

# AI ASSISTANT CODING

Name: B. Akhira Nandhini

Hall ticket: 2303A51516

Batch:22

---

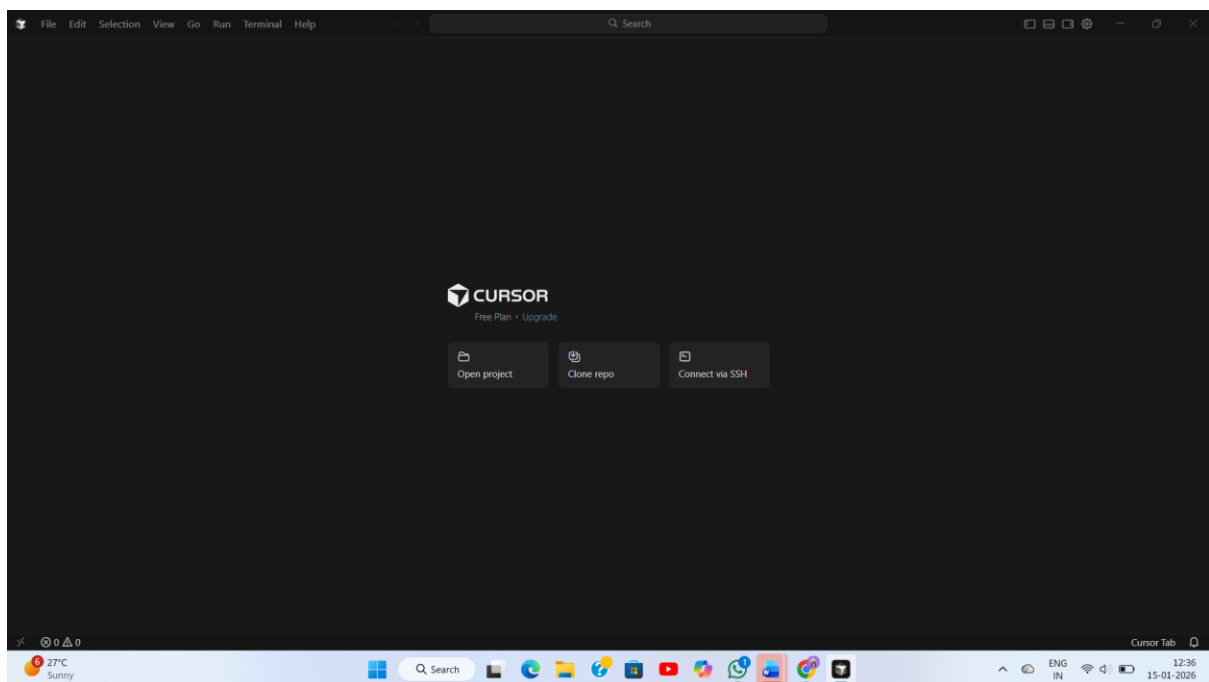
## ASSIGNMENT-02

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

#### Lab Objectives:

- ❖ To explore and evaluate the functionality of Google Gemini for AI-Week1 - Monday assisted coding within Google Colab.
- ❖ To understand and use Cursor AI for code generation, explanation, and refactoring.
- ❖ To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI.
- ❖ To perform code optimization and documentation using AI tools.

#### Lab Outcomes (LOs):



After completing this lab, students will be able to:

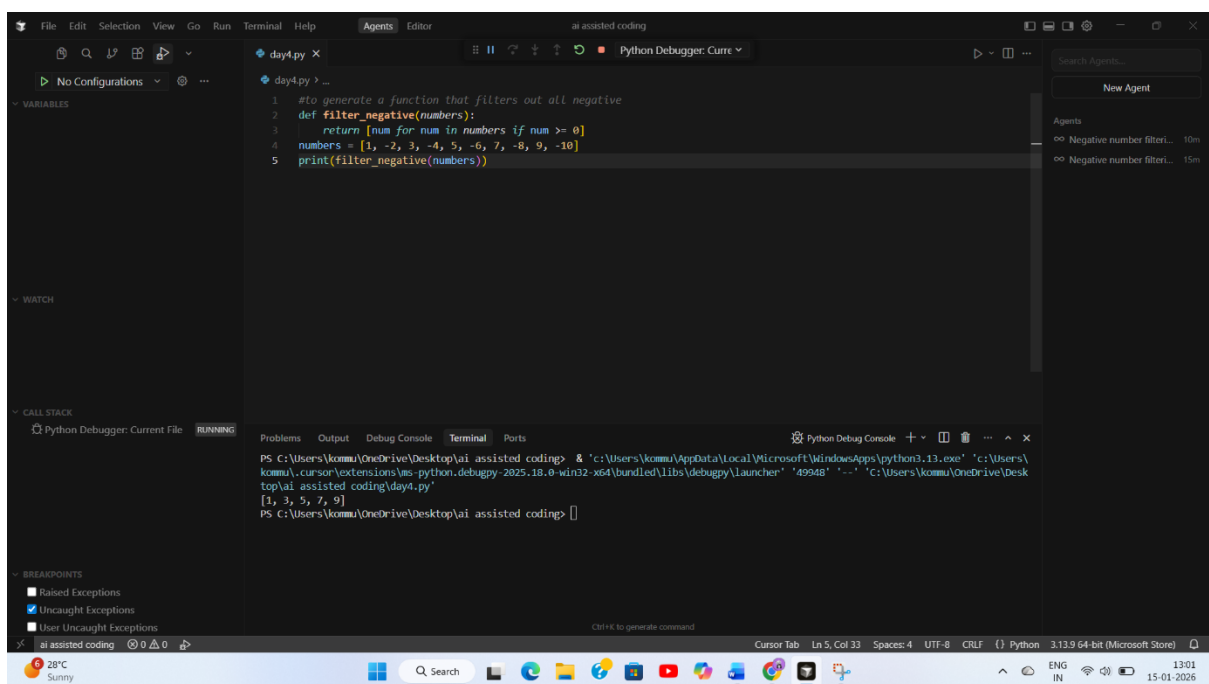
- ❖ Generate Python code using Google Gemini in Google Colab.
  - ❖ Analyze the effectiveness of code explanations and suggestions by Gemini.
  - ❖ Set up and use Cursor AI for AI-powered coding assistance.
  - ❖ Evaluate and refactor code using Cursor AI features.
  - ❖ Compare AI tool behavior and code quality across different platform.
- 

## 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.

**Code/output:**



The screenshot shows a code editor with a Python file named 'day4.py'. The code defines a function 'filter\_negative' that takes a list of numbers and returns a new list containing only the non-negative numbers. The function is then called with a list of numbers: [1, -2, 3, -4, 5, -6, 7, -8, 9, -10]. The output of the function is printed, showing the filtered list: [1, 3, 5, 7, 9]. The editor also shows a terminal window with the command 'python day4.py' and the output '[1, 3, 5, 7, 9]'. The status bar at the bottom indicates the file is 'ai assisted coding' and the editor is running on a Windows 11 system.

```
1 #to generate a function that filters out all negative
2 def filter_negative(numbers):
3     return [num for num in numbers if num >= 0]
4 numbers = [1, -2, 3, -4, 5, -6, 7, -8, 9, -10]
5 print(filter_negative(numbers))
```

```
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding> & "c:\Users\kommu\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:\Users\kommu\cursor\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher" "49948" "-." "c:\Users\kommu\OneDrive\Desktop\ai assisted coding\day4.py"
[1, 3, 5, 7, 9]
PS C:\Users\kommu\OneDrive\Desktop\ai assisted coding>
```

- Before/after list
  - Screenshot of Colab execution.
- 

## Task 2: String Character Analysis

- ❖ Scenario:

You are building a text-analysis feature.

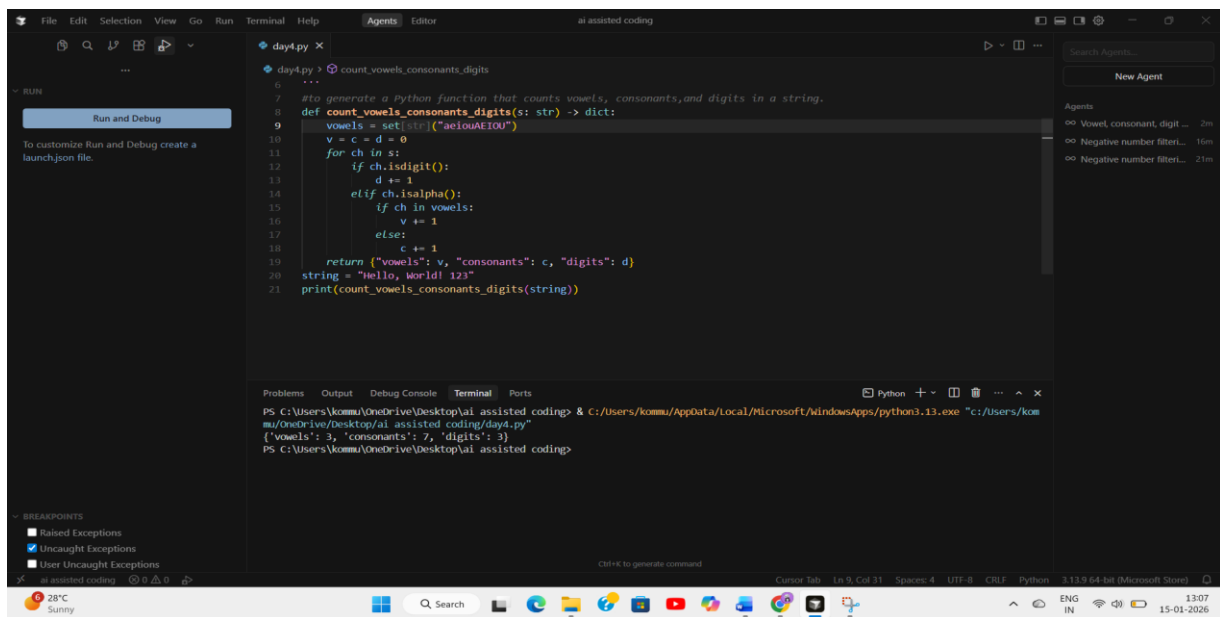
❖ Task:

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

❖ Expected Output:

➤ Working function

➤ Sample inputs and outputs.



The screenshot shows a code editor with a Python file named `day4.py`. The code defines a function `count_vowels_consonants_digits` that takes a string `s` and returns a dictionary with counts of vowels, consonants, and digits. The function uses a set of vowels and iterates through each character in the string, incrementing the respective counter. A sample string `"Hello, World! 123"` is used to test the function, and the output is printed.

```
1 # day4.py
2 # generate a Python function that counts vowels, consonants, and digits in a string.
3 def count_vowels_consonants_digits(s: str) -> dict:
4     vowels = set("aeiouAEIOU")
5     v = c = d = 0
6     for ch in s:
7         if ch.isdigit():
8             d += 1
9         elif ch.isalpha():
10            if ch in vowels:
11                v += 1
12            else:
13                c += 1
14     return {"vowels": v, "consonants": c, "digits": d}
15 string = "Hello, World! 123"
16 print(count_vowels_consonants_digits(string))
```

The terminal output shows the execution of the script, resulting in the dictionary `{'vowels': 3, 'consonants': 7, 'digits': 3}`.

### Task 3: Palindrome Check – Tool Comparison

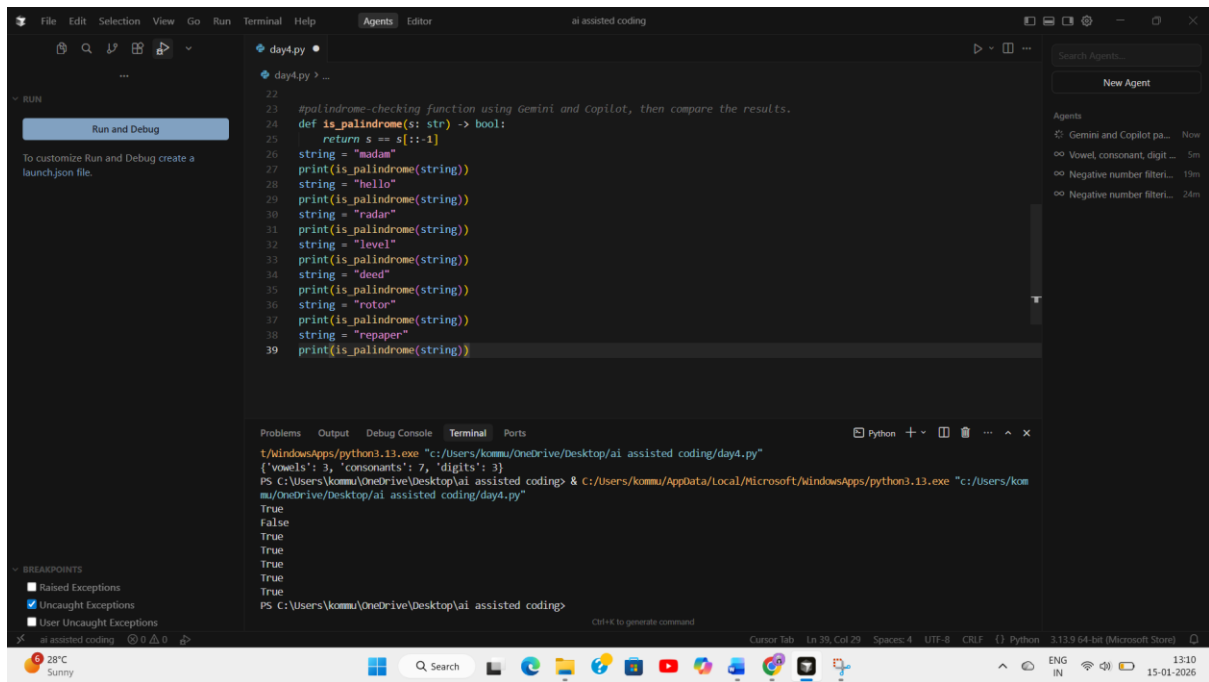
❖ Scenario:

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

## Task 4: Code Explanation Using AI

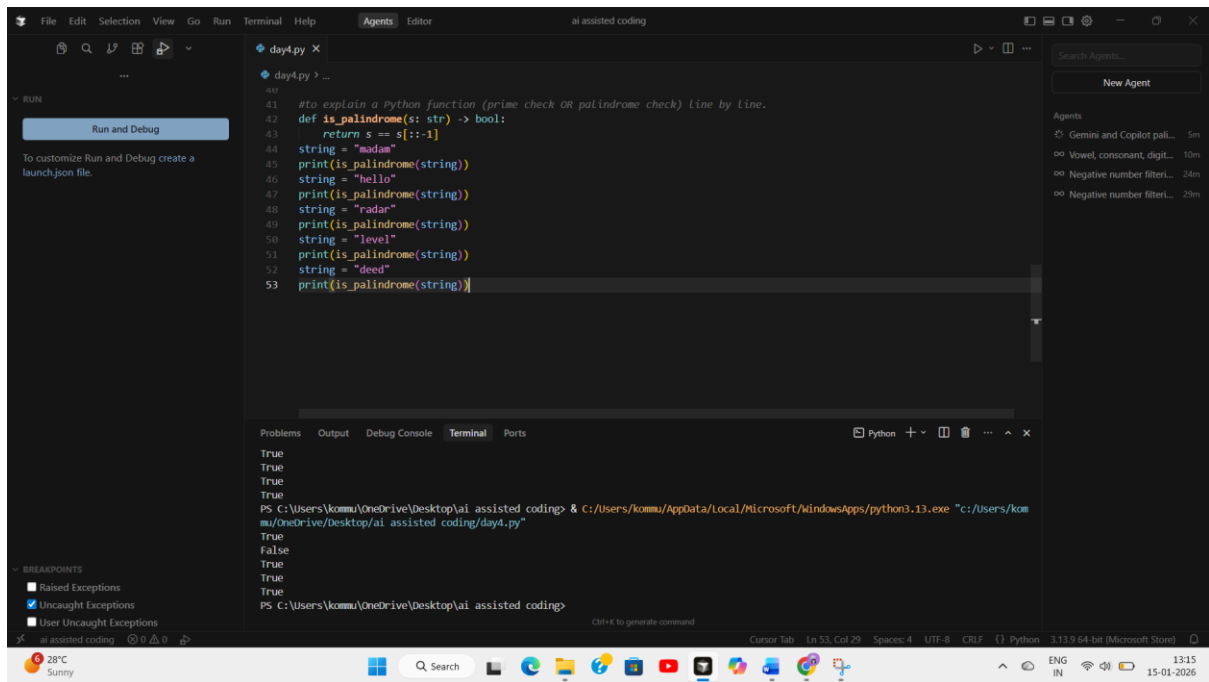
### ❖ Scenario:

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

## Code explanation for day4.py

Now · Auto

in simple words

### Simple explanation

- The function `is_palindrome(s)` checks if a word reads the same **forward and backward**.
- `s[::-1]` means **the word reversed**.
- `return s == s[::-1]` means: **if the word is the same as its reverse, return True; otherwise False**.
- Then the code tries the function with words like `"madam"`, `"hello"`, `"radar"`, etc., and prints **True** for palindromes and **False** for non-palindromes.

➤ Student comments on understanding

**My own experience using both Gemini and Cursor Ai and GitHub Copilot:**

While using Cursor Ai, I found the explanations to be very clear and helpful in understanding the logic behind the code. Cursor 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