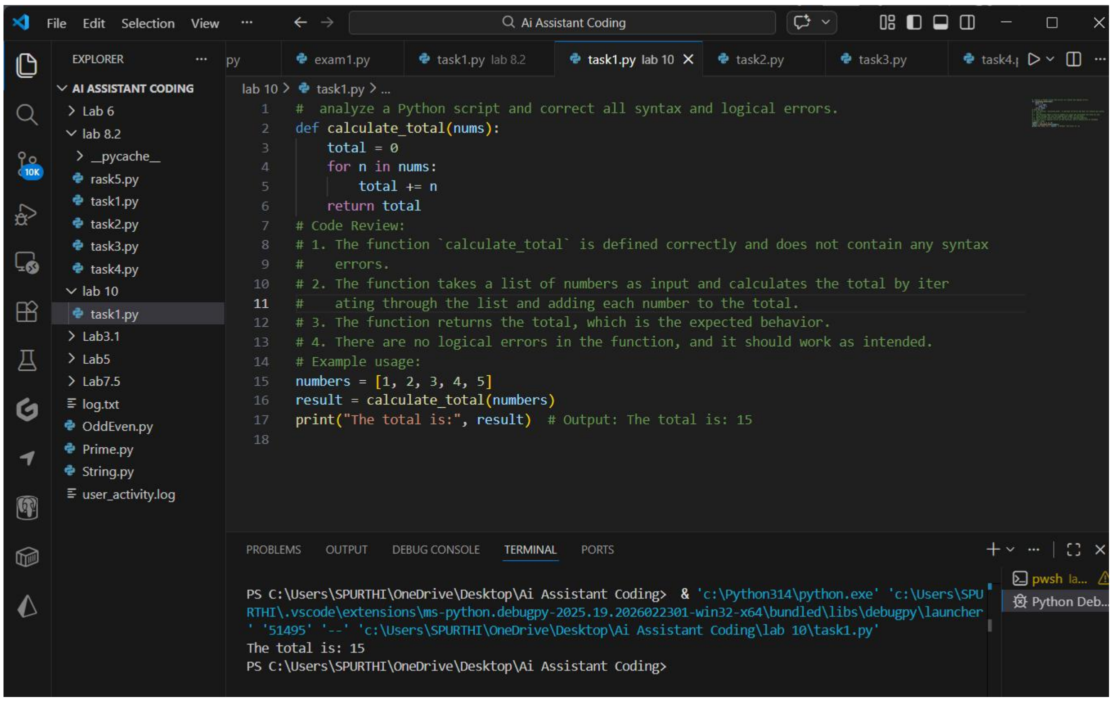
AIAC LAB 10.2
Billa Spurthi
2303A51802
Batch 29

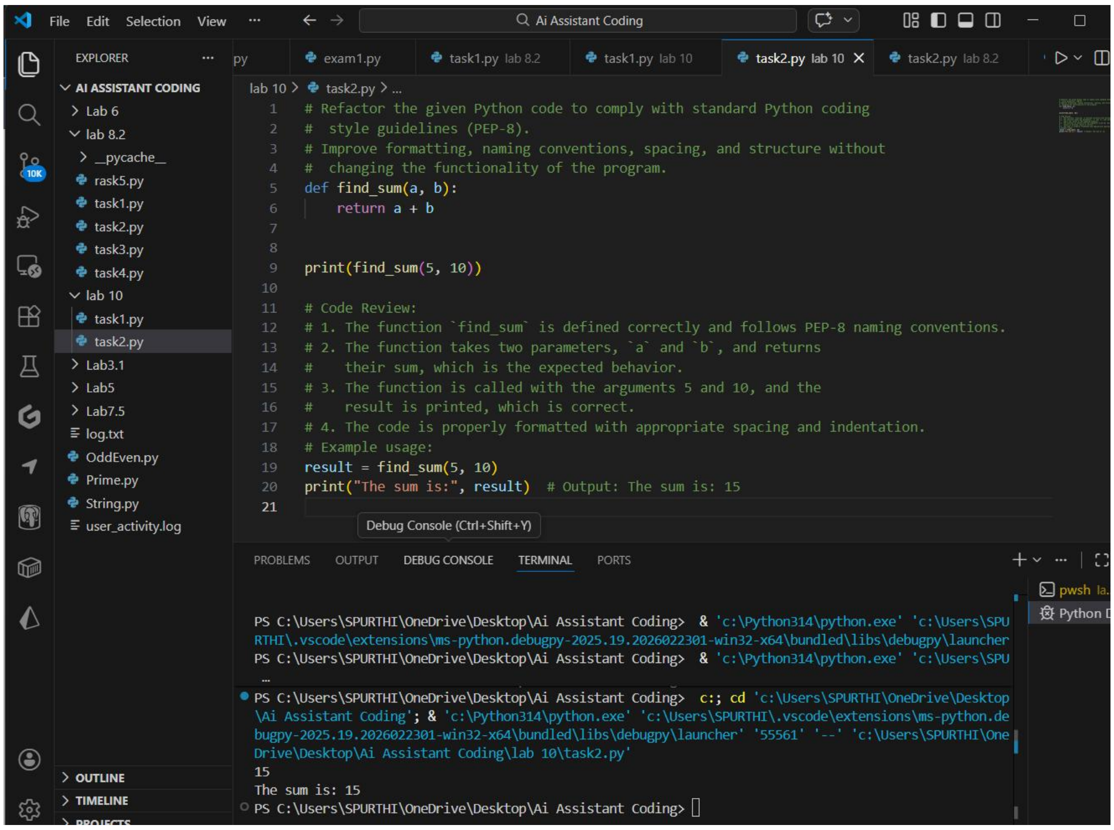
Task:
Use AI to analyze a Python script and correct all syntax and logical errors.



```python
#  analyze a Python script and correct all syntax and logical errors.
def calculate_total(nums):
    total = 0
    for n in nums:
        total += n
    return total
# Code Review:
# 1. The function `calculate_total` is defined correctly and does not contain any syntax
#    errors.
# 2. The function takes a list of numbers as input and calculates the total by iter
#    ating through the list and adding each number to the total.
# 3. The function returns the total, which is the expected behavior.
# 4. There are no logical errors in the function, and it should work as intended.
# Example usage:
numbers = [1, 2, 3, 4, 5]
result = calculate_total(numbers)
print("The total is:", result)  # Output: The total is: 15
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  & 'c:\Python314\python.exe' 'c:\Users\SPU
RTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher
' '51495' '--' 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\lab 10\task1.py'
The total is: 15
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

Task Description -2(Code Style Standardization)
Task:
Use AI to refactor Python code to comply with standard coding style guidelines.



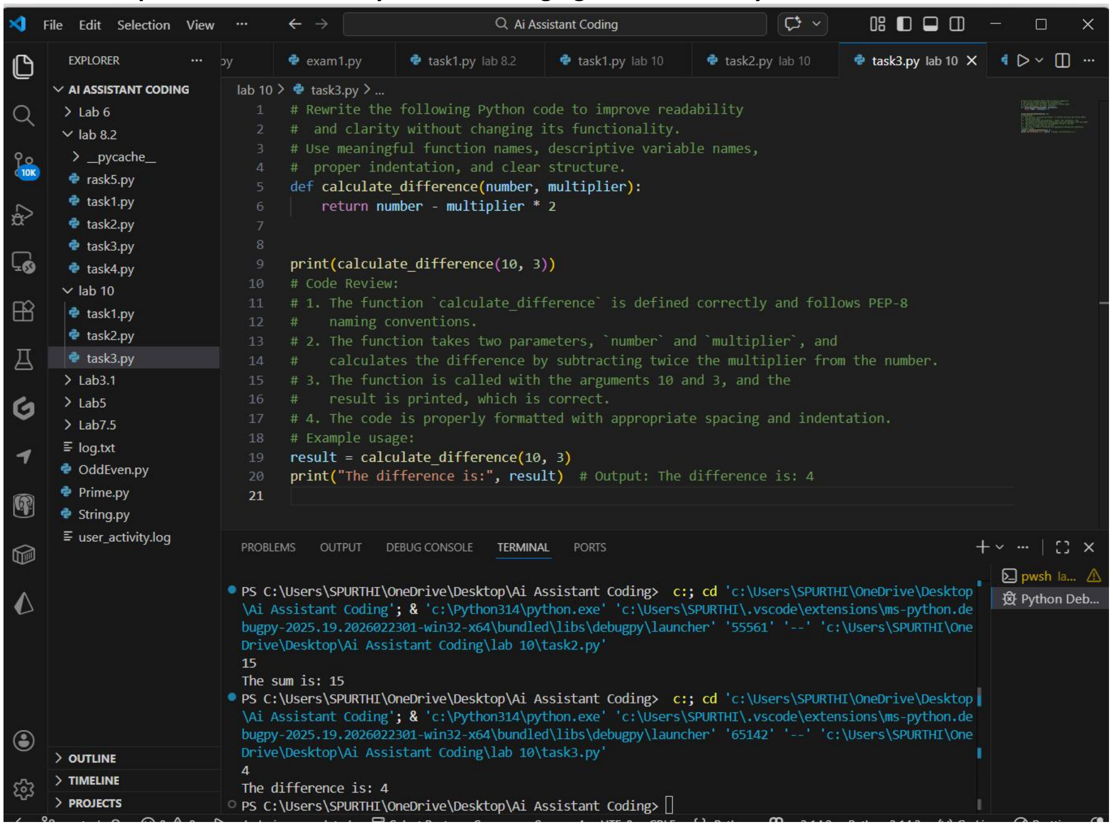
```python
# Refactor the given Python code to comply with standard Python coding
#  style guidelines (PEP-8).
# Improve formatting, naming conventions, spacing, and structure without
#  changing the functionality of the program.
def find_sum(a, b):
    return a + b


print(find_sum(5, 10))

# Code Review:
# 1. The function `find_sum` is defined correctly and follows PEP-8 naming conventions.
# 2. The function takes two parameters, `a` and `b`, and returns
#    their sum, which is the expected behavior.
# 3. The function is called with the arguments 5 and 10, and the
#    result is printed, which is correct.
# 4. The code is properly formatted with appropriate spacing and indentation.
# Example usage:
result = find_sum(5, 10)
print("The sum is:", result)  # Output: The sum is: 15
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  & 'c:\Python314\python.exe' 'c:\Users\SPU
RTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  & 'c:\Python314\python.exe' 'c:\Users\SPU
 …
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop
\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.de
bugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '55561' '--' 'c:\Users\SPURTHI\One
Drive\Desktop\Ai Assistant Coding\lab 10\task2.py'
15
The sum is: 15
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> 
```

Task Description -3(Code Clarity Improvement)
Task:
Use AI to improve code readability without changing its functionality.

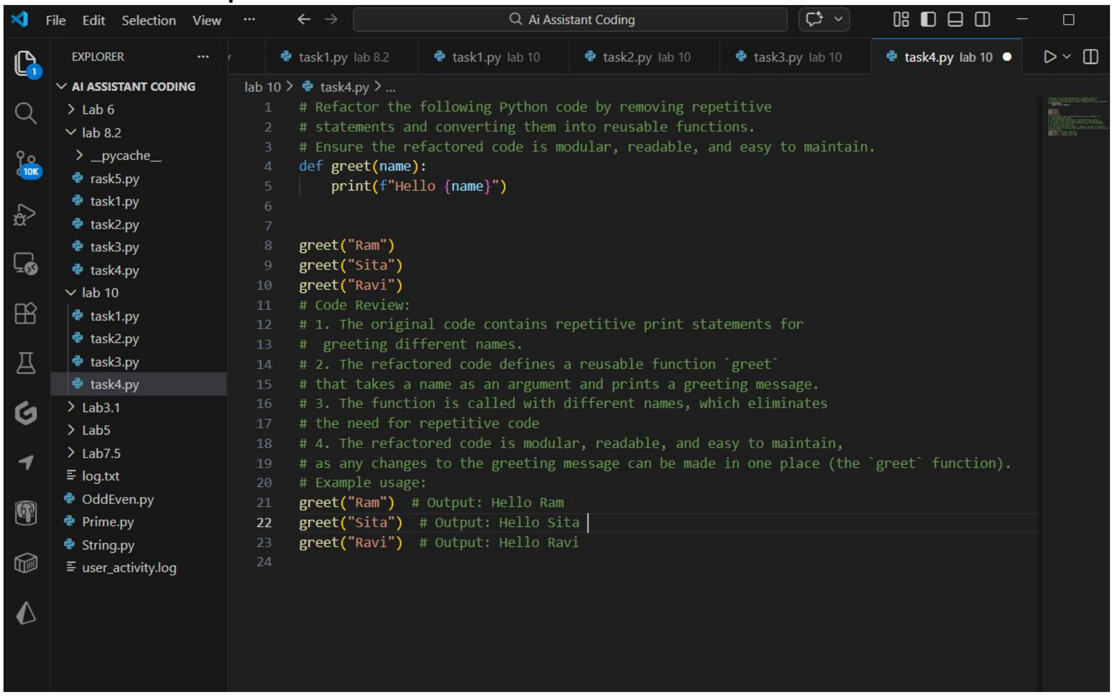


```python
# Rewrite the following Python code to improve readability
#  and clarity without changing its functionality.
# Use meaningful function names, descriptive variable names,
#  proper indentation, and clear structure.
def calculate_difference(number, multiplier):
    return number - multiplier * 2


print(calculate_difference(10, 3))
# Code Review:
# 1. The function `calculate_difference` is defined correctly and follows PEP-8
#    naming conventions.
# 2. The function takes two parameters, `number` and `multiplier`, and
#    calculates the difference by subtracting twice the multiplier from the number.
# 3. The function is called with the arguments 10 and 3, and the
#    result is printed, which is correct.
# 4. The code is properly formatted with appropriate spacing and indentation.
# Example usage:
result = calculate_difference(10, 3)
print("The difference is:", result)  # Output: The difference is: 4
```

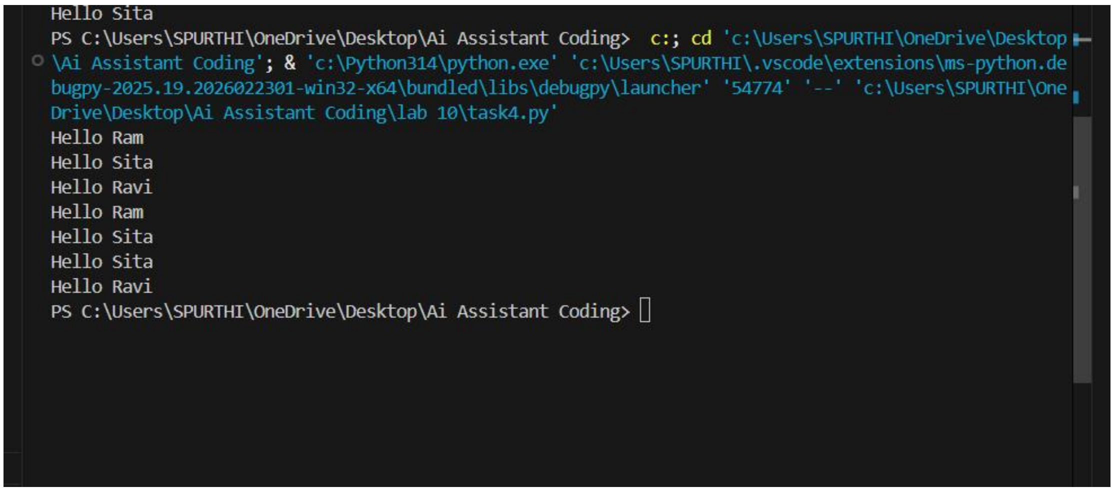
```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop
\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.de
bugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '55561' '--' 'c:\Users\SPURTHI\One
Drive\Desktop\Ai Assistant Coding\lab 10\task2.py'
15
The sum is: 15
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop
\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.de
bugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '65142' '--' 'c:\Users\SPURTHI\One
Drive\Desktop\Ai Assistant Coding\lab 10\task3.py'
4
The difference is: 4
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> 
```

Task Description -4(Structural Refactoring)
Task:
Use AI to refactor repetitive code into reusable functions.



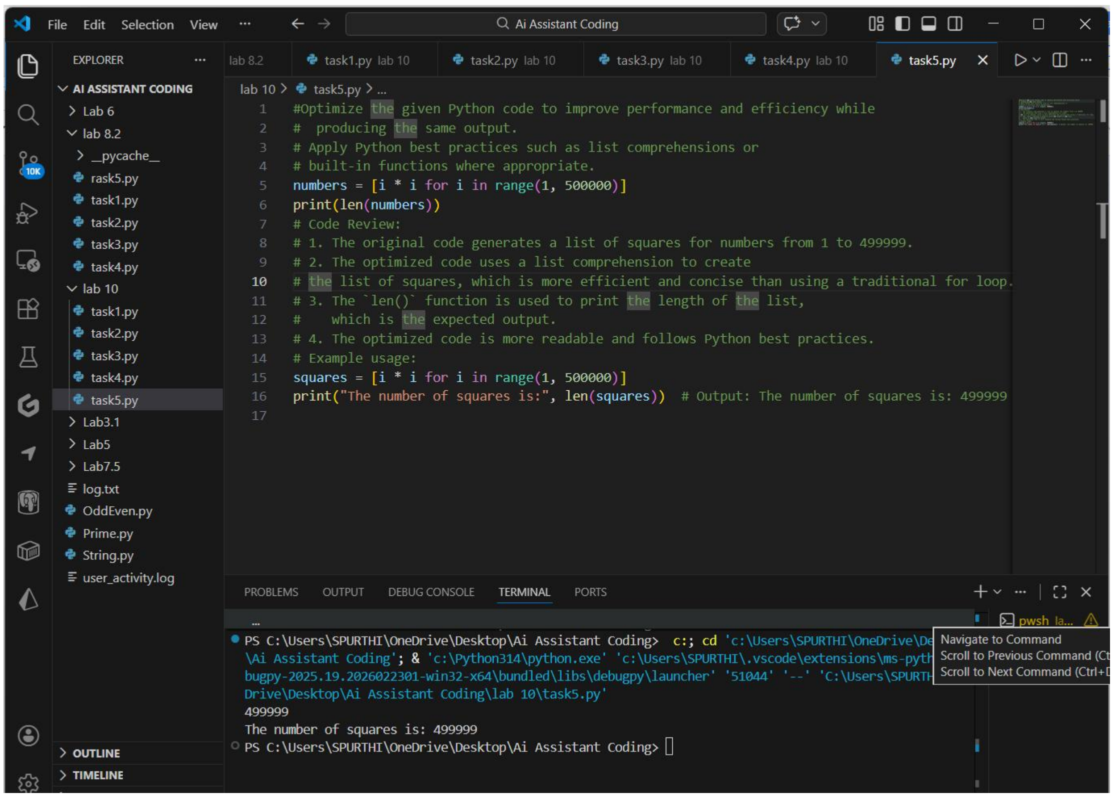
```python
# Refactor the following Python code by removing repetitive
# statements and converting them into reusable functions.
# Ensure the refactored code is modular, readable, and easy to maintain.
def greet(name):
    print(f"Hello {name}")


greet("Ram")
greet("Sita")
greet("Ravi")
# Code Review:
# 1. The original code contains repetitive print statements for
#  greeting different names.
# 2. The refactored code defines a reusable function `greet`
# that takes a name as an argument and prints a greeting message.
# 3. The function is called with different names, which eliminates
# the need for repetitive code
# 4. The refactored code is modular, readable, and easy to maintain,
# as any changes to the greeting message can be made in one place (the `greet` function).
# Example usage:
greet("Ram")  # Output: Hello Ram
greet("Sita")  # Output: Hello Sita
greet("Ravi")  # Output: Hello Ravi
```



```
Hello Sita
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop
\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.de
bugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '54774' '--' 'c:\Users\SPURTHI\One
Drive\Desktop\Ai Assistant Coding\lab 10\task4.py'
Hello Ram
Hello Sita
Hello Ravi
Hello Ram
Hello Sita
Hello Sita
Hello Ravi
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

Task Description -5(Efficiency Enhancement)
Task:
Use AI to optimize Python code for better performance.



```python
#Optimize the given Python code to improve performance and efficiency while
#  producing the same output.
# Apply Python best practices such as list comprehensions or
# built-in functions where appropriate.
numbers = [i * i for i in range(1, 500000)]
print(len(numbers))
# Code Review:
# 1. The original code generates a list of squares for numbers from 1 to 499999.
# 2. The optimized code uses a list comprehension to create
# the list of squares, which is more efficient and concise than using a traditional for loop.
# 3. The `len()` function is used to print the length of the list,
#    which is the expected output.
# 4. The optimized code is more readable and follows Python best practices.
# Example usage:
squares = [i * i for i in range(1, 500000)]
print("The number of squares is:", len(squares))  # Output: The number of squares is: 499999
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\De
\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-pyth
bugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '51044' '--' 'C:\Users\SPURTH
Drive\Desktop\Ai Assistant Coding\lab 10\task5.py'
499999
The number of squares is: 499999
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```