

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M. Tech/MCA		Assignment Type: Lab	AcademicYear:2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		Course Title	AI Assisted Problem Solving Using Python
Year/Sem	I/I	Regulation	R24
Date and Day of Assignment	Week1 - Monday	Time(s)	
Duration	2 Hours	Applicable to Batches	M. Tech/MCA
AssignmentNumber:1.3(Present assignment number)/24(Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	<p>Lab 1: Environment Setup – GitHub Copilot and VS Code Integration</p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> To install and configure GitHub Copilot in Visual Studio Code. To explore AI-assisted code generation using GitHub Copilot. To analyze the accuracy and effectiveness of Copilot's code suggestions. To understand prompt-based programming using comments and code context <p>Lab Outcomes (LOs): After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> Set up GitHub Copilot in VS Code successfully. Use inline comments and context to generate code with Copilot. Evaluate AI-generated code for correctness and readability. Compare code suggestions based on different prompts and programming styles. <p>Task Description#1</p> <ul style="list-style-type: none"> Install and configure GitHub Copilot in VS Code. Take screenshots of each step. <p>Expected Output#1</p> <ul style="list-style-type: none"> Install and configure GitHub Copilot in VS Code. Take screenshots of each step. 		Week1 - Wednesday

Task Description#2

- Use Copilot to generate a `is_prime()` Python function.

Expected Output#2

- Function to check primality with correct logic.

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

Task Description#4

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

Expected Output#4

- Two working factorial implementations

Task Description#5

- Use Copilot to find the largest number in a list. Assess code quality and efficiency.

Expected Output#5

- A valid function with your review

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.

Evaluation Criteria:

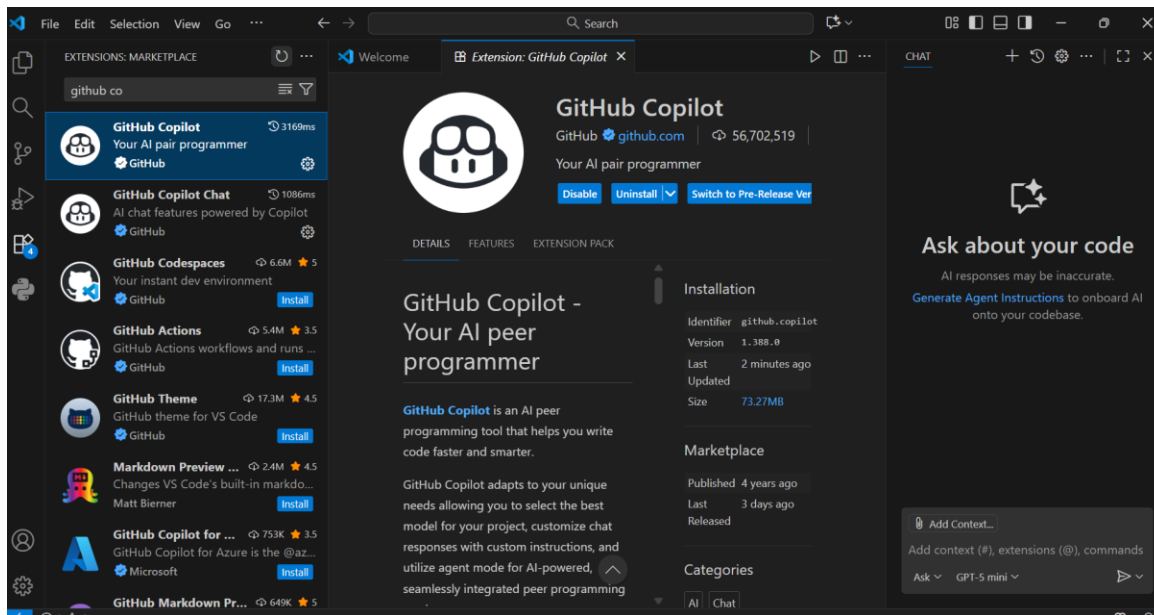
Criteria	Max Marks
Successful Setup of Copilot (Task #1)	2
<code>is_prime()</code> Python function (Task #2)	2
Reverse a string function (Task #3)	2
Factorial Function (Task #4)	2
Find the largest number (Task #5)	2
Total	10 Marks

Task Description#1

Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

Steps:

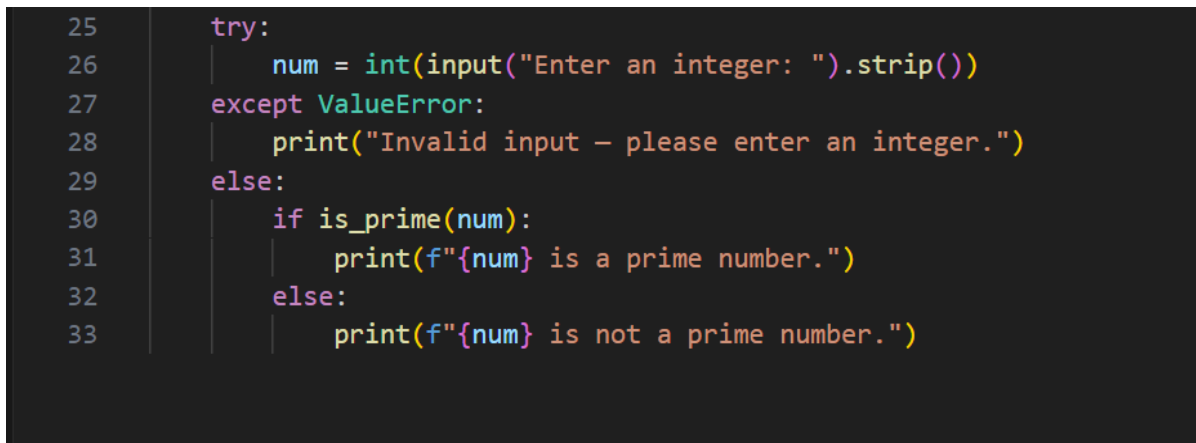
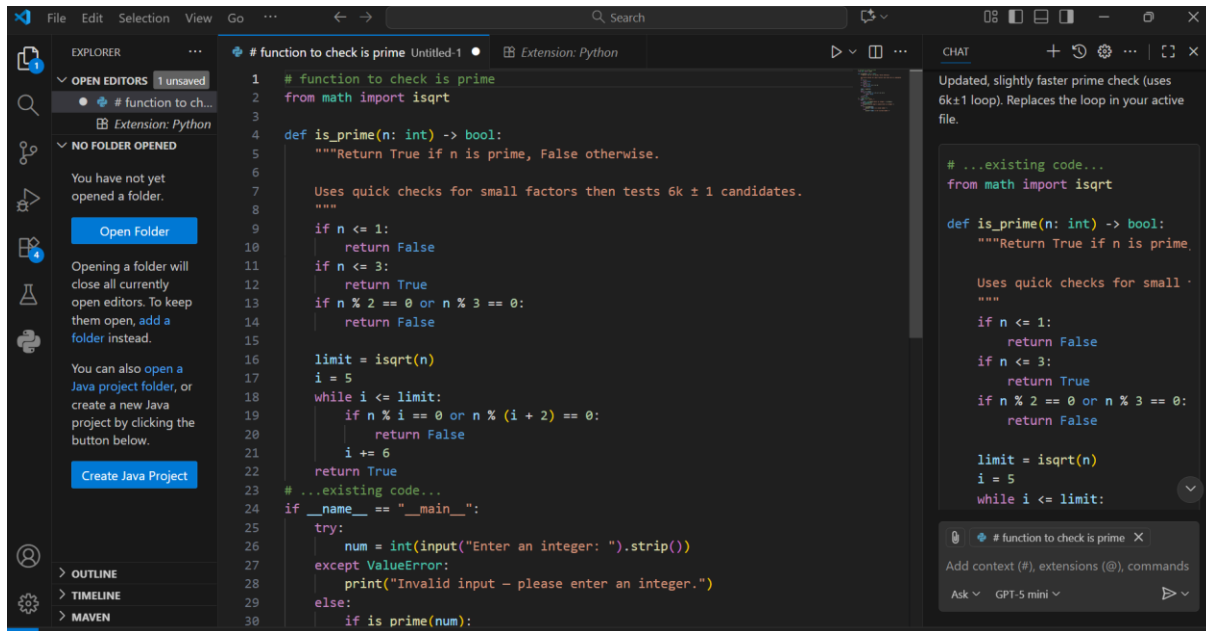
- Open VS Code.
- To open Extensions, press “**Ctrl + Shift + X**”.
- In the search bar, type "**GitHub Copilot**".
- Click **Install** on the official extension by GitHub.
- When prompted, sign in with your GitHub account.



Task Description#2

Use Copilot to generate a `is_prime()` Python function.

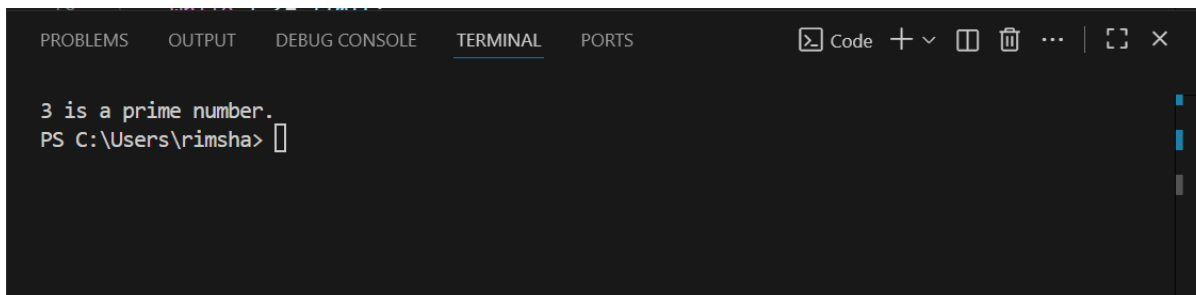
Prompt: Provide me a Function to check if a number is prime



Expected Output#2

Function to check primality with correct logic.

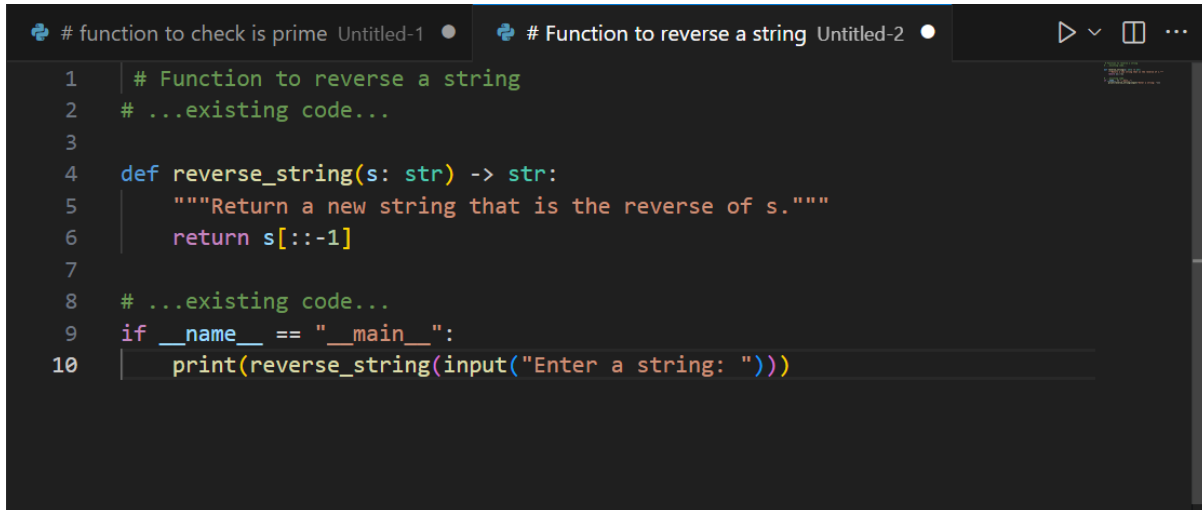
Practical output:



Task Description#3

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

Prompt: Give me a function to reverse a string



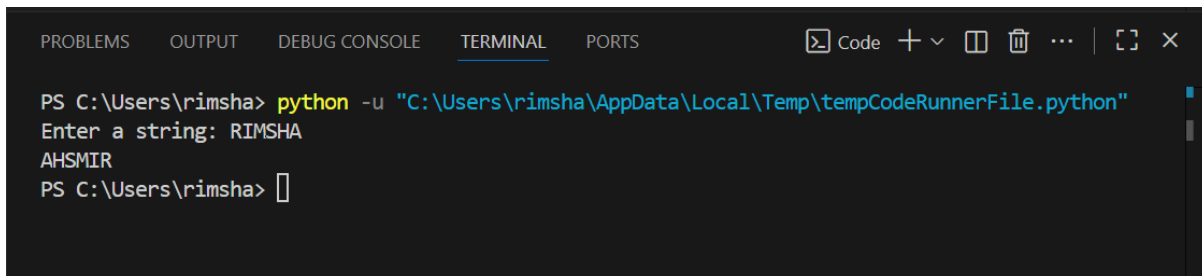
The screenshot shows a code editor with two tabs. The active tab is titled "# Function to reverse a string Untitled-2". It contains the following Python code:

```
1 # Function to reverse a string
2 # ...existing code...
3
4 def reverse_string(s: str) -> str:
5     """Return a new string that is the reverse of s."""
6     return s[::-1]
7
8 # ...existing code...
9 if __name__ == "__main__":
10     print(reverse_string(input("Enter a string: ")))
```

Expected Output#3

Auto-completed reverse function

Practical output:



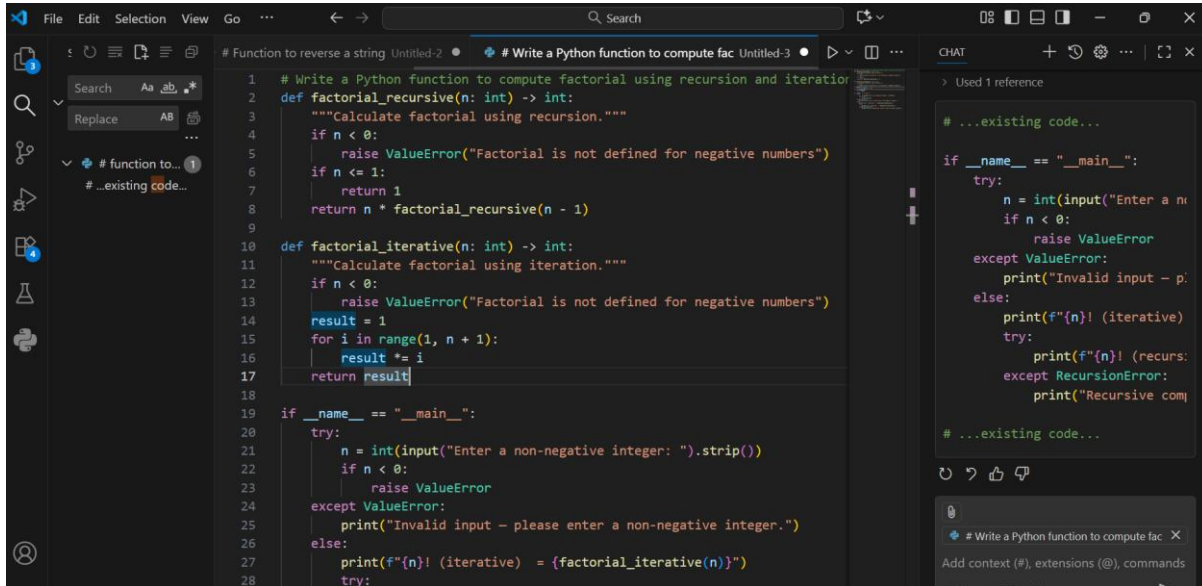
The screenshot shows a terminal window with the following output:

```
PS C:\Users\rimsha> python -u "C:\Users\rimsha\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter a string: RIMSHA
AHSMIR
PS C:\Users\rimsha>
```

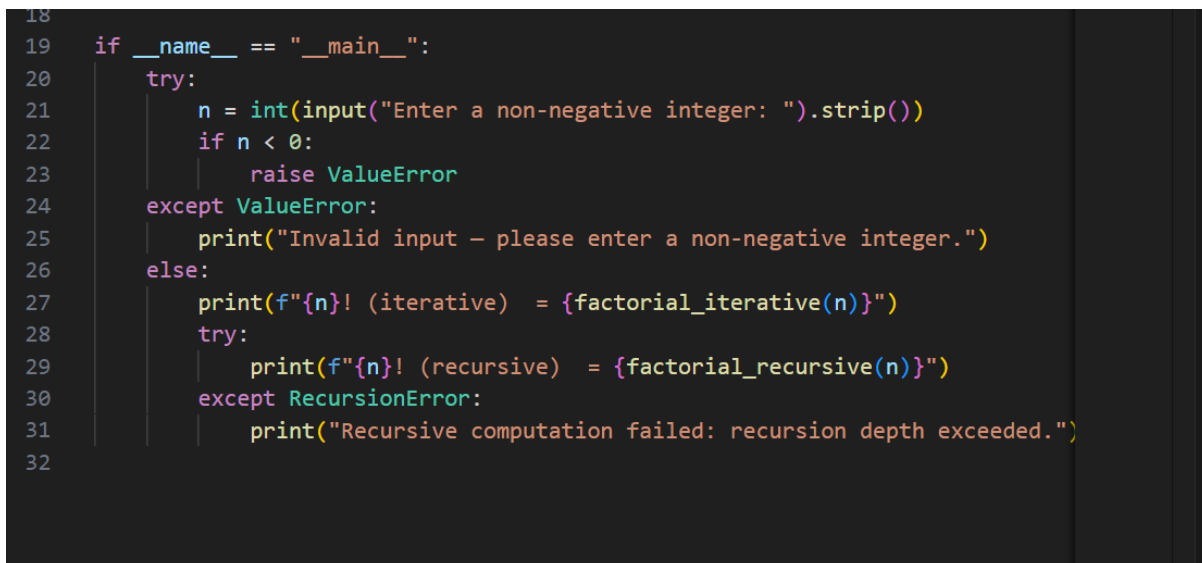
Task Description#4

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

Prompt: Write a Python function to compute factorial using recursion and iteration with user input.



```
1 # Write a Python function to compute factorial using recursion and iteration
2 def factorial_recursive(n: int) -> int:
3     """Calculate factorial using recursion."""
4     if n < 0:
5         raise ValueError("Factorial is not defined for negative numbers")
6     if n <= 1:
7         return 1
8     return n * factorial_recursive(n - 1)
9
10 def factorial_iterative(n: int) -> int:
11     """Calculate factorial using iteration."""
12     if n < 0:
13         raise ValueError("Factorial is not defined for negative numbers")
14     result = 1
15     for i in range(1, n + 1):
16         result *= i
17     return result
18
19 if __name__ == "__main__":
20     try:
21         n = int(input("Enter a non-negative integer: ").strip())
22         if n < 0:
23             raise ValueError
24     except ValueError:
25         print("Invalid input - please enter a non-negative integer.")
26     else:
27         print(f"{n}! (iterative) = {factorial_iterative(n)}")
28         try:
29             print(f"{n}! (recursive) = {factorial_recursive(n)}")
30         except RecursionError:
31             print("Recursive computation failed: recursion depth exceeded.")
```

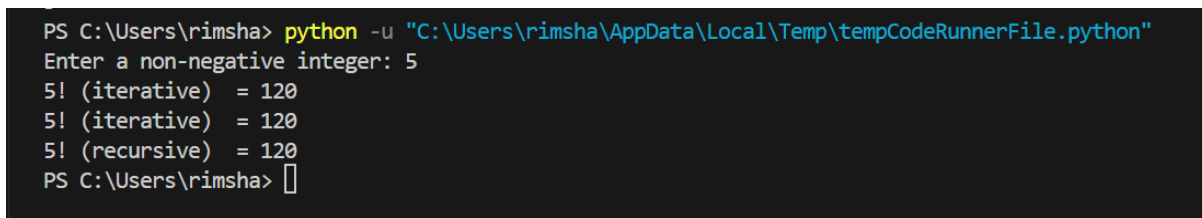


```
18
19 if __name__ == "__main__":
20     try:
21         n = int(input("Enter a non-negative integer: ").strip())
22         if n < 0:
23             raise ValueError
24     except ValueError:
25         print("Invalid input - please enter a non-negative integer.")
26     else:
27         print(f"{n}! (iterative) = {factorial_iterative(n)}")
28         try:
29             print(f"{n}! (recursive) = {factorial_recursive(n)}")
30         except RecursionError:
31             print("Recursive computation failed: recursion depth exceeded.")
32
```

Expected Output#4

Two working factorial implementations

Practical output:

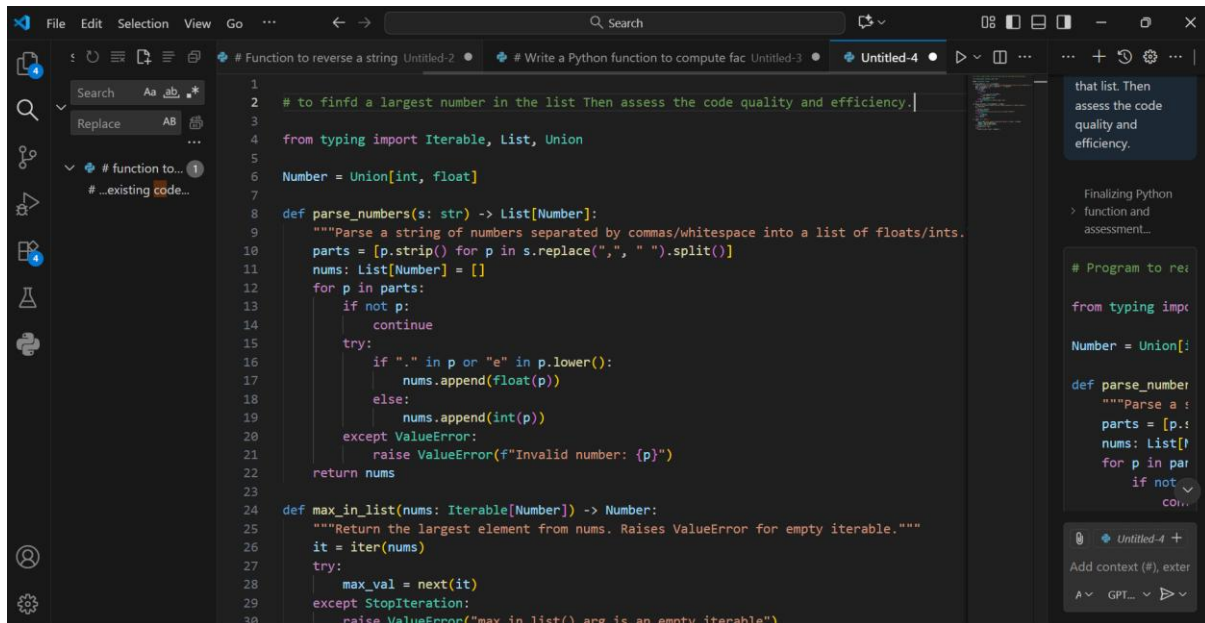


```
PS C:\Users\rimsha> python -u "C:\Users\rimsha\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter a non-negative integer: 5
5! (iterative) = 120
5! (iterative) = 120
5! (recursive) = 120
PS C:\Users\rimsha> █
```

Task Description#5

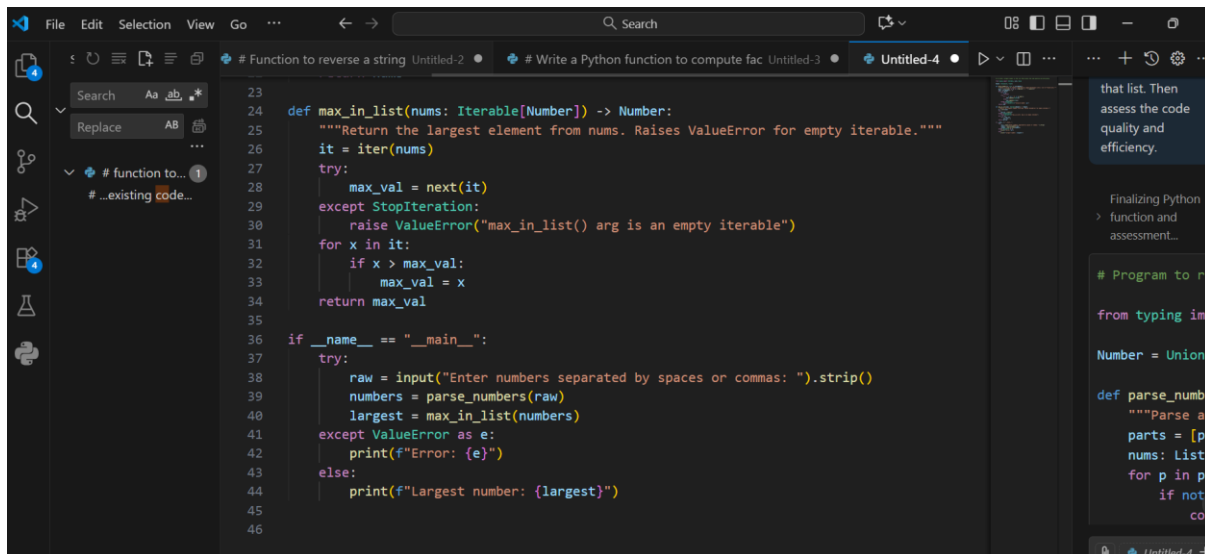
Use Copilot to find the largest number in a list. Assess code quality and efficiency.

Prompt: To find a largest number in the list Then assess the code quality and efficiency.



The screenshot shows a VS Code editor with a Python file named 'Untitled-4'. The code defines a function 'parse_numbers' that takes a string 's' and returns a list of numbers. It then defines a function 'max_in_list' that takes an iterable of numbers and returns the maximum value. The code is commented with a prompt: '# to find a largest number in the list Then assess the code quality and efficiency.' The right sidebar shows a Copilot chat window with the prompt and the code.

```
1 # to find a largest number in the list Then assess the code quality and efficiency.
2
3
4 from typing import Iterable, List, Union
5
6 Number = Union[int, float]
7
8 def parse_numbers(s: str) -> List[Number]:
9     """Parse a string of numbers separated by commas/whitespace into a list of floats/ints.
10     parts = [p.strip() for p in s.replace(",", " ").split()]
11     nums: List[Number] = []
12     for p in parts:
13         if not p:
14             continue
15         try:
16             if "." in p or "e" in p.lower():
17                 nums.append(float(p))
18             else:
19                 nums.append(int(p))
20         except ValueError:
21             raise ValueError(f"Invalid number: {p}")
22     return nums
23
24 def max_in_list(nums: Iterable[Number]) -> Number:
25     """Return the largest element from nums. Raises ValueError for empty iterable."""
26     it = iter(nums)
27     try:
28         max_val = next(it)
29     except StopIteration:
30         raise ValueError("max_in_list() arg is an empty iterable")
```



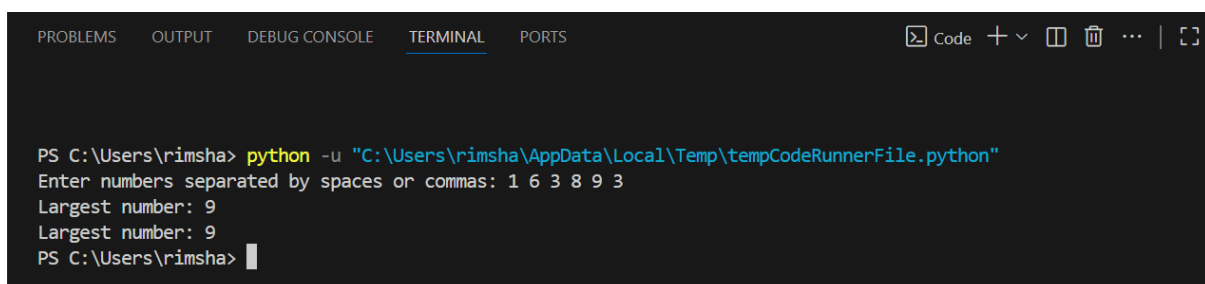
The screenshot shows the same VS Code editor with the Python file 'Untitled-4'. The code now includes a main function that takes user input, parses it, and finds the maximum value. The code is commented with a prompt: '# to find a largest number in the list Then assess the code quality and efficiency.' The right sidebar shows a Copilot chat window with the prompt and the code.

```
23
24 def max_in_list(nums: Iterable[Number]) -> Number:
25     """Return the largest element from nums. Raises ValueError for empty iterable."""
26     it = iter(nums)
27     try:
28         max_val = next(it)
29     except StopIteration:
30         raise ValueError("max_in_list() arg is an empty iterable")
31     for x in it:
32         if x > max_val:
33             max_val = x
34     return max_val
35
36 if __name__ == "__main__":
37     try:
38         raw = input("Enter numbers separated by spaces or commas: ").strip()
39         numbers = parse_numbers(raw)
40         largest = max_in_list(numbers)
41     except ValueError as e:
42         print(f"Error: {e}")
43     else:
44         print(f"Largest number: {largest}")
45
46
```

Expected Output#5

A valid function with your review

Practical output:



The screenshot shows a terminal window with the following output:

```
PS C:\Users\rimsha> python -u "C:\Users\rimsha\AppData\Local\Temp\tempCodeRunnerFile.python"
Enter numbers separated by spaces or commas: 1 6 3 8 9 3
Largest number: 9
Largest number: 9
PS C:\Users\rimsha>
```