

Sarika Palle

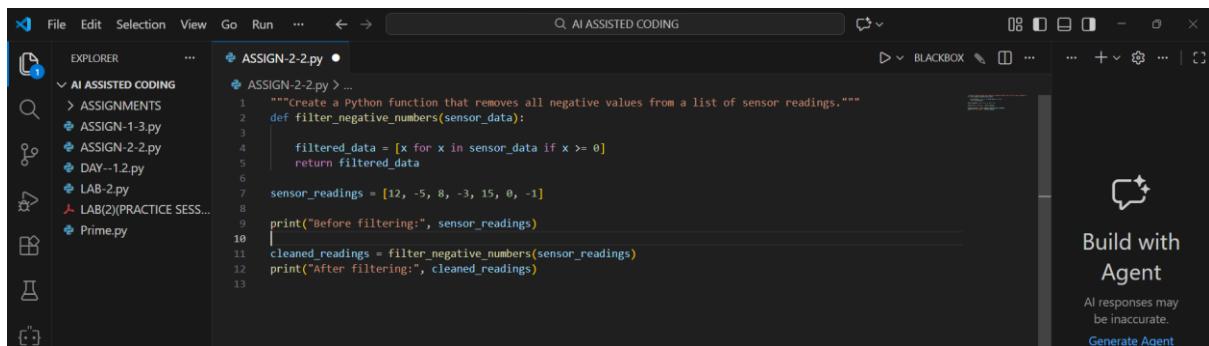
2403A51L33

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.

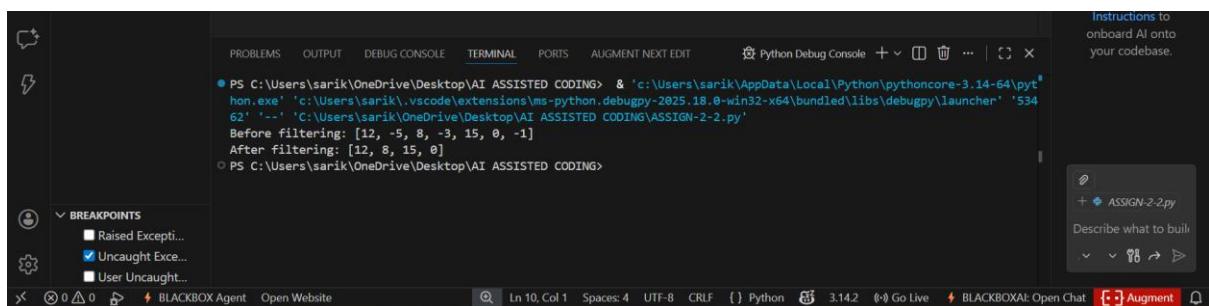


A screenshot of the Visual Studio Code interface. The left sidebar shows the file tree with files like ASSIGN-1-3.py, ASSIGN-2-2.py, DAY--12.py, LAB-2.py, and Prime.py. The main editor window contains the following Python code:

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

The right side of the interface features a "BLACKBOX" extension panel with a "Build with Agent" button and a note about AI responses being inaccurate. A "Generate Agent" button is also present.

### OUTPUT:



A screenshot of the VS Code terminal window. The output shows the execution of the Python script and its results:

```
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>
```

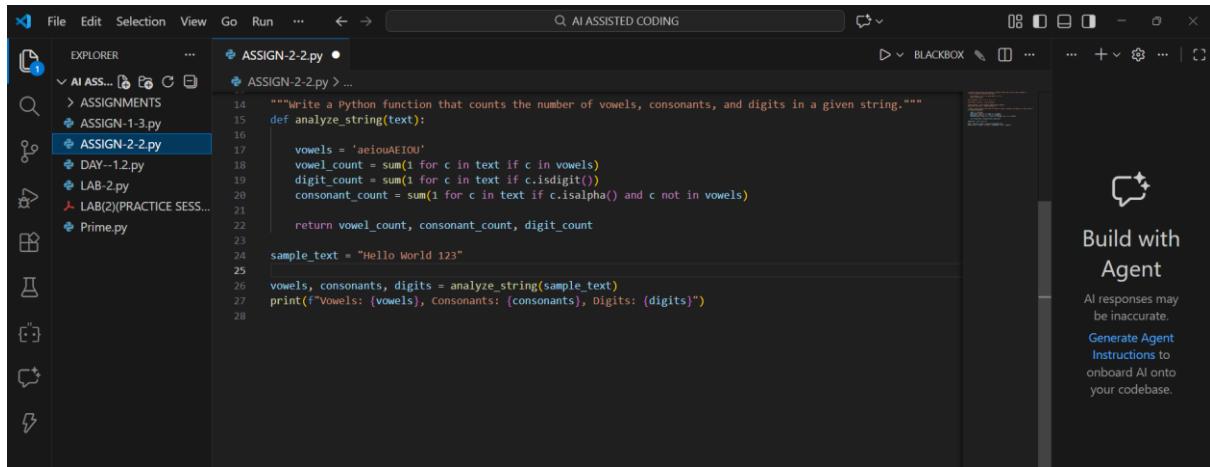
### EXPLANATION:

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

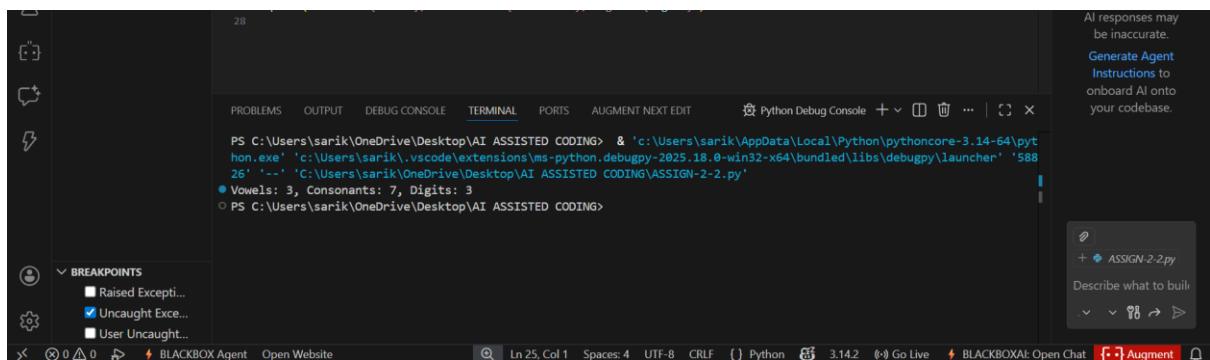
**PROMPT:** Write a Python function that counts the number of vowels, consonants, and digits in a given string.



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

The right sidebar shows a "Build with Agent" panel with the following text:  
AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

## OUTPUT:



```
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' '58826' '--' 'C:\Users\sarik\Desktop\AI ASSISTED CODING\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.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a file tree with "RUN AND DEBUG" and "RUN".
- Run View:** Displays the "Run and Debug" button.
- Terminal:** Shows a message: "To customize Run and Debug create a launch.json file." Below it, another message says: "Debug using a terminal command or in an interactive chat."
- Code Editor:** The code for "ASSIGN-2-2.py" is displayed:

```
1 #!/usr/bin/env python3
2
3 def is_palindrome_gemini(s):
4     s = s.replace(" ", "").lower()
5     return s == s[::-1]
6
7 print(is_palindrome_gemini("Racecar")) # True
```
- Output Panel:** Shows the output "True".
- AI Assistant:** A sidebar titled "Build with Agent" contains the text: "AI responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase."

## **OUTPUT:**

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

- Code Editor:** A snippet of Python code is shown, including a function definition for `is\_palindrome` and a call to it with the argument "Racecar".
- Terminal:** The terminal window displays the command PS C:\Users\... and the resulting output of the Python script, which prints "True".
- Output:** The Output tab shows the command run and the output "True".
- Breakpoints:** The Breakpoints sidebar shows three types of breakpoints: Raised Excepti..., Uncought Excep..., and User Uncought....
- Python Debug Console:** A separate window titled "Python Debug Console" is open, showing the command run and the output "True".
- Bottom Status Bar:** The status bar indicates the file is "BLACKBOX Agent", the line is "Ln 36, Col 40", and the character count is "Spaces: 4".

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

File Edit Selection View Go Run ... ← → Q AI ASSISTED CODING ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ⌊ ⌋

RUN AND DEBUG ...

RUN

Run and Debug

To customize Run and Debug create a launch.json file.

Debug using a terminal command or in an interactive chat.

ASSIGN-2-2.py ●

ASSIGN-2-2.py > ...

```
37
38     #Copilot Prompt:
39
40     "Write a Python function to check palindrome. Consider only letters and ignore case."
41     def is_palindrome_copilot(text):
42         cleaned_text = ''.join(c.lower() for c in text if c.isalnum())
43         return cleaned_text == cleaned_text[::-1]
44
45     print(is_palindrome_copilot("Racecar"))
```

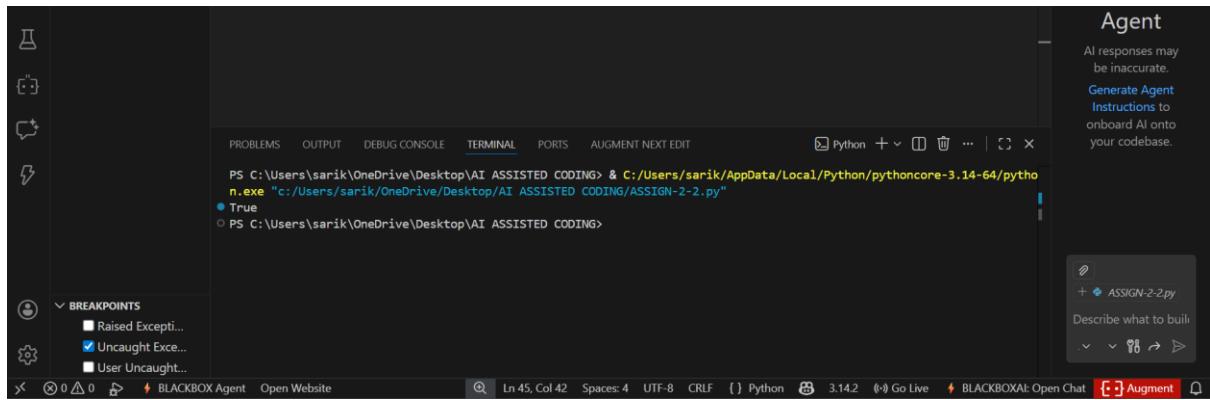
BLACKBOX ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ⌊ ⌋

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

## **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:

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with icons for file operations like Open, Save, Find, and Run. The main area displays a Python script named 'ASSIGN-2.py'. The code contains the following snippet:

```
47
48     ##Step 1 - Code Snippet(Code Explanation):
49     def is_palindrome(text):
50         text = text.replace(" ", "").lower() # Remove spaces and lowercase
51         return text == text[::-1]           # Compare string with its reverse
52
```

To the right of the code, there's a vertical panel titled 'Build with Agent' which includes a message about AI responses being inaccurate and a link to generate agent instructions.

## 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.