

# AI-assisted coding

## 2303A510B7(Harini)

### Assignment-2.1

#### Task 1:

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

The screenshot shows the Google Colab interface with a dark theme. On the left, a code cell contains a Python function named `calculate_stats`. The function calculates the mean, minimum, and maximum of a list of numbers. It includes docstrings and examples. On the right, a Gemini sidebar provides a prompt and a generated response. The response includes a link to explain the function, a link to refactor it for non-numeric input, and a link to add unit tests. The Gemini interface also shows a "Gemini 2.5 Flash" button and a note about potential mistakes.

```
def calculate_stats(numbers):
    """
    Calculates the mean, minimum, and maximum of a list of numbers.

    Args:
        numbers (list): A list of numerical values.

    Returns:
        tuple: A tuple containing the mean, minimum, and maximum values.
        Returns (None, None, None) if the list is empty.
    """
    if not numbers:
        return None, None, None
    mean_value = sum(numbers) / len(numbers)
    min_value = min(numbers)
    max_value = max(numbers)
    return mean_value, min_value, max_value

# Example usage:
my_list = [10, 20, 30, 40, 50, 5, 25]
mean, minimum, maximum = calculate_stats(my_list)
if mean is not None:
    print("List: {}")
    print("Mean: {}".format(mean))
    print("Minimum: {}".format(minimum))
    print("Maximum: {}".format(maximum))
else:
    print("The list is empty.")
empty_list = []
mean, minimum, maximum = calculate_stats(empty_list)
print("List: {}")
print("Mean: {}, Minimum: {}, Maximum: {}".format(mean, minimum, maximum))

List: [10, 20, 30, 40, 50, 5, 25]
Mean: 25.714285714285715
Minimum: 5
Maximum: 50

List: []
Mean: None, Minimum: None, Maximum: None
```

#### Task 2 :

Generate an Armstrong number checker using Gemini and GitHub Copilot.

The screenshot shows the Google Colab interface with a dark theme. On the left, a code cell contains a Python function named `is_armstrong_number`. The function checks if a given number is an Armstrong number by calculating the sum of its digits raised to the power of their count. It handles user input and errors. On the right, a Gemini sidebar provides a prompt and a generated response. The response includes links to run the code, explain the function, and test it with the number 153. The Gemini interface also shows a "Gemini 2.5 Flash" button and a note about potential mistakes.

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

    # Convert number to string to find its length (number of digits)
    str_number = str(number)
    num_digits = len(str_number)

    sum_of_powers = 0
    temp_number = number
    while temp_number > 0:
        digit = temp_number % 10
        sum_of_powers += digit ** num_digits
        temp_number //= 10

    return sum_of_powers == number

try:
    user_input = input("Enter a non-negative integer to check if it's an Armstrong number: ")
    num_to_check = int(user_input)

    if num_to_check < 0:
        print("Please enter a non-negative integer.")
    elif is_armstrong_number(num_to_check):
        print(f"The number {num_to_check} IS an Armstrong number.")
    else:
        print(f"The number {num_to_check} IS NOT an Armstrong number.")
except ValueError:
    print("Invalid input. Please enter an integer.")

Enter a non-negative integer to check if it's an Armstrong number: 153
The number 153 IS an Armstrong number.
```

```
ASS2.1(git).py > ...
1  #generate a code to find Armstrong number checker in python give user defined
2  def is_armstrong_number(num):
3      # Convert the number to string to easily iterate over digits
4      str_num = str(num)
5      num_digits = len(str_num)
6
7      # Calculate the sum of each digit raised to the power of num_digits
8      sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
9
10     # An Armstrong number is equal to the sum of its own digits each raised to the po
11     return sum_of_powers == num
12
13 # Get user input
14 user_input = int(input("Enter a number to check if it is an Armstrong number: "))
15 if is_armstrong_number(user_input):
16     print(f"{user_input} is an Armstrong number.")
17 else:
18     print(f"{user_input} is not an Armstrong number.")

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    COMMENTS    Python    +    □    ...    ^    ×

/usr/local/bin/python3 "/Users/harini/Desktop/AI_2303A510B7/ASS2.1(git).py"
● (base) harini@HARINI-MacBook-Air AI_2303A510B7 % /usr/local/bin/python3 "/Users/harini/Desktop/AI_
2303A510B7/ASS2.1(git).py"
Enter a number to check if it is an Armstrong number: 153
153 is an Armstrong number.
○ (base) harini@HARINI-MacBook-Air AI_2303A510B7 %
```

### **Task 3 :**

**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.**

The screenshot shows a Jupyter Notebook environment with the following details:

- Top Bar:** Agents, Editor, AI\_2303A510B7, View Plans, and various icons.
- Left Sidebar:** A tree view showing the project structure:
  - AI\_2303A510B7
  - ASS\_2(cursor ai).py (selected)
  - ASS-1.4-10B7.pages
  - ASS-1.4.py
  - ASS2.1(git).py
  - Assignment 1 HPC d...
- Code Cell:** The selected cell contains Python code for determining if a year is a leap year:

```
1 # generate a code Leap Year Validation in python give user defined
2 def is_leap_year(year):
3     if year % 4 == 0:
4         if year % 100 == 0:
5             if year % 400 == 0:
6                 return True
7             else:
8                 return False
9         else:
10            return True
11     else:
12        return False
13 year = int(input("Enter a year: "))
14 print(is_leap_year(year))
15 | %%L to chat, %%K to generate
```
- Bottom Navigation:** Problems, Output, Debug Console, Terminal, Ports, and a Python kernel icon.
- Terminal Output:** Shows the execution of the script and its output:

```
/usr/local/bin/python3 "/Users/harini/Desktop/AI_2303A510B7/ASS_2(cursor ai).py"
(base) harini@HARINI-MacBook-Air AI_2303A510B7 % /usr/local/bin/python3 "/Users/harini/Desktop/AI_2303A510B7/ASS_2(cursor ai).py"
Enter a year: 2004
True
(base) harini@HARINI-MacBook-Air AI_2303A510B7 %
```
- Bottom Status:** %%K to generate command, Cursor Tab, Ln 15, Col 1, Spaces: 4, UTF-8, LF, Python 3.12.3 64-bit.

The screenshot shows a Jupyter Notebook interface with the following details:

- Top Bar:** Agents, Editor, AI\_2303A510B7, View Plans, and a gear icon.
- Left Sidebar:** A tree view showing files: AI\_2303A510B7, ASS\_2(cursor ai).py, ASS-1.4-10B7.pages, ASS-1.4.py, ASS2.1(git).py, and Assignment 1 HPC d...
- Code Cell:** The code is as follows:

```
18
19     # write a program to print leap year or not without using function
20     year = int(input("Enter a year: "))
21     if year % 4 == 0:
22         if year % 100 == 0:
23             if year % 400 == 0:
24                 print("Leap year")
25             else:
26                 print("Not a leap year")
27         else:
28             print("Leap year")
29     else:
30         print("Not a leap year")
```
- Bottom Navigation:** Problems, Output, Debug Console, Terminal (selected), Ports, Python, and other icons.
- Terminal Output:** Shows the execution of the script and its output.

```
/usr/local/bin/python3 "/Users/harini/Desktop/AI_2303A510B7/ASS_2(cursor ai).py"
● (base) harini@HARINI-MacBook-Air AI_2303A510B7 % /usr/local/bin/python3 "/Users/harini/Desktop/AI_2303A510B7/ASS_2(cursor ai).py"
Enter a year: 2024
Leap year
○ (base) harini@HARINI-MacBook-Air AI_2303A510B7 %
```
- Bottom Status Bar:** Includes icons for file operations, a cursor tab, and text input fields for command generation.

#### **Task 4 :**

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

The screenshot shows a Python script named `ASS-1.4.py` in a code editor. The code uses a tuple comprehension to read elements from the user, initializes two variables `p` and `q` to 0, and then iterates through the list to calculate the sum of even and odd numbers respectively. Finally, it prints the results.

```
1 t=tuple(map(int,input("Enter elements: ").split()))
2 l=list(t)
3 p=0
4 q=0
5 for i in l:
6     if i%2==0:
7         p+=i
8     else:
9         q+=i
10 print(f"sum of even: {p} and sum of odd: {q}")
```

The terminal below shows the script being run in a Mac OS terminal window. The user enters a list of integers, and the script outputs the sum of even numbers (20) and the sum of odd numbers (25).

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

/usr/local/bin/python3 /Users/harini/Desktop/AI\_2303A510B7/ASS-1.4.py  
● (base) harini@HARINI-MacBook-Air AI\_2303A510B7 % /usr/local/bin/python3 /Users/harini/Desktop/AI\_2303A510B7/ASS-1.4.py  
Enter elements: 1 2 3 4 5 6 7 8 9 0  
sum of even: 20 and sum of odd: 25  
○ (base) harini@HARINI-MacBook-Air AI\_2303A510B7 %

Agents Editor AI\_2303A510B7 View Plans + ⌂ ⌄ ⌅ ⌆ ⌇

AI\_2303A510B7 ASS\_2(cursor ai).py > ...

```
33
34     #Write a Python program that calculates the sum of odd and even numbers in a tuple
35     t=tuple[int, ...](map[int](int,input("Enter elements: ")).split())
36     l=list[int](t)
37     p=0
38     q=0
39     for i in l:
40         if i%2==0:
41             p+=i
42         else:
43             q+=i
44     print(f"sum of even: {p} and sum of odd: {q}")
```

Problems Output Debug Console Terminal Ports Python + ⌂ ⌄ ⌅ ⌆ ⌇

```
/usr/local/bin/python3 "/Users/harini/Desktop/AI_2303A510B7/ASS_2(cursor ai).py"
● (base) harini@HARINI-MacBook-Air AI_2303A510B7 % /usr/local/bin/python3 "/Users/harini/Desktop/AI_2303A510B7/ASS_2(cursor ai).py"
Enter elements: 1 3 5 6 7 5 0 78
sum of even: 84 and sum of odd: 21
○ (base) harini@HARINI-MacBook-Air AI_2303A510B7 %
```

> OUTLINE > TIMELINE ⌂ to generate command Cursor Tab ⌂ Ln 44, Col 49 Spaces: 4 UTF-8 LF Python 3.12.3 64-bit ⌂