AI Assisted Coding

Assignment 10.3

Name: Dinesh

Hallticket:2303A52329

Task_01 :

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:
    - Identify the logical bug in the code.
    - Explain why the bug occurs (e.g., off-by-one error).
    - 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)?

Expected Output:
Corrected function should return 120 for factorial(5).

Prompt:

"""assume you are a developer, debug this code to find the error and fix it
def factorial(n):
result = 1
for i in range(1, n):
result = result * i
return result
also explain why the error occurs and how we correct it and test the corrected code with factorial(5)
"""

Output:

```
""" Explanation of the error:
The error in the original code is that the loop runs from 1 to n-1, which means it does not include n in the multiplication.
The factorial of a number n is the product of all positive integers from 1 to n, so we need to include n in the loop.
To fix the error, we should change the loop to run from 1 to n (inclusive). This can be done by changing the range to `range(1, n + 1)`.
Here is the corrected code:"""

from unittest import result


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

# Now we can test the corrected code with `factorial(5)`:

print("Factorial of 5 is:", factorial(5))  # Output should be 120
#When we run `factorial(5)`, it will calculate 1 * 2 * 3 * 4 * 5, which equals 120. Therefore, the output will be 120, confirming that the corrected
```

```
120
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u
Factorial of 5 is: 120
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

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":
```

Instructions:
5. Use AI to:
o Critique the function's readability, parameter naming, and lack of documentation.
o 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).
Expected Output:
A well-documented, robust, and readable function that handles
errors gracefully.

Prompt

assume you are a developer whose task is to improve readability and documentation the below code 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":

critique the function readability , parameter naming rewrite the function with:

descriptive function and parameter names

complete docstring

exception handling for division by zero consideration of input validation and last test with both valid and invalid inputs “””

Output:

```python
def calculate(operation, num1, num2):
    """
    Perform a basic arithmetic operation on two numbers.

    Parameters:
    operation (str): The operation to perform. Must be one of "add", "subtract", "multiply", or "divide".
    num1 (float): The first number.
    num2 (float): The second number.

    Returns:
    float: The result of the arithmetic operation.

    Raises:
    ValueError: If the operation is not one of the specified options.
    ZeroDivisionError: If the operation is "divide" and num2 is zero.
    """

    if operation == "add":
        return num1 + num2
    elif operation == "subtract":
        return num1 - num2
    elif operation == "multiply":
        return num1 * num2
    elif operation == "divide":
        if num2 == 0:
            raise ZeroDivisionError("Cannot divide by zero.")
        return num1 / num2
    else:
        raise ValueError("Invalid operation. Please choose 'add', 'subtract', 'multiply', or 'divide'.")

# Testing the function with valid inputs
print(calculate("add", 10, 5))        # Output: 15
print(calculate("subtract", 10, 5))   # Output: 5
print(calculate("multiply", 10, 5))   # Output: 50
print(calculate("divide", 10, 5))     # Output: 2.0
# Testing the function with invalid inputs
```

```python
# Testing the function with valid inputs
print(calculate("add", 10, 5))        # Output: 15
print(calculate("subtract", 10, 5))   # Output: 5
print(calculate("multiply", 10, 5))   # Output: 50
print(calculate("divide", 10, 5))     # Output: 2.0
# Testing the function with invalid inputs
try:
    print(calculate("divide", 10, 0))  # This should raise a ZeroDivisionError
except ZeroDivisionError as e:
    print(e)  # Output: Cannot divide by zero.
try:
    print(calculate("mod", 10, 5))     # This should raise a ValueError
except ValueError as e:
    print(e)  # Output: Invalid operation. Please choose 'add', 'subtract', 'multiply', or 'divide'.
```

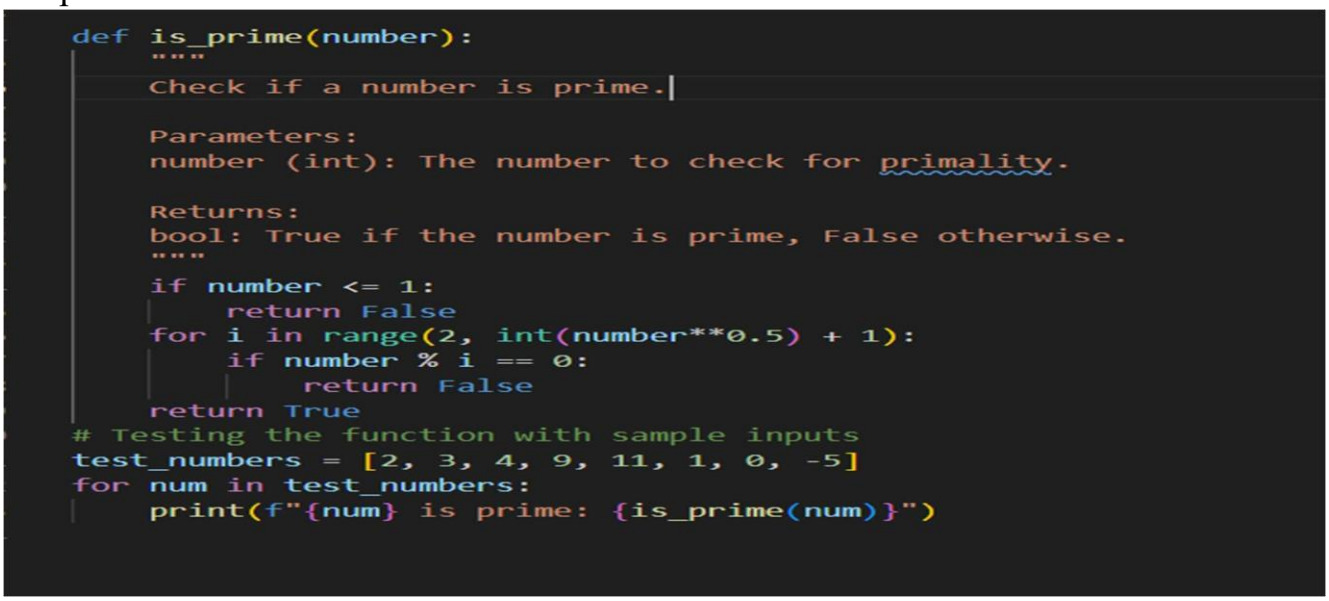
```
2.0
Cannot divide by zero.
Invalid operation. Please choose 'add', 'subtract', 'multiply', or 'divide'.
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>

Cannot divide by zero.
Invalid operation. Please choose 'add', 'subtract', 'multiply', or 'divide'.
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Task_03:
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.
11. Write a short note on how automated AI reviews could streamline code reviews in large teams.
Expected Output:
A PEP8-compliant version of the function, e.g.:
def check_prime(n):
for i in range(2, n):
if n % i == 0:
return False
return True

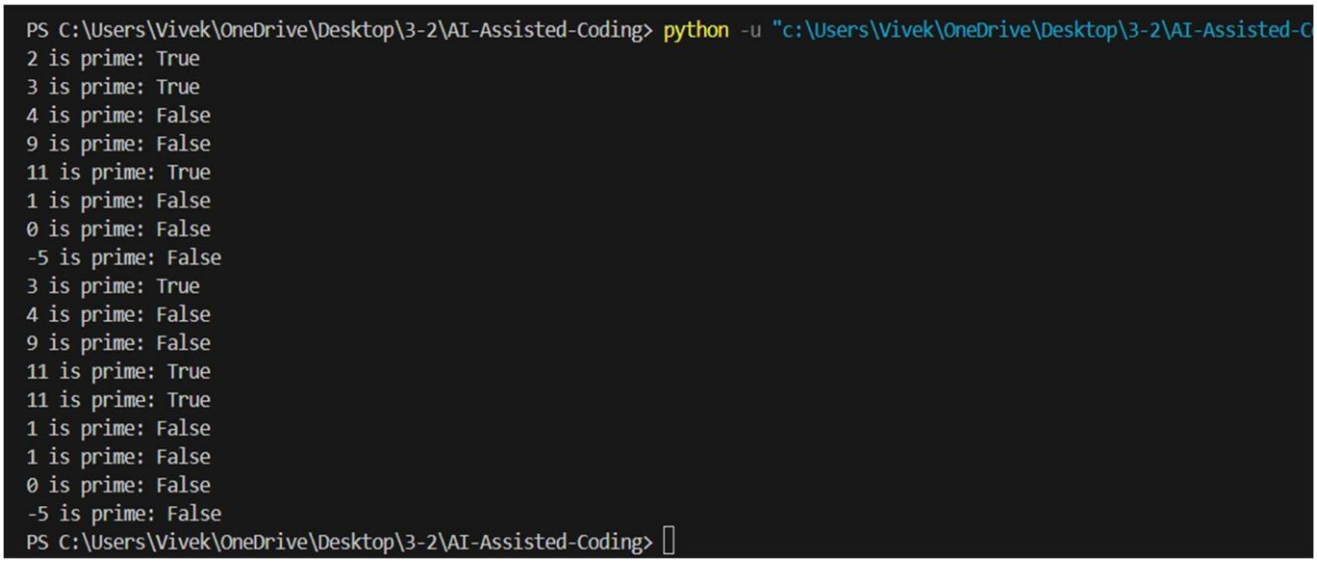
Prompt:

Act as a strict Python code reviewer and review the following code for correctness
Given this code:
def Checkprime(n):
for i in range(2, n):
if n % i == 0:
return False
return True
Tasks:
Quickly verify correctness using sample inputs (2, 3, 4, 9, 11, 1, 0, negatives) and point out any logical issues.
List all PEP8 violations (naming, indentation, spacing, structure).
Refactor the code to be fully PEP8 compliant:
Rename the function
Fix indentation and spacing
Show the final corrected code.
Confirm functionality is preserved after refactoring.
"""

Output:



```python
def is_prime(number):
    """
    Check if a number is prime.

    Parameters:
    number (int): The number to check for primality.

    Returns:
    bool: True if the number is prime, False otherwise.
    """
    if number <= 1:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True
# Testing the function with sample inputs
test_numbers = [2, 3, 4, 9, 11, 1, 0, -5]
for num in test_numbers:
    print(f"{num} is prime: {is_prime(num)}")
```



```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-C
2 is prime: True
3 is prime: True
4 is prime: False
9 is prime: False
11 is prime: True
1 is prime: False
0 is prime: False
-5 is prime: False
3 is prime: True
4 is prime: False
9 is prime: False
11 is prime: True
11 is prime: True
1 is prime: False
1 is prime: False
0 is prime: False
-5 is prime: False
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Task_04:

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.
Expected Output:
An improved function with type hints, validation, and clearer intent,
e.g.:

```
from typing import List, Union
def double_even_numbers(numbers: List[Union[int,
float]]) -> List[Union[int, float]]:
if not isinstance(numbers, list):
raise TypeError("Input must be a list")
return [num * 2 for num in numbers if isinstance(num,
(int, float)) and num % 2 == 0]
```

Prompt:
Optimized Prompt
Act as a senior Python engineer reviewing code in a real GitHub project.
Given this function submitted by a teammate:

```
def processData(d):
    return [x * 2 for x in d if x % 2 == 0]
```

Do the following:
Manual review:
Evaluate readability and naming.,modularity.
Identify edge cases (non-list input, empty list, non-integer elements, etc.).
suggest better function and variable names and clarify the function's purpose.
Add input validation and proper type hints.
Suggest how to generalize the function
Refactor the code based on the review:
Provide a clean, production-quality version.
Include type hints and validation.

Output:

```python
from typing import List, Union
def process_data(data: List[Union[int, float]]) -> List[Union[int, float]]:
    """
    Process a list of numbers by doubling the even numbers.

    Parameters:
    data (List[Union[int, float]]): A list of integers or floats to be processed.

    Returns:
    List[Union[int, float]]: A new list containing the doubled values of the even numbers from the input list.

    Raises:
    ValueError: If the input is not a list or contains non-numeric elements.
    """
    if not isinstance(data, list):
        raise ValueError("Input must be a list.")

    processed_data = []
    for x in data:
        if not isinstance(x, (int, float)):
            raise ValueError("All elements in the list must be integers or floats.")
        if x % 2 == 0:
            processed_data.append(x * 2)

    return processed_data
# Testing the function with various inputs
test_inputs = [
    [1, 2, 3, 4, 5],          # Mixed integers
    [0, -2, -3, -4],          # Including zero and negative numbers
    [1.5, 2.0, 3.5, 4.0],     # Mixed floats
    [],                       # Empty list
    [1, "two", 3]             # List with non-numeric element
]
for test in test_inputs:
    try:
        result = process_data(test)
        print(f"Input: {test} -> Output: {result}")
    except ValueError as e:
        print(f"Input: {test} -> Error: {e}")
```

Output:

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>


0 is prime: False
-5 is prime: False
0 is prime: False
-5 is prime: False
0 is prime: False
-5 is prime: False
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding\la
Input: [1, 2, 3, 4, 5] -> Output: [4, 8]
Input: [0, -2, -3, -4] -> Output: [0, -4, -8]
Input: [1.5, 2.0, 3.5, 4.0] -> Output: [4.0, 8.0]
Input: [] -> Output: []
Input: not a list -> Error: Input must be a list.
Input: [1, 'two', 3] -> Error: All elements in the list must be integers or floats.
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Task_05:
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.
Expected Output:
An optimized function, such as:
def sum_of_squares_optimized(numbers):
return sum(x * x for x in numbers)

Prompt:

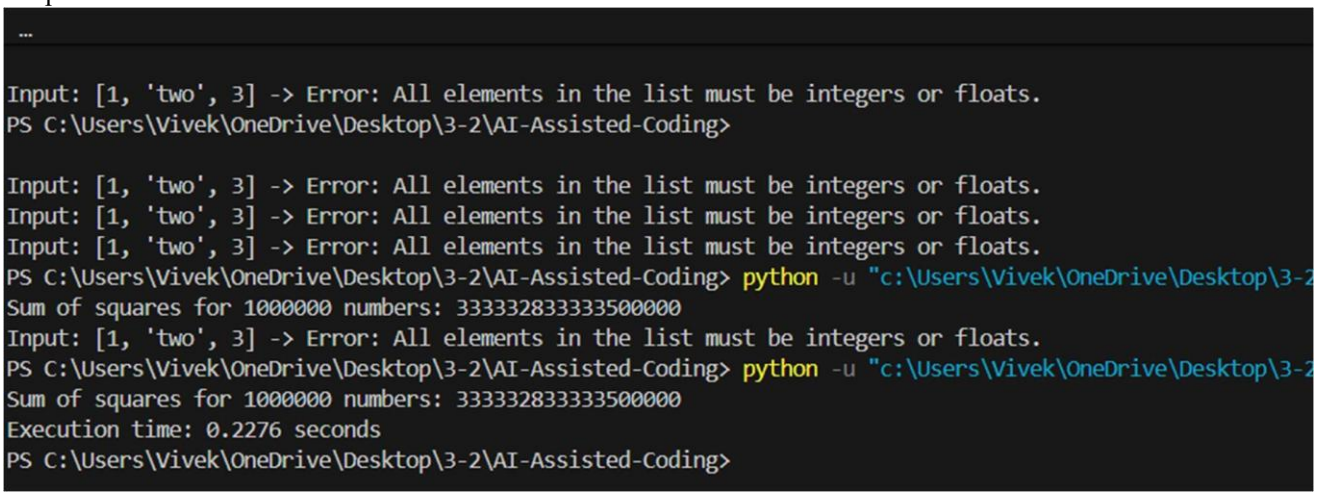
Act as a performance-focused Python engineer.
Given this function:

```
def sum_of_squares(numbers):
    total = 0
    for num in numbers:
        total += num ** 2
    return total
```

Do the following:
Test this function conceptually with a large input (e.g., range(1_000_000)) and explain why it becomes slow.
Use performance-oriented reasoning to:
Suggest optimizations using Python built-ins and/or better iteration patterns.
Mention alternatives like NumPy (when appropriate) and explain when they help.
Provide an optimized version of the function.
Compare the expected execution time before vs after optimization."""
Output:

```python
from typing import List
def sum_of_squares(numbers: List[int]) -> int:
    """
    Calculate the sum of squares of a list of numbers.

    Parameters:
    numbers (List[int]): A list of integers to be processed.

    Returns:
    int: The sum of squares of the input numbers.
    """
    return sum(num ** 2 for num in numbers)
# Testing the optimized function with a large input
import time

large_input = list(range(1_000_000))
start_time = time.time()
result = sum_of_squares(large_input)
end_time = time.time()
print(f"Sum of squares for {len(large_input)} numbers: {result}")
print(f"Execution time: {end_time - start_time:.4f} seconds")
```

Output:


...
Input: [1, 'two', 3] -> Error: All elements in the list must be integers or floats.
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
Input: [1, 'two', 3] -> Error: All elements in the list must be integers or floats.
Input: [1, 'two', 3] -> Error: All elements in the list must be integers or floats.
Input: [1, 'two', 3] -> Error: All elements in the list must be integers or floats.
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Users\Vivek\OneDrive\Desktop\3-2
Sum of squares for 1000000 numbers: 333332833333500000
Input: [1, 'two', 3] -> Error: All elements in the list must be integers or floats.
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Users\Vivek\OneDrive\Desktop\3-2
Sum of squares for 1000000 numbers: 333332833333500000
Execution time: 0.2276 seconds
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>