B.Devendar
2303A53036
Batch - 46

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

Program Name: B. Tech	**Assignment Type: Lab**	**Academic Year:**2025-2026

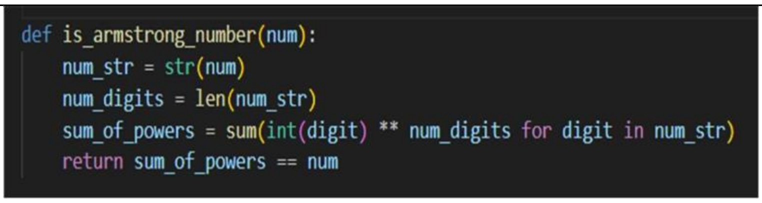
Course Coordinator Name	Dr. Rishabh Mittal
Instructor(s) Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju

CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week1 – Monday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches

Assignment Number:1.3(Present assignment number)/**24**(Total number of assignments)

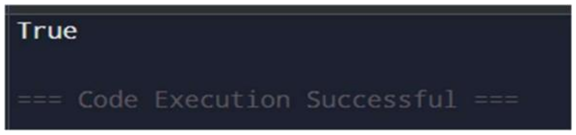
Q.No.	**Question**	***Expected Time to complete***

1	Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI **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 1: Statistical Summary for Survey Data ❖ **Scenario:** You are a **data analyst intern** working with survey responses stored as numerical lists. ❖ **Task:** Use **Google Gemini in Colab** to generate a Python function that reads a list of numbers and calculates the **mean, minimum, and maximum** values. ❖ **Expected Output:** ➢ Correct Python function ➢ Output shown in Colab ➢ Screenshot of Gemini prompt and result **Question:** Write a Python program to check whether a given number is an Armstrong number using user input and clear logic**.** **Code:**	Week1 - Monday



```python
def is_armstrong_number(num):
    num_str = str(num)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
    return sum_of_powers == num
```

Output:



```
True

=== Code Execution Successful ===
```

Task 2: Armstrong Number – AI Comparison

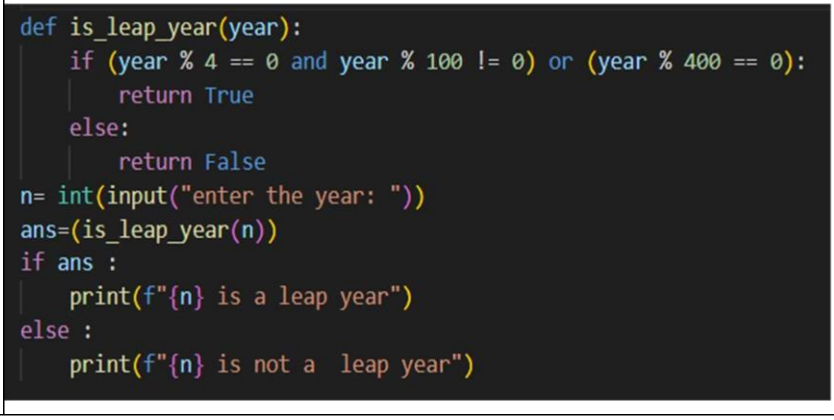
❖ **Scenario:**
You are evaluating AI tools for numeric validation logic.

❖ **Task:**
Generate an **Armstrong number checker** using **Gemini** and **GitHub Copilot**.
Compare their outputs, logic style, and clarity.

❖ **Expected Output:**
- ➢ Side-by-side comparison table
- ➢ Screenshots of prompts and generated code

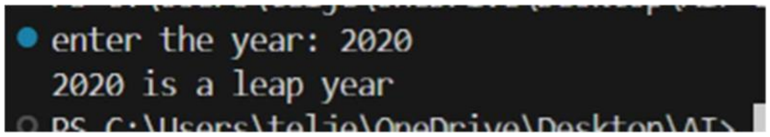
Question:
Write a Python program using if-else conditions to check whether a given year is a leap year.

Code:



```python
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
n= int(input("enter the year: "))
ans=(is_leap_year(n))
if ans :
    print(f"{n} is a leap year")
else :
    print(f"{n} is not a  leap year")
```

Output:



```
enter the year: 2020
2020 is a leap year
```

Task 3: Leap Year Validation Using Cursor AI

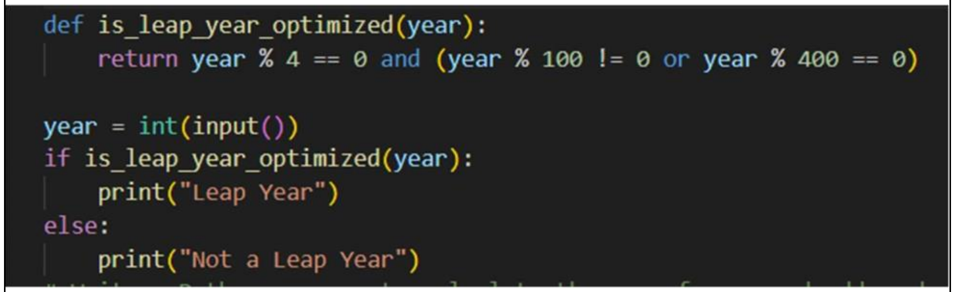
❖ **Scenario:**
You are validating a calendar module for a backend system.

❖ **Task:**
Use **Cursor AI** to generate a Python program that checks whether a given year is a leap year.
Use **at least two different prompts** and observe changes in code.

❖ **Expected Output:**
- ➢ Two versions of code
- ➢ Sample inputs/outputs
- ➢ Brief comparison

Question:
Write an optimized Python function that returns True or False to check whether a year is a leap year.

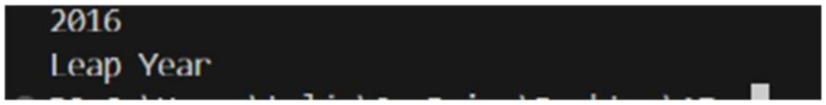
Code:



```python
def is_leap_year_optimized(year):
    return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

year = int(input())
if is_leap_year_optimized(year):
    print("Leap Year")
else:
    print("Not a Leap Year")
```

Output:



```
2016
Leap Year
```

Task 4: Student Logic + AI Refactoring (Odd/Even Sum)

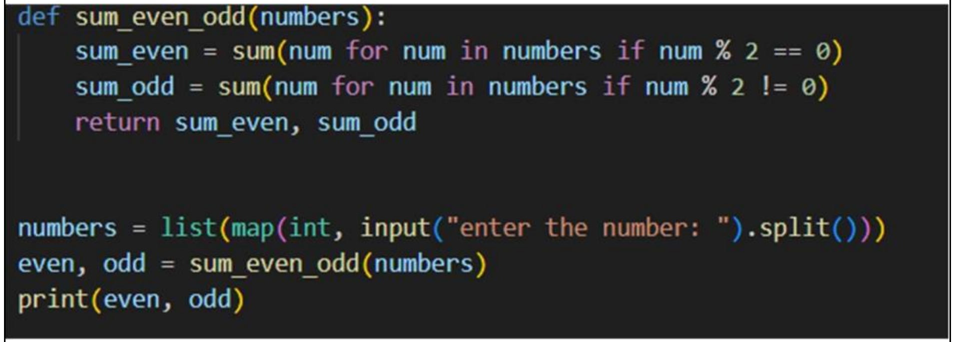
❖ **Scenario:**
Company policy requires developers to write logic before using AI.

❖ **Task:**
Write a Python program that calculates the **sum of odd and even numbers in a tuple**, then refactor it using any AI tool.

❖ **Expected Output:**
- ➢ Original code
- ➢ Refactored code
- ➢ Explanation of improvements

Question:
Write a Python program to calculate the sum of even and odd numbers in a tuple without advanced functions

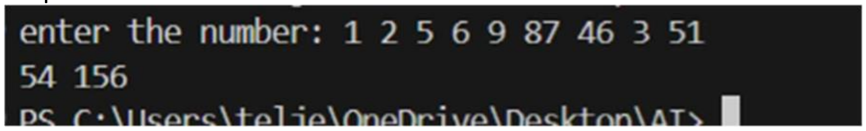
Code:



```python
def sum_even_odd(numbers):
    sum_even = sum(num for num in numbers if num % 2 == 0)
    sum_odd = sum(num for num in numbers if num % 2 != 0)
    return sum_even, sum_odd


numbers = list(map(int, input("enter the number: ").split()))
even, odd = sum_even_odd(numbers)
print(even, odd)
```

Output:



```
enter the number: 1 2 5 6 9 87 46 3 51
54 156
PS C:\Users\telje\OneDrive\Desktop\AI>
```