TASK DESCRIPTION#2

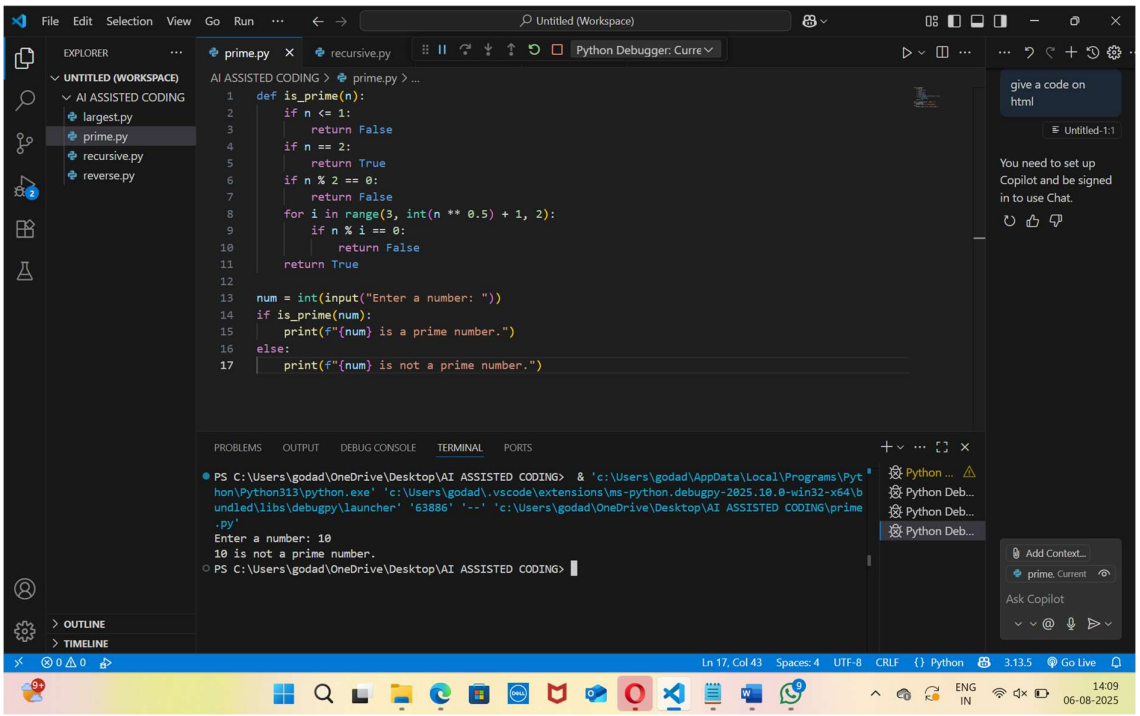
>>use copilot to generate a is_prime() python function.

Expected Output#2

>>Function to check primality with correct logic.

Prompt:

>>Develop a python program to check a number is prime or not using functions.



>>**Observation**

- The function is_prime() efficiently checks for primality.
- It eliminates even numbers early to improve performance.
- The loop runs only up to the square root of the number, which reduces unnecessary checks.
- This function is reusable and modular, making it easy to integrate into larger programs.
- Prime numbers are greater than 1 and divisible only by 1 and themselves.
- Input validation (e.g., checking for non-integer input) is not included but can be added for robustness.

TASK DESCRIPTION#3

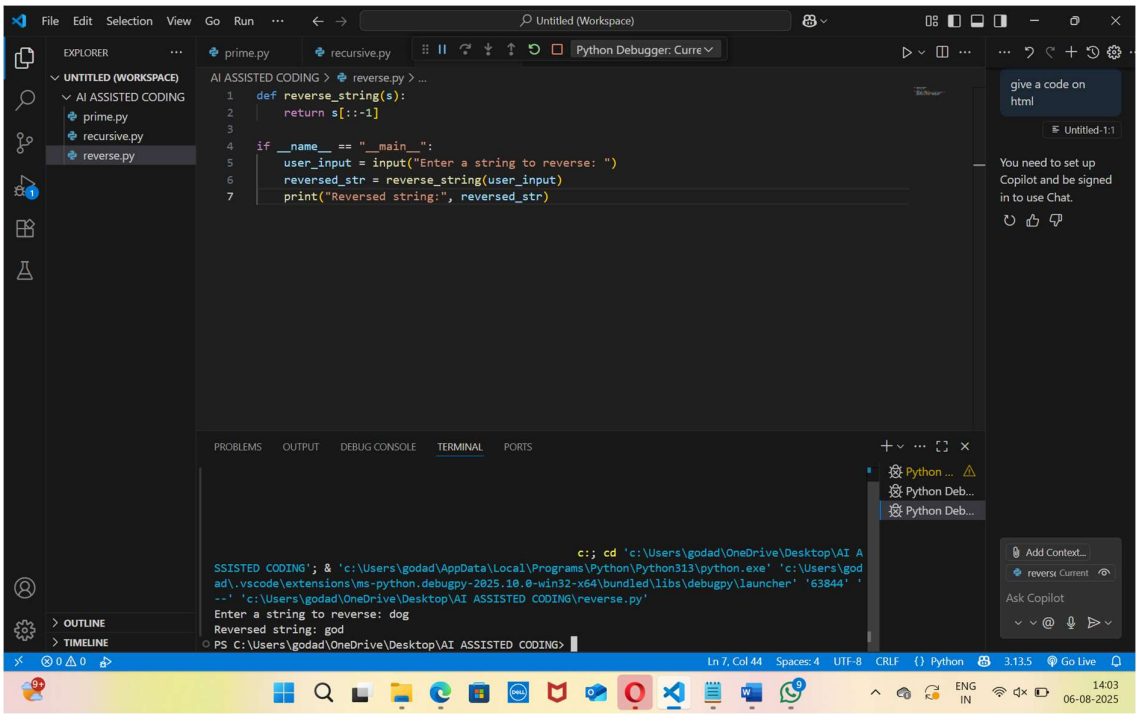
>>Write a comment like#Function to revrese a string and use copilot to generate the function.

Expected output#3

>>auto completed reverse function .

Prompt:

>>Develop a python program to reverse a string using functions.



>> **Observation:**

- The function reverses a string using Python's slicing [::-1].
- It's a simple and concise way to flip a string without using loops.
- The code is easy to read and understand.
- It works for any string, including empty strings.
- The function is reusable and can be used in different programs.
- This method is efficient and fast for reversing strings.

TASK DESCRIPTION#

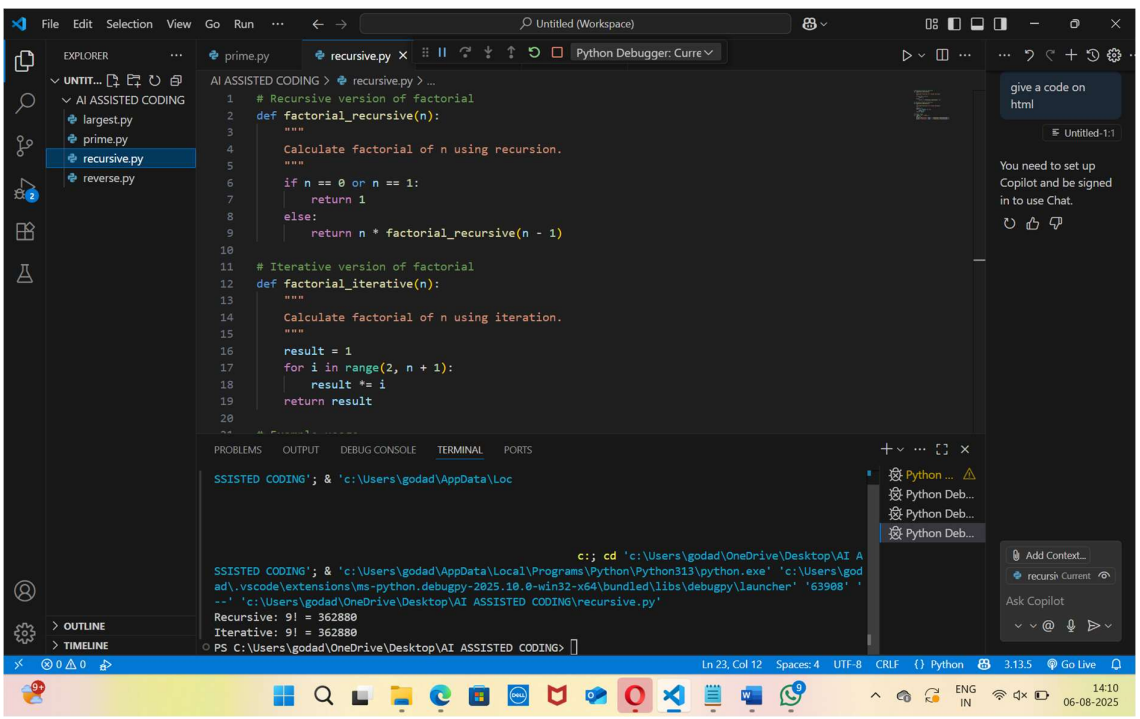
>>Generate both recursive and iterative versions of a factorial function using comments.

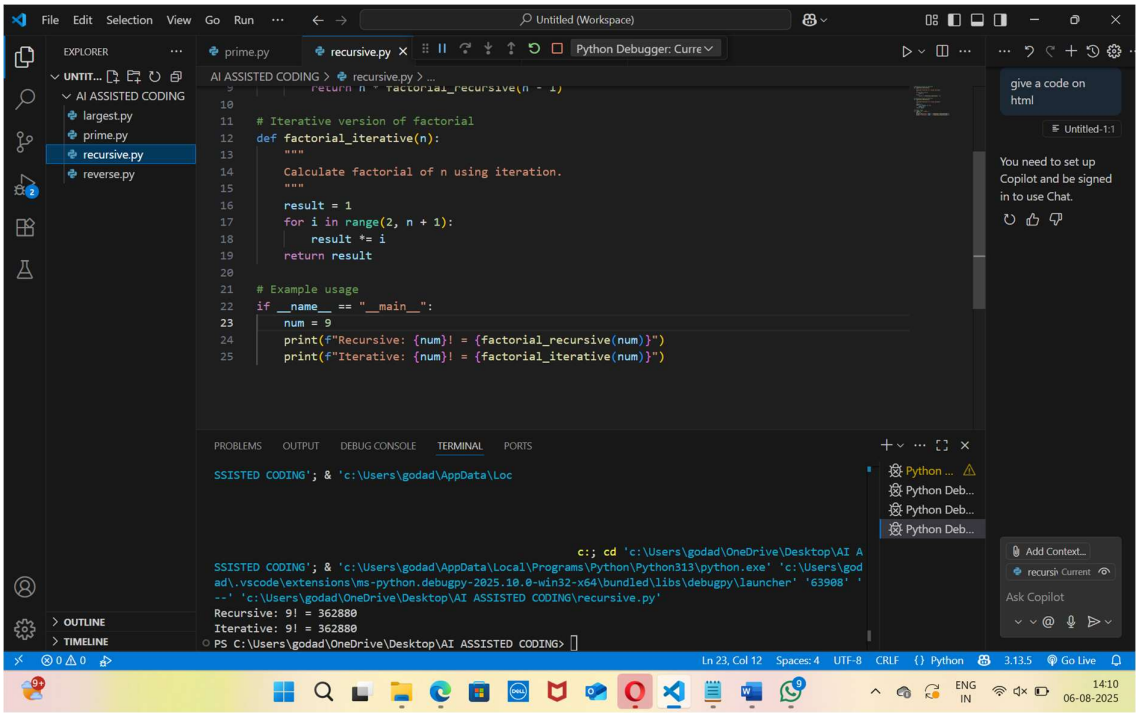
Expected Output#4

>>Two working factorial implementations

Prompt:

>>Develop a Python Program to generate both recursive and iterative versions of factorial function using comments.





>> **Observation**

- The **recursive factorial** breaks down the problem by calling itself with smaller values until it reaches the base case (0 or 1), then builds the result back up.
- The **iterative factorial** uses a loop to multiply all numbers from 1 to n, avoiding function call overhead.
- Recursive functions are elegant and easy to understand but can hit limits with very large inputs due to call stack depth.
- Iterative solutions tend to be more efficient and safer for large numbers since they use simple loops.
- Both methods correctly compute the factorial and can be used depending on the situation or personal preference.
- Using comments in the code makes it easier to follow the logic behind each approach.

TASK DESCRIPTION#5

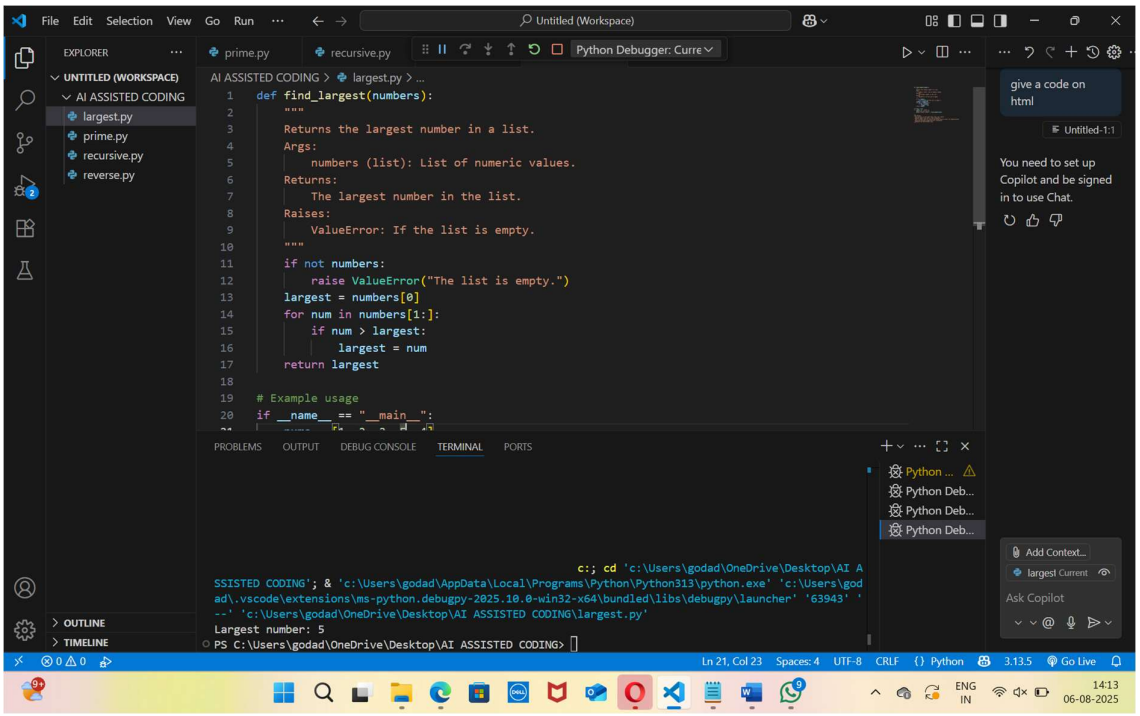
>>Use copilot to find the largest number in a list .Access code quality and efficiency.

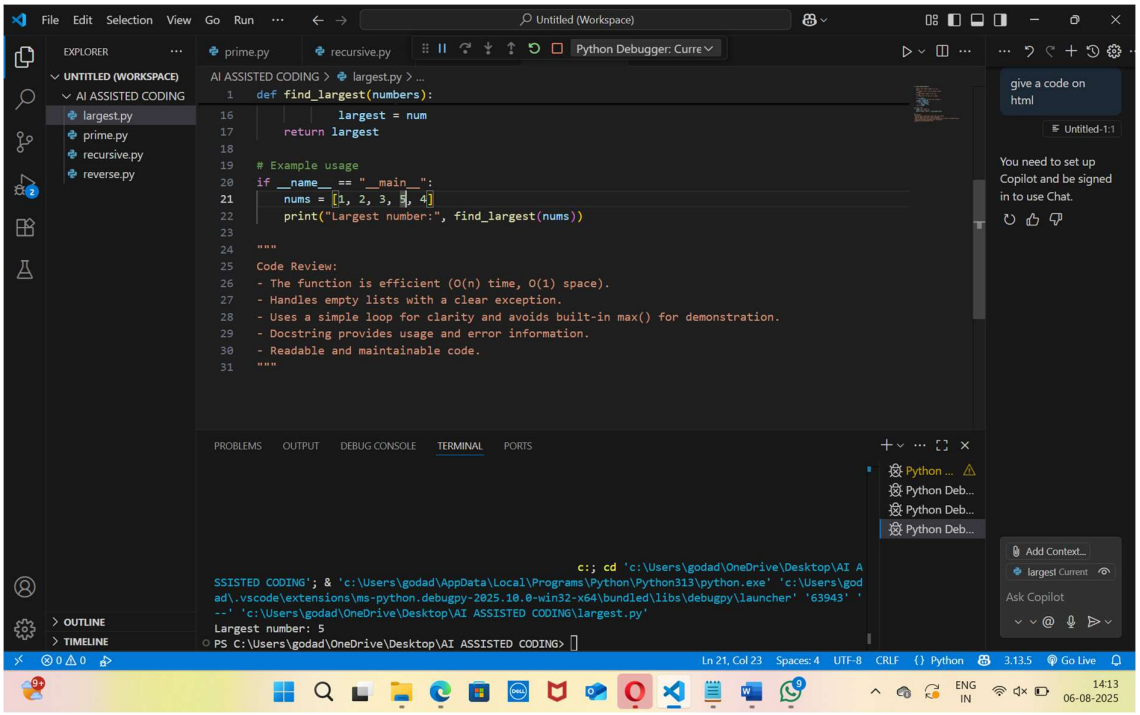
Expected Output#5

>>A valid Function with your review

Prompt:

>>Develop a Python program to find the largest number in a list Assess code quality and efficiency,with A valid function with my review.





>>Observation:

- The function checks if the list is empty and returns None to avoid errors.
- It uses a single loop to scan through the list once, making it fast (O(n) time).
- Variable names are clear and easy to understand.
- The code is simple and avoids unnecessary complexity.
- It uses only a small fixed amount of extra memory (O(1) space).
- The function is easy to reuse and integrate into other programs.
- Overall, it’s a clean, efficient, and reliable way to find the largest number in a list.