Assignment–2.5

2303A51655

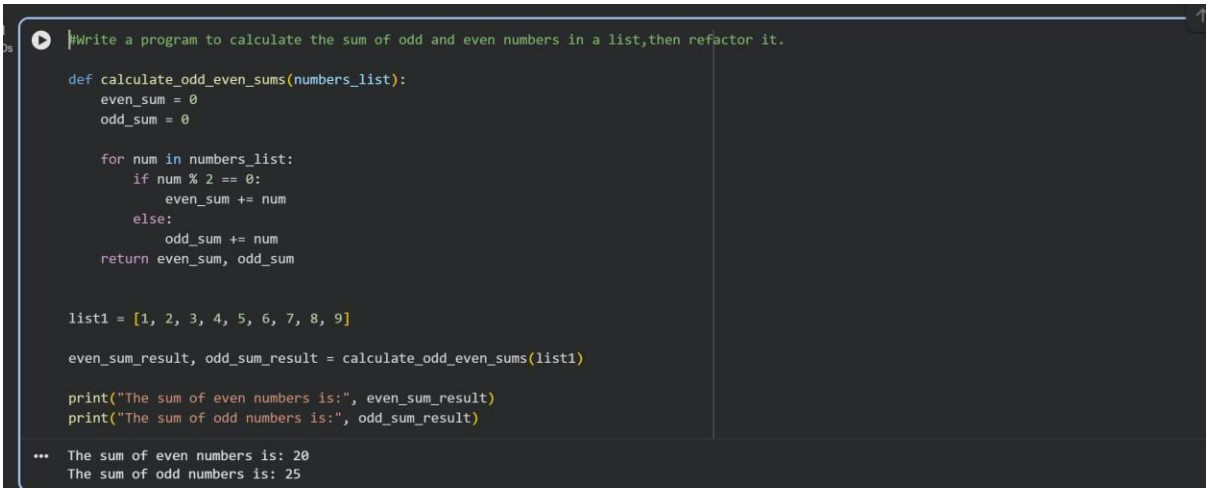
BatchNo:-29

Task1:RefactoringOdd/EvenLogic

PROMPT:-

Writeaprogramtocalculatethesumofoddandevennumbers in a list, then refactor it.

CODEandOUTPUT:



```
#Write a program to calculate the sum of odd and even numbers in a list,then refactor it.

def calculate_odd_even_sums(numbers_list):
    even_sum = 0
    odd_sum = 0

    for num in numbers_list:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum


list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

even_sum_result, odd_sum_result = calculate_odd_even_sums(list1)

print("The sum of even numbers is:", even_sum_result)
print("The sum of odd numbers is:", odd_sum_result)
```

```
The sum of even numbers is: 20
The sum of odd numbers is: 25
```

Justification:-

In this task, I was given a scenario where I had to improve an old or legacy Python program. The program was used to calculate the sum of odd and even numbersfromalist.Initially,theprogramused afor loopwith index valuesand separate variables to store the sums of odd and even numbers. Although the program worked correctly,thecodewas longand notveryclean. With thehelp of AI tools like Gemini and Cursor AI, I refactored the code to make it shorter and more readable. The improved version used Python's built-in sum()function and conditional expressions to calculate the sums directly. This refactored code is easier to understand and maintain. It also follows better coding practices. From this task, I understood how AI tools can help in improving existing code and making it more efficient

Task2:AreaCalculationExplanation:-

PROMPT:-

Explainafunctionthatcalculatestheareaof different shapes.

CODEandOUTPUT:

```python
#Explain a function that calculates the area of different shapes.
def calculate_area(shape, dimensions):
    """
    Calculate the area of different shapes based on the provided dimensions.

    Parameters:
    shape (str): The type of shape ('circle', 'rectangle', 'triangle').
    dimensions (tuple): The dimensions required to calculate the area.
                        For 'circle', provide (radius,).
                        For 'rectangle', provide (length, width).
                        For 'triangle', provide (base, height).

    Returns:
    float: The area of the shape.
    """
    if shape == 'circle':
        radius = dimensions[0]
        area = 3.14159 * (radius ** 2)
    elif shape == 'rectangle':
        length, width = dimensions
        area = length * width
    elif shape == 'triangle':
        base, height = dimensions
        area = 0.5 * base * height
    else:
        raise ValueError("Unsupported shape type. Supported shapes are: circle, rectangle, triangle.")

    return area
# Example usage:
```

```python
circle_area = calculate_area('circle', (5,))
rectangle_area = calculate_area('rectangle', (4, 6))
triangle_area = calculate_area('triangle', (3, 7))
print(f"Circle Area: {circle_area}")
print(f"Rectangle Area: {rectangle_area}")
print(f"Triangle Area: {triangle_area}")
# This function `calculate_area` takes in two parameters:
# 1. `shape`: a string that specifies the type of shape for which we want
#    to calculate the area. It can be 'circle', 'rectangle', or 'triangle'.
# 2. `dimensions`: a tuple that contains the necessary dimensions to compute
#    the area of the specified shape.
# The function checks the type of shape and uses the appropriate formula to
# calculate the area based on the provided dimensions. It then returns the
# calculated area as a float. If an unsupported shape is provided, it raises
# a ValueError with a message indicating the supported shapes.
# The example usage demonstrates how to call the function for each shape
# and prints the calculated areas.
```

```
Circle Area: 78.53975
Rectangle Area: 24
Triangle Area: 10.5
```

Justification:

Thesecondtaskfocusedon explainingaPythonfunctionthat calculatestheareaofdifferent shapes.Thescenariowasthatajuniordeveloperisjoiningtheteamandneedstounderstand howthefunction works.Iwrote afunction that calculates theareaof acircle, rectangle, and triangleusingconditionalstatements.IthenusedGoogleGeminitoexplainthecode.Gemini explained the purpose of each parameter, how the if and elif conditions work, and how the correct formula is selected based on the shape.The explanation given byGemini was simple and easy to understand. This made it helpful for beginners. From this task, I learned that Geminiisveryusefulforlearningandteachingpurposesbecauseitexplainscodelogicclearly.

Task3:PromptSensitivityExperiment:-

PROMPT :

Writeaprogramtocalculatethesumofoddandevennumbersinalist, then refactor it.

CODE:

```python
#Write a program to calculate the sum of odd and even numbers in a list,then refactor it.

def sum_odd_even(numbers):
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
numbers_list = [10, 15, 20, 25, 30, 35]
even_sum, odd_sum = sum_odd_even(numbers_list)
print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
#Write a program to calculate the sum of odd and even numbers in a list

def calculate_sums(numbers):
    even_sum = sum(num for num in numbers if num % 2 == 0)
    odd_sum = sum(num for num in numbers if num % 2 != 0)
    return even_sum, odd_sum
numbers_list = [10, 15, 20, 25, 30, 35]
even_sum, odd_sum = calculate_sums(numbers_list)
print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
```

OUTPUT:

```
Sum of even numbers: 60
Sum of odd numbers: 75
Sum of even numbers: 60
Sum of odd numbers: 75
PS D:\AI>
```

Justification:-

In this task, I tested how Cursor AI reacts to different prompts for the same programming problem. The problem was to write a function to check whether a number is prime. I gave CursorAIdifferentpromptssuchasabasicprompt,anoptimizedprompt,andapromptasking for explanation. Each prompt resulted in different versions of the code. When I asked for an optimized solution,the AI generated a faster and more efficient algorithm. When I asked for explanation, the AI added comments and made the code easier to understand. This experiment helped me understand that AI tools are very sensitive to the way prompts are written.Clear anddetailed promptsgive betterresults.Thistask showedme the importance of writing good prompts while using AI tools.

Task4:ToolComparisonReflection:-

Based on myexperiencewithGemini,GitHubCopilot,andCursorAI,allthreeareeffectiveAI tools for programming, but they vary in terms of ease of use and the quality of code they generate.Geminiisparticularlystronginexplainingconceptsandprovidingguidance.Itoffers simple, well-organized code along with clear, beginner-friendly explanations, which makes it very suitable for learning and for helping new developers understand programming tasks. GitHub Copilot is mainly focused on real-time development support. Since it works directly insidethecodeeditor,itprovidesquick,context-basedcodesuggestions.Thecodeit

generates is usually practical and reliable, making it a powerful tool for improving coding speed and overallproductivity.Cursor AI isespecially usefulfor experimenting withprompts andimprovingexistingcode.Ithandlesdetailedinstructionswell,generatesmultipleversions of solutions, and is effective for refactoring, optimization, and working with legacy code. In conclusion, Gemini is best for learning and concept clarification, Copilot excels in live coding assistance, and Cursor AIis most suitable for refining code and prompt-driven development.