

# AI ASSISTED CODING

## ASSIGNMENT – 2.3

**G.Swamy**

**2403A51274**

### 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 that reads a CSV file and calculates mean,

min, max

Code –

```
1 import csv
2 import io
3 import statistics
4
5 def analyze_csv(file_content, column_name):
6     """
7     Reads CSV data from a string, analyzes a specified numerical column,
8     and returns the mean, minimum, and maximum values.
9
10    Args:
11        file_content (str): The content of the CSV file as a string.
12        column_name (str): The name of the column to analyze.
13
14    Returns:
15        dict or str: A dictionary with 'mean', 'min', and 'max' if successful,
16        otherwise a string explaining the error.
17    """
18    try:
19        # Use io.StringIO to treat the string content as a file
20        reader = csv.reader(io.StringIO(file_content))
21        header = next(reader)
22
23        if column_name not in header:
24            return f"Error: Column '{column_name}' not found in the CSV header."
25
26        column_index = header.index(column_name)
27
28        data_list = []
29        for row in reader:
30            try:
31                # Attempt to convert the data to a float
32                value = float(row[column_index])
33                data_list.append(value)
34            except (ValueError, IndexError):
35                # Skip rows where the value is not a number or the row is too short
36                continue
37
38        if not data_list:
39            return "Error: No numerical data found in the specified column."
40
41        # Calculate statistics
42        mean_value = statistics.mean(data_list)
```

```

5  def analyze_csv(file_content, column_name):
18  try:
42      mean_value = statistics.mean(data_list)
43      min_value = min(data_list)
44      max_value = max(data_list)
45
46      return {
47          'mean': mean_value,
48          'min': min_value,
49          'max': max_value
50      }
51
52  except Exception as e:
53      return f"An unexpected error occurred: {e}"
54
55  if __name__ == "__main__":
56      # Sample CSV data to test the function
57      sample_csv_data = """Name,Age,Score
58  Alice,30,88.5
59  Bob,25,75.0
60  Charlie,40,92.1
61  David,35,88.5
62  Eve,28,text_data
63  Frank,45,100.0"""
64
65      # Analyze the 'Score' column
66      results = analyze_csv(sample_csv_data, 'Score')
67
68      if isinstance(results, dict):
69          print("Analysis for the 'Score' column:")
70          print(f"Mean: {results['mean']:.2f}")
71          print(f"Min: {results['min']:.2f}")
72          print(f"Max: {results['max']:.2f}")
73      else:
74          print(results)
75
76      print("\n---")
77
78      # Example with a non-existent column
79      results_invalid = analyze_csv(sample_csv_data, 'Height')
80      print(results_invalid)
81
82
83
84      # Example with data containing only non-numeric values
85      sample_csv_data_empty = """Name,Age,Score
86  Alice,30,text_data
87  Bob,25,more_text
88  Charlie,40,na"""
89
90      results_empty = analyze_csv(sample_csv_data_empty, 'Score')
91      print(results_empty)
92

```

Output –

```

[1]  ✓ 0.0s
... Analysis for the 'Score' column:
Mean: 88.82
Min: 75.00
Max: 100.00

---
Error: Column 'Height' not found in the CSV header.

---
Error: No numerical data found in the specified column.

```

Task Description#2

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

Code by gemini –

Palindrome Check

```
1 import re
2
3 def is_palindrome(text):
4     """
5     Checks if a given string is a palindrome.
6
7     A palindrome is a word, phrase, or sequence that reads the same
8     backward as forward, ignoring case, punctuation, and spaces.
9
10    Args:
11        text (str): The string to check.
12
13    Returns:
14        bool: True if the string is a palindrome, False otherwise.
15    """
16    # Convert the string to lowercase and remove all non-alphanumeric characters.
17    # The re.sub() function is used to replace all characters that are not
18    # letters or numbers with an empty string.
19    cleaned_text = re.sub(r'^a-zA-Z0-9', '', text).lower()
20
21    # Compare the cleaned string with its reverse.
22    # The slice[::-1] is a simple way to reverse a string in Python.
23    return cleaned_text == cleaned_text[::-1]
24
25 if __name__ == "__main__":
26     # Test cases
27     test_strings = [
28         "A man, a plan, a canal: Panama",
29         "racecar",
30         "hello world",
31         "No 'x' in 'Nixon'",
32         "12321",
33         "Was it a car or a cat I saw?",
34         "not a palindrome"
35     ]
36
37     for s in test_strings:
38         if is_palindrome(s):
39             print(f'{s} is a palindrome.')
40         else:
41             print(f'{s} is not a palindrome.')
42
```

Output –

```
[2] ✓ 0.0s
... 'A man, a plan, a canal: Panama' is a palindrome.
    'racecar' is a palindrome.
    'hello world' is not a palindrome.
    'No 'x' in 'Nixon'' is a palindrome.
    '12321' is a palindrome.
    'Was it a car or a cat I saw?' is a palindrome.
    'not a palindrome' is not a palindrome.
```

Code by copilot–

```

import re

def is_palindrome(text):
    """
    Checks if a given string is a palindrome.
    A palindrome reads the same backward as forward,
    ignoring case, punctuation, and spaces.

    Args:
        text (str): The string to check.

    Returns:
        bool: True if the string is a palindrome, False otherwise.
    """
    # Remove non-alphanumeric characters and convert to lowercase
    cleaned_text = re.sub(r'^a-zA-Z0-9', '', text).lower()
    # Compare the cleaned string with its reverse
    return cleaned_text == cleaned_text[::-1]

# Example usage
if __name__ == "__main__":

```

```

# Example usage
if __name__ == "__main__":
    test_strings = [
        "A man, a plan, a canal: Panama",
        "racecar",
        "hello world",
        "No 'x' in Nixon",
        "12321",
        "Was it a car or a cat I saw?",
        "not a palindrome"
    ]

    for s in test_strings:
        result = is_palindrome(s)
        print(f"'{s}' is {'a' if result else 'not a'} palindrome.")

```

Output –

```

[3] ✓ 0.0s
... 'A man, a plan, a canal: Panama' is a palindrome.
    'racecar' is a palindrome.
    'hello world' is not a palindrome.
    'No 'x' in Nixon' is a palindrome.
    '12321' is a palindrome.
    'Was it a car or a cat I saw?' is a palindrome.
    'not a palindrome' is not a palindrome.

```

Code given by copilot is simpler than compared to gemini but both the outcomes are exactly the same

Task Description#3

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

prompt – explain a Python function (to calculate area of various shapes)

line by line.

Code by gemini–

Area Calculator

```

1  import math
2
3  def calculate_area(shape, **kwargs):
4      """
5      Calculates the area of a given shape.
6
7      Args:
8          shape (str): The name of the shape (e.g., 'circle', 'square', 'rectangle').
9          **kwargs: Keyword arguments for the dimensions of the shape.
10                 - For 'circle': 'radius' is required.
11                 - For 'square': 'side' is required.
12                 - For 'rectangle': 'length' and 'width' are required.
13
14      Returns:
15          float or str: The calculated area or an error message if the input is invalid.
16      """
17      # This line defines the function 'calculate_area' which takes 'shape' as a
18      # string and accepts any number of additional keyword arguments (**kwargs).
19
20      if shape.lower() == 'circle':
21          # This is a conditional statement that checks if the shape is a circle.
22          if 'radius' in kwargs:
23              # It then checks if the 'radius' keyword argument was provided.
24              radius = kwargs['radius']
25              # The value of the radius is assigned to a local variable.
26              if radius > 0:
27                  # This checks if the radius is a positive number, as a negative
28                  # or zero radius would not make sense for a circle's area.
29                  return math.pi * (radius ** 2)
30                  # If valid, it returns the area using the formula: pi * r^2.
31              else:
32                  return "Error: Radius must be a positive number."
33                  # If the radius is not positive, it returns an error message.
34          else:
35              return "Error: Please provide a radius for a circle."
36              # If the radius was not provided at all, it returns an error message.
37
38      elif shape.lower() == 'square':
39          # This conditional branch is executed if the shape is a square.
40          if 'side' in kwargs:
41              # It checks if the 'side' keyword argument was provided.
42              side = kwargs['side']

```

Output –

```

[4] ✓ 0.0s
... Area of a circle with radius 5: 78.53981633974483
Area of a square with side 4: 16
Area of a rectangle with length 6 and width 8: 48
Invalid shape: Error: Unsupported shape. Please use 'circle', 'square', or 'rectangle'.
Missing argument for circle: Error: Please provide a radius for a circle.

```

Code by copilot –



```

Python ^ Cop

def calculate_area(shape, **kwargs):
    shape = shape.lower()

    if shape == 'circle':
        if 'radius' in kwargs:
            radius = kwargs['radius']
            if radius > 0:
                return 3.14159 * radius ** 2
            else:
                return "Error: Radius must be a positive number."
        else:
            return "Error: Please provide a radius for a circle."

    elif shape == 'square':
        if 'side' in kwargs:
            side = kwargs['side']
            if side > 0:
                return side ** 2
            else:
                return "Error: Side length must be a positive number."
        else:
            return "Error: Please provide a side length for a square."

    elif shape == 'rectangle':
        if 'length' in kwargs and 'width' in kwargs:
            length = kwargs['length']
            width = kwargs['width']
            if length > 0 and width > 0:
                return length * width
            else:
                return "Error: Length and width must be positive numbers."
        else:
            return "Error: Please provide both length and width for a rectar

    else:
        return "Error: Unsupported shape. Please use 'circle', 'square', or

# Example usage
print("Circle:", calculate_area('circle', radius=5))
print("Square:", calculate_area('square', side=4))
print("Rectangle:", calculate_area('rectangle', length=6, width=8))

```

Output –

```

[5] ✓ 0.0s
... Circle: 78.53975
    Square: 16
    Rectangle: 48

```

Copilot is again shorter and easy to understand compared to gemini and again the output is simpler in copilot

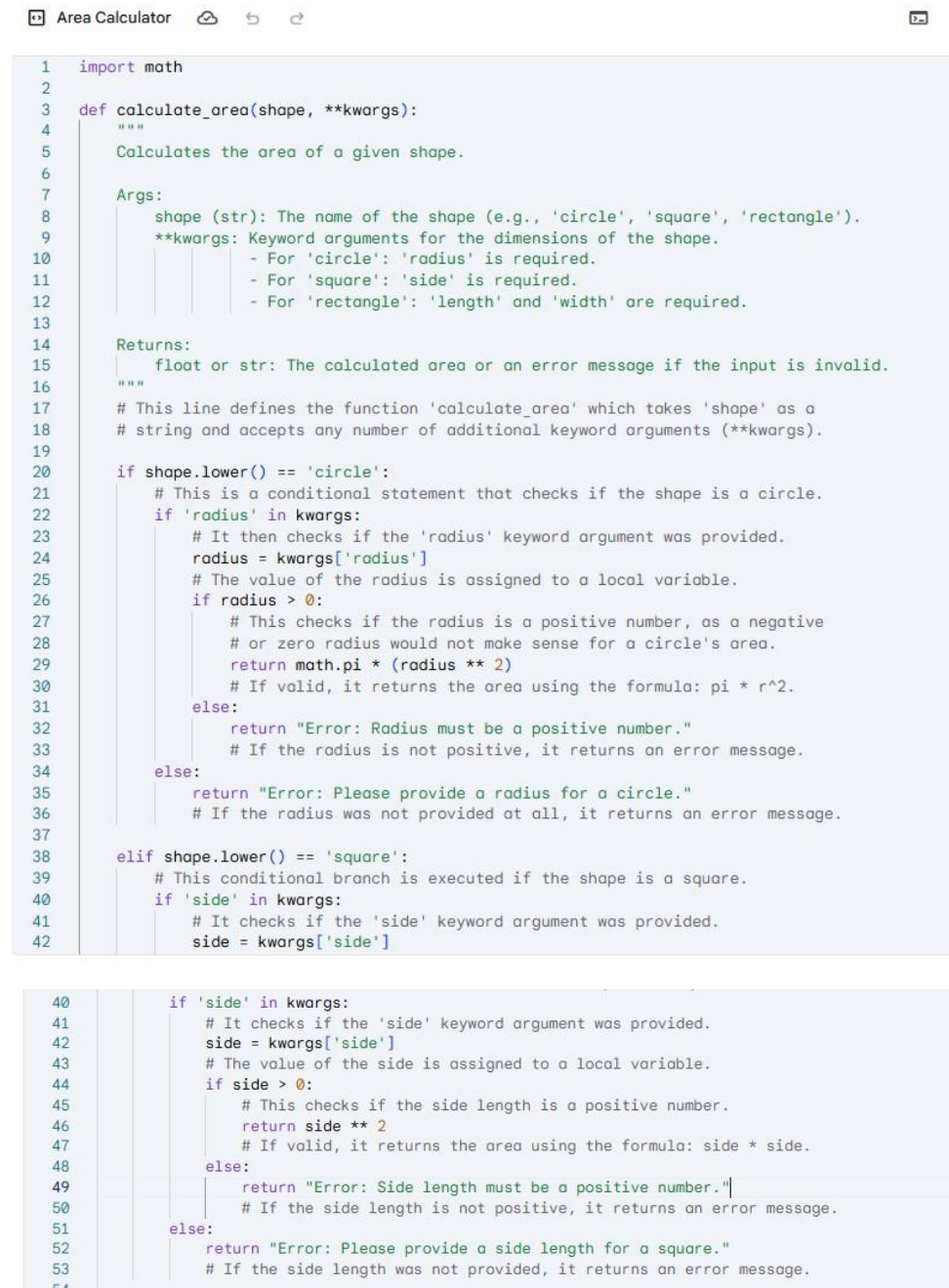
Task Description#4

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

prompt – generate a

Python function

Code –



```

1  import math
2
3  def calculate_area(shape, **kwargs):
4      """
5      Calculates the area of a given shape.
6
7      Args:
8          shape (str): The name of the shape (e.g., 'circle', 'square', 'rectangle').
9          **kwargs: Keyword arguments for the dimensions of the shape.
10                 - For 'circle': 'radius' is required.
11                 - For 'square': 'side' is required.
12                 - For 'rectangle': 'length' and 'width' are required.
13
14      Returns:
15          float or str: The calculated area or an error message if the input is invalid.
16      """
17      # This line defines the function 'calculate_area' which takes 'shape' as a
18      # string and accepts any number of additional keyword arguments (**kwargs).
19
20      if shape.lower() == 'circle':
21          # This is a conditional statement that checks if the shape is a circle.
22          if 'radius' in kwargs:
23              # It then checks if the 'radius' keyword argument was provided.
24              radius = kwargs['radius']
25              # The value of the radius is assigned to a local variable.
26              if radius > 0:
27                  # This checks if the radius is a positive number, as a negative
28                  # or zero radius would not make sense for a circle's area.
29                  return math.pi * (radius ** 2)
30                  # If valid, it returns the area using the formula: pi * r^2.
31              else:
32                  return "Error: Radius must be a positive number."
33                  # If the radius is not positive, it returns an error message.
34          else:
35              return "Error: Please provide a radius for a circle."
36              # If the radius was not provided at all, it returns an error message.
37
38      elif shape.lower() == 'square':
39          # This conditional branch is executed if the shape is a square.
40          if 'side' in kwargs:
41              # It checks if the 'side' keyword argument was provided.
42              side = kwargs['side']
43              # The value of the side is assigned to a local variable.
44              if side > 0:
45                  # This checks if the side length is a positive number.
46                  return side ** 2
47                  # If valid, it returns the area using the formula: side * side.
48              else:
49                  return "Error: Side length must be a positive number."
50                  # If the side length is not positive, it returns an error message.
51          else:
52              return "Error: Please provide a side length for a square."
53              # If the side length was not provided, it returns an error message.
54

```



```

54
55     elif shape.lower() == 'rectangle':
56         # This branch is for when the shape is a rectangle.
57         if 'length' in kwargs and 'width' in kwargs:
58             # It checks if both 'length' and 'width' keyword arguments were provided.
59             length = kwargs['length']
60             width = kwargs['width']
61             # The values are assigned to local variables.
62             if length > 0 and width > 0:
63                 # This checks if both dimensions are positive numbers.
64                 return length * width
65                 # If valid, it returns the area using the formula: length * width.
66             else:
67                 return "Error: Length and width must be positive numbers."
68                 # If either dimension is not positive, it returns an error message.
69         else:
70             return "Error: Please provide both length and width for a rectangle."
71             # If both dimensions were not provided, it returns an error message.
72
73     else:
74         # This 'else' block handles any shape name that isn't recognized.
75         return "Error: Unsupported shape. Please use 'circle', 'square', or 'rectangle'."
76         # It returns a general error message.
77
78 # Example usage of the function
79 print("Area of a circle with radius 5:", calculate_area('circle', radius=5))
80 print("Area of a square with side 4:", calculate_area('square', side=4))
81 print("Area of a rectangle with length 6 and width 8:", calculate_area('rectangle', length=6, width=8))
82
83 # Examples of error handling
84 print("Invalid shape:", calculate_area('triangle', base=5, height=10))
85 print("Missing argument for circle:", calculate_area('circle', diameter=10))
86

```

Output –

```

[6] ✓ 0.0s
... Area of a circle with radius 5: 78.53981633974483
Area of a square with side 4: 16
Area of a rectangle with length 6 and width 8: 48
Invalid shape: Error: Unsupported shape. Please use 'circle', 'square', or 'rectangle'.
Missing argument for circle: Error: Please provide a radius for a circle.

```

## Task Description#5

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

prompt – write code to calculate sum of add number and even numbers

in the list.

Code –

```

1  def calculate_odd_even_sums(numbers):
2      """
3      Calculates the sum of odd and even numbers in a list.
4
5      Args:
6      |   numbers (list): A list of numerical values (integers or floats).
7
8      Returns:
9      |   dict: A dictionary containing the sums of odd and even numbers.
10     |       Keys are 'even_sum' and 'odd_sum'.
11     """
12     even_sum = 0
13     odd_sum = 0
14
15     for number in numbers:
16         # Check if the number is an integer to use the modulo operator.
17         # This handles cases where the list might contain floats, though
18         # the concept of 'odd'/'even' is typically for integers.
19         if isinstance(number, int):
20             if number % 2 == 0:
21                 even_sum += number
22             else:
23                 odd_sum += number
24
25     return {'even_sum': even_sum, 'odd_sum': odd_sum}
26
27 # Example usage
28 my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
29 sums = calculate_odd_even_sums(my_list)
30
31 print("Original List:", my_list)
32 print("Sum of even numbers:", sums['even_sum'])
33 print("Sum of odd numbers:", sums['odd_sum'])
34
35 # Another example with different numbers
36 another_list = [15, 22, 37, 40, 51, 6]
37 another_sums = calculate_odd_even_sums(another_list)
38
39 print("\nOriginal List:", another_list)
40 print("Sum of even numbers:", another_sums['even_sum'])
41 print("Sum of odd numbers:", another_sums['odd_sum'])
42

```

Output –

```

[7]  ✓ 0.0s
...  Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      Sum of even numbers: 30
      Sum of odd numbers: 25

      Original List: [15, 22, 37, 40, 51, 6]
      Sum of even numbers: 68
      Sum of odd numbers: 103

```