AI Assiginment

Assignment -10.1

Hallticket no:2303A510A6

Batch no:02

Task Description #1 – Syntax and Logic Errors

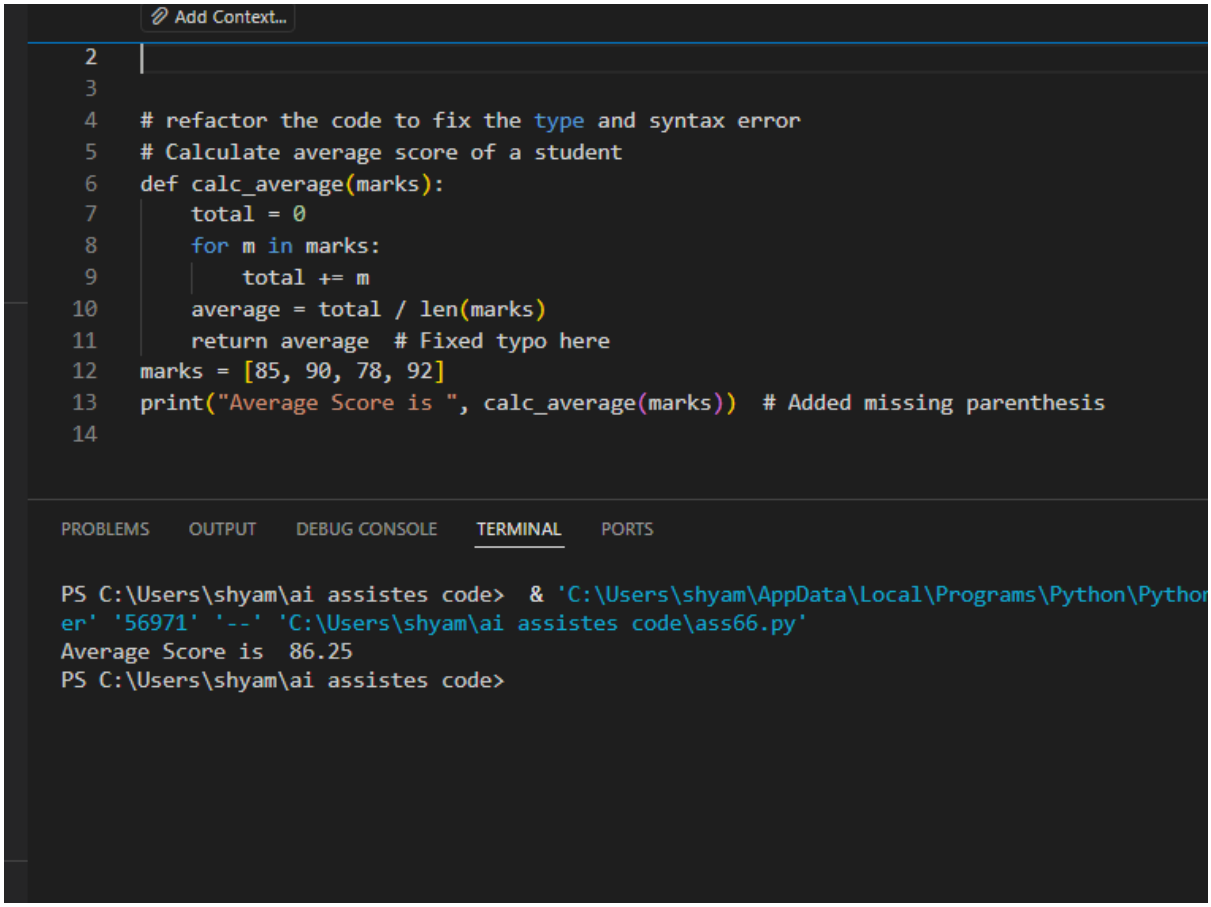
Task: Use AI to identify and fix syntax and logic errors in a faulty Python script.

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.



Indentation was fixed so the code runs properly.

Corrected the typo avrage → average.

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

Now the program is error-free and runnable. The code correctly calculates and prints the average score.

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

```python
# refactor the above code to enhance readability and maintainability well commented and add type hints
def area_of_rect(L,B) : return L*B
print(area_of_rect(10,20))

def area_of_rectangle(length: float, breadth: float) -> float:
    """
    Calculate the area of a rectangle.

    Parameters:
    length (float): The length of the rectangle.
    breadth (float): The breadth of the rectangle.

    Returns:
    float: The area of the rectangle.
    """
    return length * breadth
# Example usage
length = 10.0
breadth = 20.0
area = area_of_rectangle(length, breadth)
print(f"The area of the rectangle with length {length} and breadth {breadth} is: {area}")
```

```
18.0-win32-x64\bundled\libs\debugpy\launcher' '53873' '--' 'C:\Users\shyam\ai assistes code\ass66.py'
200
The area of the rectangle with length 10.0 and breadth 20.0 is: 200.0
PS C:\Users\shyam\ai assistes 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.

```python
# refactor the above code to enhance readability and maintainability well commented and add type hints

def calculate_percentage(value: float, percentage: float) -> float:
    """
    Calculate the percentage of a given value.

    Args:
        value (float): The original value.
        percentage (float): The percentage to calculate.

    Returns:
        float: The calculated percentage of the value.
    """
    return value * percentage / 100
# Example usage
original_value = 200
percentage_to_calculate = 15
result = calculate_percentage(original_value, percentage_to_calculate)
print(result)
```

```
18.0-win32-x64\bundled\libs\debugpy\launcher' '54823' '--' 'C:\Users\shyam\ai assistes code\ass66.py'
30.0
PS C:\Users\shyam\ai assistes 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"]

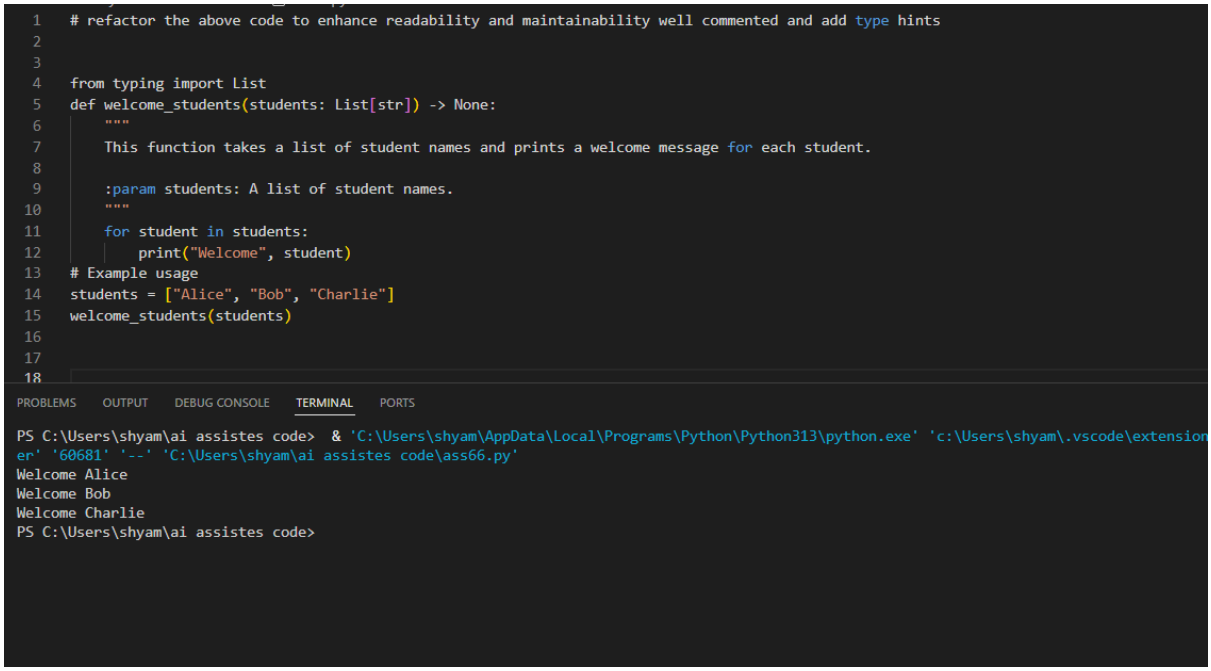
print("Welcome", students[0])

print("Welcome", students[1])

print("Welcome", students[2])

Expected Output:

• Modular code with reusable functions.



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

C: > Users > shyam > ai assistes code > ass66.py

```python

# Refactored code to enhance readability and maintainability with type hints and comments
from typing import List
def generate_squares(nums: List[int]) -> List[int]:
    """
    Generate a list of squares from a list of integers.

    Args:
        nums (List[int]): A list of integers.
    Returns:
        List[int]: A list of squares corresponding to the input integers.
    """
    # Using list comprehension to generate squares for better readability and performance
    return [n ** 2 for n in nums]
# Generate a list of integers from 1 to 999,999
numbers: List[int] = list(range(1, 1000000))
# Generate squares using the defined function
squares: List[int] = generate_squares(numbers)
# Print the length of the squares list to verify the output
print(len(squares))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shyam\ai assistes code>  & 'C:\Users\shyam\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users
er' '51599' '--' 'C:\Users\shyam\ai assistes code\ass66.py'
999999
PS C:\Users\shyam\ai assistes code>
```