Name: Shryitha suddala

Hall ticket:2303A51637

Batch:29

Assessment: 2.5

Task 1:

Prompt:

1. Program to calculate sum of even and odd numbers (Legacy Code)
2. Program to calculate sum of even and odd numbers (Refactored Code)

Code & output:

Justification:

The original code correctly computes the sum of odd and even numbers but uses a repetitive loop and indexing, which reduces readability and efficiency. The refactored version uses Python's built-in sum() function with conditional expressions, making the code shorter, clearer, and easier to maintain. This improvement follows modern coding practices and reduces the chances of errors while producing the same output

Task 2:

Prompt: Program to calculate area of different shapes (Legacy Code)

Code & output:

```
#Program to calculate area of different shapes
shape = input("Enter the shape (circle, rectangle, triangle): ")
if shape.lower() == "circle":
    radius = float(input("Enter the radius of the circle: "))
    area = 3.14159 * radius * radius
    print("Area of the circle:", area)

elif shape.lower() == "rectangle":
    length = float(input("Enter the length of the rectangle: "))
    breadth = float(input("Enter the breadth of the rectangle: "))
    area = length * breadth
    print("Area of the rectangle:", area)

elif shape.lower() == "triangle":
    base = float(input("Enter the base of the triangle: "))
    height = float(input("Enter the height of the triangle: "))
    area = 0.5 * base * height
    print("Area of the triangle:", area)

else:
    print("Invalid shape entered")
```

```
Enter the shape (circle, rectangle, triangle): Circle
Enter the radius of the circle: 2
Area of the circle: 12.56636
```

Justification:

This program is designed to calculate the area of different geometric shapes such as a circle, rectangle, and triangle based on user input. It uses conditional statements to identify the selected shape and then applies the appropriate mathematical formula. Separate inputs are taken for each shape to ensure correct dimensions are used in the calculation. The step-by-step structure makes the program easy to understand, especially for beginners, and helps in clearly demonstrating how decision-making works in Python. It also includes validation for invalid shape input, which improves reliability. Overall, the program effectively meets the objective of computing areas while maintaining clarity, correctness, and ease of use.

Task 3:

Prompt: Write a Python program to find the sum of even and odd numbers in a list.

Code:

```python
# Write a Python program to find the sum of even and odd numbers in a list.
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum = sum(x for x in numbers if x % 2 == 0)
odd_sum = sum(x for x in numbers if x % 2 != 0)
print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
```

Output:

```
PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/Task3.5.py
Sum of even numbers: 30
Sum of odd numbers: 25
PS C:\Users\shryi\Downloads\AI>
```

Justification:

This program calculates the sum of even and odd numbers using Python's built-in sum() function combined with conditional expressions. By eliminating explicit loops, the code becomes more concise, readable, and efficient. This approach follows modern Python practices, reduces complexity, and makes the program easier to maintain while producing accurate results

Prompt: Write an optimized and readable Python program to calculate the sum of even and odd numbers in a list

Code:

```python
#Write an optimized and readable Python program to calculate the sum of even and odd numbers in a list.
def sum_even_odd(numbers):
    even_sum = 0
    odd_sum = 0

    for number in numbers:
        if number % 2 == 0:
            even_sum += number
        else:
            odd_sum += number

    return even_sum, odd_sum
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_even_odd(numbers)
print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
```

Output:

```
PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/Task3.5.py
Sum of even numbers: 30
Sum of odd numbers: 25
PS C:\Users\shryi\Downloads\AI>
```

Justification:

This program defines a function sum_even_odd() that separates the logic for calculating the sum of even and odd numbers from the main execution flow. The function iterates through the given list of numbers and uses the modulus operator to check each number's parity. Even numbers are added to even_sum, and odd numbers are added to odd_sum.By using a function and returning the results, the program improves modularity, reusability, and clarity. The main program simply provides the input list and prints the returned values, making the code well-structured, easy to test, and suitable for reuse in larger applications

Prompt: Write a beginner-friendly Python program with user input to calculate the sum of even and odd numbers in a list.

Code:

```
# Write a beginner-friendly Python program with user input to calculate the sum of even and odd numbers in a list.
def sum_even_odd_user_input():
    numbers = input("Enter numbers separated by spaces: ")
    numbers_list = list(map(int, numbers.split()))

    even_sum = 0
    odd_sum = 0

    for number in numbers_list:
        if number % 2 == 0:
            even_sum += number
        else:
            odd_sum += number

    print("Sum of even numbers:", even_sum)
    print("Sum of odd numbers:", odd_sum)
sum_even_odd_user_input()
```

Output:

```
PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/Task3.5.py
Enter numbers separated by spaces: 1 2 3 4 5 6 7 8 9 10
Sum of even numbers: 30
Sum of odd numbers: 25
PS C:\Users\shryi\Downloads\AI>
```

Justification:

This program accepts a list of numbers from the user and calculates the sum of even and odd values separately. It converts the input string into a list of integers, then iterates through each number to check whether it is even or odd using the modulus operator. Based on this condition, the program updates the corresponding sum.The use of a function improves modularity and reusability, making the code easier to maintain and test. Clear variable names and a simple loop structure enhance readability, which is especially helpful for beginners. Overall, the program effectively fulfills the requirement of separating and summing even and odd numbers in a structured and reliable manner.

Task 4:

Based on my experience using **Gemini, GitHub Copilot, and Cursor AI,** all three tools are useful for coding support, but they differ in usability and code quality.

Gemini is very helpful for explanations and understanding concepts. It provides clear, beginner-friendly descriptions along with simple and structured code, which makes it suitable for learning and onboarding new developers.

GitHub Copilot is best suited for real-time coding. It integrates directly into the code editor and gives fast, context-aware suggestions. Its code quality is generally high and follows practical programming patterns, making it effective for increasing developer productivity.

Cursor AI stands out for experimentation and refactoring. It responds well to detailed prompts, produces multiple variations of code, and is especially useful for prompt-sensitivity testing, optimization, and improving legacy code.

Overall, **Gemini** is ideal for learning and explanation, **Copilot** is strongest for live coding assistance, and **Cursor AI** is most effective for code refinement and prompt-driven development.