

AI ASSISTANT CODING ASSIGNMENT - 2

NAME: JILLELA AKSHAYA

HT.NO: 2303A51629

BATCH: 22

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 Jupyter Notebook titled 'Untitled3.ipynb' in a web browser. The notebook contains a Python function definition for `filter_negative_numbers` and an example usage. A Gemini AI chat window is open, displaying a prompt to generate a Python function and the AI's response, which includes the function definition and an example usage. The chat window also shows a text input field and a 'What can I help you build?' prompt.

```
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]
```

```
# Example usage with your existing 'my_list'  
filtered_list = filter_negative_numbers(my_list)  
  
print(f"Original list: {my_list}")  
print(f"Filtered list (non-negative numbers): {filtered_list}")
```

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

Certainly! Here's a Python function to filter out negative numbers from a list, along with an example of how to use it with your existing `my_list` variable.

Accept and run Accept Cancel

What can I help you build?

Gemini 2.5 Flash

OUTPUT:

The screenshot shows the output of the Jupyter Notebook. It displays the function definition and the execution results of the example usage. The output shows the original list and the filtered list (non-negative numbers).

```
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]
```

```
# Example usage with your existing 'my_list'  
filtered_list = filter_negative_numbers(my_list)  
  
print(f"Original list: {my_list}")  
print(f"Filtered list (non-negative numbers): {filtered_list}")
```

```
... Original list: [-5, 10, -2, 8, 0, -1]  
Filtered list (non-negative numbers): [10, 8, 0]
```

Task 2: String Character Analysis

❖ Scenario:

You are building a text-analysis feature.

❖ Task:

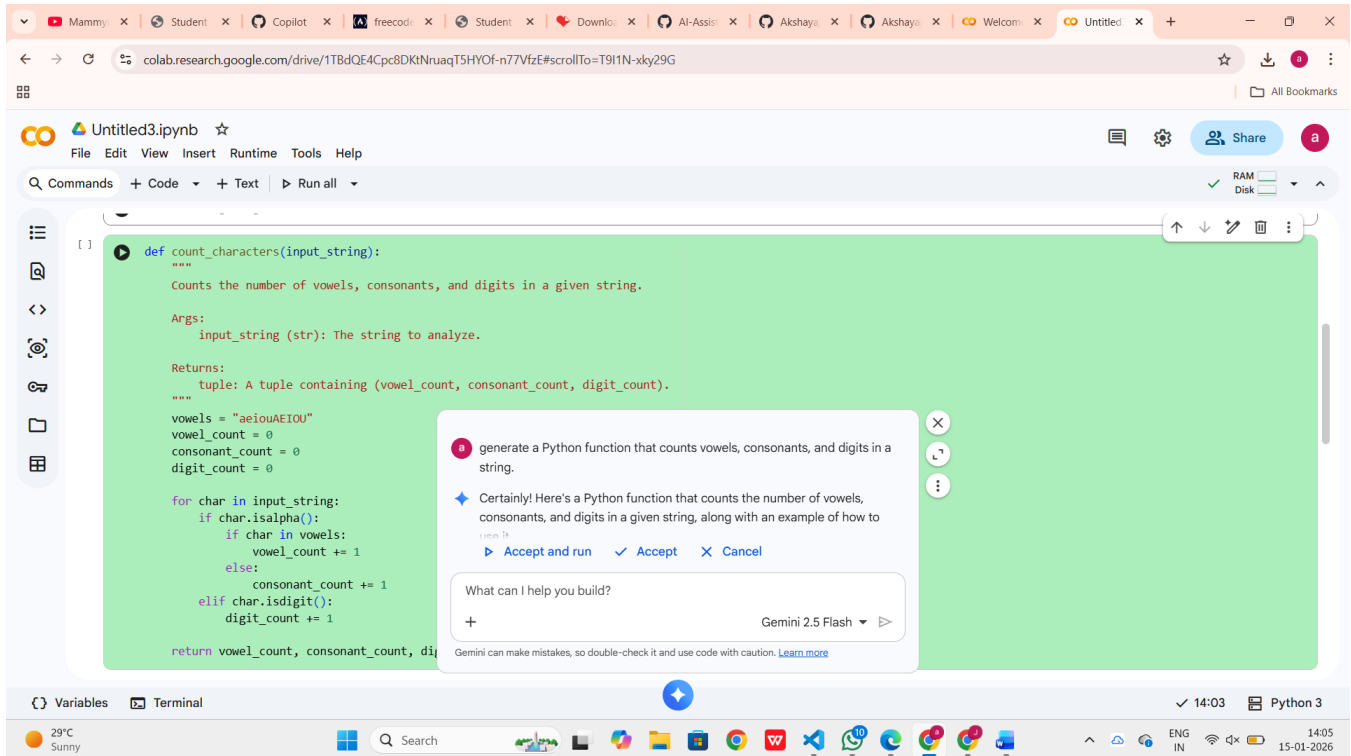
Use Gemini to generate a Python function that counts vowels, consonants, and digits in a string.

❖ Expected Output:

➤ Working function

➤ Sample inputs and outputs

CODE :



The screenshot shows a Google Colab notebook titled "Untitled3.ipynb". The code in the notebook is as follows:

```
[ ] def count_characters(input_string):  
    """  
    Counts the number of vowels, consonants, and digits in a given string.  
  
    Args:  
        input_string (str): The string to analyze.  
  
    Returns:  
        tuple: A tuple containing (vowel_count, consonant_count, digit_count).  
    """  
    vowels = "aeiouAEIOU"  
    vowel_count = 0  
    consonant_count = 0  
    digit_count = 0  
  
    for char in input_string:  
        if char.isalpha():  
            if char in vowels:  
                vowel_count += 1  
            else:  
                consonant_count += 1  
        elif char.isdigit():  
            digit_count += 1  
  
    return vowel_count, consonant_count, digit_count
```

A Gemini AI chat window is open, showing a prompt and a response:

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

Certainly! Here's a Python function that counts the number of vowels, consonants, and digits in a given string, along with an example of how to use it

Accept and 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](#)

OUTPUT:

```
return vowel_count, consonant_count, digit_count
```

```
[ ] # Example usage:  
sample_string = "Hello World 123!"  
v, c, d = count_characters(sample_string)  
  
print(f"Original string: '{sample_string}'")  
print(f"Number of vowels: {v}")  
print(f"Number of consonants: {c}")  
print(f"Number of digits: {d}")
```

Task 3: Palindrome Check – Tool Comparison

❖ Scenario:

You must decide which AI tool is clearer for string logic.

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 Google Colab notebook titled 'Untitled3.ipynb'. The code cell contains a Python function `is_palindrome(s)` generated by Gemini. The function checks if a string is a palindrome, ignoring case and non-alphanumeric characters. It includes docstrings for the function, arguments, and returns, as well as example usage.

```
def is_palindrome(s):  
    """  
    Checks if a string is a palindrome, ignoring case and non-alphanumeric characters.  
    Args:  
        s (str): The input string.  
    Returns:  
        bool: True if the string is a palindrome, False otherwise.  
    """  
    # Convert to lowercase and filter out non-alphanumeric characters  
    cleaned_string = ''.join(char for char in s if char.isalnum()).lower()  
    # Compare the cleaned string with its reverse  
    return cleaned_string == cleaned_string[::-1]  
  
# Example usage:  
print("--- Palindrome Function (Core Subage...)  
  
# a. A string that is a palindrome  
string1 = "Racecar"  
print(f'{string1} is a palindrome: {is_palindrome(string1)}')  
  
# b. A string that is a palindrome with spaces  
string2 = "A man, a plan, a canal: Panama"  
print(f'{string2} is a palindrome: {is_palindrome(string2)}')
```

A chat window for Gemini 2.5 Flash is overlaid on the code. It shows the prompt 'Generate a palindrome-checking function' and the response: 'Certainly! I'll generate a Python function to check for palindromes. Here's the plan: Generate Palindrome Function: Generate a Python function that takes a string as input and returns True if it's a palindrome (ignoring case and non-alphanumeric characters), and False otherwise. The plan: What can I help you build?'

OUTPUT:

The output of the code is shown in a green box. It displays the results of the `is_palindrome` function for various inputs:

```
+print(is_palindrome("A man, a plan, a canal: Panama")) # Output: True  
+print(is_palindrome("race a car")) # Output: False  
+print(is_palindrome("hello")) # Output: False  
+
```

Below the code, the output is displayed as a list of boolean values:

```
... True  
True  
False  
False
```

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.

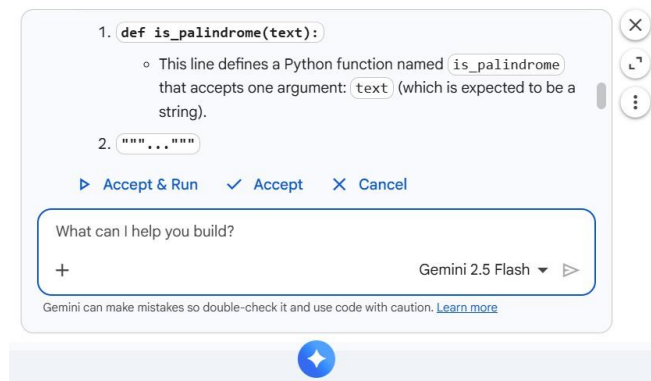
❖ Expected Output:

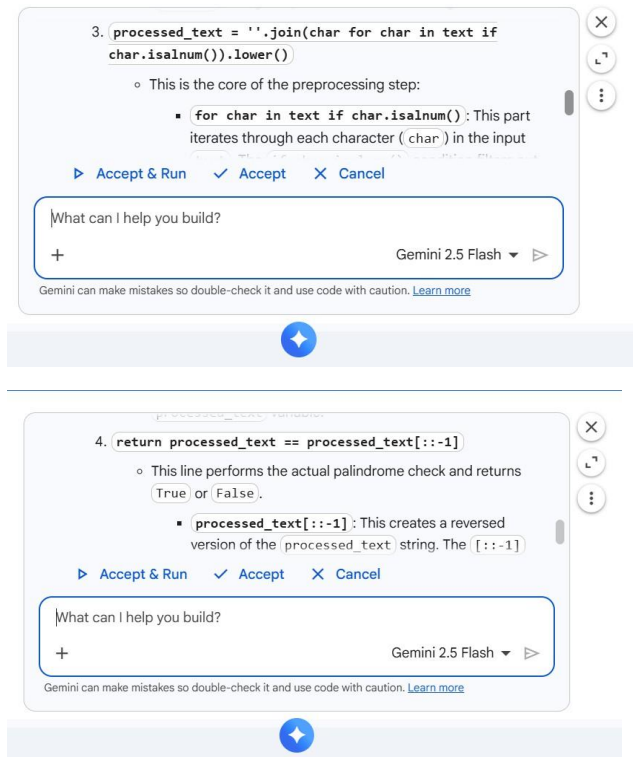
➤ Code snippet

➤ AI explanation

➤ Student comments on understanding

➤ OUPUT





My own experience using both Gemini and GitHub Copilot:

While working with **Gemini** in **Google Colab**, I noticed that the explanations were well-structured and easy to follow. It helped me understand the reasoning and flow behind Python programs by breaking them down step by step. This made Gemini particularly helpful for learning, debugging, and analyzing code logic.

On the other hand, **GitHub Copilot** focused more on speed and productivity. By generating code suggestions directly inside the editor, it allowed me to implement solutions quickly and continue coding without interruptions. Copilot felt more effective during active development, especially when working on longer programs.

In summary, Gemini supported deeper conceptual understanding, while GitHub Copilot enhanced coding speed. Using both tools together significantly improved my learning experience as well as overall coding efficiency.