ASSIGNMENT-10.2

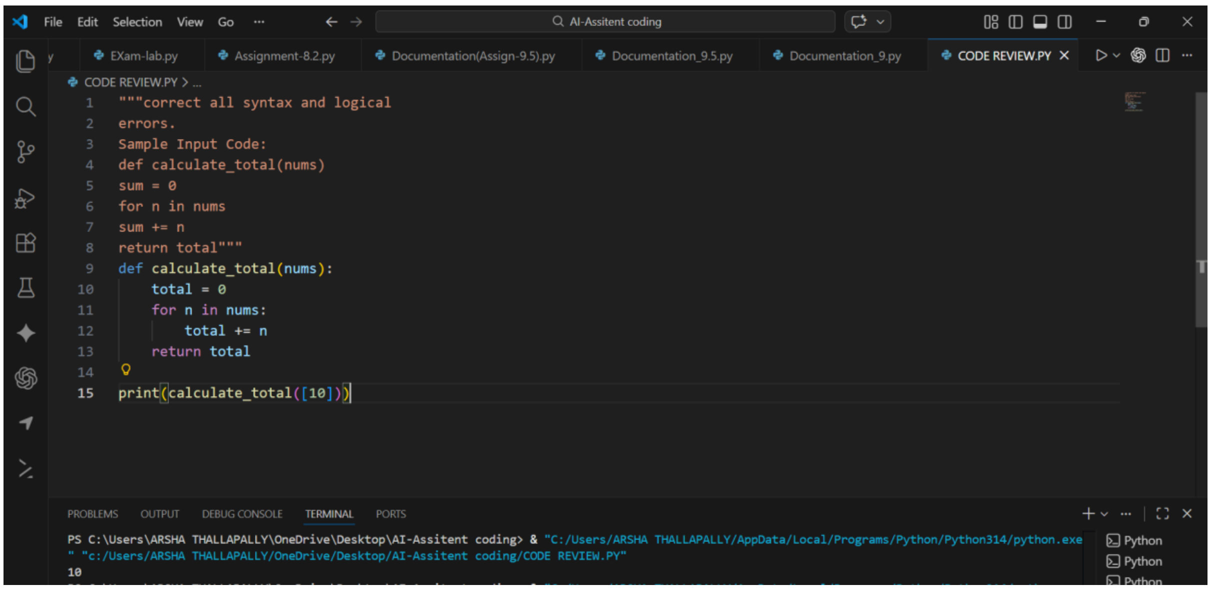
2303A51600

BATCH-29

TASK-1

PROMPT: correct all syntax and logical errors

CODE:



```python
"""correct all syntax and logical
errors.
Sample Input Code:
def calculate_total(nums)
sum = 0
for n in nums
sum += n
return total"""
def calculate_total(nums):
    total = 0
    for n in nums:
        total += n
    return total

print(calculate_total([10]))
```

```
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding> & "C:/Users/ARSHA THALLAPALLY/AppData/Local/Programs/Python/Python314/python.exe
" "c:/Users/ARSHA THALLAPALLY/OneDrive/Desktop/AI-Assitent coding/CODE REVIEW.PY"
10
```

OBSERVATION:

Function names should follow **snake_case** as per PEP 8 standards.

Proper **indentation and spacing** improve readability.

One statement per line avoids confusion and errors.

Code becomes easier to maintain and debug.

TASK-2

PROMPT: comply with standard coding style

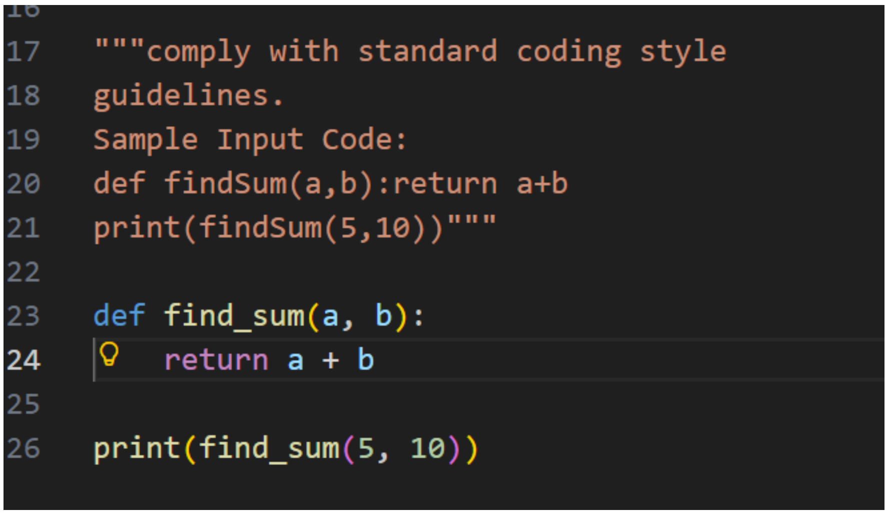
guidelines.

Sample Input Code:

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

print(findSum(5,10))

CODE:



```
"""comply with standard coding style
guidelines.
Sample Input Code:
def findSum(a,b):return a+b
print(findSum(5,10))"""

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

print(find_sum(5, 10))
```

OBSERVATION:

Descriptive function names make the purpose clear.

Proper indentation avoids logical mistakes.

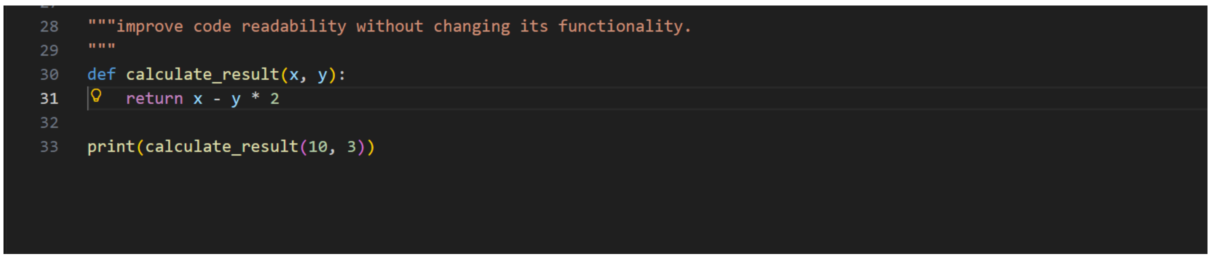
Spaces around operators increase clarity.

Readable code is easier for others to understand and modify.

TASK-3

PROMPT: improve code readability without changing its functionality.

CODE:



```
"""improve code readability without changing its functionality.
"""
def calculate_result(x, y):
    return x - y * 2

print(calculate_result(10, 3))
```

OBSERVAATION:

Repetition is reduced by using a single reusable function.

Changes need to be made only in one place (better maintainability).

Code becomes more modular and structured.

Improves scalability when more inputs are added.

TASK-4

PROMPT:

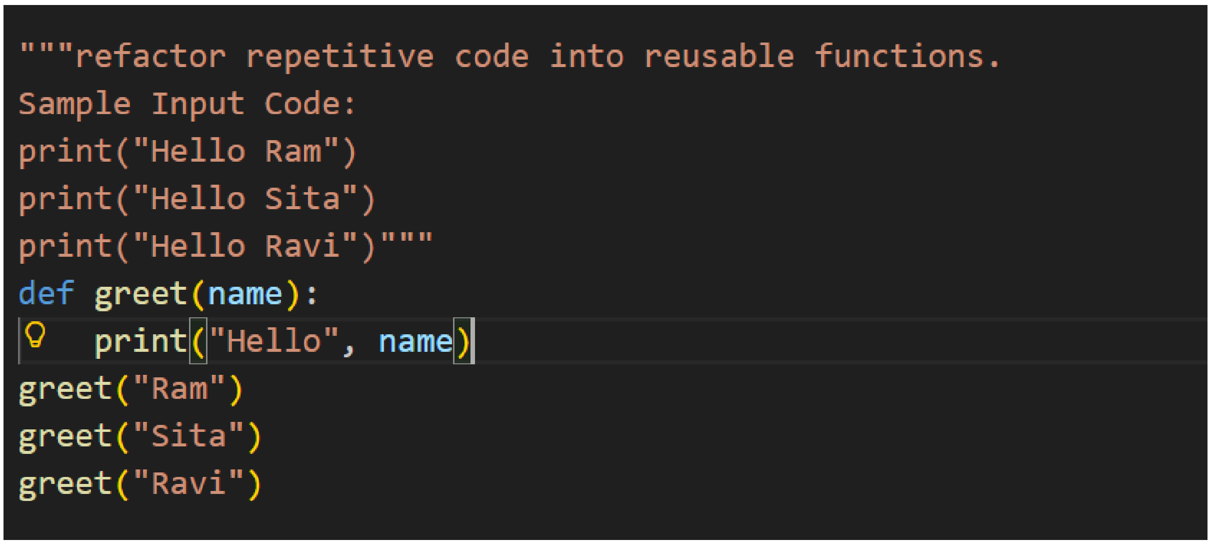
CODE: refactor repetitive code into reusable functions.

Sample Input Code:

print("Hello Ram")

print("Hello Sita")

print("Hello Ravi")



```python
"""refactor repetitive code into reusable functions.
Sample Input Code:
print("Hello Ram")
print("Hello Sita")
print("Hello Ravi")"""
def greet(name):
    print("Hello", name)
greet("Ram")
greet("Sita")
greet("Ravi")
```

OBSERVATION:

List comprehension is faster than using a loop with .append().

Reduces execution time and lines of code.

Minimizes function call overhead.

Efficient memory usage improves performance for large data.

TASK-5

PROMPT: optimize Python code for better performance.

Sample Input Code:

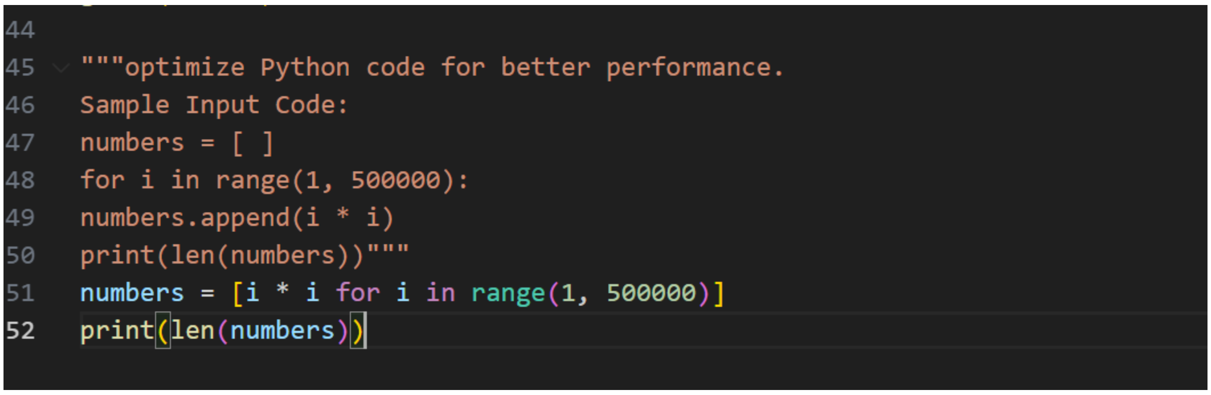
numbers = [ ]

for i in range(1, 500000):

numbers.append(i * i)

print(len(numbers))

CODE:



```
"""optimize Python code for better performance.
Sample Input Code:
numbers = [ ]
for i in range(1, 500000):
numbers.append(i * i)
print(len(numbers))"""
numbers = [i * i for i in range(1, 500000)]
print(len(numbers))
```

OBSERVATION:

Function input type must match expected data type.

Lists are required when iteration is performed.

Clear parameter design avoids runtime errors.

Logic must match problem intent (list sum vs range sum).