ASSIGNMENT - 02

Name : Aravind Reddy

Hall Ticket No : 2303a51027

Batch No. : 01

Course : **AI Assisstant Coding**

Task 1: Statistical Summary for Survey Data

❖ Scenario:

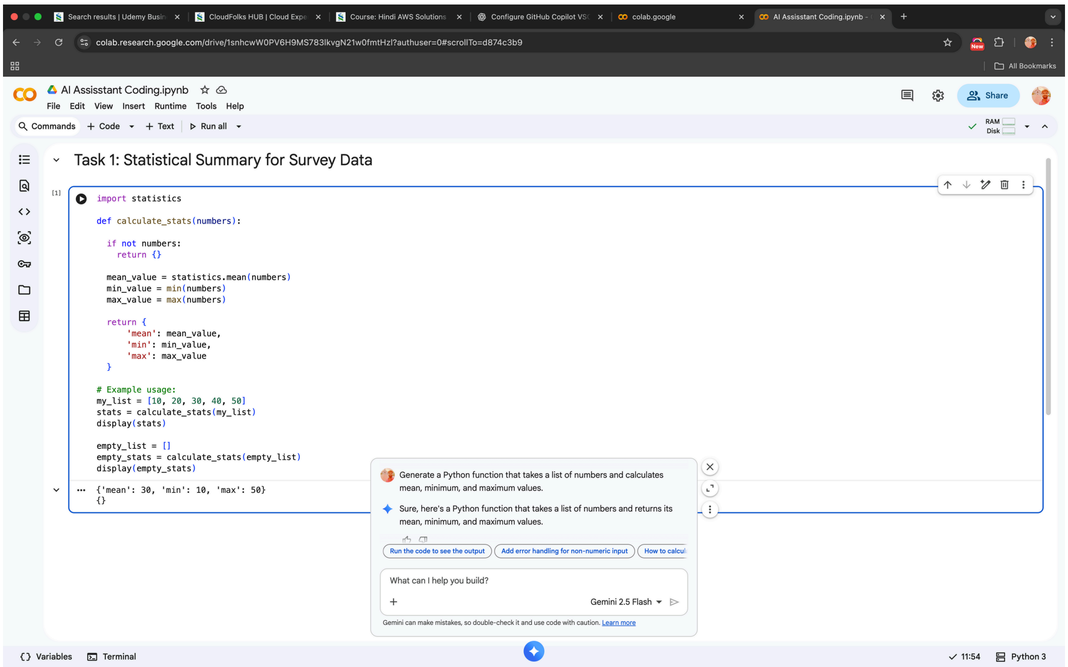
You are a data analyst intern working with survey responses stored as numerical lists.

❖ 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**


colab.research.google.com/drive/1snhcwW0PV6H9MS783lkvgN21w0fmtHzl?authuser=0#scrollTo=d874c3b9
AI Assistant Coding.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Variables
Terminal
Python 3

Task 1: Statistical Summary for Survey Data

```
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)
display(stats)

empty_list = []
empty_stats = calculate_stats(empty_list)
display(empty_stats)
```

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

Generate a Python function that takes a list of numbers and calculates mean, minimum, and maximum values.

Sure, here's a Python function that takes a list of numbers and returns its mean, minimum, and maximum values.

Run the code to see the output
Add error handling for non-numeric input
How to calcul
What can I help you build?
Gemini 2.5 Flash

Gemini can make mistakes, so double-check it and use code with caution. Learn more

---

Task 2: Armstrong Number – AI Comparison

✣ Scenario:

You are evaluating AI tools for numeric validation logic.

✣ Task:

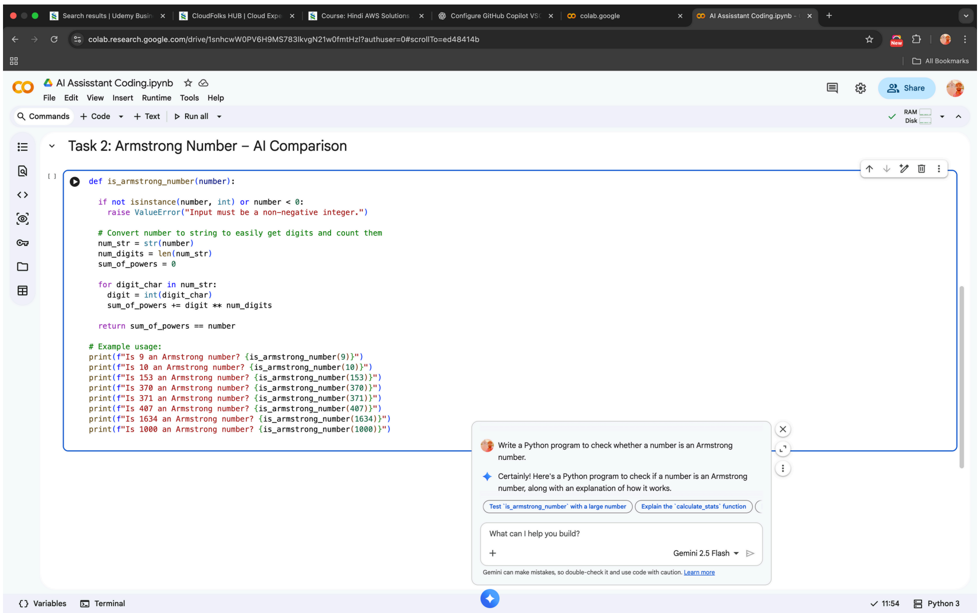
Generate an Armstrong number checker using Gemini and GitHub Copilot.

Compare their outputs, logic style, and clarity.

✣ Expected Output:

➢ Side-by-side comparison table

➢ Screenshots of prompts and generated code


AI Assisstant Coding.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Variables
Terminal
11:54
Python 3
Test `is_armstrong_number` with a large number
Explain the `calculate_stats` function
What can I help you build?
Gemini 2.5 Flash

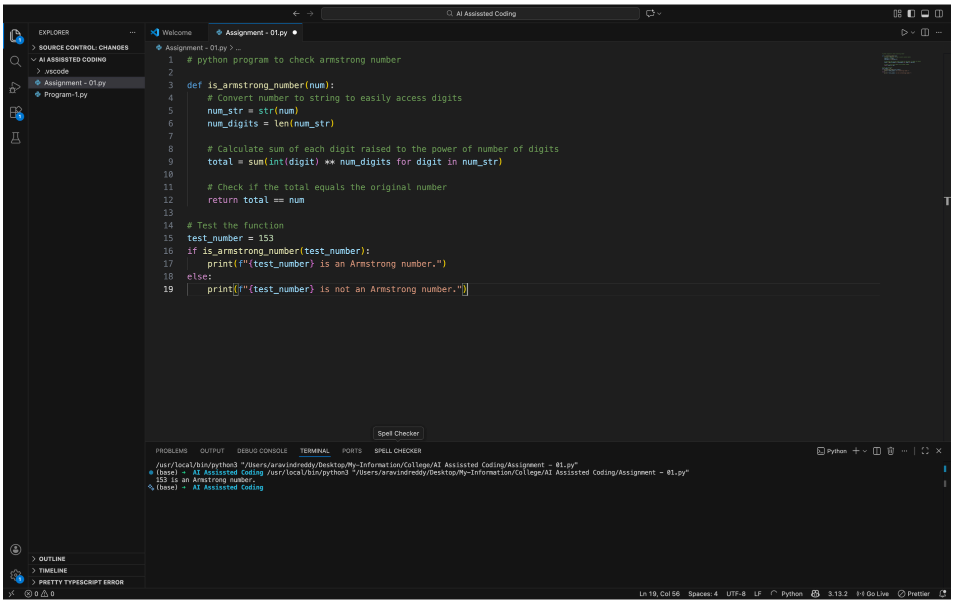
Task 2: Armstrong Number – AI Comparison

```
def is_armstrong_number(number):

  if not isinstance(number, int) or number < 0:
    raise ValueError("Input must be a non-negative integer.")

  # Convert number to string to easily get digits and count them
  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 9 an Armstrong number? {is_armstrong_number(9)}")
print(f"Is 10 an Armstrong number? {is_armstrong_number(10)}")
print(f"Is 153 an Armstrong number? {is_armstrong_number(153)}")
print(f"Is 370 an Armstrong number? {is_armstrong_number(370)}")
print(f"Is 371 an Armstrong number? {is_armstrong_number(371)}")
print(f"Is 407 an Armstrong number? {is_armstrong_number(407)}")
print(f"Is 1634 an Armstrong number? {is_armstrong_number(1634)}")
print(f"Is 1000 an Armstrong number? {is_armstrong_number(1000)}")
```

Write a Python program to check whether a number is an Armstrong number.

Certainly! Here's a Python program to check if a number is an Armstrong number, along with an explanation of how it works.

Gemini can make mistakes, so double-check it and use code with caution. Learn more



```
# 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.

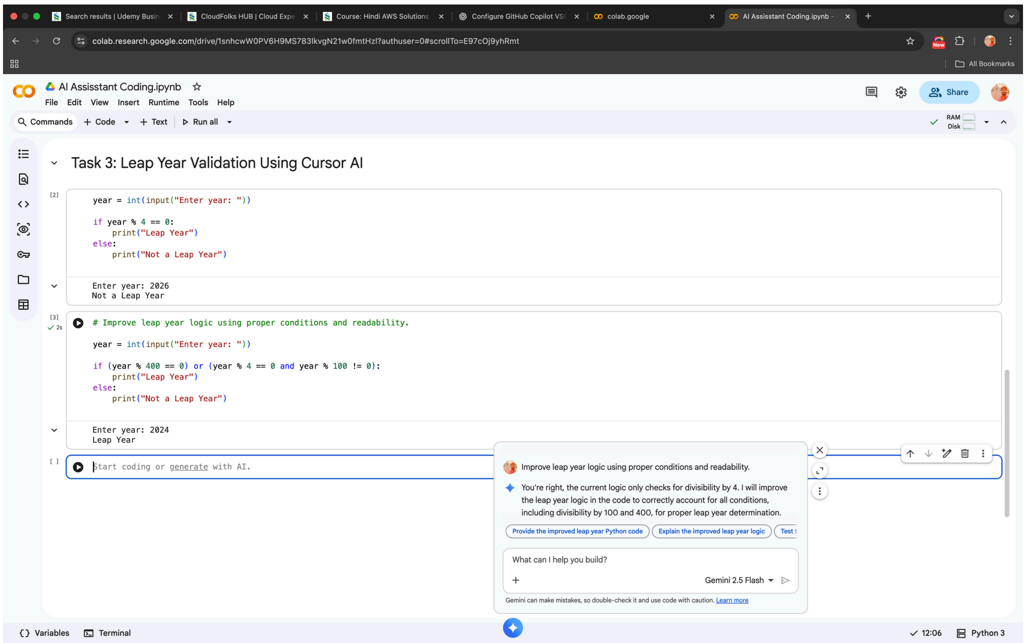
Use at least two different prompts and observe changes in code.

❖ Expected Output:

➢ Two versions of code

➢ Sample inputs/outputs

➢ Brief comparison



colab.research.google.com/drive/1snhcwW0PV6H9MS783lkvgN21w0fmtHzl?authuser=0#scrollTo=E97cOj9yhRmt
AI Assistant Coding.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Commands
+ Code
+ Text
Run all
Share
RAM
Disk

Task 3: Leap Year Validation Using Cursor AI

[2]

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

if year % 4 == 0:
    print("Leap Year")
else:
    print("Not a Leap Year")
```

```
Enter year: 2026
Not a Leap Year
```

[3]

```
# Improve leap year logic using proper conditions and readability.

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

if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
    print("Leap Year")
else:
    print("Not a Leap Year")
```

```
Enter year: 2024
Leap Year
```

Improve leap year logic using proper conditions and readability.

You're right, the current logic only checks for divisibility by 4. I will improve the leap year logic in the code to correctly account for all conditions, including divisibility by 100 and 400, for proper leap year determination.

Start coding or generate with AI.
Provide the improved leap year Python code
Explain the improved leap year logic
Test t
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes, so double-check it and use code with caution. Learn more
Variables
Terminal
12:06
Python 3

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

❖ Scenario:

Company policy requires developers to write logic before using AI.

❖ Task:

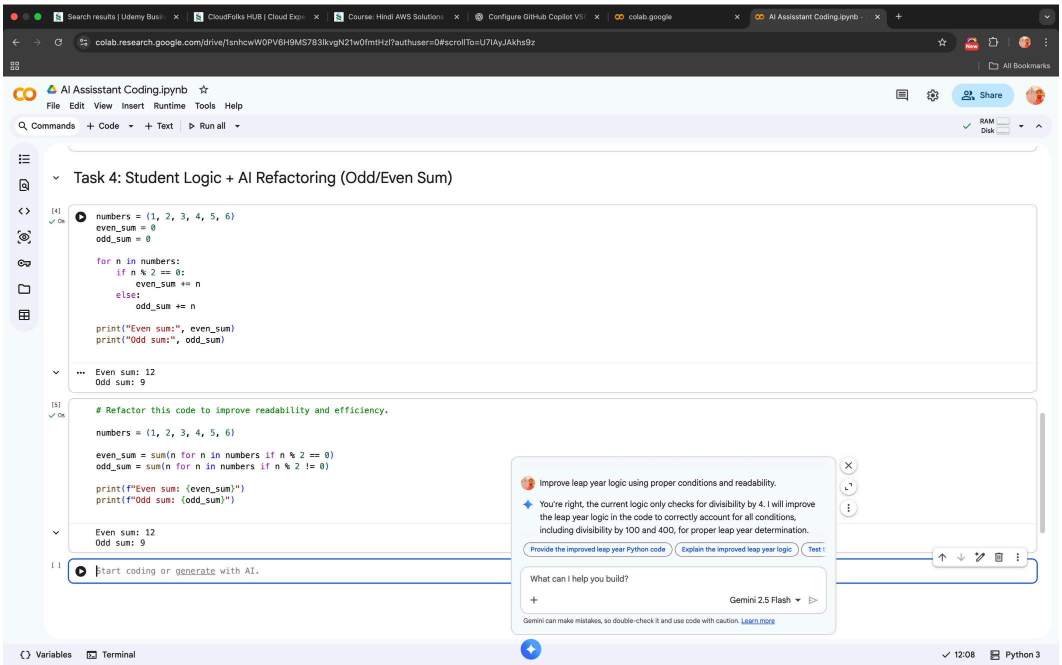
Write a Python program that calculates the sum of odd and even numbers in a tuple, then refactor it using any AI tool.

❖ Expected Output:

➢ Original code

➢ Refactored code

➢ Explanation of improvements


Search results | Udemy Busin
CloudFolks HUB | Cloud Expe
Course: Hindi AWS Solutions
Configure GitHub Copilot VS
colab.google
AI Assistant Coding.ipynb
colab.research.google.com/drive/1snhcwW0PV6H9MS783lkvgN21w0fmtHzl?authuser=0#scrollTo=U7lAyJAkhs9z
All Bookmarks
AI Assisstant Coding.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Commands
+ Code
+ Text
Run all
Share
RAM
Disk
Variables
Terminal
12:08
Python 3

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

```
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("Even sum:", even_sum)
print("Odd sum:", odd_sum)
```

```
Even sum: 12
Odd sum: 9
```

```
# 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"Even sum: {even_sum}")
print(f"Odd sum: {odd_sum}")
```

```
Even sum: 12
Odd sum: 9
```

```
Start coding or generate with AI.
```

Improve leap year logic using proper conditions and readability.

You're right, the current logic only checks for divisibility by 4. I will improve the leap year logic in the code to correctly account for all conditions, including divisibility by 100 and 400, for proper leap year determination.

Provide the improved leap year Python code
Explain the improved leap year logic
Test t
What can I help you build?
Gemini 2.5 Flash

Gemini can make mistakes, so double-check it and use code with caution. Learn more