

Sarika Pale

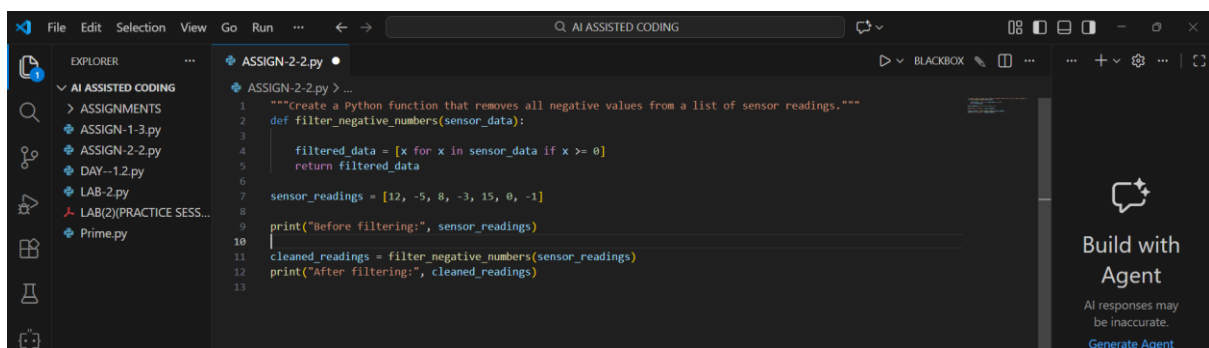
2403A51L33

B-52

ASSIGNMENT -2.2

Task 1: Cleaning Sensor Data

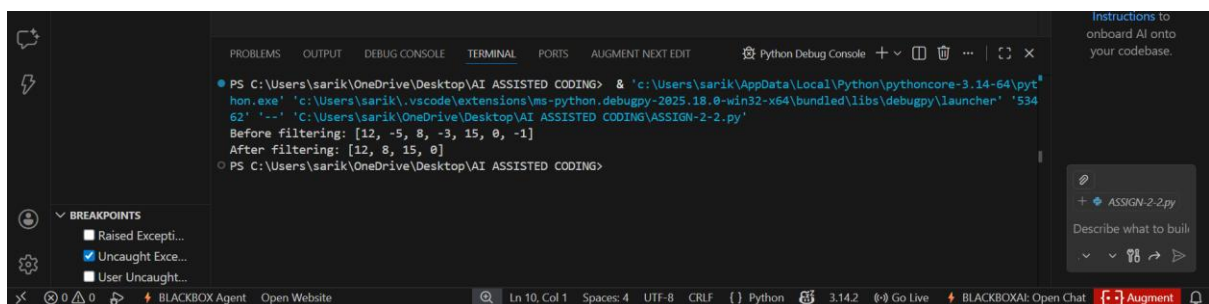
PROMPT: Create a Python function that removes all negative values from a list of sensor readings.



The screenshot shows the Visual Studio Code editor with a file named 'ASSIGN-2-2.py'. The code defines a function 'filter_negative_numbers' that takes a list 'sensor_data' and returns a new list 'filtered_data' containing only non-negative values. The function is tested with a list of sensor readings: [12, -5, 8, -3, 15, 0, -1]. The output shows the original list and the cleaned list after filtering out negative values.

```
1 """create a Python function that removes all negative values from a list of sensor readings."""
2 def filter_negative_numbers(sensor_data):
3
4     filtered_data = [x for x in sensor_data if x >= 0]
5     return filtered_data
6
7 sensor_readings = [12, -5, 8, -3, 15, 0, -1]
8
9 print("Before filtering:", sensor_readings)
10
11 cleaned_readings = filter_negative_numbers(sensor_readings)
12 print("After filtering:", cleaned_readings)
13
```

OUTPUT:



The screenshot shows the terminal output of the Python code. It displays the command to run the script and the resulting output, which shows the original list of sensor readings and the cleaned list after filtering out negative values.

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & 'c:\Users\sarik\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\sarik\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53462' '--' 'C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING\ASSIGN-2-2.py'
Before filtering: [12, -5, 8, -3, 15, 0, -1]
After filtering: [12, 8, 15, 0]
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

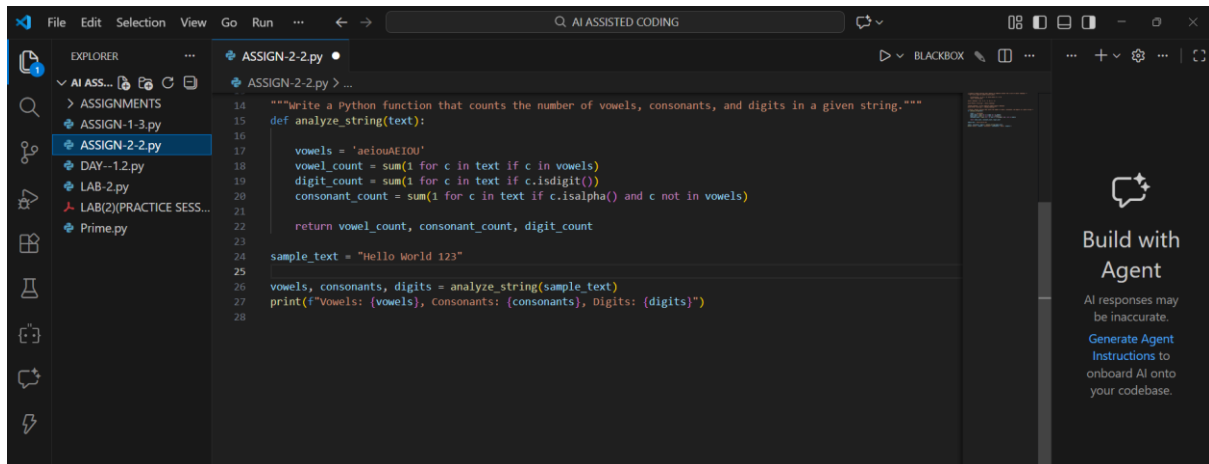
EXPLANATION:

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

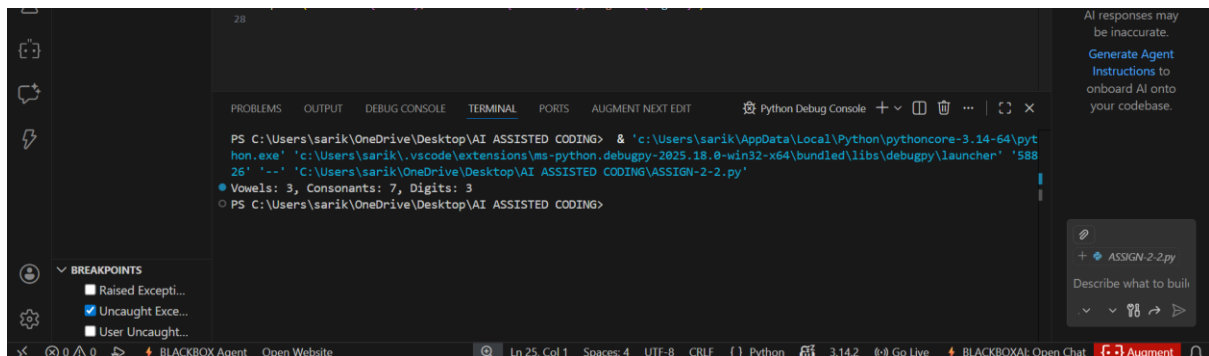
PROMPT: Write a Python function that counts the number of vowels, consonants, and digits in a given string.



The screenshot shows the Visual Studio Code editor with a Python file named 'ASSIGN-2-2.py'. The code defines a function 'analyze_string(text)' that counts vowels, consonants, and digits in a given string. The function uses the 'vowels' string 'aeiouAEIOU' and the 'isalpha()' and 'isdigit()' methods. A sample text 'Hello World 123' is used to demonstrate the function's output.

```
14 """write a Python function that counts the number of vowels, consonants, and digits in a given string."""
15 def analyze_string(text):
16
17     vowels = 'aeiouAEIOU'
18     vowel_count = sum(1 for c in text if c in vowels)
19     digit_count = sum(1 for c in text if c.isdigit())
20     consonant_count = sum(1 for c in text if c.isalpha() and c not in vowels)
21
22     return vowel_count, consonant_count, digit_count
23
24 sample_text = "Hello World 123"
25
26 vowels, consonants, digits = analyze_string(sample_text)
27 print(f"Vowels: {vowels}, Consonants: {consonants}, Digits: {digits}")
28
```

OUTPUT:



The screenshot shows the terminal output of the Python code. The command executed is 'python .\ASSIGN-2-2.py' and the output is 'Vowels: 3, Consonants: 7, Digits: 3'.

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> python .\ASSIGN-2-2.py
Vowels: 3, Consonants: 7, Digits: 3
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

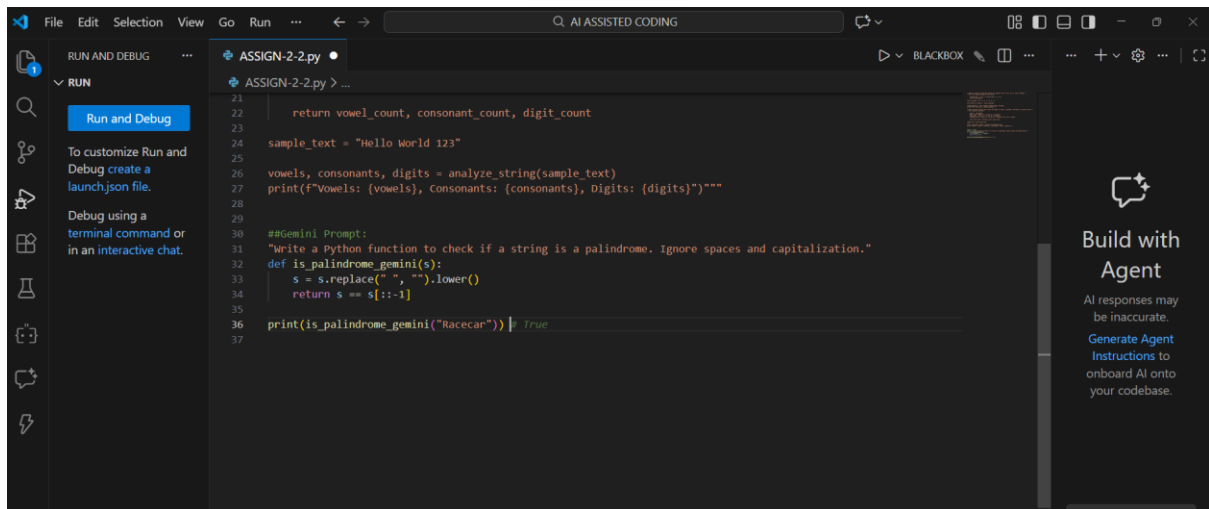
EXPLANATION:

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

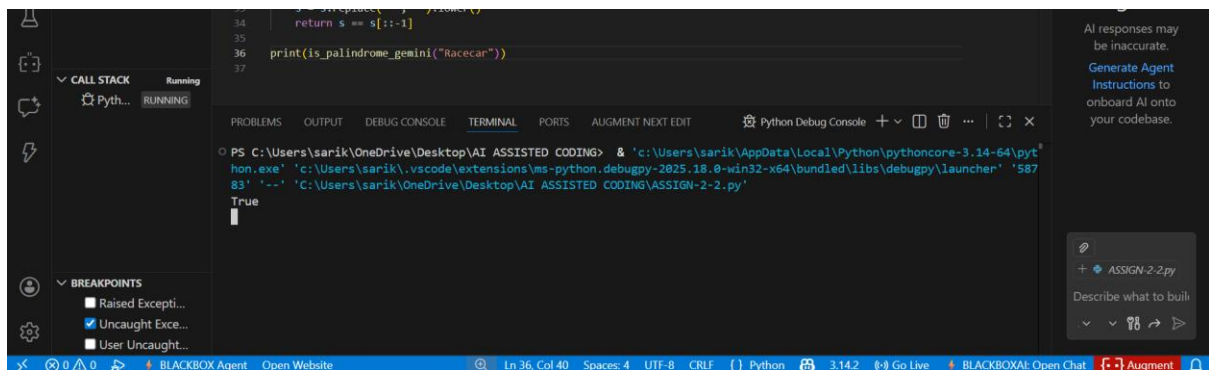
Python string methods like 'isalpha()' and 'isdigit()' improve accuracy and readability.

Task 3: Palindrome Check – Tool Comparison

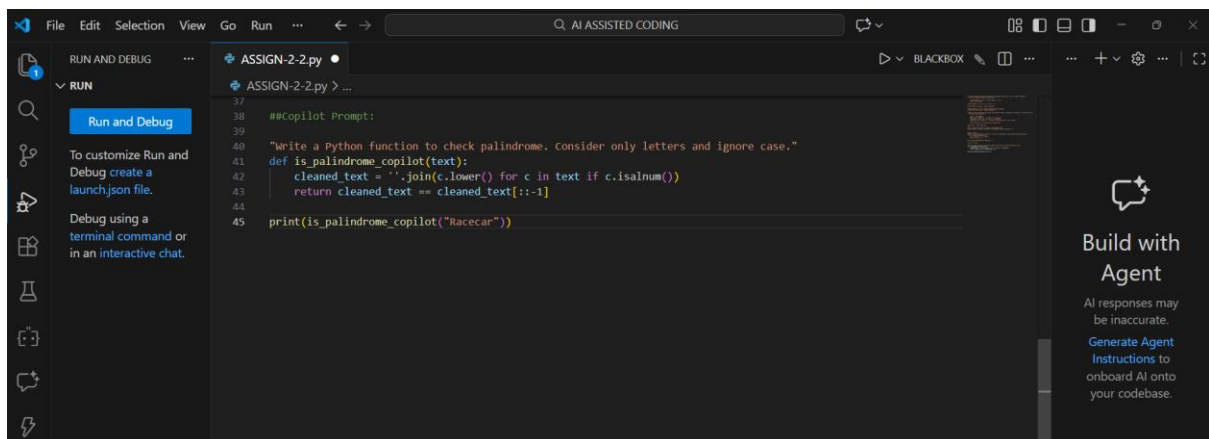
Gemini Prompt: Write a Python function to check if a string is a palindrome. Ignore spaces and capitalization.



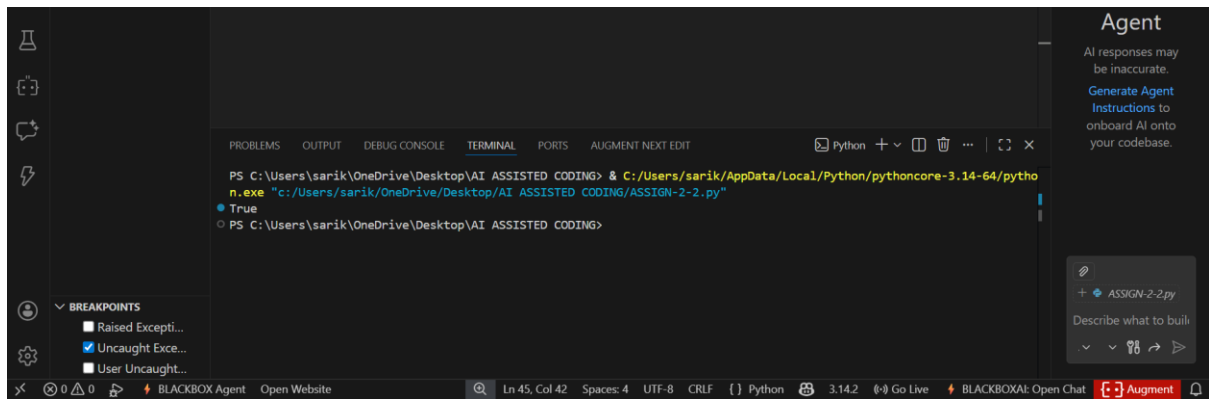
OUTPUT:



Copilot Prompt: Write a Python function to check palindrome. Consider only letters and ignore case.



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

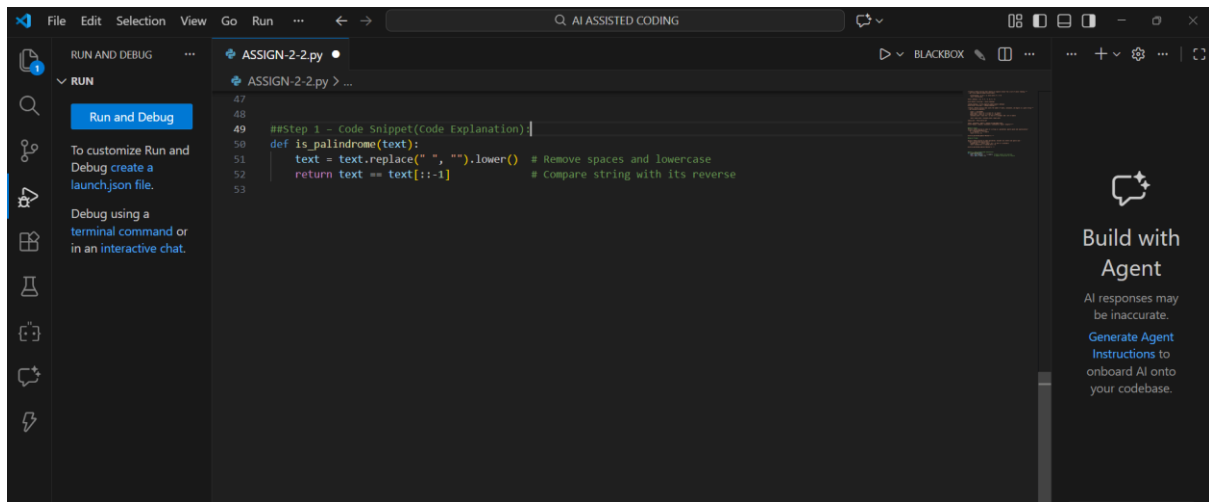
EXPLANATION:

Gemini provides concise and easy-to-read logic, making it beginner-friendly.

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.