AI-ASSISTED CODING Assignment-2.5

Name: CH. MANMITH VARDHAN

Hall ticket: 2303A51321

Task – 1: Refactoring Odd/Even Logic (List Version)

```python
def separate_odd_even(numbers):
    odd_numbers = []
    even_numbers = []
    for num in numbers:
        if num % 2 == 0:
            even_numbers.append(num)
        else:
            odd_numbers.append(num)
    return odd_numbers, even_numbers

# Example usage:
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
odds, evens = separate_odd_even(my_list)

print(f"Original list: {my_list}")
print(f"Odd numbers: {odds}")
print(f"Even numbers: {evens}")
```

Output:

```
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Odd numbers: [1, 3, 5, 7, 9]
Even numbers: [2, 4, 6, 8, 10]
```

Explanation:

The program separates a list of numbers into odd and even numbers using a user-defined function. It iterates through each element and checks divisibility by 2 to classify numbers correctly. The output clearly displays the original list along with separated odd and even lists. Using a function improves code readability and reusability. The logic is easy to understand and debug. This approach is efficient and suitable for real-world applications where list processing is required.

Task -2: Area Calculation Explanation

```python
def calculate_rectangle_area(length, width):
    """
    Calculates the area of a rectangle.

    Args:
        length (float): The length of the rectangle.
        width (float): The width of the rectangle.

    Returns:
        float: The area of the rectangle.
    """
    return length * width

# Example usage:
rect_length = 10
rect_width = 5
area = calculate_rectangle_area(rect_length, rect_width)

print(f"The length of the rectangle is: {rect_length}")
print(f"The width of the rectangle is: {rect_width}")
print(f"The area of the rectangle is: {area}")

# Explanation:
# The area of a rectangle is calculated by multiplying its length by its width.
# In this example, length = 10 and width = 5, so the area = 10 * 5 = 50.
```

Output:

```
The length of the rectangle is: 10
The width of the rectangle is: 5
The area of the rectangle is: 50
```

Explanation:

The program calculates the area of a rectangle using a user-defined function. It takes length and width as parameters and returns their product as the area. Using a function makes the code modular and easy to reuse. The docstring clearly explains the purpose, inputs, and output of the function. The output correctly shows the length, width, and calculated area. This approach improves readability, maintainability, and is suitable for realworld applications.

Task-3: Prompt Sensitivity Experiment

```
prompt_list = [
    "Reverse a given string.",
    "Write a Python function to reverse a string using slicing.",
    "Reverse the string 'Python', the output should be 'nohtyP'.",
    "Reverse a string. Ensure your solution handles empty strings and strings containing spaces.",
    "Implement a function in Python that reverses a string without using any built-in reverse functions or slicing. Your solution should handle strings with mixed case characters."
]

for i, prompt in enumerate(prompt_list):
    print(f"Prompt {i+1}: {prompt}")
```

Output:

```
Prompt 1: Reverse a given string.
Prompt 2: Write a Python function to reverse a string using slicing.
Prompt 3: Reverse the string 'Python', the output should be 'nohtyP'.
Prompt 4: Reverse a string. Ensure your solution handles empty strings and strings containing spaces.
Prompt 5: Implement a function in Python that reverses a string without using any built-in reverse functions or slicing. Your solution should handle strings with mixed case characters.
```

Explanation:

The prompts show how changing instructions affects the AI-generated solution. Simple prompts produce basic string reversal, while detailed prompts generate more robust and structured code. Prompts that mention slicing lead to concise solutions using [::-1]. More specific prompts ensure handling of edge cases like empty strings, spaces, and mixed case characters. This demonstrates that clearer and more detailed prompts improve code quality. Hence, prompt-based programming plays an important role in effective AI-assisted coding.

Task-4: Tool Comparison Reflection

Here's a reflection on Gemini, Copilot, and Cursor AI, considering their general usability and code quality, as understood from public information and common developer discourse. Please note that this is not based on direct operational experience with these tools within our current Colab session.