SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

ProgramName:B. Tech	**Assignment Type: Lab**	**AcademicYear:**2025-2026

CourseCoordinatorName	Venkataramana Veeramsetty
Instructor(s)Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)

CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week1 - Thursday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	24CSBTB01 To 24CSBTB39

AssignmentNumber:2.4(Present assignment number)/**24**(Total number of assignments)

Q.No.	**Question**	***ExpectedTime to complete***
1		Week1

Lab 2: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI

- Thursday

Lab Objectives:

- To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab.
- To understand and use Cursor AI for code generation, explanation, and refactoring.
- To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI.
- To perform code optimization and documentation using AI tools.

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- Generate Python code using Google Gemini in Google Colab.
- Analyze the effectiveness of code explanations and suggestions by Gemini.
- Set up and use Cursor AI for AI-powered coding assistance.
- Evaluate and refactor code using Cursor AI features.
- Compare AI tool behavior and code quality across different platforms.

Task Description #1

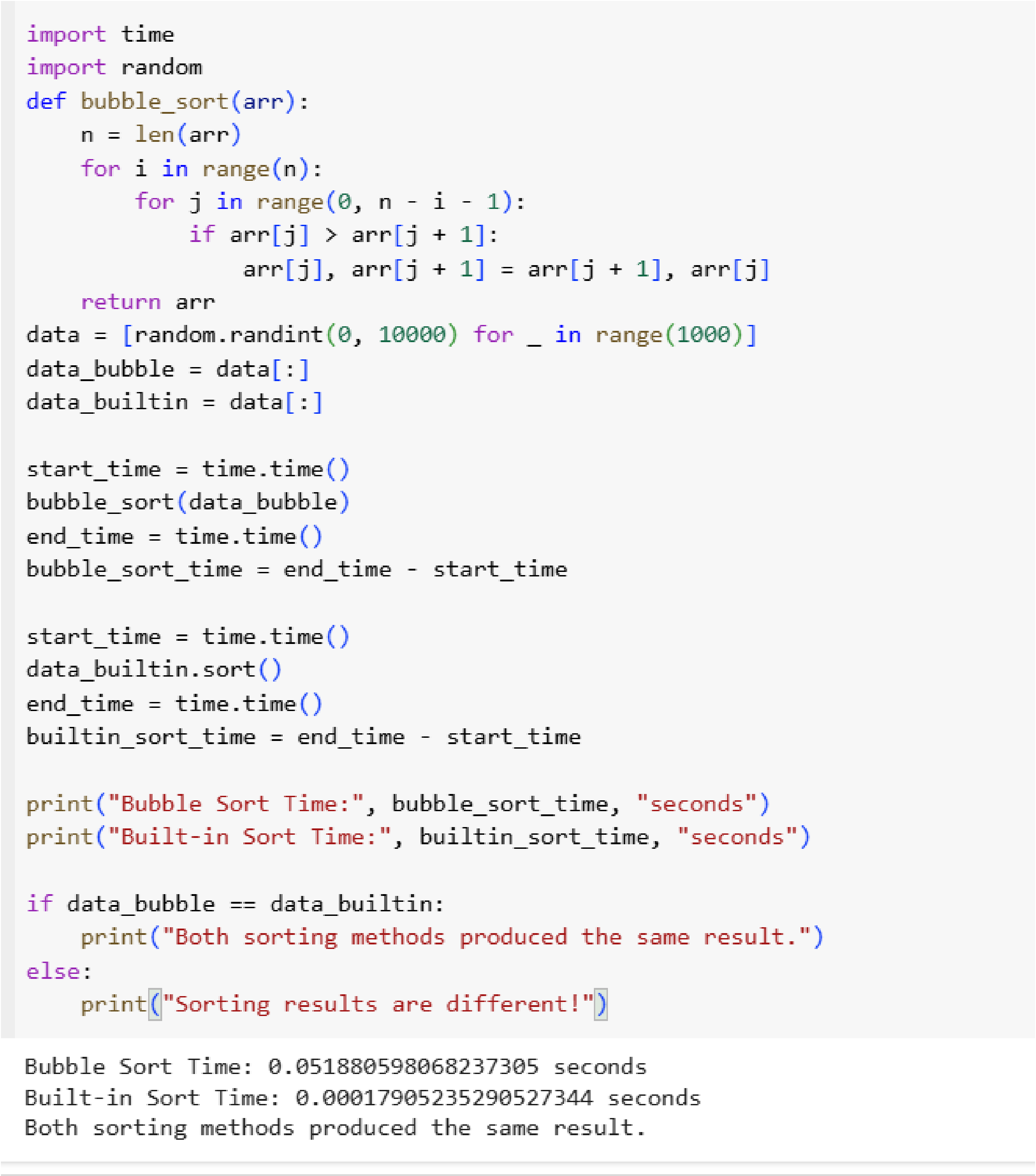
• Open Google Colab and use Google Gemini to generate Python code that performs sorting of a list using both the bubble sort algorithm and Python’s built-in sort() function. Compare the two implementations.

Expected Output #1

• Two sorting implementations: Bubble sort (manual logic) and Built-in sort()

PROMPT: Generate a python code that performs sorting of a list using both the bubble sort algorithm and python’s Python’s built-in sort() function. Compare the two implementations.

CODE:



```
import time
import random
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
data = [random.randint(0, 10000) for _ in range(1000)]
data_bubble = data[:]
data_builtin = data[:]

start_time = time.time()
bubble_sort(data_bubble)
end_time = time.time()
bubble_sort_time = end_time - start_time

start_time = time.time()
data_builtin.sort()
end_time = time.time()
builtin_sort_time = end_time - start_time

print("Bubble Sort Time:", bubble_sort_time, "seconds")
print("Built-in Sort Time:", builtin_sort_time, "seconds")

if data_bubble == data_builtin:
    print("Both sorting methods produced the same result.")
else:
    print("Sorting results are different!")
```

```
Bubble Sort Time: 0.051880598068237305 seconds
Built-in Sort Time: 0.00017905235290527344 seconds
Both sorting methods produced the same result.
```

OBSERVATION: The built-in sorting in Python is much, much faster than the bubble sort code. Python's built-in sort is designed to be very efficient, while bubble sort is a simple way to sort that isn't the quickest. Even though one is faster, both methods correctly sort the list to be the same as the original list, just in order. The code sets up the same random numbers for both sorts so you can see a fair comparison of their speeds.

Task Description #2

- In Colab, use Google Gemini to generate a Python function that takes a string and returns:

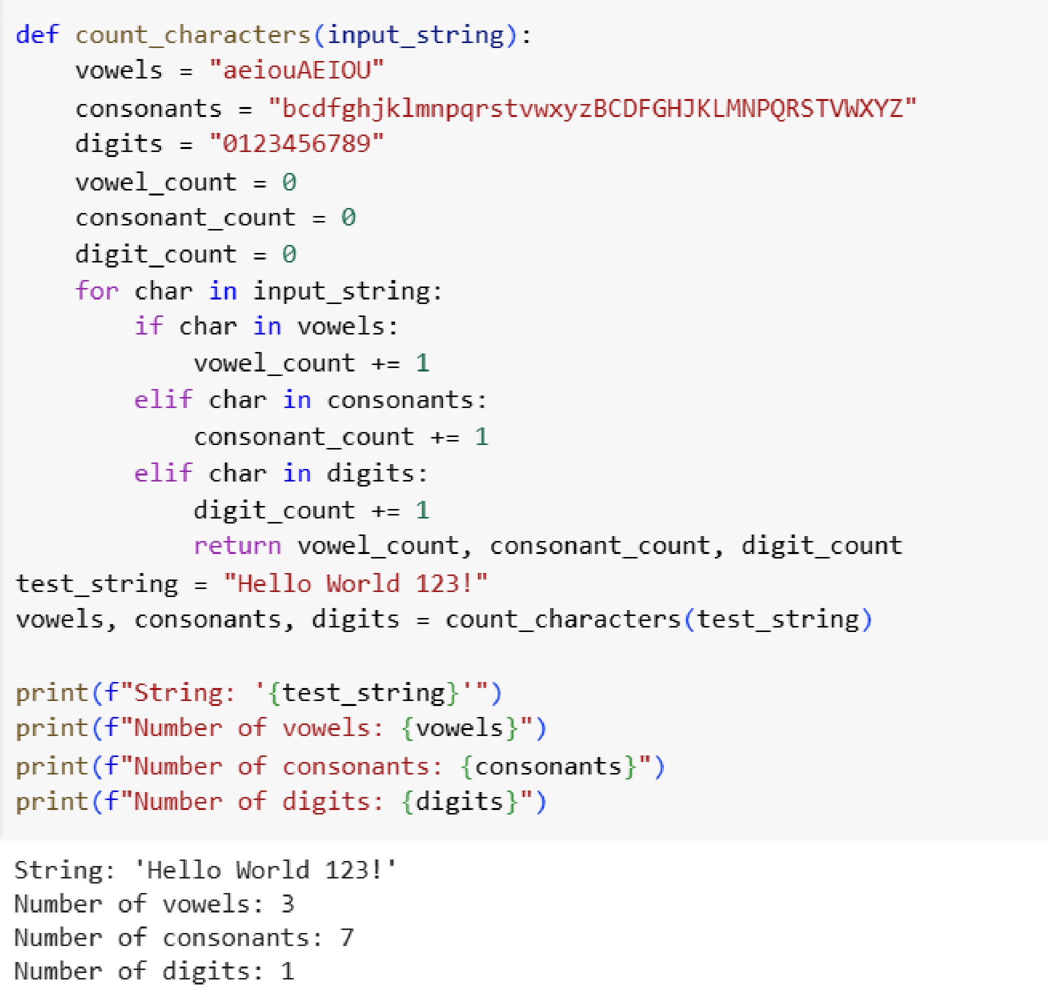
The number of vowels, The number of consonants, The number of digits in the string

Expected Output #2-

- Complete function that Iterates through characters of a string and Counts vowels, consonants, and digits

PROMPT: Generate a Python code using function that takes a string and returns: The number of vowels, The number of consonants and The number of digits in the string.

CODE:



```python
def count_characters(input_string):
    vowels = "aeiouAEIOU"
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
    digits = "0123456789"
    vowel_count = 0
    consonant_count = 0
    digit_count = 0
    for char in input_string:
        if char in vowels:
            vowel_count += 1
        elif char in consonants:
            consonant_count += 1
        elif char in digits:
            digit_count += 1
            return vowel_count, consonant_count, digit_count
test_string = "Hello World 123!"
vowels, consonants, digits = count_characters(test_string)

print(f"String: '{test_string}'")
print(f"Number of vowels: {vowels}")
print(f"Number of consonants: {consonants}")
print(f"Number of digits: {digits}")
```

```
String: 'Hello World 123!'
Number of vowels: 3
Number of consonants: 7
Number of digits: 1
```

OBSERVATION: This code looks at each character in a sentence or string you give it. It counts how many letters are vowels (a, e, i, o, u), how many are consonants (other letters), and how many are numbers (0-9). It then shows you the total count for each of those categories. You can easily change the test_string to count characters in a different sentence.

Task Description #3

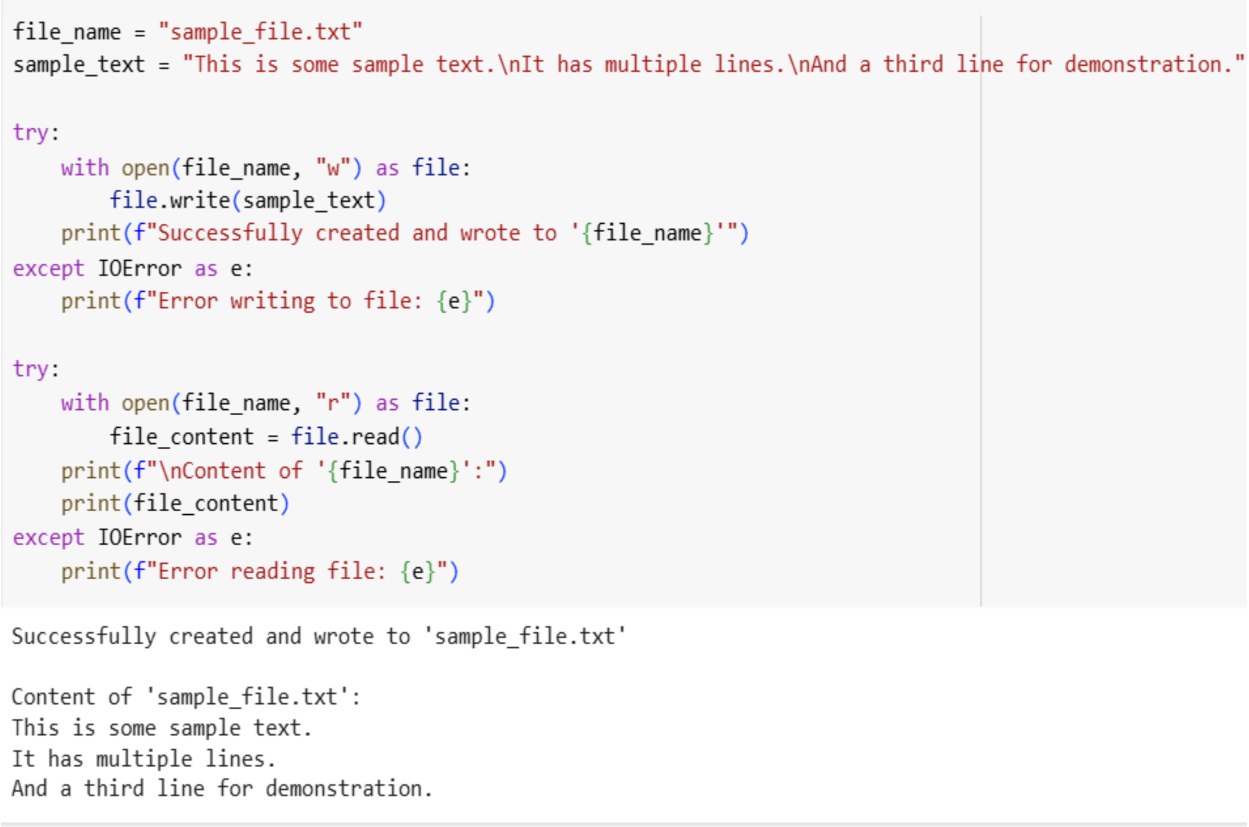
- Install and set up Cursor AI. Use it to generate a Python program that performs file handling:

Create a text file
Write sample text
Read and display the content

Expected Output #3

- Functional code that creates a .txt file, writes content to it, and reads it back.
- Screenshot of Cursor AI interface showing: Prompt used, Generated code, Output of file operations

PROMPT: Generate a Python program that performs file handling: Create a text file, Write sample text and Read and display the content..



```
file_name = "sample_file.txt"
sample_text = "This is some sample text.\nIt has multiple lines.\nAnd a third line for demonstration."

try:
    with open(file_name, "w") as file:
        file.write(sample_text)
    print(f"Successfully created and wrote to '{file_name}'")
except IOError as e:
    print(f"Error writing to file: {e}")

try:
    with open(file_name, "r") as file:
        file_content = file.read()
    print(f"\nContent of '{file_name}':")
    print(file_content)
except IOError as e:
    print(f"Error reading file: {e}")
```

```
Successfully created and wrote to 'sample_file.txt'

Content of 'sample_file.txt':
This is some sample text.
It has multiple lines.
And a third line for demonstration.
```

OBSERVATION: This program demonstrates basic file operations in Python. It starts by creating a text file named sample_file.txt. Inside this newly created file, it writes some sample text. After successfully writing the content, the program then opens the same file again, this time to read everything that was written. Finally, it prints the content it read from the file, confirming that the writing and reading processes were successful.

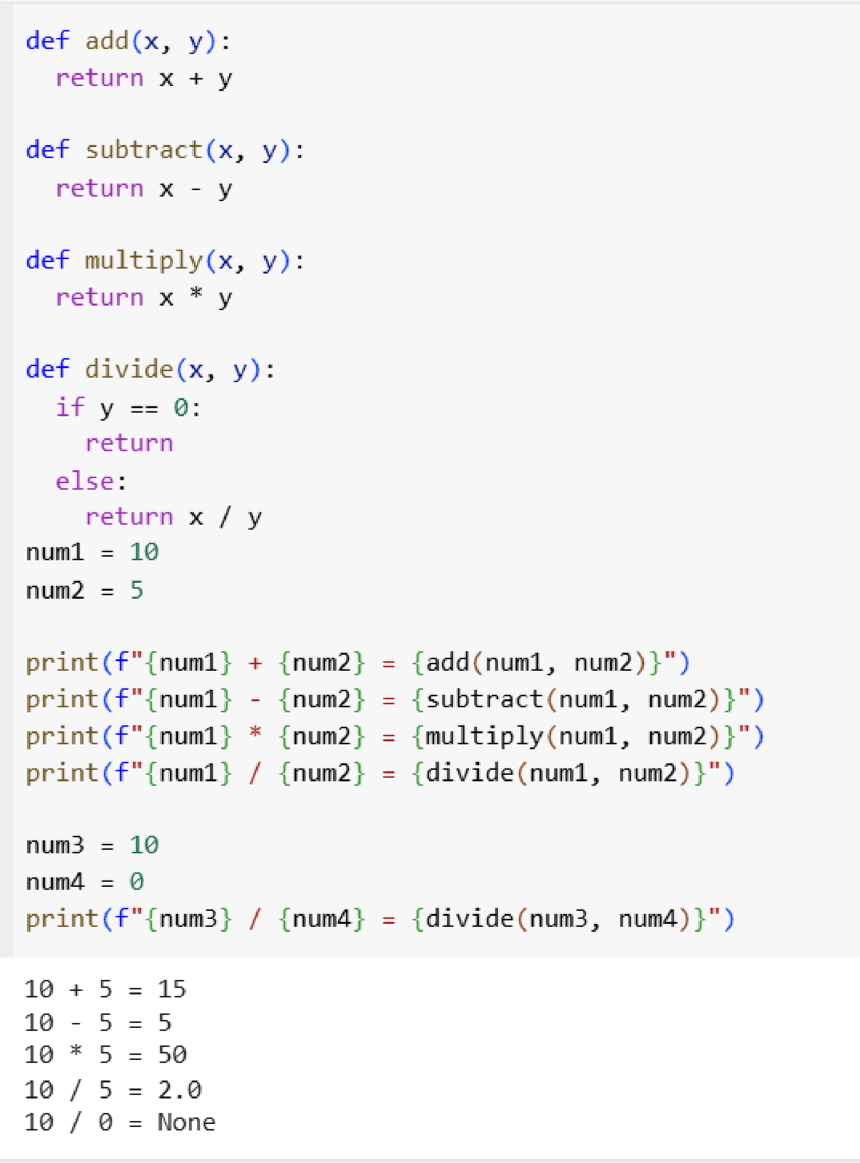
Task Description #4

- Ask Google Gemini to generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). Then, ask Gemini to explain how the code works.

Expected Output #4

- Complete calculator code with user input and operation selection.
- Line-by-line explanation or markdown-style explanation provided by Gemini.
- Screenshot of both the code and explanation in Colab.

PROMPT: Generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). Then, ask Gemini to explain how the code works.
CODE:



```python
def add(x, y):
  return x + y

def subtract(x, y):
  return x - y

def multiply(x, y):
  return x * y

def divide(x, y):
  if y == 0:
    return
  else:
    return x / y
num1 = 10
num2 = 5

print(f"{num1} + {num2} = {add(num1, num2)}")
print(f"{num1} - {num2} = {subtract(num1, num2)}")
print(f"{num1} * {num2} = {multiply(num1, num2)}")
print(f"{num1} / {num2} = {divide(num1, num2)}")

num3 = 10
num4 = 0
print(f"{num3} / {num4} = {divide(num3, num4)}")
```

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
10 / 0 = None
```

OBSERVATION: This code provides a simple calculator using functions for addition, subtraction, multiplication, and division. Each function performs its specific math operation, and the division function includes a check to avoid errors when dividing by zero. The program then shows examples of using these functions and prints the outcomes.

Task Description #5

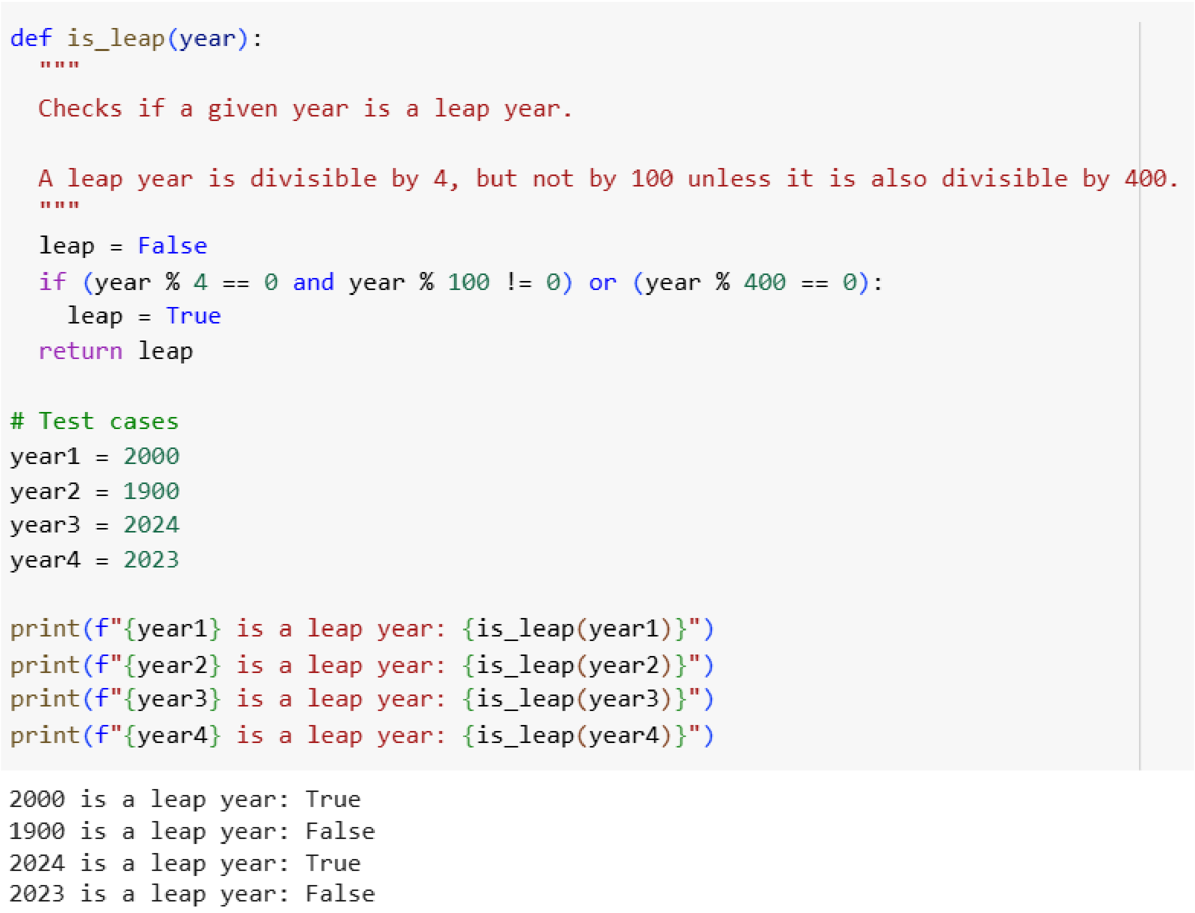
- Use Cursor AI to create a Python program that checks if a given year is a leap year or not. Try different prompt styles and see how Cursor modifies its code suggestions.

Expected Output #5

- A functional program to check leap year with sample input/output
- At least two versions of the code (from different prompts)
- A short comparison of which version is better and why

PROMPT: Python program that checks if a given year is a leap year or not. Try different prompt styles and see how Cursor modifies its code suggestions.

CODE:



```python
def is_leap(year):
  """
  Checks if a given year is a leap year.

  A leap year is divisible by 4, but not by 100 unless it is also divisible by 400.
  """
  leap = False
  if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    leap = True
  return leap

# Test cases
year1 = 2000
year2 = 1900
year3 = 2024
year4 = 2023

print(f"{year1} is a leap year: {is_leap(year1)}")
print(f"{year2} is a leap year: {is_leap(year2)}")
print(f"{year3} is a leap year: {is_leap(year3)}")
print(f"{year4} is a leap year: {is_leap(year4)}")
```

```
2000 is a leap year: True
1900 is a leap year: False
2024 is a leap year: True
2023 is a leap year: False
```

OBSERVATION: This code checks if a specific year is a leap year using a set of rules. It looks to see if a year can be evenly divided by 4. However, there's a twist: years divisible by 100 are usually not leap years, *unless* they can also be evenly divided by 400. The code puts these rules together to decide if a year is a leap year or not and then shows you the answer for some example years.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Two sorting implementations: Bubble sort (manual logic) and Built-in sort() (Task#1)	0.5
Counts vowels, consonants, and digits(Task#2)	0.5
Functional code that creates a .txt file, writes content to it, and reads it back- Use cursor (Task#3)	0.5
Complete calculator code with user input and operation selection. (Task#4)	0.5
A functional program to check leap year with sample input/output-use Cursor (Task#5)	0.5
Total	**2.5 Marks**