ASSIGNMENT - 02

Name	**:**	**Naga Rishik Reddy**
Hall Ticket No	**:**	2303A51089
Batch No.	**:**	2
Course	**:**	**AI Assisstant Coding**

Task 1: Statistical Summary for Survey Data

- **Scenario:**

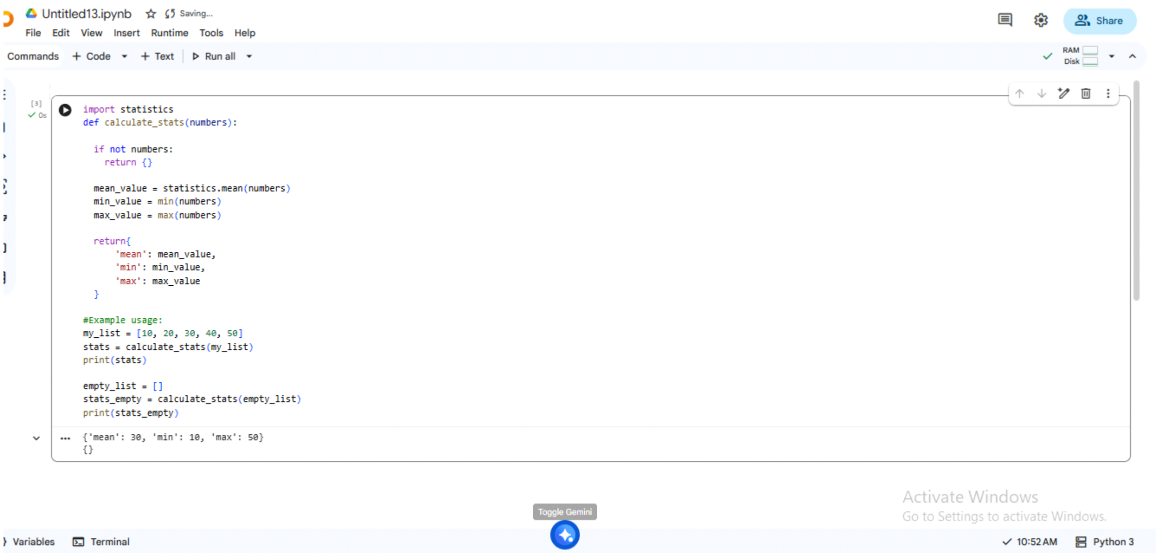
You are a data analyst intern working with survey responses stored as numerical lists.

- **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**



```python
import statistics
def calculate_stats(numbers):

  if not numbers:
    return {}

  mean_value = statistics.mean(numbers)
  min_value = min(numbers)
  max_value = max(numbers)

  return{
      'mean': mean_value,
      'min': min_value,
      'max': max_value
  }

#Example usage:
my_list = [10, 20, 30, 40, 50]
stats = calculate_stats(my_list)
print(stats)

empty_list = []
stats_empty = calculate_stats(empty_list)
print(stats_empty)
```

```
{'mean': 30, 'min': 10, 'max': 50}
{}
```

Task 2: Armstrong Number – AI Comparison

- **Scenario:**

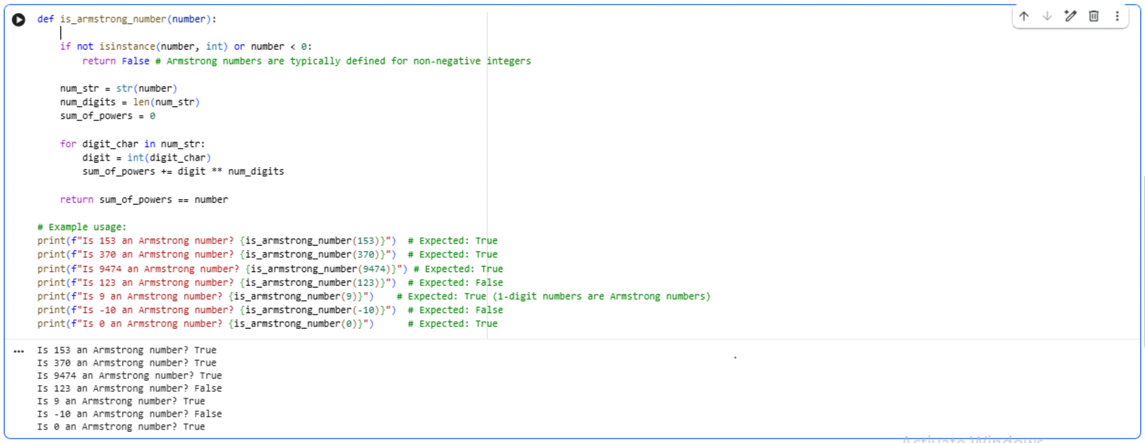
You are evaluating AI tools for numeric validation logic.

- **Task:**

Generate an Armstrong number checker using Gemini and GitHub Copilot.

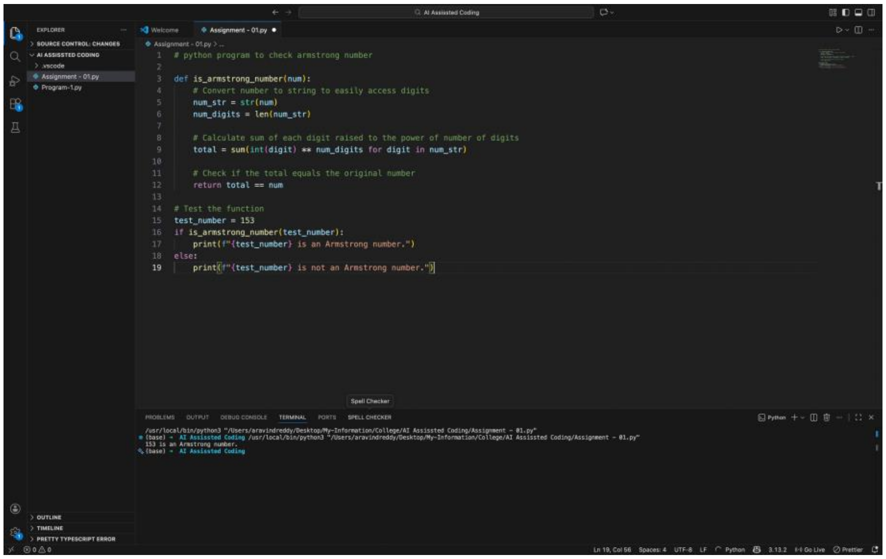
Compare their outputs, logic style, and clarity.

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



```python
def is_armstrong_number(number):

    if not isinstance(number, int) or number < 0:
        return False # Armstrong numbers are typically defined for non-negative integers

    num_str = str(number)
    num_digits = len(num_str)
    sum_of_powers = 0

    for digit_char in num_str:
        digit = int(digit_char)
        sum_of_powers += digit ** num_digits

    return sum_of_powers == number

# Example usage:
print(f"Is 153 an Armstrong number? {is_armstrong_number(153)}")  # Expected: True
print(f"Is 370 an Armstrong number? {is_armstrong_number(370)}")  # Expected: True
print(f"Is 9474 an Armstrong number? {is_armstrong_number(9474)}") # Expected: True
print(f"Is 123 an Armstrong number? {is_armstrong_number(123)}")  # Expected: False
print(f"Is 9 an Armstrong number? {is_armstrong_number(9)}")    # Expected: True (1-digit numbers are Armstrong numbers)
print(f"Is -10 an Armstrong number? {is_armstrong_number(-10)}")  # Expected: False
print(f"Is 0 an Armstrong number? {is_armstrong_number(0)}")      # Expected: True
```

```
Is 153 an Armstrong number? True
Is 370 an Armstrong number? True
Is 9474 an Armstrong number? True
Is 123 an Armstrong number? False
Is 9 an Armstrong number? True
Is -10 an Armstrong number? False
Is 0 an Armstrong number? True
```



```python
# python program to check armstrong number

def is_armstrong_number(num):
    # Convert number to string to easily access digits
    num_str = str(num)
    num_digits = len(num_str)

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

    # Check if the total equals the original number
    return total == num

# Test the function
test_number = 153
if is_armstrong_number(test_number):
    print(f"{test_number} is an Armstrong number.")
else:
    print(f"{test_number} is not an Armstrong number.")
```

```
/usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignment - 01.py"
(base) → AI Assissted Coding /usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignment - 01.py"
153 is an Armstrong number.
(base) → AI Assissted Coding
```

Task 3: Leap Year Validation Using Cursor AI

- **Scenario:**

You are validating a calendar module for a backend system.

- **Task:**

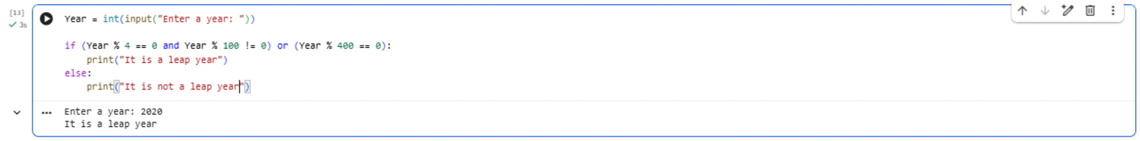
Use Cursor AI to generate a Python program that checks whether a given year is a leap year.

Use at least two different prompts and observe changes in code.

- **Expected Output:**
- **Two versions of code**

- **Sample inputs/outputs**

- **Brief comparison**



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

if (Year % 4 == 0 and Year % 100 != 0) or (Year % 400 == 0):
    print("It is a leap year")
else:
    print("It is not a leap year")
```

```
Enter a year: 2020
It is a leap year
```

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

- **Scenario:**

Company policy requires developers to write logic before using AI.

- **Task:**

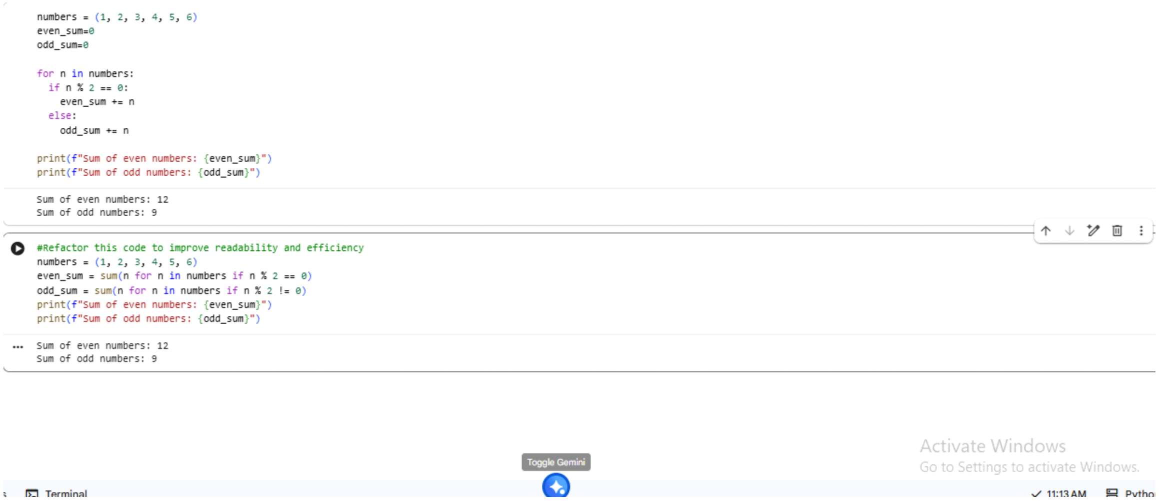
Write a Python program that calculates the sum of odd and even numbers in a tuple, then refactor it using any AI tool.

- **Expected Output:**

- **Original code**

- **Refactored code**

- **Explanation of improvements**



```python
numbers = (1, 2, 3, 4, 5, 6)
even_sum=0
odd_sum=0

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

print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

```
Sum of even numbers: 12
Sum of odd numbers: 9
```

```python
#Refactor this code to improve readability and efficiency
numbers = (1, 2, 3, 4, 5, 6)
even_sum = sum(n for n in numbers if n % 2 == 0)
odd_sum = sum(n for n in numbers if n % 2 != 0)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

```
Sum of even numbers: 12
Sum of odd numbers: 9
```