

AI ASSISTANT CODING ASSIGNMENT - 2

NAME: Deekshith Kowdagani

HT.NO: 2303A51414

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 the Gemini AI interface. A code completion window is open, suggesting the following Python function:

```
+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]
```

The interface includes standard code editor tools like copy, paste, and search, along with a sidebar for variables and terminal access.

OUTPUT:

The screenshot shows the execution results of the provided Python code. The output is:

```
[1] 0s
+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 :

```
+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)
```

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](#)

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:

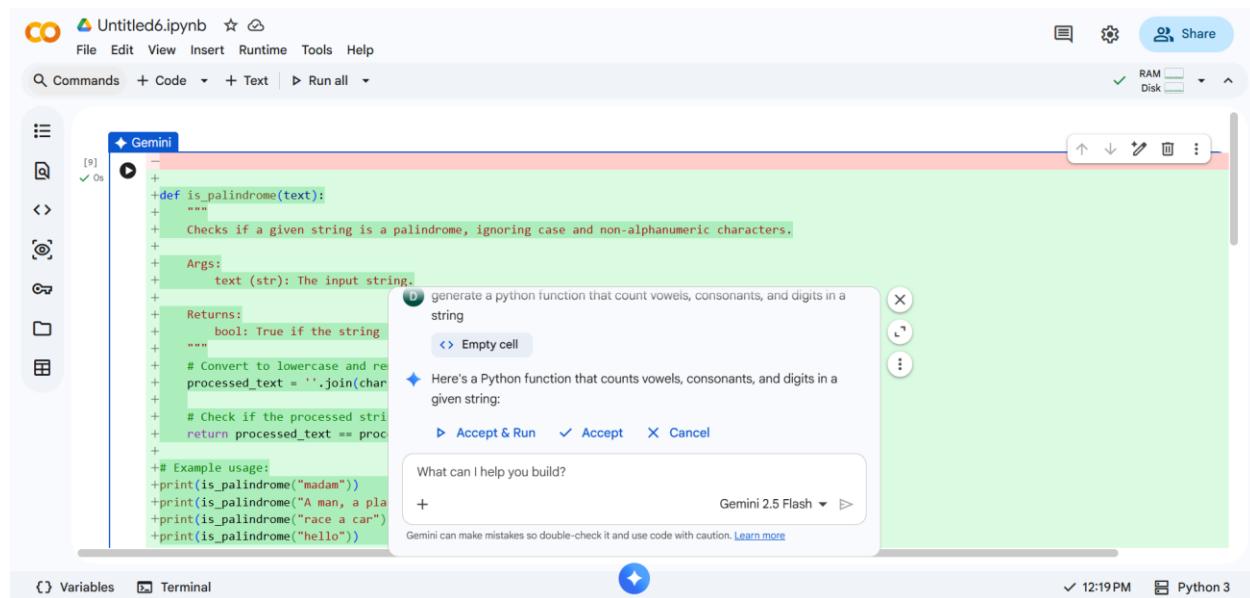
Generate a palindrome-checking function using Gemini and Copilot, then compare the results.

❖ Expected Output:

➤ Side-by-side code comparison

➤ Observations on clarity and structure

CODE:



The screenshot shows a Jupyter Notebook interface with a single code cell containing Python code for a palindrome checker. The code is annotated with docstrings and type hints. A tooltip from Gemini provides a related task about generating a vowel/consonant/digit counter. The notebook also shows a history of print statements at the bottom.

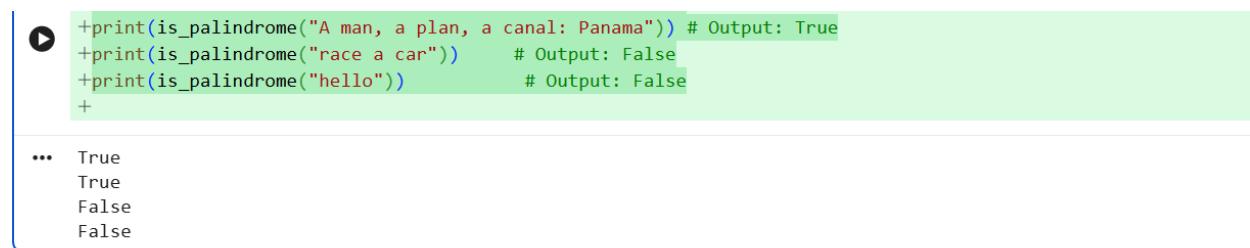
```
+def is_palindrome(text):
    """
    Checks if a given string is a palindrome, ignoring case and non-alphanumeric characters.

    Args:
        text (str): The input string.

    Returns:
        bool: True if the string
    """
    # Convert to lowercase and remove punctuation
    processed_text = ''.join(char for char in text if char.isalnum())
    # Check if the processed string is equal to its reverse
    return processed_text == processed_text[::-1]

# Example usage:
+print(is_palindrome("madam"))
+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
```

OUTPUT:



The screenshot shows the AI-generated output of the code execution. It includes the original code cell and the resulting terminal output showing the execution of the function with various inputs and their outputs.

```
+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:

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):`
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](#)

3. `processed_text = ''.join(char for char in text if char.isalnum()).lower()`

- 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]`

- 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.