

# AI ASSISTANT CODING

## ASSIGNMENT-2

---

NAME : MOGILI KEERTHI

HT.NO : 2303A51427

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

---

**Prompt:**

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

The screenshot shows a code editor interface with a Python script. The script defines a function `filter\_negative\_numbers` that takes a list of numbers and returns a new list containing only non-negative numbers. It includes example usage with a list of integers.

```
[2] 0s
def filter_negative_numbers(numbers):
    return [num for num in numbers if num >= 0]

# Example usage:
my_list = [-1, 2, -3, 4, -5, 6, 0]
filtered_list = filter_negative_numbers(my_list)
print(f"Original list: {my_list}")
print(f"Filtered list (non-negative): {filtered_list}")


... Original list: [-1, 2, -3, 4, -5, 6, 0]
Filtered list (non-negative): [2, 4, 6, 0]
```

A modal window from Gemini 2.5 Flash displays the generated code and provides options to accept or cancel the response.

K 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 list.

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](#)

**BEFORE LIST & AFTER LIST :**

.. Original list: [-1, 2, -3, 4, -5, 6, 0]  
Filtered list (non-negative): [2, 4, 6, 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

---

Prompt:

Generate a python function that counts vowels, consonants, and digits in a string.

The screenshot shows the Gemini AI interface with a code completion session. On the left, there is a code editor window containing a Python script. The script defines a function `count_chars` that takes an input string and counts vowels, consonants, and digits. It includes a usage example where it prints the counts for two strings: "Hello World 123!" and "Python is fun! 4ever". The code editor has a toolbar at the top with icons for up, down, copy, paste, and more. To the right of the code editor is a large green workspace area. In the bottom right corner of the workspace, there is a small interface for interacting with Gemini, which includes a text input field, a "Gemini 2.5 Flash" button, and a "Learn more" link.

```
+def count_chars(input_string):
+    vowels = "aeiou"
+    vowel_count = 0
+    consonant_count = 0
+    digit_count = 0
+
+    for char in input_string:
+        char_lower = char.lower()
+        if char_lower.isalpha():
+            if char_lower in vowels:
+                vowel_count += 1
+            else:
+                consonant_count += 1
+        elif char_lower.isdigit():
+            digit_count += 1
+
+    return {"vowels": vowel_count, "consonants": consonant_count, "digits": digit_count}
+
+# Example usage:
+text = "Hello World 123!"
+counts = count_chars(text)
+print(f"Original string: '{text}'")
+print(f"Vowels: {counts['vowels']}")
+print(f"Consonants: {counts['consonants']}")
+print(f"Digits: {counts['digits']}")
+
+text2 = "Python is fun! 4ever"
+counts2 = count_chars(text2)
+print(f"\nOriginal string: '{text2}'")
+print(f"Vowels: {counts2['vowels']}")
+print(f"Consonants: {counts2['consonants']}")
+print(f"Digits: {counts2['digits']}
```

SAMPLE INPUT & OUTPUT :



```
+print(f"Vowels: {counts2['vowels']}")
+print(f"Consonants: {counts2['consonants']}")  
+print(f"Digits: {counts2['digits']}")  
+
```

---

... Original string: 'Hello World 123!'  
Vowels: 3  
Consonants: 7  
Digits: 3

Original string: 'Python is fun! 4ever'  
Vowels: 5  
Consonants: 10  
Digits: 1

---

### 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

---

---

Prompt:

---

## Generate a palindrome-checking function

```
s def is_palindrome(text):
    # Convert to lowercase and remove non-alphanumeric characters
    processed_text = re.sub(r'[^a-zA-Z0-9]', '', text).lower()
    # Compare the processed string with its reverse
    return processed_text == processed_text[::-1]

# Example usage:
print(f"'madam' is a palindrome: {is_palindrome('madam')}")
print(f"'A man, a plan, a canal: Panama' is a palindrome: {is_palindrome('A man, a plan, a canal: Panama')}")
print(f"'hello' is a palindrome: {is_palindrome('hello')}")
print(f"'racecar' is a palindrome: {is_palindrome('racecar')}")
print(f"'No lemon, no melon' is a palindrome: {is_palindrome('No lemon, no melon')}")
```

... 'madam' is a palindrome: True  
'A man, a plan, a canal: Panama' is a palindrome: True  
'hello' is a palindrome: False  
'racecar' is a palindrome: True  
'No lemon, no melon' is a palindrome: True

K Generate a palindrome-checking function

◆ Certainly! Here's a Python function that checks if a string is a palindrome, ignoring case and non-alphanumeric characters.

↶ ↷

▶ 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](#)

## INPUT & OUTPUT :

```
... 'madam' is a palindrome: True
'A man, a plan, a canal: Panama' is a palindrome: True
'hello' is a palindrome: False
'racecar' is a palindrome: True
'No lemon, no melon' is a palindrome: True
```

## OUTPUT USING COPILOT :

The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** AIASSCoding.
- Explorer:** Shows a tree view of files in the 'AIASSCoding' folder, including 'two numbers.py', 'leapyear.py', 'Task1.py', 'Task3.py', 'Task5.py', and 'T3.py' (which is selected).
- Search Bar:** A search bar at the top right.
- Code Editor:** Displays the content of 'T3.py'. The code defines a function 'is\_palindrome' that checks if a string is a palindrome by removing spaces and converting it to lowercase. It then compares the string to its reverse. If the string is equal to its reverse, it prints that it's a palindrome; otherwise, it prints that it's not.
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS.

## INPUT & OUTPUT :

The terminal window shows the following interaction:

```
C:\Users\Terry\appdata\Local\Programs\Python\Python311\python.exe: can't open file D:\AIASSCoding\T3.py: [Errno 2] No such file or directory
(.venv) PS D:\AIASSCoding> python T3.py
(.venv) PS D:\AIASSCoding> python T3.py
(.venv) PS D:\AIASSCoding> python T3.py
(.venv) PS D:\AIASSCoding> ^C
(.venv) PS D:\AIASSCoding> & D:/AIASSCoding/.venv/Scripts/python.exe d:/AIASSCoding/T3.py
Enter a string to check if it's a palindrome: Panama
"Panama" is not a palindrome.
● (.venv) PS D:\AIASSCoding> python T3.py
Enter a string to check if it's a palindrome: radar
"radar" is a palindrome.
❖ (.venv) PS D:\AIASSCoding>
```

## 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

**Prompt:**

Explain the palindrome function line-by-line.

```
def is_palindrome(text):
    # Convert to lowercase and remove non-alphanumeric characters
    processed_text = re.sub(r'[^a-zA-Z0-9]', '', text).lower()
    # Compare the processed string with its reverse
    return processed_text == processed_text[::-1]
```

1. `import re`:

- This line imports the `re` module, which provides regular expression operations. Regular expressions are used here to easily remove non-alphanumeric characters from the string.

2. `def is_palindrome(text):`:

2. `def is_palindrome(text):`:

- This defines a function named `is_palindrome` that takes one argument, `text`, which is the string we want to check.

3. `processed_text = re.sub(r'[^a-zA-Z0-9]', '', text).lower()`:

- `re.sub(r'[^a-zA-Z0-9]', '', text)`: This is the core of the preprocessing. `re.sub()` is used to find and replace patterns in a string.

- `r'[^a-zA-Z0-9]'`: This is the regular expression pattern. It matches any character that is *not* an uppercase letter (`A-Z`), a lowercase letter (`a-z`), or a

digit (0-9). The ^ inside the square brackets [] negates the character set.

- ': This is the replacement string. All characters matched by the pattern will be replaced with an empty string, effectively removing them.
- : This is the input string on which the substitution

is performed.

- .lower(): After removing non-alphanumeric characters, this method converts the entire resulting string to lowercase. This ensures that the palindrome check is case-insensitive (e.g., 'Madam' is treated the same as 'madam').
- The result of these operations is stored in the

processed\_text variable.

#### 4. return processed\_text == processed\_text[::-1]:

- processed\_text[::-1]: This uses Python's slicing feature to create a reversed copy of the processed\_text string. The [::-1] slice means start at the end, end at the beginning, and step -1 (i.e., reverse the string).
- processed\_text == processed\_text[::-1]: This compares the original processed\_text with its reversed version. If they are identical, the expression evaluates to True, indicating it's a palindrome; otherwise, it evaluates to False.
- The return statement sends this boolean result back as the output of the function.

#### 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.