SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name	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)		
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week1 - Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	24CSBTB01 To 24CSBTB39
Assignment Number: 2.1(Present assignment number)/**24**(Total number of assignments)			

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

- 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:

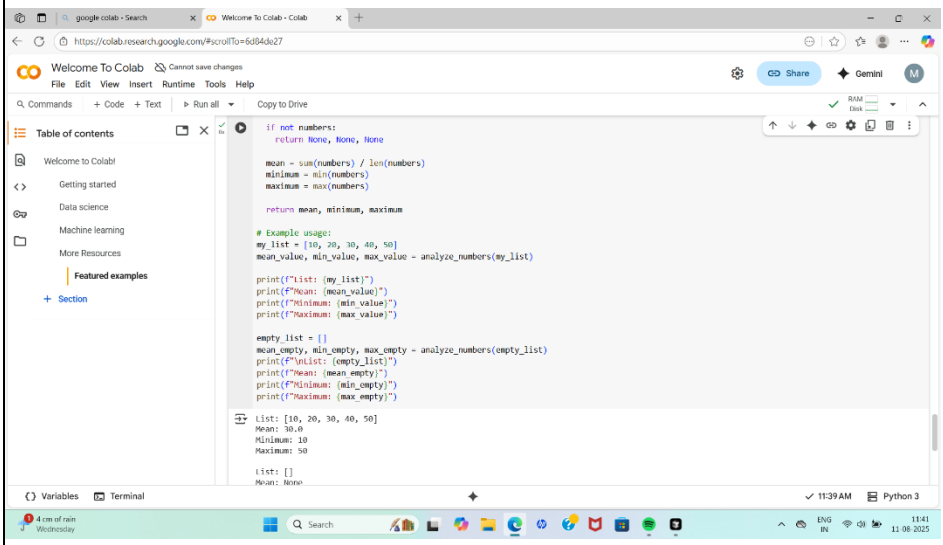
- Generate Python code using Google Gemini in Google Colab.
- Analyze the effectiveness of code explanations and suggestions by Gemini.
- Set up and use Cursor AI for AI-powered coding assistance.
- Evaluate and refactor code using Cursor AI features.
- Compare AI tool behavior and code quality across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

Task Description #1

- Use Google Gemini in Colab to write a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values.

Expected Output #1

- Functional code with correct output and screenshot.



Task Description #2

- Compare Gemini and Copilot outputs for a Python function that

checks whether a number is an Armstrong number. Document the steps, prompts, and outputs.

Expected Output #2

- Side-by-side comparison table with observations and screenshots.

```python
def is_armstrong_number(number):
  """
  Checks if a number is an Armstrong number.

  Args:
    number: An integer.

  Returns:
    True if the number is an Armstrong number, False otherwise.
  """
  # Convert the number to a string to get the number of digits
  num_str = str(number)
  num_digits = len(num_str)

  # Calculate the sum of each digit raised to the power of the number of digits
  armstrong_sum = 0
  for digit in num_str:
    armstrong_sum += int(digit) ** num_digits

  # Check if the sum is equal to the original number
  return armstrong_sum == number

# Example usage:
num1 = 153
num2 = 123

print(f"{num1} is an Armstrong number: {is_armstrong_number(num1)}")
print(f"{num2} is an Armstrong number: {is_armstrong_number(num2)}")
```

```
153 is an Armstrong number: True
123 is an Armstrong number: False
```

```python
def is_armstrong(number):
  num_str = str(number)
  num_digits = len(num_str)
  total = sum(int(digit) ** num_digits for digit in num_str)
  return total == number

# Example usage:
num = 56
if is_armstrong(num):
  print(f"{num} is an Armstrong number.")
else:
  print(f"{num} is not an Armstrong number.")
```

Task Description #3

- Ask Gemini to explain a Python function (e.g., is_prime(n) or is_palindrome(s)) line by line.
- Choose either a prime-checking or palindrome-checking function and document the explanation provided by Gemini.

Expected Output #3

- Detailed explanation with the code snippet and Gemini's response.

```python
def is_palindrome(s):
  """
  Checks if a string is a palindrome.

  Args:
    s: A string.

  Returns:
    True if the string is a palindrome, False otherwise.
  """
  cleaned_s = ''.join(char for char in s if char.isalnum()).lower()
  return cleaned_s == cleaned_s[::-1]

letter = "a"
print(f"Is '{letter}' a palindrome? {is_palindrome(letter)}")

letter = "Z"
print(f"Is '{letter}' a palindrome? {is_palindrome(letter)}")
```

```
Is 'a' a palindrome? True
Is 'Z' a palindrome? True
```

Task Description #4

- Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of the first N natural numbers) and test its output.
- Optionally, compare Cursor AI's generated code with Gemini's output.

Expected Output #4

- Screenshots of Cursor AI setup, prompts used, and generated code with outputs.

```python
def sum_of_natural_numbers(n):
  """Calculates the sum of the first n natural numbers."""
  if n < 0:
    return "Input should be a non-negative integer."
  else:
    return n * (n + 1) // 2

# Example usage
num = 10
sum_n = sum_of_natural_numbers(num)
print(f"The sum of the first {num} natural numbers is: {sum_n}")

num = 0
sum_n = sum_of_natural_numbers(num)
print(f"The sum of the first {num} natural numbers is: {sum_n}")

num = -5
sum_n = sum_of_natural_numbers(num)
print(f"The sum of the first {num} natural numbers is: {sum_n}")
```

```
The sum of the first 10 natural numbers is: 55
The sum of the first 0 natural numbers is: 0
The sum of the first -5 natural numbers is: Input should be a non-negative integer.
```

-

```python
def sum_natural_numbers(n):
    """
    Calculate the sum of the first n natural numbers

    Parameters:
    n (int): The number up to which the sum is calcu

    Returns:
    int: The sum of the first n natural numbers.
    """
    total = 0
    for i in range(1, n + 1):
        total += i
    return total

# Example usage
n = 10
result = sum_natural_numbers(n)
print(f"The sum of the first {n} natural numbers is:
```

Output:

text

```
The sum of the first 10 natural numbers is: 55
```

Task Description #5

- Students need to write a Python program to calculate the sum of odd numbers and even numbers in a given tuple.
- Refactor the code to improve logic and readability.

Expected Output #5

- Student-written refactored code with explanations and output screenshots.

```python
# Initialize sums
sum_even = 0
sum_odd = 0

# Manually compute sums using a loop
for num in numbers:
    if num % 2 == 0:
        sum_even = sum_even + num
    else:
        sum_odd = sum_odd + num

# Display results
print("Sum of even numbers:", sum_even)
print("Sum of odd numbers:", sum_odd)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\HARSHINI\Desktop\aiac> & C:/Users/HARSHINI/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/HARSHINI/Desktop/aiac/tuple.py
Sum of even numbers: 42
Sum of odd numbers: 26
PS C:\Users\HARSHINI\Desktop\aiac>
```

- **Improvements made:**

  1. **Better variable names:** Changed `sum_even` to `even_sum` and `sum_odd` to `odd_sum` for clarity
  2. **Clearer loop variable:** Changed `num` to `number` for better readability
  3. **Simplified addition:** Used `+=` operator instead of `= +` for cleaner code
  4. **Better comments:** Added more descriptive comments explaining each section
  5. **F-string formatting:** Used f-strings for cleaner output formatting
  6. **Consistent spacing:** Improved spacing and indentation for better readability

  Output:

  text

  ```text
  Sum of even numbers: 42
  Sum of odd numbers: 26
  ```

-

-

Note:

- Students must submit a single Word document including:
  - Prompts used for AI tools
  - Copilot/Gemini/Cursor outputs
  - Code explanations
  - Screenshots of outputs and environments

Evaluation Criteria:

Criteria	Max Marks
Successful Use of Gemini in Colab (Task#1 & #2)	1.0
Code Explanation Accuracy (Gemini) (Task#3)	0.5
Cursor AI Setup and Usage (Task#4)	0.5
Refactoring and Improvement Analysis (Task#5)	0.5
Total	**2.5 Marks**