

## ASSIGNMENT -2.2

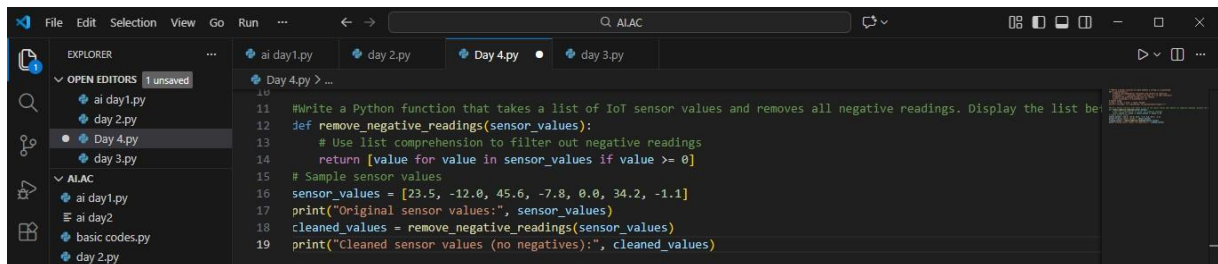
A.Abhiram

2403A51L13

Batch:51

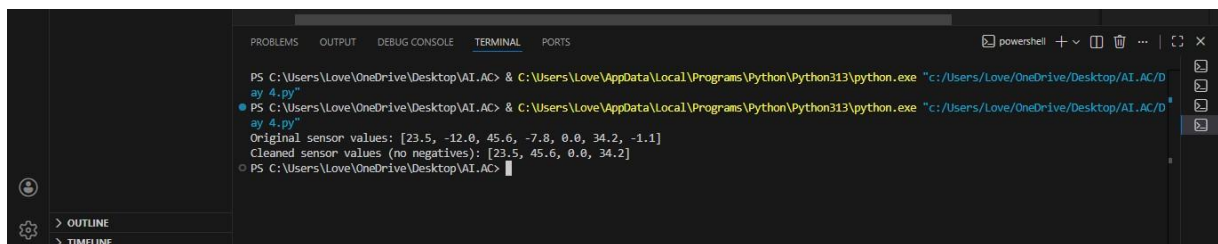
### Task 1: Cleaning Sensor Data

**PROMPT:** Write a Python function that takes a list of IoT sensor values and removes all negative readings. Display the list before and after cleaning.



```
11 #Write a Python function that takes a list of IoT sensor values and removes all negative readings. Display the list before and after cleaning.
12 def remove_negative_readings(sensor_values):
13     # Use list comprehension to filter out negative readings
14     return [value for value in sensor_values if value >= 0]
15 # Sample sensor values
16 sensor_values = [23.5, -12.0, 45.6, -7.8, 0.0, 34.2, -1.1]
17 print("Original sensor values:", sensor_values)
18 cleaned_values = remove_negative_readings(sensor_values)
19 print("Cleaned sensor values (no negatives):", cleaned_values)
```

### OUTPUT:



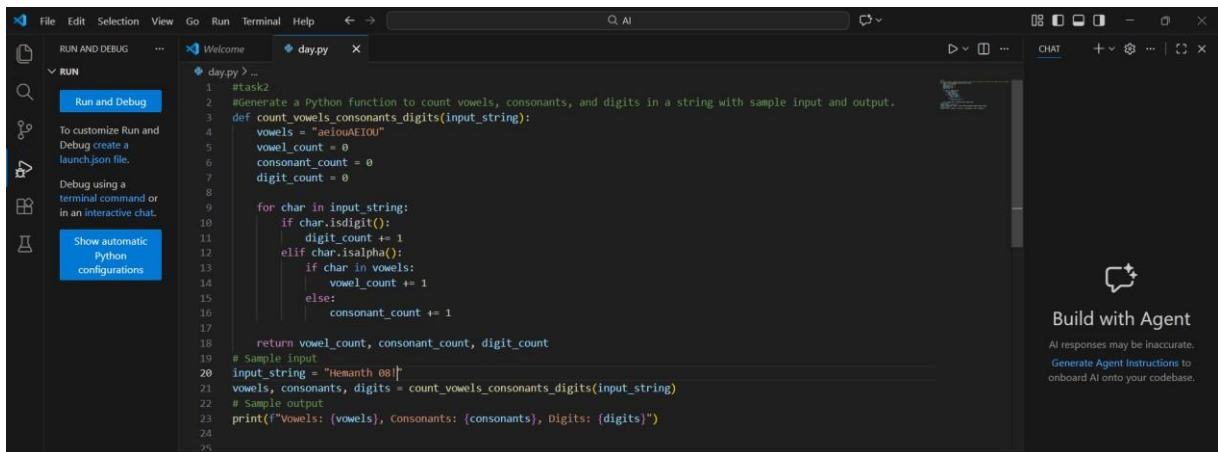
```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/Day 4.py"
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/Day 4.py"
Original sensor values: [23.5, -12.0, 45.6, -7.8, 0.0, 34.2, -1.1]
Cleaned sensor values (no negatives): [23.5, 45.6, 0.0, 34.2]
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

### EXPLANATION:

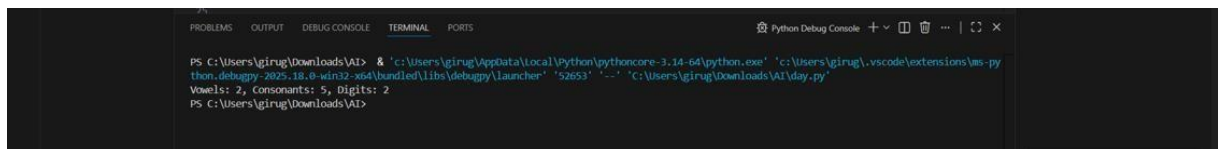
This function removes invalid negative sensor values using list comprehension. Only values greater than or equal to zero are retained, ensuring clean IoT sensor data.

### Task 2: String Character Analysis

**PROMPT:** Generate a python function to count vowels, consonants and digits in a string with sample input and output.



## OUTPUT:



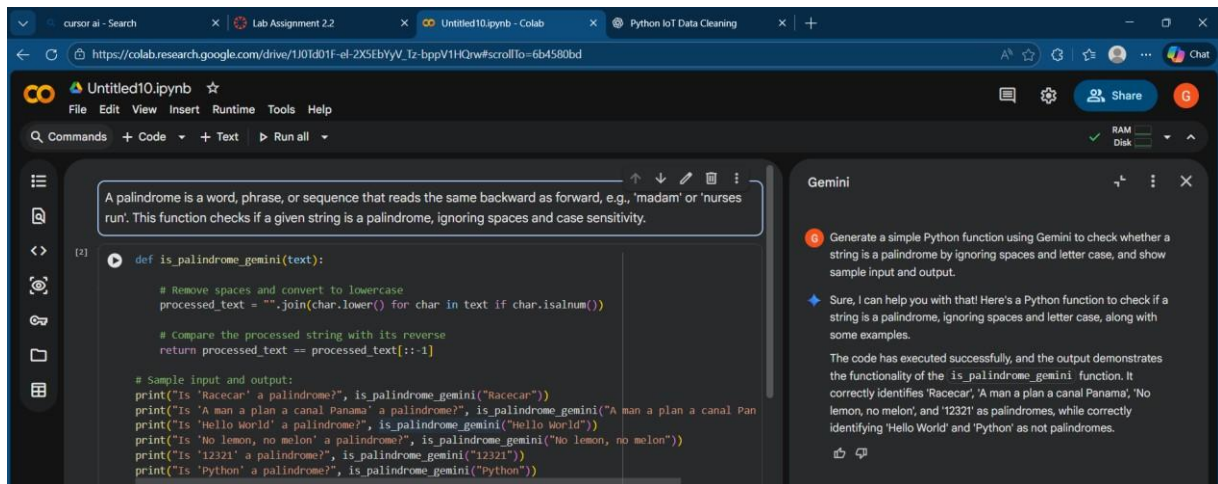
## EXPLANATION:

The function iterates through each character and classifies it as a vowel, consonant, or digit.

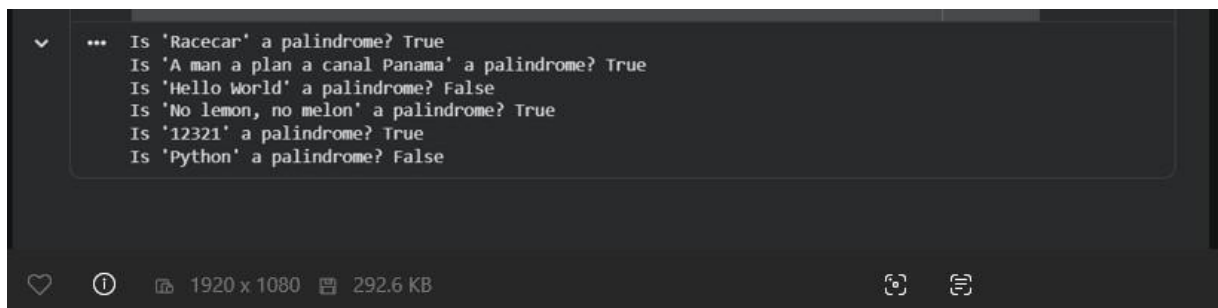
Python string methods like `isalpha()` and `isdigit()` improve accuracy and readability.

## Task 3: Palindrome Check – Tool Comparison

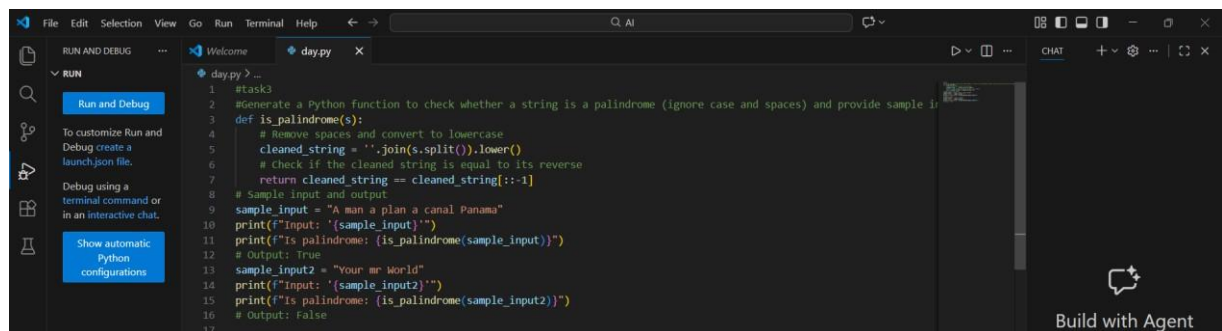
**Gemini Prompt:** Write a Python function to check if a string is a palindrome. Ignore spaces and capitalization.



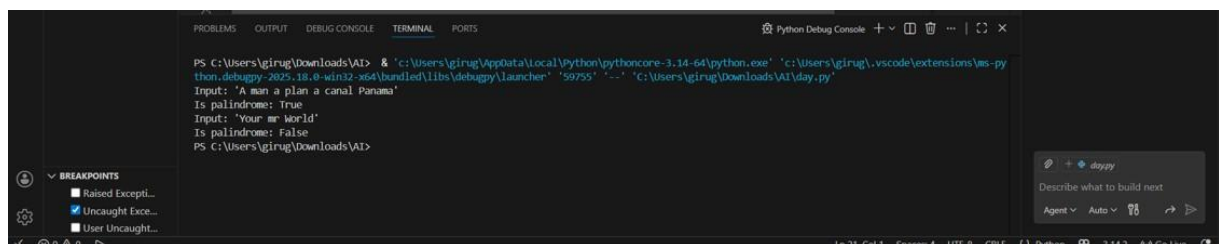
OUTPUT:



**Copilot Prompt:** Generate a python function to check whether a string is a palindrome (ignore case and spaces) and provide sample input and output.



OUTPUT:



### Comparison Table:

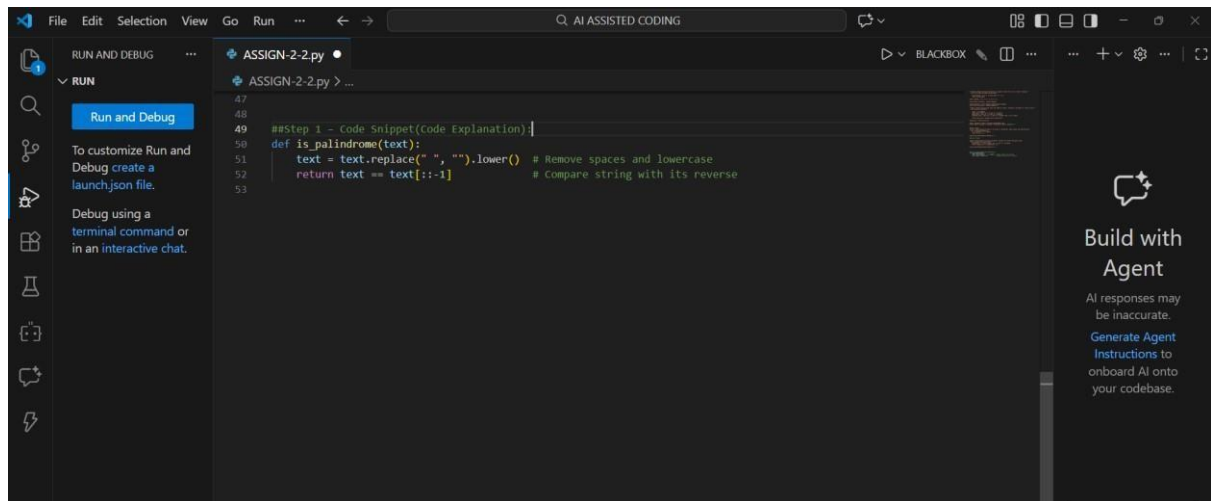
Feature	Gemini	Copilot
Clarity	Simple, minimal code	Slightly longer, more robust
Handling spaces/case	Ignores spaces, converts to lowercase	Ignores spaces and punctuation, lowercase
Readability	Very clear	Clear, slightly more detailed
Efficiency	Uses string slicing	Uses string comprehension

### EXPLANATION:

Gemini provides concise and easy-to-read logic, making it beginnerfriendly. Copilot generates more robust code that handles punctuation and special characters.

### Task 4: Code Explanation Using AI Step 1 – Code

#### Snippet:



## Step 2 – AI Explanation:

1. `text.replace(" ", "").lower()` → Removes spaces and converts letters to lowercase.
2. `text == text[::-1]` → Checks if the string is equal to its reverse.

## EXPLANATION:

The function normalizes the string to avoid case and space mismatches. It then compares the string with its reverse to verify palindrome logic.