

# Lab\_Assignment\_2.1

2303A51511

Batch-27

Task 1:

**#prompt:** “Write a Python function that takes a list of numerical survey responses and calculates the mean, minimum, and maximum values. Also show sample output.”

**Code:**

```
def statistical_summary(data):
    total = sum(data)
    mean = total / len(data)
    minimum = min(data)
    maximum = max(data)
    return mean, minimum, maximum

# Sample data
survey_data = [12, 45, 23, 67, 34, 89, 10]
mean, minimum, maximum = statistical_summary(survey_data)
print("Mean:", mean)
print("Minimum:", minimum)
print("Maximum:", maximum)
```

```
print("Mean:", mean)
print("Minimum:", minimum)
print("Maximum:", maximum)
```

```
... Mean: 40.0
      Minimum: 10
      Maximum: 89
```

**Explanation:** The `statistical_summary()` function takes a list of numerical survey responses and computes basic statistics.

It calculates the mean by dividing the total sum of values by the number of elements. The function also finds the minimum and maximum values and returns all three results for display.

## Task 2:

# **prompt** : Write a program to check whether a given number is an Armstrong number or not.

### #code:

```
def is_armstrong_number(num):

    # Convert the number to string to easily iterate over digits

    str_num = str(num)

    num_digits = len(str_num)

    # Calculate the sum of the cubes of each digit

    sum_of_cubes = sum(int(digit) ** num_digits for digit in str_num)

    # Check if the sum of cubes is equal to the original number

    return sum_of_cubes == num

# Test the function

number = int(input("Enter a number: "))

if is_armstrong_number(number):

    print(f"{number} is an Armstrong number.")

else:

    print(f"{number} is not an Armstrong number.")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ - | X
PS C:\Users\vaish\OneDrive\Desktop\AI ASSISTED CODING> python -u "c:\Users\vaish\OneDrive\Desktop\AI ASSISTED CODING\task-2.py"
PS C:\Users\vaish\OneDrive\Desktop\AI ASSISTED CODING> python -u "c:\Users\vaish\OneDrive\Desktop\AI ASSISTED CODING\task-2.py"
Enter a number: 153
153 is an Armstrong number.
PS C:\Users\vaish\OneDrive\Desktop\AI ASSISTED CODING> []
```

**Explanation:** The function converts the given number into a string to easily access each digit and counts the total number of digits.

Each digit is raised to the power of the number of digits and summed to calculate the Armstrong value. If this calculated sum equals the original number, the function confirms it as an Armstrong number.

### Task 3: Cursor AI

#prompt: Generate a Python program that checks whether a given year is a leap year.

#code:

```
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False

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

if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```



The screenshot shows a terminal window with the following text:

```
Problems Output Debug Console Terminal Ports
PS C:\Users\vamsh\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/vamsh/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/vamsh/OneDrive/Desktop/AI ASSISTED CODING/lab2-1.py"
Enter a year: 2024
2024 is a leap year.
PS C:\Users\vamsh\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/vamsh/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/vamsh/OneDrive/Desktop/AI ASSISTED CODING/lab2-1.py"
Enter a year: 2026
2026 is not a leap year.
PS C:\Users\vamsh\OneDrive\Desktop\AI ASSISTED CODING> []
```

**Explanation:** The function checks if a year is divisible by 4, which is the first condition for a leap year.

If the year is divisible by 100, it must also be divisible by 400 to qualify as a leap year.

The function returns True if all conditions are met, otherwise False, and prints the result accordingly.

## Task 4:

# **prompt:** Write a Python program that calculates the sum of odd and even numbers in a tuple, then refactor

### Code:

```
def sum_odd_even(numbers):
    sum_odd = 0
    sum_even = 0

    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num

    return sum_odd, sum_even

# Test the function
numbers = (1, 2, 3, 4, 5, 6)
odd_sum, even_sum = sum_odd_even(numbers)
print(f"Sum of odd numbers: {odd_sum}")
print(f"Sum of even numbers: {even_sum}")

# Refactored version using list comprehensions
def sum_odd_even_refactored(numbers):
    sum_odd = sum(num for num in numbers if num % 2 != 0)
    sum_even = sum(num for num in numbers if num % 2 == 0)
    return sum_odd, sum_even

# Test the refactored function
odd_sum_ref, even_sum_ref = sum_odd_even_refactored(numbers)
print(f"Refactored - Sum of odd numbers: {odd_sum_ref}")
print(f"Refactored - Sum of even numbers: {even_sum_ref}")
```

A screenshot of a terminal window from a development environment. The tab bar at the top includes PROBLEMS, OUTPUT (which is selected), DEBUG CONSOLE, TERMINAL, and PORTS. A search bar labeled 'Filter (e.g. text, excludeText, t...)' is followed by a 'Code' button and some icons. The main area displays the following text:

```
[Running] python -u "c:\Users\vamsh\OneDrive\Desktop\AI ASSISTED CODING\task-22.py"
Sum of odd numbers: 9
Sum of even numbers: 12
Refactored - Sum of odd numbers: 9
Refactored - Sum of even numbers: 12

[Done] exited with code=0 in 0.056 seconds
```

**Explanation:** The function separates numbers into odd and even by checking the remainder when divided by 2 and sums them separately.

It returns the sum of odd and even numbers, which are then printed for verification.

The refactored version uses list comprehensions to calculate the sums more concisely while producing the same results.