

**School of Computer Science and Artificial  
Intelligence**

**LabAssignment-10.2**

<b>Program</b>	<b>:B.Tech(CSE)</b>
<b>Specialization</b>	<b>:AIML</b>
<b>Course Title</b>	<b>:AI Assisted Coding</b>
<b>Course Code</b>	<b>: 23CS002PC304</b>
<b>Semester</b>	<b>: VI</b>
<b>Academic Session</b>	<b>:2025-2026</b>
<b>Name of Student</b>	<b>: P.Sindhu</b>
<b>Enrollment No.</b>	<b>: 2303A52207</b>
<b>Batch No.</b>	<b>: 33</b>
<b>Date</b>	<b>:17-2-2026</b>

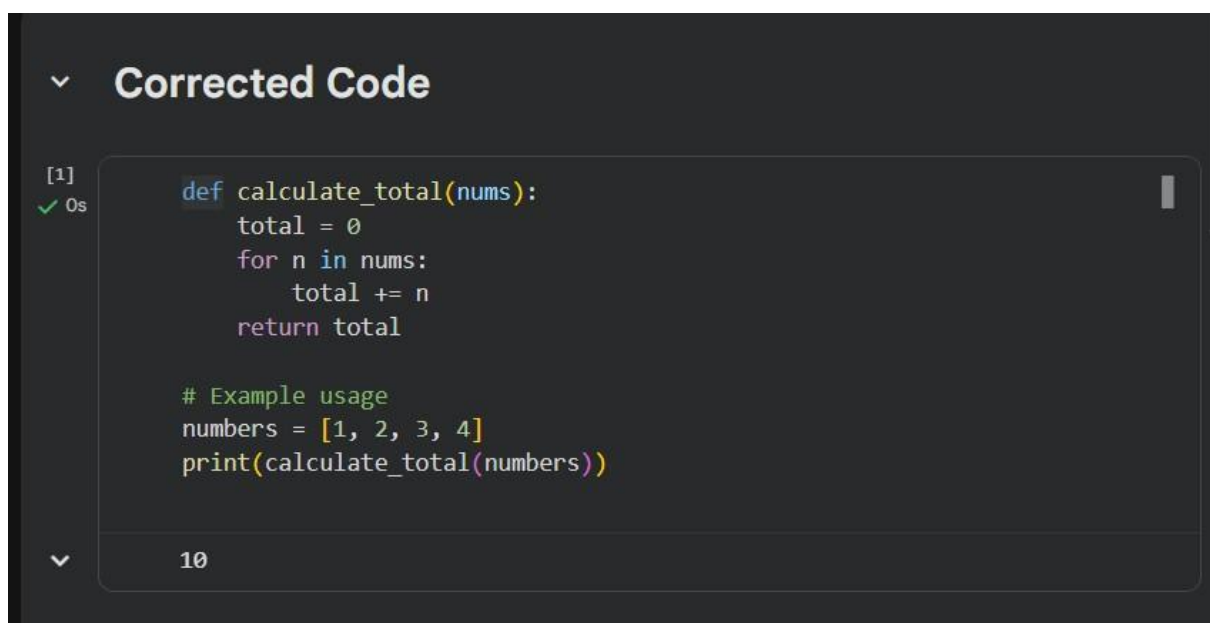
## Lab 10 – Code Review and Quality: Using AI to Improve Code Quality and Readability

### Lab Objectives

- Use AI for automated code review and quality enhancement.
- Identify and fix syntax, logical, performance, and security issues in Python code.
- Improve readability and maintainability through structured Week5 - Tuesday refactoring and comments.
- Apply prompt engineering for targeted improvements.
- Evaluate AI-generated suggestions against PEP 8 standards and software engineering best practices

### Task 1 – Error Detection and Correction Issues in Given Code

- Missing colon : after function definition.
- Wrong indentation.
- Missing colon in for loop.
- Variable total not defined (should return sum).
- Avoid using built-in name sum as a variable.



The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there is a section titled 'Corrected Code' with a dropdown arrow. Below this, a code cell is displayed with the following Python code:

```
[1] ✓ 0s
def calculate_total(nums):
    total = 0
    for n in nums:
        total += n
    return total

# Example usage
numbers = [1, 2, 3, 4]
print(calculate_total(numbers))
```

At the bottom of the code cell, the output '10' is shown, indicating the sum of the numbers in the list [1, 2, 3, 4].

### Fix Summary

- Added missing : symbols.
- Corrected indentation.
- Replaced undefined variable.

- Used total instead of built-in sum.

---

## Task 2 – Code Style Standardization (PEP 8)

```
▼ Refactored Code

[2]
✓ 0s ▶ def find_sum(a, b):
      """Return the sum of two numbers."""
      return a + b

      result = find_sum(5, 10)
      print(result)

▼ ... 15
```

### Improvements

- Function name changed to snake\_case.
- Added docstring.
- Proper spacing and formatting.

---

## Task 3 – Code Clarity Improvement

```
▼ Refactored Code

[3]
✓ 0s ▶ def calculate_adjusted_value(number, multiplier):
      """Subtract twice the multiplier from the number."""
      return number - (multiplier * 2)

      result = calculate_adjusted_value(10, 3)
      print(result)

▼ ... 4
```

### Improvements

- Meaningful function and variable names.
  - Added explanation through docstring.
  - Improved readability.
-

## Task 4 – Structural Refactoring (Reusable Functions)

```
✓ Refactored Code

[4] 0s ▶ def greet(name):
      """Print a greeting message for the given name."""
      print(f"Hello {name}")

      names = ["Ram", "Sita", "Ravi"]

      for name in names:
          greet(name)

... Hello Ram
     Hello Sita
     Hello Ravi
```

### Improvements

- Removed repetition.
- Created reusable function.
- Used loop for scalability.

---

## Task 5 – Efficiency Enhancement

```
Optimized Code

▶ # Using list comprehension (faster than loop + append)
  numbers = [i * i for i in range(1, 500000)]

  print(len(numbers))

... 499999

+ Code + Text
```

### Why Faster?

- List comprehensions are optimized in Python.
  - Avoids repeated method calls (append).
  - Cleaner and more memory-efficient.
-

## Conclusion (For Lab Submission)

Using AI-assisted review helped to:

- Fix syntax and logical errors.
- Improve code readability and structure.
- Ensure compliance with **PEP 8**.
- Reduce redundancy through modular design.
- Optimize performance using Python best practices.