

# AI ASSISTANT CODING ASSIGNMENT -

## 2

**NAME:** Uddemari sharwan

**HT.NO:** 2303A51957

**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 the Gemini AI interface. A code completion dialog is open, asking to generate a Python function that filters out all negative numbers from a list. The generated code is:

```
+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]
```

Below the code, there are buttons for "Accept & Run", "Accept", and "Cancel". A message at the bottom says "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

## OUTPUT:

The terminal window shows the execution of the generated Python code. The code defines a function `filter\_negative\_numbers` and uses it to filter a list of numbers.

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

❖ Expected Output:

➤ Working function

➤ Sample inputs and outputs

---

**CODE :**

The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code:

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

The code defines a function `count_char_types` that takes a string `text` and returns a dictionary with counts of vowels, consonants, and digits. It uses `lower()` to handle both uppercase and lowercase letters. The function iterates over each character in the string, checks if it's a vowel or consonant, and increments the respective counter. Finally, it returns a dictionary with the counts.

A tooltip from Gemini 2.5 Flash provides instructions for generating a function that filters out negative numbers from a list. The tooltip includes options to "Accept & Run", "Accept", or "Cancel".

**OUTPUT:**

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

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

---

**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:

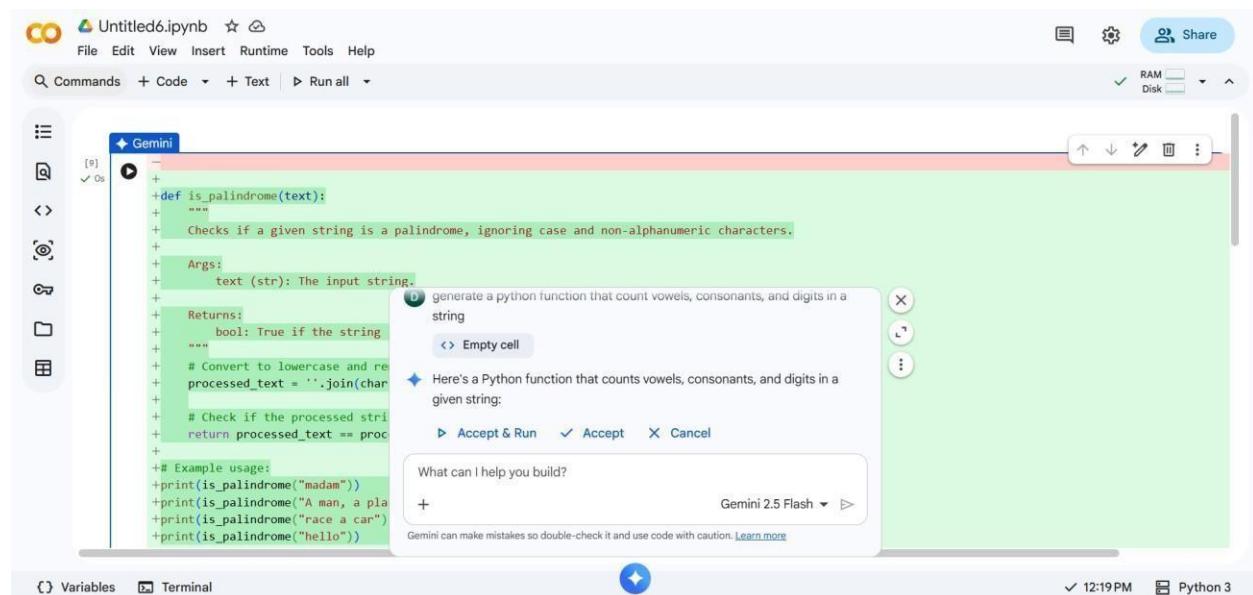
➤ Side-by-side code comparison

➤ Observations on clarity and structure

---

---

**CODE:**



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains a Python function for checking if a string is a palindrome, ignoring case and non-alphanumeric characters. The second cell contains a Python function for generating a function that counts vowels, consonants, and digits in a string. A tooltip from Gemini provides the code for the second function.

```
+def is_palindrome(text):
+    """
+        Checks if a given string is a palindrome, ignoring case and non-alphanumeric characters.
+
+        Args:
+            text (str): The input string.
+
+        Returns:
+            bool: True if the string
+            ===
+            # Convert to lowercase and re
+            processed_text = ''.join(char
+            # Check if the processed stri
+            return processed_text == proc
+
+    # Example usage:
+    +print(is_palindrome("madam"))
+    +print(is_palindrome("A man, a pla
+    +print(is_palindrome("race a car"))
+    +print(is_palindrome("hello"))
```

generate a python function that count vowels, consonants, and digits in a string

Here's a Python function that counts vowels, consonants, and digits in a given string:

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](#)

## OUTPUT:

---

```
+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
+
...
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

---

## OUTPUT:

The screenshot shows a Gemini AI interface. A code completion dialog is open, containing 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](#)



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

2. `"""\n """`

▶ 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](#)

3. `processed_text = ''.join(char for char in text if char.isalnum()).lower()`  
o This is the core of the preprocessing step:  
▪ `for char in text if char.isalnum()`: This part iterates through each character (`char`) in the input

▶ 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](#)

4. `return processed_text == processed_text[::-1]`  
o This line performs the actual palindrome check and returns `True` or `False`.  
▪ `processed_text[::-1]`: This creates a reversed version of the `processed_text` string. The `[::-1]`

▶ 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.