

# ASSIGNMENT-4

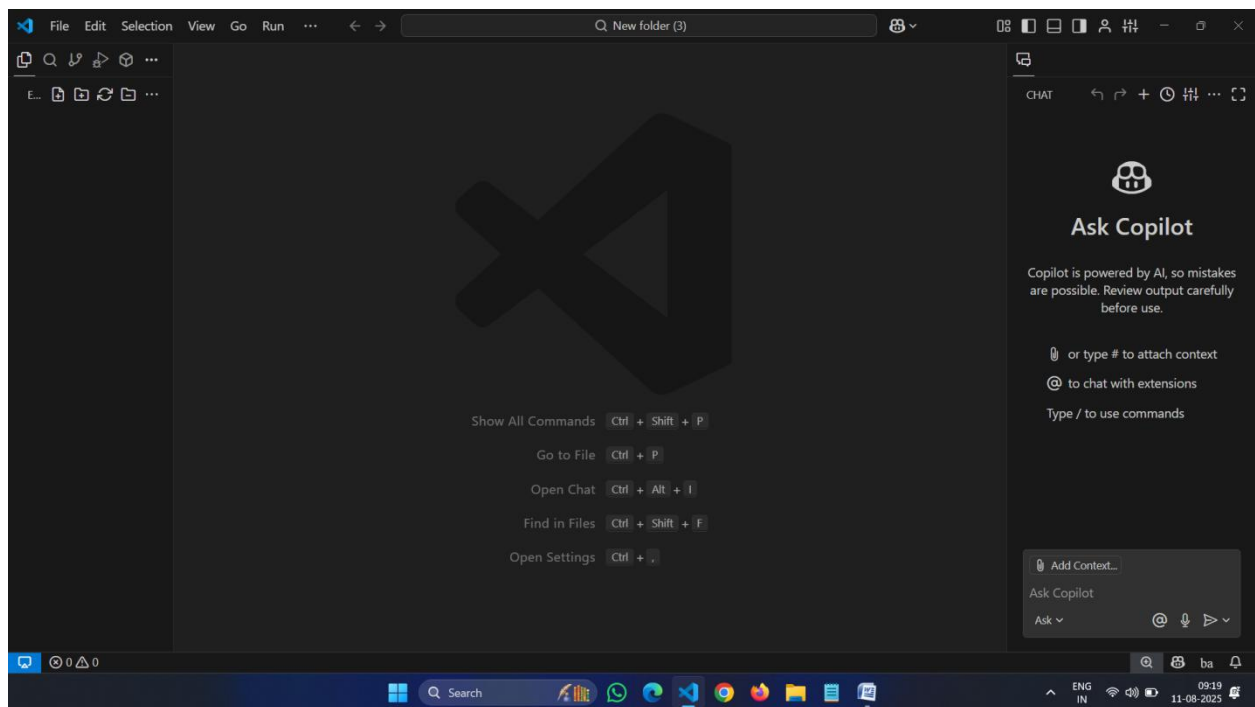
**NAME : R.Shivani**

**ROLL NUMBER : 2403A52411**

**BATCH:15**

## Task 0 :

- Install and configure of GitHub Copilot Take Screenshot of each one.
- Expected output :

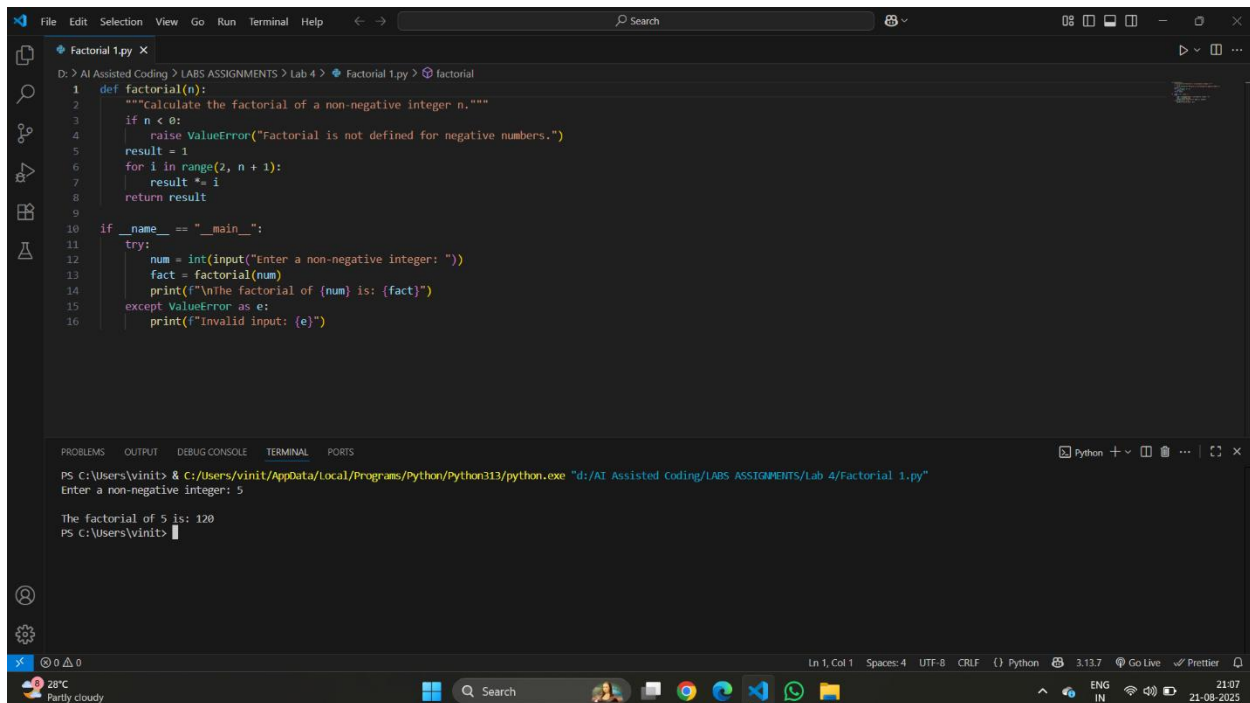


## Task 1 : Factorial without function

- Description

Use GitHub copilot to generate a python program that calculates the factorial of number without defining any function using loops directly in the main code.

- Expected output :



```
1 def factorial(n):
2     """Calculate the factorial of a non-negative integer n."""
3     if n < 0:
4         raise ValueError("Factorial is not defined for negative numbers.")
5     result = 1
6     for i in range(2, n + 1):
7         result *= i
8     return result
9
10 if __name__ == "__main__":
11     try:
12         num = int(input("Enter a non-negative integer: "))
13         fact = factorial(num)
14         print(f"\nThe factorial of {num} is: {fact}")
15     except ValueError as e:
16         print(f"Invalid input: {e}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\vinit> & C:\Users\vinit\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Assisted Coding/LABS ASSIGNMENTS/Lab 4/Factorial 1.py"

Enter a non-negative integer: 5

The factorial of 5 is: 120

PS C:\Users\vinit>

## Task 2 :Improving Efficiency

- Description
- Examine the Copilot generated code .

The code is improved by removing the function definition and calculating the factorial directly in the main block using a loop. This reduces function call overhead and simplifies the logic. The while loop starts from 2 and multiplies up to the input number, which is efficient for factorial calculation. Error handling for invalid input is also included.

The image shows a Windows 11 desktop environment. The primary application is Visual Studio Code (VS Code), which is open to a Python file named 'factorial.py'. The code defines a recursive function 'factorial(n)' that calculates the factorial of a non-negative integer 'n'. It includes error handling for negative numbers using 'raise ValueError' and a while loop for efficiency. The main block of the script prompts the user to enter a non-negative integer and prints the result.

Below the editor, the 'TERMINAL' panel is active, showing the command prompt output. The user has run the script, entered '5', and the output is 'The factorial of 5 is: 120'.

The taskbar at the bottom of the screen displays various icons, including the Start button, search bar, and several pinned applications like Chrome, VS Code, and WhatsApp. The system tray on the right shows the date and time as 21:08 on 21-08-2025.

```

1 def factorial(n):
2     """Calculate the factorial of a non-negative integer n."""
3     if n < 0:
4         raise ValueError("Factorial is not defined for negative numbers.")
5     result = 1
6     # Using a while loop for slightly better efficiency (less overhead than for loop in some cases)
7     i = 2
8     while i <= n:
9         result *= i
10        i += 1
11    return result
12
13 if __name__ == "__main__":
14     try:
15         num = int(input("Enter a non-negative integer: "))
16         if num < 0:
17             print("Factorial is not defined for negative numbers.")
18         else:
19             fact = factorial(num)
20             print(f"\nthe factorial of {num} is: {fact}")
21     except ValueError as e:
22         print(f"Invalid input: {e}")
  
```

```

PS C:\Users\vinit> & C:\Users\vinit\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Assisted Coding/LABS ASSIGNMENTS/Lab 4/Factorial 2.py"
Enter a non-negative integer: 5

The factorial of 5 is: 120
PS C:\Users\vinit>
  
```

### Task 3 :Factorial with functions

- Use GitHub copilot to generate a python program that calculates the factorial of number Using user defining any function
- Expected output :
- Defined a function factorial(n) to calculate the factorial of a non-negative integer.
- Checked if the input n is negative; if so, raised a ValueError.
- Initialized result to 1.
- Used a for loop to multiply result by each integer from 2 to n.
- Returned the computed factorial value.

- In the main block, prompted the user to enter a non-negative integer.
- Called the factorial function with the user input and printed the result.
- Handled invalid input using a try-except block.

```

1 def factorial(n):
2     """Calculate the factorial of a non-negative integer n."""
3     if n < 0:
4         raise ValueError("Factorial is not defined for negative numbers.")
5     result = 1
6     for i in range(2, n + 1):
7         result *= i
8     return result
9
10 if __name__ == "__main__":
11     try:
12         num = int(input("Enter a non-negative integer: "))
13         print(f"The factorial of {num} is: {factorial(num)}")
14     except ValueError as e:
15         print(f"Invalid input: {e}")
  
```

PS C:\Users\vinit> & C:\Users\vinit\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Assisted coding/LABS ASSIGNMENTS/Lab 4/Factorial 3.py"  
 Enter a non-negative integer: 5  
 The factorial of 5 is: 120  
 PS C:\Users\vinit>

## Task 4 : Comparative Analysis – With vs Without Functions

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



- **With Function:**

The logic is encapsulated in a user-defined function ([factorial\(n\)](#)), which takes an argument and returns the factorial. The main code calls this function.

- **Without Function:**

The logic is written directly in the main code block. The loop and calculation are performed inline, without any encapsulation.

## 2. Reusability

- **With Function:**

Highly reusable. The [factorial](#) function can be called multiple times with different arguments, in other parts of the program or from other modules.

- **Without Function:**

Not reusable. The code can only be executed as written, and cannot be called with different inputs without rewriting or copying the logic.

## 3. Execution

- **With Function:**

Slight overhead due to function calls, but allows for cleaner code, easier testing, and better error handling.

- **Without Function:**

Executes sequentially in the main block. Slightly faster for very simple scripts due to no function call overhead, but less organized and harder to maintain for larger programs.

## Summary:

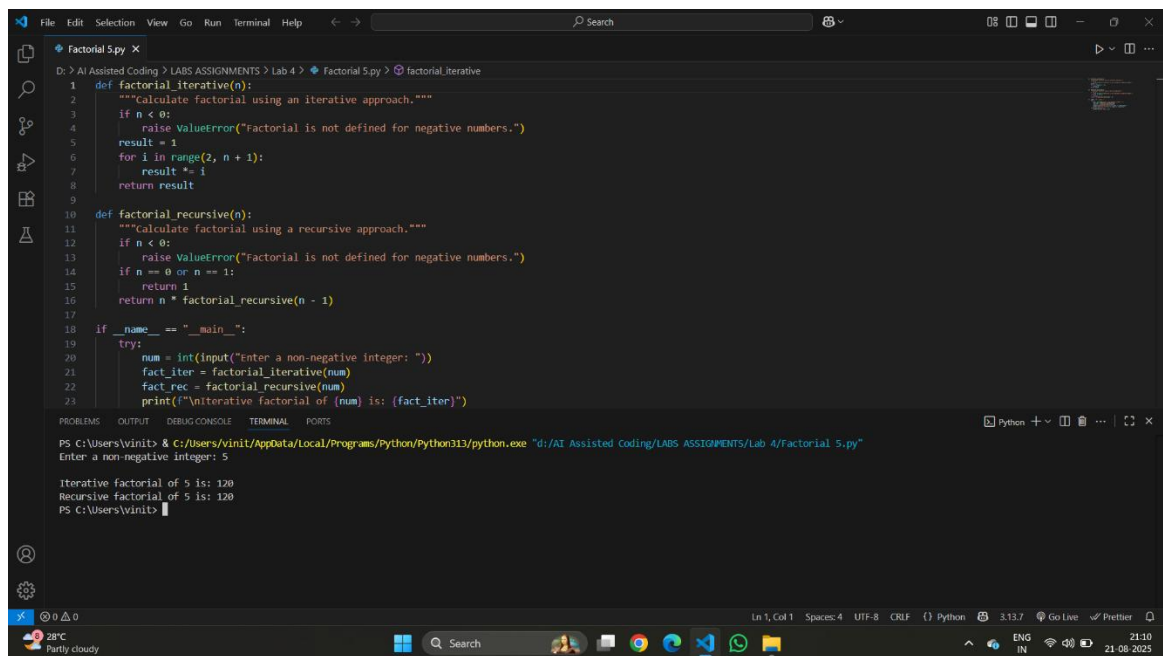
Using functions improves code organization, reusability, and maintainability, while direct logic in the main block is simpler but less flexible and harder to reuse.

## Task 5 :Iterative vs Recursive Factorial

- Description :

Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function

- Expected Output :



```
1 def factorial_iterative(n):
2     """Calculate factorial using an iterative approach."""
3     if n < 0:
4         raise ValueError("Factorial is not defined for negative numbers.")
5     result = 1
6     for i in range(2, n + 1):
7         result *= i
8     return result
9
10 def factorial_recursive(n):
11     """Calculate factorial using a recursive approach."""
12     if n < 0:
13         raise ValueError("Factorial is not defined for negative numbers.")
14     if n == 0 or n == 1:
15         return 1
16     return n * factorial_recursive(n - 1)
17
18 if __name__ == "__main__":
19     try:
20         num = int(input("Enter a non-negative integer: "))
21         fact_iter = factorial_iterative(num)
22         fact_rec = factorial_recursive(num)
23         print(f"Iterative factorial of {num} is: {fact_iter}")
24
25         print(f"Recursive factorial of {num} is: {fact_rec}")
26     except ValueError as e:
27         print(e)
```

PS C:\Users\vinit> & C:\Users\vinit\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Assisted Coding/LABS ASSIGNMENTS/Lab 4/Factorial 5.py"

Enter a non-negative integer: 5

Iterative factorial of 5 is: 120

Recursive factorial of 5 is: 120

PS C:\Users\vinit>