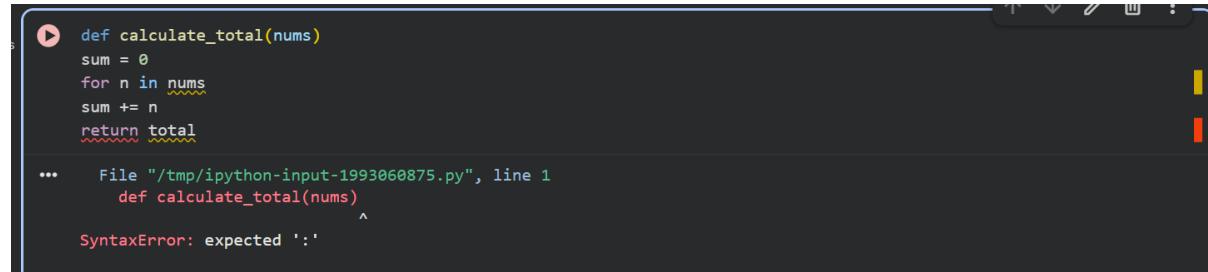


School of Computer Science and Artificial Intelligence**Lab Assignment # 10.2**

Program : B. Tech (CSE)
Specialization : -
Course Title : AI Assisted Coding
Course Code : 23CS002PC304
Semester : II
Academic Session : 2025-2026
Name of Student : P.Harshavardhan
Enrollment No. : 2403A51L37
Batch No. : 52
Date : 10/02/26

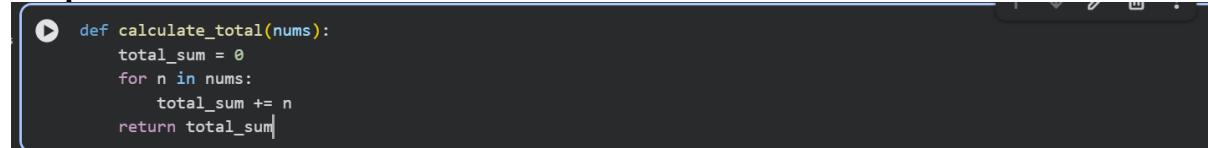
Submission Starts here**Screenshots:****Task Description -1(Error Detection and Correction)****Code used:**

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

File "/tmp/ipython-input-1993060875.py", line 1
    def calculate_total(nums)
                           ^
SyntaxError: expected ':'
```

A screenshot of a Jupyter Notebook cell. The code defines a function 'calculate_total' that sums up a list of numbers. There is a syntax error: the colon at the end of the 'def' statement is missing. The error message 'SyntaxError: expected ':'' is displayed at the bottom of the cell.**Prompt:**

Correct the code and give brief explanations of the identified syntax and logic errors.

Output:

```
def calculate_total(nums):
    total_sum = 0
    for n in nums:
        total_sum += n
    return total_sum
```

The code has been corrected. The variable 'sum' is renamed to 'total_sum' to avoid shadowing Python's built-in 'sum()' function. The function now returns the correct total sum of the input list.**Explanation:**

- 1) Missing colons: Python requires a colon : at the end of def (function definition) and for (loop definition) statements.
- 2) Variable shadowing: I renamed the sum variable to total_sum to avoid overwriting Python's built-in sum() function, which can lead to unexpected behavior.
- 3) Undefined variable: The function was trying to return total, but the variable was named sum (or total_sum after the fix). I've corrected it to return total_sum.

Task Description -2(Code Style Standardization)

Code Used:

```
▶ def findSum(a,b):return a+b
  print(findSum(5,10))
...
... 15
```

Prompt:

convert the code to follow standard style conventions.

Output:

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

  print(find_sum(5, 10))
...
... 15
```

Task Description -3(Code Clarity Improvement)

Code used:

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

...
File "/tmp/ipython-input-2982719317.py", line 2
    return x-y*2
      ^
IndentationError: expected an indented block after function definition on line 1
```

Prompt:

rewrite the code with meaningful function and variable names, proper indentation, and improved clarity.

Output:

```
▶ def calculate_value(first_number, second_number):
    """Calculates the first number minus two times the second number."""
    result = first_number - (second_number * 2)
    return result

# Example usage with meaningful variable names for the call
input_a = 10
input_b = 3
output = calculate_value(input_a, input_b)
print(output)

...
... 4
```

Task Description -4(Structural Refactoring)

Code used:

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

Prompt:

make the code modular using reusable functions to eliminate repetition.

Output:

```
◆ Gemini  
▶ def greet_person(name):
    print(f"Hello {name}")

greet_person("Ram")
greet_person("Sita")
greet_person("Ravi")  
... Hello Ram
Hello Sita
Hello Ravi
```

Task Description -5(Efficiency Enhancement)

Code used:

```
▶ numbers = []
for i in range(1, 500000):
    numbers.append(i * i)
print(len(numbers))  
... 499999
```

Prompt:

Optimize the code to achieve the same result with improved performance.

Output:

```
◆ Gemini  
▶ numbers = [i * i for i in range(1, 500000)]
print(len(numbers))
... 499999
```