

# AI ASSISTANT CODING ASSIGNMENT - 2

**NAME: K.LAXMAN**

**HT.NO: 2303A51428**

**BATCH: 21**

---

**LAB 2:**

**Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and  
Cursor AI**

**Task 1: Cleaning Sensor Data**

❖ **Scenario:**

❖ **You are cleaning IoT sensor data where negative values are invalid.**

❖ **Task:**

**Use Gemini in Colab to generate a function that filters out all negative  
numbers from a list.**

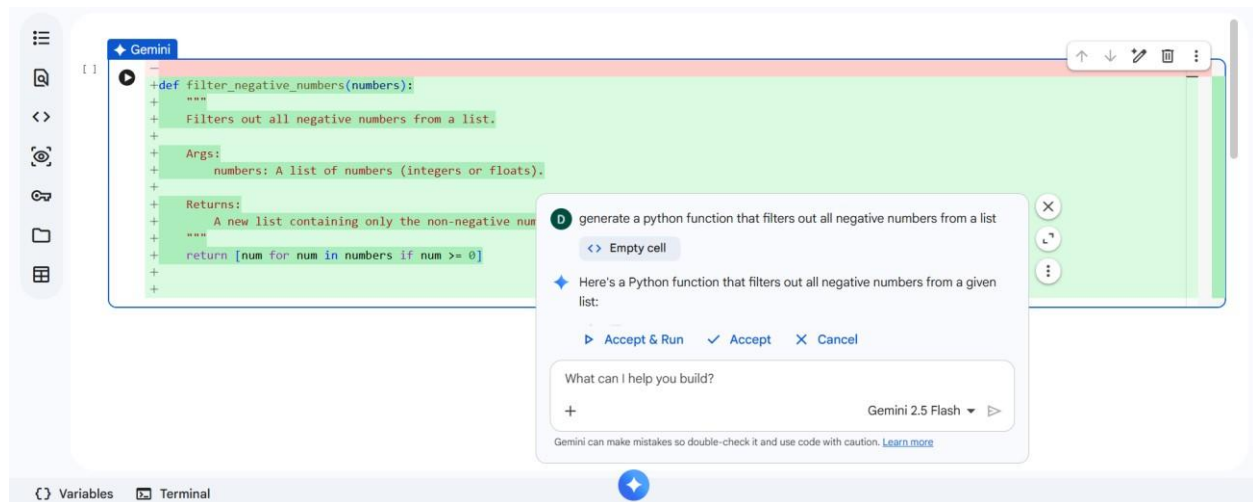
❖ **Expected Output:**

➤ **Before/after list**

➤ **Screenshot of Colab execution**

---

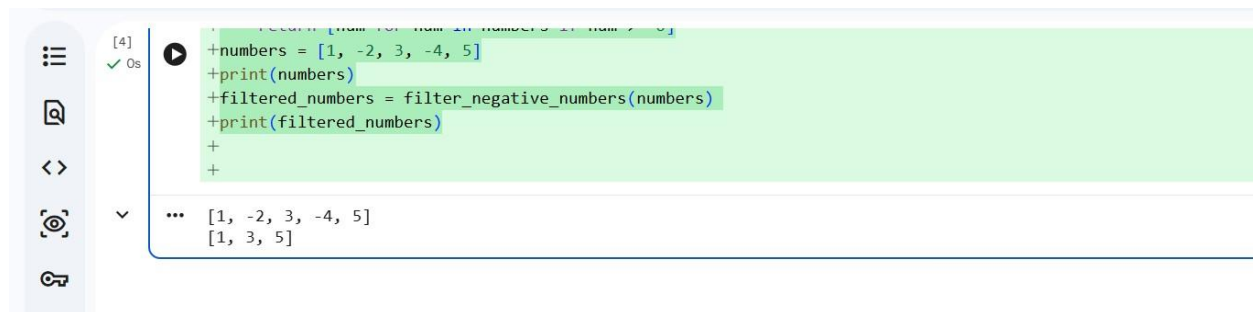
## CODE :



The screenshot shows a code editor with a Python function definition for `filter_negative_numbers`. The function takes a list of numbers and returns a new list containing only non-negative numbers. A Gemini chat window is open, displaying the prompt "generate a python function that filters out all negative numbers from a list" and the response "Here's a Python function that filters out all negative numbers from a given list:". The chat window also includes buttons for "Accept & Run", "Accept", and "Cancel", and a text input field for further prompts.

```
def filter_negative_numbers(numbers):  
    """  
    Filters out all negative numbers from a list.  
    Args:  
        numbers: A list of numbers (integers or floats).  
    Returns:  
        A new list containing only the non-negative numbers.  
    """  
    return [num for num in numbers if num >= 0]
```

## OUTPUT:



The screenshot shows the code editor with the function definition and a test case. The test case defines a list `numbers = [1, -2, 3, -4, 5]` and calls the `filter_negative_numbers` function. The output shows the original list and the filtered list, which contains only the non-negative numbers.

```
numbers = [1, -2, 3, -4, 5]  
print(numbers)  
filtered_numbers = filter_negative_numbers(numbers)  
print(filtered_numbers)
```

Output:

```
[1, -2, 3, -4, 5]  
[1, 3, 5]
```

## Task 2: String Character Analysis

### ❖ Scenario:

You are building a text-analysis feature.

### ❖ Task:

Use Gemini to generate a Python function that counts vowels, consonants, and digits in a string.

➤ **Sample inputs and outputs**

The screenshot shows the Google Colab environment. At the top, the file name is 'Untitled6.ipynb'. The menu bar includes File, Edit, View, Insert, Runtime, Tools, and Help. The toolbar shows 'Commands', 'Code', 'Text', and 'Run all'. On the right, there are icons for chat, settings, and a 'Share' button. Below the toolbar, the RAM and Disk usage is shown as '0 / 0 MB'.

The main area contains a Python script with the following code:

```
[7] ✓ Os
def count_char_types(text):
    vowels = 0
    consonants = 0
    digits = 0
    all_vowels = "aeiou"

    for char in text:
        char_lower = char.lower()
        if char_lower.isalpha():
            if char_lower in all_vowels:
                vowels += 1
            else:
                consonants += 1
        elif char_lower.isdigit():
            digits += 1

    return {
        "vowels": vowels,
        "consonants": consonants,
        "digits": digits
    }

# Example usages
my_string = "Hello World 123!"
counts = count_char_types(my_string)
print(counts)
```

At the bottom of the script, the output is shown: `{'vowels': 3, 'consonants': 7, 'digits': 3}`.

A Gemini chat window is open on the right side of the screen. It contains the following text:

**D** generate a python function that filters out all negative numbers from a list

[Empty cell](#)

Here's a Python function that filters out all negative numbers from a given list:

[Accept & Run](#) [Accept](#) [Cancel](#)

What can I help you build?

[+](#) Gemini 2.5 Flash [▶](#)

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

At the bottom of the screen, the status bar shows 'Variables', 'Terminal', '12:17 PM', and 'Python 3'.

```

+ # Example usage:
+ my_string = "Hello World 123!"
+ counts = count_char_types(my_string)
+ print(counts)

```

```

... {'vowels': 3, 'consonants': 7, 'digits': 3}

```

❖ **Task:**

Generate a palindrome-checking function using Gemini and Copilot, then compare the results.

❖ Expected Output:

- Side-by-side code comparison
- Observations on clarity and structure

CODE:

The screenshot shows a Jupyter Notebook titled 'Untitled6.ipynb'. The code cell contains a Python function `is_palindrome` that checks if a string is a palindrome by converting it to lowercase, removing non-alphanumeric characters, and comparing it to its reverse. The function is tested with 'madam', 'A man, a plan, a canal: Panama', 'race a car', and 'hello'. A Gemini AI chat window is open, showing a prompt to 'generate a python function that count vowels, consonants, and digits in a string' and a response providing a similar function. The chat window also includes a 'What can I help you build?' prompt and a 'Gemini 2.5 Flash' model selector.

OUTPUT:

The output of the Jupyter Notebook shows the results of the `is_palindrome` function calls. The first call returns `True` for 'A man, a plan, a canal: Panama'. The second call returns `False` for 'race a car'. The third call returns `False` for 'hello'.

Task 4: Code Explanation Using AI

❖ Scenario:

You are reviewing unfamiliar code written by another developer.

❖ Task:

Ask Gemini to explain a Python function (prime check OR palindrome check)

line by line.

❖ Expected Output:

➤ Code snippet

➤ AI explanation

➤ Student comments on understanding

---

OUTPUT:

generate a python function that filters out all negative numbers from a list

<> Empty cell

◆ Here's a Python function that filters out all negative numbers from a given list:

▶ Accept & Run

✓ Accept

✕ Cancel

What can I help you build?

+

Gemini 2.5 Flash ▾ ▶

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)



1. `def is_palindrome(text):`

◦ This line defines a Python function named `is_palindrome` that accepts one argument: `text` (which is expected to be a string).

2. `"""`

▶ Accept & Run

✓ Accept

✕ Cancel

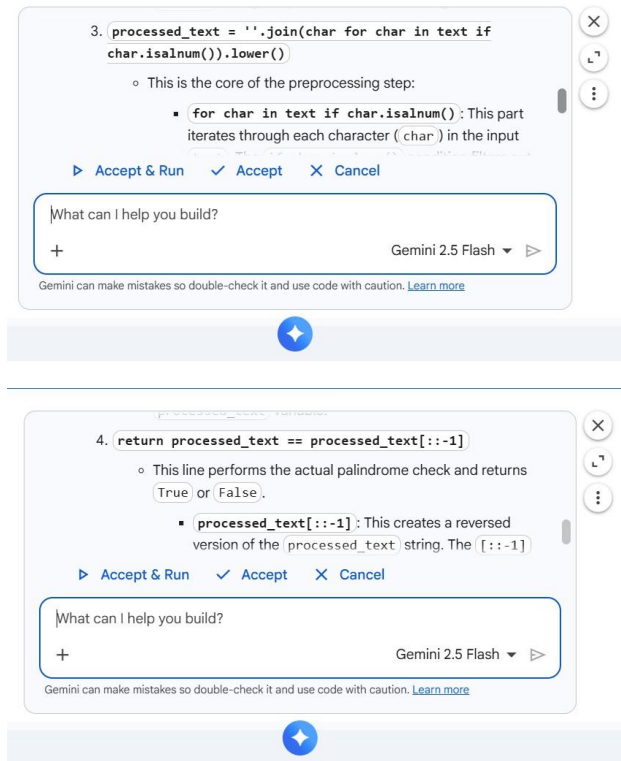
What can I help you build?

+

Gemini 2.5 Flash ▾ ▶

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)





## My own experience using both Gemini and GitHub Copilot:

While using Gemini in Google Colab, I found the explanations to be very clear and helpful in understanding the logic behind the code. Gemini was especially useful for learning and analyzing Python programs step by step. GitHub Copilot, was faster in generating code directly inside the editor and helped me complete tasks quickly. Copilot felt more suitable for continuous coding, while Gemini was better for conceptual clarity. Overall, using both tools together improved my coding efficiency and understanding.