ASSIGNMENT-4.3

2303A51337

BATCH-10

❖TASK-1:

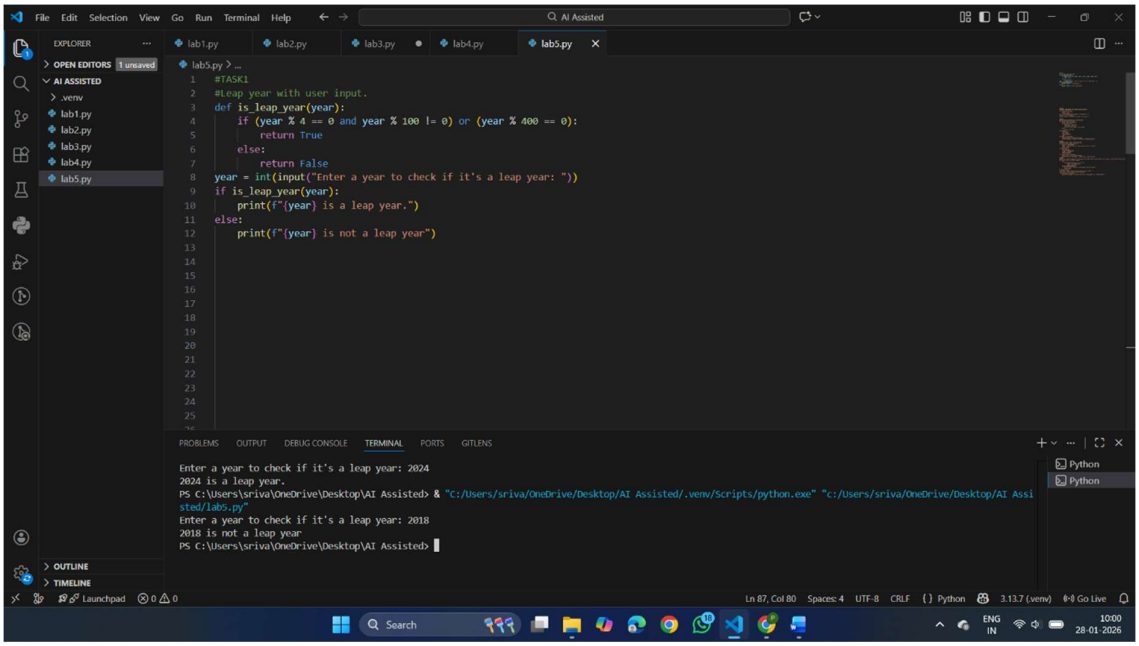
PROMPT:

#Leap year with user input.

CODE:

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
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:



EXPLANATION:

A year is a leap year if:

1. It is divisible by 4
2. BUT NOT divisible by 100,
3. UNLESS it is also divisible by 400

- year % 4 == 0 → year is divisible by 4
- year % 100 != 0 → year is not divisible by 100
- year % 400 == 0 → year is divisible by 400

If the condition is true, the function returns True (leap year).
Otherwise, it returns False (not a leap year).

❖ TASK-2

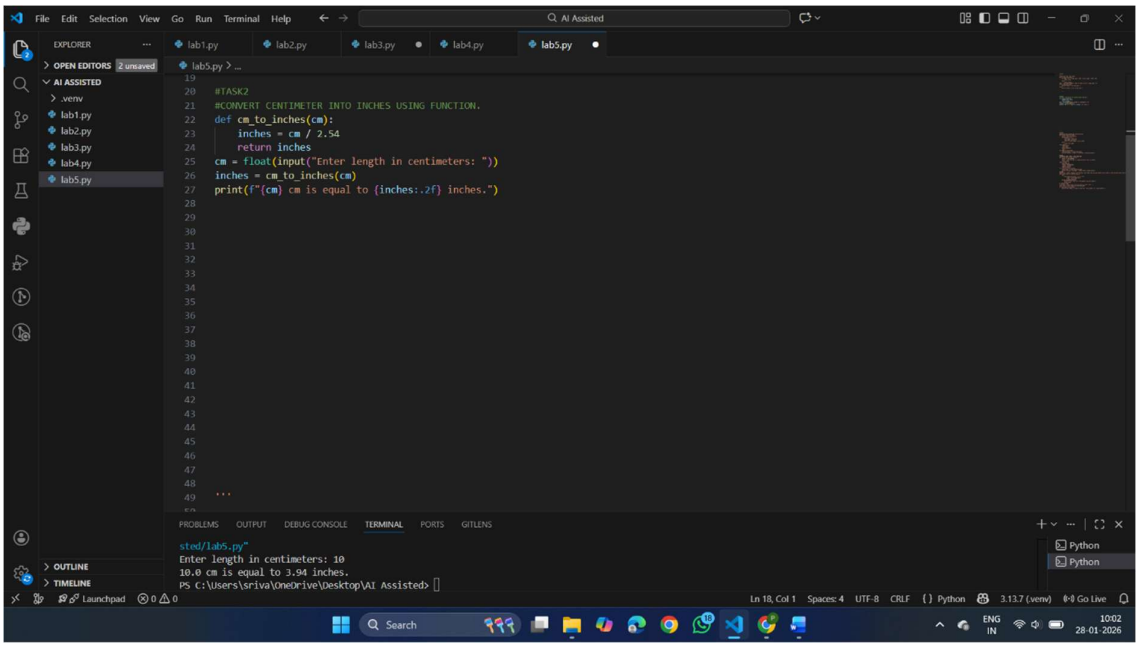
PROMPT:

#CONVERT CENTIMETER INTO INCHES USING FUNCTION.

CODE:

```
def cm_to_inches(cm):
    inches = cm / 2.54
    return inches
cm = float(input("Enter length in centimeters: "))
inches = cm_to_inches(cm)
print(f"{cm} cm is equal to {inches:.2f} inches.")
```

OUTPUT:



EXPLANATION:

This program converts a length given in centimeters into inches using a function.

The function applies the standard conversion formula, where 1 inch equals 2.54 centimeters.

The user is asked to enter a value in centimeters, which is then passed to the function.

The function performs the calculation and returns the equivalent length in inches.

Finally, the result is displayed to the user, formatted to two decimal places for better readability.

❖ TASK-3:

PROMPT:

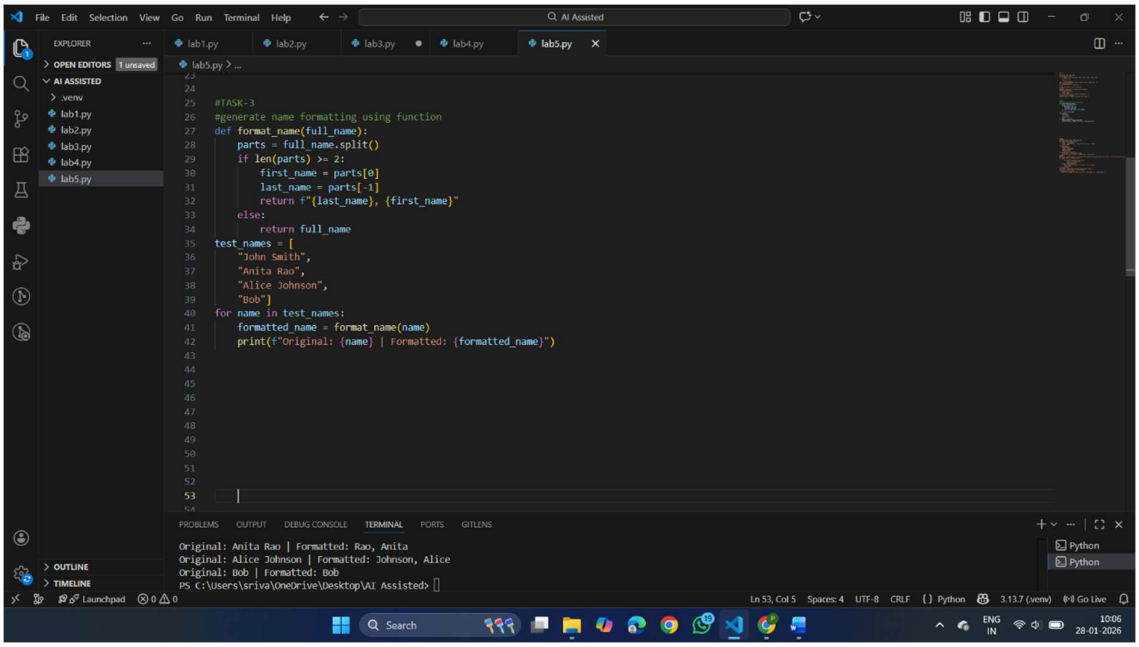
#Generate name formatting using function.

CODE:

```
def format_name(full_name):
  parts = full_name.split()
  if len(parts) >= 2:
    first_name = parts[0]
    last_name = parts[-1]
    return f"{last_name}, {first_name}"
  else:
```

```
    return full_name
test_names = [
  "John Smith",
  "Anita Rao",
  "Alice Johnson",
  "Bob"]
for name in test_names:
  formatted_name = format_name(name)
  print(f"Original: {name} | Formatted: {formatted_name}")
```

OUTPUT:



EXPLANATION:

This program formats names into a “Last name, First name” style.

It defines a function that takes a full name as input and splits it into individual parts using spaces.

If the name contains two or more words, the function treats the first word as the first name and the last word as the last name, then rearranges them into the required format.

If the name has only one word, it is returned unchanged.

A list of sample names is used to test the function. Each name in the list is processed, and the program prints both the original name and the formatted version.

❖ TASK-4:

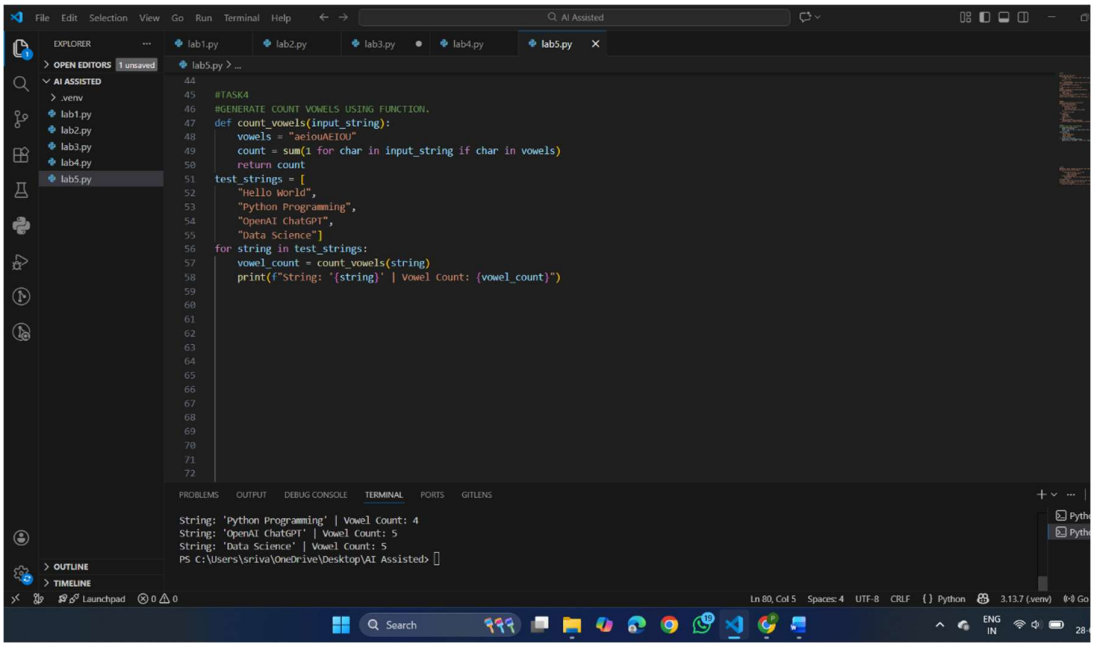
PROMPT:

#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:



EXPLANATION:

The program counts how many vowels (a, e, i, o, u) appear in each string (both uppercase and lowercase) and prints the result for every test string.

❖ TASK-5:

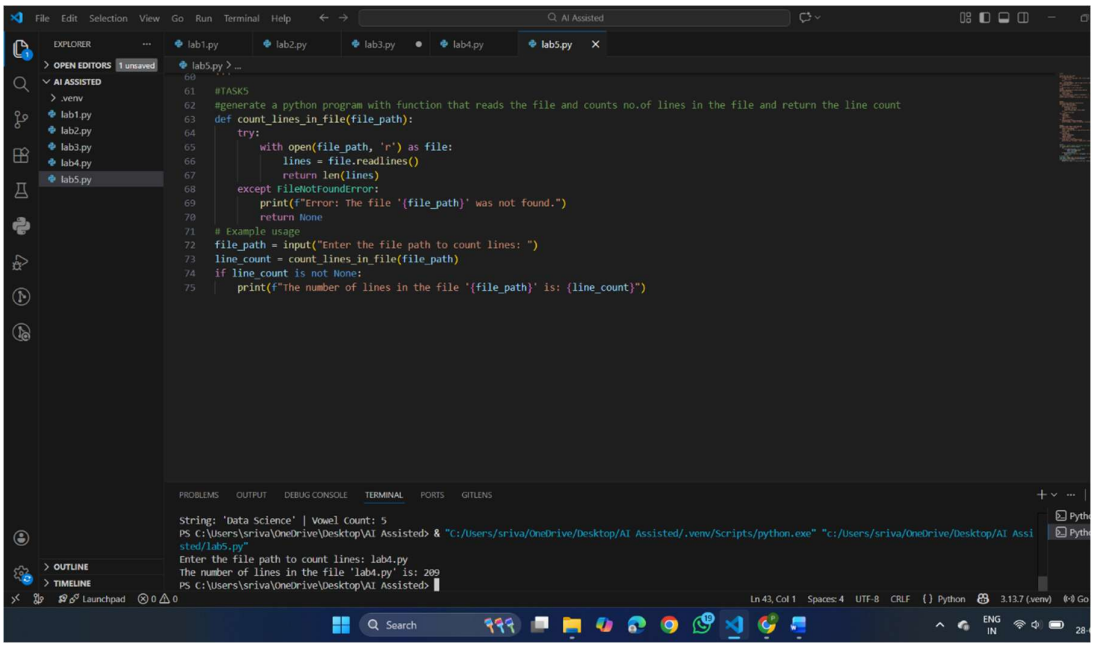
PROMPT:

#generate a python program with function that reads the file and counts no.of lines in the file and return the line count.

CODE:

```
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}")
```

OUTPUT:



EXPLANATION:

This defines a function that reads a text file and counts the number of lines in it.

The function accepts a file path as input and attempts to open the file in readmode.

If the file is successfully opened, it reads all the lines and returns the total number of lines present in the file.

The user is prompted to enter the file path, which is passed to the function. If a valid line count is returned, the program prints the total number of lines in the file.