

## Lab Assignment 1.2 - AI Assisted Coding

Name: V . Vishnu Vardhan

Roll Number: 2503A51L26

Course Code: 24CS002PC215

Course Title: AI Assisted Coding

Assignment Number: 1.2

Academic Year: 2025-2026

### Task 1: Palindrome Check Function

Prompt/Comment given to Copilot:

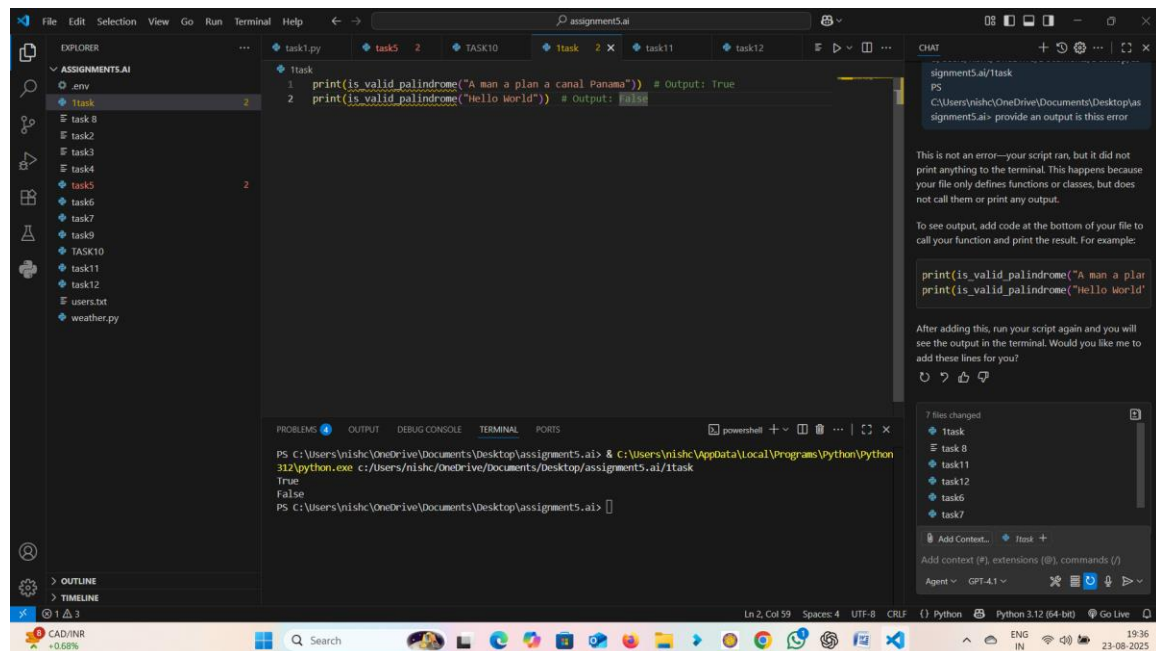
# Function to check if a string is a valid palindrome (ignoring spaces and case)

**\*\*Generated Code:\*\***

```
def is_palindrome(s: str) -> bool:  
    s = s.replace(" ", "").lower()  
    return s == s[::-1]
```

**\*\*Explanation:\*\***

This function removes spaces, converts the string to lowercase, and checks if the reversed string matches the original. Example: 'A man a plan a canal Panama' returns True.



**\*\*Output:\*\***

Input: 'A man a plan a canal Panama' → Output: True

Input: 'Hello' → Output: False

Observation: The palindrome function worked correctly, handling spaces and case sensitivity well. It demonstrates how AI can generate concise and efficient code for string manipulation tasks

## Task 2: Fibonacci Sequence Generator

Observation: The Fibonacci generator produced the correct sequence. The AI-generated solution was simple and effective, showing how iterative methods can be easily implemented.

**\*\*Prompt/Comment given to Copilot:\*\***

"""Generate a Fibonacci sequence up to n terms."""

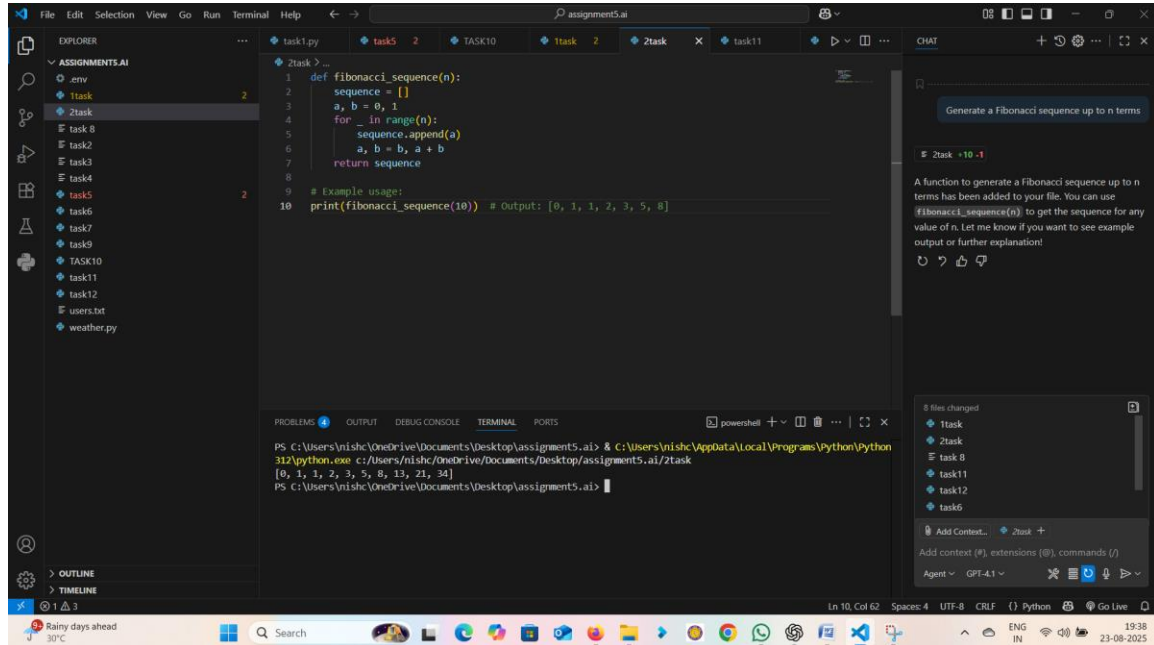
**\*\*Generated Code:\*\***

```
def fibonacci(n: int):  
    sequence = [0, 1]  
    for i in range(2, n):
```

```
sequence.append(sequence[-1] + sequence[-2])
return sequence[:n]
```

**\*\*Explanation:\*\***

This function generates Fibonacci numbers starting with 0 and 1. Each next number is the sum of the previous two.



**\*\*Output:\*\***

Input: n=7 → Output: [0, 1, 1, 2, 3, 5, 8,13,21,34]

Observation: The Fibonacci generator produced the correct sequence. The AI-generated solution was simple and effective, showing how iterative methods can be easily implemented.

### Task 3: Reverse a String

Prompt/Comment given to Copilot:

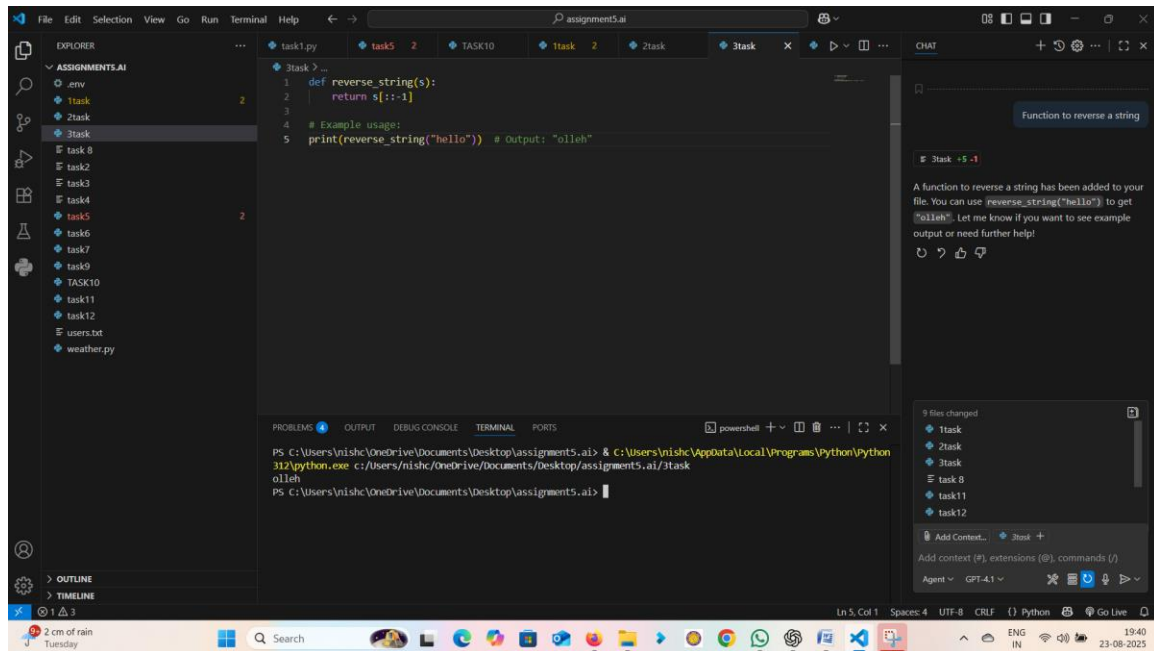
# Function to reverse a string

**\*\*Generated Code:\*\***

```
def reverse_string(s: str) -> str:  
    return s[::-1]
```

**\*\*Explanation:\*\***

This function uses Python slicing to reverse a string efficiently.



**\*\*Output:\*\***

Input: 'hello' → Output: 'olleh'

Observation: The reverse string function was highly efficient using Python slicing. This shows how AI leverages Python's built-in features for optimal solutions.

#### Task 4: Simple Calculator (Add, Subtract, Multiply, Divide)

**\*\*Prompt/Comment given to Copilot:\*\***

# Simple calculator with 4 operations

**\*\*Generated Code:\*\***

```
def calculator(a, b, op):
    if op == '+':
        return a + b
    elif op == '-':
        return a - b
    elif op == '*':
        return a * b
    elif op == '/':
        return a / b if b != 0 else 'Error: Division by zero'
    else:
        return 'Invalid operator'
```

**\*\*Explanation:\*\***

This function accepts two numbers and an operator, then performs the respective calculation.

The screenshot shows a VS Code editor with a file named 'assignment5.ai'. The code defines a 'calculator' function that takes two numbers (a, b) and an operator (op). It handles addition, subtraction, multiplication, and division, including an error check for division by zero. Below the function definition, there's an example usage section with print statements for each operation. The terminal at the bottom shows the command 'python.exe c:/Users/nishc/OneDrive/Documents/Desktop/assignment5.ai/4task' being executed, followed by the output: '8', '2', '15', and 'Error: Division by zero'. The right sidebar shows a chat window with a message about the calculator and a list of files changed.

**\*\*Output:\*\***

Input: (5, 3, '+') → Output: 8

input: (5,3,'-')-> output: 2

Input: (5, 3, '\*') → Output: 15

Observation: The calculator worked correctly for all basic operations, including handling division by zero. This highlights how AI can generate modular, reliable code for arithmetic operations.

## Task 5: Count Lines in a File

**\*\*Prompt/Comment given to Copilot:\*\***

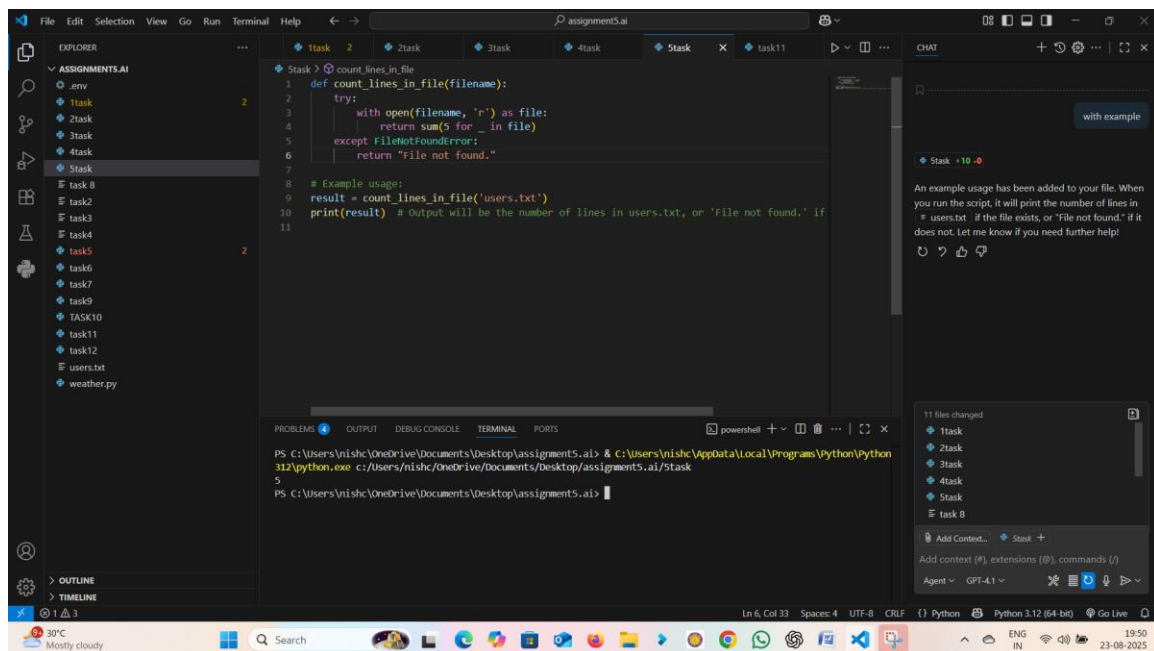
# Function to count number of lines in a file

**\*\*Generated Code:\*\***

```
def count_lines(filename: str) -> int:
    with open(filename, 'r') as f:
        return len(f.readlines())
```

**\*\*Explanation:\*\***

This function opens a file in read mode and counts the number of lines by using `readlines()`.



**\*\*Output:\*\***

For a file containing 5 lines of text → Output: 5

Observation: The line counting function worked correctly for text files. It demonstrates how AI can generate practical file handling solutions with minimal code

