ASSIGNMENT-10.2

2303A510i9

BATCH:30

TASK1: Use AI to analyze a Python script and correct all syntax and logical

errors.

Sample Input Code:

def calculate_total(nums)

sum = 0

for n in nums

sum += n

return total

CODE:

AIASS 10.2.PY

```
#Use AI to analyze a Python script and correct all syntax and logical errors.
#def calculate_total(nums)
#sum = 0
#for n in nums
#sum += n
#return total
#Corrected and executable Python code with brief explanations of the identified syntax and logic errors.
"""
Lab 7: Error Debugging with AI - Systematic Approaches to Finding and Fixing Bugs
Week 4 - Monday
This file contains analysis and corrections for 8 common Python bugs.
Each task demonstrates a bug, explains the issue, and provides the corrected code.
"""
def calculate_total(nums):
    """Calculates the total sum of a list of numbers."""
    total = 0  # Initialize total variable to store the sum
    for n in nums:  # Loop through each number in the input list 'nums'
        total += n  # Add the current number 'n' to the total
    return total  # Return the final calculated total
```

JUSTIFICATION:

The original code had syntax errors (missing colons) and a logical mistake (returning an undefined variable). Correcting these ensures the function runs properly and consistently returns the intended sum.

TASK 2: Use AI to refactor Python code to comply with standard coding style

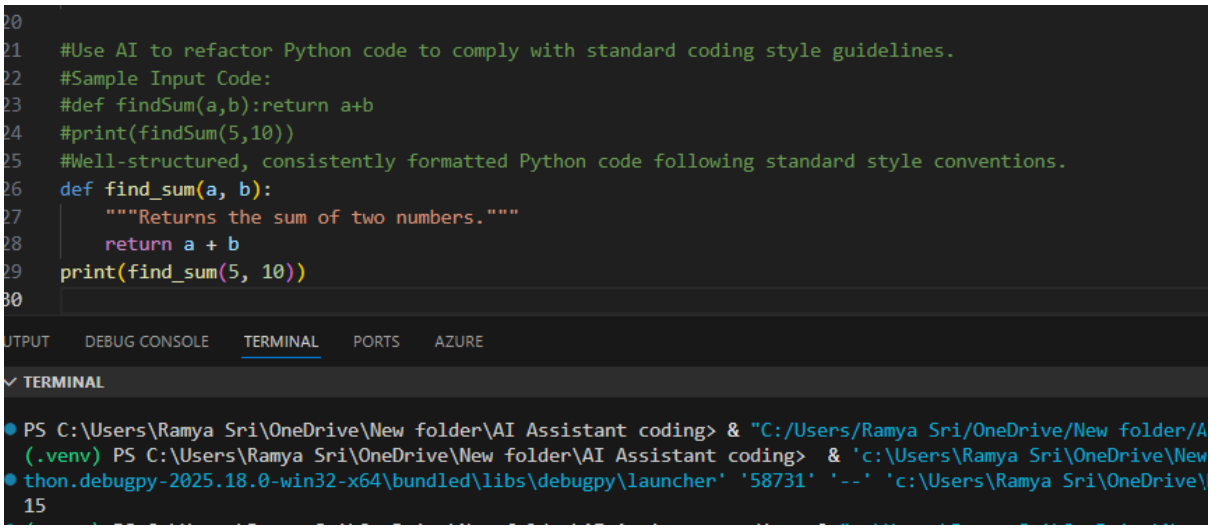
guidelines.

Sample Input Code:

def findSum(a,b):return a+b

print(findSum(5,10))

CODE:



JUSTIFICATION:

Refactoring to PEP 8 standards (snake_case, spacing, multi-line structure) makes the code easier to read, maintain, and share with others, aligning with professional Python practices.

TASK3: Use AI to improve code readability without changing its functionality.

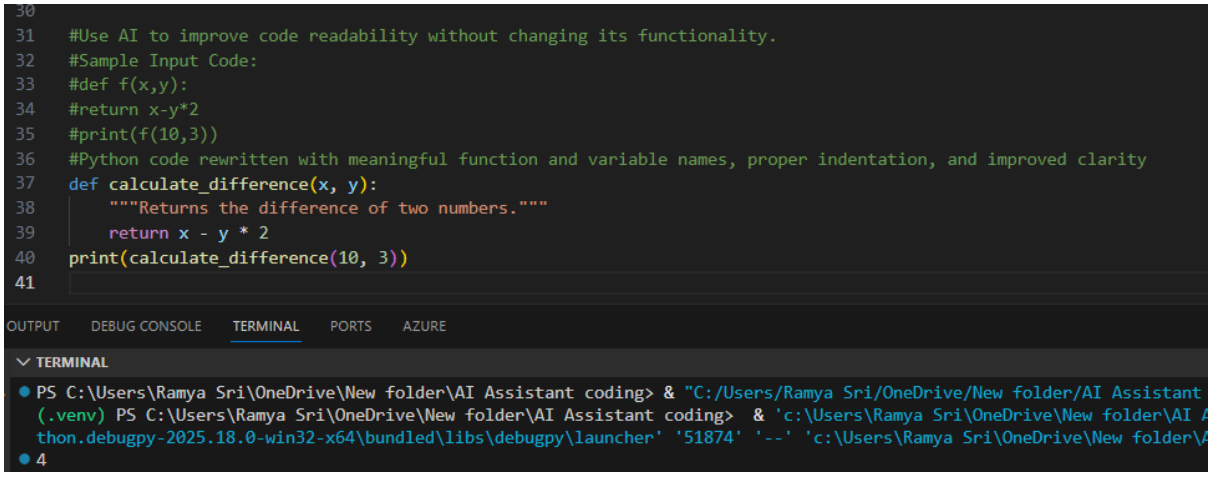
Sample Input Code:

def f(x,y):

return x-y*2

print(f(10,3))

CODE:



JUSTICATION:

Using meaningful names for functions and variables clarifies the purpose of the code. This improves readability and makes it easier for others (or your future self) to understand without guessing.

TASK 4: Use AI to refactor repetitive code into reusable functions.

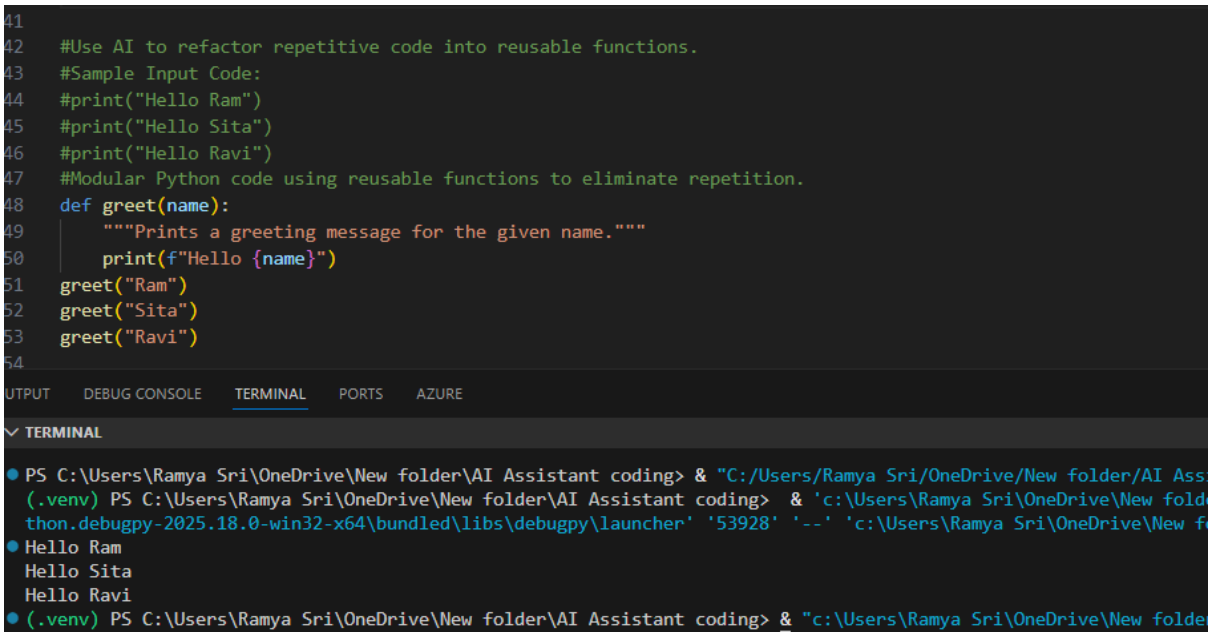
Sample Input Code:

print("Hello Ram")

print("Hello Sita")

print("Hello Ravi")

CODE:



JUSTIFICATION:

By introducing a reusable function, repetitive print statements are eliminated. This modular approach reduces redundancy, improves maintainability, and allows easy scaling if more names need greetings.

Task 5: Use AI to optimize Python code for better performance.

Sample Input Code:

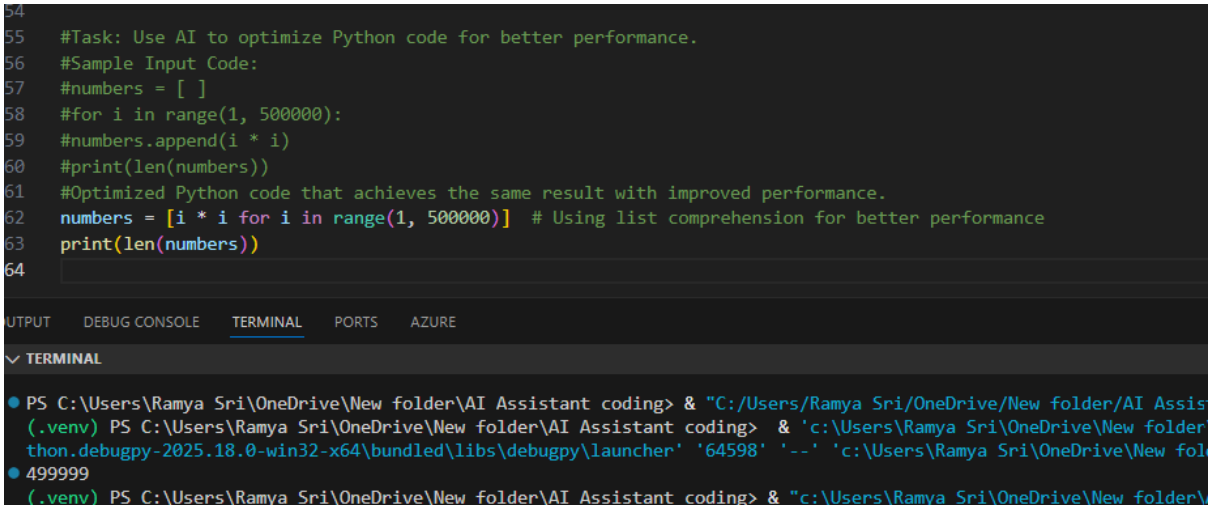
numbers = [ ]

for i in range(1, 500000):

numbers.append(i * i)

print(len(numbers))

CODE:



JUSTIFICATION:

Switching from a loop with .append() to a list comprehension improves performance because comprehensions are optimized internally. The result is the same, but execution is faster and more Pythonic.