

TASK DESCRIPTION#2

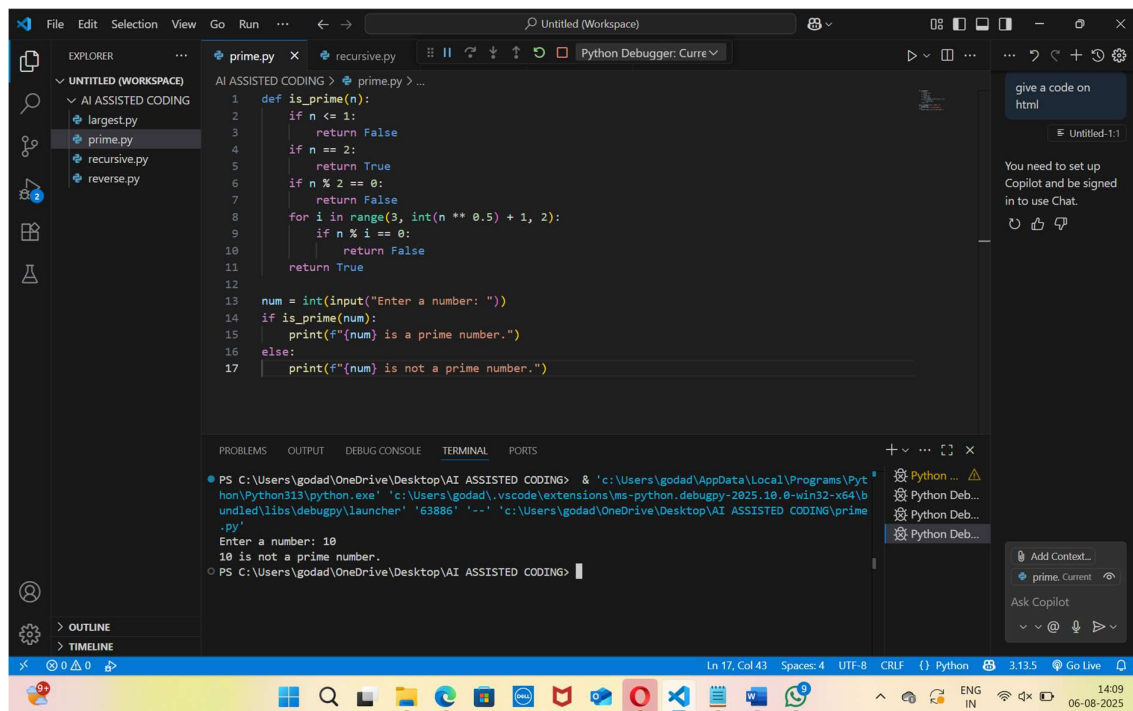
>>use copilot to generate a is_prime() python function.

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.



```
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 num = int(input("Enter a number: "))
14 if is_prime(num):
15     print(f"{num} is a prime number.")
16 else:
17     print(f"{num} is not a prime number.")
```

Enter a number: 10
10 is not a prime number.

>>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

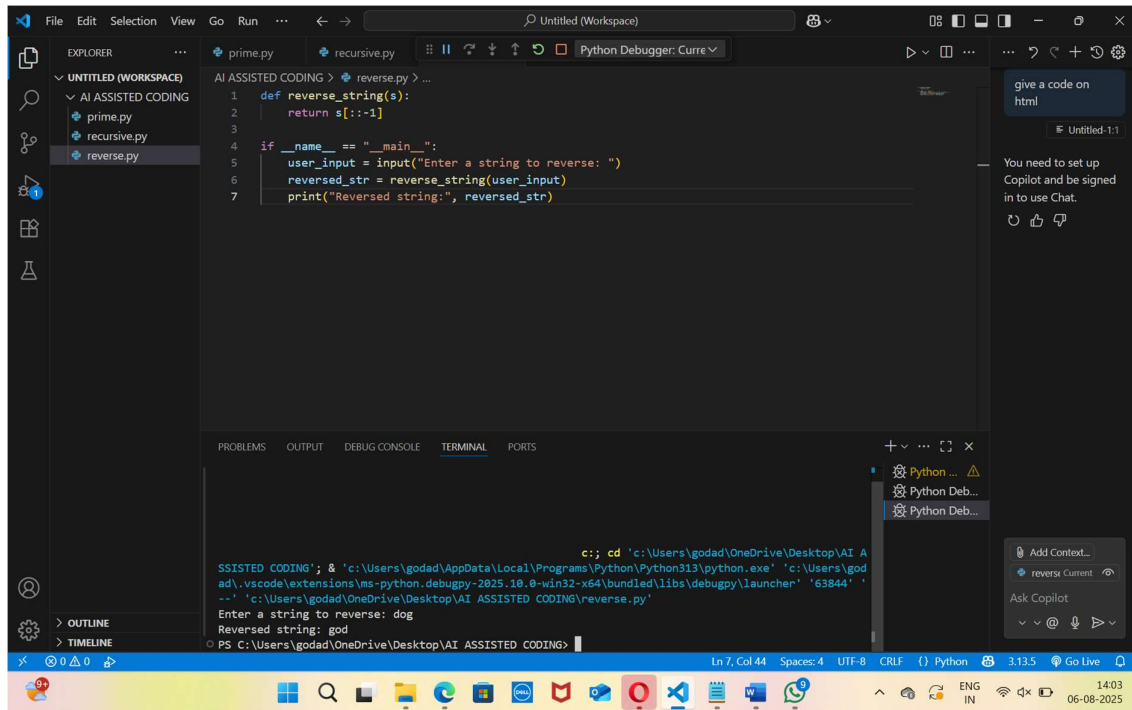
>>Write a comment like#Function to reverse a string and use copilot to generate the function.

Expected output#3

>>auto completed reverse function .

Prompt:

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



The screenshot shows the Visual Studio Code editor interface. The Explorer pane on the left shows a workspace with files 'prime.py', 'recursive.py', and 'reverse.py'. The main editor window displays the code for 'reverse.py':

```
1 def reverse_string(s):
2     return s[::-1]
3
4 if __name__ == "__main__":
5     user_input = input("Enter a string to reverse: ")
6     reversed_str = reverse_string(user_input)
7     print("Reversed string:", reversed_str)
```

The bottom panel shows the TERMINAL with the following output:

```
c::; cd 'c:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING'; & 'c:\Users\godad\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\godad\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '63844' '-.' 'c:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING\reverse.py'
Enter a string to reverse: dog
Reversed string: god
PS C:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING>
```

On the right side, there is a Copilot chat window with the prompt 'give a code on html' and a response indicating that Copilot needs to be set up and signed in to use Chat.

>> **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#

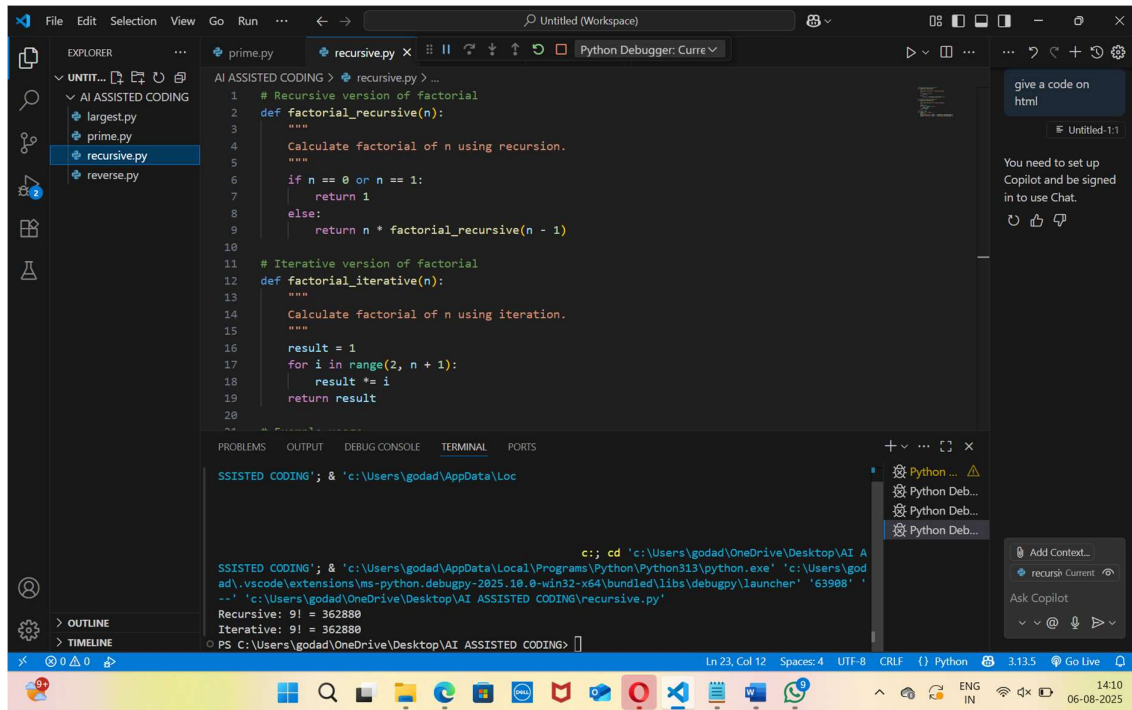
>>Generate both recursive and iterative versions of a factorial function using comments.

Expected Output#4

>>Two working factorial implementations

Prompt:

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



```
1 # Recursive version of factorial
2 def factorial_recursive(n):
3     """
4     Calculate factorial of n using recursion.
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 using iteration.
15     """
16     result = 1
17     for i in range(2, n + 1):
18         result *= i
19     return result
20
21 if __name__ == '__main__':
22     n = 9
23     print("Recursive: 9! = 362880")
24     print("Iterative: 9! = 362880")
```

Terminal Output:

```
PS C:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING> python recursive.py
Recursive: 9! = 362880
Iterative: 9! = 362880
```

```
10
11 # Iterative version of factorial
12 def factorial_iterative(n):
13     """
14     Calculate factorial of n using iteration.
15     """
16     result = 1
17     for i in range(2, n + 1):
18         result *= i
19     return result
20
21 # Example usage
22 if __name__ == "__main__":
23     num = 9
24     print(f"Recursive: {num}! = {factorial_recursive(num)}")
25     print(f"Iterative: {num}! = {factorial_iterative(num)}")
```

```
SSISTED CODING'; & 'c:\Users\godad\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\godad\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '63908' '-.' 'c:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING\recursive.py'
Recursive: 9! = 362880
Iterative: 9! = 362880
PS C:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING>
```

>> 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

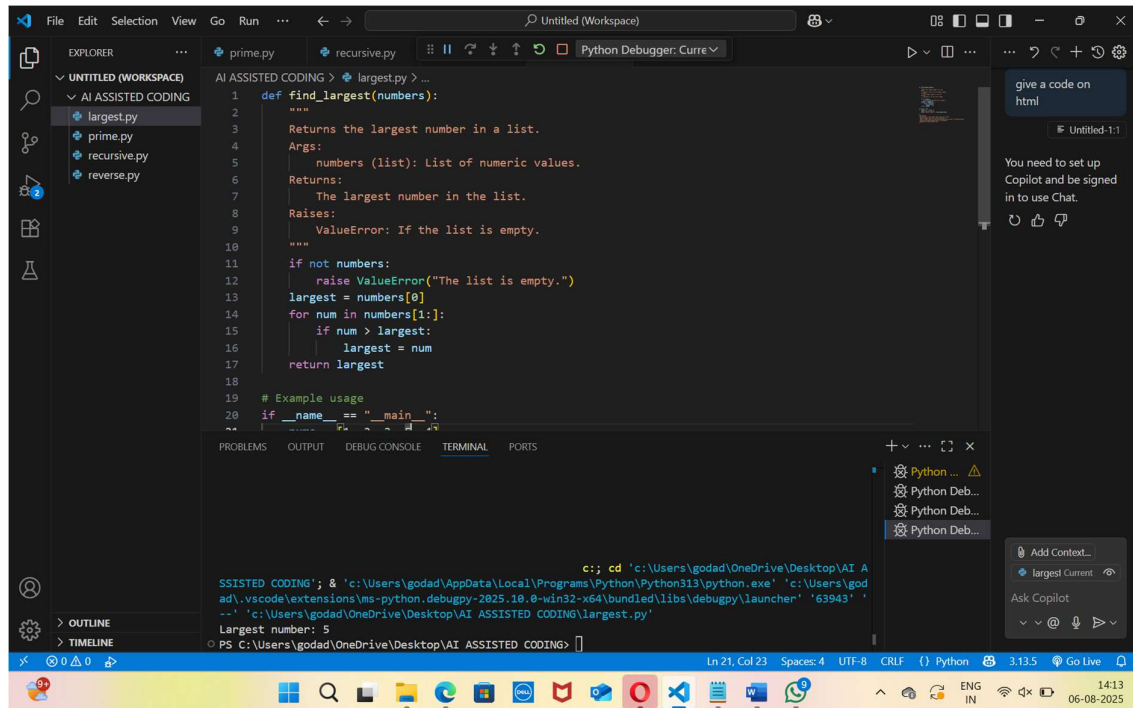
>>Use copilot to find the largest number in a list .Access code quality and efficiency.

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.

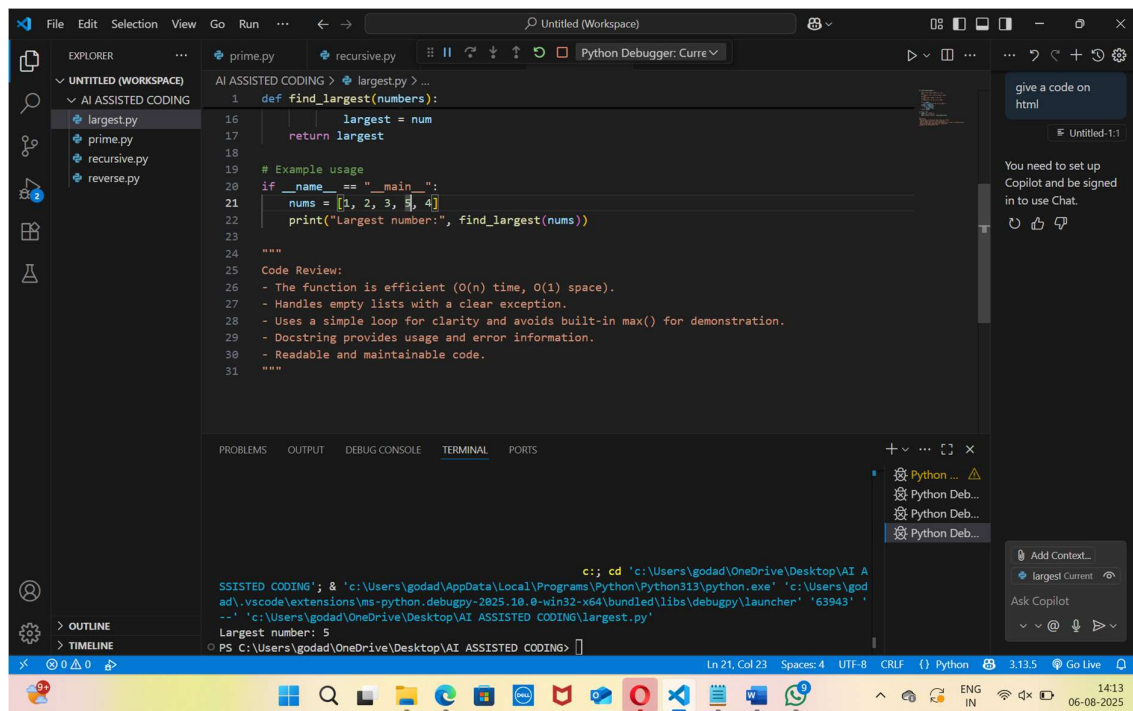


The screenshot shows the Visual Studio Code editor with a workspace named 'Untitled (Workspace)'. The Explorer panel on the left shows a file named 'largest.py' under the 'AI ASSISTED CODING' folder. The main editor window displays the following Python code:

```
1 def find_largest(numbers):
2     """
3     Returns the largest number in a list.
4     Args:
5     numbers (list): List of numeric values.
6     Returns:
7     The largest number in the list.
8     Raises:
9     ValueError: If the list is empty.
10    """
11    if not numbers:
12        raise ValueError("The list is empty.")
13    largest = numbers[0]
14    for num in numbers[1:]:
15        if num > largest:
16            largest = num
17    return largest
18
19 # Example usage
20 if __name__ == "__main__":
21     # ...
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
c:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING> python largest.py
Largest number: 5
```



The screenshot shows the Visual Studio Code editor with the same workspace. The main editor window displays the following Python code:

```
16     largest = num
17     return largest
18
19 # Example usage
20 if __name__ == "__main__":
21     nums = [1, 2, 3, 4]
22     print("Largest number:", find_largest(nums))
23
24 """
25 Code Review:
26 - The function is efficient (O(n) time, O(1) space).
27 - Handles empty lists with a clear exception.
28 - Uses a simple loop for clarity and avoids built-in max() for demonstration.
29 - Docstring provides usage and error information.
30 - Readable and maintainable code.
31 """
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
c:\Users\godad\OneDrive\Desktop\AI ASSISTED CODING> python largest.py
Largest number: 5
```

>>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.