

School of Computer Science and Artificial Intelligence

Lab Assignment # 2.2

Program : B. Tech (CSE)
Specialization :
Course Title : AI Assisted coding
Course Code :
Semester : II
Academic Session : 2025-2026
Name of Student : M.Srinath
Enrollment No. : 2403A51L29
Batch No. : 51
Date : 013-01-2026

Task -1: Cleaning Sensor Data

Prompt : Generate a Python function that filters out all negative numbers from a list.

```
[1] 
✓ 0s
def remove_negative_values(sensor_data):
    cleaned_data = []
    for value in sensor_data:
        if value >= 0:
            cleaned_data.append(value)
    return cleaned_data
```

Output

```
[2]
✓ 0s
sensor_data = [12, -5, 7, -3, 0, 15, -8]
print("Before Cleaning:", sensor_data)
print("After Cleaning:", remove_negative_values(sensor_data))

▼
Before Cleaning: [12, -5, 7, -3, 0, 15, -8]
After Cleaning: [12, 7, 0, 15]
```

Code Explanation

The function checks each value in the list.

Only valid non-negative values are stored and returned.

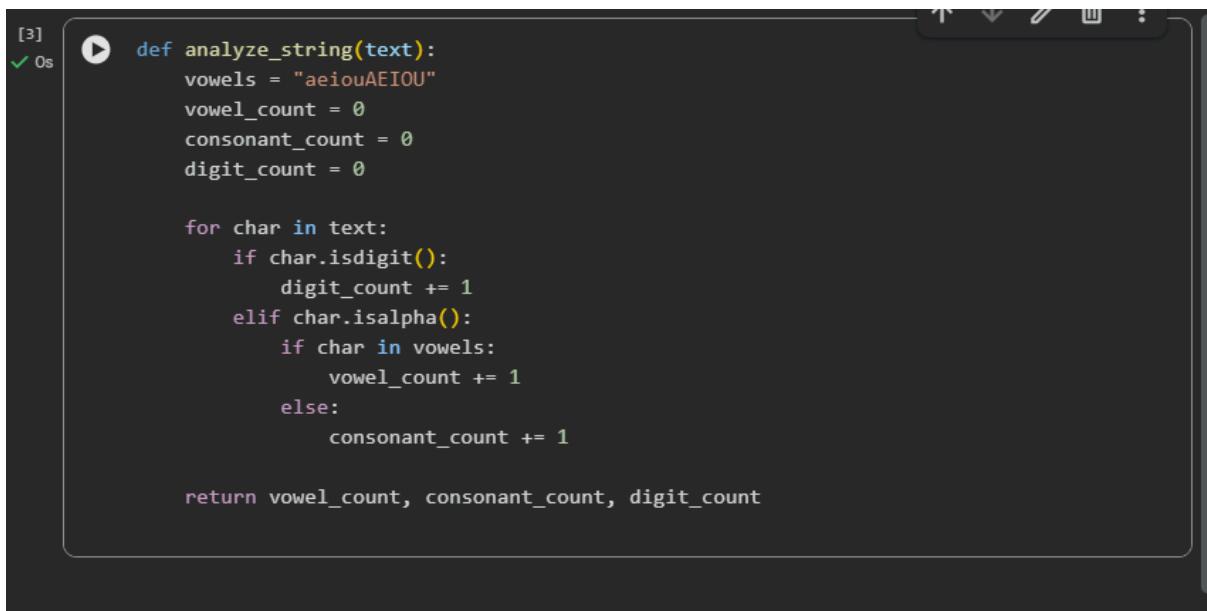
Comments

Gemini generated correct and simple logic.

The code is easy to understand and reusable.

Task 2: String Character Analysis

Prompt: Generate a Python function that counts vowels, consonants, and digits in a string.

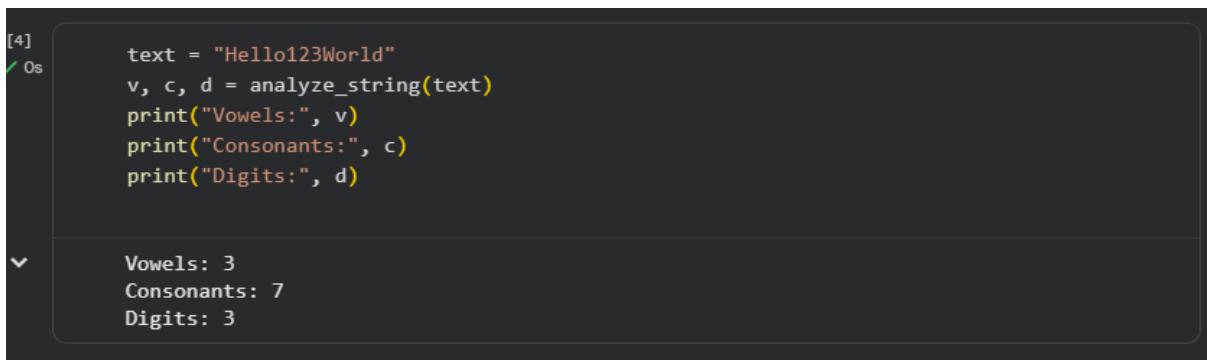


```
[3]  ✓ Os
    def analyze_string(text):
        vowels = "aeiouAEIOU"
        vowel_count = 0
        consonant_count = 0
        digit_count = 0

        for char in text:
            if char.isdigit():
                digit_count += 1
            elif char.isalpha():
                if char in vowels:
                    vowel_count += 1
                else:
                    consonant_count += 1

        return vowel_count, consonant_count, digit_count
```

Output:



```
[4]  ✓ Os
    text = "Hello123World"
    v, c, d = analyze_string(text)
    print("Vowels:", v)
    print("Consonants:", c)
    print("Digits:", d)

    Vowels: 3
    Consonants: 7
    Digits: 3
```

Code Explanation:

Digits, vowels, and consonants are counted separately.

Uses built-in string methods like `isdigit()` and `isalpha()`.

Comments:

Gemini helped understand string handling.

Code is well structured and efficient.

Task 3: Palindrome Check – Tool Comparison**Gemini Generated code**

```
[5] ✓ 0s
def is_palindrome_gemini(s):
    s = s.lower()
    return s == s[::-1]
```

Copilot Generated Code:

```
[6] ✓ 0s
def is_palindrome_copilot(text):
    cleaned = text.lower()
    reversed_text = cleaned[::-1]
    if cleaned == reversed_text:
        return True
    return False
```

Code Explanation:

Both codes check whether a string reads the same forward and backward.

Gemini uses a shorter, optimized approach.

Copilot uses step-by-step logic for better readability.

Comments:

Gemini is concise and fast.

Copilot is easier to understand for beginners.

Task 4: Code Explanation Using AI

```
[7] 
✓ 0s
def is_palindrome(s):
    s = s.lower()
    return s == s[::-1]
```

AI Explanation

- The function defines a palindrome check.
- Converts input to lowercase.
- Compares the string with its reversed version.
- Returns True or False.

Student Understood:

Gemini clearly explained slicing (`[::-1]`).

Helpful for quick code review.

Cursor AI – Explanation

What is Cursor AI?

- Cursor AI is an AI-powered code editor.
- It helps generate, refactor, and explain code.
- Useful for debugging and improving code quality.

Student Comments on Cursor AI

Good for editing large code files.

Helpful in understanding existing code.

Improves productivity.

Final Conclusion:

This lab helped me understand how AI tools like **Gemini, Copilot, and Cursor AI** assist in coding, explanation, and optimization. Each tool has its own strengths and improves learning efficiency.