# Assignment

# Lab 10 – Code Review and Quality

**Htno : 2503A52L10**

## Task 1 – Syntax and Logic Errors

**Original Code:**  
def calc\_average(marks):  
total = 0  
for m in marks:  
total += m  
average = total / len(marks)  
return avrage  
marks = [85, 90, 78, 92]  
print("Average Score is ", calc\_average(marks)

**Issues Identified:**

- Improper indentation.  
- Typo in variable name 'avrage'.  
- Missing closing parenthesis in print statement.  
- No input validation (possible ZeroDivisionError).  
- No docstring or type hints.

**Fixed Code + Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

## Task 2 – PEP 8 Compliance

**Original Code:**

def area\_of\_rect(L,B):return L\*B  
print(area\_of\_rect(10,20))

**Issues Identified:**

- Function parameters use uppercase letters (should be lowercase and descriptive).  
- One-liner function reduces readability.  
- Missing docstring and type hints.  
- No space after commas.

**Refactored Code + Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

## Task 3 – Readability Enhancement

**Original Code:**

def c(x,y):  
return x\*y/100  
a=200  
b=15  
print(c(a,b))

**Issues Identified:**

- Non-descriptive function name ('c').  
- Non-descriptive variable names ('a', 'b').  
- No docstring or inline comments.  
- Missing indentation and poor formatting.

**Refactored Code:**

A screenshot of a computer program

AI-generated content may be incorrect.

## Task 4 – Refactoring for Maintainability

**Original Code:**

students = ["Alice", "Bob", "Charlie"]  
print("Welcome", students[0])  
print("Welcome", students[1])  
print("Welcome", students[2])

**Issues Identified:**

- Repeated print statements (not scalable).  
- No modularity, cannot handle larger student lists.  
- No function abstraction.  
- Harder to maintain and reuse.

**Refactored Code + Output:**

A screen shot of a computer

AI-generated content may be incorrect.

## Task 5 – Performance Optimization

**Original Code:**

nums = [i for i in range(1,1000000)]  
squares = []  
for n in nums:  
 squares.append(n\*\*2)  
print(len(squares))

**Issues Identified:**

- Inefficient loop with append (slower than list comprehension).  
- Creates unnecessary intermediate list 'nums'.  
- High memory usage for large ranges.  
- Not scalable.

**Refactored Code + Output:**

## A screenshot of a computer program AI-generated content may be incorrect.

## Task 6 – Complexity Reduction

Original Code:

def grade(score):  
 if score >= 90:  
 return "A"  
 else:  
 if score >= 80:  
 return "B"  
 else:  
 if score >= 70:  
 return "C"  
 else:  
 if score >= 60:  
 return "D"  
 else:  
 return "F"

Issues Identified:

- Overly nested if-else statements (hard to read).  
- Poor maintainability.  
- No type hints or docstring.

Refactored Code with elif + Output:

A screenshot of a computer program

AI-generated content may be incorrect.

Observation :

•The tasks highlighted common issues in Python code, including syntax, logic, readability, and maintainability problems.

•Applying PEP 8 guidelines improved code structure, naming conventions, and overall clarity.

•Refactoring repetitive and complex logic into functions enhanced modularity and scalability.

•Performance optimization using list comprehensions, generators, and NumPy reduced execution time and memory usage.

•Overall, AI-assisted code review improved code quality, making it cleaner, efficient, and easier to maintain.