

## Lab Assignment 10.2

**Program Name:** B. Tech

**Assignment Type:** Lab

**Academic Year:** 2025-2026

**Course Code:** 23CS002PC304

**Course Title:** AI Assisted Coding

**Year/Sem:** III/II

**Regulation:** R23

**Student Name:** Ravula seetharam reddy

**Roll No:** 2303A52440

**Assignment Number:** 10.1 / 24

**Day:** Tuesday

---

## 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 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

#### Prompt Used

Analyze the Python code and fix all syntax and logical errors. Provide corrected executable code and explanation.

#### Given Code

```
def calculate_total(nums)
sum = 0
```

```
for n in nums
sum += n
return total
```

## AI Corrected Code

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

numbers = [1,2,3,4,5]
print(calculate_total(numbers))
```

## Explanation

- Missing colon after function and for loop fixed
- Indentation corrected
- Variable name changed from sum (reserved name) to total
- Returned variable corrected

## Output

15

### SCREENSHOTS:

The screenshot shows a code editor interface with two panes. The left pane is titled "main.py" and contains the following Python code:

```
1 def calculate_total(nums):
2     total = 0
3     for n in nums:
4         total += n
5     return total
6
7 numbers = [1, 2, 3, 4, 5]
8 print(calculate_total(numbers))
```

The right pane is titled "44duj6ty2" and shows the output of the code execution:

Output:  
15

---

## Task 2 – Code Style Standardization

### Prompt Used

Refactor the code to follow PEP 8 coding standards with proper formatting.

## Given Code

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

## Refactored Code (PEP 8)

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

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

## Explanation

- Function renamed using snake\_case
- Proper spacing added
- Stored output in variable for readability

## Output

15

## SCREENSHOTS:

The screenshot shows a code editor window titled "main.py". The code is as follows:

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

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

To the right of the code editor, there is a vertical bar labeled "Output:" followed by the number "15", indicating the result of running the code.

## Task 3 – Code Clarity Improvement

### Prompt Used

Improve readability using meaningful variable and function names without changing functionality.

## Given Code

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

## Improved Readable Code

```
def subtract_double(value, multiplier):  
    return value - multiplier * 2  
  
result = subtract_double(10, 3)  
print(result)
```

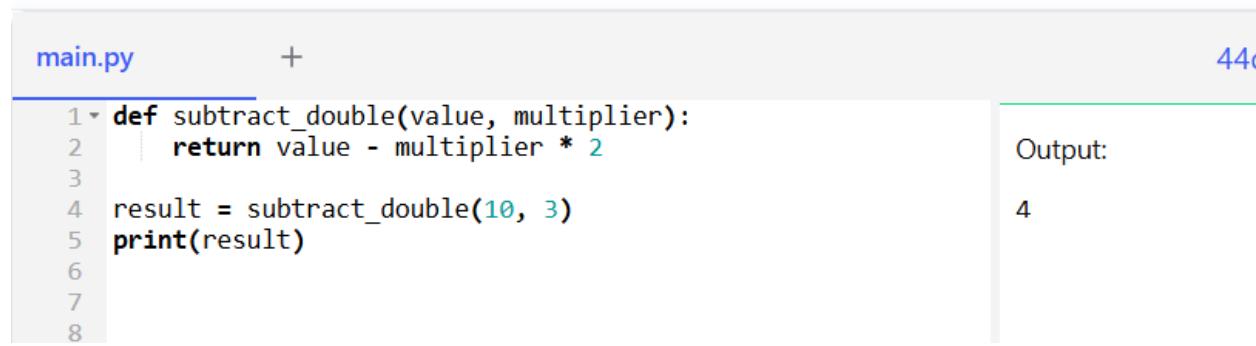
## Explanation

- Function name made descriptive
- Variable names improved for clarity
- Output stored in variable

## Output

4

### SCREENSHOTS:



The screenshot shows a code editor interface. On the left, there is a file named "main.py" with the following content:

```
1 def subtract_double(value, multiplier):  
2     return value - multiplier * 2  
3  
4 result = subtract_double(10, 3)  
5 print(result)
```

On the right, there is a "Output:" section showing the result of running the code, which is "4".

## Task 4 – Structural Refactoring

### Prompt Used

Refactor repetitive code using reusable functions.

## Given Code

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

## Modular Code

```
def greet(name):
    print(f"Hello {name}")

greet("Ram")
greet("Sita")
greet("Ravi")
```

## Explanation

- Created reusable function greet()
- Eliminated repeated print statements

## Output

Hello Ram Hello Sita Hello Ravi

The screenshot shows a code editor interface. On the left, there is a file named "main.py" containing the following Python code:

```
1 def greet(name):
2     print(f"Hello {name}")
3
4 greet("Ram")
5 greet("Sita")
6 greet("Ravi")
```

On the right, the output window displays the text "44duj6ty2 ⚡" at the top, followed by "Output:" and the three lines of greeting text: "Hello Ram", "Hello Sita", and "Hello Ravi".

## Task 5 – Efficiency Enhancement

### Prompt Used

Optimize the Python code for better performance.

### Given Code

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

### Optimized Code

```
numbers = [i * i for i in range(1, 500000)]
print(len(numbers))
```

## Explanation

- Replaced loop with list comprehension
- Faster execution and cleaner code

## Output

499999

### SCREENSHOTS:

The screenshot shows a code editor interface. On the left, there is a file named "main.py" containing the following Python code:

```
1 numbers = [i * i for i in range(1, 500000)]
2 print(len(numbers))
3
4
5
6
7
8
9
```

In the center, there is a small plus sign icon. On the right, the text "44duj6ty2 ⚡" is displayed above the word "Output". Below "Output" is the number "499999".

## Conclusion

AI tools helped identify syntax errors, improve readability, enforce coding standards and optimize performance. The experiment demonstrated how prompt engineering can enhance code quality and maintainability effectively.