Assignment-10.3

NAME : P.CHARAN KUMAR

BATCH :-41

ROLL NO :-2303A52138

Problem Statement 1: AI-Assisted Bug Detection

Scenario: A junior developer wrote the following Python function to calculate factorials:

```
def factorial(n):
result = 1
for i in range(1, n):
result = result * i
return result
```

Instructions:

1. Run the code and test it with factorial(5).
2. Use an AI assistant to:
   - o Identify the logical bug in the code.
   - o Explain why the bug occurs (e.g., off-by-one error).
   - o Provide a corrected version.
3. Compare the AI's corrected code with your own manual fix.
4. Write a brief comparison: Did AI miss any edge cases (e.g., negative numbers, zero)?

Code:

```
def factorial(n):
    result = 1
    for i in range(1, n):
        result = result * i
    return result
number = int(input("Enter a number: "))
print("Factorial of", number, "is", factorial(number))
```

PROMPT: Identify the error in the above code and correct it and explain the bug errors in the above code. Provide me the corrected code and the explanation of the bug errors.

Ai -corrected -code:

```
# PROMPT: identify the error in the above code and correct it and explain  the bug errors in the above code.
# Provide me the corrected code and the explanation of the bug errors.
# The error in the above code is that the loop in the factorial function is not including the number 'n' itself.
# The loop should run from 1 to n (inclusive) instead of 1 to n-1.
# This means that the factorial of a number will be calculated incorrectly,
# as it will not include the multiplication by 'n'.
# The corrected code is as follows:
def factorial(n):
    result = 1
    for i in range(1, n + 1):  # Change n to n + 1 to include n in the loop
        result = result * i
    return result
number = int(input("Enter a number: "))
print("Factorial of", number, "is", factorial(number))
```

Output:

```
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Progr
PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.v
PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.v
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Progr
graming/Assignment-10.3.py"
Enter a number: 5
Factorial of 5 is 24
Enter a number: 5
Factorial of 5 is 120
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing>
```

Justification: By changing the loop to run from 1 to n + 1, we ensure that the multiplication includes 'n' itself, which is essential for calculating the correct factorial value.

Problem Statement 2: Task 2 — Improving Readability &

Documentation

Scenario:The following code works but is poorly written:

```
def calc(a, b, c):
if c == "add":
return a + b
elif c == "sub":
return a – b
elif c == "mul":
return a * b
elif c == "div":
return a/b
```

5. Use AI to:

o Critique the function's readability, parameter naming, and lack of documentation. Rewrite the function with:

1. Descriptive function and parameter names.

2. A complete docstring (description, parameters, return value, examples).

3. Exception handling for division by zero.

4. Consideration of input validation.

6. Compare the original and AI-improved versions.

7. Test both with valid and invalid inputs (e.g., division by zero, non-string operation).

Code:

```
def calc(a, b, c):
    if c == "add":
        return a + b
    elif c == "sub":
        return a - b
    elif c == "mul":
        return a * b
    elif c == "div":
        return a/b
print(calc(10, 5, "add"))
print(calc(10, 5, "sub"))
```

Output:

```
PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.venv/Scr
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.
graming/Assignment-10.3.py"
15
5
```

Prompt:Create a python function that makes calculation on add ,sub,div,and mul.It should handle exception handling and return the values and improve the code the original code.

Ai -Corrected code:

```
def calculate(num1, num2, operation):
    """
    This function performs basic arithmetic operations based on the specified operation type.
    Parameters:
    num1 (float): The first number for the operation.
    num2 (float): The second number for the operation.
    operation (str): The type of operation to perform. It can be "add", "sub", "mul", or "div".
    Returns:
    float: The result of the arithmetic operation, or an error message if the operation is invalid or if there is
    """
    if operation == "add":
        return num1 + num2
    elif operation == "sub":
        return num1 - num2
    elif operation == "mul":
        return num1 * num2
    elif operation == "div":
        if num2 != 0:
            return num1 / num2
        else:
            return "Error: Division by zero is not allowed."
    else:
        return "Error: Invalid operation type. Please choose 'add', 'sub', 'mul', or 'div'."
# Testing the function with different inputs
print(calculate(10, 5, "add"))  # Output: 15
print(calculate(10, 5, "sub"))  # Output: 5
print(calculate(10, 5, "mul"))  # Output: 50
print(calculate(10, 5, "div"))  # Output: 2.0
print(calculate(10, 0, "div"))  # Output: Error: Division by zero is not allowed.
print(calculate(10, 5, "mod"))  # Output: Error: Invalid operation type. Please choose 'add', 'sub', 'mul', or '
```

Output:

```
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.venv/Scripts/python.exe" "c:/Users/
graming/Assignment-10.3.py"
15
5
15
5
50
2.0
Error: Division by zero is not allowed.
Error: Invalid operation type. Please choose 'add', 'sub', 'mul', or 'div'.
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing>
```

Justification:

The function is designed to be clear and descriptive, with meaningful parameter names and a comprehensive docstring that explains its purpose, parameters, and return value. It also includes error handling for division by zero and invalid operation types, ensuring that it behaves predictably in various scenarios. The function is tested with different inputs to confirm its correctness and robustness.

Problem Statement 3: Enforcing Coding Standards

Scenario: A team project requires PEP8 compliance. A developer

submits:

def Checkprime(n):

for i in range(2, n):

if n % i == 0:

return False

return   True

Instructions:

8. Verify the function works correctly for sample inputs.

9. Use an AI tool (e.g., ChatGPT, GitHub Copilot, or a PEP8 linter

with AI explanation) to:o List all PEP8 violations.

o Refactor the code (function name, spacing, indentation,naming).

10. Apply the AI-suggested changes and verify functionality is preserved.

Code:

```
def Checkprime(n):
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
num = int(input("Enter a number: "))
if Checkprime(num):
    print(num, "is a prime number.")
else:    print(num, "is not a prime number.")
```

Output:

```
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.venv/Scripts/python.exe" "c:/Users/ksair/Dc
graming/Assignment-10.3.py"
Enter a number: 5
5 is a prime number.
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.venv/Scripts/python.exe" "c:/Users/ksair/Dc
graming/Assignment-10.3.py"
Enter a number: 4
4 is not a prime number.
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing>
```

Prompt:

Verify the given Python function using sample inputs. Identify all PEP8 style violations and explain them. Refactor the code to follow PEP8 naming, spacing, and indentation rules, then confirm the refactored code still works correctly.

Ai -Corrected Code:

```
def check_prime(n):
    """Check if a number is prime."""
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def main():
    num = int(input("Enter a number: "))
    if check_prime(num):
        print(f"{num} is a prime number.")
    else:
        print(f"{num} is not a prime number.")
```

Output:

```
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/k
graming/Assignment-10.3.py"
Enter a number: 11
11 is a prime number.
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/k
graming/Assignment-10.3.py"
Enter a number: 4
4 is not a prime number.
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing>
```

Justification:

The function name is changed to check_prime to follow the convention. Added a docstring to explain the purpose of the function. Added a check for numbers less than or equal to 1, as they are not prime. Used `int(n**0.5) + 1` to optimize the prime checking process, as a number is not prime if it has a divisor less than or equal to its square root.

Problem Statement 4: AI as a Code Reviewer in Real Projects

Scenario: In a GitHub project, a teammate submits:

def processData(d):

return [x * 2 for x in d if x % 2 == 0]

Instructions:

1. Manually review the function for:

o Readability and naming.

o Reusability and modularity.

o Edge cases (non-list input, empty list, non-integer

elements).

2. Use AI to generate a code review covering:

a. Better naming and function purpose clarity.

b. Input validation and type hints.

c. Suggestions for generalization (e.g., configurable

multiplier).

3. Refactor the function based on AI feedback.

4. Write a short reflection on whether AI should be a

standalone reviewer or an assistant.

Code:

```
def processData(d):
    return [x * 2 for x in d if x % 2 == 0]
data = [1, 2, 3, 4, 5, 6]
result = processData(data)
print(result)
```

Output:

```
PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.venv/Sc
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/
graming/Assignment-10.3.py"
[4, 8, 12]
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing>
```

Prompt:

Review the following Python function for readability, naming, reusability, and edge cases. Suggest improvements with better naming, type hints, input validation, and generalization (configurable multiplier). Refactor the code and explain whether AI should be a standalone code reviewer or an assistant.

Ai-Corrected-code:

```
from typing import List
def process_data(data: List[int], multiplier: int = 2) -> List[int]:
    """
    Processes the input data by multiplying even numbers by a specified multiplier.

    Args:
        data (List[int]): A list of integers to be processed.
        multiplier (int, optional): The value to multiply even numbers by. Defaults to 2.

    Returns:
        List[int]: A list of processed integers where even numbers are multiplied by the multiplier.
    """
    if not isinstance(data, list):
        raise ValueError("Input data must be a list of integers.")

    if not all(isinstance(x, int) for x in data):
        raise ValueError("All elements in the input data must be integers.")

    return [x * multiplier for x in data if x % 2 == 0]
# Example usage
data = [1, 2, 3, 4, 5, 6]
result = process_data(data, multiplier=3)
print(result)  # Output: [6, 12, 18]
```

Output:

```
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Program
graming/Assignment-10.3.py"
[4, 8, 12]
[6, 12, 18]
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing>
```

Justification:

The refactored code includes improved naming conventions for better readability, type hints for clarity, and input validation to handle edge cases. The function is now more reusable with a configurable multiplier.

Problem Statement 5: — AI-Assisted Performance Optimization

Scenario: You are given a function that processes a list of integers, but

it runs slowly on large datasets:

def sum_of_squares(numbers):

total = 0

for num in numbers:

total += num ** 2

return total

Instructions:

1. Test the function with a large list (e.g., range(1000000)).

2. Use AI to:

o Analyze time complexity.

o Suggest performance improvements (e.g., using built-in

functions, vectorization with NumPy if applicable).

o Provide an optimized version.

3. Compare execution time before and after optimization.

4. Discuss trade-offs between readability and performance.

Code:

```
def sum_of_squares(numbers):
    total = 0
    for num in numbers:
        total += num ** 2
    return total
# Example usage
numbers = [1, 2, 3, 4]
result = sum_of_squares(numbers)
print("The sum of squares is:", result)
```

Output:

```
PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.venv/Scripts/Ac
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing> & "C:/Users/ksair/Downloads/Competitive Programing/.venv/Sc
graming/Assignment-10.3.py"
The sum of squares is: 30
(.venv) PS C:\Users\ksair\Downloads\Competitive Programing>
```

Prompt:

Analyze the time complexity of the given function that computes the sum of squares. Suggest performance optimizations using Python built-ins or NumPy, provide an optimized version, compare execution time, and discuss trade-offs between readability and performance.

Ai -Correcred-code:

```
# Analyze the time complexity of the given function that computes the sum of squares. S
# code:
import time
import numpy as np
def sum_of_squares_optimized(numbers):
    return np.sum(np.square(numbers))
# Example usage
numbers = [1, 2, 3, 4]
# Measure execution time of the original function
start_time = time.time()
original_result = sum_of_squares(numbers)
end_time = time.time()
print("Original function execution time:", end_time - start_time)

# Measure execution time of the optimized function
start_time = time.time()
optimized_result = sum_of_squares_optimized(numbers)
end_time = time.time()
print("Optimized function execution time:", end_time - start_time)
```

Output:

```
TASK 5 OUTPUT:
Performance Issue: Loop-based accumulation (slower)
Fix Applied      : Used generator expression with sum()
Optimized Result : 285
--------------------------------------------------
```

Justification:

Task 5 shows how AI can analyze code performance and suggest faster implementations. It helps optimize functions for large datasets using efficient Python constructs or libraries like NumPy. This demonstrates AI's role in improving efficiency while maintaining code readability.