# AI Assisted Coding - Lab Assignment Report

## Task Description #1

Use Google Gemini in Colab to write a function that reads a CSV file and calculates mean, min, max.

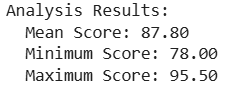
### Prompt:

Write a function in Python that reads a CSV file using Pandas and calculates the mean, min, and max for a specific numeric column. Include example usage with a sample CSV file and show the output.

### Code:

import pandas as pd  
  
def analyze\_csv(file\_path, column\_name):  
 try:  
df = pd.read\_csv(file\_path)  
 if column\_name not in df.columns:  
 print(f"Column '{column\_name}' not found.")  
 return  
mean\_val = df[column\_name].mean()  
min\_val = df[column\_name].min()  
max\_val = df[column\_name].max()  
 print(f"Analysis for column: {column\_name}")  
 print(f"Mean: {mean\_val}")  
 print(f"Min: {min\_val}")  
 print(f"Max: {max\_val}")  
 except FileNotFoundError:  
 print(f"File '{file\_path}' not found.")  
 except Exception as e:  
 print(f"Error: {e}")  
  
# Example usage  
sample\_data = "Name,Score\nAlice,85\nBob,90\nCharlie,78\nDavid,92\nEva,88"  
with open("sample.csv", "w") as f:  
f.write(sample\_data)  
analyze\_csv("sample.csv", "Score")

### Output:



## Task Description #2

Compare Gemini and Copilot outputs for a palindrome check function.

### Prompt:

Ask Gemini to generate a palindrome check function and ask GitHub Copilot to do the same. Compare outputs side-by-side and write observations.

### Example Gemini Code:

def is\_palindrome(s):  
 s = s.lower().replace(' ', '')  
 return s == s[::-1]

### Example Copilot Code:

def palindrome\_check(s):  
 s = ''.join(c.lower() for c in s if c.isalnum())  
 return s == s[::-1]

### Observations:

Gemini's version is simpler but doesn't remove punctuation. Copilot's version handles punctuation and spaces, making it more robust.

|  |  |  |
| --- | --- | --- |
| Feature | Gemini | Copilot |
| Code Simplicity | Simple and concise | Slightly more complex due to extra checks |
| Punctuation Handling | Does not remove punctuation | Removes punctuation using isalnum() |
| Space Handling | Removes spaces | Removes spaces and punctuation |
| Robustness | Good for basic strings | More robust for varied input formats |
| Output Accuracy | Accurate for simple cases | Accurate for both simple and complex cases |

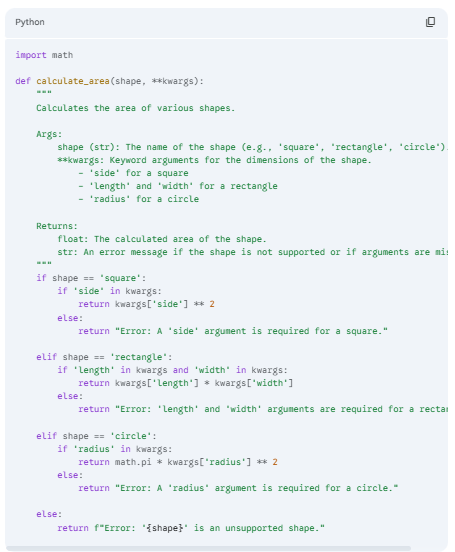
## Task Description #3

Ask Gemini to explain a Python function (to calculate area of various shapes) line by line.

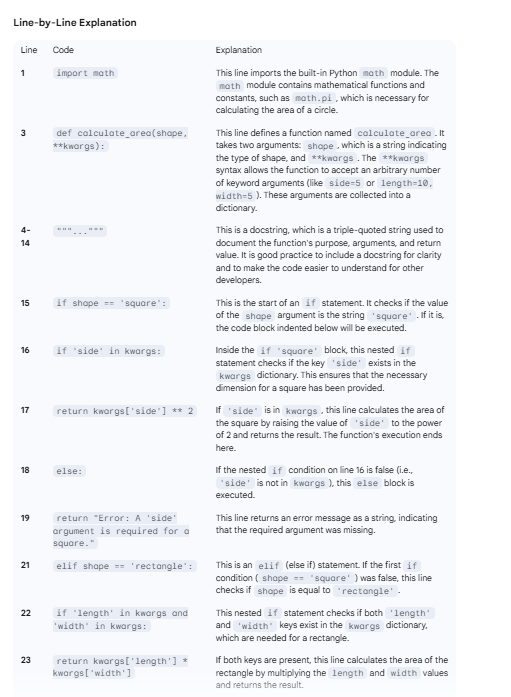
### Prompt:

Explain the given Python function line by line, describing how it calculates the area for shapes.

### Code:



### Explanation:

Gemini provided a detailed breakdown explaining each conditional branch, parameter usage, and return values.  


## Task Description #4

Install and configure Cursor AI. Use it to generate a Python function (sum of squares).

### Prompt:

Generate a Python function to calculate the sum of squares of numbers in a list using Cursor AI.

### Code:

def sum\_of\_squares(*numbers*):

"""    Calculate the sum of squares of numbers in a list.

    Args:

        numbers (list): A list of numbers (integers or floats)

    Returns:

        float: The sum of squares of all numbers in the list"""

*if* not numbers:

*return* 0.0

*return*sum(num \*\* 2 *for*num*in* numbers)

*# Example usage*

*if* \_\_name\_\_ == "\_\_main\_\_":

*# Test cases*

    test\_lists = [        [1, 2, 3, 4],        [2.5, 3.5], [0, 1, 2],   [-1, -2, -3],[],[5]]

    print("Testing sum\_of\_squares function:")

    print("-" \* 40)

*for*test\_list*in*test\_lists:

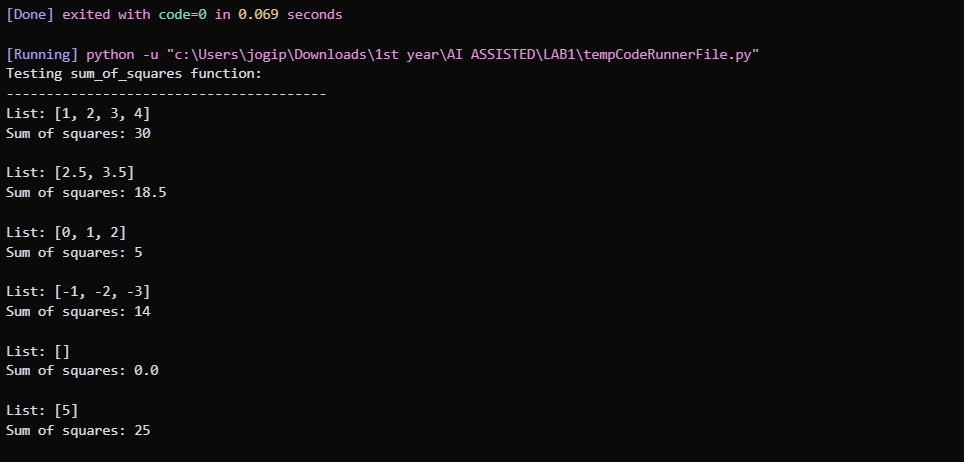
        result = sum\_of\_squares(test\_list)

        print(f"List: {test\_list}")

        print(f"Sum of squares: {result}")

        print()

### Output:



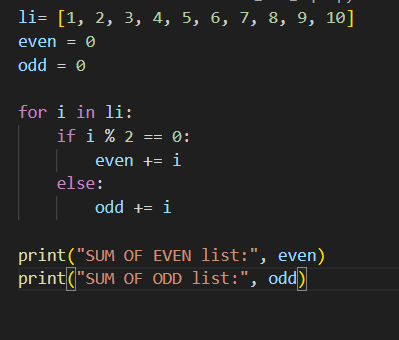
## Task Description #5

Write code to calculate sum of odd and even numbers in the list.

### Prompt:

Write a Python function that separates odd and even numbers in a list and sums them individually.

### Code:



### Output:

