AI ASSISTED CODING

ASSIGNMENT – 2.3

NAME:Sathwik

H.NO:2403A51273

BATCH:12

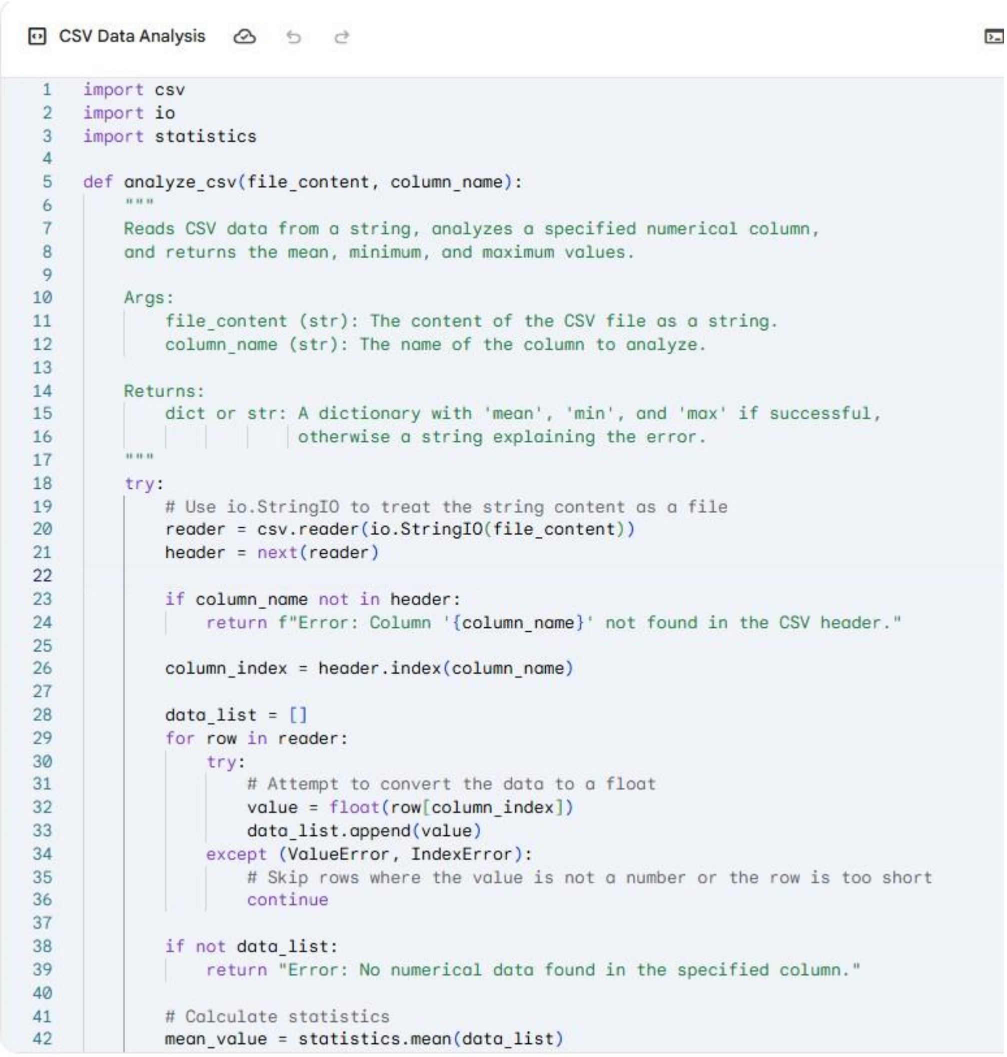
Task Description#1

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

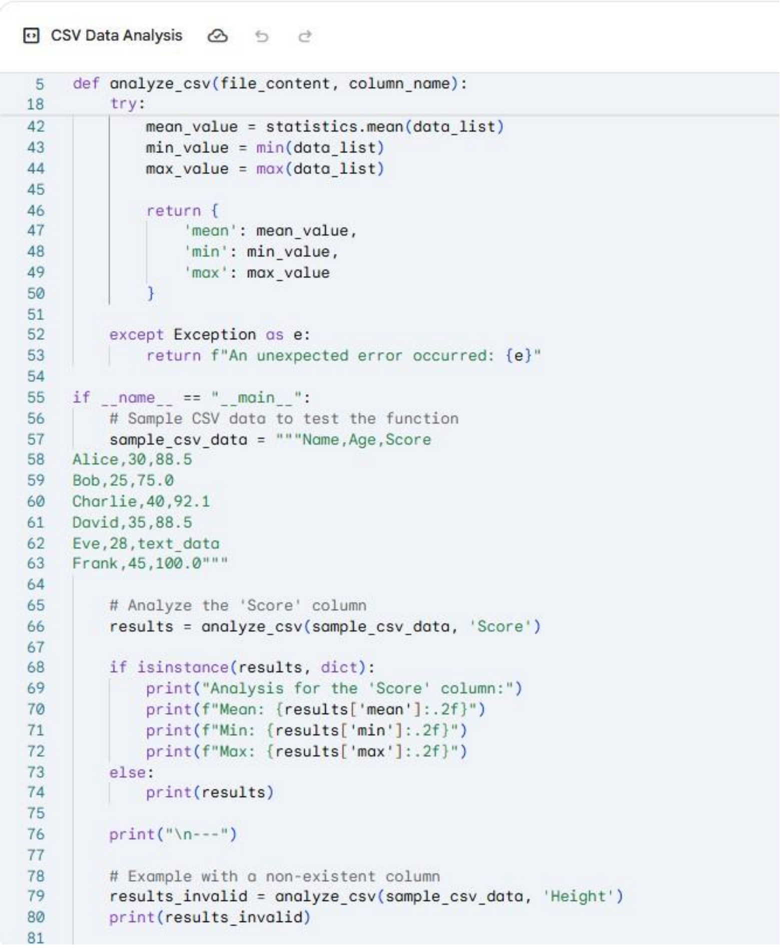
prompt – write a function that reads a CSV file and calculates mean,

min, max

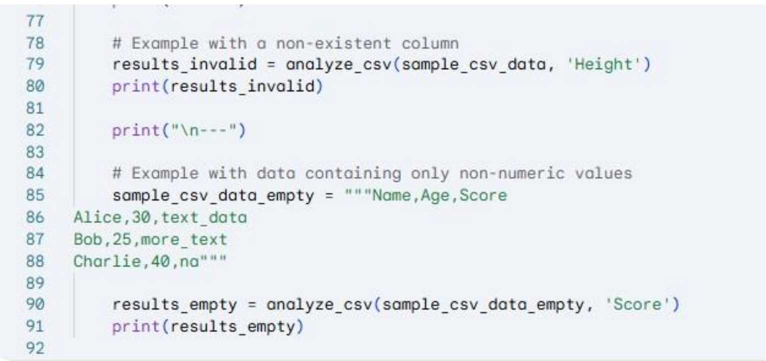
Code –



```python
import csv
import io
import statistics

def analyze_csv(file_content, column_name):
    """
    Reads CSV data from a string, analyzes a specified numerical column,
    and returns the mean, minimum, and maximum values.

    Args:
        file_content (str): The content of the CSV file as a string.
        column_name (str): The name of the column to analyze.

    Returns:
        dict or str: A dictionary with 'mean', 'min', and 'max' if successful,
                     otherwise a string explaining the error.
    """
    try:
        # Use io.StringIO to treat the string content as a file
        reader = csv.reader(io.StringIO(file_content))
        header = next(reader)

        if column_name not in header:
            return f"Error: Column '{column_name}' not found in the CSV header."

        column_index = header.index(column_name)

        data_list = []
        for row in reader:
            try:
                # Attempt to convert the data to a float
                value = float(row[column_index])
                data_list.append(value)
            except (ValueError, IndexError):
                # Skip rows where the value is not a number or the row is too short
                continue

        if not data_list:
            return "Error: No numerical data found in the specified column."

        # Calculate statistics
        mean_value = statistics.mean(data_list)
```

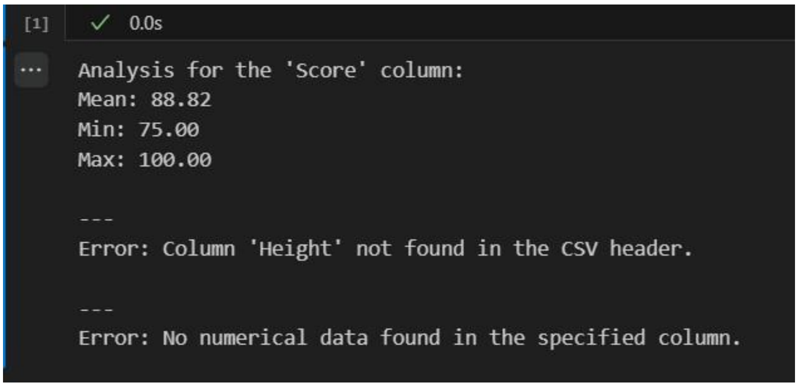


```python
def analyze_csv(file_content, column_name):
    try:
        mean_value = statistics.mean(data_list)
        min_value = min(data_list)
        max_value = max(data_list)

        return {
            'mean': mean_value,
            'min': min_value,
            'max': max_value
        }

    except Exception as e:
        return f"An unexpected error occurred: {e}"

if __name__ == "__main__":
    # Sample CSV data to test the function
    sample_csv_data = """Name,Age,Score
Alice,30,88.5
Bob,25,75.0
Charlie,40,92.1
David,35,88.5
Eve,28,text_data
Frank,45,100.0"""

    # Analyze the 'Score' column
    results = analyze_csv(sample_csv_data, 'Score')

    if isinstance(results, dict):
        print("Analysis for the 'Score' column:")
        print(f"Mean: {results['mean']:.2f}")
        print(f"Min: {results['min']:.2f}")
        print(f"Max: {results['max']:.2f}")
    else:
        print(results)

    print("\n---")

    # Example with a non-existent column
    results_invalid = analyze_csv(sample_csv_data, 'Height')
    print(results_invalid)

```



```python

    # Example with a non-existent column
    results_invalid = analyze_csv(sample_csv_data, 'Height')
    print(results_invalid)

    print("\n---")

    # Example with data containing only non-numeric values
    sample_csv_data_empty = """Name,Age,Score
Alice,30,text_data
Bob,25,more_text
Charlie,40,na"""

    results_empty = analyze_csv(sample_csv_data_empty, 'Score')
    print(results_empty)

```

Output –

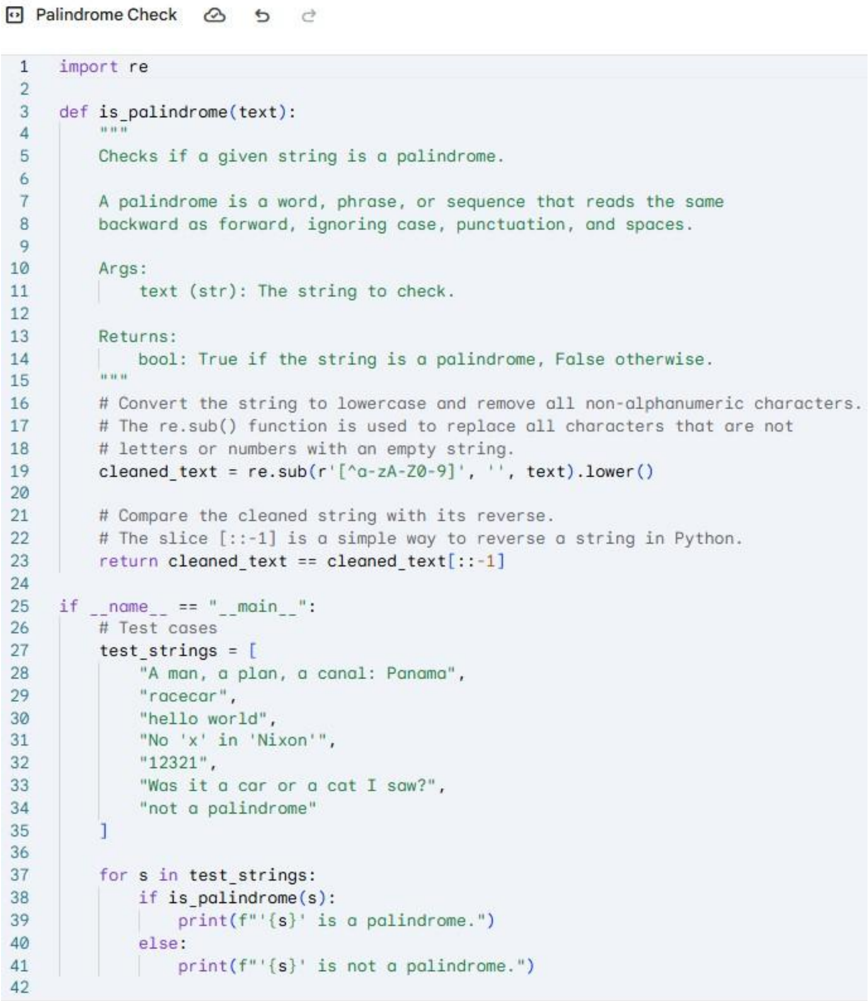


```
Analysis for the 'Score' column:
Mean: 88.82
Min: 75.00
Max: 100.00

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

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

Task Description#2

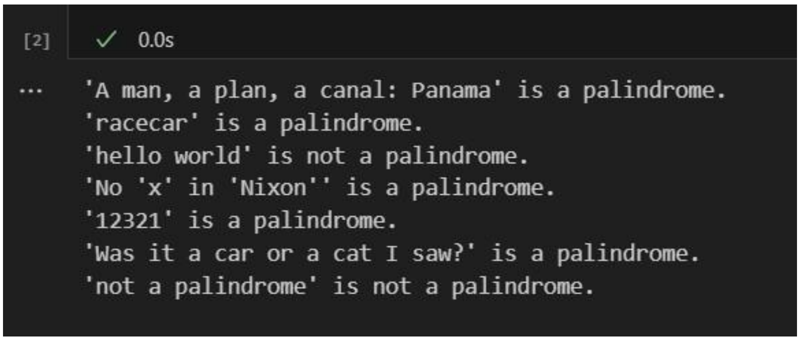
- Compare Gemini and Copilot outputs for a palindrome check function. prompt – generate a palindrome check function.

Code by gemini –



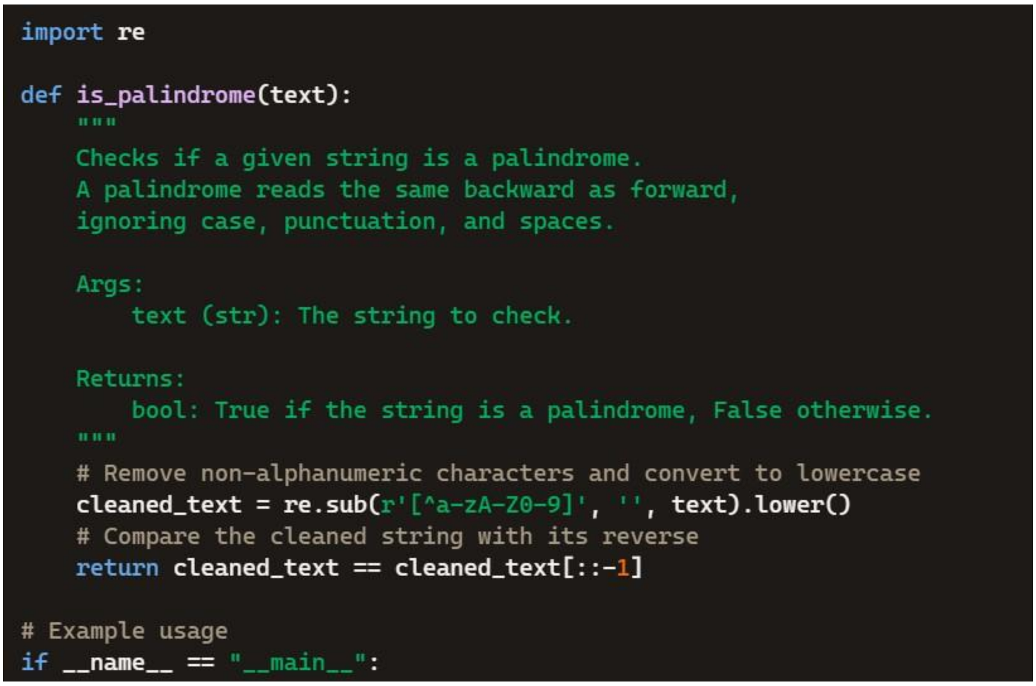
```python
import re

def is_palindrome(text):
    """
    Checks if a given string is a palindrome.

    A palindrome is a word, phrase, or sequence that 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.
    """
    # Convert the string to lowercase and remove all non-alphanumeric characters.
    # The re.sub() function is used to replace all characters that are not
    # letters or numbers with an empty string.
    cleaned_text = re.sub(r'[^a-zA-Z0-9]', '', text).lower()

    # Compare the cleaned string with its reverse.
    # The slice [::-1] is a simple way to reverse a string in Python.
    return cleaned_text == cleaned_text[::-1]

if __name__ == "__main__":
    # Test cases
    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:
        if is_palindrome(s):
            print(f"'{s}' is a palindrome.")
        else:
            print(f"'{s}' is not a palindrome.")
```

Output –

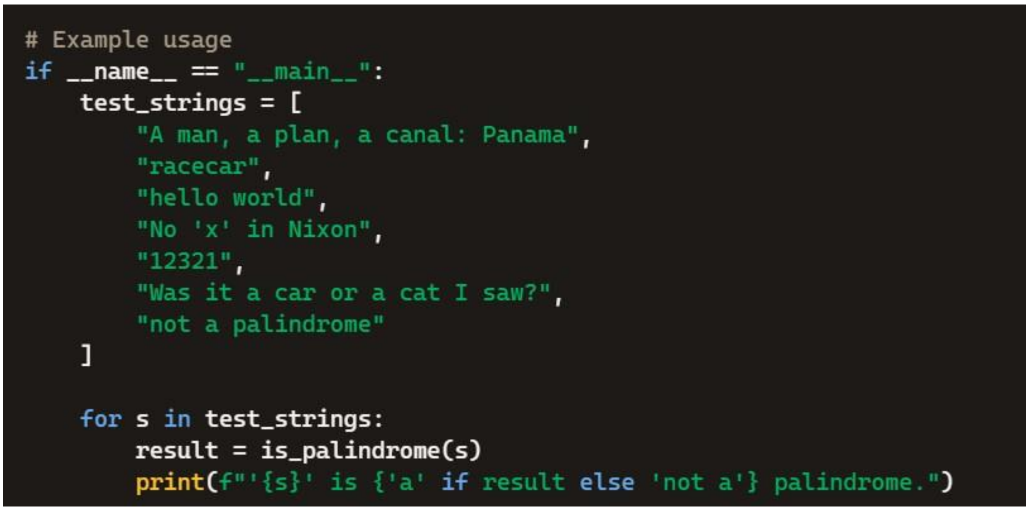


```
'A man, a plan, a canal: Panama' is a palindrome.
'racecar' is a palindrome.
'hello world' is not a palindrome.
'No 'x' in 'Nixon'' is a palindrome.
'12321' is a palindrome.
'Was it a car or a cat I saw?' is a palindrome.
'not a palindrome' is not a palindrome.
```

Code by copilot–

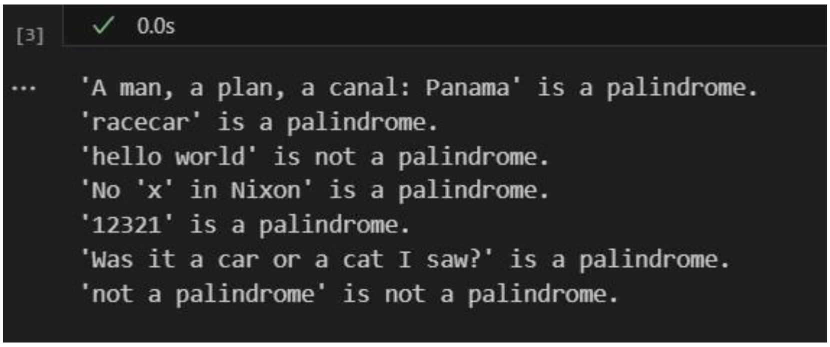


```
import re

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 –



```
'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

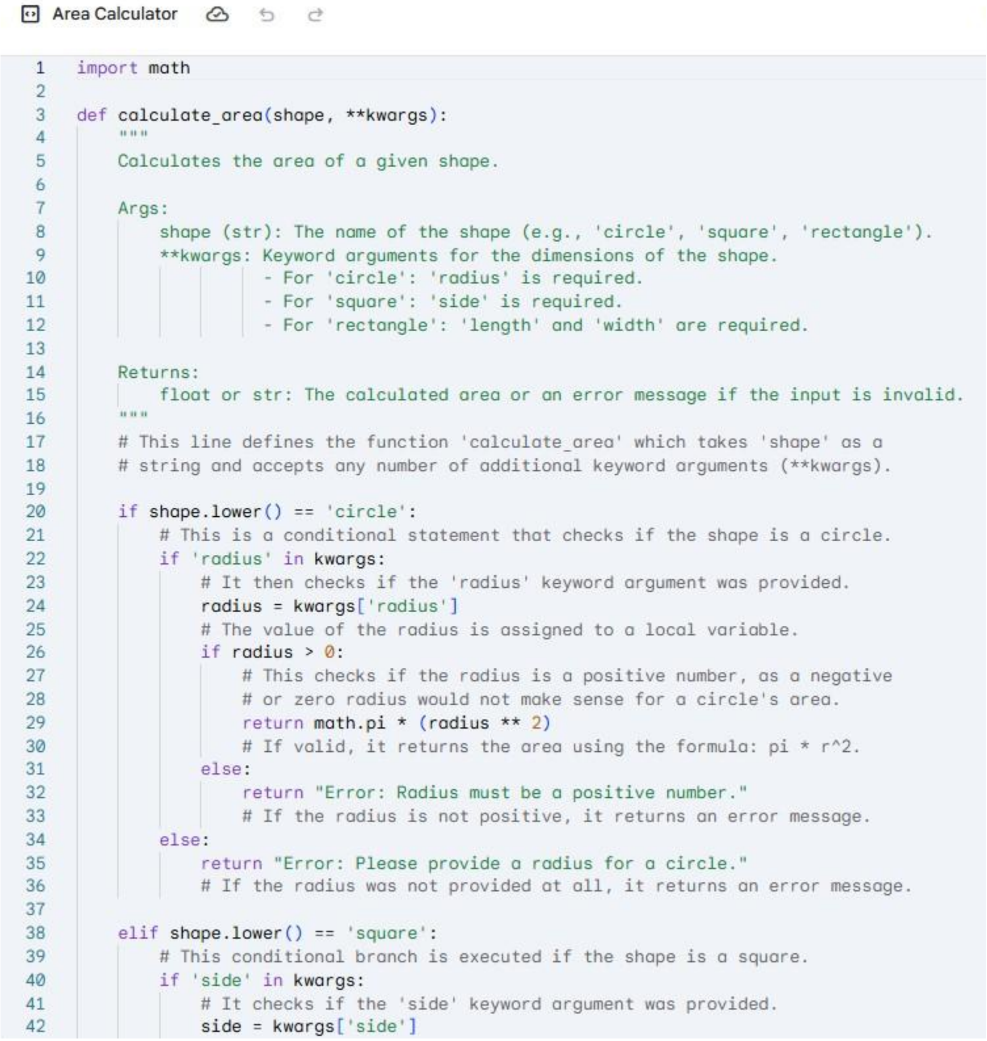
Task Description#3

- 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)

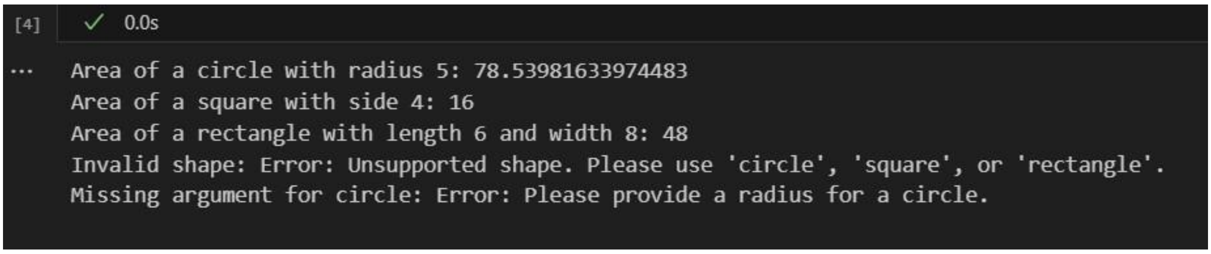
line by line.

Code by gemini–



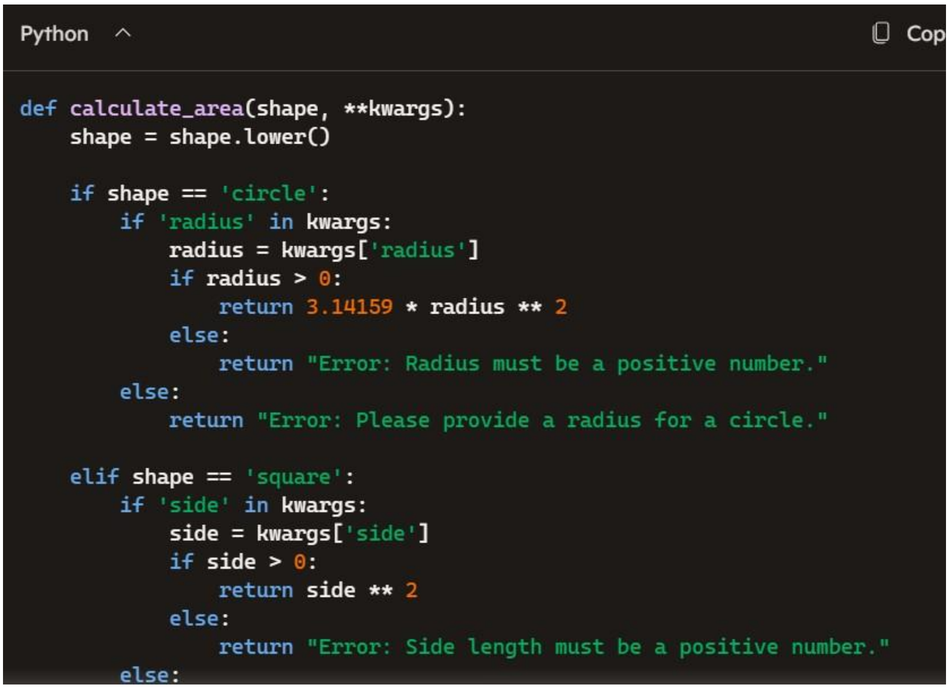
```python
import math

def calculate_area(shape, **kwargs):
    """
    Calculates the area of a given shape.

    Args:
        shape (str): The name of the shape (e.g., 'circle', 'square', 'rectangle').
        **kwargs: Keyword arguments for the dimensions of the shape.
                  - For 'circle': 'radius' is required.
                  - For 'square': 'side' is required.
                  - For 'rectangle': 'length' and 'width' are required.

    Returns:
        float or str: The calculated area or an error message if the input is invalid.
    """
    # This line defines the function 'calculate_area' which takes 'shape' as a
    # string and accepts any number of additional keyword arguments (**kwargs).

    if shape.lower() == 'circle':
        # This is a conditional statement that checks if the shape is a circle.
        if 'radius' in kwargs:
            # It then checks if the 'radius' keyword argument was provided.
            radius = kwargs['radius']
            # The value of the radius is assigned to a local variable.
            if radius > 0:
                # This checks if the radius is a positive number, as a negative
                # or zero radius would not make sense for a circle's area.
                return math.pi * (radius ** 2)
                # If valid, it returns the area using the formula: pi * r^2.
            else:
                return "Error: Radius must be a positive number."
                # If the radius is not positive, it returns an error message.
        else:
            return "Error: Please provide a radius for a circle."
            # If the radius was not provided at all, it returns an error message.

    elif shape.lower() == 'square':
        # This conditional branch is executed if the shape is a square.
        if 'side' in kwargs:
            # It checks if the 'side' keyword argument was provided.
            side = kwargs['side']
```

Output –

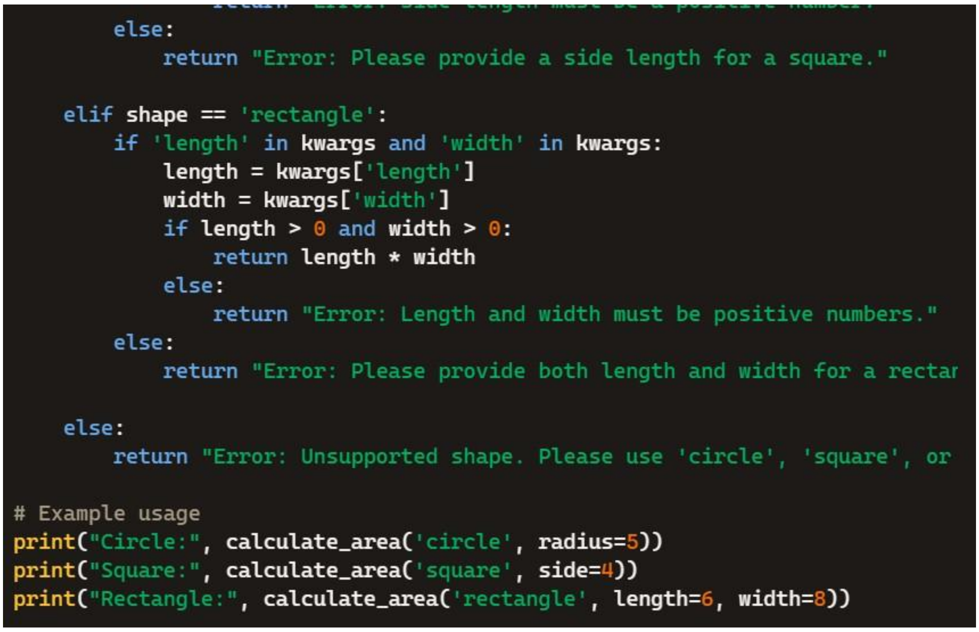


```
Area of a circle with radius 5: 78.53981633974483
Area of a square with side 4: 16
Area of a rectangle with length 6 and width 8: 48
Invalid shape: Error: Unsupported shape. Please use 'circle', 'square', or 'rectangle'.
Missing argument for circle: Error: Please provide a radius for a circle.
```

Code by copilot –

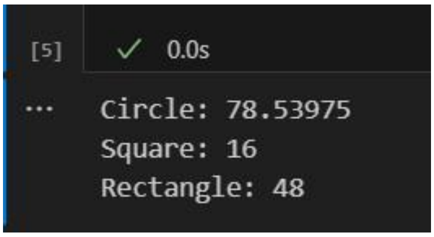


```
def calculate_area(shape, **kwargs):
    shape = shape.lower()

    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:
```



```
        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 –



```
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

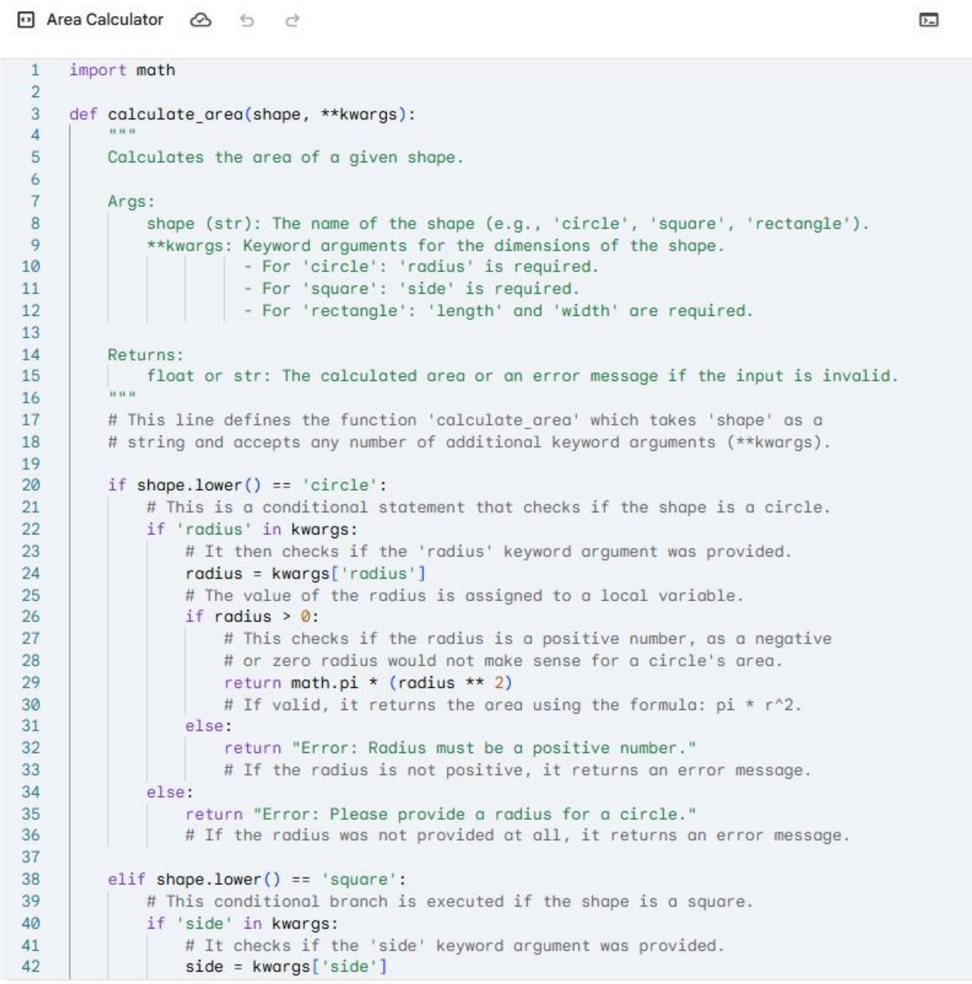
Task Description#4

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

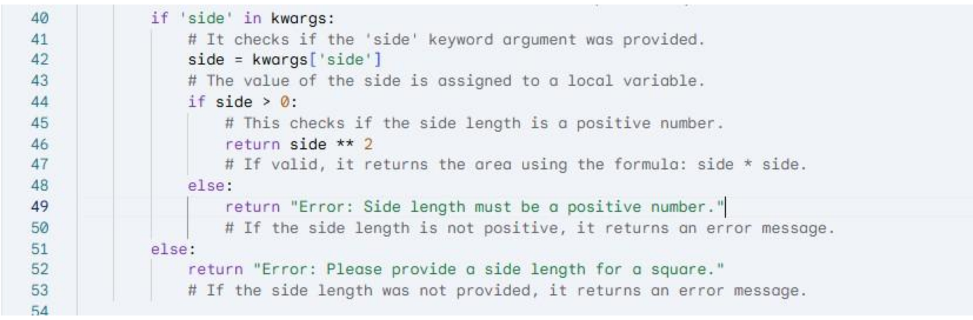
prompt – generate a

Python function

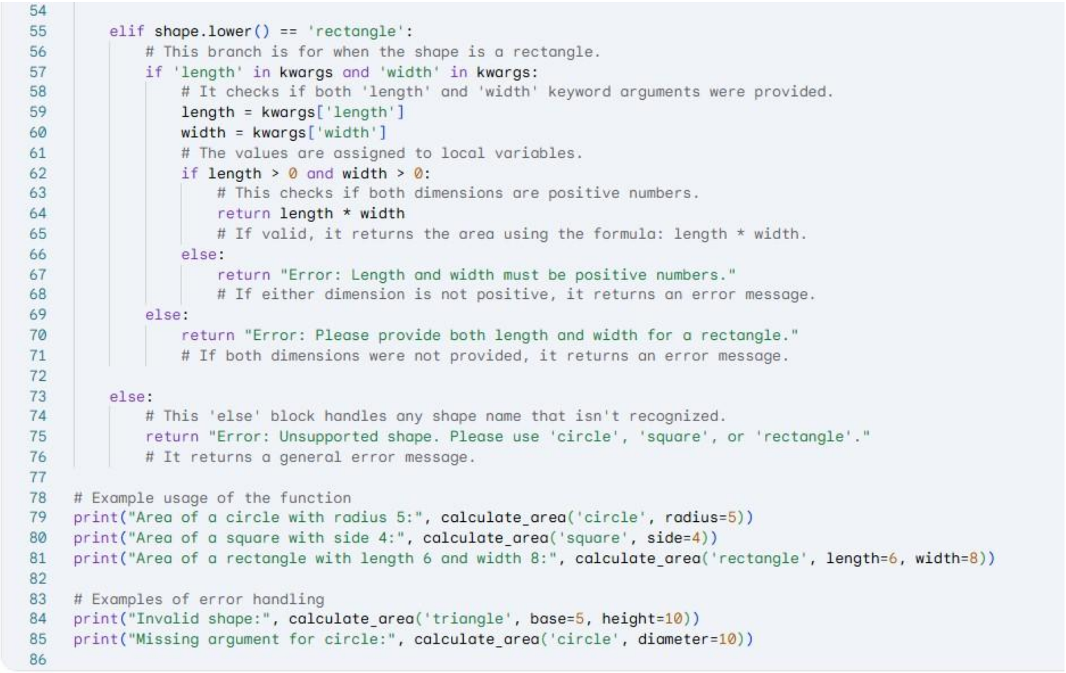
Code –



```python
import math

def calculate_area(shape, **kwargs):
    """
    Calculates the area of a given shape.

    Args:
        shape (str): The name of the shape (e.g., 'circle', 'square', 'rectangle').
        **kwargs: Keyword arguments for the dimensions of the shape.
                  - For 'circle': 'radius' is required.
                  - For 'square': 'side' is required.
                  - For 'rectangle': 'length' and 'width' are required.

    Returns:
        float or str: The calculated area or an error message if the input is invalid.
    """
    # This line defines the function 'calculate_area' which takes 'shape' as a
    # string and accepts any number of additional keyword arguments (**kwargs).

    if shape.lower() == 'circle':
        # This is a conditional statement that checks if the shape is a circle.
        if 'radius' in kwargs:
            # It then checks if the 'radius' keyword argument was provided.
            radius = kwargs['radius']
            # The value of the radius is assigned to a local variable.
            if radius > 0:
                # This checks if the radius is a positive number, as a negative
                # or zero radius would not make sense for a circle's area.
                return math.pi * (radius ** 2)
                # If valid, it returns the area using the formula: pi * r^2.
            else:
                return "Error: Radius must be a positive number."
                # If the radius is not positive, it returns an error message.
        else:
            return "Error: Please provide a radius for a circle."
            # If the radius was not provided at all, it returns an error message.

    elif shape.lower() == 'square':
        # This conditional branch is executed if the shape is a square.
        if 'side' in kwargs:
            # It checks if the 'side' keyword argument was provided.
            side = kwargs['side']
```

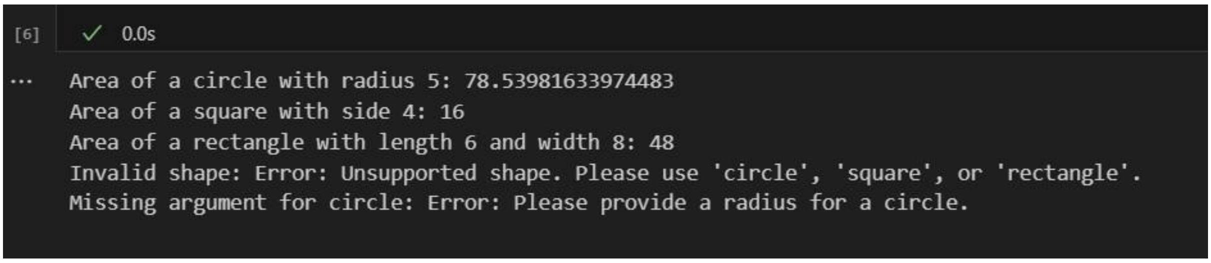


```python
        if 'side' in kwargs:
            # It checks if the 'side' keyword argument was provided.
            side = kwargs['side']
            # The value of the side is assigned to a local variable.
            if side > 0:
                # This checks if the side length is a positive number.
                return side ** 2
                # If valid, it returns the area using the formula: side * side.
            else:
                return "Error: Side length must be a positive number."
                # If the side length is not positive, it returns an error message.
        else:
            return "Error: Please provide a side length for a square."
            # If the side length was not provided, it returns an error message.

```



```python

    elif shape.lower() == 'rectangle':
        # This branch is for when the shape is a rectangle.
        if 'length' in kwargs and 'width' in kwargs:
            # It checks if both 'length' and 'width' keyword arguments were provided.
            length = kwargs['length']
            width = kwargs['width']
            # The values are assigned to local variables.
            if length > 0 and width > 0:
                # This checks if both dimensions are positive numbers.
                return length * width
                # If valid, it returns the area using the formula: length * width.
            else:
                return "Error: Length and width must be positive numbers."
                # If either dimension is not positive, it returns an error message.
        else:
            return "Error: Please provide both length and width for a rectangle."
            # If both dimensions were not provided, it returns an error message.

    else:
        # This 'else' block handles any shape name that isn't recognized.
        return "Error: Unsupported shape. Please use 'circle', 'square', or 'rectangle'."
        # It returns a general error message.

# Example usage of the function
print("Area of a circle with radius 5:", calculate_area('circle', radius=5))
print("Area of a square with side 4:", calculate_area('square', side=4))
print("Area of a rectangle with length 6 and width 8:", calculate_area('rectangle', length=6, width=8))

# Examples of error handling
print("Invalid shape:", calculate_area('triangle', base=5, height=10))
print("Missing argument for circle:", calculate_area('circle', diameter=10))

```

Output –



```
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.
```

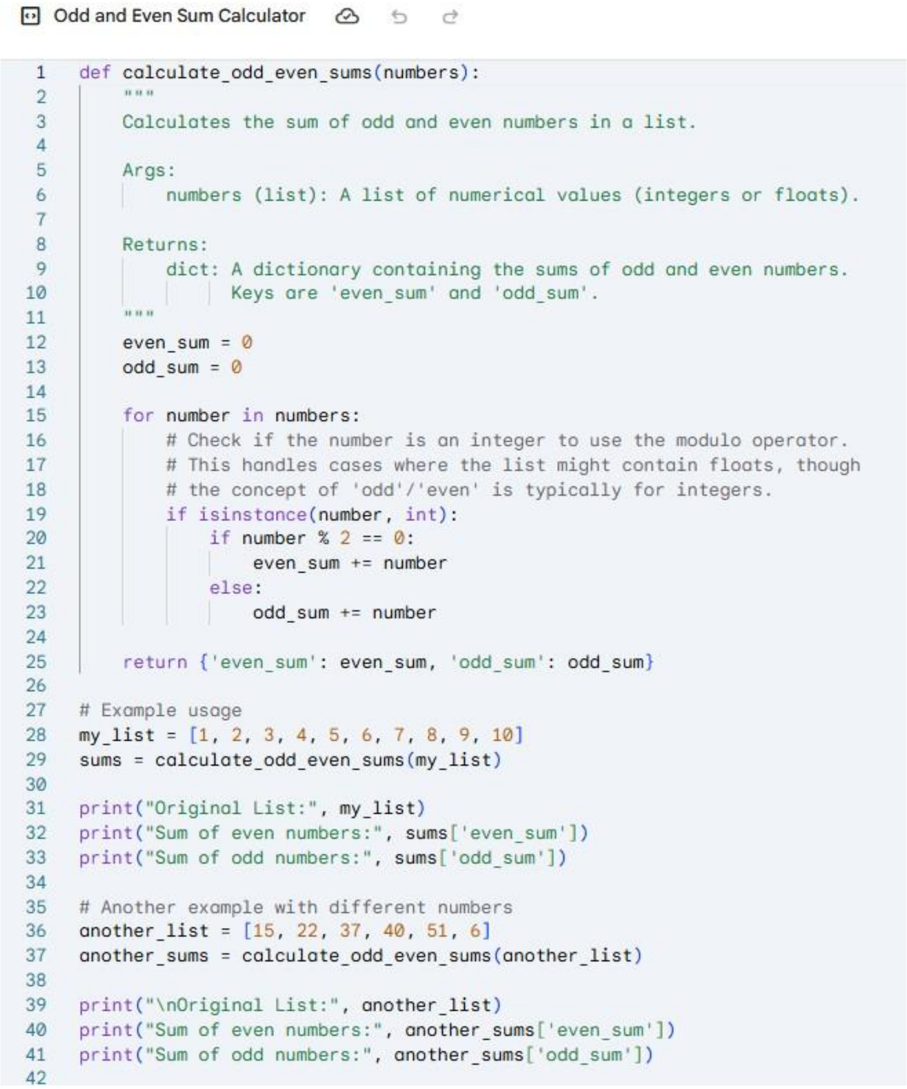
Task Description#5

● 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

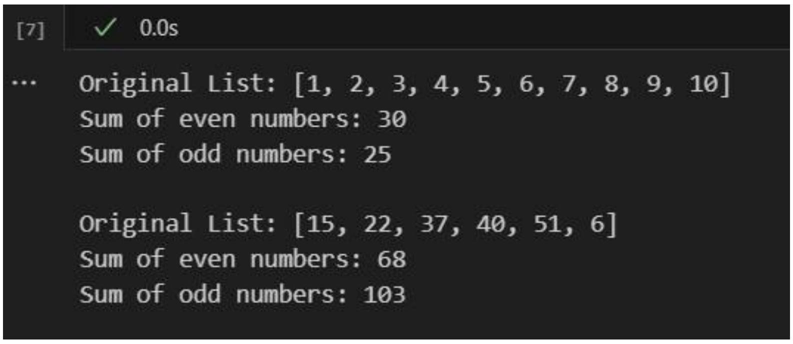
in the list.

Code –



```python
def calculate_odd_even_sums(numbers):
    """
    Calculates the sum of odd and even numbers in a list.

    Args:
        numbers (list): A list of numerical values (integers or floats).

    Returns:
        dict: A dictionary containing the sums of odd and even numbers.
              Keys are 'even_sum' and 'odd_sum'.
    """
    even_sum = 0
    odd_sum = 0

    for number in numbers:
        # Check if the number is an integer to use the modulo operator.
        # This handles cases where the list might contain floats, though
        # the concept of 'odd'/'even' is typically for integers.
        if isinstance(number, int):
            if number % 2 == 0:
                even_sum += number
            else:
                odd_sum += number

    return {'even_sum': even_sum, 'odd_sum': odd_sum}

# Example usage
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sums = calculate_odd_even_sums(my_list)

print("Original List:", my_list)
print("Sum of even numbers:", sums['even_sum'])
print("Sum of odd numbers:", sums['odd_sum'])

# Another example with different numbers
another_list = [15, 22, 37, 40, 51, 6]
another_sums = calculate_odd_even_sums(another_list)

print("\nOriginal List:", another_list)
print("Sum of even numbers:", another_sums['even_sum'])
print("Sum of odd numbers:", another_sums['odd_sum'])
```

Output –



```
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
```