ASSIGNMENT 4

TASK 1

PROMPT:

Write a python program to check whether the given year is leap or not .

CODE:

```python

#write a python program to check a year if the given year is leap or not

year = int(input("Enter a year: "))

# A leap year is divisible by 4, but not by 100 unless also divisible by 400
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

OUTPUT:

```
ocal\Programs\Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/as
s4.1.py"
Enter a year: 2024
2024 is a leap year.
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> & C:\Users\sahit\AppData\L
ocal\Programs\Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/as
s4.1.py"
Enter a year: 2025
2025 is not a leap year.
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:

This Python program determines whether a given year is a leap year. It starts by prompting the user to enter a year, using the input() function. The input, which is initially a string, is converted to an integer using int(), ensuring that mathematical operations can be performed on it.

The core logic checks the leap year rules: a year is a leap year if it is divisible by 4, except for years that are divisible by 100—unless they are also divisible by 400. This is implemented using a conditional statement. The expression (year % 4 == 0 and year % 100 != 0) checks if the year is divisible by 4 but not by 100. The second part, (year % 400 == 0), accounts for years like 2000, which are divisible by 400 and are leap years even though they are also divisible by 100.

Depending on the result of this condition, the program prints whether the entered year is a leap year or not. This approach follows the official leap year rules used in the Gregorian calendar and is both concise and accurate.

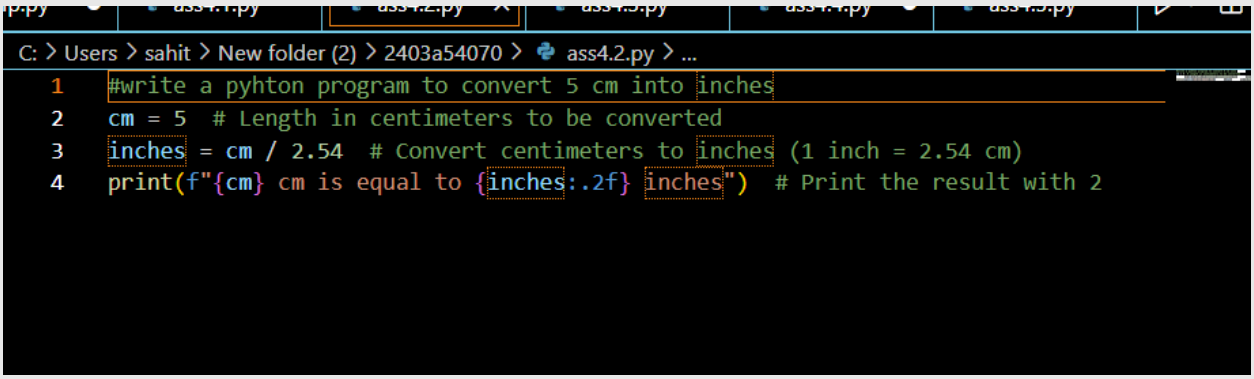
TASK 2

PROMPT:

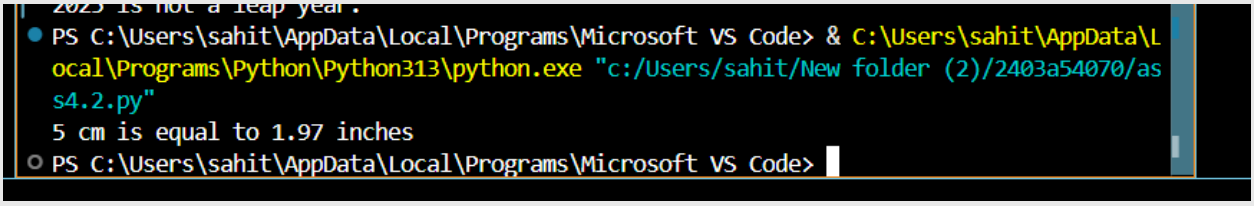
Write a python program to convert 5 cm to inches

CODE:

:



OUTPUT:



EXPLANATION:

This Python code snippet converts a length from centimeters to inches. It starts by assigning the value 5 to the variable cm, representing the length in centimeters that needs to be converted. The conversion is performed by dividing cm by 2.54, since one inch is exactly 2.54 centimeters. The result of this calculation is stored in the variable inches.

To display the result, the code uses a formatted string with the print() function. The expression {inches:.2f} ensures that the output shows the number of inches rounded to two decimal places, making the result more readable and precise. The final output message clearly states the original value in centimeters and its equivalent in inches. This approach is straightforward and demonstrates a common unit conversion technique in Python.

TASK 3

PROMPT:

Write a python program to generate a function that formats full into first name and last name .

CODE:

```python
def format_full_name(full_name):
    # Split the full name into parts
    parts = full_name.strip().split()
    if len(parts) >= 2:
        first_name = parts[0]
        last_name = parts[-1]
        return f"First Name: {first_name}, Last Name: {last_name}"
    else:
        return "Please enter both first and last name."

# Example usages:
print(format_full_name("John Doe"))
print(format_full_name("Alice Smith"))
print(format_full_name("Michael Andrew Johnson"))
print(format_full_name("Priya"))
```

OUTPUT:

```
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> & C:\Users\sahit\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/ass4.3.py"
First Name: John, Last Name: Doe
First Name: Alice, Last Name: Smith
First Name: Michael, Last Name: Johnson
Please enter both first and last name.
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:

This code defines a function to format a person's full name into first and last name components. The function, format_full_name, takes a string argument full_name. It first removes any leading or trailing whitespace using strip(), then splits the string into a list of words using split(). If the resulting list has two or more elements, the function assumes the first element is the first name and the last element is the last name, returning them in a formatted string. If the input does not contain at least two words, the function prompts the user to enter both a first and last name.

After defining the function, the code prompts the user to enter their full name using input(). However, in the provided selection, the function is defined and user input is collected, but the function is not yet called with the user's input. This code structure is useful for extracting and formatting names from user input, but to see the output, you would need to call format_full_name(user_input) and print the result.

TASK 4

PROMPT:

Write a python program that counts the number of vowels in the given string.

CODE:

```python
#write a python program that counts the number of vowels in a given string

def count_vowels(s):
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count

# Take input from the user
user_input = input("Enter a string: ")
print(f"Number of vowels in the string: {count_vowels(user_input)}")
```

OUTPUT:

```
Enter a string: hello sr university
Number of vowels in the string: 6
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:

> Used 5 references

This Python program counts the number of vowels in a user-provided string. It defines a function called count_vowels that takes a string s as input. Inside the function, a string vowels is defined containing all lowercase and uppercase vowel characters ("aeiouAEIOU"). The function initializes a counter variable, count, to zero. It then iterates through each character in the input string s. For every character that is found in the vowels string, the counter is incremented by one. After checking all characters, the function returns the total count of vowels.

The program then prompts the user to enter a string using input(). The entered string is passed to the count_vowels function, and the result is printed in a formatted message. This approach efficiently counts both uppercase and lowercase vowels and provides immediate feedback to the user. The logic is straightforward and easy to understand, making it a good example of string processing and function usage in Python.