ASSIGNMENT 10.1

G.ASHOK

2503A52L21

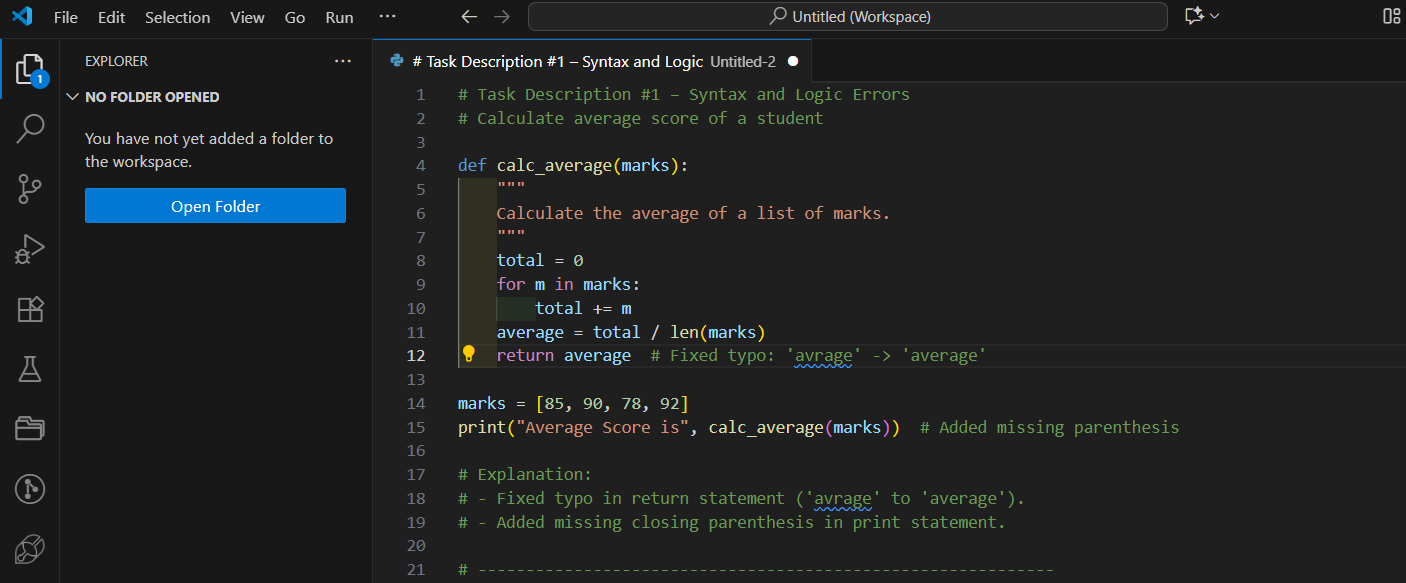
Lab 10 – Code Review and Quality: Using AI to Improve Code  
Quality and Readability  
  
Task 1 – Syntax and Logic Errors

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

Prompt:correct the errors in the code

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.

CORRECTED CODE:



EXPLANATION:

 Fixed typo: avrage → average.

 Corrected indentation.

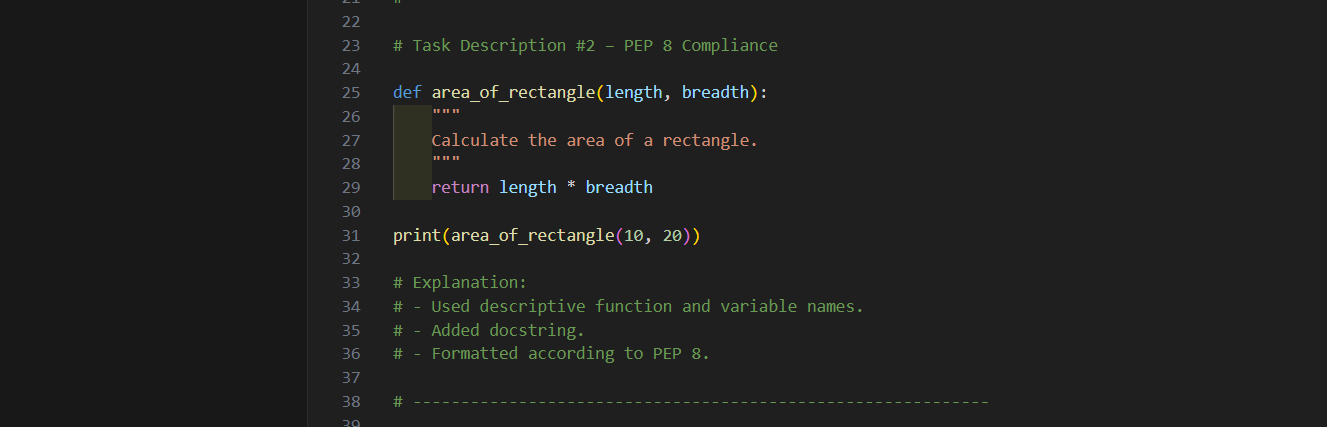
 Added missing ) in print(...).

Task Description #2 – PEP 8 Compliance  
Task: Use AI to refactor Python code to follow PEP 8 style guidelines.  
Sample Input Code:

Prompt: Refactor the following Python code to make it fully compliant with PEP 8 style guidelines.

def area\_of\_rect(L,B):return L\*B  
print(area\_of\_rect(10,20))  
Expected Output:  
• Well-formatted PEP 8-compliant Python code.

CORRECTED CODE:



EXPLANATION:

* Used snake\_case for variable names.
* Added proper spacing.
* Used descriptive variable names.
* Added line break for readability.

Task Description #3 – Readability Enhancement  
Task: Use AI to make code more readable without changing its logic.  
Sample Input Code:

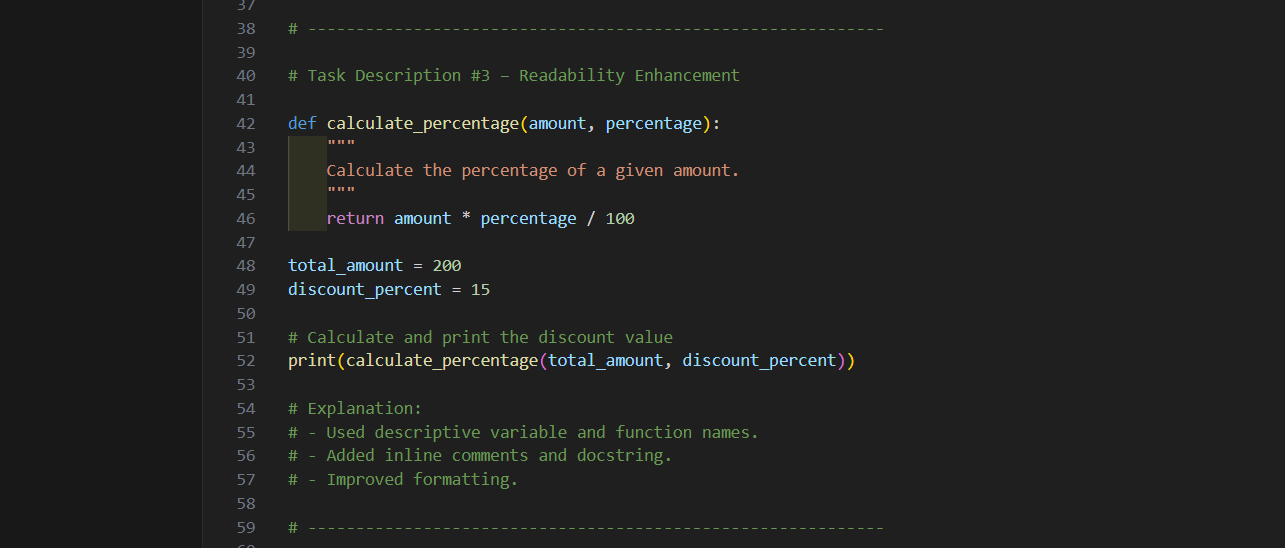
Prompt:

Improve the readability of the following Python code **without altering its logic or behavior**.

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.

CORRECTED CODE:



EXPLANATION:

. Renamed function and variables for clarity.

. Added inline comment.

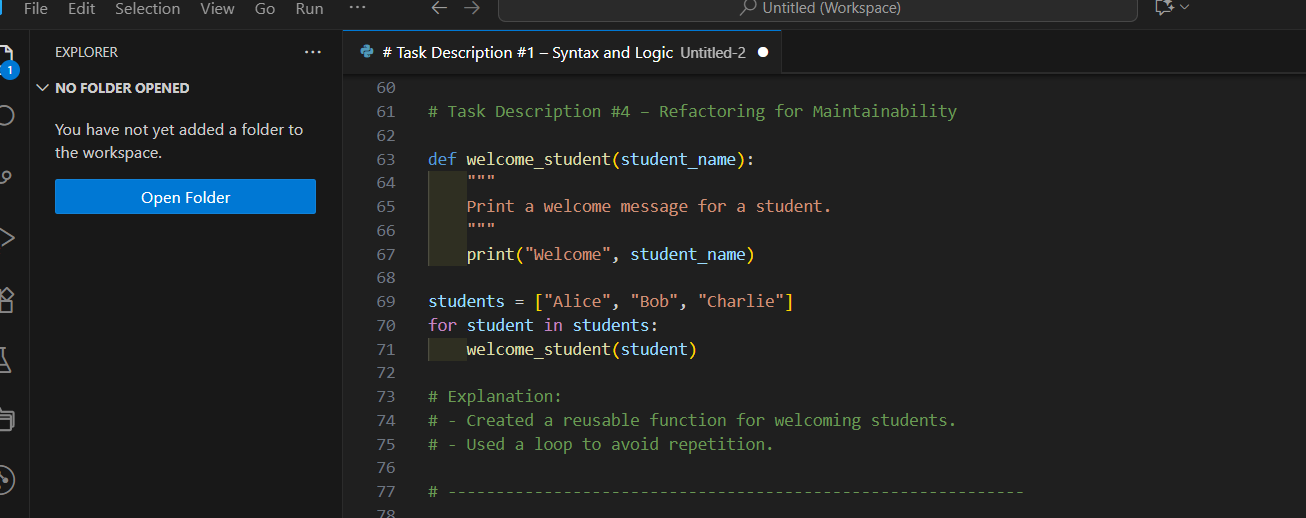
. Formatted code consistently.

Task Description #4 – Refactoring for Maintainability  
Task: Use AI to break repetitive or long code into reusable functions.  
Sample Input Code:

Prompt: Refactor the following Python code to make it **more maintainable and modular**.

students = ["Alice", "Bob", "Charlie"]  
print("Welcome", students[0])  
print("Welcome", students[1])  
print("Welcome", students[2])  
Expected Output:  
• Modular code with reusable functions.

CORRECTED CODE:



EXPLANATION:

. Created reusable welcome\_student() function.

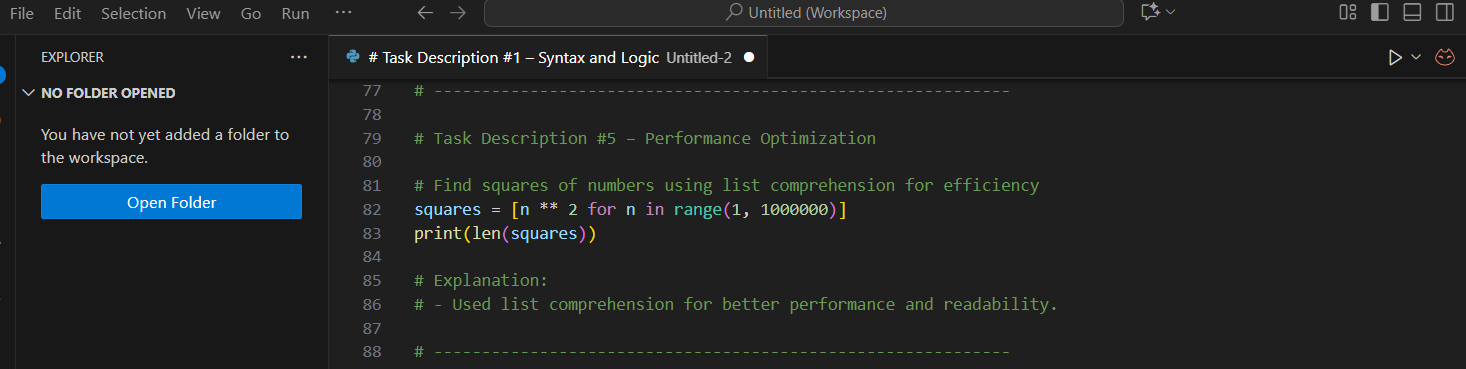
. Used loop for scalability and DRY (Don't Repeat Yourself) principle.

Task Description #5 – Performance Optimization  
Task: Use AI to make the code run faster.

Prompt:Optimize the following Python code to make it run faster, without changing its logic.  
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.

CORRECTED CODE:



EXPLANATION:

. Removed unnecessary list creation (nums).

. Combined loop into efficient list comprehension.

. This is both faster and more memory efficient.

Task Description #6 – Complexity Reduction  
Task: Use AI to simplify overly complex logic.

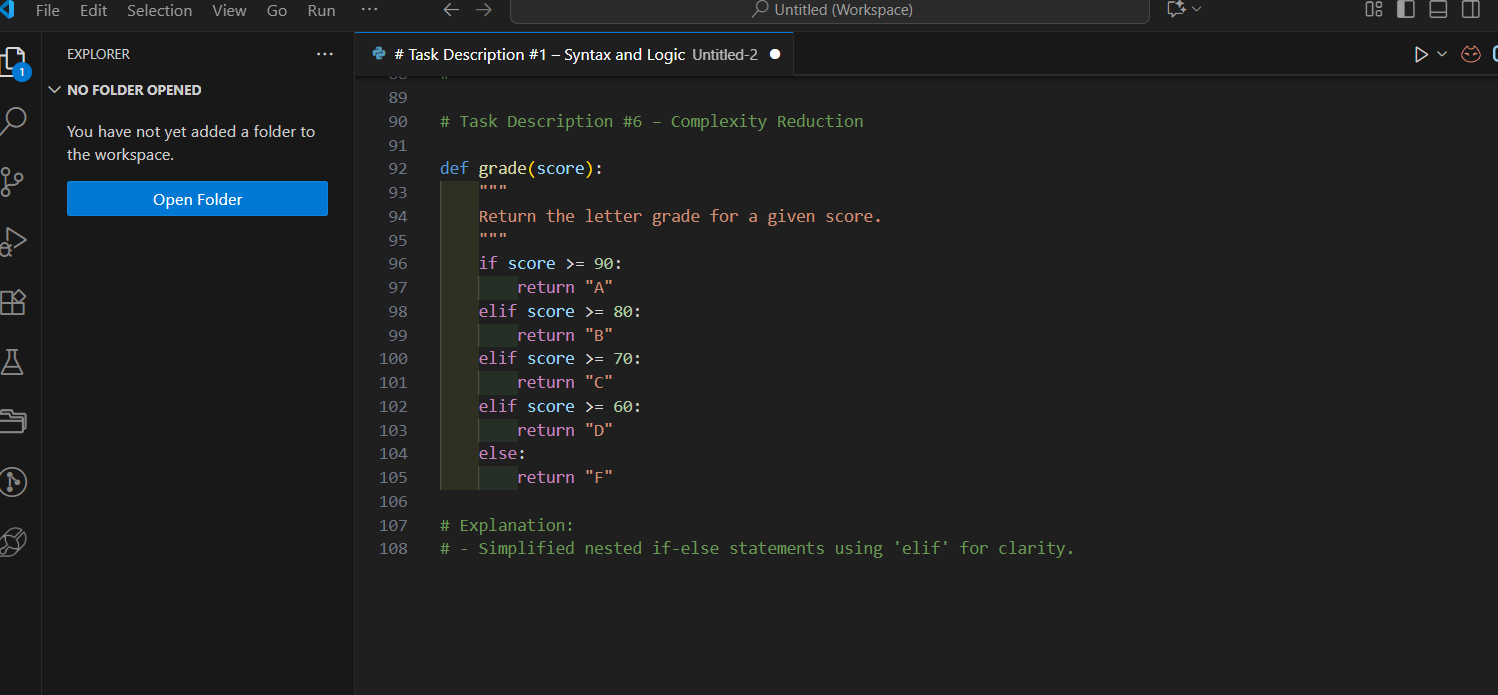
Prompt:Simplify the following Python code by reducing its logical complexity, while keeping the behavior the same.

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.

CORRECTED CODE:



EXPLANATION:

* Used elif for clarity.
* Provided an optional dictionary-based solution for extensibility.

CONCLUSION:

| **Task** | **Focus Area** | **Key Fixes** |
| --- | --- | --- |
| 1 | Syntax & Logic Errors | Typo fix, parenthesis, indentation |
| 2 | PEP 8 Compliance | Naming, spacing, structure |
| 3 | Readability Enhancement | Descriptive names, comments |
| 4 | Maintainability Refactor | Function reuse, loop |
| 5 | Performance Optimization | List comprehension |
| 6 | Complexity Reduction | elif chain, dictionary option |