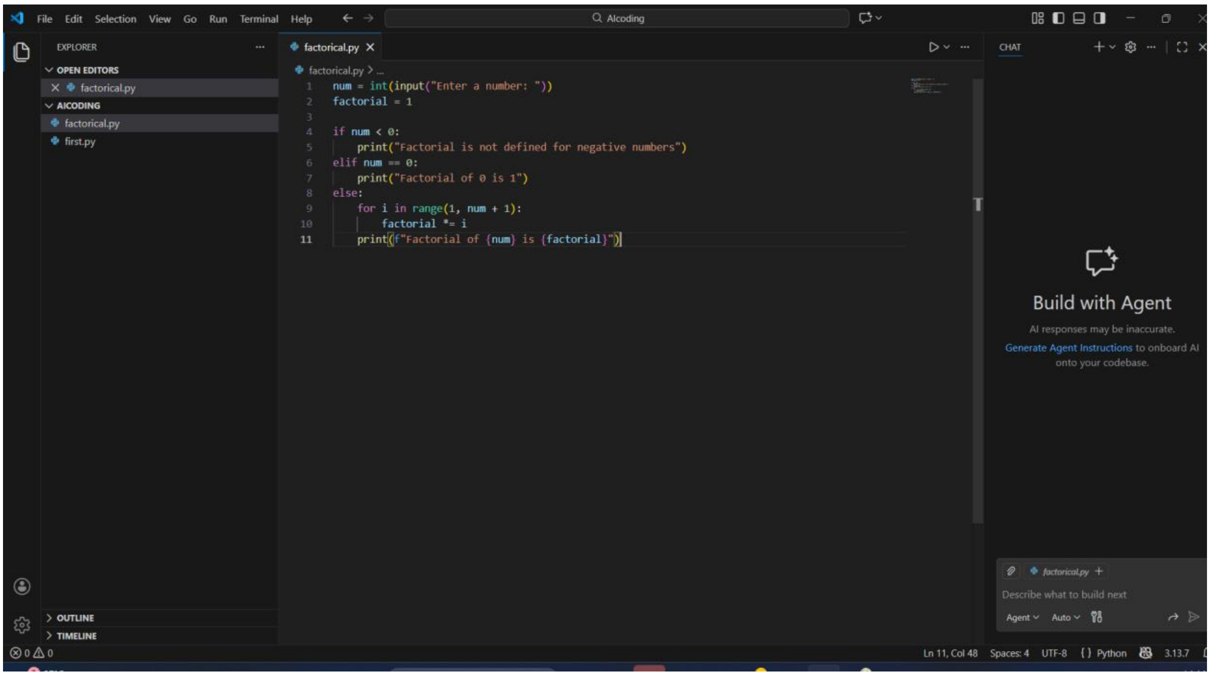
Task1: Task Description

Use GitHub Copilot to generate a Python program that computes a mathematical product-based value (factorial-like logic) directly in the main execution flow, without using any user-defined functions.



write a python program for factorial of number without function

File Edit Selection View Go Run Terminal Help
Alcoding
EXPLORER
OPEN EDITORS
factorical.py
AICODING
factorical.py
first.py
Add Context...
Auto
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
OUTLINE



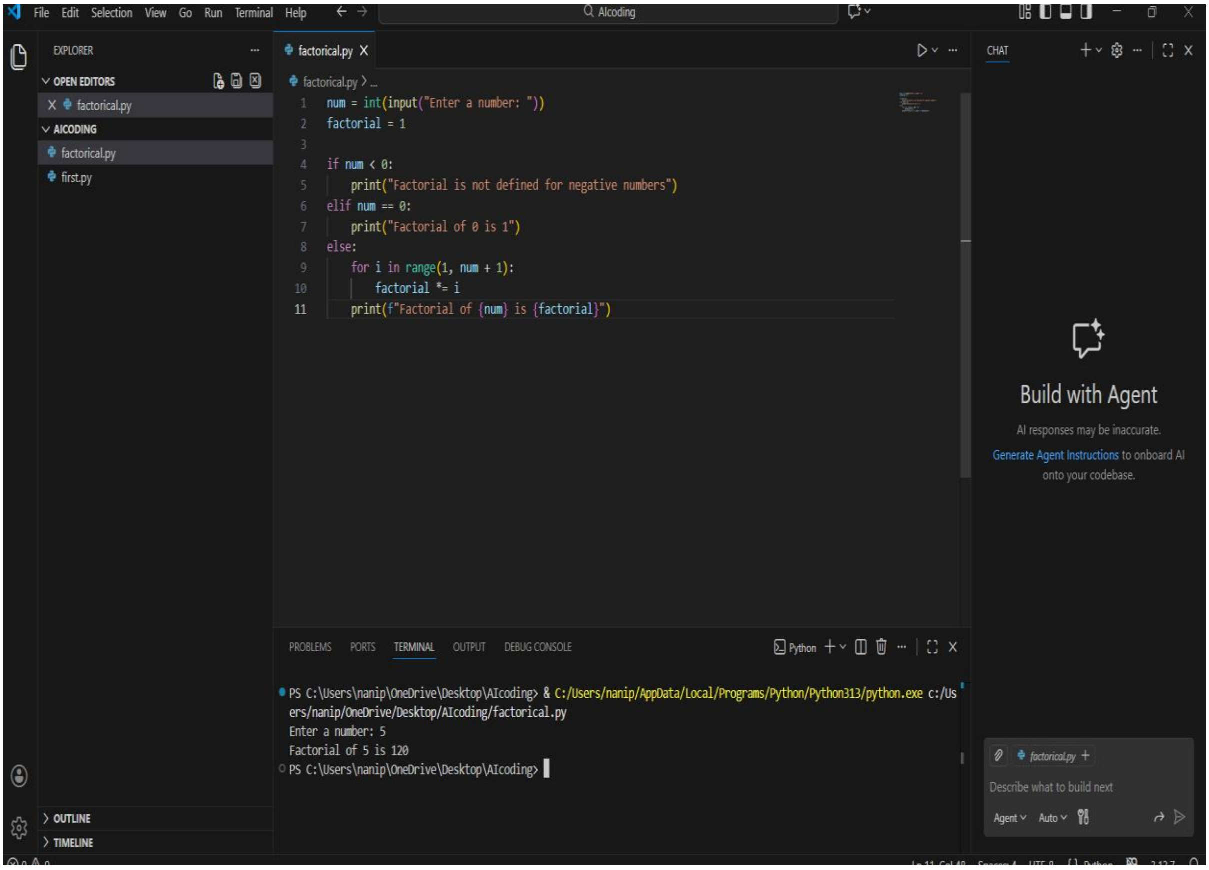
```python
num = int(input("Enter a number: "))
factorial = 1

if num < 0:
    print("Factorial is not defined for negative numbers")
elif num == 0:
    print("Factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial *= i
    print(f"Factorial of {num} is {factorial}")
```

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build next



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

if num < 0:
    print("Factorial is not defined for negative numbers")
elif num == 0:
    print("Factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial *= i
    print(f"Factorial of {num} is {factorial}")
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\nanip\OneDrive\Desktop\AIcoding>
```

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build next

- ❖ **The Copilot is very helpful because we can generate code by just giving a prompt in Copilot Chat (ctrl + I)**

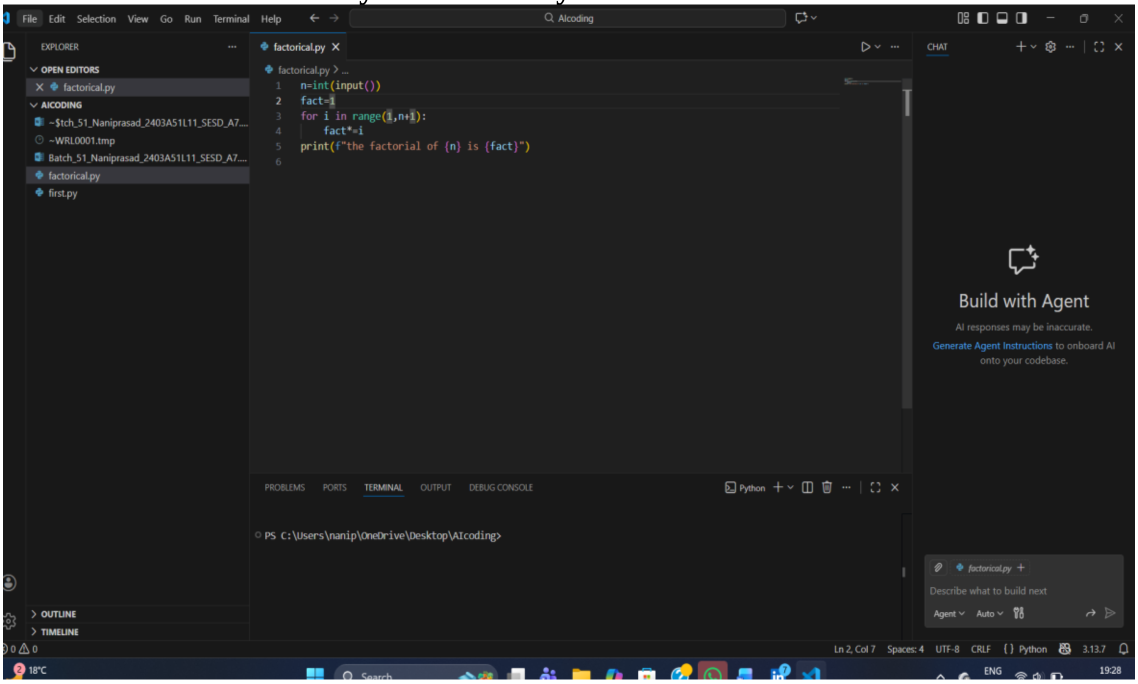
- ❖ **The code generated was as requested in the prompt**

# TASK - 2

Task Description

Analyze the code generated in Task 1 and use Copilot again to:

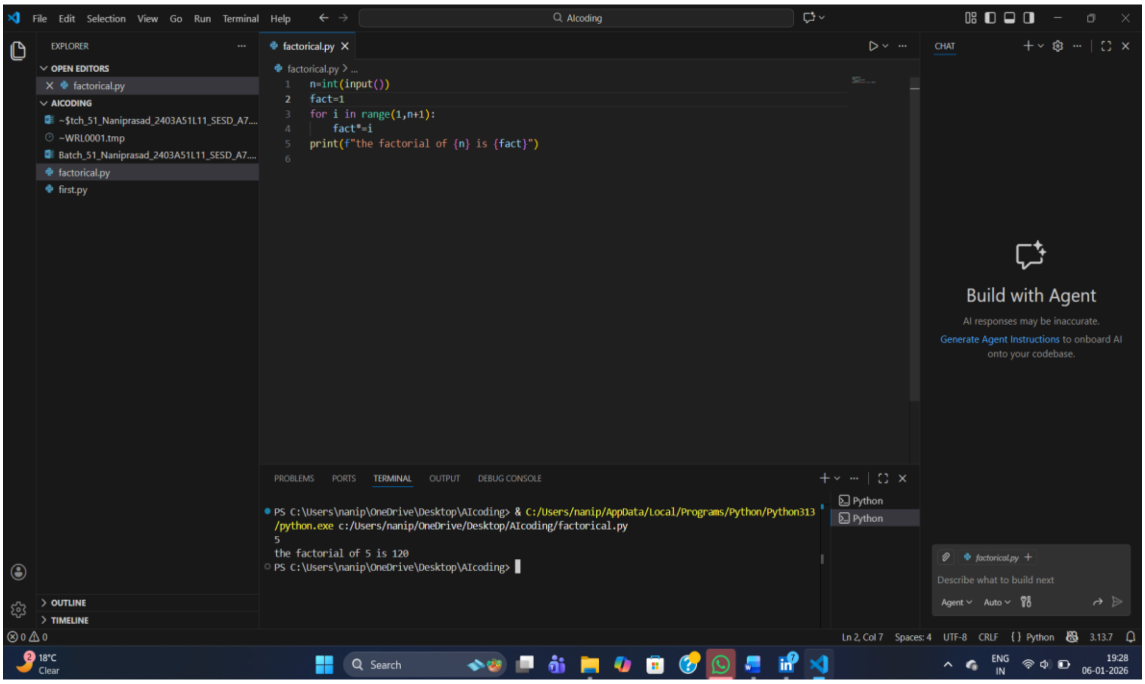
- ❖ **Reduce unnecessary variables**
- ❖ **Improve loop clarity**
- ❖ **Enhance readability and efficiency**



```python
n=int(input())
fact=1
for i in range(1,n+1):
    fact*=i
print(f"the factorial of {n} is {fact}")
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding>
```

File Edit Selection View Go Run Terminal Help
AIcoding
EXPLORER
OPEN EDITORS
factorical.py
AICODING
~$tch_51_Naniprasad_2403A51L11_SESD_A7....
~WRL0001.tmp
Batch_51_Naniprasad_2403A51L11_SESD_A7....
factorical.py
first.py
OUTLINE
TIMELINE
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Agent
Auto
PROBLEMS
PORTS
TERMINAL
OUTPUT
DEBUG CONSOLE
Python
Ln 2, Col 7
Spaces: 4
UTF-8
CRLF
{ } Python
3.13.7
18°C
Search
ENG
19:28



```python
n=int(input())
fact=1
for i in range(1,n+1):
    fact*=i
print(f"the factorial of {n} is {fact}")
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313
/python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py
5
the factorial of 5 is 120
PS C:\Users\nanip\OneDrive\Desktop\AIcoding>
```

# What was improved?

- **Shorter multiplication statement**
- **factorial = factorial * i → factorial *= i**

- Removed unnecessary comment

❖ The loop logic is self-explanatory, so the comment was removed.

❖ # Why the new version is better?

❖ Readability
❖ *= is clearer and more concise.

- Fewer lines and less clutter make the code easier to read.

❖ Maintainability
- Cleaner code is easier to modify and debug.
- Reduced redundancy lowers the chance of mistakes.

❖ Performance
- Performance is effectively the same.

❖ *= is marginally optimized at the bytecode level, but the difference is negligible.

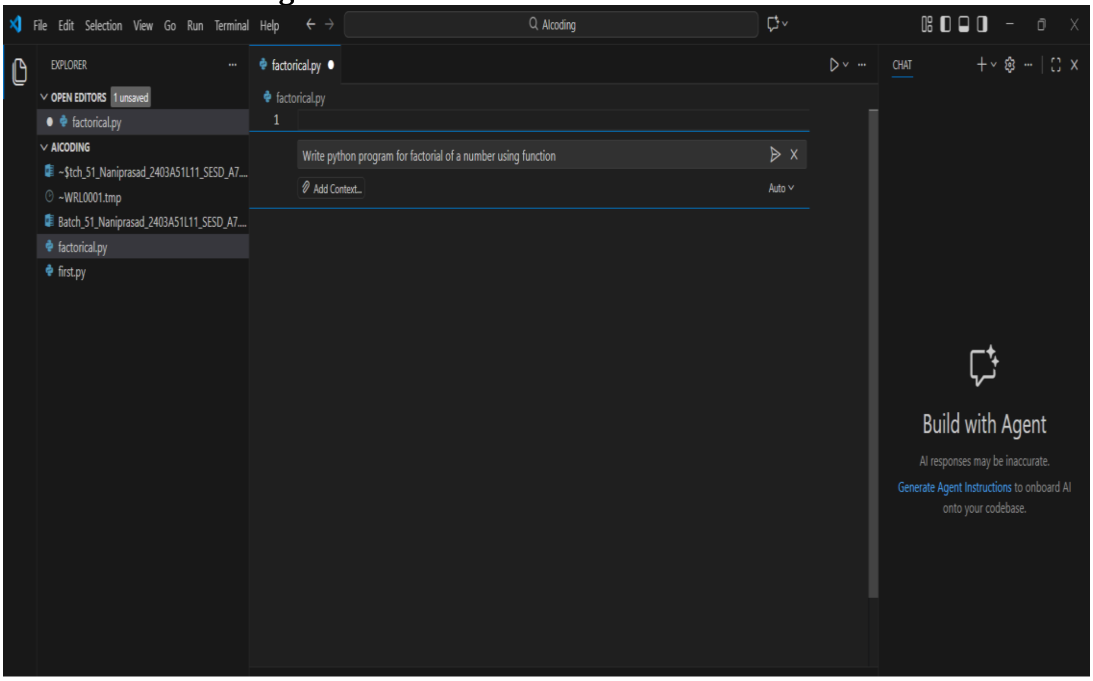
# Task3

Task Description

Use GitHub Copilot to generate a modular version of the program by:

❖ Creating a user-defined function

❖ Calling the function from the main block

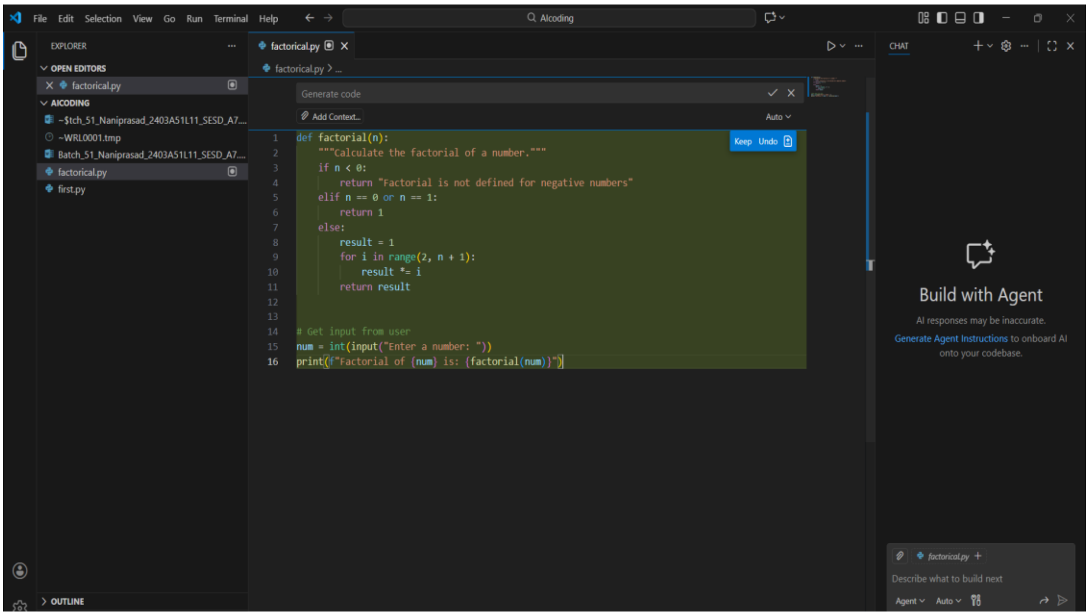


Write python program for factorial of a number using function

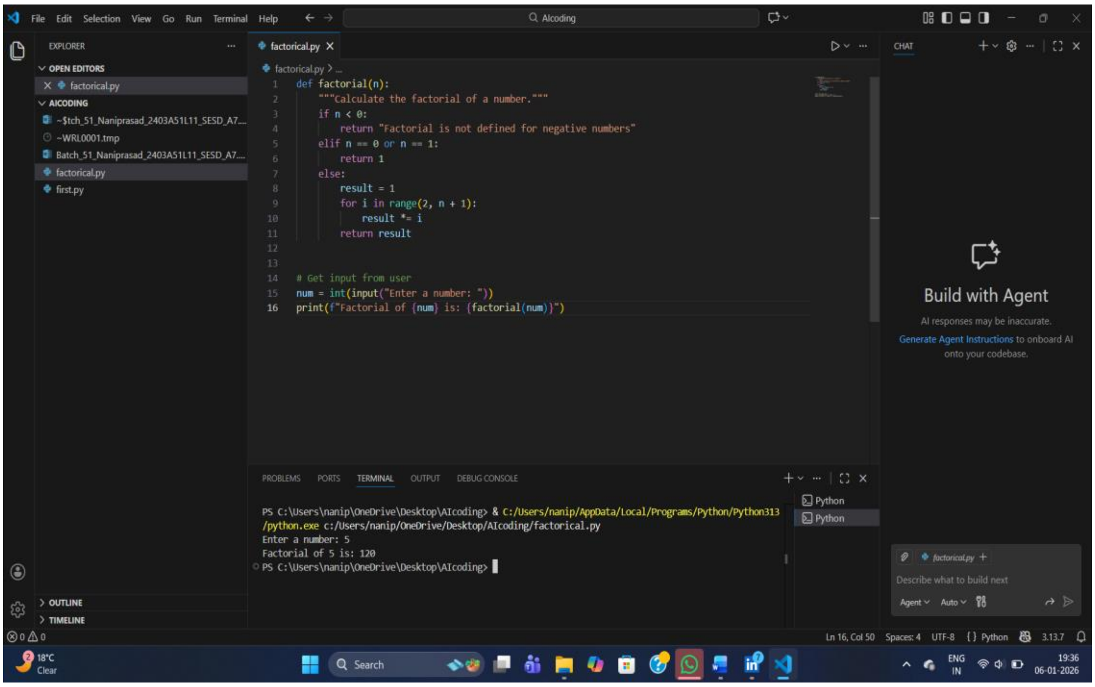
Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.



```python
def factorial(n):
    """Calculate the factorial of a number."""
    if n < 0:
        return "Factorial is not defined for negative numbers"
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result


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



```python
def factorial(n):
    """Calculate the factorial of a number."""
    if n < 0:
        return "Factorial is not defined for negative numbers"
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result


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

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313
/python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py
Enter a number: 5
Factorial of 5 is: 120
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> 
```

Modularity improves reusability by:

Allowing the factorial() function to be reused in multiple programs without rewriting code.

Making the program easier to test, update, and debug.

Improving code organization, where logic is separated from input/output handling.

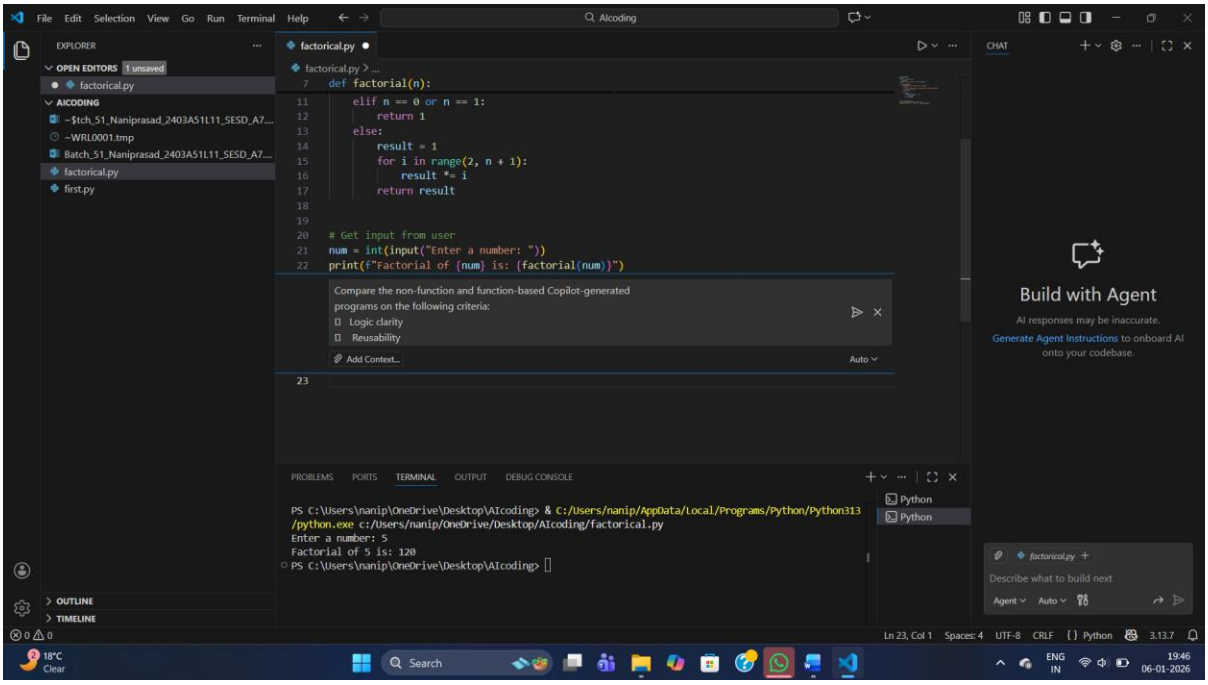
Supporting scalability, as the same function can be extended or integrated into larger projects.

#Task4

Task Description

Compare the non-function and function-based Copilot-generated programs on the following criteria:

- ❖ Logic clarity
- ❖ Reusability
- ❖ Debugging ease
- ❖ Suitability for large projects
- ❖ AI dependency risk

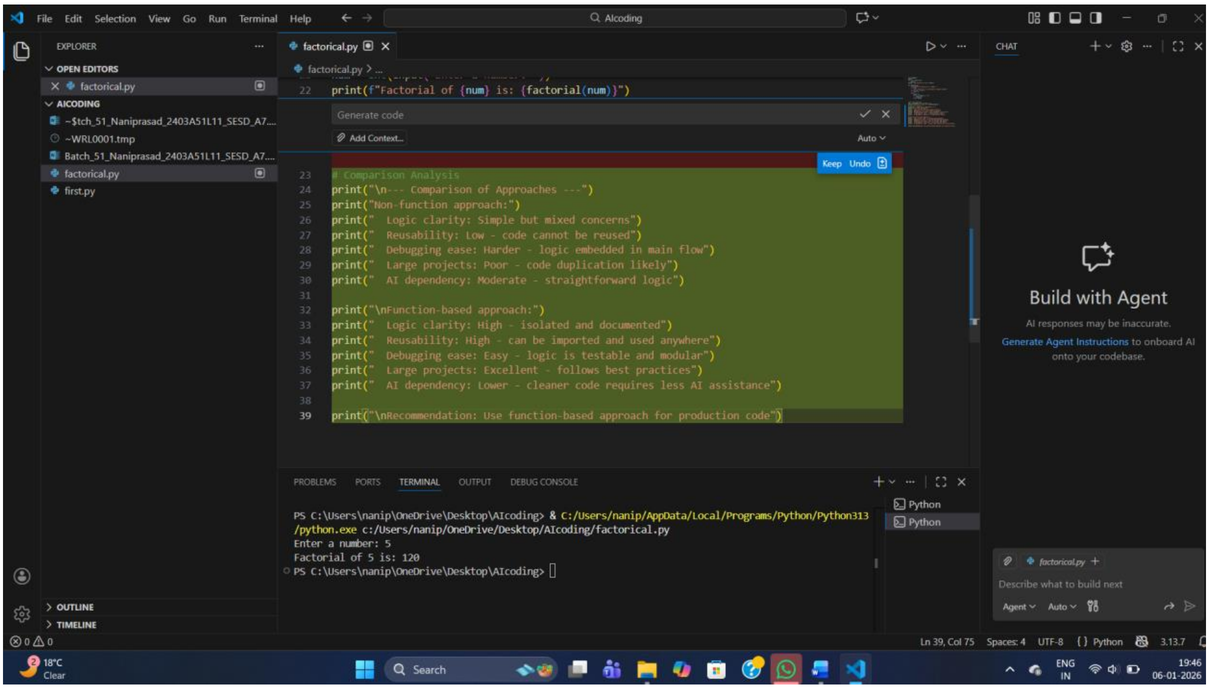


```python
def factorial(n):
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result


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

Compare the non-function and function-based Copilot-generated programs on the following criteria:
□ Logic clarity
□ Reusability

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313
/python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py
Enter a number: 5
Factorial of 5 is: 120
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> 
```

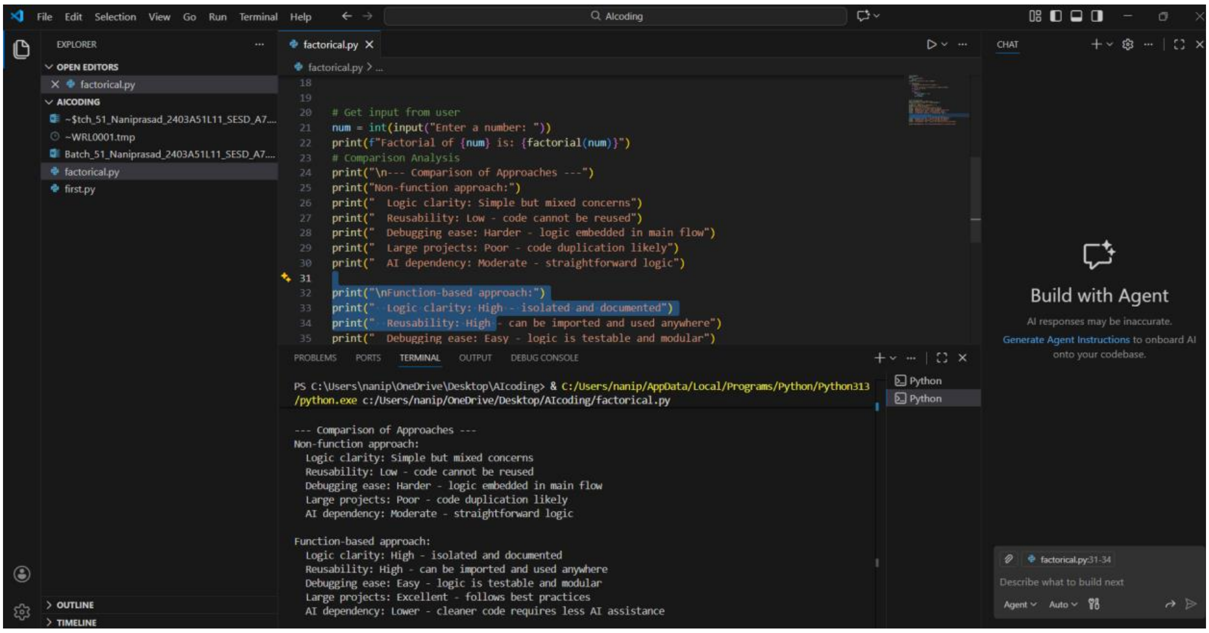
factorical.py
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Ln 23, Col 1
19:46
06-01-2026



```python
print(f"Factorial of {num} is: {factorial(num)}")
# Comparison Analysis
print("\n--- Comparison of Approaches ---")
print("Non-function approach:")
print("  Logic clarity: Simple but mixed concerns")
print("  Reusability: Low - code cannot be reused")
print("  Debugging ease: Harder - logic embedded in main flow")
print("  Large projects: Poor - code duplication likely")
print("  AI dependency: Moderate - straightforward logic")

print("\nFunction-based approach:")
print("  Logic clarity: High - isolated and documented")
print("  Reusability: High - can be imported and used anywhere")
print("  Debugging ease: Easy - logic is testable and modular")
print("  Large projects: Excellent - follows best practices")
print("  AI dependency: Lower - cleaner code requires less AI assistance")

print("\nRecommendation: Use function-based approach for production code")
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313
/python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py
Enter a number: 5
Factorial of 5 is: 120
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> 
```

Generate code
Keep
Undo
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Ln 39, Col 75
19:46
06-01-2026



File Edit Selection View Go Run Terminal Help
AIcoding
EXPLORER
OPEN EDITORS
factorical.py
AICODING
~$tch_51_Naniprasad_2403A51L11_SESD_A7....
~WRL0001.tmp
Batch_51_Naniprasad_2403A51L11_SESD_A7....
factorical.py
first.py
OUTLINE
TIMELINE
factorical.py > ...
PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE
Python
Python
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
factorical.py:31-34
Describe what to build next
Agent Auto

```python
# Get input from user
num = int(input("Enter a number: "))
print(f"Factorial of {num} is: {factorial(num)}")
# Comparison Analysis
print("\n--- Comparison of Approaches ---")
print("Non-function approach:")
print("  Logic clarity: Simple but mixed concerns")
print("  Reusability: Low - code cannot be reused")
print("  Debugging ease: Harder - logic embedded in main flow")
print("  Large projects: Poor - code duplication likely")
print("  AI dependency: Moderate - straightforward logic")

print("\nFunction-based approach:")
print("  Logic clarity: High - isolated and documented")
print("  Reusability: High - can be imported and used anywhere")
print("  Debugging ease: Easy - logic is testable and modular")
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313
/python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py

--- Comparison of Approaches ---
Non-function approach:
  Logic clarity: Simple but mixed concerns
  Reusability: Low - code cannot be reused
  Debugging ease: Harder - logic embedded in main flow
  Large projects: Poor - code duplication likely
  AI dependency: Moderate - straightforward logic

Function-based approach:
  Logic clarity: High - isolated and documented
  Reusability: High - can be imported and used anywhere
  Debugging ease: Easy - logic is testable and modular
  Large projects: Excellent - follows best practices
  AI dependency: Lower - cleaner code requires less AI assistance
```

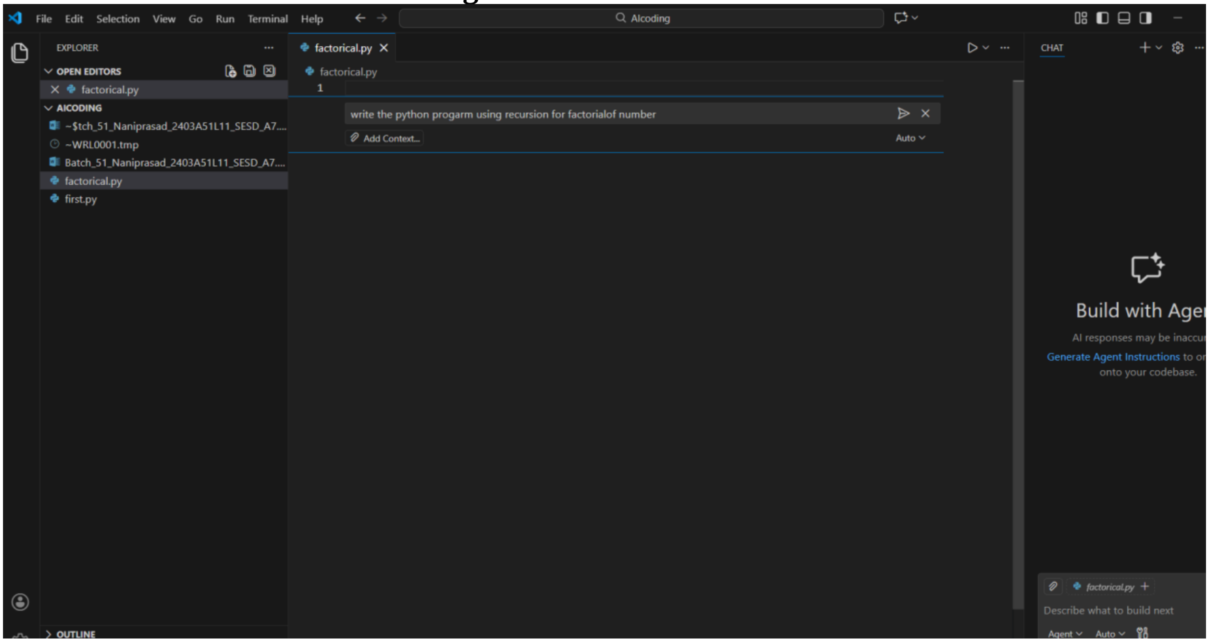
# TASK - 5

Task Description

Prompt Copilot to generate:

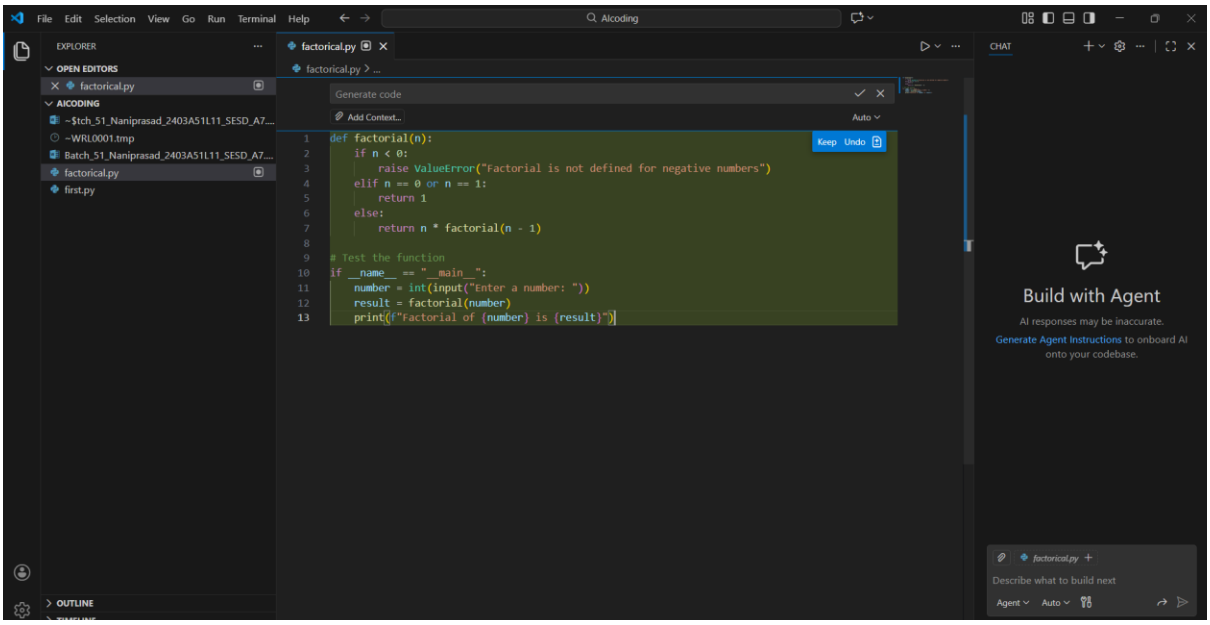
An iterative version of the logic

A recursive version of the same logic

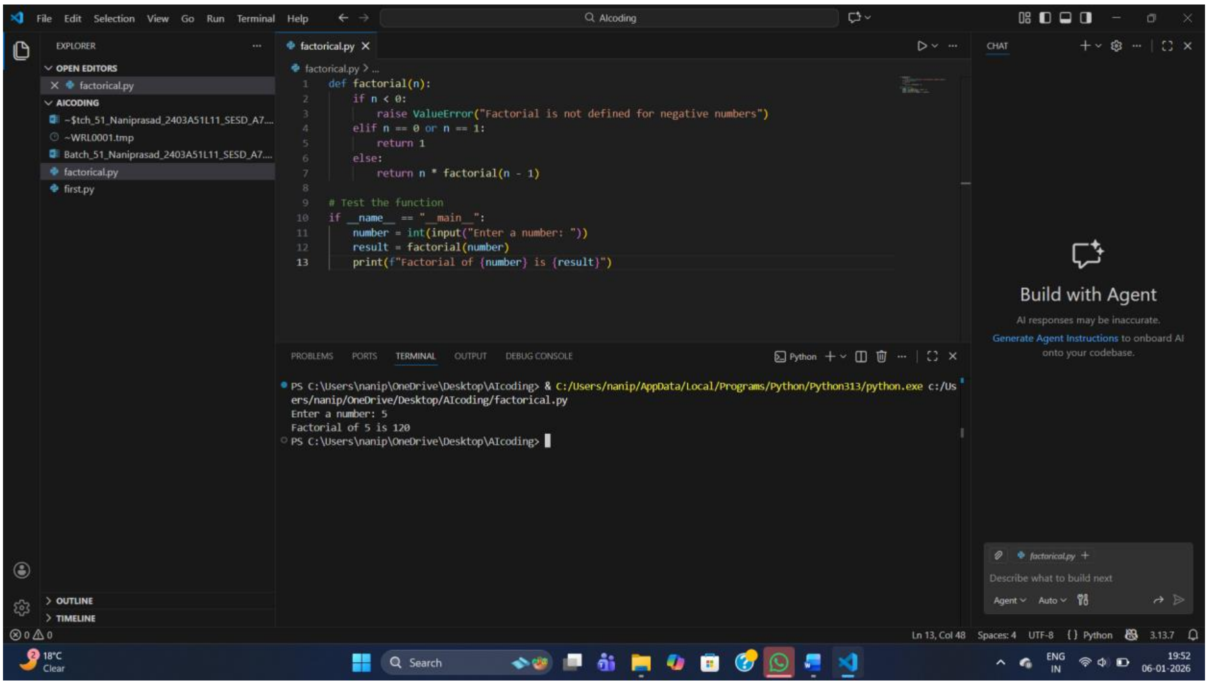


File Edit Selection View Go Run Terminal Help
AIcoding
EXPLORER
OPEN EDITORS
factorical.py
AICODING
~$tch_51_Naniprasad_2403A51L11_SESD_A7....
~WRL0001.tmp
Batch_51_Naniprasad_2403A51L11_SESD_A7....
factorical.py
first.py
OUTLINE
factorical.py
Add Context...
Auto
CHAT
Build with Age
AI responses may be inaccu
Generate Agent Instructions to o
onto your codebase.
factorical.py
Describe what to build next
Agent Auto

write the python progarm using recursion for factorialof number


Generate code
Add Context...
Auto
Keep
Undo
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next

```python
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    elif n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Test the function
if __name__ == "__main__":
    number = int(input("Enter a number: "))
    result = factorial(number)
    print(f"Factorial of {number} is {result}")
```


PROBLEMS
PORTS
TERMINAL
OUTPUT
DEBUG CONSOLE
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next

```python
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    elif n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Test the function
if __name__ == "__main__":
    number = int(input("Enter a number: "))
    result = factorial(number)
    print(f"Factorial of {number} is {result}")
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Us
ers/nanip/OneDrive/Desktop/AIcoding/factorical.py
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\nanip\OneDrive\Desktop\AIcoding>
```

Explanation:

How the Function Works

1. **Negative number check**
   Factorials are not defined for negative numbers. If the input is negative, the program raises an error message.
2. **Base cases**
   For 0 and 1, the factorial is defined as 1. This acts as the stopping condition for recursion.
3. **Recursive case**
   For numbers greater than 1, the function calls itself with n-1. This recursive process continues until it reaches the base case.
   Example:

- **To compute 5!, the function calculates 5\times 4!.**
  Then 4! becomes 4\times 3!, and so on, until it reaches 1!.

- **Main Program Flow**
- **The program asks the user to enter a number.**
- **It then calls the factorial function with that number.**
- **Finally, it prints the result in a clear message.**
- **Example Execution**

If the user enters 5:

- **The recursive calls break it down step by step until reaching 1.**
- **The final result is 120.**

So the program outputs: *Factorial of 5 is 120*.

Summary

This program demonstrates:

- **Recursion (function calling itself).**
- **Error handling (for negative inputs).**
- **Base cases (to stop recursion).**
- **User interaction (taking input and displaying output).**