AIAC-LAB ASSIGNMENT

Lab 10.2 – Code Review and Quality: Using AI to Improve Code Quality and Readability

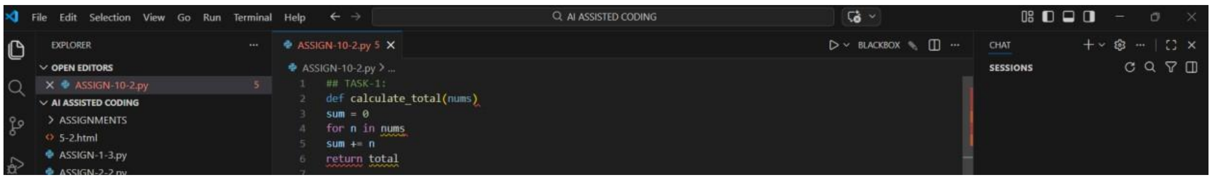
Name: P.Vyshnavi

2403a51l34

B-52

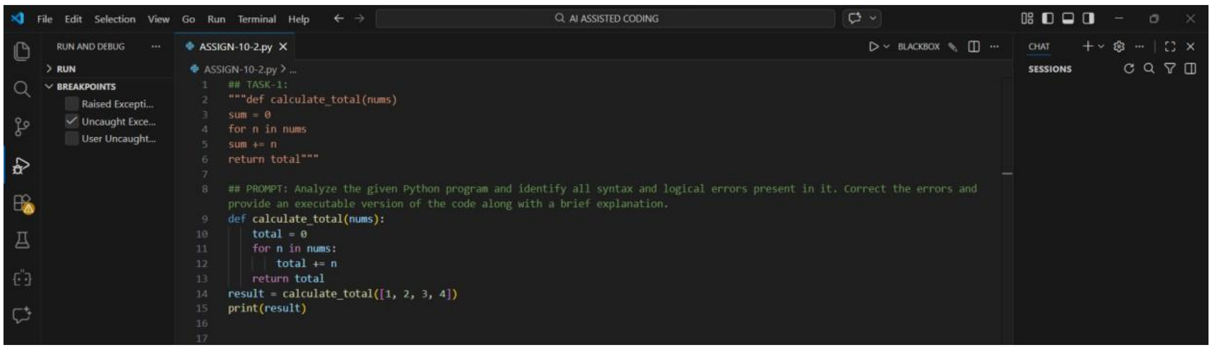
TASK-1: Error Detection and Correction

Sample Input Code:



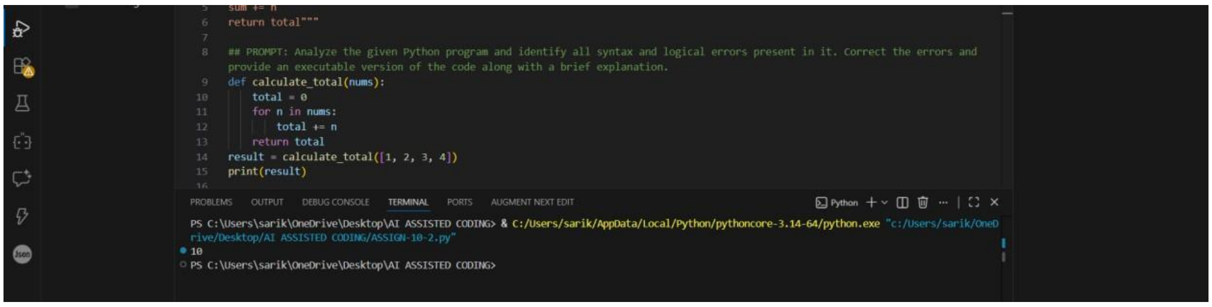
```
## TASK-1:
def calculate_total(nums)
sum = 0
for n in nums
sum += n
return total
```

Prompt: Analyse the given Python program and identify all syntax and logical errors present in it. Correct the errors and provide an executable version of the code along with a brief explanation. **Corrected Input Code:**



```
## TASK-1:
"""def calculate_total(nums)
sum = 0
for n in nums
sum += n
return total"""

## PROMPT: Analyze the given Python program and identify all syntax and logical errors present in it. Correct the errors and provide an executable version of the code along with a brief explanation.
def calculate_total(nums):
    total = 0
    for n in nums:
        total += n
    return total
result = calculate_total([1, 2, 3, 4])
print(result)
```

Output:

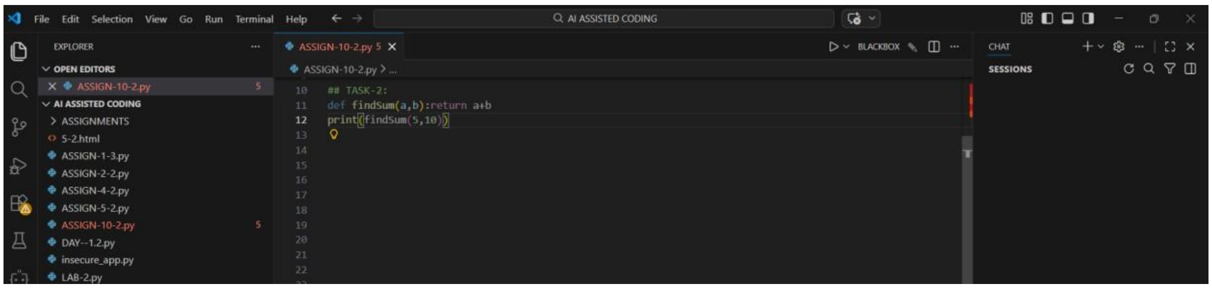


```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
10
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

Explanation: The original code contained syntax errors such as missing colons and improper indentation, which prevented execution. A logical error was also present where an undefined variable was returned. These issues were corrected to make the function executable and reliable.

TASK-2: Code Style Standardization Sample

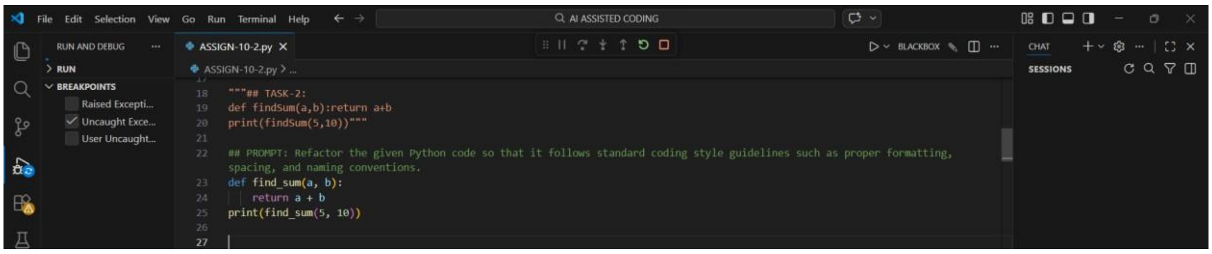
Input Code:



```python
## TASK-2:
def findSum(a,b):return a+b
print(findSum(5,10))
```

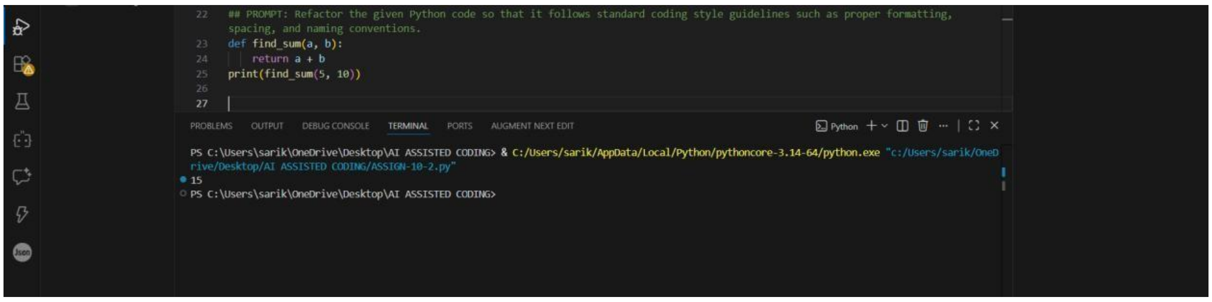
Prompt: Refactor the given Python code so that it follows standard coding style guidelines such as proper formatting, spacing, and naming conventions.

Corrected Input Code:



```python
"""## TASK-2:
def findSum(a,b):return a+b
print(findSum(5,10))"""

## PROMPT: Refactor the given Python code so that it follows standard coding style guidelines such as proper formatting,
spacing, and naming conventions.
def find_sum(a, b):
    return a + b
print(find_sum(5, 10))
```

Output:



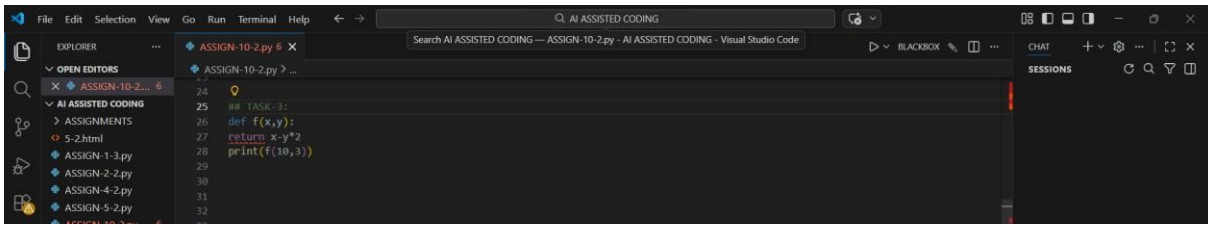
```python
## PROMPT: Refactor the given Python code so that it follows standard coding style guidelines such as proper formatting,
spacing, and naming conventions.
def find_sum(a, b):
    return a + b
print(find_sum(5, 10))
```

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneD
rive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
15
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

Explanation: The code was reformatted to follow Python's PEP 8 style guidelines by improving spacing, naming conventions, and structure. These changes enhance readability and make the code easier to maintain without altering its functionality.

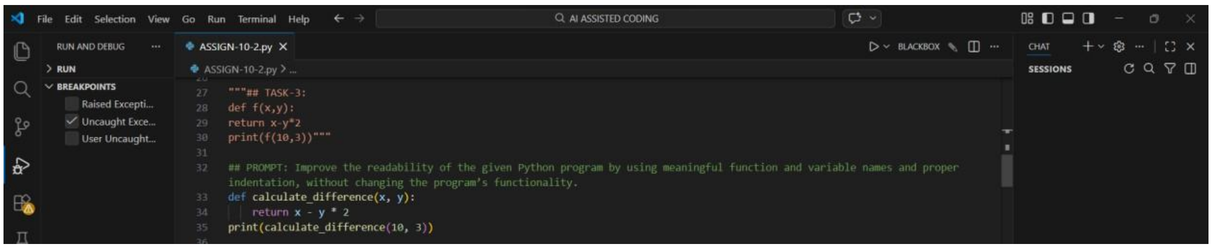
TASK-3: Code Clarity Improvement Sample

Input Code:



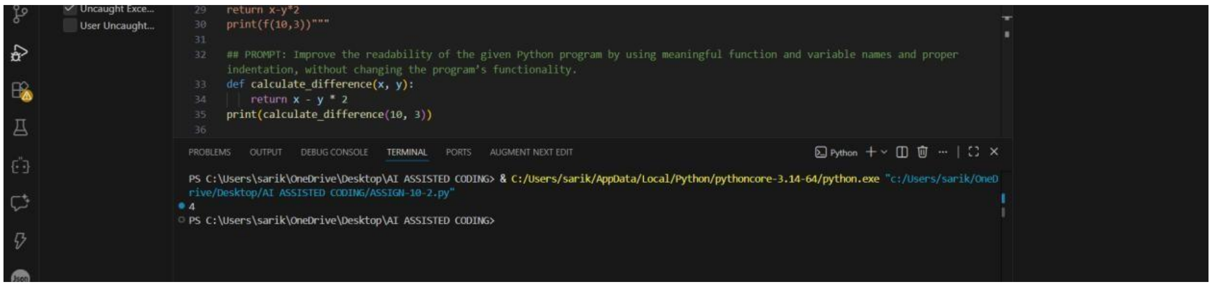
```
## TASK-3:
def f(x,y):
return x-y*2
print(f(10,3))
```

Prompt: Improve the readability of the given Python program by using meaningful function and variable names and proper indentation, without changing the program's functionality. **Corrected Input Code:**



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

## PROMPT: Improve the readability of the given Python program by using meaningful function and variable names and proper
indentation, without changing the program's functionality.
def calculate_difference(x, y):
    return x - y * 2
print(calculate_difference(10, 3))
```

Output:



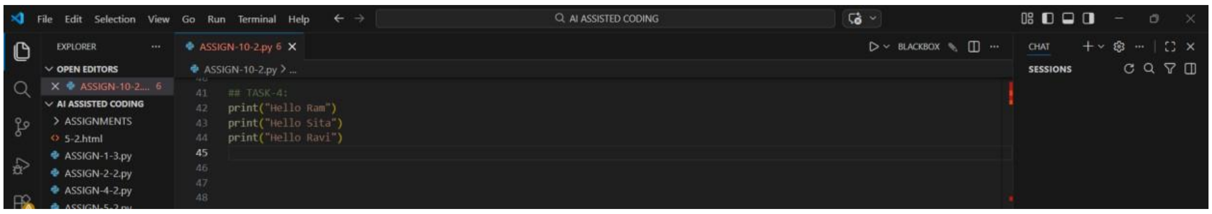
```
return x-y*2
print(f(10,3))"""

## PROMPT: Improve the readability of the given Python program by using meaningful function and variable names and proper
indentation, without changing the program's functionality.
def calculate_difference(x, y):
    return x - y * 2
print(calculate_difference(10, 3))
```

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneD
rive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
4
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

Explanation: The function and variable names were updated to clearly describe their purpose, making the code easier to understand. Proper indentation and clearer expressions were also applied while preserving the original logic.

TASK-4: Structural Refactoring Sample Input

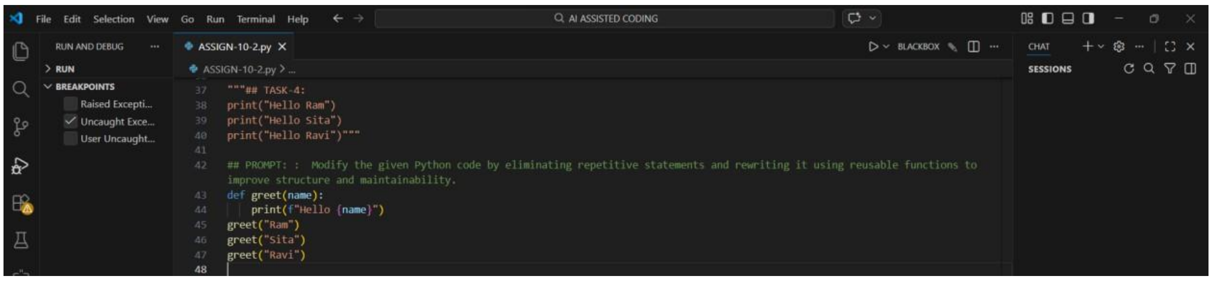
Code:



```python
## TASK-4:
print("Hello Ram")
print("Hello Sita")
print("Hello Ravi")
```

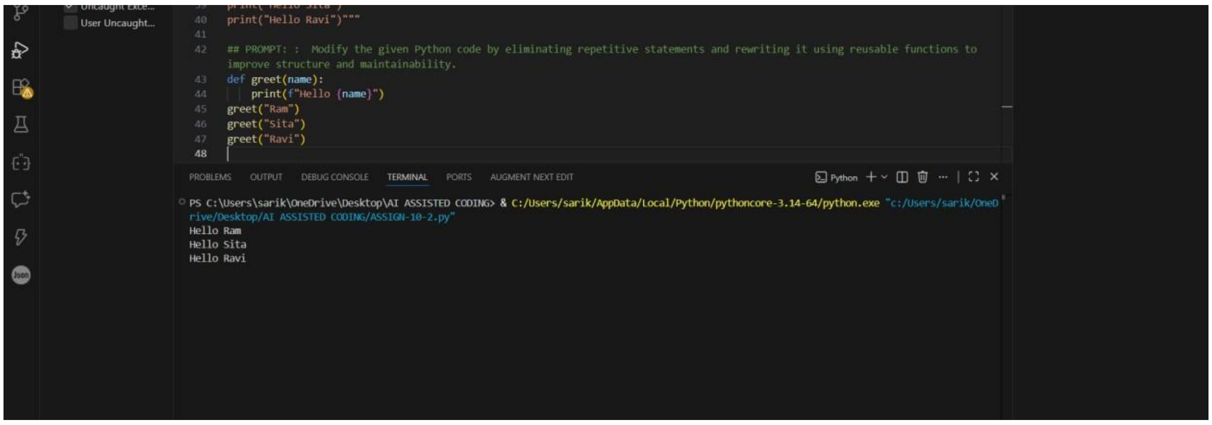
Prompt: Modify the given Python code by eliminating repetitive statements and rewriting it using reusable functions to improve structure and maintainability.

Corrected Input Code:



```python
"""## TASK-4:
print("Hello Ram")
print("Hello Sita")
print("Hello Ravi")"""

## PROMPT: :  Modify the given Python code by eliminating repetitive statements and rewriting it using reusable functions to
improve structure and maintainability.
def greet(name):
    print(f"Hello {name}")
greet("Ram")
greet("Sita")
greet("Ravi")
```

Output:

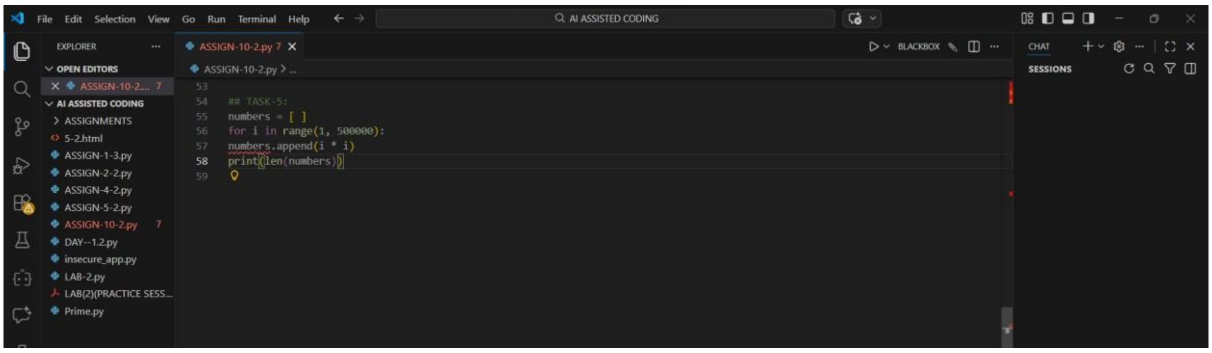


```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneD
rive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
Hello Ram
Hello Sita
Hello Ravi
```

Explanation: The repetitive print statements were replaced with a reusable function that accepts a name as input. This approach reduces redundancy and makes the code more scalable and easier to modify.

TASK-5: Efficiency Enhancement Sample

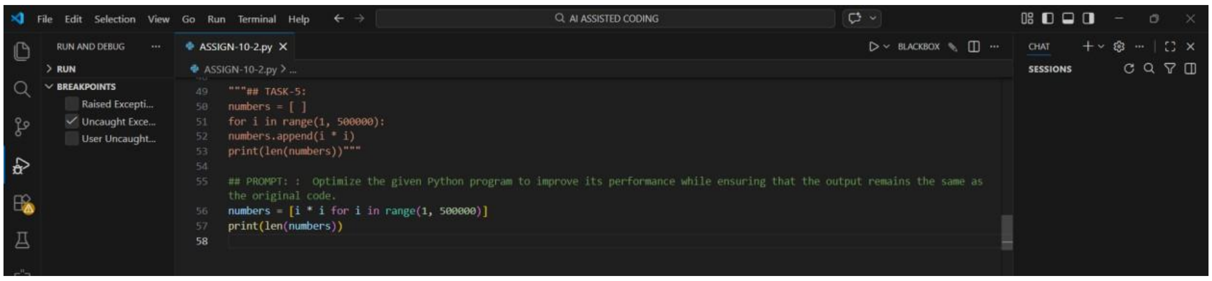
Input Code:



```python
## TASK-5:
numbers = [ ]
for i in range(1, 500000):
numbers.append(i * i)
print(len(numbers))
```

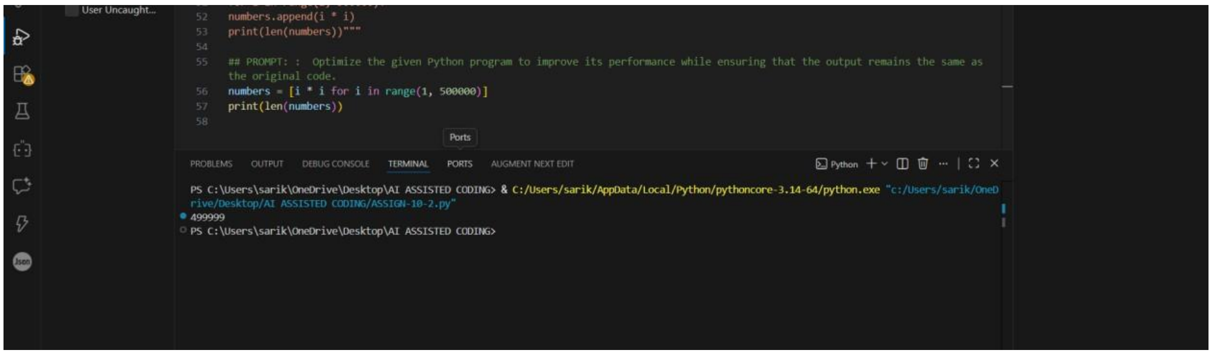
Prompt: Optimize the given Python program to improve its performance while ensuring that the output remains the same as the original code.

Corrected Input Code:



```python
"""## TASK-5:
numbers = [ ]
for i in range(1, 500000):
numbers.append(i * i)
print(len(numbers))"""

## PROMPT: :  Optimize the given Python program to improve its performance while ensuring that the output remains the same as the original code.
numbers = [i * i for i in range(1, 500000)]
print(len(numbers))
```

Output:



```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
499999
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

Explanation: The loop-based implementation was optimized using a list comprehension, which is faster and more concise in Python. This improves performance while producing the same output as the original code.