

SR UNIVERSITY

AI ASSIST CODING

Lab-1.2

ROLL NO:2503A51L17

NAME: Simra Tahseen

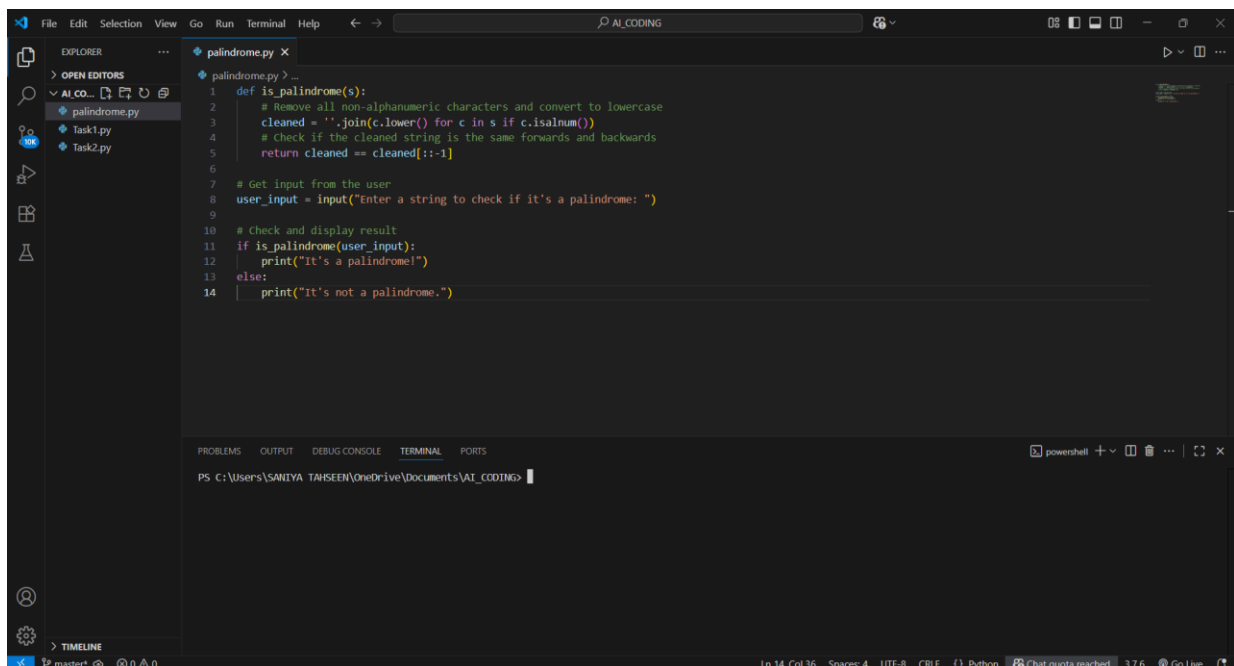
BATCH:19

TASK #1:

Prompt Used:

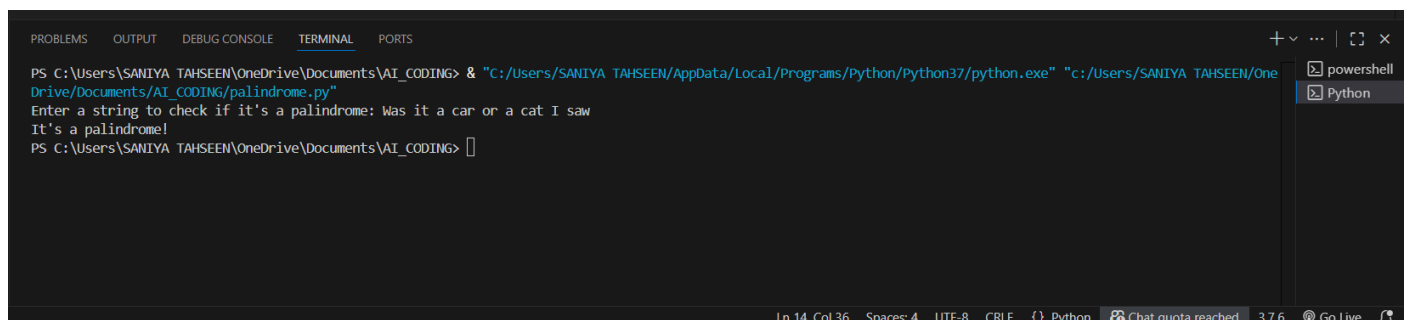
Write a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and allow Copilot to complete it.

Code Generated:



```
1 def is_palindrome(s):
2     # Remove all non-alphanumeric characters and convert to lowercase
3     cleaned = ''.join(c.lower() for c in s if c.isalnum())
4     # Check if the cleaned string is the same forwards and backwards
5     return cleaned == cleaned[::-1]
6
7 # Get input from the user
8 user_input = input("Enter a string to check if it's a palindrome: ")
9
10 # Check and display result
11 if is_palindrome(user_input):
12     print("It's a palindrome!")
13 else:
14     print("It's not a palindrome.")
```

Output After executing Code:



```
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/palindrome.py"
Enter a string to check if it's a palindrome: Was it a car or a cat I saw
It's a palindrome!
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

Observations:

- It cleans the input by removing all non-alphanumeric characters and converting everything to lowercase, ensuring accurate results regardless of punctuation, spacing, or case.

- Checks for palindrome by comparing the cleaned string to its reverse (cleaned[::-1]). If both match, the string is a palindrome.
- Works for sentences and phrases (not just single words), as spaces, punctuation, and cases are ignored in the check.

TASK #2:

Prompt Used:

Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring.

Code Generated:

```

fibonacci.py X
fibonacci.py > ...
1  def fibonacci(n):
2      """
3      Generate and return the Fibonacci sequence up to n terms.
4      """
5      sequence = []
6      a, b = 0, 1
7      for _ in range(n):
8          sequence.append(a)
9          a, b = b, a + b
10     return sequence
11
12 # Get user input
13 num_terms = int(input("Enter the number of Fibonacci terms to generate: "))
14
15 print(f"Fibonacci sequence up to {num_terms} terms:")
16 for num in fibonacci(num_terms):
17     print(num, end=' ')
18 print() # for newline after printing all numbers
19

```

Output After executing Code:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/Fibonacci.py"
Enter the number of Fibonacci terms to generate: 5
Fibonacci sequence up to 5 terms:
0 1 1 2 3
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/Fibonacci.py"
Enter the number of Fibonacci terms to generate: 9
Fibonacci sequence up to 9 terms:
0 1 1 2 3 5 8 13 21
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/Fibonacci.py"
Enter the number of Fibonacci terms to generate: 

```

Observations:

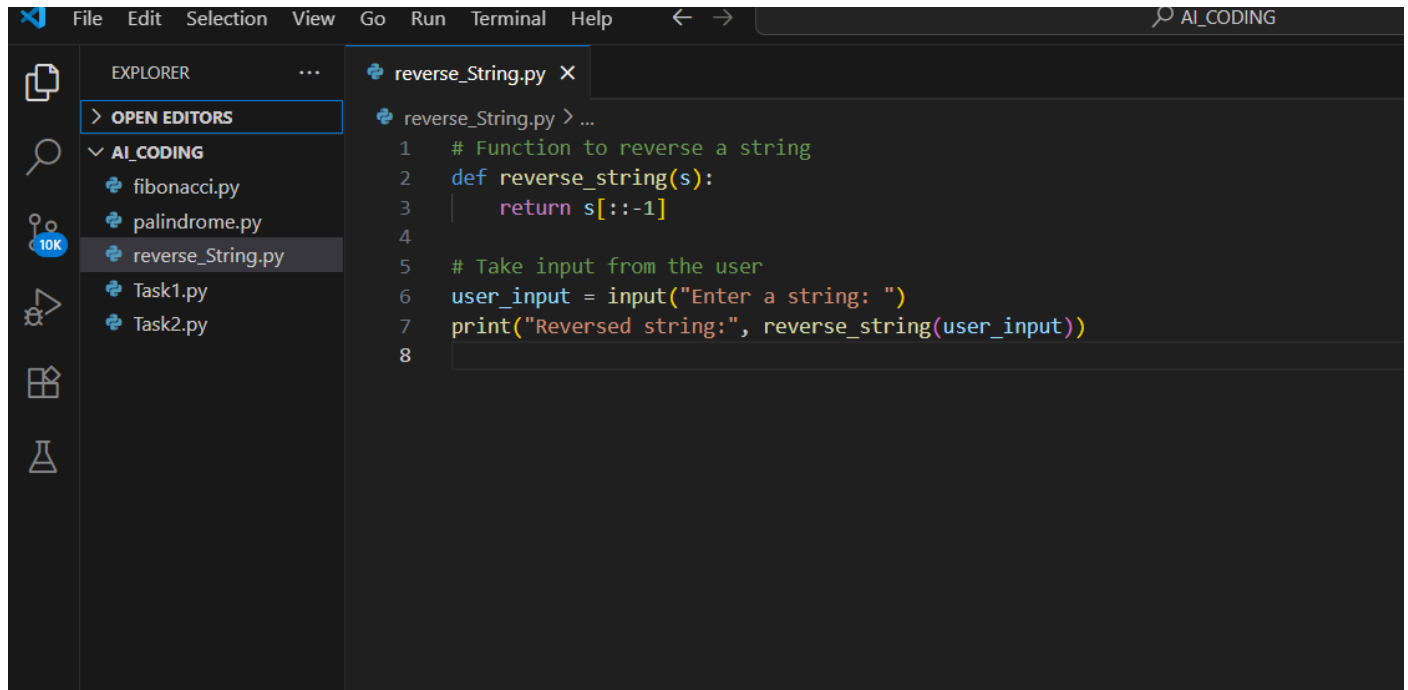
- The code generates the Fibonacci sequence up to a user-specified number of terms using a simple iterative approach.
- The sequence is generated by initializing the first two terms (a = 0, b = 1) and iteratively updating them with a, b = b, a + b, ensuring each new term is the sum of its two immediate predecessors.
- Each generated term is stored in a list, which is then returned and printed.

TASK #3:

Prompt Used:

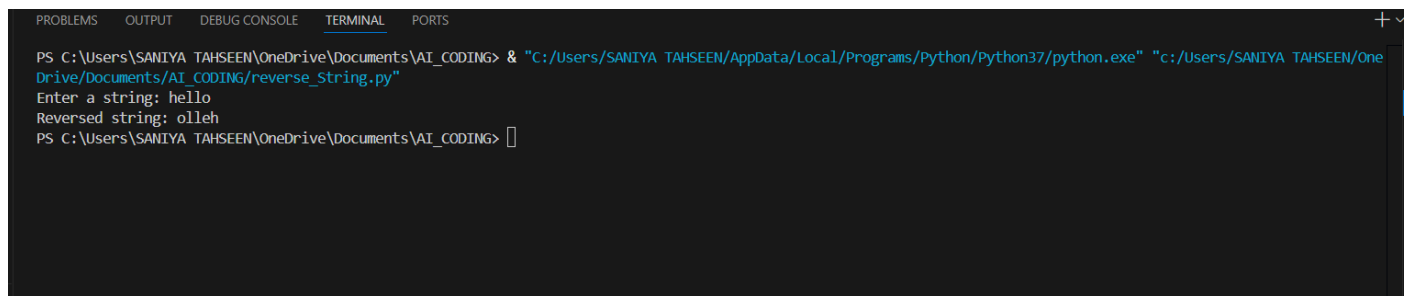
Write a comment like # Function to reverse a string and use Copilot to generate the function.

Code Generated:

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'AI_CODING' with files 'fibonacci.py', 'palindrome.py', 'reverse_String.py' (selected), 'Task1.py', and 'Task2.py'. The main editor window displays the code for 'reverse_String.py'. The code includes a comment, a function definition, and user input handling.

```
1 # Function to reverse a string
2 def reverse_string(s):
3     return s[::-1]
4
5 # Take input from the user
6 user_input = input("Enter a string: ")
7 print("Reversed string:", reverse_string(user_input))
8
```

Output After executing Code:

A screenshot of a terminal window showing the execution of the Python script. The prompt is 'PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>'. The command executed is '& "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/reverse_String.py"'. The output shows the user entering 'hello' and the program printing 'Reversed string: olleh'.

```
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/reverse_String.py"
Enter a string: hello
Reversed string: olleh
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

Observations:

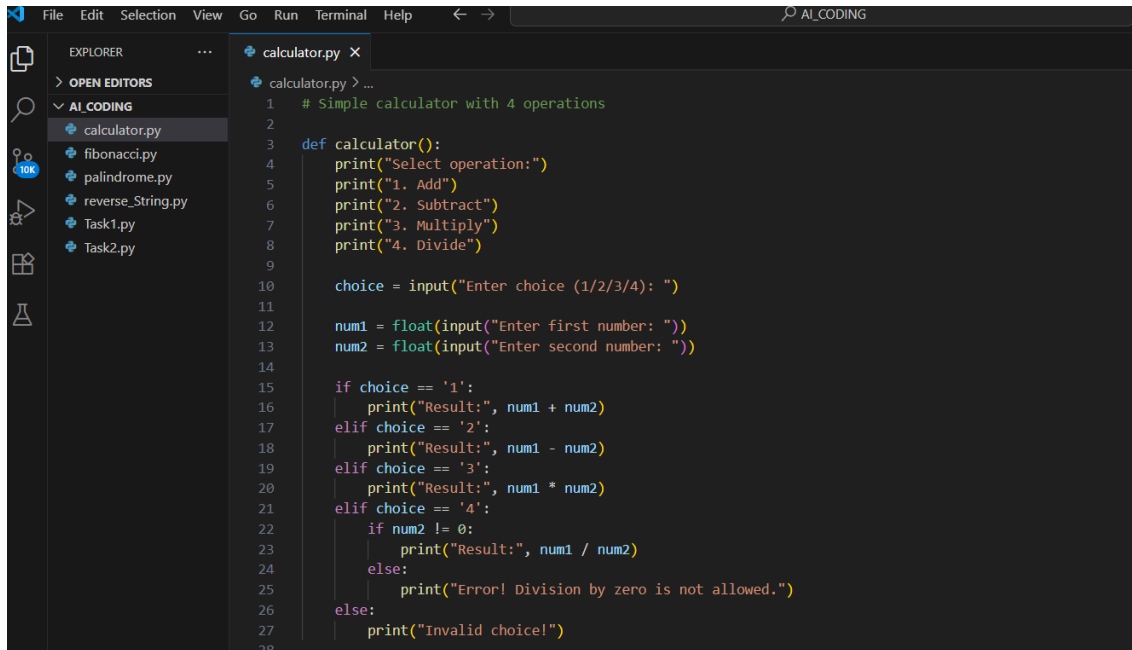
- The function uses Python's string slicing syntax `s[::-1]` to reverse the string.
- The slice step of `-1` means characters are taken from end to start, effectively reversing the string.
- User input is taken and passed to the function, with the reversed result printed.

TASK #4:

Prompt Used:

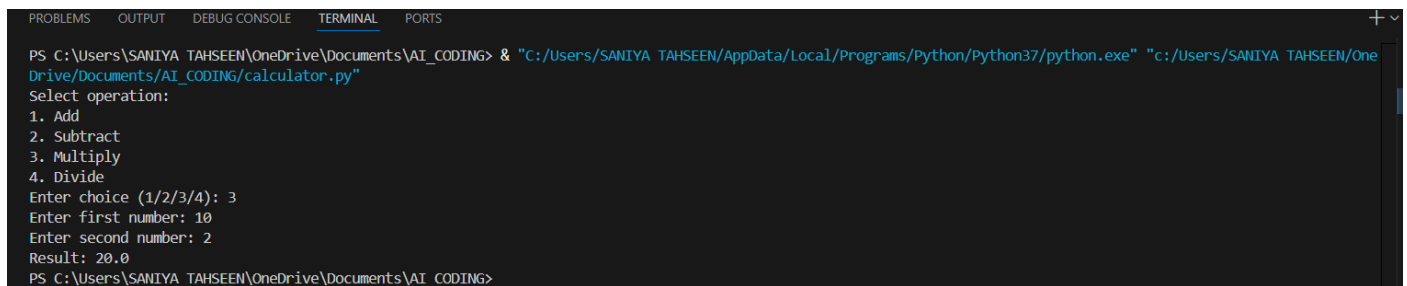
Generate a program that simulates a basic calculator (add, subtract, multiply, divide).
Write the comment: # Simple calculator with 4 operations and let AI complete it.

Code Generated:

A screenshot of a code editor window. The left sidebar shows a file explorer with a folder named 'AI_CODING' containing several files: calculator.py, fibonacci.py, palindrome.py, reverse_String.py, Task1.py, and Task2.py. The main editor area displays the code for 'calculator.py'. The code is a simple calculator with four operations: Add, Subtract, Multiply, and Divide. It uses the 'def' keyword to define a function 'calculator()' and includes input prompts for the user to select an operation and enter numbers. It also includes error handling for division by zero and invalid choices.

```
1 # Simple calculator with 4 operations
2
3 def calculator():
4     print("Select operation:")
5     print("1. Add")
6     print("2. Subtract")
7     print("3. Multiply")
8     print("4. Divide")
9
10    choice = input("Enter choice (1/2/3/4): ")
11
12    num1 = float(input("Enter first number: "))
13    num2 = float(input("Enter second number: "))
14
15    if choice == '1':
16        print("Result:", num1 + num2)
17    elif choice == '2':
18        print("Result:", num1 - num2)
19    elif choice == '3':
20        print("Result:", num1 * num2)
21    elif choice == '4':
22        if num2 != 0:
23            print("Result:", num1 / num2)
24        else:
25            print("Error! Division by zero is not allowed.")
26    else:
27        print("Invalid choice!")
28
```

Output After executing Code:

A screenshot of a terminal window showing the execution of the calculator program. The prompt is 'PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>'. The user enters the command to run the calculator.py file. The program prompts the user to 'Select operation:' and lists the four options. The user enters '3'. The program then prompts for 'Enter choice (1/2/3/4):' and the user enters '3'. The program then prompts for 'Enter first number:' and the user enters '10'. The program then prompts for 'Enter second number:' and the user enters '2'. The final output is 'Result: 20.0'.

```
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/calculator.py"
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice (1/2/3/4): 3
Enter first number: 10
Enter second number: 2
Result: 20.0
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

Observations:

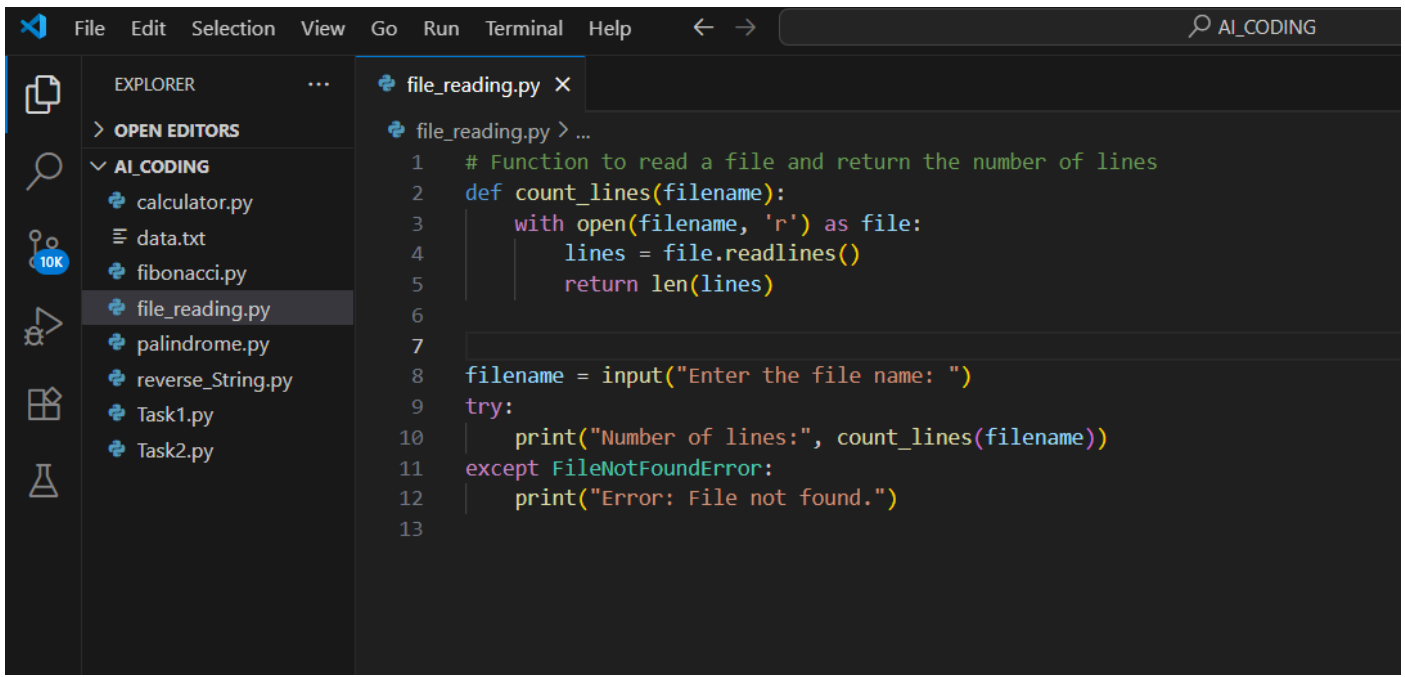
- The program allows the user to choose from four basic arithmetic operations.
- It accepts user input as float, enabling both integer and decimal calculations.
- Division includes error handling to avoid division by zero.

TASK #5:

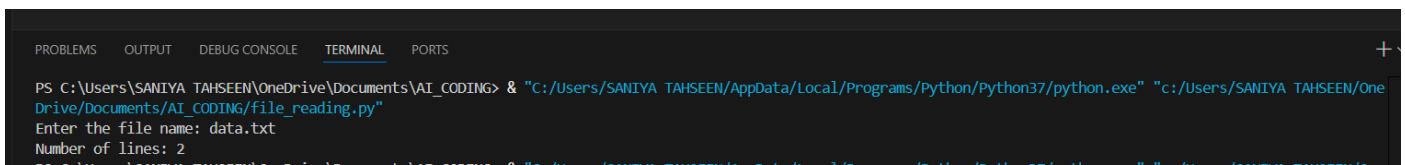
Prompt Used:

Use a comment to instruct AI to write a function that reads a file and returns the number of lines

Code Generated:



Output After executing Code:



Observations:

- The program defines a function `count_lines()` that opens a file and counts the number of lines using `readlines()`.
- It uses `with open()` to ensure the file is properly handled and closed automatically.
- The program takes the filename as user input, making it flexible for different files.
- Error handling with `try-except` prevents crashes in case the file is not found.