

School of Computer Science and Artificial Intelligence

LabAssignment-10.2

Program	:B.Tech(CSE)
Specialization	:AIML
Course Title	:AI Assisted Coding
Course Code	: 23CS002PC304
Semester	: VI
Academic Session	:2025-2026
Batch No :	33

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 -

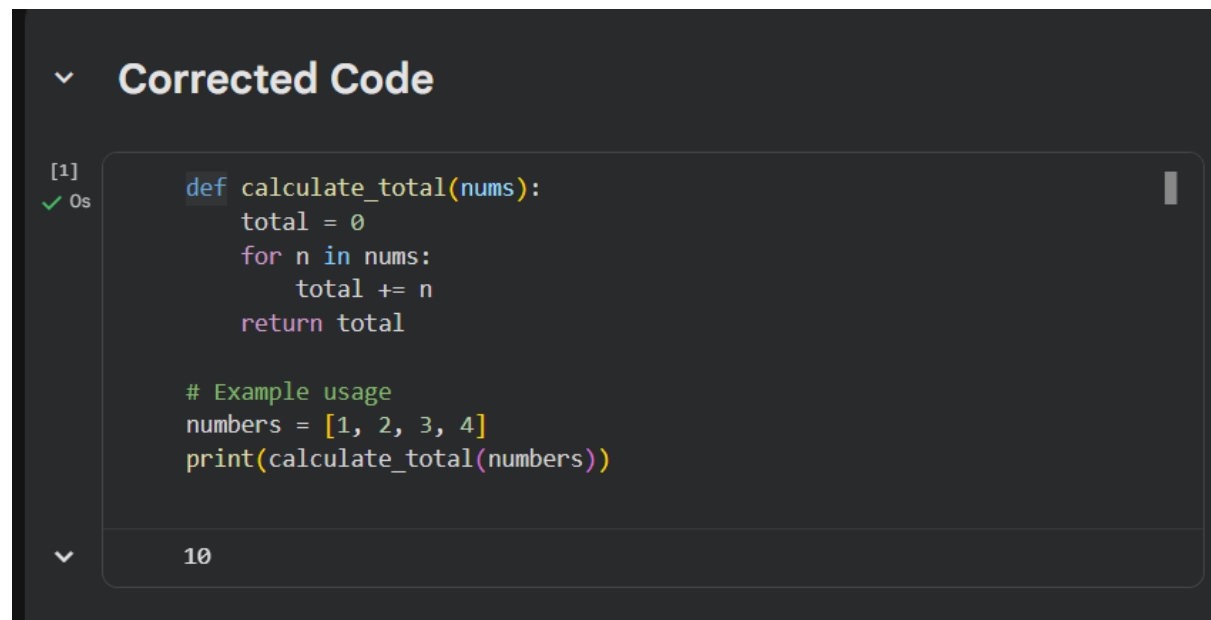
Tuesday

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 code editor with a dark background. At the top, there is a section header "Corrected Code" with a downward arrow. Below it, there is a code block with the following Python code:

```
[1]
✓ Os
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 block, there is a small output area showing the result "10".

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

```
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.