ASSIGNMENT -4.3

2303A51343

B-10

TASK-1

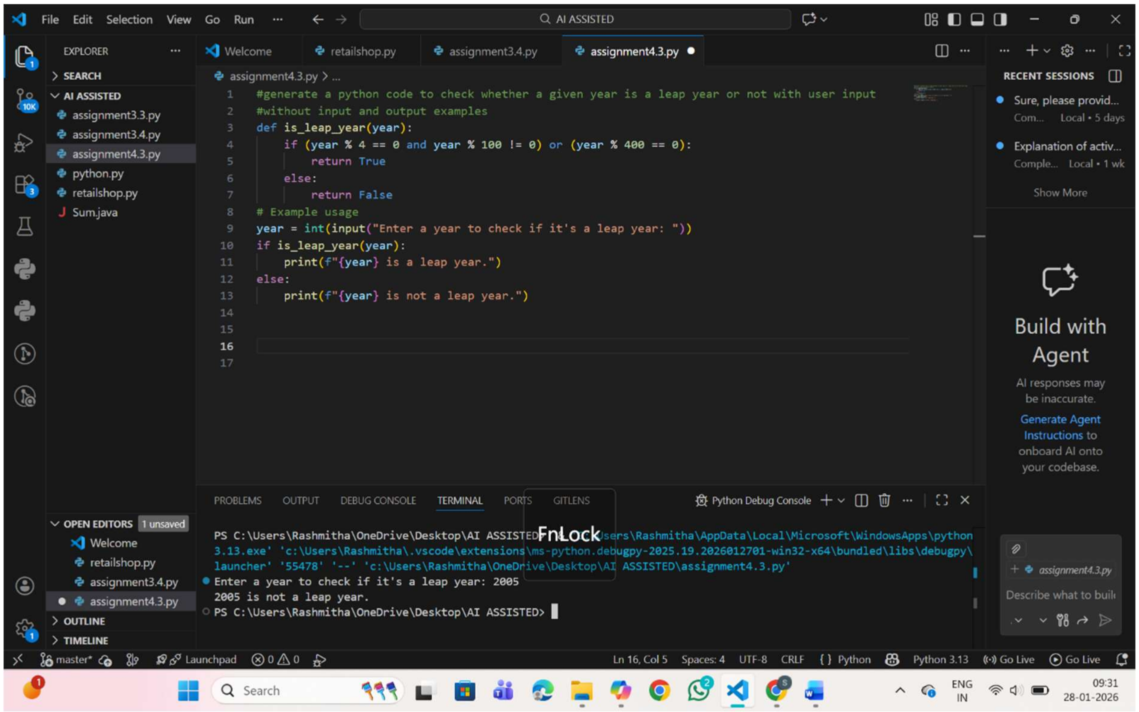
Prompt: generate a python code to check whether a given year is a leap year or not with user input

without input and output examples

code:

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
# Example usage
year = int(input("Enter a year to check if it's a leap year: "))
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

output:



```python
#generate a python code to check whether a given year is a leap year or not with user input
#without input and output examples
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
# Example usage
year = int(input("Enter a year to check if it's a leap year: "))
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTEDFnLockUsers\Rashmitha\AppData\Local\Microsoft\WindowsApps\python
3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\
launcher' '55478' '--' 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\assignment4.3.py'
Enter a year to check if it's a leap year: 2005
2005 is not a leap year.
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

Analysis: The code defines a function that applies the correct leap year rules, accepts user input, and displays whether the year is a leap year. It is straightforward, easy to understand, and logically sound.

TASK-2

Prompt:

#generate a Python code that:

Converts centimeters to inches using the correct formula (1 cm = 0.3937 inches)

#Example: Input: 10 cm → Output: 3.94 inches

Code:

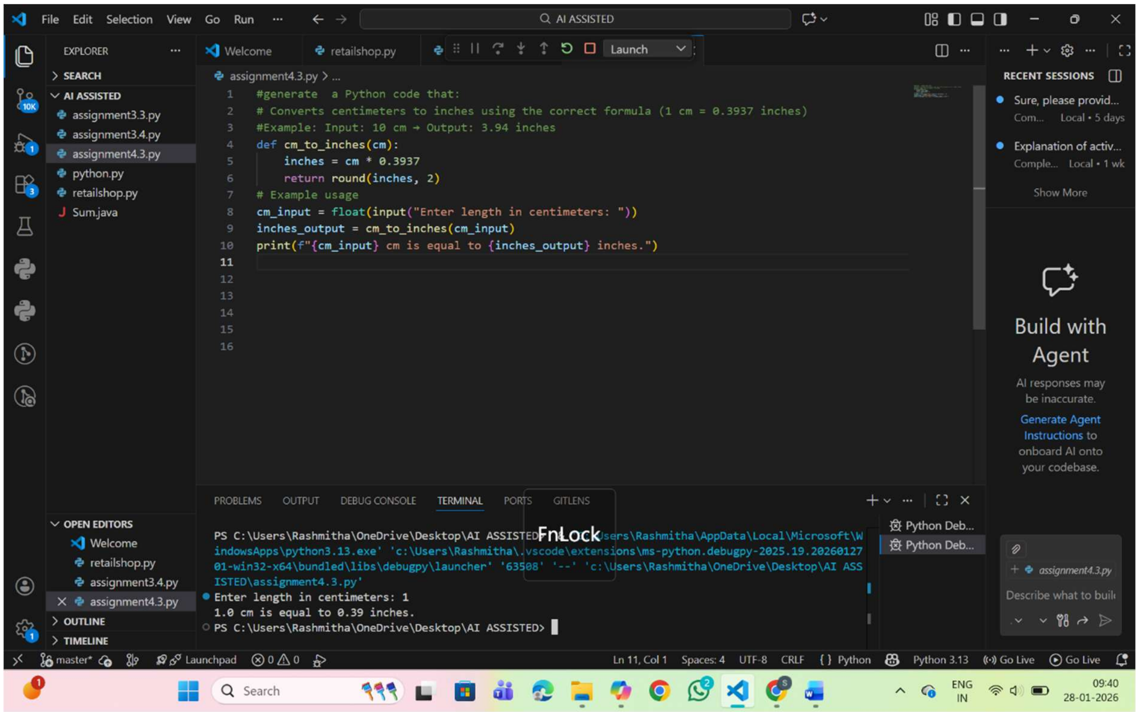
```python
def cm_to_inches(cm):
    inches = cm * 0.3937
    return round(inches, 2)
# Example usage
```

cm_input = float(input("Enter length in centimeters: "))

inches_output = cm_to_inches(cm_input)

print(f"{cm_input} cm is equal to {inches_output} inches.")

output:



```python
#generate  a Python code that:
# Converts centimeters to inches using the correct formula (1 cm = 0.3937 inches)
#Example: Input: 10 cm → Output: 3.94 inches
def cm_to_inches(cm):
    inches = cm * 0.3937
    return round(inches, 2)
# Example usage
cm_input = float(input("Enter length in centimeters: "))
inches_output = cm_to_inches(cm_input)
print(f"{cm_input} cm is equal to {inches_output} inches.")
```

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTEDFnLock Users\Rashmitha\AppData\Local\Microsoft\W
indowsApps\python3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.20260127
01-win32-x64\bundled\libs\debugpy\launcher' '63508' '--' 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASS
ISTED\assignment4.3.py'
Enter length in centimeters: 1
1.0 cm is equal to 0.39 inches.
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

Analysis: This program converts centimeters to inches using the formula (1 \text{ cm} = 0.3937 \text{ inches}).
It defines a function that multiplies the input by 0.3937 and rounds to two decimals.
The user enters a centimeter value, which is passed to the function.
Finally, the result is printed clearly showing both input and output.

TASK-3

Prompt: #generate a Python code that:

# Accepts a full name as input

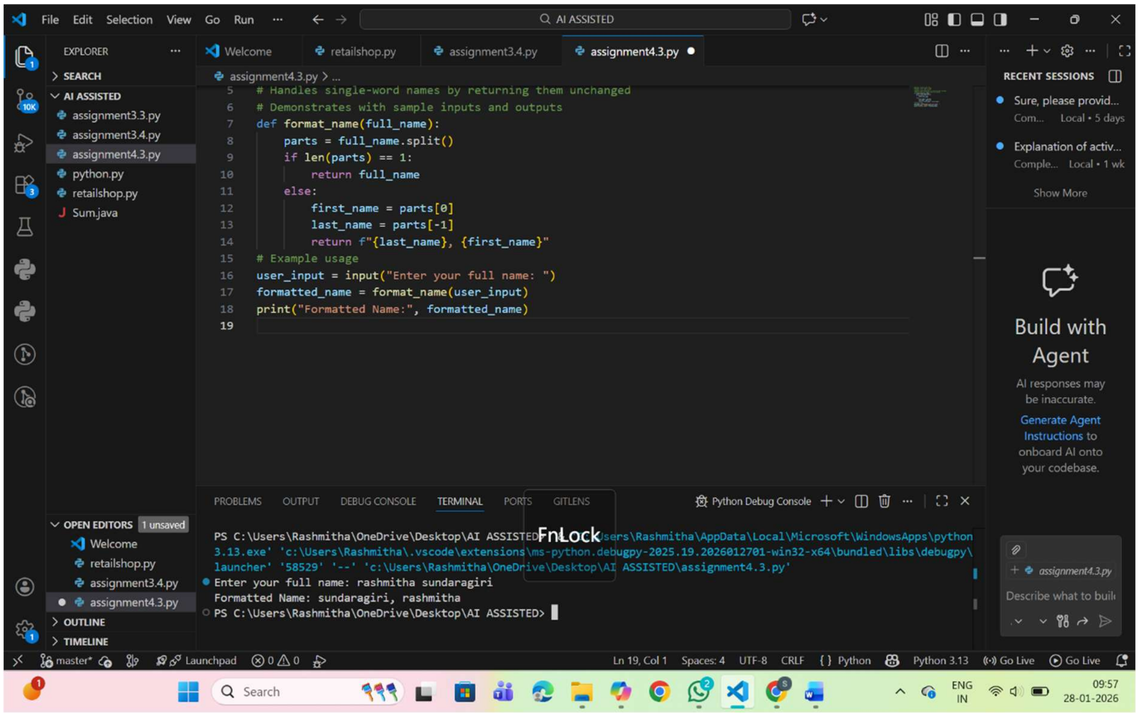
# Formats it as "Last, First"

# Handles single-word names by returning them unchanged

# Demonstrates with sample inputs and outputs

Code:
```
def format_name(full_name):
    parts = full_name.split()
    if len(parts) == 1:
        return full_name
    else:
        first_name = parts[0]
        last_name = parts[-1]
        return f"{last_name}, {first_name}"
# Example usage
user_input = input("Enter your full name: ")
formatted_name = format_name(user_input)
print("Formatted Name:", formatted_name)
```

output:



```
# Handles single-word names by returning them unchanged
# Demonstrates with sample inputs and outputs
def format_name(full_name):
    parts = full_name.split()
    if len(parts) == 1:
        return full_name
    else:
        first_name = parts[0]
        last_name = parts[-1]
        return f"{last_name}, {first_name}"
# Example usage
user_input = input("Enter your full name: ")
formatted_name = format_name(user_input)
print("Formatted Name:", formatted_name)

```

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED          sers\Rashmitha\AppData\Local\Microsoft\WindowsApps\python
3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\
launcher' '58529' '--' 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\assignment4.3.py'
Enter your full name: rashmitha sundaragiri
Formatted Name: sundaragiri, rashmitha
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

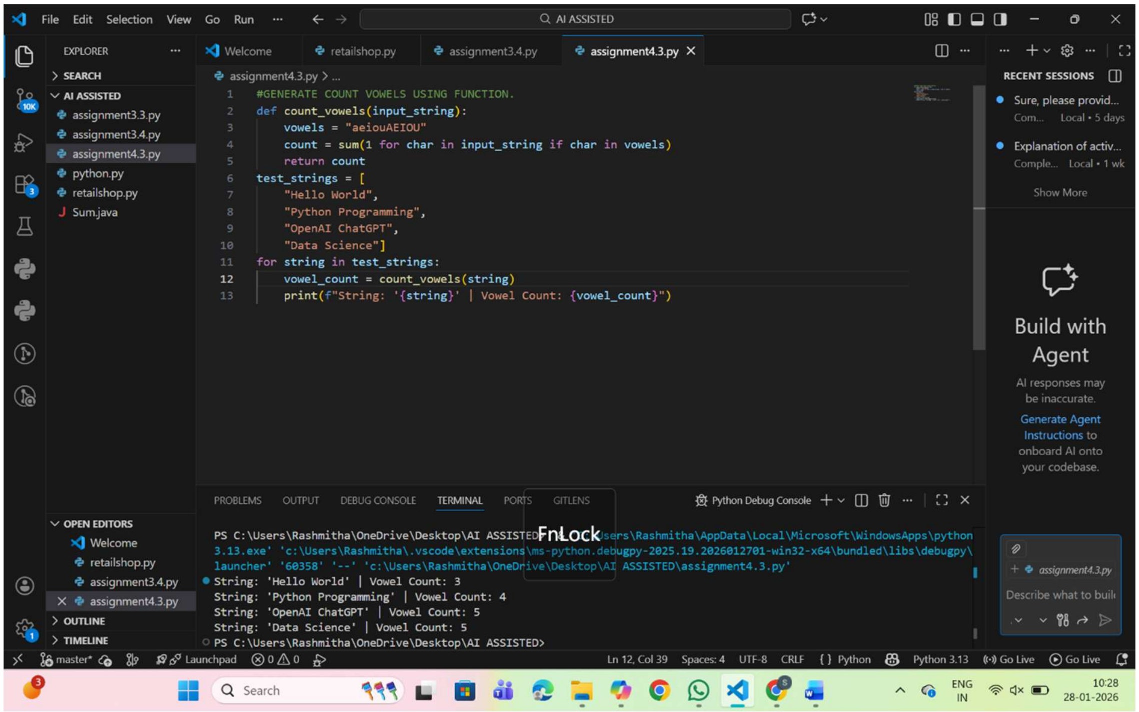
Analysis: The code correctly formats names as “Last, First” and returns single-word names unchanged. It is simple, efficient, and demonstrates usage clearly, though it does not handle middle or compound names. Overall, it fulfills the assignment requirements effectively.

TASK-4:

Prompt: GENERATE COUNT VOWELS USING FUNCTION.

Code:
```
def count_vowels(input_string):
    vowels = "aeiouAEIOU"
    count = sum(1 for char in input_string if char in vowels)
    return count
test_strings = [
    "Hello World",
    "Python Programming",
    "OpenAI ChatGPT",
    "Data Science"]
for string in test_strings:
    vowel_count = count_vowels(string)
    print(f"String: '{string}' | Vowel Count: {vowel_count}")
```

output:



```
#GENERATE COUNT VOWELS USING FUNCTION.
def count_vowels(input_string):
    vowels = "aeiouAEIOU"
    count = sum(1 for char in input_string if char in vowels)
    return count
test_strings = [
    "Hello World",
    "Python Programming",
    "OpenAI ChatGPT",
    "Data Science"]
for string in test_strings:
    vowel_count = count_vowels(string)
    print(f"String: '{string}' | Vowel Count: {vowel_count}")
```

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED [illegible] sers\Rashmitha\AppData\Local\Microsoft\WindowsApps\python
3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\
launcher' '60358' '--' 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\assignment4.3.py'
String: 'Hello World' | Vowel Count: 3
String: 'Python Programming' | Vowel Count: 4
String: 'OpenAI ChatGPT' | Vowel Count: 5
String: 'Data Science' | Vowel Count: 5
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

analysis: The function count_vowels checks each character in a string and counts if it is a vowel. It uses a generator expression with sum() for concise and efficient calculation. The program then tests multiple strings and prints each with its corresponding vowel count.

TASK-5

Prompt: #generate a python program with function that reads the file and

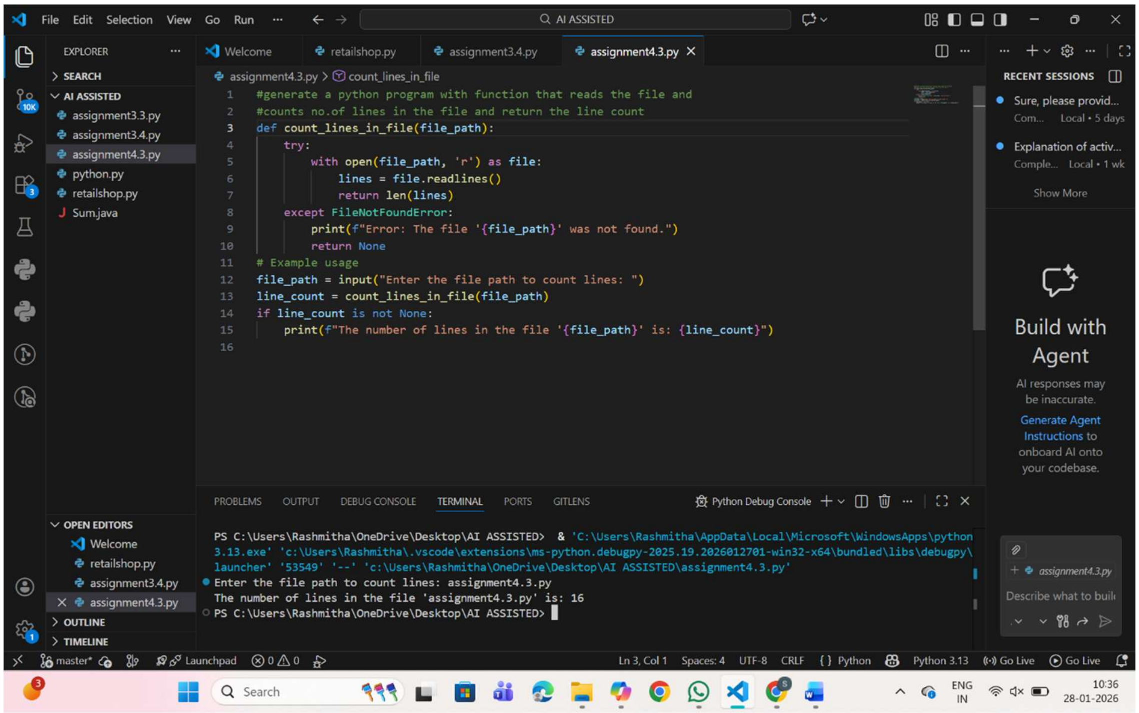
#counts no.of lines in the file and return the line count

Code:

```
def count_lines_in_file(file_path):
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
            return len(lines)
    except FileNotFoundError:
        print(f"Error: The file '{file_path}' was not found.")
        return None
# Example usage
file_path = input("Enter the file path to count lines: ")
line_count = count_lines_in_file(file_path)
```

```
if line_count is not None:
    print(f"The number of lines in the file '{file_path}' is: {line_count}")
```

output:



```
#generate a python program with function that reads the file and
#counts no.of lines in the file and return the line count
def count_lines_in_file(file_path):
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
            return len(lines)
    except FileNotFoundError:
        print(f"Error: The file '{file_path}' was not found.")
        return None
# Example usage
file_path = input("Enter the file path to count lines: ")
line_count = count_lines_in_file(file_path)
if line_count is not None:
    print(f"The number of lines in the file '{file_path}' is: {line_count}")
```

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>  & 'C:\Users\Rashmitha\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '53549' '--' 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\assignment4.3.py'
Enter the file path to count lines: assignment4.3.py
The number of lines in the file 'assignment4.3.py' is: 16
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

analysis:

This program defines a function count_lines_in_file that reads a file and returns its line count. It uses a try-except block to handle missing files gracefully by showing an error message. The function opens the file in read mode, counts lines with len(), and prints the result after the user enters a file path. Overall, it is a simple and reliable solution.