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)			

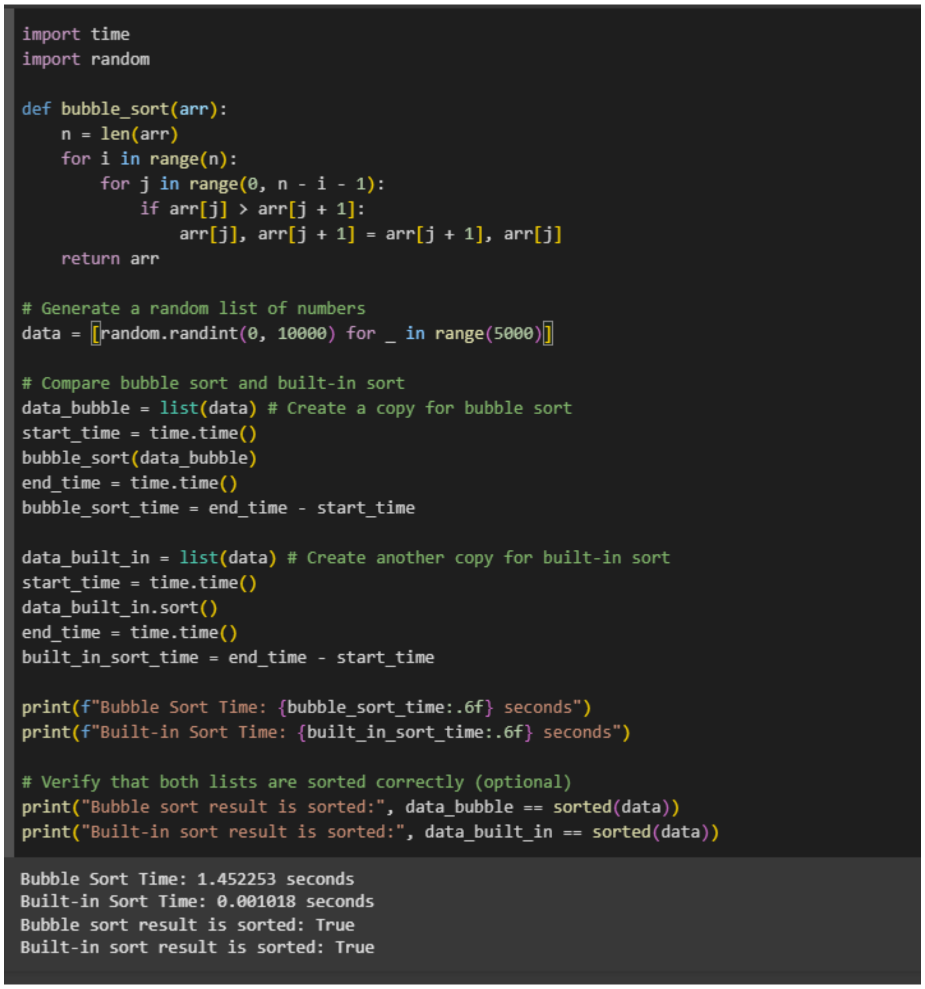
Q.No.	**Question**	***ExpectedTime to complete***
1	Lab 2: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI **Lab Objectives:**	Week1 - Thursday

- 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.

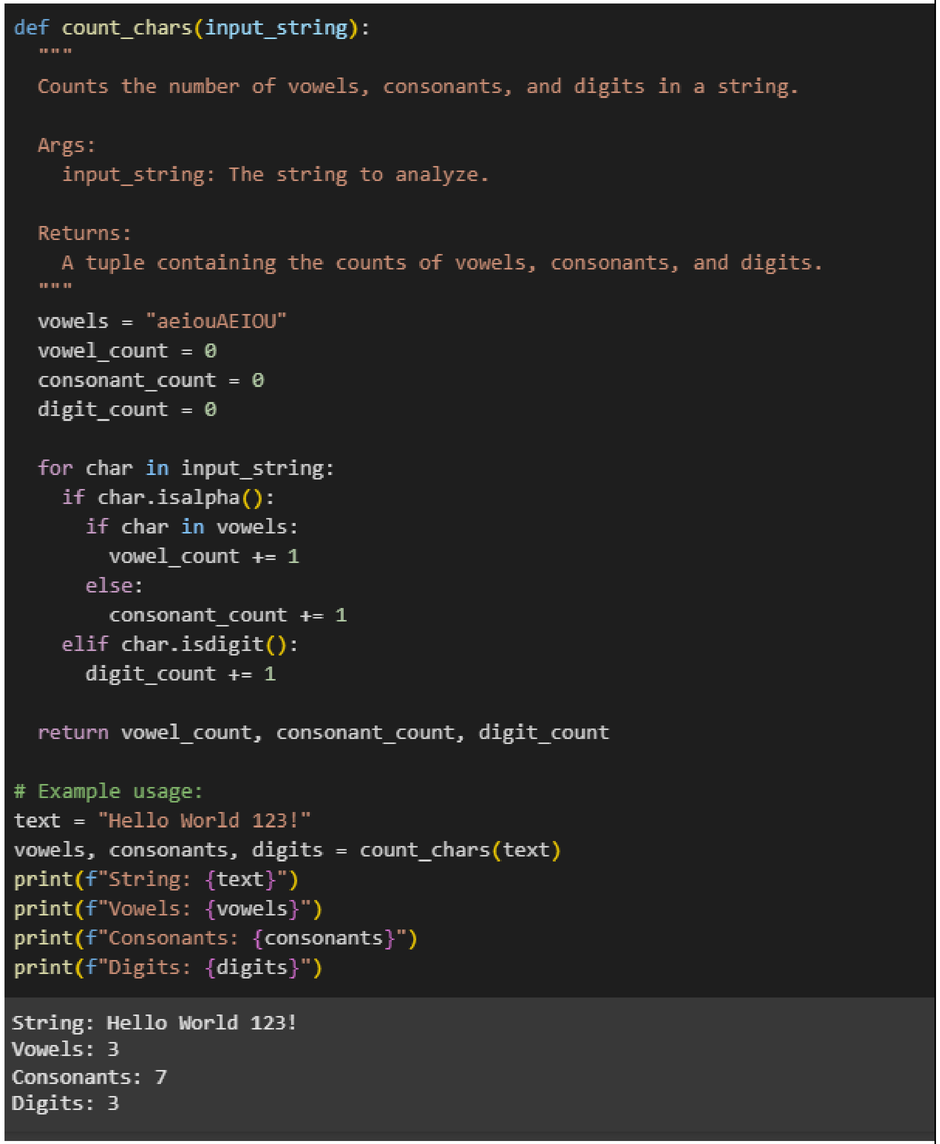
Task Description #1
• Open Google Colab and use Google Gemini to generate Python code that performs sorting of a list using both the bubble sort algorithm and Python's built-in sort() function. Compare the two implementations.
Expected Output #1
• Two sorting implementations: Bubble sort (manual logic) and Built-in sort()



```python
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

# Generate a random list of numbers
data = [random.randint(0, 10000) for _ in range(5000)]

# Compare bubble sort and built-in sort
data_bubble = list(data) # Create a copy for bubble sort
start_time = time.time()
bubble_sort(data_bubble)
end_time = time.time()
bubble_sort_time = end_time - start_time

data_built_in = list(data) # Create another copy for built-in sort
start_time = time.time()
data_built_in.sort()
end_time = time.time()
built_in_sort_time = end_time - start_time

print(f"Bubble Sort Time: {bubble_sort_time:.6f} seconds")
print(f"Built-in Sort Time: {built_in_sort_time:.6f} seconds")

# Verify that both lists are sorted correctly (optional)
print("Bubble sort result is sorted:", data_bubble == sorted(data))
print("Built-in sort result is sorted:", data_built_in == sorted(data))
```

```
Bubble Sort Time: 1.452253 seconds
Built-in Sort Time: 0.001018 seconds
Bubble sort result is sorted: True
Built-in sort result is sorted: True
```

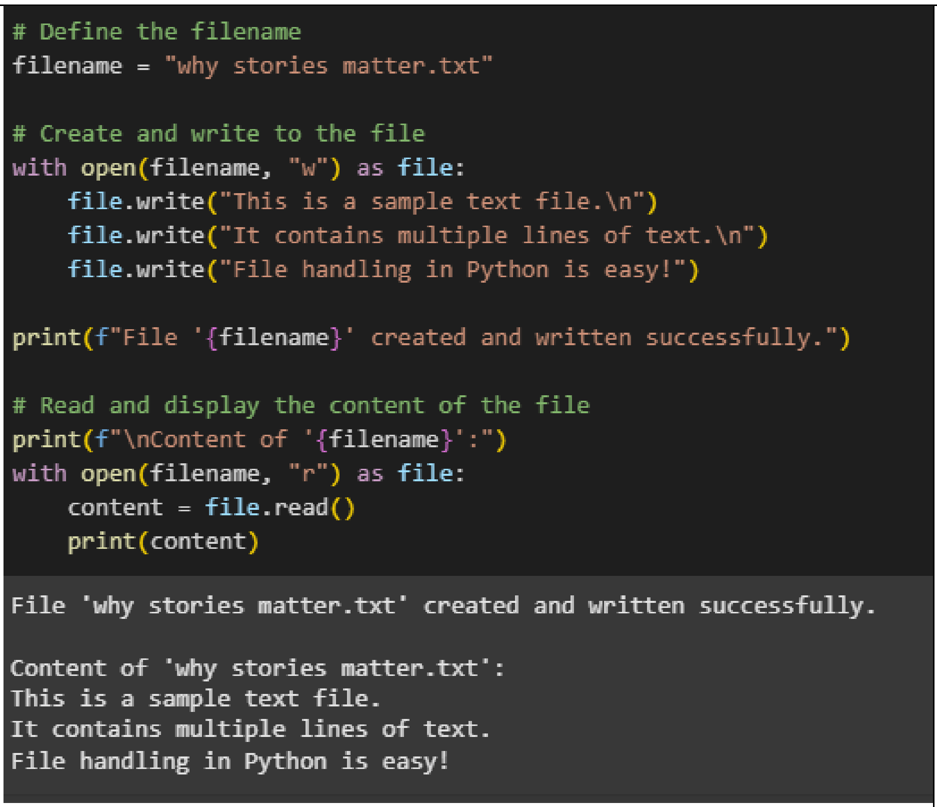
Task Description #2
• In Colab, use Google Gemini to generate a Python function that takes a string and returns: The number of vowels, The number of consonants, The number of digits in the string
Expected Output #2-
• Complete function that Iterates through characters of a string and Counts vowels, consonants, and digits



```python
def count_chars(input_string):
  """
  Counts the number of vowels, consonants, and digits in a string.

  Args:
    input_string: The string to analyze.

  Returns:
    A tuple containing the counts of vowels, consonants, and digits.
  """
  vowels = "aeiouAEIOU"
  vowel_count = 0
  consonant_count = 0
  digit_count = 0

  for char in input_string:
    if char.isalpha():
      if char in vowels:
        vowel_count += 1
      else:
        consonant_count += 1
    elif char.isdigit():
      digit_count += 1

  return vowel_count, consonant_count, digit_count

# Example usage:
text = "Hello World 123!"
vowels, consonants, digits = count_chars(text)
print(f"String: {text}")
print(f"Vowels: {vowels}")
print(f"Consonants: {consonants}")
print(f"Digits: {digits}")
```

```
String: Hello World 123!
Vowels: 3
Consonants: 7
Digits: 3
```

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



```
# Define the filename
filename = "why stories matter.txt"

# Create and write to the file
with open(filename, "w") as file:
    file.write("This is a sample text file.\n")
    file.write("It contains multiple lines of text.\n")
    file.write("File handling in Python is easy!")

print(f"File '{filename}' created and written successfully.")

# Read and display the content of the file
print(f"\nContent of '{filename}':")
with open(filename, "r") as file:
    content = file.read()
    print(content)
```

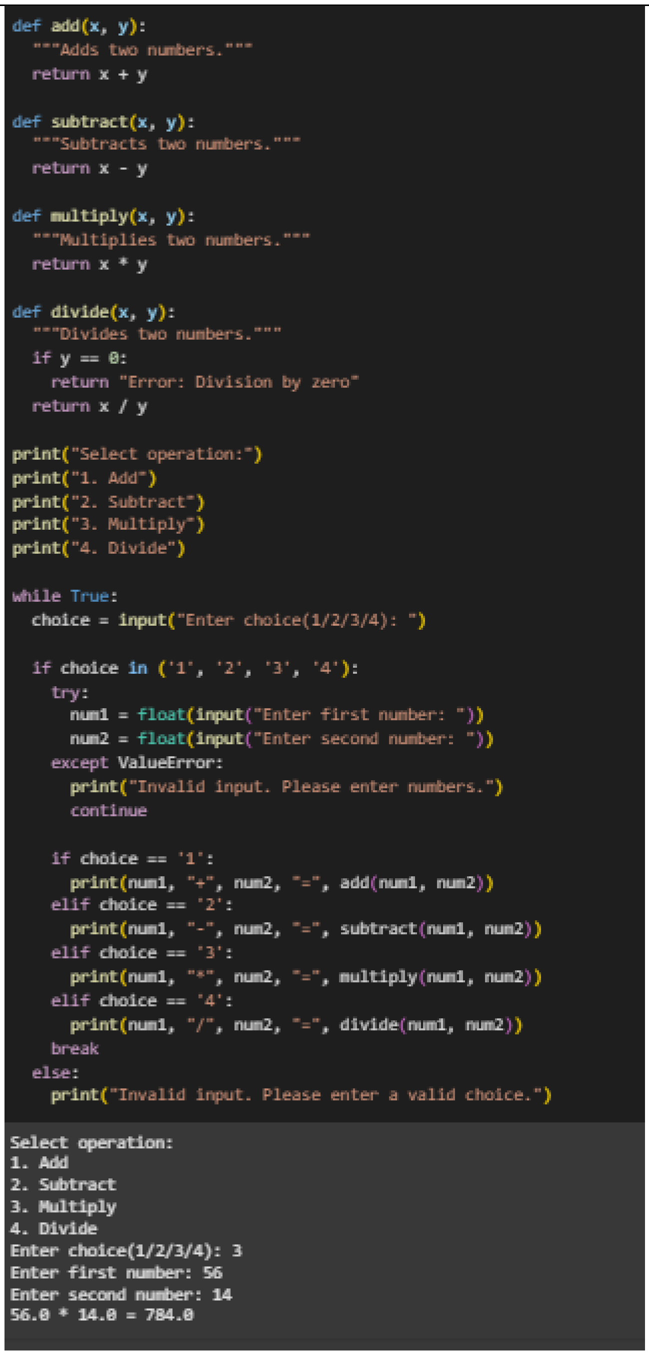
```
File 'why stories matter.txt' created and written successfully.

Content of 'why stories matter.txt':
This is a sample text file.
It contains multiple lines of text.
File handling in Python is easy!
```

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

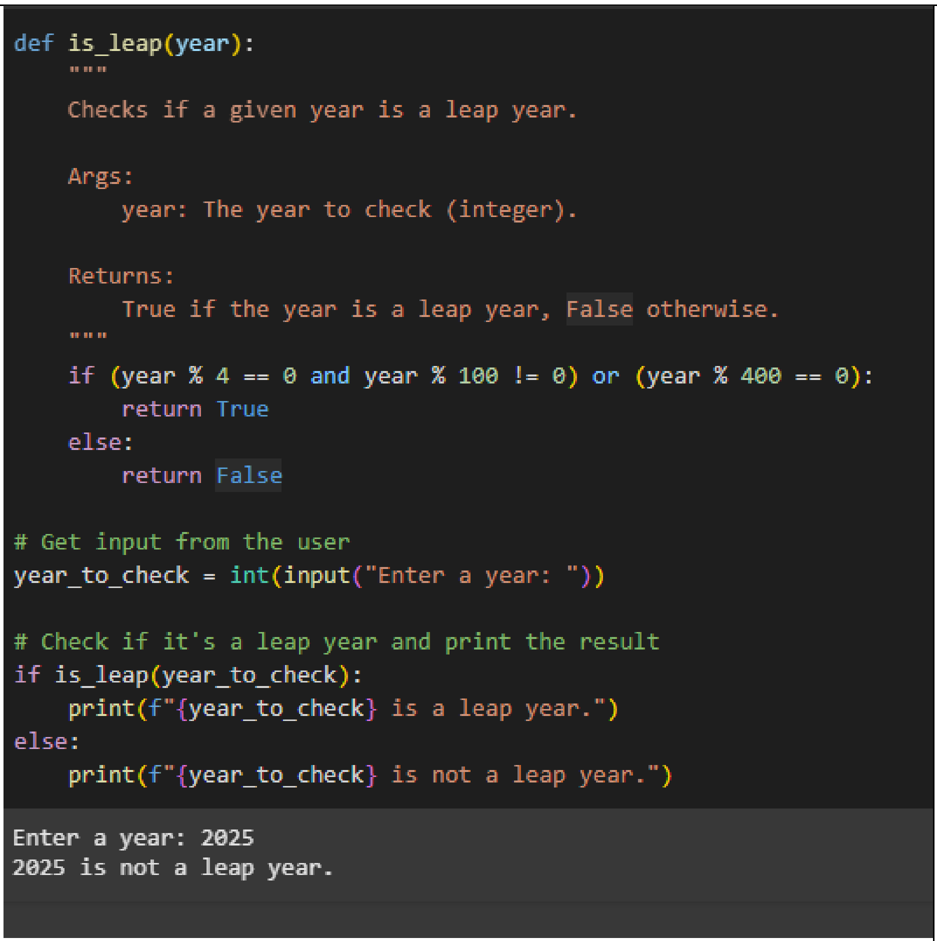
• Complete calculator code with user input and operation selection.
• Line-by-line explanation or markdown-style explanation provided by Gemini.
• Screenshot of both the code and explanation in Colab.



```python
def add(x, y):
  """Adds two numbers."""
  return x + y

def subtract(x, y):
  """Subtracts two numbers."""
  return x - y

def multiply(x, y):
  """Multiplies two numbers."""
  return x * y

def divide(x, y):
  """Divides two numbers."""
  if y == 0:
    return "Error: Division by zero"
  return x / y

print("Select operation:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

while True:
  choice = input("Enter choice(1/2/3/4): ")

  if choice in ('1', '2', '3', '4'):
    try:
      num1 = float(input("Enter first number: "))
      num2 = float(input("Enter second number: "))
    except ValueError:
      print("Invalid input. Please enter numbers.")
      continue

    if choice == '1':
      print(num1, "+", num2, "=", add(num1, num2))
    elif choice == '2':
      print(num1, "-", num2, "=", subtract(num1, num2))
    elif choice == '3':
      print(num1, "*", num2, "=", multiply(num1, num2))
    elif choice == '4':
      print(num1, "/", num2, "=", divide(num1, num2))
    break
  else:
    print("Invalid input. Please enter a valid choice.")
```

```
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice(1/2/3/4): 3
Enter first number: 56
Enter second number: 14
56.0 * 14.0 = 784.0
```

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



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

    Args:
        year: The year to check (integer).

    Returns:
        True if the year is a leap year, False otherwise.
    """
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

# Get input from the user
year_to_check = int(input("Enter a year: "))

# Check if it's a leap year and print the result
if is_leap(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")
```

```
Enter a year: 2025
2025 is not a leap year.
```

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**