# ASSIGNMENT-10.1

**TASK 1: Syntax and Logic Errors**

**PROMPT**: Identify and fix syntax and logic errors in the following Python code.

Provide:

1) Corrected runnable Python code

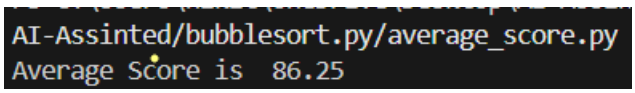
2) Explanation of each fix

**CODE:**

```
# Calculate average score of a student
def calc_average(marks):
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average


marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks))
```

**OUTPUT:**




**Explanation:**

1) Fixed indentation inside the function to follow Python syntax rules.

2) Corrected the spelling mistake in the return statement (avrage → average).

3) Added the missing closing bracket in the print() statement.

4)After corrections, the program runs successfully and prints 86.25 as the average score.

**Task 2 : PEP 8 Compliance**

**PROMPT:** Refactor the following Python code to follow PEP 8 style guidelines. Provide well-formatted and PEP 8-compliant Python code.

**CODE:** def area_of_rect(length, breadth):

    return length * breadth

print(area_of_rect(10, 20))

**OUTPUT:**

```
AI-Assinted/bubblesort.py/Compliance.PY
200
```

**Explanation:**

- Function parameters renamed to lowercase (L, B → length, breadth).
- Proper spacing after commas.
- Removed inline function definition and placed return on a new line.
- Added proper indentation.
- Added blank line before function call (PEP 8 standard).

**Task 3: Readability Enhancement**

**PROMPT**: Improve the readability of the following Python code without changing its logic.
Use descriptive variable names, proper formatting, and add inline comments.

**CODE:**

def calculate_percentage(amount, percentage):

    # Calculate the percentage of a given amount

    return amount * percentage / 100

# Given values

total_amount = 200

discount_percentage = 15

```
# Print the calculated result
print(calculate_percentage(total_amount, discount_percentage))
```

**OUTPUT:**

```
AI-Assinted/bubblesort.py/readability_enhancement.py
30.0
```

**Explanation:**

1) Renamed the function c() to calculate_percentage() to make it more descriptive.

2) Replaced unclear variables (x, y, a, b) with meaningful names like amount and percentage.

3) Added proper indentation to follow Python syntax rules.

4) Included inline comments to explain the logic clearly.

5) Formatted the code with proper spacing to improve readability.

**Task 4: Refactoring for Maintainability**

**PROMPT:** Refactor the following Python code to improve maintainability. Break repetitive code into reusable functions and make it modular.

**CODE:**

```
def welcome_student(name):
    """Print a welcome message for a student."""
    print(f"Welcome {name}")

def welcome_all_students(students):
    """Welcome all students in the list."""
    for student in students:
        welcome_student(student)

if __name__ == "__main__":
    students = ["Alice", "Bob", "Charlie"]
    welcome_all_students(students)
```

**OUTPUT:**

```
AI-Assinted/bubblesort.py/Maintainability.py
Welcome Alice
Welcome Bob
Welcome Charlie
```

**Explanation:**

1) Created a reusable function welcome_student() to avoid repeating print statements.

2)Used a for loop to iterate through the student list.

3) Removed hardcoded index values (students[0], students[1], etc.).

4) The code is now modular, cleaner, and easier to maintain.

5) If more students are added, no extra print statements are needed.

**Task 5 : Performance Optimization**

**PROMPT**: Optimize the following Python code to improve performance.
Use list comprehensions or efficient methods without changing the output.

**CODE:** # Find squares of numbers

```
nums = [i for i in range(1, 1000000)]

squares = [n**2 for n in nums]

print(len(squares))
```

**OUTPUT:**

```
AI-Assinted/bubblesort.py/performance_optimization.py
999999
```

**Explanation:**

1) Removed the unnecessary nums list to save memory.

2) Replaced the for loop with a list comprehension for faster execution.

3) Combined square calculation directly inside range().

4) Reduced number of lines and improved readability.

5) The output remains the same but the code runs more efficiently.

**Task 6: Complexity Reduction**

**PROMPT:** Simplify the following Python code to reduce complexity.
Use cleaner logic with elif statements without changing the output.

**CODE:**

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



# Example usage
print(grade(85))
```

**OUTPUT:**

```
AI-Assinted/bubblesort.py/complexity_reduction.py
B
```

**Explanation:**

1. Replaced multiple nested if-else statements with elif for cleaner structure.
2. Reduced unnecessary indentation levels.
3. Improved readability and maintainability.
4. Logic remains the same but code is easier to understand.
5. The function now looks simpler and more professional.