Lab 1: Environment Setup
GitHub Copilot and VS Code Integration

Course Code: 24CSBTB41
Course Title: Assistant Coding
Assignment No: 1.2
Roll Number: 2503A51L36

Lab Objectives

- 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.

Lab Outcomes

After completing this lab, students will be able to:
- 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.

Task 1:Valid palindrom

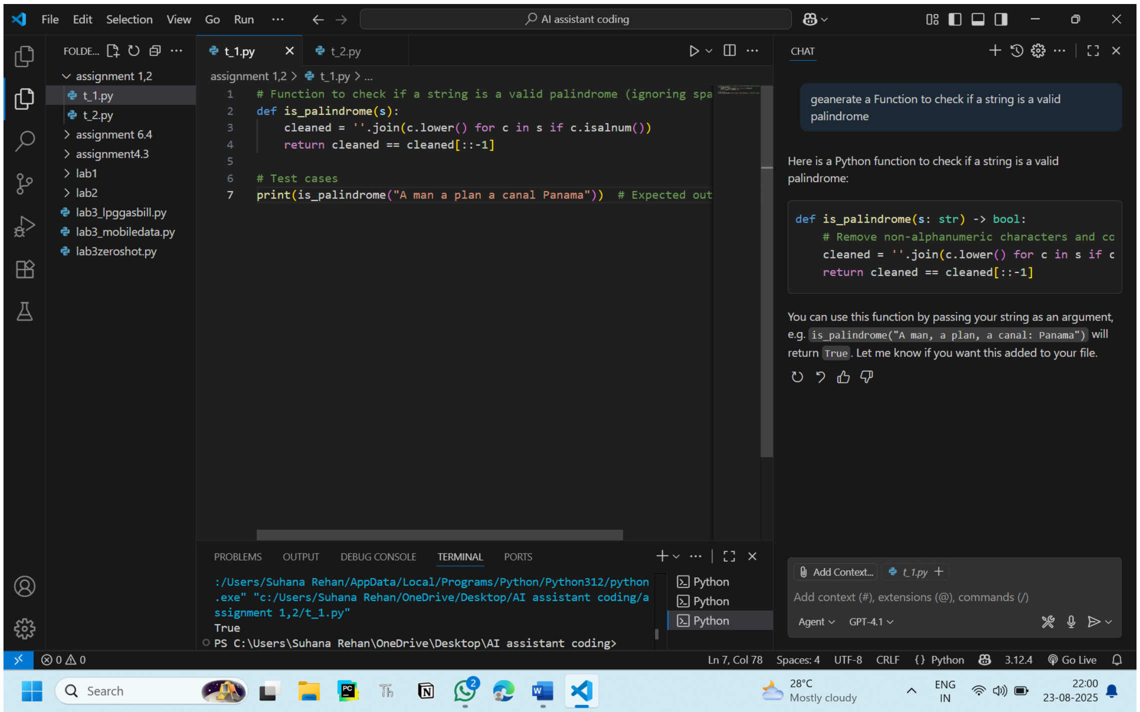
Prompt: Write a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and allow Copilot to complete it.

Code:

```
# Function to check if a string is a valid palindrome (ignoring spaces and
case)
def is_palindrome(s):
    cleaned = ''.join(c.lower() for c in s if c.isalnum())
    return cleaned == cleaned[::-1]

# Test cases
print(is_palindrome("A man a plan a canal Panama"))  # Expected output: False
```

OBSERVATION: Here the AI Complete the fuction to find if srting is a vaild palindrome, it can be an example of Zero-Shot prompt



assignment 1,2 > t_1.py > ...

```python
# Function to check if a string is a valid palindrome (ignoring spa
def is_palindrome(s):
    cleaned = ''.join(c.lower() for c in s if c.isalnum())
    return cleaned == cleaned[::-1]

# Test cases
print(is_palindrome("A man a plan a canal Panama"))  # Expected out
```

geanerate a Function to check if a string is a valid palindrome

Here is a Python function to check if a string is a valid palindrome:

```python
def is_palindrome(s: str) -> bool:
    # Remove non-alphanumeric characters and co
    cleaned = ''.join(c.lower() for c in s if c
    return cleaned == cleaned[::-1]
```

You can use this function by passing your string as an argument, e.g. `is_palindrome("A man, a plan, a canal: Panama")` will return `True`. Let me know if you want this added to your file.

```
:/Users/Suhana Rehan/AppData/Local/Programs/Python/Python312/python
.exe" "c:/Users/Suhana Rehan/OneDrive/Desktop/AI assistant coding/a
ssignment 1,2/t_1.py"
True
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding>
```

Expected Output:

True

Task 2: Fibonacci Fuction

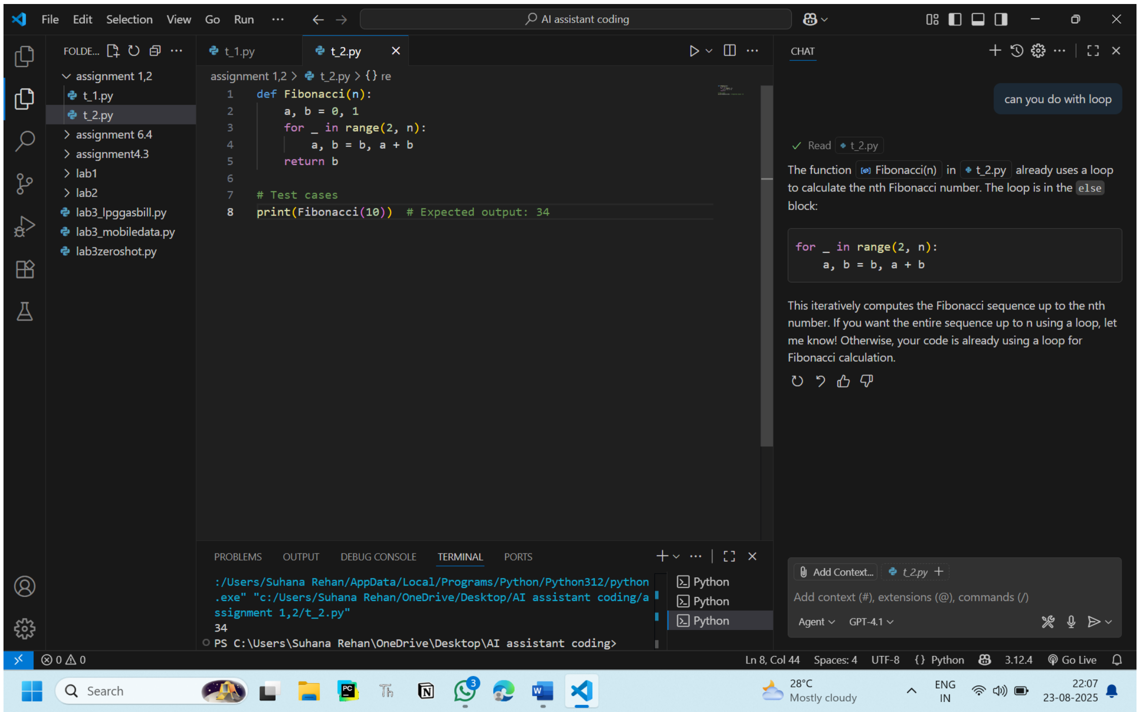
Prompt: Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring

Code:

```python
def Fibonacci(n):
    if n <= 0:
        return "Input should be a positive integer."
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        a, b = 0, 1
        for _ in range(2, n):
            a, b = b, a + b
        return b


# Test cases
print(Fibonacci(10))  # Expected output: 34
```

OBSERVATION: The AI completed the program even when I just wrote the head of function



```python
def Fibonacci(n):
    a, b = 0, 1
    for _ in range(2, n):
        a, b = b, a + b
    return b

# Test cases
print(Fibonacci(10))  # Expected output: 34
```

can you do with loop

✓ Read t_2.py

The function Fibonacci(n) in t_2.py already uses a loop to calculate the nth Fibonacci number. The loop is in the `else` block:

```python
for _ in range(2, n):
    a, b = b, a + b
```

This iteratively computes the Fibonacci sequence up to the nth number. If you want the entire sequence up to n using a loop, let me know! Otherwise, your code is already using a loop for Fibonacci calculation.

```
:/Users/Suhana Rehan/AppData/Local/Programs/Python/Python312/python
.exe" "c:/Users/Suhana Rehan/OneDrive/Desktop/AI assistant coding/a
ssignment 1,2/t_2.py"
34
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding>
```

Expected Output:

34

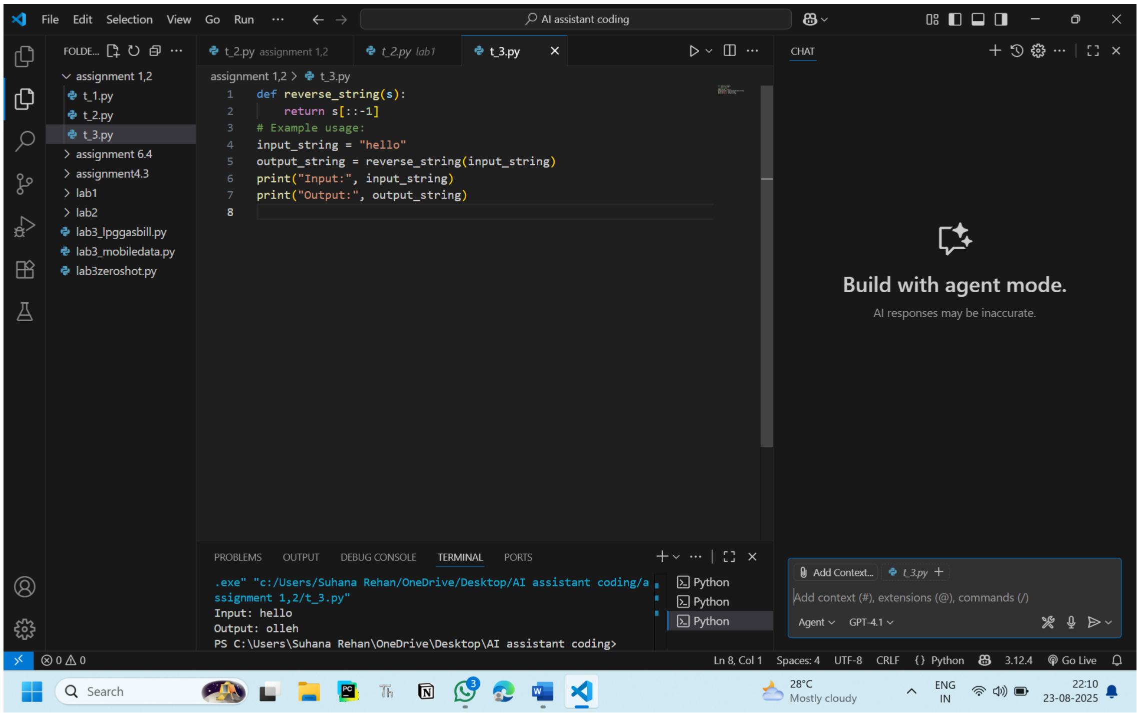
Task 3: Function to reverse a string

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

Code:

```python
def reverse_string(s):
   return s[::-1]
# Example usage:
input_string = "hello"
output_string = reverse_string(input_string)
print("Input:", input_string)
print("Output:", output_string)
```

OBSERVATION: AI complete the function, When I just wrote the 2 lines. Hence, it Auto-generated



```
def reverse_string(s):
    return s[::-1]
# Example usage:
input_string = "hello"
output_string = reverse_string(input_string)
print("Input:", input_string)
print("Output:", output_string)
```

```
.exe" "c:/Users/Suhana Rehan/OneDrive/Desktop/AI assistant coding/a
ssignment 1,2/t_3.py"
Input: hello
Output: olleh
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding>
```

Expected Output:

Input: hello
Output: olleh

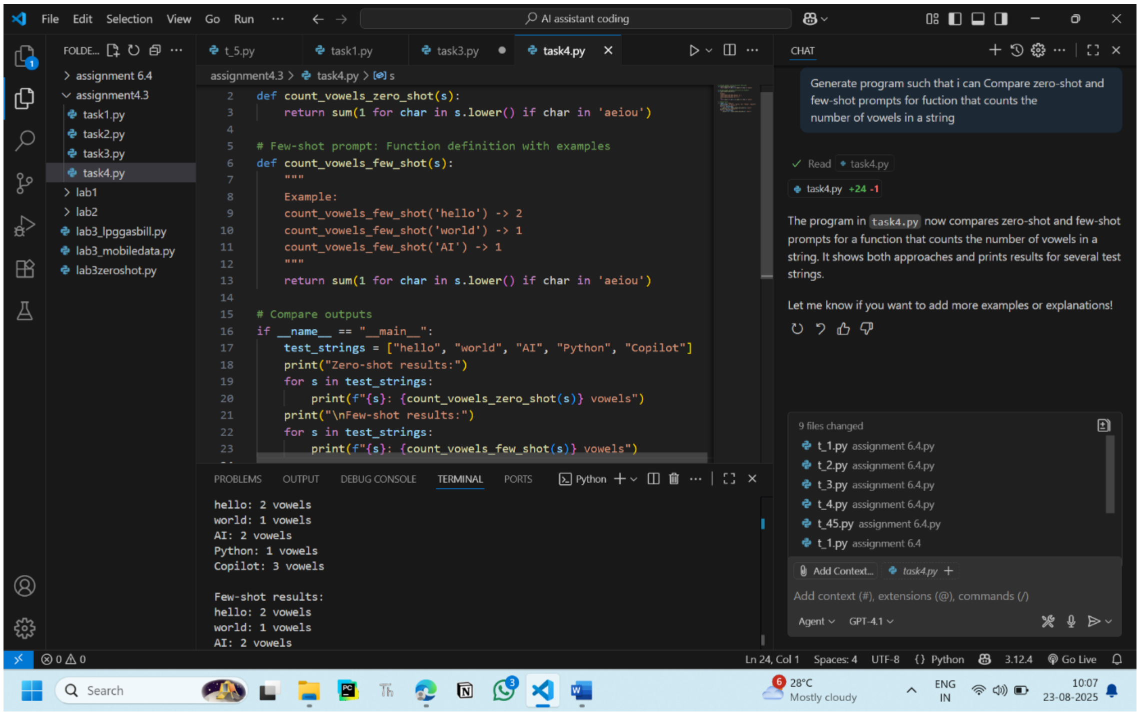
Task 4: Basic Calculator

Prompt: 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:

```
#function basic_calculator
def basic_calculator(num1, num2, operator):
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 != 0:
            return num1 / num2
        else:
            return "Error: Division by zero"
    else:
        return "Error: Invalid operator"
print(basic_calculator(10, 5, '+'))  # Expected output: 15
```

OBSERVATION: AI generated the rest of function, When I wrote just basic_calculators



```python
def count_vowels_zero_shot(s):
    return sum(1 for char in s.lower() if char in 'aeiou')

# Few-shot prompt: Function definition with examples
def count_vowels_few_shot(s):
    """
    Example:
    count_vowels_few_shot('hello') -> 2
    count_vowels_few_shot('world') -> 1
    count_vowels_few_shot('AI') -> 1
    """
    return sum(1 for char in s.lower() if char in 'aeiou')

# Compare outputs
if __name__ == "__main__":
    test_strings = ["hello", "world", "AI", "Python", "Copilot"]
    print("Zero-shot results:")
    for s in test_strings:
        print(f"{s}: {count_vowels_zero_shot(s)} vowels")
    print("\nFew-shot results:")
    for s in test_strings:
        print(f"{s}: {count_vowels_few_shot(s)} vowels")
```

```
hello: 2 vowels
world: 1 vowels
AI: 2 vowels
Python: 1 vowels
Copilot: 3 vowels

Few-shot results:
hello: 2 vowels
world: 1 vowels
AI: 2 vowels
```

Generate program such that i can Compare zero-shot and few-shot prompts for fuction that counts the number of vowels in a string

✓ Read task4.py

task4.py +24 -1

The program in `task4.py` now compares zero-shot and few-shot prompts for a function that counts the number of vowels in a string. It shows both approaches and prints results for several test strings.

Let me know if you want to add more examples or explanations!

9 files changed
t_1.py assignment 6.4.py
t_2.py assignment 6.4.py
t_3.py assignment 6.4.py
t_4.py assignment 6.4.py
t_45.py assignment 6.4.py
t_1.py assignment 6.4

File Edit Selection View Go Run
AI assistant coding
t_5.py task1.py task3.py task4.py
assignment4.3 > task4.py > s
assignment 6.4
assignment4.3
task1.py
task2.py
task3.py
task4.py
lab1
lab2
lab3_lpggasbill.py
lab3_mobiledata.py
lab3zeroshot.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
CHAT
Add Context...
task4.py
Add context (#), extensions (@), commands (/)
Agent
GPT-4.1
Ln 24, Col 1 Spaces: 4 UTF-8 {} Python 3.12.4 Go Live
Search
28°C
Mostly cloudy
ENG
IN
10:07
23-08-2025

Expected Output:

15

Task 5: Read a File

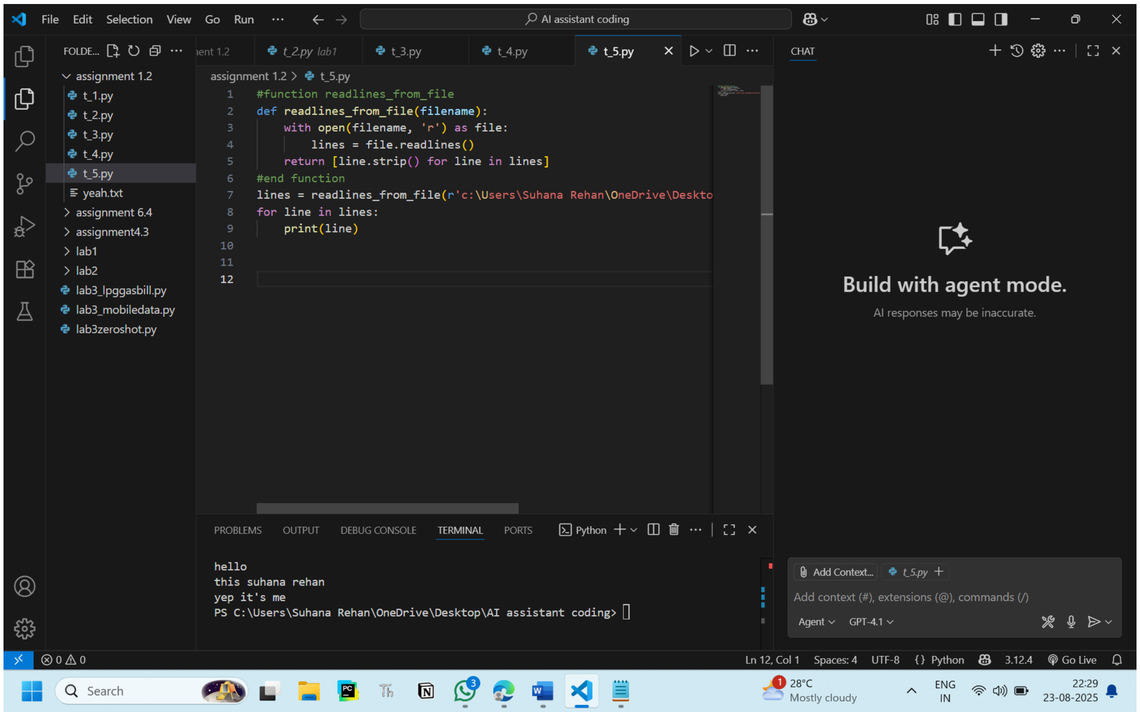
Prompt: Use a comment to instruct AI to write a function that reads a file and returns the number of lines..

Code:

```
#function readlines_from_file
def readlines_from_file(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()
    return [line.strip() for line in lines]
#end function
lines = readlines_from_file(r'c:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding\assignment 1.2\yeah.txt')
```

```
for line in lines:
    print(line)
```

OBSERVATION: AI completed the fuction of readlines from the given file



```python
#function readlines_from_file
def readlines_from_file(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()
    return [line.strip() for line in lines]
#end function
lines = readlines_from_file(r'c:\Users\Suhana Rehan\OneDrive\Deskto
for line in lines:
    print(line)
```

```
hello
this suhana rehan
yep it's me
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding> 
```

Expected Output:
hello
this suhana rehan
yep it's me