Assignment 10.1

Ashmitha Reddy

2303A52111

BATCH-40

Task Description #1 – Syntax and Logic Errors

Task: Use AI to identify and fix syntax and logic errors in a faulty

Python script.

Sample Input Code:

Calculate average score of a student def

calc_average(marks):

total = 0 for m

in marks:

total += m average = total / len(marks) return

avrage # Typo here marks = [85, 90, 78, 92]

print("Average Score is ", calc_average(marks)

Expected Output:

• Corrected and runnable Python code with explanations of the fixes.

PROMPT AND CODE:

```
# Find and fix all syntax and logic errors in the following Python code. Return the corrected runnable code
def calc_average(marks):
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average # Fixed typo
marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks))
```

Task Description #2 – PEP 8 Compliance

Task: Use AI to refactor Python code to follow PEP 8 style

guidelines. Sample

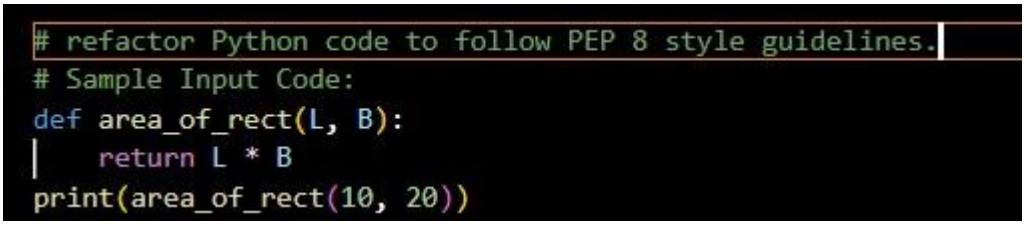
Input Code:

def area_of_rect(L,B) : return L*B print(area_of_rect(10,20))

Expected Output:

• Well-formatted PEP 8-compliant Python code.

PROMPT AND CODE:



Task Description #3 – Readability Enhancement

Task: Use AI to make code more readable without changing its

logic.

Sample Input Code:

def c(x,y):

return x*y/100

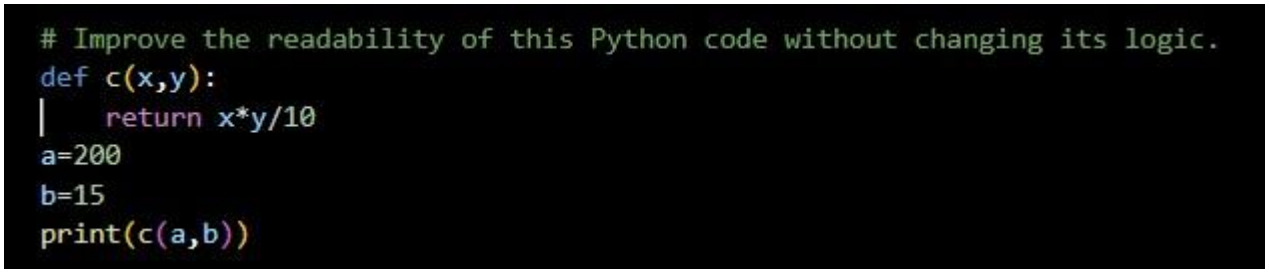
a=200 b=15

print(c(a,b))

Expected Output:

• Python code with descriptive variable names, inline comments, and clear formatting.

PROMPT AND CODE:



Task Description #4 – Refactoring for Maintainability

Task: Use AI to break repetitive or long code into reusable functions.

Sample Input Code:

students = ["Alice", "Bob", "Charlie"]

print("Welcome", students[0]) print("Welcome",

students[1]) print("Welcome", students[2])

Expected Output:

• Modular code with reusable functions.

PROMPT AND CODE:

```
# Improve the following code by removing repetition and creating reusable functions.
students = ["Alice", "Bob", "Charlie"]

def welcome_student(student_name):
    print("Welcome", student_name)

for student in students:
    welcome_student(student)
```

Task Description #5 – Performance Optimization

Task: Use AI to make the code run faster.

Sample Input Code: # Find squares

of numbers nums = [i for i in

range(1,1000000)] squares = [] for

n in nums:

squares.append(n**2) print(len(squares))

Expected Output:

• Optimized code using list comprehensions or vectorized operations.

PROMPT AND CODE:

```
# optimize the code to improve the performance of the following function that calculates the sum of squares of a list of numbers.
nums = [i for i in range(1,1000000)]
squares = [n**2 for n in nums]
print(len(squares))
```

Task Description #6 – Complexity Reduction Task:

Use AI to simplify overly complex logic.

Sample Input 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

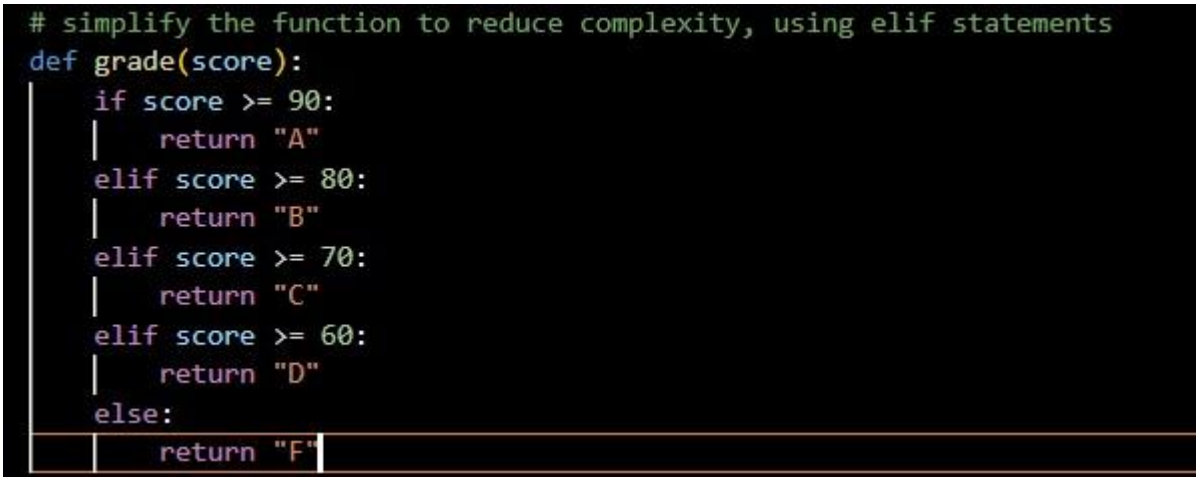
"D" else:

return "F"

Expected Output:

• Cleaner logic using elif or dictionary mapping.

PROMPT AND CODE:



```
# simplify the function to reduce complexity, using elif statements
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"
```