SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 1

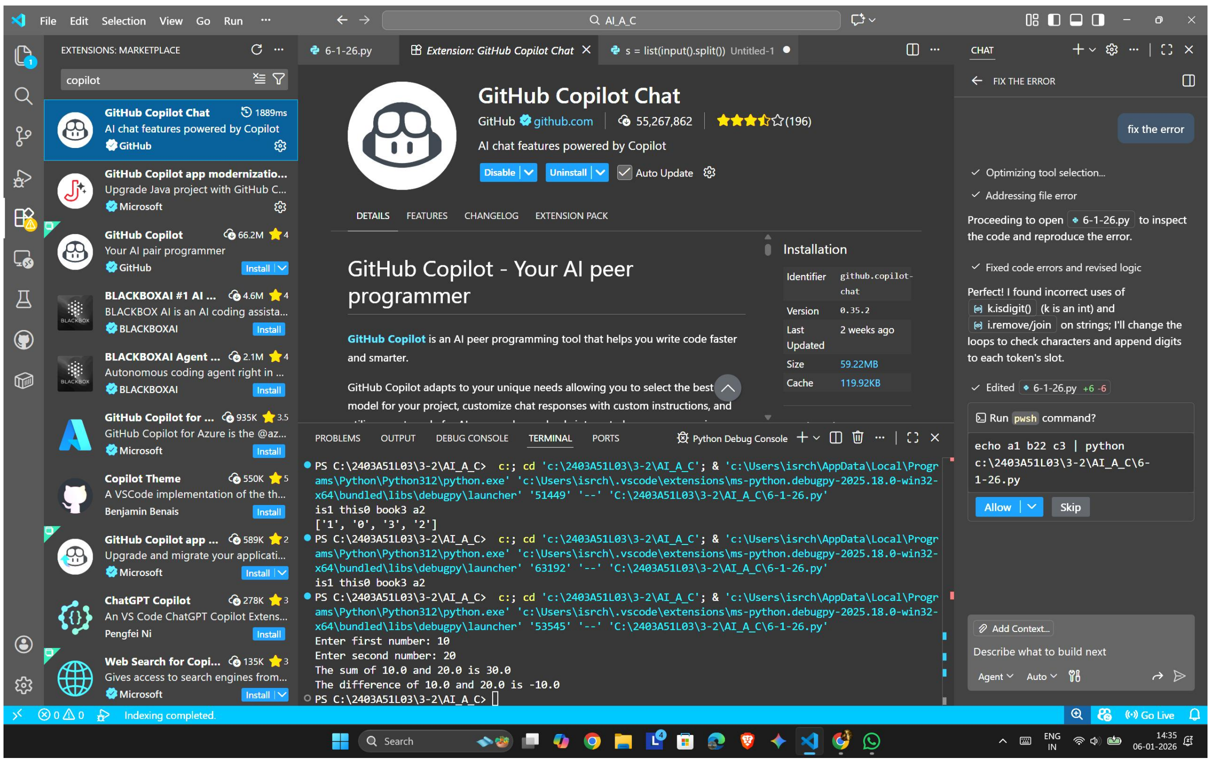
Program	**: B. Tech (CSE)**
Specialization	**:**
Course Title	**: AI Assisted Coding**
Course Code	**: 23CS002PC304**
Semester	**: II**
Academic Session	**: 2025-2026**
Name of Student	**: I.Sathwik Rajeshwara Chary**
Enrollment No.	**: 2403A51L03**
Batch No.	**: 51**
Date	**: 06/01/26**

Submission Starts here

Screenshots:

# TASK - 0

- Install "Github Copilot" in Visual Studio Code



GitHub Copilot Chat

GitHub github.com 55,267,862 (196)

AI chat features powered by Copilot

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

Installation	
Identifier	github.copilot-chat
Version	0.35.2
Last Updated	2 weeks ago
Size	59.22MB
Cache	119.92KB

```
PS C:\2403A51L03\3-2\AI_A_C>  c:; cd 'c:\2403A51L03\3-2\AI_A_C'; & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51449' '--' 'C:\2403A51L03\3-2\AI_A_C\6-1-26.py'
is1 this0 book3 a2
['1', '0', '3', '2']
PS C:\2403A51L03\3-2\AI_A_C>  c:; cd 'c:\2403A51L03\3-2\AI_A_C'; & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63192' '--' 'C:\2403A51L03\3-2\AI_A_C\6-1-26.py'
is1 this0 book3 a2
PS C:\2403A51L03\3-2\AI_A_C>  c:; cd 'c:\2403A51L03\3-2\AI_A_C'; & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53545' '--' 'C:\2403A51L03\3-2\AI_A_C\6-1-26.py'
Enter first number: 10
Enter second number: 20
The sum of 10.0 and 20.0 is 30.0
The difference of 10.0 and 20.0 is -10.0
PS C:\2403A51L03\3-2\AI_A_C>
```

FIX THE ERROR

fix the error

✓ Optimizing tool selection...

✓ Addressing file error

Proceeding to open 6-1-26.py to inspect the code and reproduce the error.

✓ Fixed code errors and revised logic

Perfect! I found incorrect uses of k.isdigit() (k is an int) and i.remove/join on strings; I'll change the loops to check characters and append digits to each token's slot.

✓ Edited 6-1-26.py +6 -6

Run pwsh command?

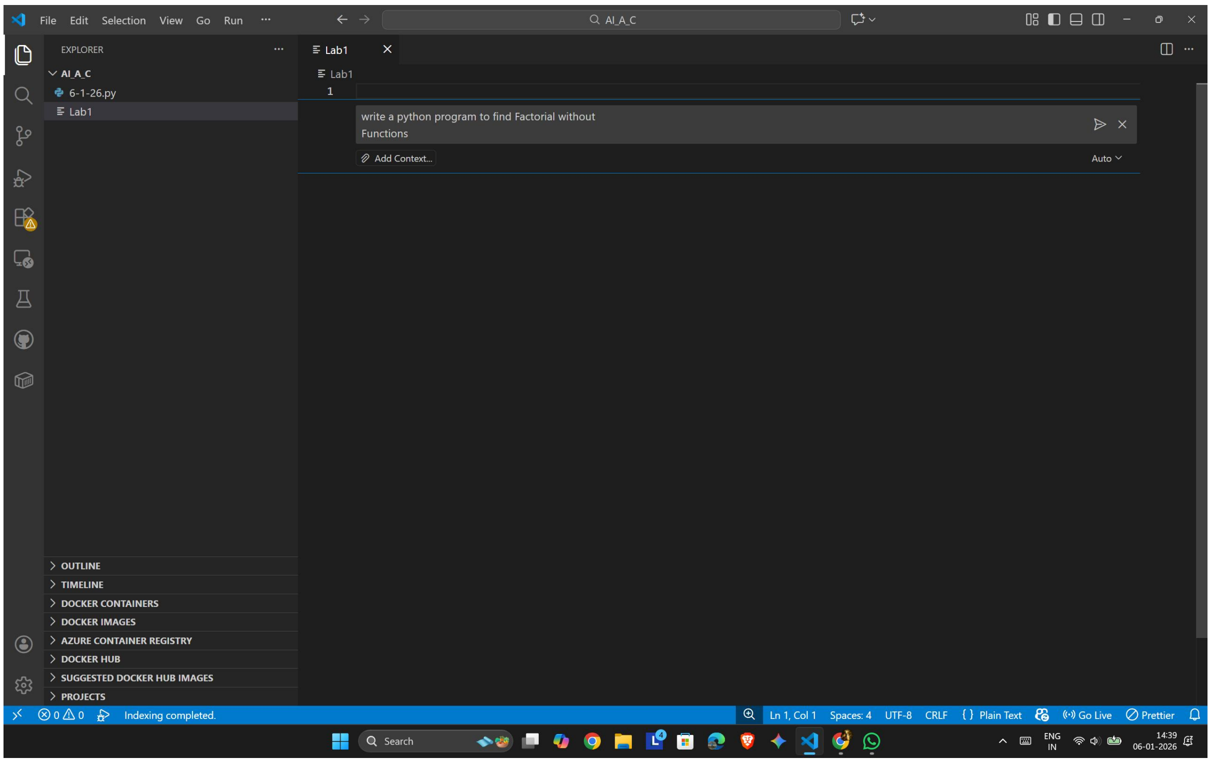
```
echo a1 b22 c3 | python c:\2403A51L03\3-2\AI_A_C\6-1-26.py
```

Describe what to build next

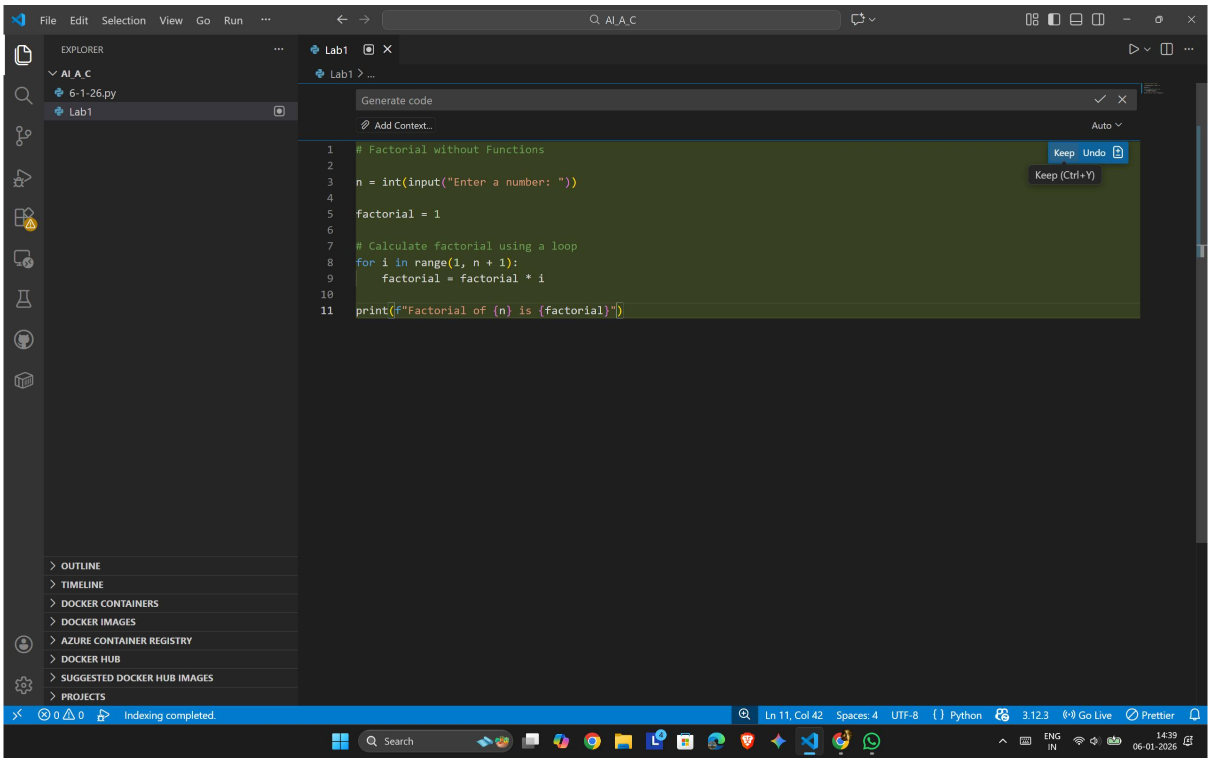
# TASK - 1

- 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.

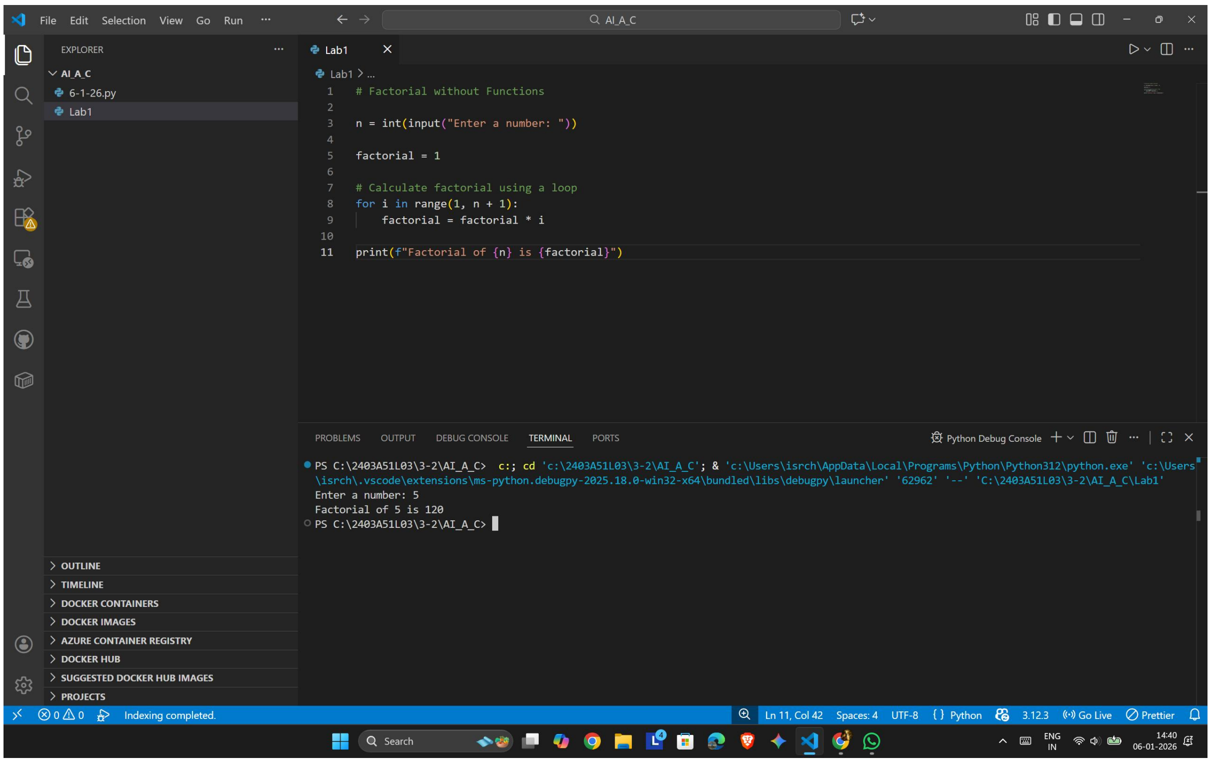


write a python program to find Factorial without Functions



```python
# Factorial without Functions

n = int(input("Enter a number: "))

factorial = 1

# Calculate factorial using a loop
for i in range(1, n + 1):
    factorial = factorial * i

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

# OUTPUT:



```python
# Factorial without Functions

n = int(input("Enter a number: "))

factorial = 1

# Calculate factorial using a loop
for i in range(1, n + 1):
    factorial = factorial * i

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

```
PS C:\2403A51L03\3-2\AI_A_C>  c:; cd 'c:\2403A51L03\3-2\AI_A_C'; & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62962' '--' 'C:\2403A51L03\3-2\AI_A_C\Lab1'
Enter a number: 5
Factorial of 5 is 120
PS C:\2403A51L03\3-2\AI_A_C>
```

- 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

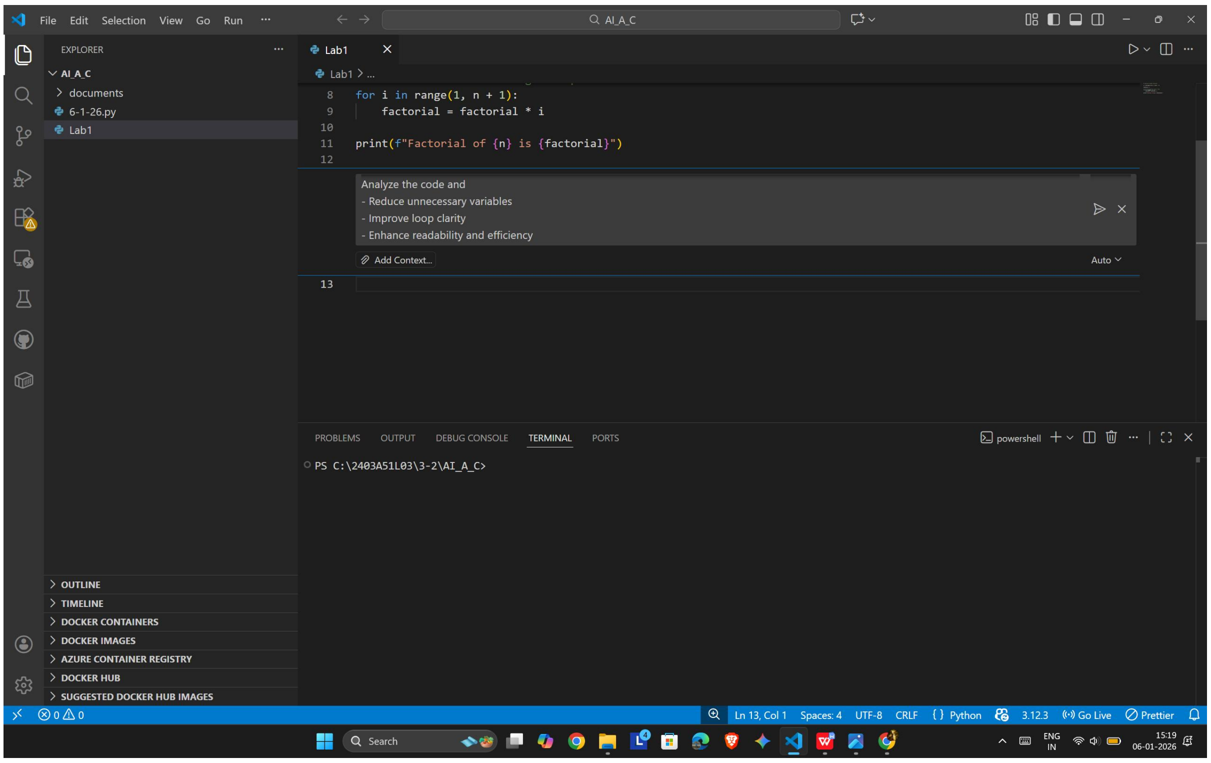
SR UNIVERSITY

# TASK - 2

Task Description

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

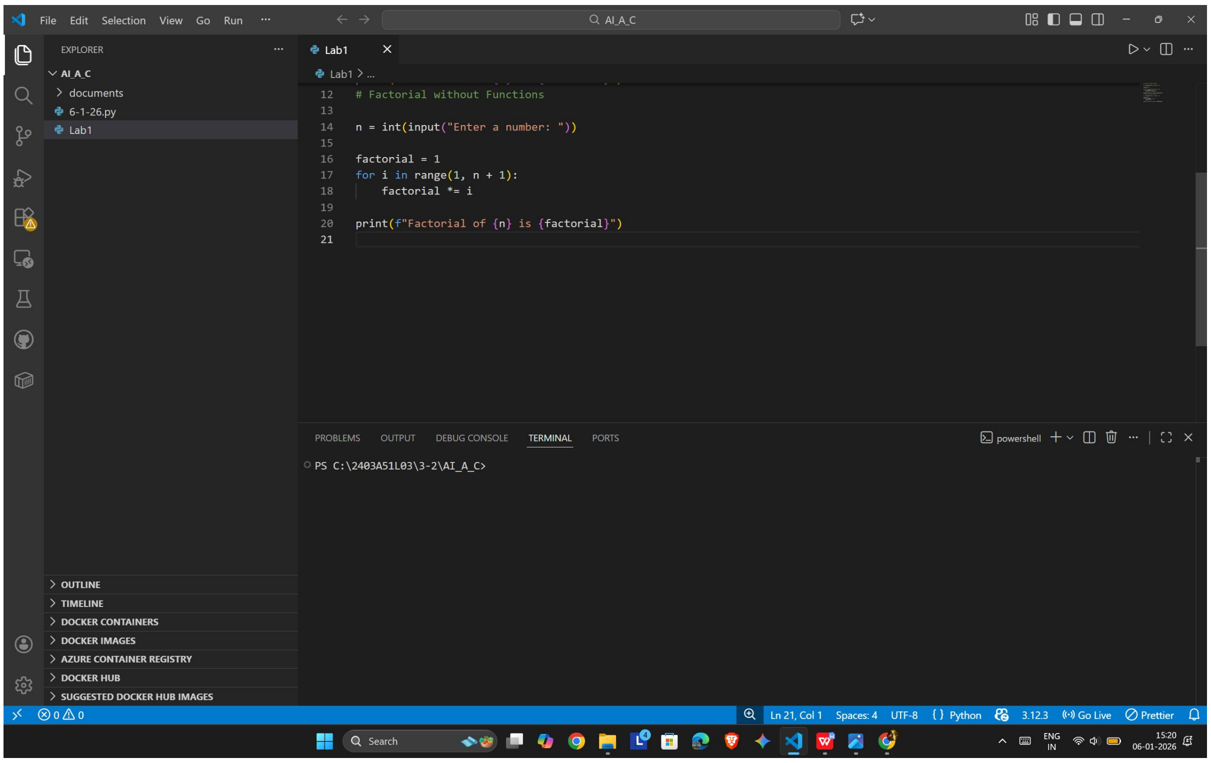
- ➢ **Reduce unnecessary variables**
- ➢ **Improve loop clarity**
- ➢ **Enhance readability and efficiency**



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

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

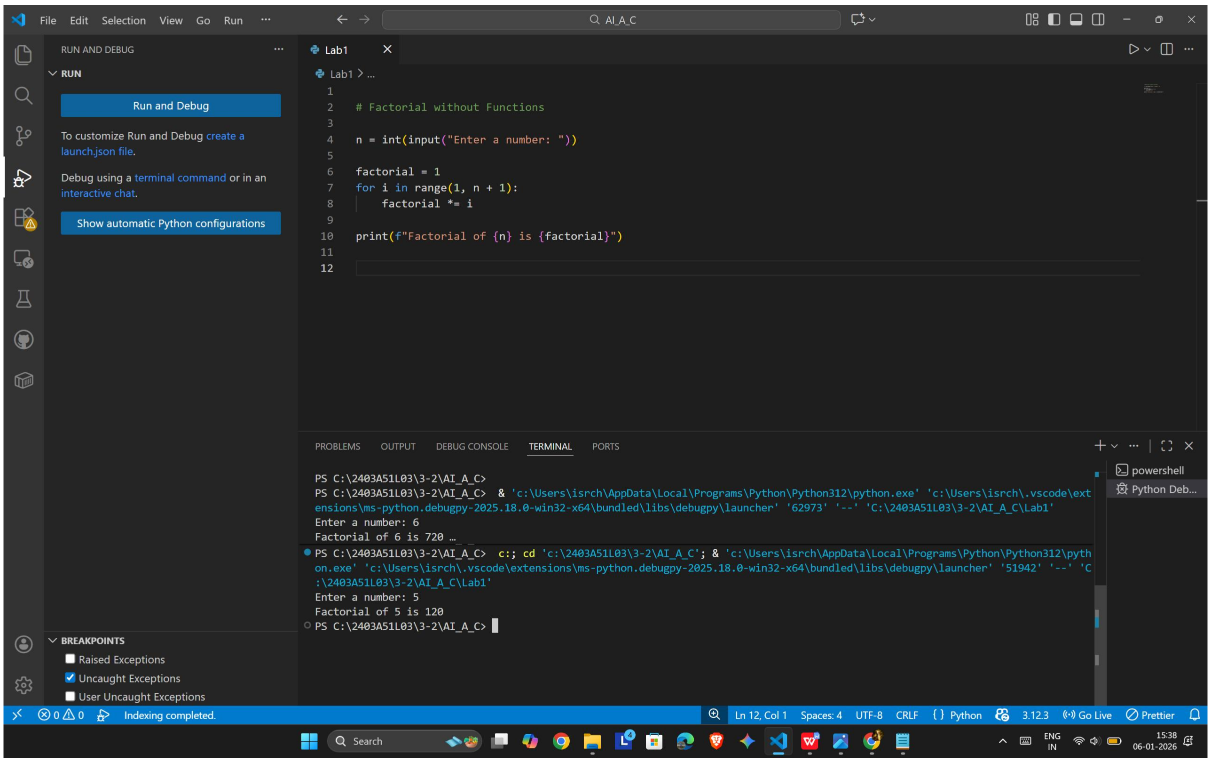
```
Analyze the code and
- Reduce unnecessary variables
- Improve loop clarity
- Enhance readability and efficiency
```

```
PS C:\2403A51L03\3-2\AI_A_C>
```



```
# Factorial without Functions

n = int(input("Enter a number: "))

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

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

```
PS C:\2403A51L03\3-2\AI_A_C>
```



```python
# Factorial without Functions

n = int(input("Enter a number: "))

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

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

```
PS C:\2403A51L03\3-2\AI_A_C>
PS C:\2403A51L03\3-2\AI_A_C>  & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\ext
ensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62973' '--' 'C:\2403A51L03\3-2\AI_A_C\Lab1'
Enter a number: 6
Factorial of 6 is 720 …
PS C:\2403A51L03\3-2\AI_A_C>  c:; cd 'c:\2403A51L03\3-2\AI_A_C'; & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\pyth
on.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51942' '--' 'C
:\2403A51L03\3-2\AI_A_C\Lab1'
Enter a number: 5
Factorial of 5 is 120
PS C:\2403A51L03\3-2\AI_A_C>
```

# 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?

1. Readability
*= is clearer and more concise.

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

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

3. Performance
- Performance is effectively the same.

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

SR UNIVERSITY

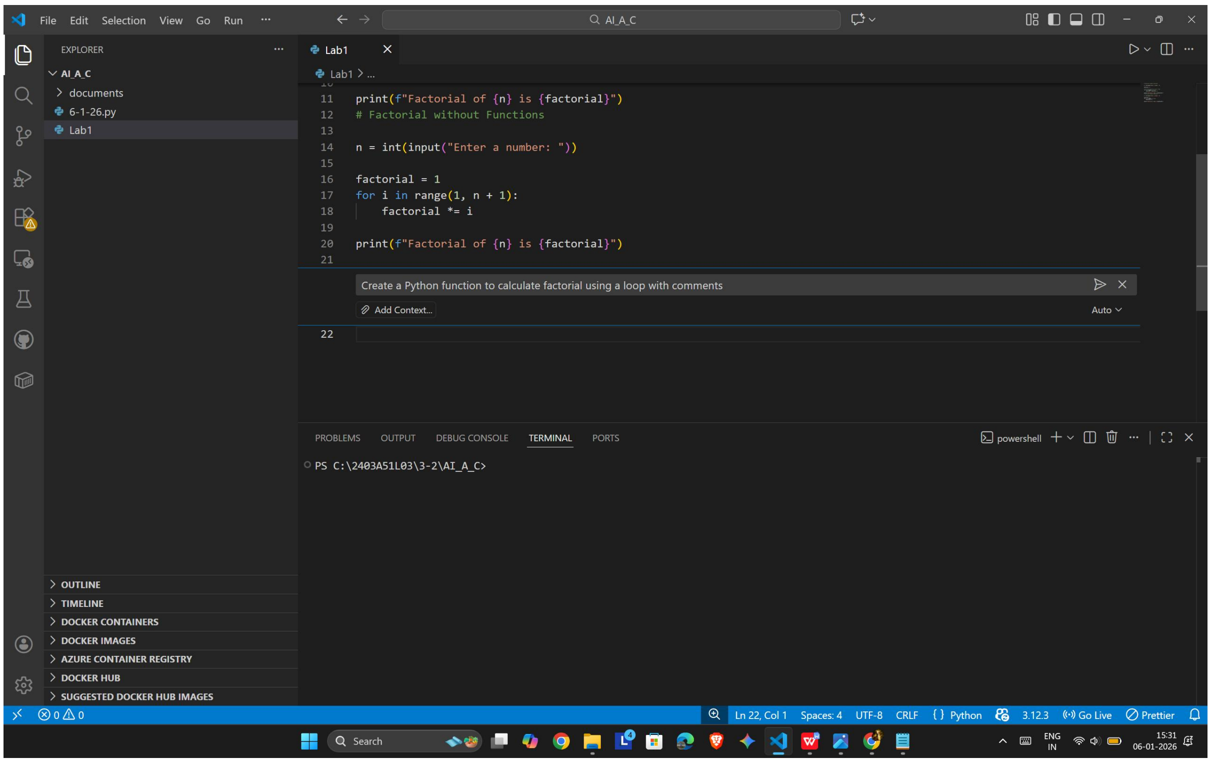
# TASK - 3

Task Description

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

➢ Creating a user-defined function

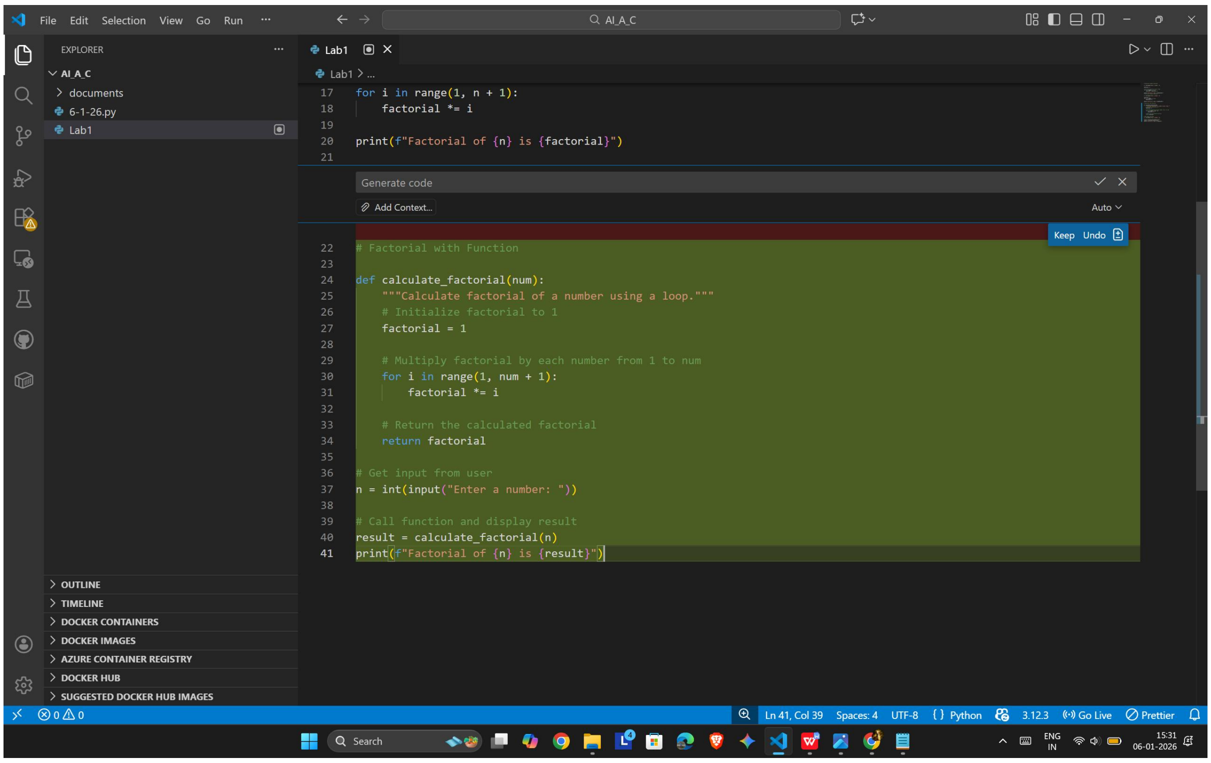
➢ Calling the function from the main block



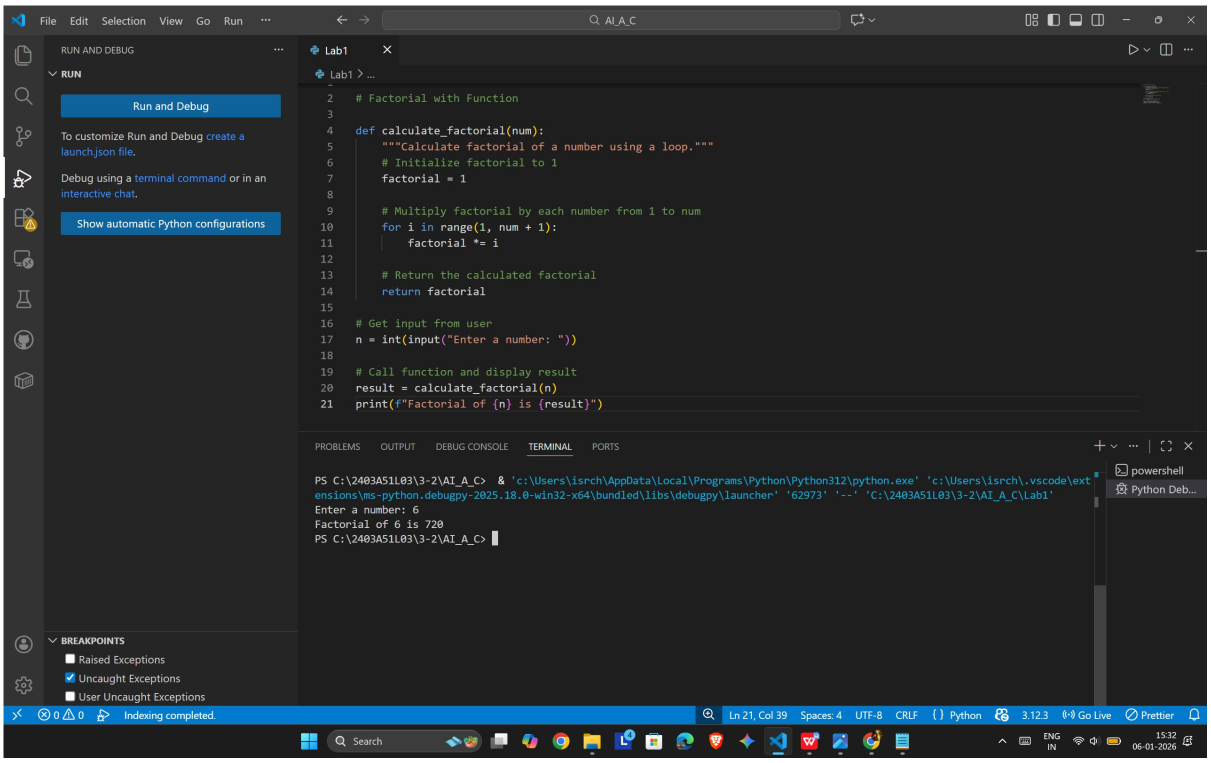
```
print(f"Factorial of {n} is {factorial}")
# Factorial without Functions

n = int(input("Enter a number: "))

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

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

Create a Python function to calculate factorial using a loop with comments

```
PS C:\2403A51L03\3-2\AI_A_C>
```



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

print(f"Factorial of {n} is {factorial}")

# Factorial with Function

def calculate_factorial(num):
    """Calculate factorial of a number using a loop."""
    # Initialize factorial to 1
    factorial = 1

    # Multiply factorial by each number from 1 to num
    for i in range(1, num + 1):
        factorial *= i

    # Return the calculated factorial
    return factorial

# Get input from user
n = int(input("Enter a number: "))

# Call function and display result
result = calculate_factorial(n)
print(f"Factorial of {n} is {result}")
```



```python
# Factorial with Function

def calculate_factorial(num):
    """Calculate factorial of a number using a loop."""
    # Initialize factorial to 1
    factorial = 1

    # Multiply factorial by each number from 1 to num
    for i in range(1, num + 1):
        factorial *= i

    # Return the calculated factorial
    return factorial

# Get input from user
n = int(input("Enter a number: "))

# Call function and display result
result = calculate_factorial(n)
print(f"Factorial of {n} is {result}")
```

```
PS C:\2403A51L03\3-2\AI_A_C>  & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\ext
ensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62973' '--' 'C:\2403A51L03\3-2\AI_A_C\Lab1'
Enter a number: 6
Factorial of 6 is 720
PS C:\2403A51L03\3-2\AI_A_C>
```

- Modularity improves reusability by:

Allowing the calculate_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.

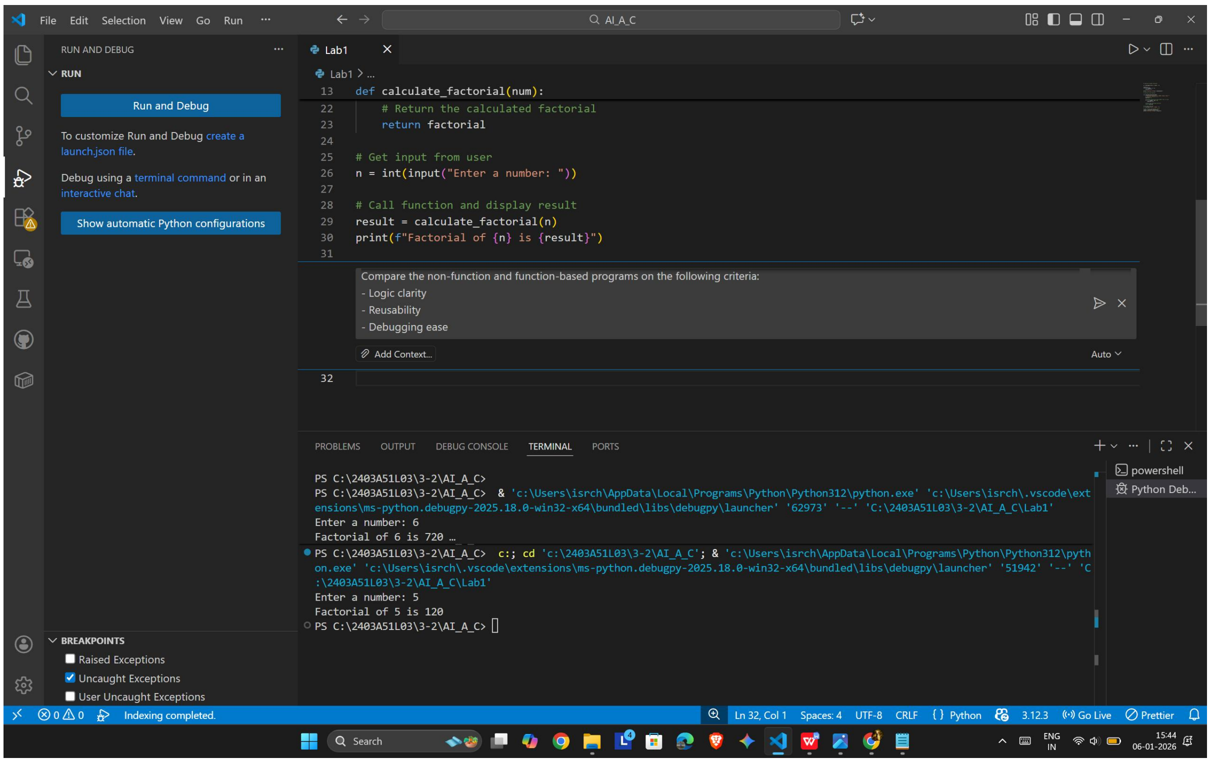
Supporting scalability, as the same function can be extended or integrated into larger projects.

# TASK - 4

Task Description

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

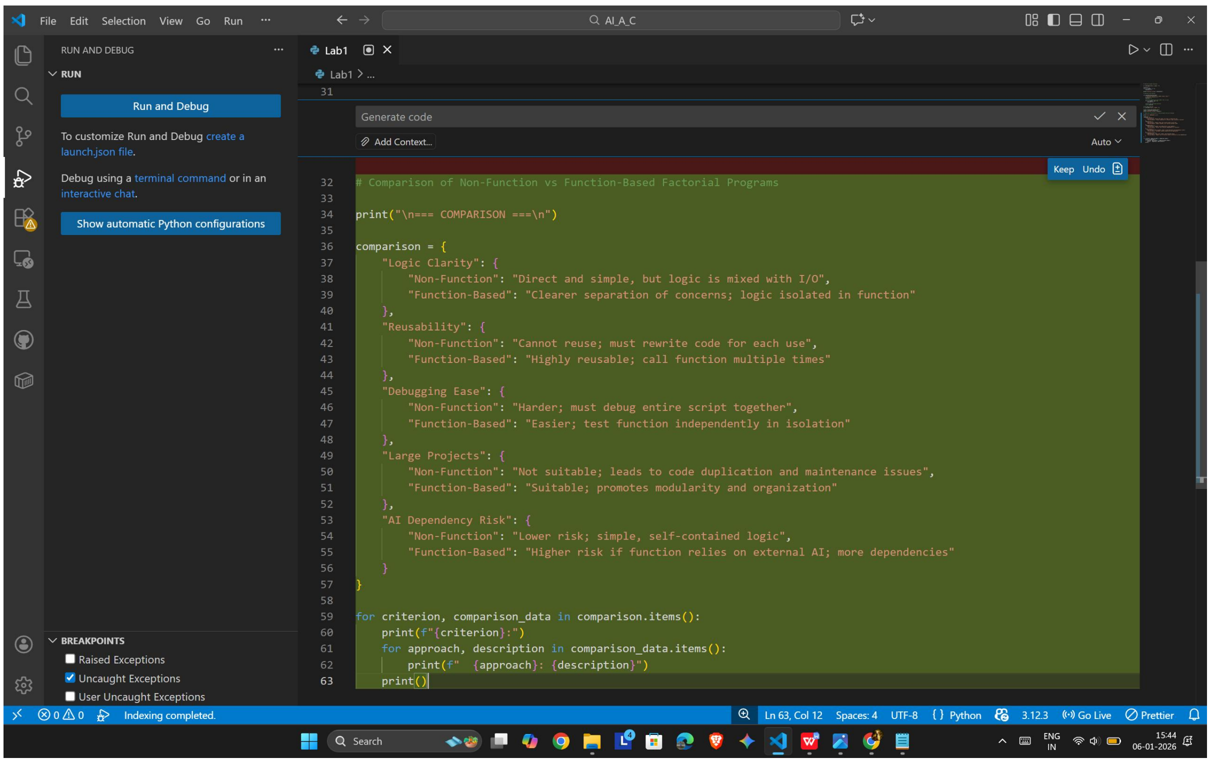
- ➢ Logic clarity
- ➢ Reusability
- ➢ Debugging ease
- ➢ Suitability for large projects
- ➢ AI dependency risk



```python
def calculate_factorial(num):
    # Return the calculated factorial
    return factorial

# Get input from user
n = int(input("Enter a number: "))

# Call function and display result
result = calculate_factorial(n)
print(f"Factorial of {n} is {result}")
```

Compare the non-function and function-based programs on the following criteria:
- Logic clarity
- Reusability
- Debugging ease

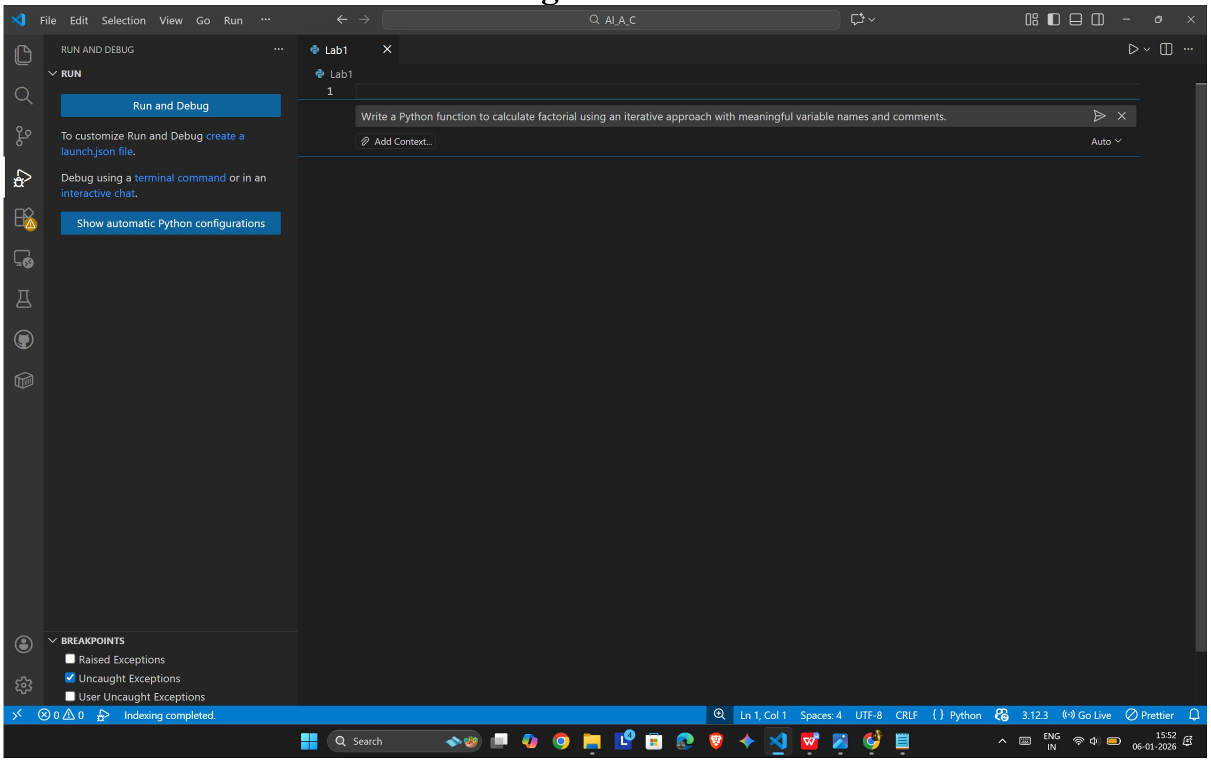
```
PS C:\2403A51L03\3-2\AI_A_C>
PS C:\2403A51L03\3-2\AI_A_C>  & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62973' '--' 'C:\2403A51L03\3-2\AI_A_C\Lab1'
Enter a number: 6
Factorial of 6 is 720 …
PS C:\2403A51L03\3-2\AI_A_C>  c:; cd 'c:\2403A51L03\3-2\AI_A_C'; & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51942' '--' 'C:\2403A51L03\3-2\AI_A_C\Lab1'
Enter a number: 5
Factorial of 5 is 120
PS C:\2403A51L03\3-2\AI_A_C>
```



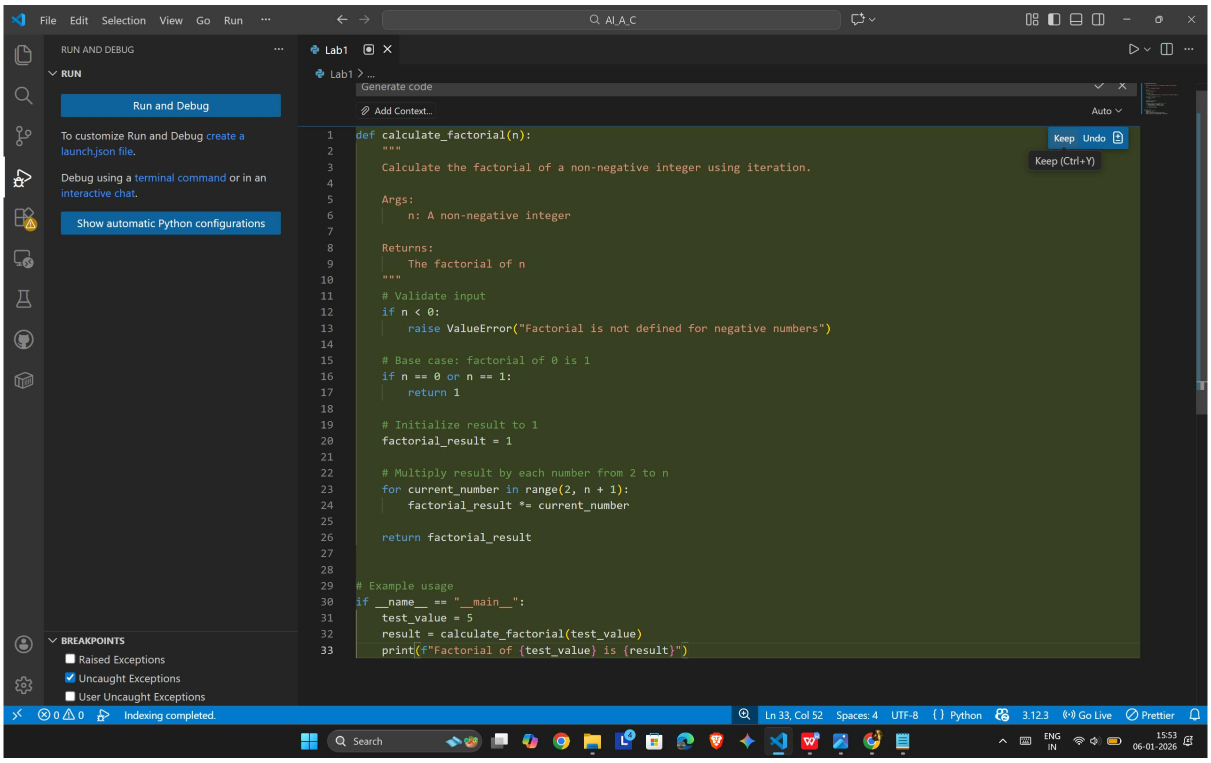
```python
# Comparison of Non-Function vs Function-Based Factorial Programs

print("\n=== COMPARISON ===\n")

comparison = {
    "Logic Clarity": {
        "Non-Function": "Direct and simple, but logic is mixed with I/O",
        "Function-Based": "Clearer separation of concerns; logic isolated in function"
    },
    "Reusability": {
        "Non-Function": "Cannot reuse; must rewrite code for each use",
        "Function-Based": "Highly reusable; call function multiple times"
    },
    "Debugging Ease": {
        "Non-Function": "Harder; must debug entire script together",
        "Function-Based": "Easier; test function independently in isolation"
    },
    "Large Projects": {
        "Non-Function": "Not suitable; leads to code duplication and maintenance issues",
        "Function-Based": "Suitable; promotes modularity and organization"
    },
    "AI Dependency Risk": {
        "Non-Function": "Lower risk; simple, self-contained logic",
        "Function-Based": "Higher risk if function relies on external AI; more dependencies"
    }
}

for criterion, comparison_data in comparison.items():
    print(f"{criterion}:")
    for approach, description in comparison_data.items():
        print(f"  {approach}: {description}")
    print()
```

# TASK - 5

Task Description
Prompt Copilot to generate:
An iterative version of the logic
A recursive version of the same logic



Write a Python function to calculate factorial using an iterative approach with meaningful variable names and comments.

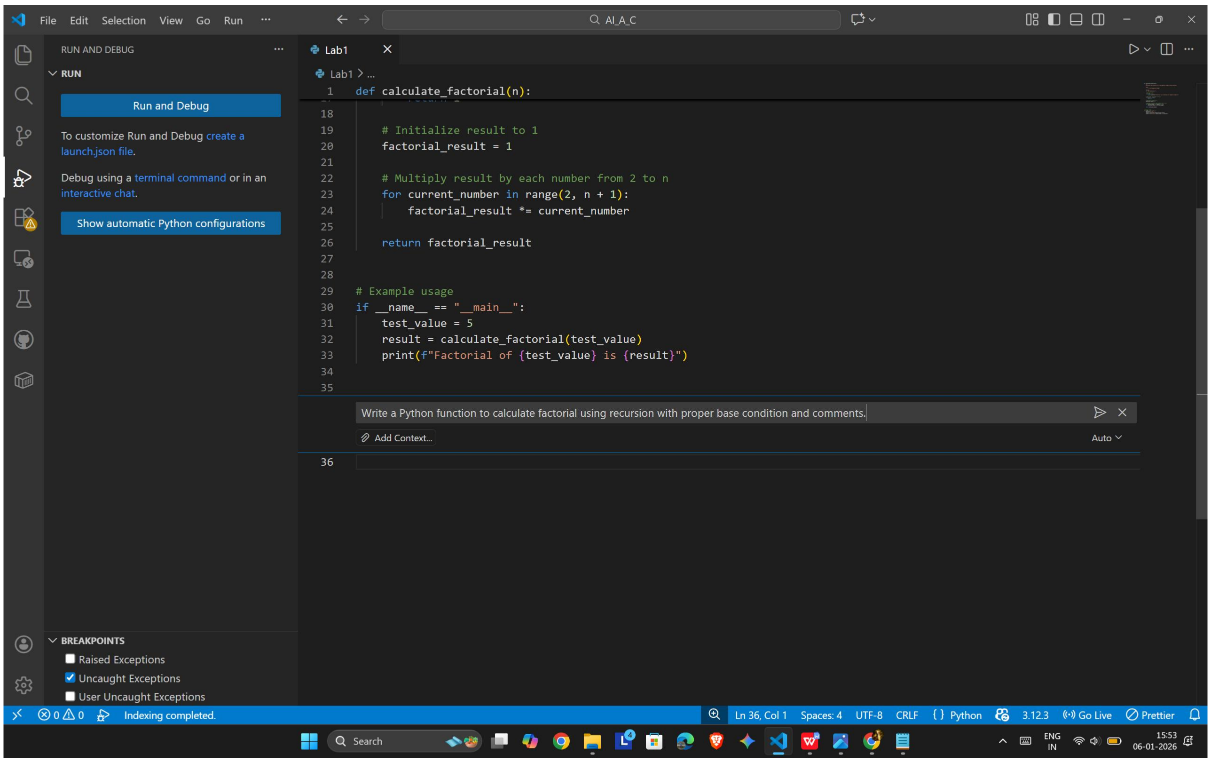


To customize Run and Debug create a launch.json file.

Debug using a terminal command or in an interactive chat.

Generate code

```python
def calculate_factorial(n):
    """
    Calculate the factorial of a non-negative integer using iteration.

    Args:
        n: A non-negative integer

    Returns:
        The factorial of n
    """
    # Validate input
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")

    # Base case: factorial of 0 is 1
    if n == 0 or n == 1:
        return 1

    # Initialize result to 1
    factorial_result = 1

    # Multiply result by each number from 2 to n
    for current_number in range(2, n + 1):
        factorial_result *= current_number

    return factorial_result


# Example usage
if __name__ == "__main__":
    test_value = 5
    result = calculate_factorial(test_value)
    print(f"Factorial of {test_value} is {result}")
```

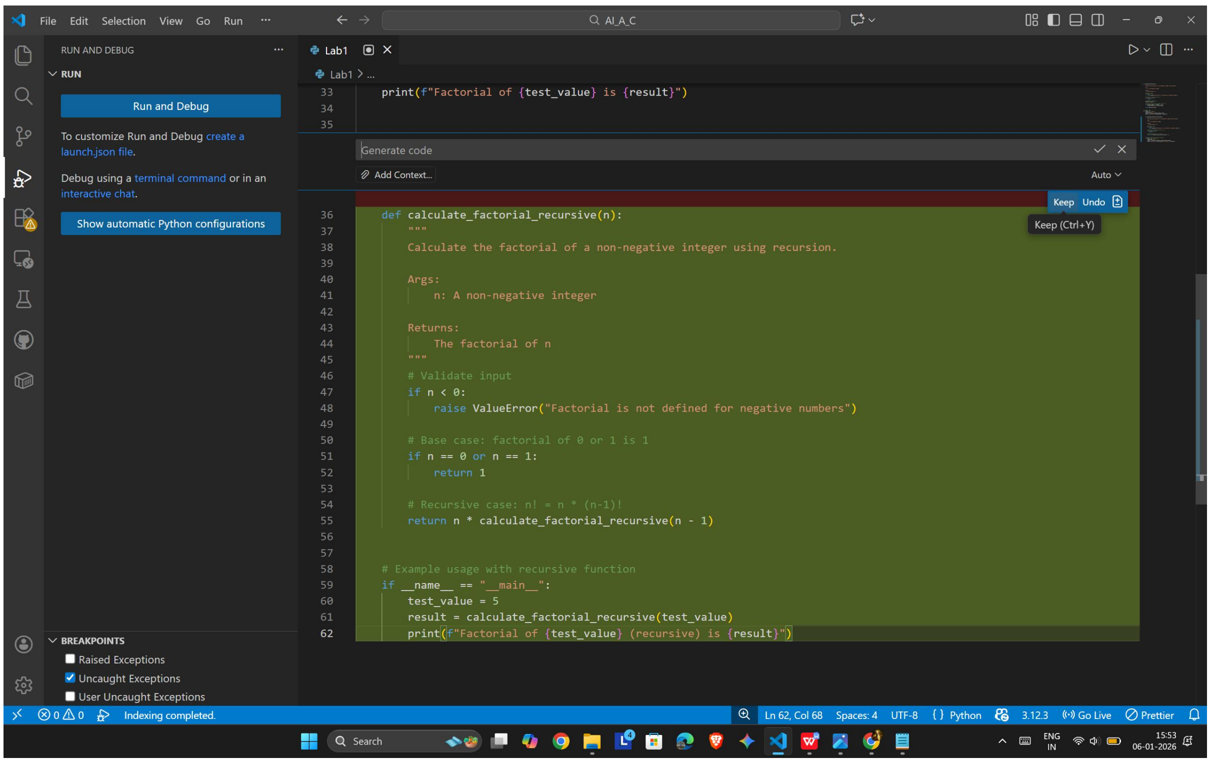


To customize Run and Debug create a launch.json file.

Debug using a terminal command or in an interactive chat.

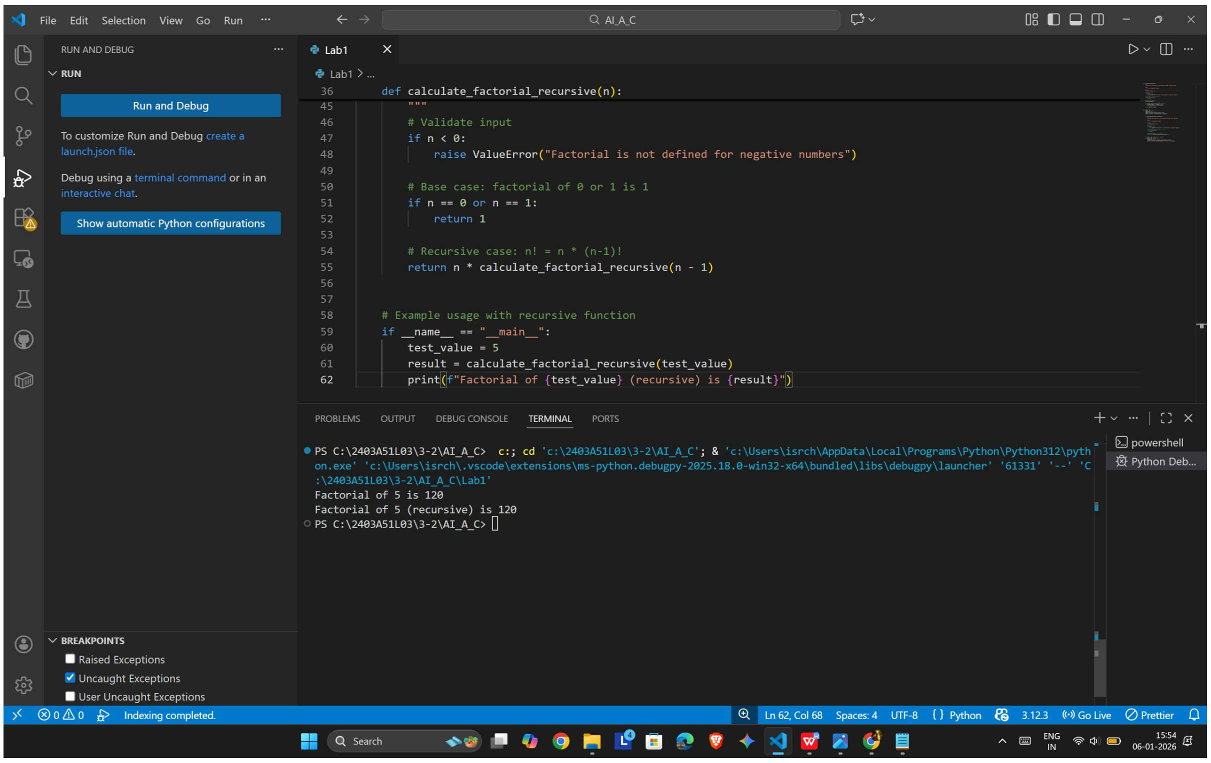
```python
def calculate_factorial(n):

    # Initialize result to 1
    factorial_result = 1

    # Multiply result by each number from 2 to n
    for current_number in range(2, n + 1):
        factorial_result *= current_number

    return factorial_result


# Example usage
if __name__ == "__main__":
    test_value = 5
    result = calculate_factorial(test_value)
    print(f"Factorial of {test_value} is {result}")
```

Write a Python function to calculate factorial using recursion with proper base condition and comments.



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


def calculate_factorial_recursive(n):
    """
    Calculate the factorial of a non-negative integer using recursion.

    Args:
        n: A non-negative integer

    Returns:
        The factorial of n
    """
    # Validate input
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")

    # Base case: factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1

    # Recursive case: n! = n * (n-1)!
    return n * calculate_factorial_recursive(n - 1)


# Example usage with recursive function
if __name__ == "__main__":
    test_value = 5
    result = calculate_factorial_recursive(test_value)
    print(f"Factorial of {test_value} (recursive) is {result}")
```

# OUTPUT:



```python
def calculate_factorial_recursive(n):
    """
    # Validate input
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")

    # Base case: factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1

    # Recursive case: n! = n * (n-1)!
    return n * calculate_factorial_recursive(n - 1)


# Example usage with recursive function
if __name__ == "__main__":
    test_value = 5
    result = calculate_factorial_recursive(test_value)
    print(f"Factorial of {test_value} (recursive) is {result}")
```

```
PS C:\2403A51L03\3-2\AI_A_C>  c:; cd 'c:\2403A51L03\3-2\AI_A_C'; & 'c:\Users\isrch\AppData\Local\Programs\Python\Python312\pyth
on.exe' 'c:\Users\isrch\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61331' '--' 'C
:\2403A51L03\3-2\AI_A_C\Lab1'
Factorial of 5 is 120
Factorial of 5 (recursive) is 120
PS C:\2403A51L03\3-2\AI_A_C>
```

Explaination

- Iterative Approach
- Starts with a result value of 1.
- Repeats multiplication from 1 up to the given number using a loop.
- Stores the intermediate result in the same variable.
- Executes sequentially without extra memory overhead.

- Recursive Approach
- Breaks the problem into smaller subproblems.
- Each function call multiplies the current number by the factorial of the previous number.
- Stops when it reaches the base case (0 or 1).
- Uses the call stack to remember previous function calls.