

Assignment-2.1

S.Thrishla

2303A51165

Batch-18

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

PROMPT:

Write a Python function that takes a list of numbers from user input and calculates mean, minimum, and maximum values. Handle empty input and display results clearly.

CODE:

```
def statistical_summary(data):  
    if not data:  
        print("List is empty. No statistics available.")  
        return  
  
    mean_val = sum(data) / len(data)  
    min_val = min(data)  
    max_val = max(data)  
  
    print("\nStatistical Summary")  
    print("Mean : ", mean_val)  
    print("Minimum:", min_val)
```

```

print("Maximum:", max_val)

nums = list(map(float, input("Enter numbers separated by space: ").split()))

statistical_summary(nums)

● PS C:\Users\srini\OneDrive\Desktop\ai> cd 'c:\Users\srini\OneDrive\Desktop\ai'; 18.0\bundled\libs\debugpy\launcher' '54182' '--' 'c:\Users\srini\OneDrive\Desktop\2.
Enter numbers separated by space: 3 4 5

    Statistical Summary
    Mean   : 4.0
    Minimum: 3.0
    Maximum: 5.0
● PS C:\Users\srini\OneDrive\Desktop\ai> cd 'c:\Users\srini\OneDrive\Desktop\ai'; 18.0\bundled\libs\debugpy\launcher' '50677' '--' 'c:\Users\srini\OneDrive\Desktop\2.
Enter numbers separated by space: 10 23 45

    Statistical Summary
    Mean   : 26.0
    Minimum: 10.0
    Maximum: 45.0
● PS C:\Users\srini\OneDrive\Desktop\ai> cd 'c:\Users\srini\OneDrive\Desktop\ai'.

```

Explanation Steps :

1. Take numbers from user using `input()`.
2. Convert input into list using `split()` and `map()`.
3. Calculate mean using `sum()/len()`.
4. Use `min()` and `max()` functions.
5. Print results.

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

PROMPT:

Write a Python program to check whether a number is an Armstrong number. Take input from the user and print the result with clear logic and comments.

CODE:

```
def is_armstrong(num):  
    digits = len(str(num))  
    temp = num  
    total = 0  
  
    while temp > 0:  
        digit = temp % 10  
        total += digit ** digits  
        temp //= 10  
  
    return total == num  
  
number = int(input("Enter a number: "))  
  
if is_armstrong(number):  
    print("It is an Armstrong number.")  
  
else:  
    print("It is NOT an Armstrong number.")
```

```
● PS C:\Users\srini\OneDrive\Desktop\ai> c;; cd 'c:\Users\srini'  
18.0\bundled\libs\debugpy\launcher' '64201' '--' 'c:\Users\srin  
Enter a number: 456  
It is NOT an Armstrong number.  
● PS C:\Users\srini\OneDrive\Desktop\ai> c;; cd 'c:\Users\srini'
```

Explanation Steps:

1. Take number as input.
2. Count digits using `len(str())`.
3. Extract each digit.
4. Raise digit to power of digits and sum.
5. Compare with original number.

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

PROMPT:

Write a simple Python program to check leap year using if-else.

CODE:

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

if year % 4 == 0:

    if year % 100 == 0:

        if year % 400 == 0:

            print("Leap Year")

        else:

            print("Not a Leap Year")

    else:

        print("Leap Year")

else:

    print("Not a Leap Year")
```

```
● PS C:\Users\srini\OneDrive\Desktop\ai> cd 'c:\Users\srini\OneDrive\Desktop\ai' & python -m bundled.libs.debugpy.launcher '53563' '--'
Enter a year: 2028
Leap Year
● PS C:\Users\srini\OneDrive\Desktop\ai>
```

Explanation Steps:

1. Take year as input.
2. Check divisibility rules:
 - o divisible by 4
 - o not divisible by 100 unless divisible by 400
3. Print result.

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

PROMPT:

Refactor my Python code that calculates sum of odd and even numbers in a tuple. Make it cleaner, reusable, and more Pythonic.

CODE:

```
numbers = (10, 15, 20, 25, 30, 35)

even_sum = 0
odd_sum = 0

for num in numbers:
    if num % 2 == 0:
        even_sum += num
    else:
        odd_sum += num
```

```
print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)

● PS C:\Users\srini\OneDrive\Desktop\ai> c;; cd 'c:\Users\srini\18.0\bundled\libs\debug\launcher' '56445' '--' 'c:\Users\srini\Sum of even numbers: 60
Sum of odd numbers: 75
○ PS C:\Users\srini\OneDrive\Desktop\ai> █
```

Explanation Steps:

1. Loop through tuple.
 2. Check even/odd using % 2.
 3. Add to respective sums.
 4. AI refactors into function.
 5. Uses sum() and generator expressions