

ASSIGNMENT -10.2

NAME: HABEEBA KHANAM

HT.NO:2303A51474

BT.NO:29

Lab 10 – Code Review and Quality: Using AI to Improve Code Quality and Readability

TASK DESCRIPTION -1 (ERROR DETECTION AND CORRECTION)

Task:

Use AI to analyze a Python script and correct all syntax and logical errors.

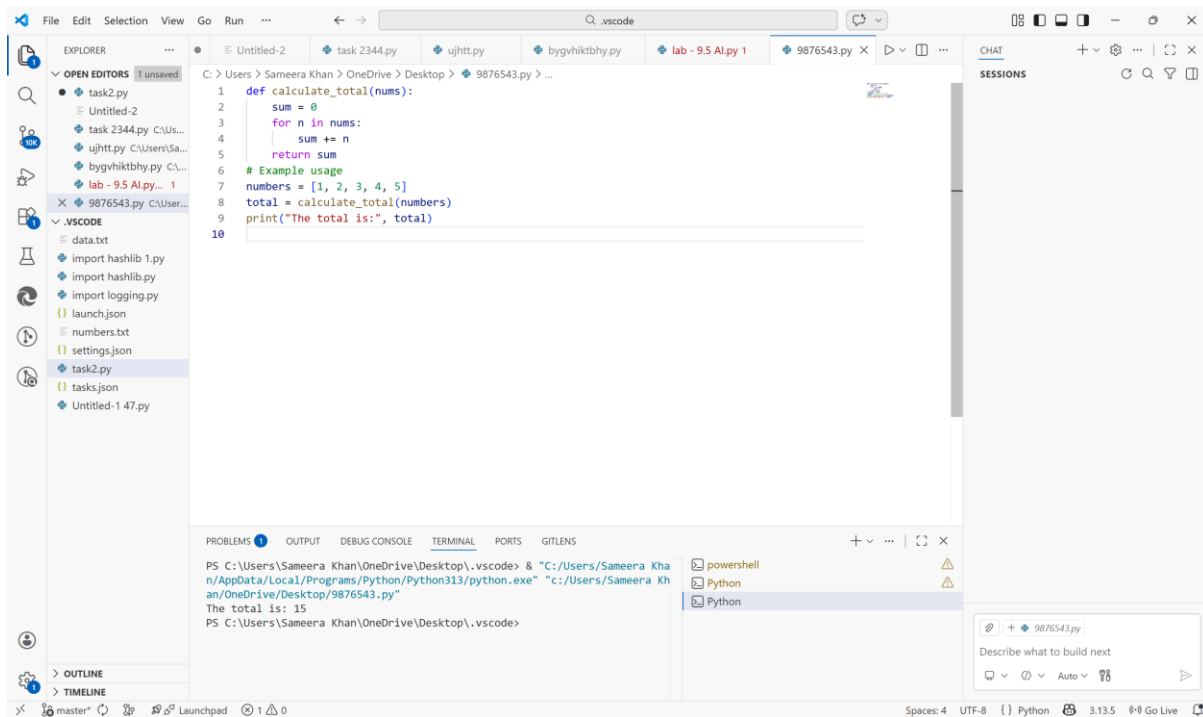
Sample Input Code:

```
def calculate_total(nums)
sum = 0
for n in nums
sum += n
return total
```

Expected Output-1:

Corrected and executable Python code with brief explanations of the identified syntax and logic errors.

CODE:



Errors Identified and Explanation

- Missing Colon (:)
- Indentation Errors
- Missing Colon in for Loop
- Variable Name Mismatch (Logical Error)
- Avoid Using Reserved Name sum

TASK DESCRIPTION -2(CODE STYLESTANDARDIZATION)

Task:

Use AI to refactor Python code to comply with standard coding style guidelines.

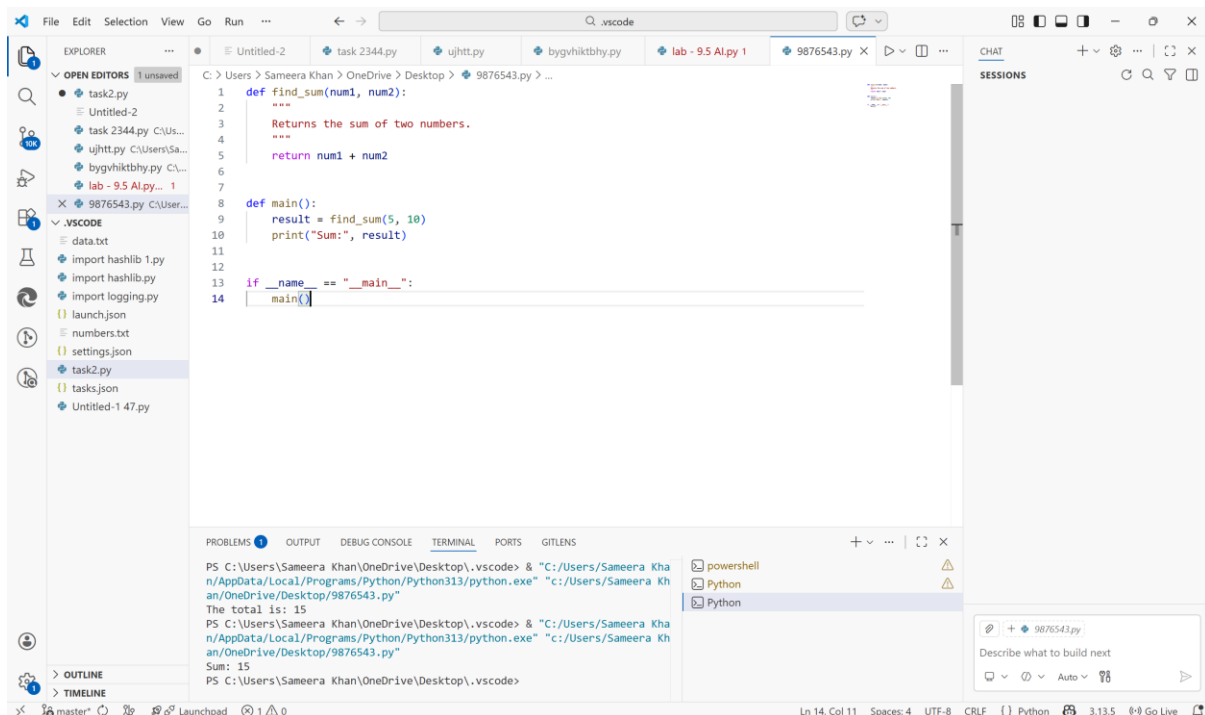
Sample Input Code:

```
def findSum(a,b):return a+b  
print(findSum(5,10))
```

Expected Output-2:

Well-structured, consistently formatted Python code following standard style conventions.

CODE:



Improvements Made According to Coding Standards

- Function Naming (PEP 8)
- Proper Spacing
- Meaningful Variable Names
- Structured Program Flow
- Added Docstring

TASK DESCRIPTION -3(CODE CLARITY IMPROVEMENT)

Task:

Use AI to improve code readability without changing its functionality.

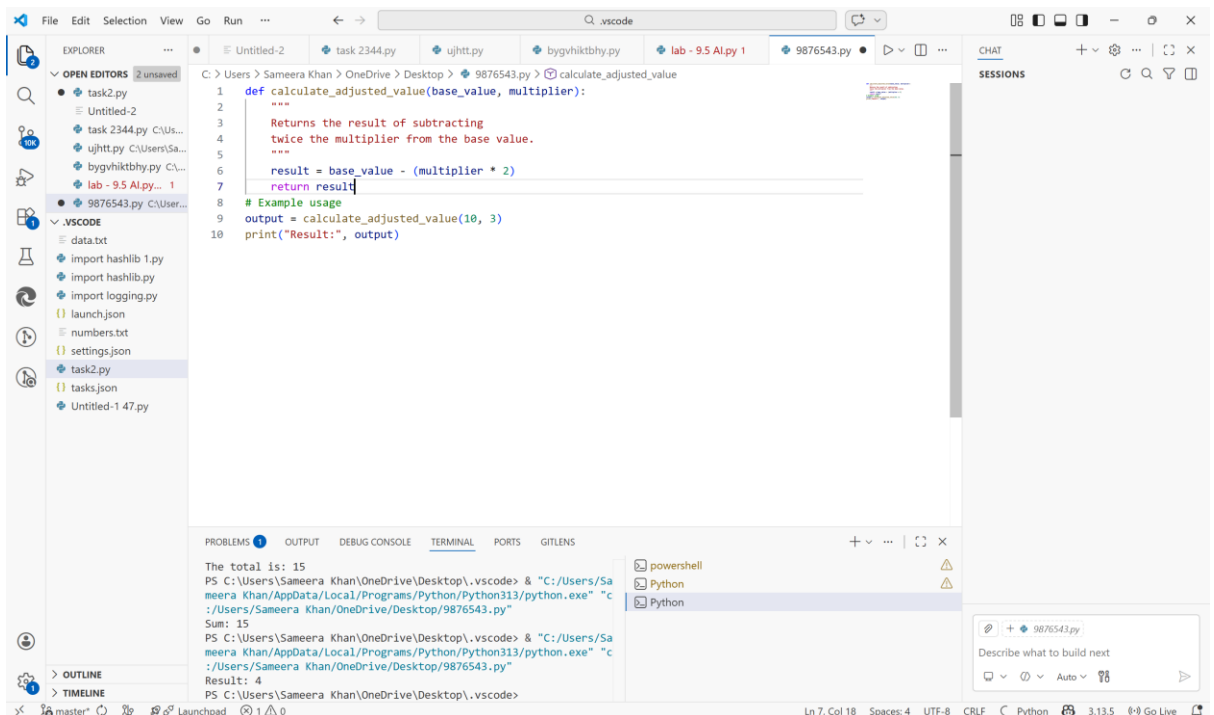
Sample Input Code:

```
def f(x,y):  
    return x-y*2  
print(f(10,3))
```

Expected Output-3:

Python code rewritten with meaningful function and variable names, proper indentation, and improved clarity.

CODE:



Improvements Made:

- Meaningful Function Name
- Descriptive Variable Names
- Proper Indentation & Formatting
- Added Parentheses for Clarity
- `base_value - (multiplier * 2)`
- Added Docstring

TASK DESCRIPTION -4(STRUCTURAL REFACTORING)

Task:

Use AI to refactor repetitive code into reusable functions.

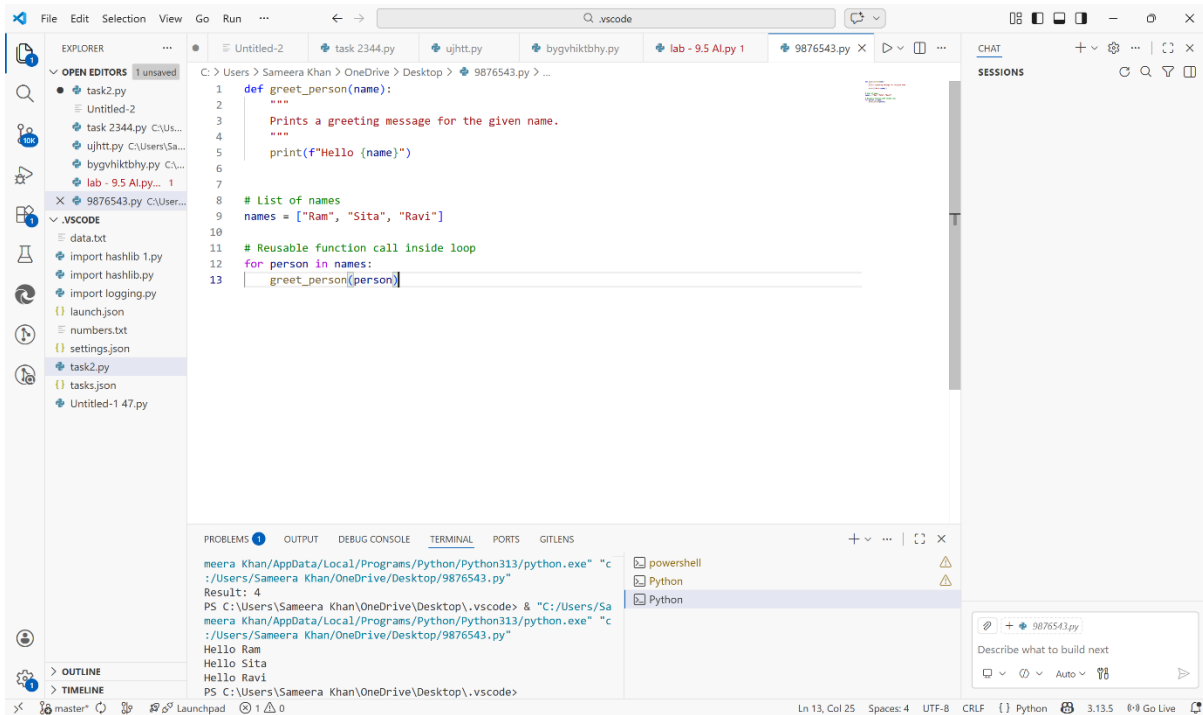
Sample Input Code:

```
print("Hello Ram")  
print("Hello Sita")  
print("Hello Ravi")
```

Expected Output-4:

Modular Python code using reusable functions to eliminate repetition.

CODE:



Improvements Made:

- Eliminated Repetition
- Improved Maintainability
- Used Loop for Scalability
- Modular Programming Approach

TASK DESCRIPTION -5(EFFICIENCY ENHANCEMENT)

Task:

Use AI to optimize Python code for better performance.

Sample Input Code:

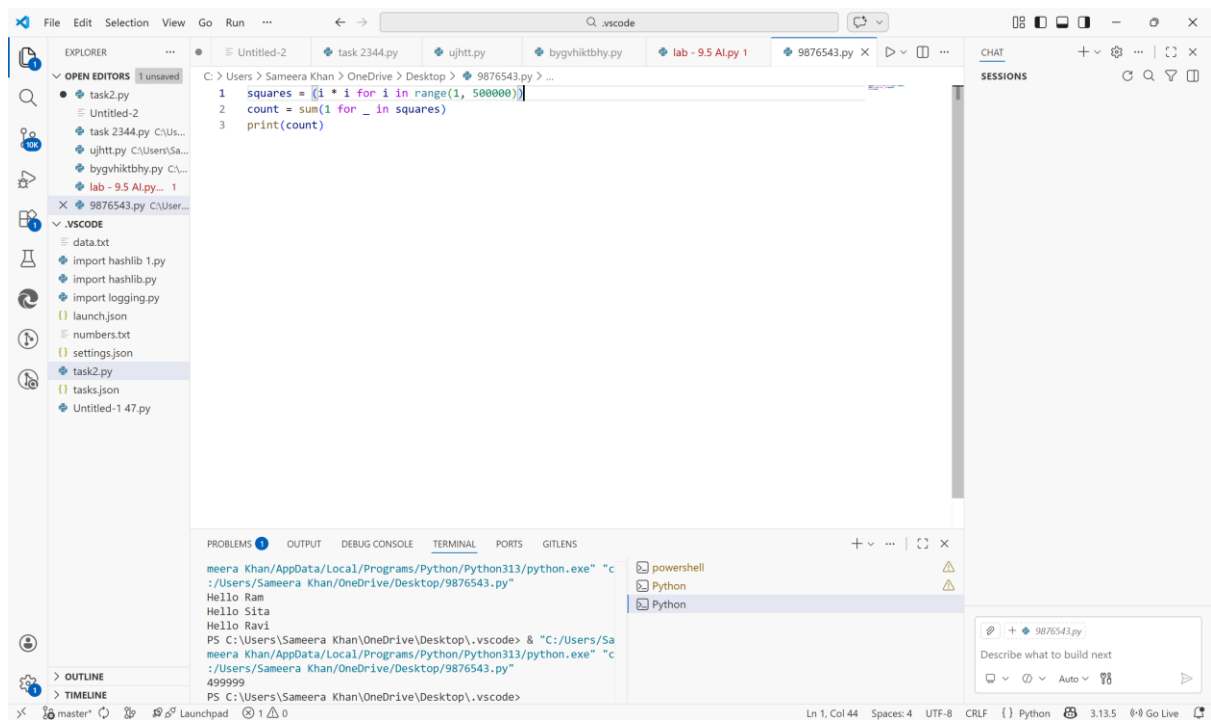
```
numbers = [ ]
```

```
for i in range(1, 500000):  
    numbers.append(i * i)  
print(len(numbers))
```

Expected Output-5:

Optimized Python code that achieves the same result with improved performance.

CODE:



Issues in the Original Code:

- Uses a manual for loop with .append()
- Slower for large datasets
- Not the most Pythonic approach

