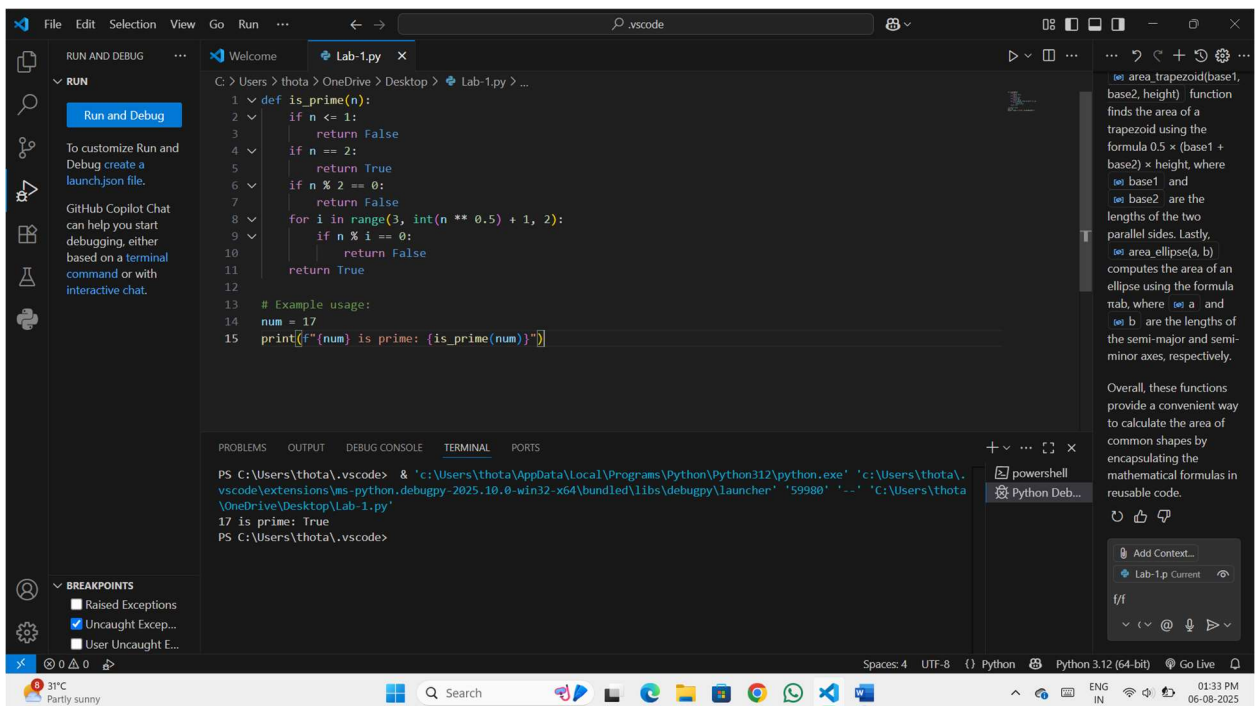


## ★ AI ASSISTED CODING

✓ **Task Description 2:** Use a Copilot to generate a `is_prime()` python functions:

❖ **Expected output:** Functions to check primality with correct Logic.

❖ **Prompt2:** write a python code to check whether number is prime are not.



The screenshot shows the Visual Studio Code editor with a Python file named `Lab-1.py`. The code defines a function `is_prime(n)` that checks if a number is prime. It includes comments and an example usage. The terminal shows the command to run the script, and the output confirms that 17 is a prime number.

```
1 def is_prime(n):
2     if n <= 1:
3         return False
4     if n == 2:
5         return True
6     if n % 2 == 0:
7         return False
8     for i in range(3, int(n ** 0.5) + 1, 2):
9         if n % i == 0:
10            return False
11    return True
12
13 # Example usage:
14 num = 17
15 print(f"{num} is prime: {is_prime(num)}")
```

Terminal Output:

```
PS C:\Users\thota\.vscode> & 'c:\Users\thota\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\thota\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '59980' '--' 'C:\Users\thota\OneDrive\Desktop\Lab-1.py'
17 is prime: True
PS C:\Users\thota\.vscode>
```

❖ **Observation:** A prime number is a number greater than 1 that has no positive divisors other than 1 and itself.

- **Functionality:** Returns True for prime numbers and False otherwise.
- **Suitable for checking primality of large numbers due to reduced iterations.**

- **Task Description 3:** write a comment like # functions to Reverse a String and use copilot to generate the functions:
- ❖ **Expected output:** Auto-completed reverse function.
- ✓ **Prompt3:** write a python code to Reverse a string.

The screenshot shows the Visual Studio Code editor with a Python file named `task-2.py`. The code defines a function `display(s)` that reverses a string `s` using slicing `s[::-1]` and prints both the original and reversed strings. Examples are provided for "hello", "Python123", and "racecar". The terminal output shows the execution results: Original: hello, Reversed: olleh, Original: Python123, Reversed: 321nohtyP, Original: racecar, Reversed: racecar. A Copilot chat window on the right shows a prompt to write a function to reverse a string, and the response provides the function definition and examples. The status bar at the bottom indicates the file is Python 3.12 (64-bit).

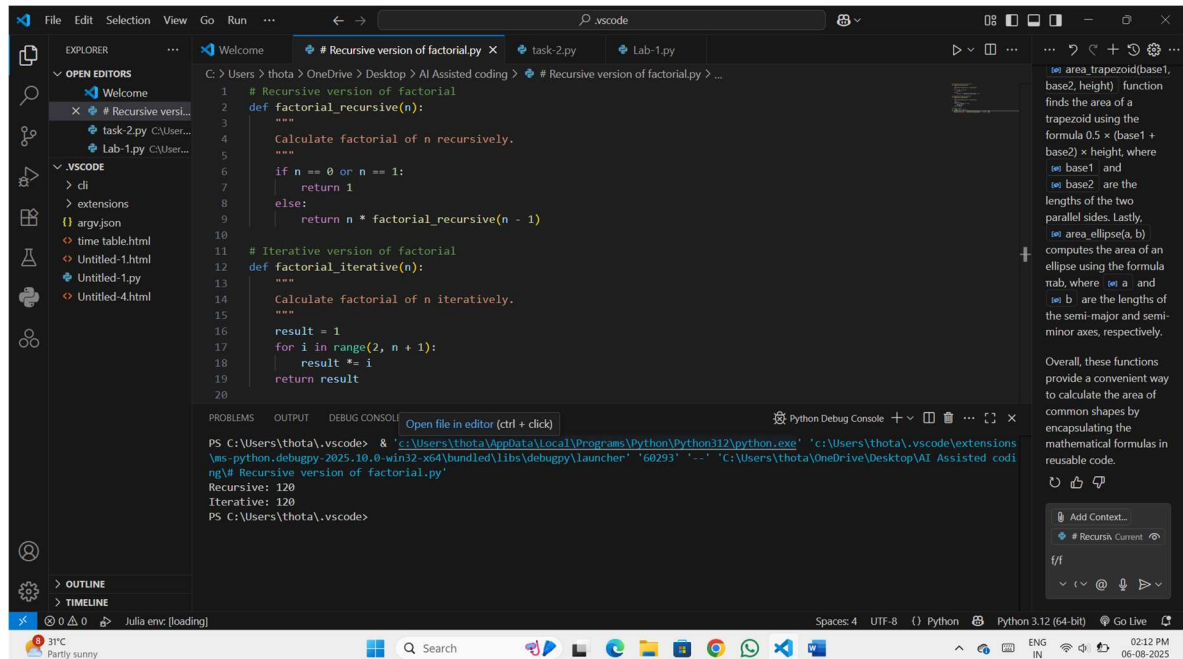
## Observation:

- **Logic:** Uses Python slicing `[::-1]` to reverse the string efficiently.
- **Simplicity:** One-liner function; concise and readable.
- **Functionality:** Works for letters, numbers, symbols, and even empty strings.

- **Task Description 4:** Generate both Recursive and Iterative version of a factorial using comments :

❖ **Expected output:** Two Working Factorial of Implementation.

✓ **Prompt4:** write a python code for both recursive and Iterative factorial.



```
1 # Recursive version of factorial
2 def factorial_recursive(n):
3     """
4     Calculate factorial of n recursively.
5     """
6     if n == 0 or n == 1:
7         return 1
8     else:
9         return n * factorial_recursive(n - 1)
10
11 # Iterative version of factorial
12 def factorial_iterative(n):
13     """
14     Calculate factorial of n iteratively.
15     """
16     result = 1
17     for i in range(2, n + 1):
18         result *= i
19     return result
20
```

PROBLEMS OUTPUT DEBUG CONSOLE Open file in editor (ctrl + click)

```
PS C:\Users\thota\.vscode> & "c:\Users\thota\AppData\Local\Programs\Python\Python121\python.exe" "c:\Users\thota\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher" "60293" "--" "C:\Users\thota\OneDrive\Desktop\AI Assisted coding\# Recursive version of factorial.py"
Recursive: 120
Iterative: 120
PS C:\Users\thota\.vscode>
```

## Observation:

### ➤ Recursive Version:

- Elegant and mirrors the mathematical definition.
- May cause stack overflow for large  $n$  due to deep recursion.
- Time complexity:  $O(n)$ ; Space complexity:  $O(n)$  (due to call stack).

### ➤ Version: Iterative

- More memory-efficient and avoids recursion limits.
- Preferred for large values of  $n$ .
- Time complexity:  $O(n)$ ; Space complexity:  $O(1)$ .

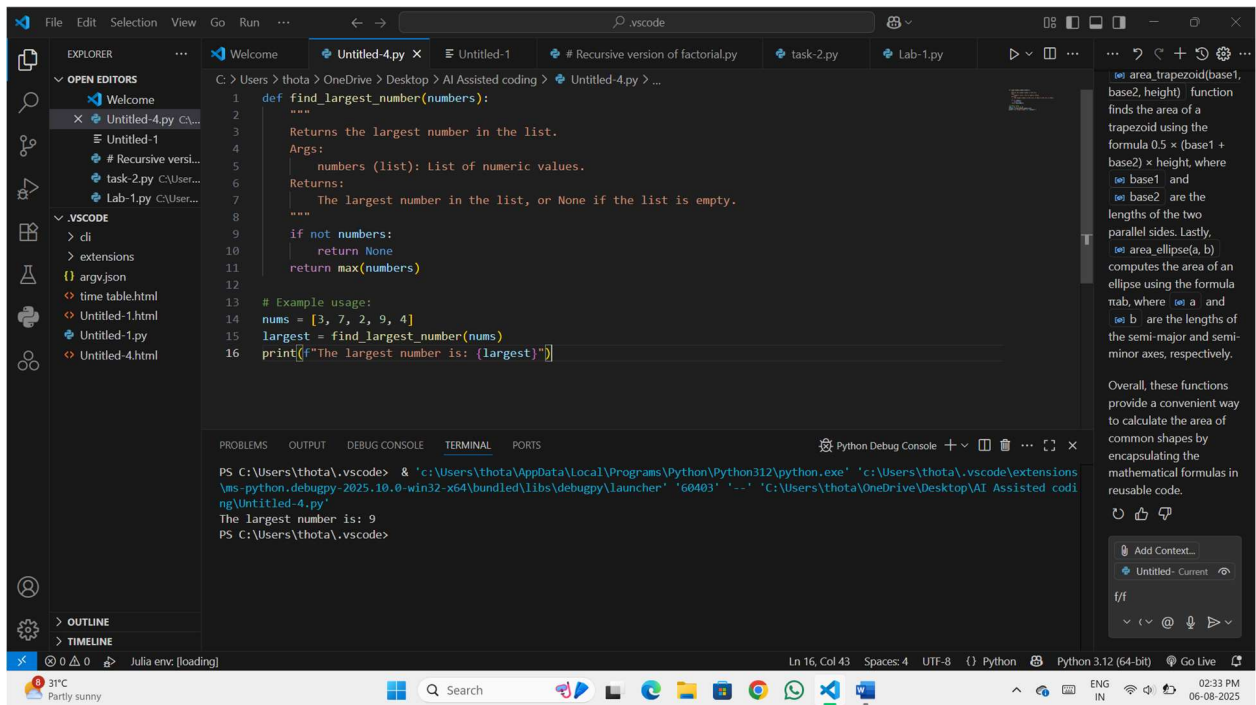
➤ **Both Implementations:**

- Correctly handle base cases ( $0! = 1$ ,  $1! = 1$ ).
- Produce identical results for valid non-negative integers.

- ✓ **Task Description 5:** Use Copilot to find the largest Number in a List and Access code quality and efficiency:

**Expected output:** a valid function with your review  
**Prompt5:** write a python code to find largest number. Prompt: Generate a largest number

Prompt: Generate the python code of largest number



The screenshot shows the Visual Studio Code editor with a Python file named 'Untitled-4.py'. The code defines a function 'find\_largest\_number' that takes a list of numbers and returns the largest one. It includes a docstring, a return statement for an empty list, and a return statement for a non-empty list. The code also includes a test case with a list of numbers [3, 7, 2, 9, 4] and prints the result. The terminal output shows the command to run the code and the output 'The largest number is: 9'.

```
def find_largest_number(numbers):  
    """  
    Returns the largest number in the list.  
    Args:  
        numbers (list): List of numeric values.  
    Returns:  
        The largest number in the list, or None if the list is empty.  
    """  
    if not numbers:  
        return None  
    return max(numbers)  
  
# Example usage:  
nums = [3, 7, 2, 9, 4]  
largest = find_largest_number(nums)  
print(f"The largest number is: {largest}")
```

Terminal Output:

```
PS C:\Users\thota\.vscode> &'c:\Users\thota\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\thota\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '60403' '-.' 'C:\Users\thota\OneDrive\Desktop\AI Assisted coding\Untitled-4.py'  
The largest number is: 9  
PS C:\Users\thota\.vscode>
```

**Observation:**

- **Correctness:** Accurately finds the largest number by comparing each element.
- **Edge Case Handling:** Returns None for an empty list, avoiding errors.

➤ **Efficiency:**

- Time complexity:  $O(n)$  — linear scan through the list.
- Space complexity:  $O(1)$  — uses constant extra space.

➤ **Code Quality:**

- Clear variable naming (largest, numbers).