LAB RE-TEST 02

NAME:ALLA SREEMANTH REDDY

ROLL NO : 2403A510G1

BATCH NO:06

Question 1: AI-Assisted Code Review

Task 1:

Submit a sample Python script (with inefficient logic and poor naming) to an AI model for review.

Prompt:

Review this Python code for inefficiency and poor naming. Suggest improvements for readability, performance, and Pythonic style.

Code:

C: > Users > allas > 2.1.test.py > ...

```python
def do(a):
 b=[]
 for i in range(len(a)):
  b.append(a[i]*2)
 return b
def main():
 x=[1,2,3,4,5]
 y=do(x)
 print("result is:")
 for i in range(0,len(y)):
  print(y[i])
main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code> & C:\Python313\python.exe c:/Users/allas/2.1.test.py
result is:
2
4
6
8
10
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code>
```

Observation:

1. Rename do() to something meaningful like double_elements().
2. Use **list comprehension** for simplicity and efficiency.
3. Use descriptive variable names (numbers, doubled_numbers).
4. Print results more cleanly.

Task 2 – Ask the AI to rewrite the code following PEP 8 guidelines and include explanations of improvements in naming, indentation, and structure.

Prompt:

Rewrite the following Python code according to **PEP 8 style guidelines**.
Explain what improvements were made in terms of naming, indentation, and structure

Code:

2test.py | 2.2task.py | re.py12.py | retest.2py.py | 2.1.test.py

C: > Users > allas > 2test.py > ...

```python
def double_elements(numbers):
    """Return a list with each number doubled."""
    return [num * 2 for num in numbers]

def main():
    numbers = [1, 2, 3, 4, 5]
    doubled_numbers = double_elements(numbers)
    print("Doubled numbers:", doubled_numbers)

if __name__ == "__main__":
    main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code> & C:\Python313\python.exe c:/Users/allas/2test.py
Doubled numbers: [2, 4, 6, 8, 10]
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code>
```

Observation:

Aspect	Issue in Original Code	PEP 8 Improvement
Naming	Function name do() and variables a, b, x, y not descriptive	Renamed to double_elements, numbers, doubled_numbers
Indentation	Mixed/incorrect indentation (2 spaces instead of 4)	Corrected to 4 spaces per indentation level
Structure	Code not separated clearly, missing docstring, no __main__ check	Added docstring, line breaks, and proper structure
Readability	Dense, hard to read	Added whitespace, improved print statement

Question 2: Improving Readability and Maintainability

Task 1:

Provide AI with a long, nested function (e.g., multiple if-else or loops). Ask it to refactor the code into modular, readable functions.

Prompt:

The following Python function has long nested logic with multiple if-else and loops. Please refactor it into modular, readable functions following clean code principles. Add comments and make the code easier to maintain.

CODE:

```python
# Refactored version for readability and maintainability

def assign_grade(marks):
    """Assign grade based on marks."""
    if marks >= 90:
        return "A"
    elif marks >= 75:
        return "B"
    elif marks >= 60:
        return "C"
    else:
        return "F"


def categorize_students(students):
    """Separate students into passed and failed lists."""
    passed = []
    failed = []
    for student in students:
        grade = assign_grade(student["marks"])
        if grade != "F":
            passed.append(student["name"])
        else:
            failed.append(student["name"])
    return passed, failed


def calculate_average(students):
    """Calculate the average score of all students."""
    total_score = sum(student["marks"] for student in students)
    return total_score / len(students)


def process_student_data(students):
    """Main function to process student data in a modular way."""
    avg_score = calculate_average(students)
    passed, failed = categorize_students(students)

    print("Average Score:", avg_score)
    print("Passed Students:", passed)
    print("Failed Students:", failed)


# Example test data
students = [
    {"name": "Alice", "marks": 95},
    {"name": "Bob", "marks": 72},
    {"name": "Charlie", "marks": 58},
]

process_student_data(students)
```

```
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code> & C:\Python313\python.exe c:/Users/allas/retest.2py.py
Average Score: 75.0
Passed Students: ['Alice', 'Bob']
Failed Students: ['Charlie']
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code>
```

Observation:

Aspect	Before Refactoring	After Refactoring
Structure	Single long function with mixed logic	Modularized into 3 helper functions
Readability	Hard to follow due to nested if-else and loops	Clear, descriptive function names
Maintainability	Difficult to modify grading or logic	Easy to adjust grading rules or output format
Comments	None	Added meaningful docstrings and inline

Task 2:

Use AI to generate code comments and a short design summary, explaining how refactoring improved maintainability and clarity.

Prompt: Refactor the following nested Python function into a more modular and readable format.Add helper functions and a short design summary explaining how the refactoring improved maintainability and clarity.

Code:

C: > Users > allas > 2.2task.py > ...

```python
# Task 2: Refactoring for readability and maintainability

def is_even(n): return n % 2 == 0
def transform_even(n): return n * 2 if n > 10 else n
def transform_odd(n): return n + 5 if n < 5 else n - 1

def process_data(data):
    """Refactored function with improved readability."""
    result = [transform_even(i) if is_even(i) else transform_odd(i) for i in data]
    print("Processed data:", result)

# Example execution
data = [2, 4, 7, 12, 15, 3]
process_data(data)

# Design Summary
print("\nDesign Summary:")
print("• Code simplified using helper functions and list comprehension.")
print("• Easier to read, test, and maintain.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code> & C:\Python313\python.exe c:/Users/allas/2.2task.py
Processed data: [2, 4, 6, 24, 14, 8]

Design Summary:
• Code simplified using helper functions and list comprehension.
• Easier to read, test, and maintain.
PS C:\Users\allas\AppData\Local\Programs\Microsoft VS Code>
```

- Observation: The **original function** had **nested if-else** blocks, which made it longer and harder to follow–.

- The **refactored version**:
    - Uses **three helper functions** — is_even(), transform_even(), and transform_odd() — making each operation clear and reusable.
    - Replaces the loop with a **list comprehension**, improving conciseness and readability.
    - Adds a **docstring** and a **design summary** to clearly explain intent.
- The result is **cleaner, modular, and easier to maintain**.