

# Assignment-2.3

Name:K. Shiva Shankar

Id:2403a510e5

## Task Description#1

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

## Expected Output#1

- Functional code with output and screenshot

Prompt:generate code on function that reads a csv file and calculates mean,min,max

Code:

```
# Create a dummy CSV file for demonstration
data = {'col1': [1, 2, 3, 4, 5], 'col2': ['a', 'b', 'c', 'd', 'e'], 'col3': [10.5, 20.1, 5.9, 15.0, 25.3]}
dummy_df = pd.DataFrame(data)
dummy_file_path = 'dummy_data.csv'
dummy_df.to_csv(dummy_file_path, index=False)

# Use the function to analyze a column
analysis_results = analyze_csv_column(dummy_file_path, 'col1')

# Print the results
if analysis_results:
    print("Analysis Results:")
    print(f"Mean: {analysis_results['mean']}")
    print(f"Min: {analysis_results['min']}")
    print(f"Max: {analysis_results['max']}")

# Analyze a non-numeric column to show the warning
analysis_results_non_numeric = analyze_csv_column(dummy_file_path, 'col2')

# Analyze a column with mixed data types
analysis_results_mixed = analyze_csv_column(dummy_file_path, 'col3')

if analysis_results_mixed:
    print("\nAnalysis Results for mixed data column:")
    print(f"Mean: {analysis_results_mixed['mean']}")
    print(f"Min: {analysis_results_mixed['min']}")
    print(f"Max: {analysis_results_mixed['max']}")
```

Output:

Analysis Results:

Mean: 3.0

Min: 1

Max: 5

Warning: Column 'col2' contains no valid numeric data for calculations.

Analysis Results for mixed data column:

Mean: 15.36

Min: 5.9

Max: 25.3

## Task Description#2

- Compare Gemini and Copilot outputs for a palindrome check function.

## Expected Output#2

- Side-by-side comparison and observations

Prompt:generate code for palindrome check function.

Code:

```

def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# Test cases
test_numbers = [
    1,
    2,
    3,
    4,
    11,
    15,
    29
]

# Perform the check and store results
results = {}
for number in test_numbers:
    results[number] = is_prime(number)

# Display results side-by-side
print("Input Number | Is Prime?")
print("-----|-----")
for number, result in results.items():
    print(f"{number:<12} | {result}")

```

Output:

Input Number	Is Prime?
1	False
2	True
3	True
4	False
11	True
15	False
29	True

1. **Test with other numbers:** Provide more numbers to check if they are prime.
2. **Modify the function:** Suggest changes to the `is_prime` function.
3. **Explain the code:** Ask for an explanation of how the `is_prime` function works.
4. **Try a different problem:** Ask for a different type of code example.

### Task Description#3

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

### Expected Output#3

- Detailed explanation with code snippet.

Prompt:

Generate code Python function (to calculate area of various shapes)

Code:

```

import math

def area_circle(radius):
    """Calculates the area of a circle."""
    if radius < 0:
        return "Error: Radius cannot be negative"
    return math.pi * (radius ** 2)

def area_rectangle(length, width):
    """Calculates the area of a rectangle."""
    if length < 0 or width < 0:
        return "Error: Length and width cannot be negative"
    return length * width

def area_triangle(base, height):
    """Calculates the area of a triangle."""
    if base < 0 or height < 0:
        return "Error: Base and height cannot be negative"
    return 0.5 * base * height

# Test the functions and display results
print("--- Area Calculations ---")

# Circle
radius = 5
circle_area = area_circle(radius)
print(f"Area of circle with radius {radius}: {circle_area}")

# Rectangle
length = 10

# Rectangle
length = 10
width = 6
rectangle_area = area_rectangle(length, width)
print(f"Area of rectangle with length {length} and width {width}: {rectangle_area}")

# Triangle
base = 8
height = 4
triangle_area = area_triangle(base, height)
print(f"Area of triangle with base {base} and height {height}: {triangle_area}")

# Example with invalid input
invalid_radius = -2
invalid_circle_area = area_circle(invalid_radius)
print(f"Area of circle with radius {invalid_radius}: {invalid_circle_area}")

```

Output:

```
--- Area Calculations ---  
Area of circle with radius 5: 78.53981633974483  
Area of rectangle with length 10 and width 6: 60  
Area of triangle with base 8 and height 4: 16.0  
Area of circle with radius -2: Error: Radius cannot be negative
```

Explanation:

**Explanation:**

- We define three functions: `area_circle`, `area_rectangle`, and `area_triangle`.
- Each function takes the necessary dimensions as arguments and returns the calculated area.
- Basic error handling is included to check for negative input values.
- We then call these functions with example values and print the results.

Task Description#4

- Install and configure Cursor AI. Use it to generate a Python function(e.g.,sumof squares).

Expected Output#4

- Screenshots of working environments with few prompts to generate python code

Prompt:

Generate code on python function of sum of squares

Code:

```

def sum_of_squares(numbers):
    """Calculate the sum of squares of a list of numbers."""
    return sum(x**2 for x in numbers)

# Example usage
numbers = [1, 2, 3, 4, 5]
result = sum_of_squares(numbers)
print(f"Sum of squares of {numbers}: {result}")

# Alternative using list comprehension
def sum_of_squares_alt(numbers):
    return sum([x**2 for x in numbers])

# For a range of numbers
def sum_of_squares_range(n):
    """Calculate sum of squares from 1 to n."""
    return sum(i**2 for i in range(1, n + 1))

print(f"Sum of squares from 1 to 5: {sum_of_squares_range(5)}")

# Mathematical formula approach
def sum_of_squares_formula(n):
    """Calculate sum of squares using mathematical formula."""
    return n * (n + 1) * (2 * n + 1) // 6

# More examples
print(f"Sum of squares from 1 to 10: {sum_of_squares_range(10)}")
print(f"Sum of squares from 1 to 10 (formula): {sum_of_squares_formula(10)}")

# Test with different numbers
test_numbers = [2, 4, 6, 8, 10]
test_result = sum_of_squares(test_numbers)
print(f"Sum of squares of {test_numbers}: {test_result}")

```

Output:

Output when you run this code:

≡ text

```

Sum of squares of [1, 2, 3, 4, 5]: 55
Sum of squares from 1 to 5: 55
Sum of squares from 1 to 10: 385
Sum of squares from 1 to 10 (formula): 385
Sum of squares of [2, 4, 6, 8, 10]: 220

```

### Task Description#5

- Student need to write code to calculate sum of add number and even numbers in the list

### Expected Output#5

- Refactored code written by student with improved logic.

Prompt: write code to calculate sum of add number and even numbers in the list

Code:

```
def sum_odd_even(numbers):  
    sum_odd = 0  
    sum_even = 0  
    for number in numbers:  
        if number % 2 == 0:  
            sum_even += number  
        else:  
            sum_odd += number  
    return sum_odd, sum_even  
  
# Test list  
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
  
# Calculate sums  
odd_sum, even_sum = sum_odd_even(my_list)  
  
# Display results  
print(f"List: {my_list}")  
print(f"Sum of odd numbers: {odd_sum}")  
print(f"Sum of even numbers: {even_sum}")
```

Output:

```
List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
Sum of odd numbers: 25  
Sum of even numbers: 30
```

---