# Lab Assignment 4.3 - AI Assisted Coding

Name: V.VISHNU VARDHAN

Roll Number: 2503A51L26

Course Code: 24CS002PC215

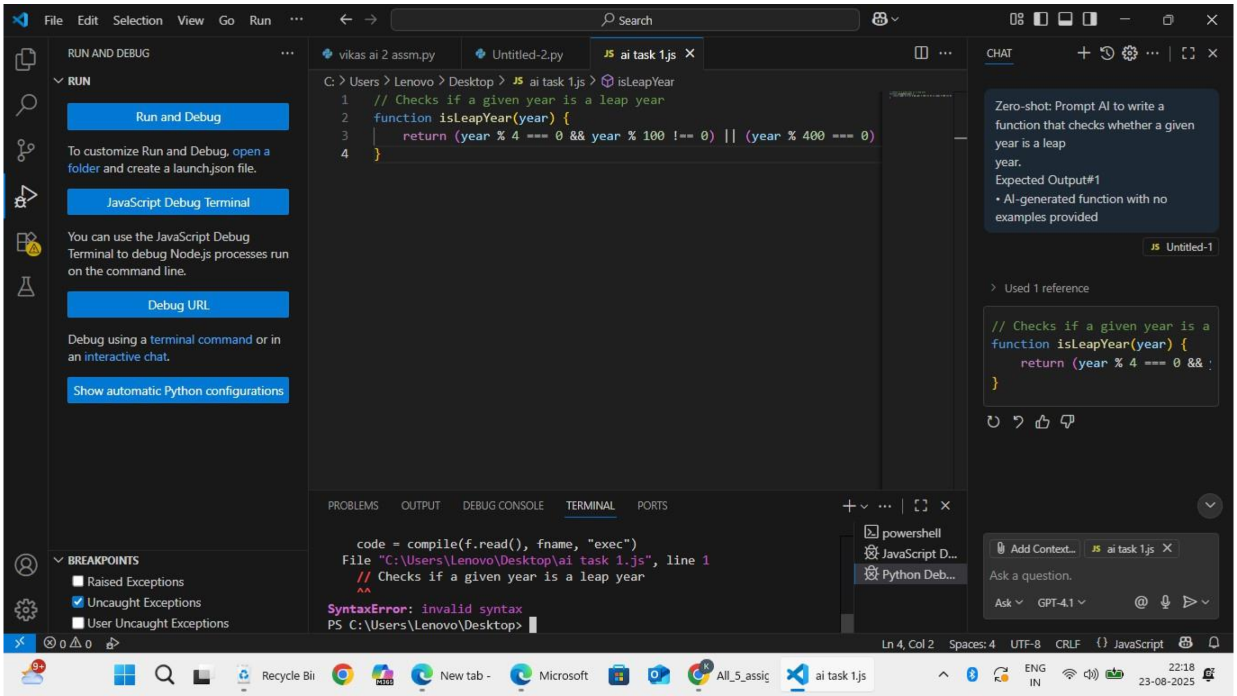
Course Title: AI Assisted Coding

Assignment Number: 4.3

Academic Year: 2025-2026

Task:1

- Zero-shot: Prompt AI to write a function that checks whether a given year is a leapyear



To customize Run and Debug, open a folder and create a launch.json file.

You can use the JavaScript Debug Terminal to debug Node.js processes run on the command line.

Debug using a terminal command or in an interactive chat.

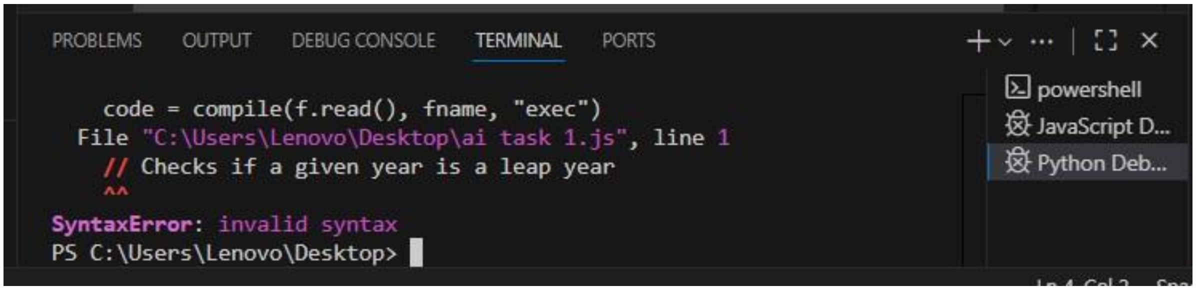
```
// Checks if a given year is a leap year
function isLeapYear(year) {
    return (year % 4 === 0 && year % 100 !== 0) || (year % 400 === 0)
}
```

Zero-shot: Prompt AI to write a function that checks whether a given year is a leap
year.
Expected Output#1
• AI-generated function with no examples provided

```
// Checks if a given year is a
function isLeapYear(year) {
    return (year % 4 === 0 && :
}
```

```
    code = compile(f.read(), fname, "exec")
  File "C:\Users\Lenovo\Desktop\ai task 1.js", line 1
    // Checks if a given year is a leap year
    ^^
SyntaxError: invalid syntax
PS C:\Users\Lenovo\Desktop>
```

Out put:



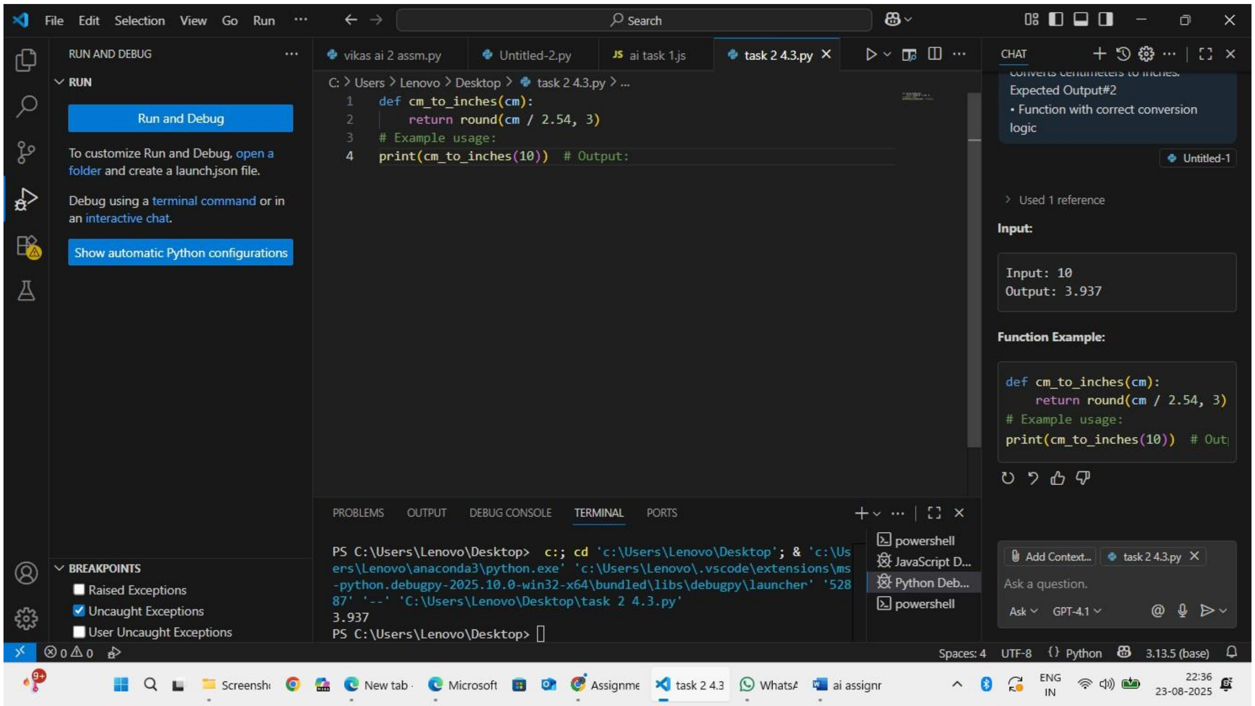
```
    code = compile(f.read(), fname, "exec")
  File "C:\Users\Lenovo\Desktop\ai task 1.js", line 1
    // Checks if a given year is a leap year
    ^^
SyntaxError: invalid syntax
PS C:\Users\Lenovo\Desktop>
```

Observation :

- The function isLeapYear correctly implements the leap year logic using JavaScript.
- It returns true for leap years and false otherwise.
- The function expects a single argument: the year (as a number).
- There are no input validations or example usages in the file
- The code is concise and readable.

## Task :2

ne-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches



To customize Run and Debug, open a folder and create a launch.json file.

Debug using a terminal command or in an interactive chat.

BREAKPOINTS

☐ Raised Exceptions

☑ Uncaught Exceptions

☐ User Uncaught Exceptions

```
def cm_to_inches(cm):
    return round(cm / 2.54, 3)
# Example usage:
print(cm_to_inches(10))  # Output:
```

converts centimeters to inches.
Expected Output#2
• Function with correct conversion logic

Used 1 reference

**Input:**

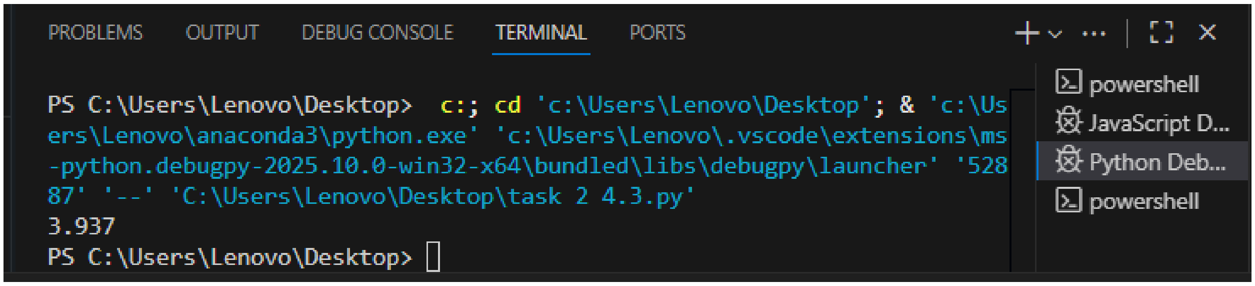
```
Input: 10
Output: 3.937
```

**Function Example:**

```
def cm_to_inches(cm):
    return round(cm / 2.54, 3)
# Example usage:
print(cm_to_inches(10))  # Out
```

```
PS C:\Users\Lenovo\Desktop>  c:; cd 'c:\Users\Lenovo\Desktop'; & 'c:\Us
ers\Lenovo\anaconda3\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms
-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '528
87' '--' 'C:\Users\Lenovo\Desktop\task 2 4.3.py'
3.937
PS C:\Users\Lenovo\Desktop> 
```

Out put:



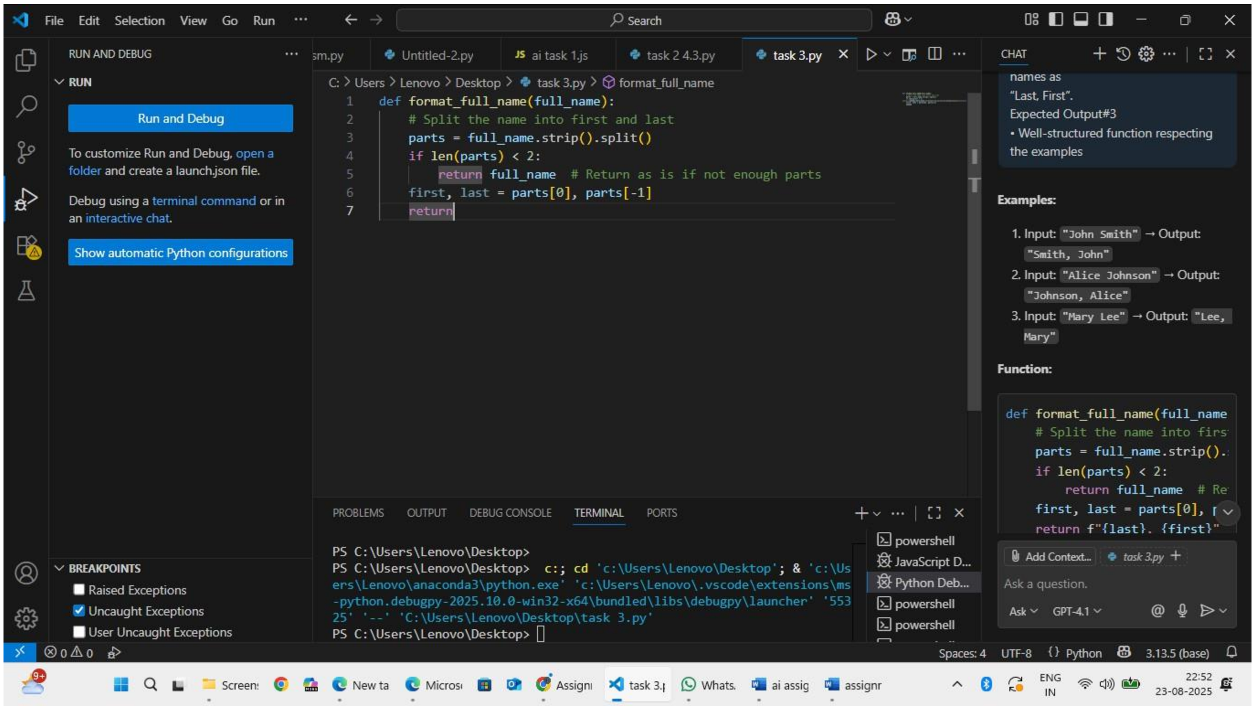
```
PS C:\Users\Lenovo\Desktop>  c:; cd 'c:\Users\Lenovo\Desktop'; & 'c:\Us
ers\Lenovo\anaconda3\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms
-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '528
87' '--' 'C:\Users\Lenovo\Desktop\task 2 4.3.py'
3.937
PS C:\Users\Lenovo\Desktop> 
```

Observation :

The function cm_to_inches(cm) correctly implements the conversion from centimeters to inches by dividing the input by 2.54 and rounding to three decimal places. The example usage prints the result for an input of 10 cm, but the expected output value (3.937) is missing in the comment. The code is syntactically correct and will output 3.937 when executed.

## Task :3

Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”



```python
def format_full_name(full_name):
    # Split the name into first and last
    parts = full_name.strip().split()
    if len(parts) < 2:
        return full_name  # Return as is if not enough parts
    first, last = parts[0], parts[-1]
    return
```

```
PS C:\Users\Lenovo\Desktop>
PS C:\Users\Lenovo\Desktop>  c:; cd 'c:\Users\Lenovo\Desktop'; & 'c:\Us
ers\Lenovo\anaconda3\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms
-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '553
25' '--' 'C:\Users\Lenovo\Desktop\task 3.py'
PS C:\Users\Lenovo\Desktop>
```

names as
“Last, First”.
Expected Output#3
- Well-structured function respecting the examples

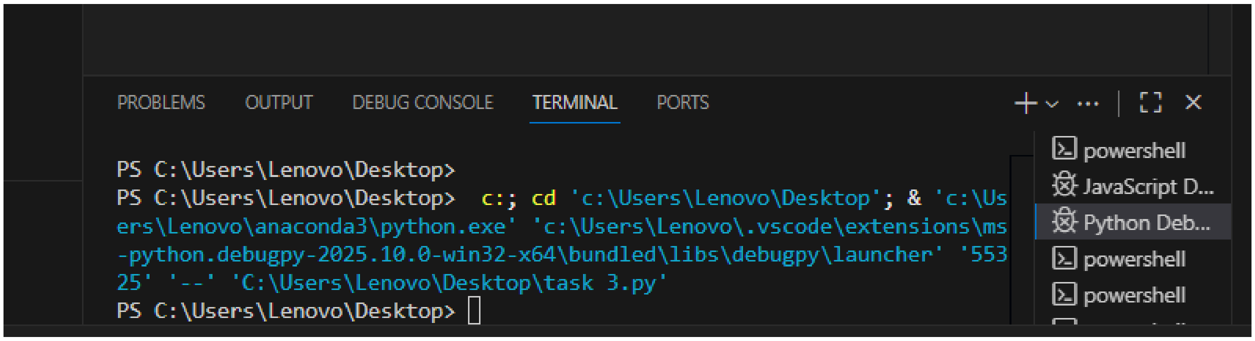
**Examples:**

1. Input: `"John Smith"` → Output: `"Smith, John"`
2. Input: `"Alice Johnson"` → Output: `"Johnson, Alice"`
3. Input: `"Mary Lee"` → Output: `"Lee, Mary"`

**Function:**

```python
def format_full_name(full_name
    # Split the name into firs
    parts = full_name.strip().
    if len(parts) < 2:
        return full_name  # Re
    first, last = parts[0], 
    return f"{last}, {first}"
```

Out Put:



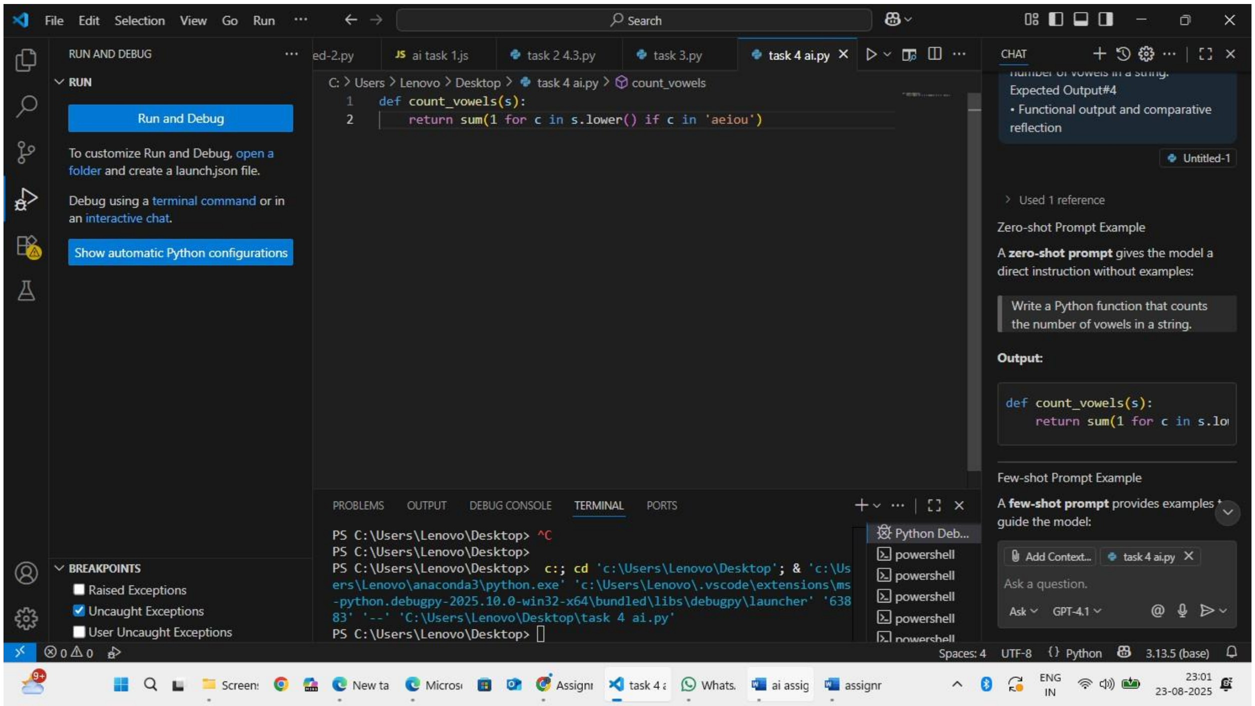
```
PS C:\Users\Lenovo\Desktop>
PS C:\Users\Lenovo\Desktop>  c:; cd 'c:\Users\Lenovo\Desktop'; & 'c:\Us
ers\Lenovo\anaconda3\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms
-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '553
25' '--' 'C:\Users\Lenovo\Desktop\task 3.py'
PS C:\Users\Lenovo\Desktop>
```

## Observation:

The function assumes the input is a two-part name ("First Last"). It splits the string and rearranges it as "Last, First". If the input does not contain at least two parts, it returns the original string unchanged. This approach works well for simple names but may not handle middle names or compound surnames accurately

## Task:4

Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string



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

Expected Output#4

- Functional output and comparative reflection

Zero-shot Prompt Example

A **zero-shot prompt** gives the model a direct instruction without examples:

> Write a Python function that counts the number of vowels in a string.

**Output:**

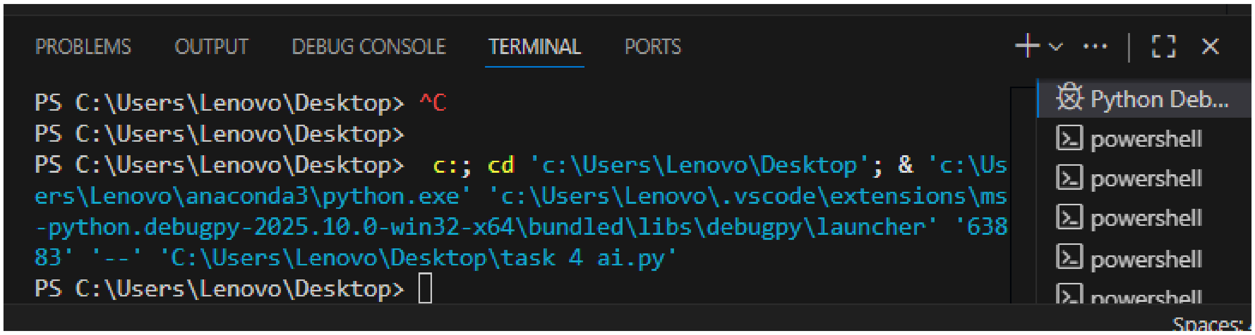
```python
def count_vowels(s):
    return sum(1 for c in s.lo
```

Few-shot Prompt Example

A **few-shot prompt** provides examples t guide the model:

```
PS C:\Users\Lenovo\Desktop> ^C
PS C:\Users\Lenovo\Desktop>
PS C:\Users\Lenovo\Desktop>  c:; cd 'c:\Users\Lenovo\Desktop'; & 'c:\Us
ers\Lenovo\anaconda3\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms
-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '638
83' '--' 'C:\Users\Lenovo\Desktop\task 4 ai.py'
PS C:\Users\Lenovo\Desktop> 
```

## Out put:



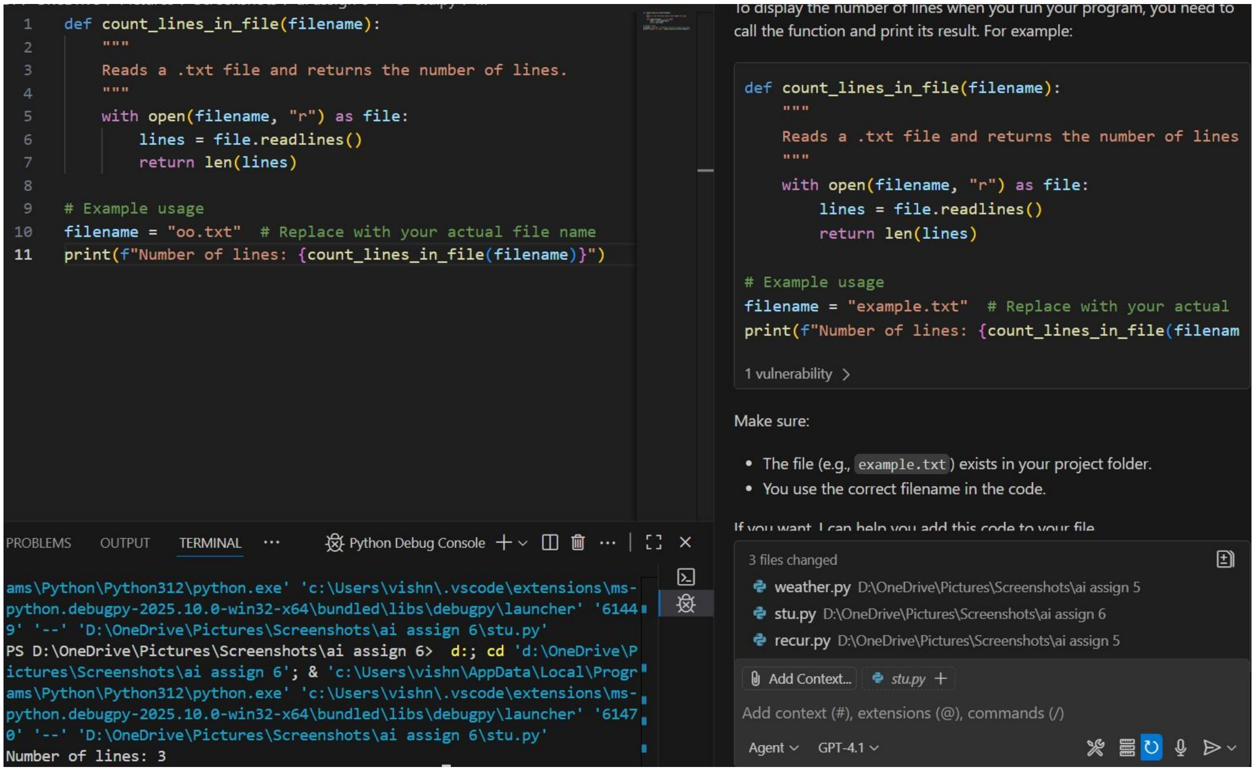
```
PS C:\Users\Lenovo\Desktop> ^C
PS C:\Users\Lenovo\Desktop>
PS C:\Users\Lenovo\Desktop>  c:; cd 'c:\Users\Lenovo\Desktop'; & 'c:\Us
ers\Lenovo\anaconda3\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms
-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '638
83' '--' 'C:\Users\Lenovo\Desktop\task 4 ai.py'
PS C:\Users\Lenovo\Desktop> 
```

## Observation:

The count_vowels function efficiently counts the number of vowels in a string by iterating through each character (converted to lowercase) and summing those that match any of the vowels 'aeiou'. The implementation is concise and leverages a generator expression for optimal performance. No error handling is present, but for typical string inputs, the function works as intended.

## TASK 5

Use few-shot prompting to generate a function that reads a .txt file and returns thenumber of lines.



```python
def count_lines_in_file(filename):
    """
    Reads a .txt file and returns the number of lines.
    """
    with open(filename, "r") as file:
        lines = file.readlines()
        return len(lines)

# Example usage
filename = "oo.txt"  # Replace with your actual file name
print(f"Number of lines: {count_lines_in_file(filename)}")
```

```
ams\Python\Python312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-
python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '6144
9' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign 6\stu.py'
PS D:\OneDrive\Pictures\Screenshots\ai assign 6>  d:; cd 'd:\OneDrive\P
ictures\Screenshots\ai assign 6'; & 'c:\Users\vishn\AppData\Local\Progr
ams\Python\Python312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-
python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '6147
0' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign 6\stu.py'
Number of lines: 3
```

To display the number of lines when you run your program, you need to call the function and print its result. For example:

```python
def count_lines_in_file(filename):
    """
    Reads a .txt file and returns the number of lines
    """
    with open(filename, "r") as file:
        lines = file.readlines()
        return len(lines)

# Example usage
filename = "example.txt"  # Replace with your actual
print(f"Number of lines: {count_lines_in_file(filenam
```

1 vulnerability >

Make sure:

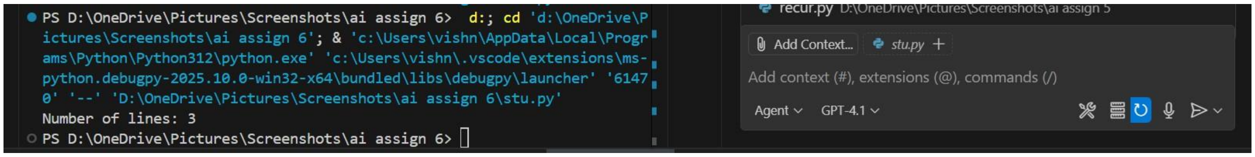
- The file (e.g., `example.txt`) exists in your project folder.
- You use the correct filename in the code.

If you want I can help you add this code to your file

3 files changed

- weather.py D:\OneDrive\Pictures\Screenshots\ai assign 5
- stu.py D:\OneDrive\Pictures\Screenshots\ai assign 6
- recur.py D:\OneDrive\Pictures\Screenshots\ai assign 5

OUTPUT



```
● PS D:\OneDrive\Pictures\Screenshots\ai assign 6>  d:; cd 'd:\OneDrive\P
  ictures\Screenshots\ai assign 6'; & 'c:\Users\vishn\AppData\Local\Progr
  ams\Python\Python312\python.exe' 'c:\Users\vishn\.vscode\extensions\ms-
  python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '6147
  0' '--' 'D:\OneDrive\Pictures\Screenshots\ai assign 6\stu.py'
  Number of lines: 3
○ PS D:\OneDrive\Pictures\Screenshots\ai assign 6> 
```

recur.py D:\OneDrive\Pictures\Screenshots\ai assign 5
Add Context...
stu.py
Add context (#), extensions (@), commands (/)
Agent
GPT-4.1

**OBSERVATION**

- The function count_lines_in_file reads a text file and returns the number of lines, using readlines() and len().
- The code is simple, efficient, and easy to understand.
- The example usage prints the number of lines in the specified file.
- If the file does not exist or the filename is incorrect,
  a FileNotFoundError will occur.

- For improved robustness, consider adding error handling (try-except) to inform the user if the file is missing or inaccessible.