

ASSIGNMENT -2.2

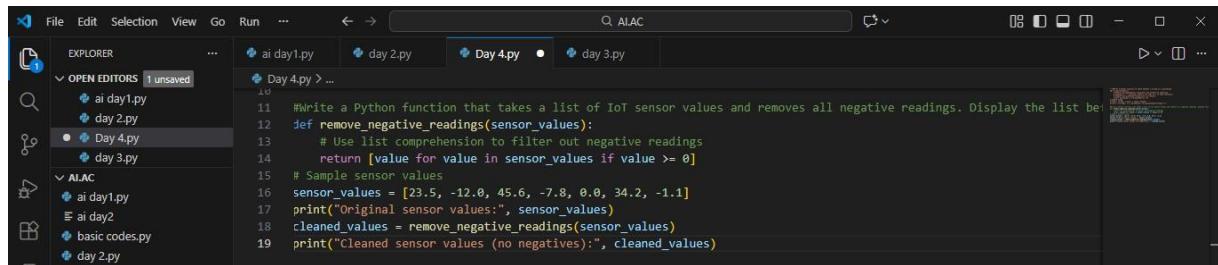
A.Abiram

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.



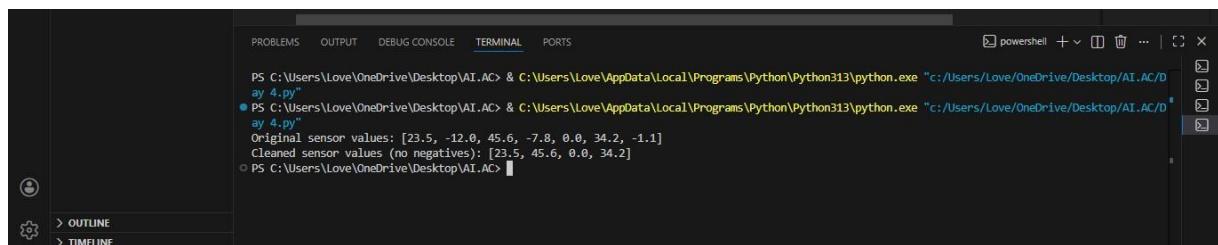
```
File Edit Selection View Go Run ... ⟲ ⟳ Day 4.py • Day 3.py

OPEN EDITORS 1 unsaved
ai day1.py
day 2.py
Day 4.py
day 3.py

ALAC
ai day1.py
ai day2
basic codes.py
day 2.py

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:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell + - x

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]
```

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.

```
#task2
#Generate a Python function to count vowels, consonants, and digits in a string with sample input and output.
def count_vowels_consonants_digits(input_string):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0
    digit_count = 0

    for char in input_string:
        if char.isdigit():
            digit_count += 1
        elif char.isalpha():
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count, digit_count

# Sample input
input_string = "Neeranah osai"
vowels, consonants, digits = count_vowels_consonants_digits(input_string)
# Sample output
print(f"Vowels: {vowels}, Consonants: {consonants}, Digits: {digits}")
```

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

OUTPUT:

```
PS C:\Users\girug\Downloads\AI> & 'c:\users\girug\appdata\local\python\pythoncore-3.14-64\python.exe' 'c:\users\girug\vscode\extensions\ms-python.debugger-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '52653' '--' 'C:\Users\girug\Downloads\AI\day.py'
Vowels: 2, consonants: 5, Digits: 2
PS C:\Users\girug\Downloads\AI>
```

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.

A screenshot of Google Colab showing a code cell and a Gemini AI sidebar.

Code Cell Content:

```

def is_palindrome_gemini(text):
    # Remove spaces and convert to lowercase
    processed_text = ''.join(char.lower() for char in text if char.isalnum())

    # Compare the processed string with its reverse
    return processed_text == processed_text[::-1]

# Sample input and output:
print("Is 'Racecar' a palindrome?", is_palindrome_gemini("Racecar"))
print("Is 'A man a plan a canal Panama' a palindrome?", is_palindrome_gemini("A man a plan a canal Panama"))
print("Is 'Hello World' a palindrome?", is_palindrome_gemini("Hello World"))
print("Is 'No lemon, no melon' a palindrome?", is_palindrome_gemini("No lemon, no melon"))
print("Is '12321' a palindrome?", is_palindrome_gemini("12321"))
print("Is 'Python' a palindrome?", is_palindrome_gemini("Python"))

```

Gemini Sidebar:

Generate a simple Python function using Gemini to check whether a string is a palindrome by ignoring spaces and letter case, and show sample input and output.

Sure, I can help you with that! Here's a Python function to check if a string is a palindrome, ignoring spaces and letter case, along with some examples.

The code has executed successfully, and the output demonstrates the functionality of the `is_palindrome_gemini` function. It correctly identifies "Racecar", "A man a plan a canal Panama", "No lemon, no melon", and "12321" as palindromes, while correctly identifying "Hello World" and "Python" as not palindromes.

OUTPUT:

Execution results from the Colab output pane:

- Is 'Racecar' a palindrome? True
- Is 'A man a plan a canal Panama' a palindrome? True
- Is 'Hello World' a palindrome? False
- Is 'No lemon, no melon' a palindrome? True
- Is '12321' a palindrome? True
- Is 'Python' a palindrome? False

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

VS Code interface showing the generation of a Python function:

```

# Task3
# Generate a Python function to check whether a string is a palindrome (ignore case and spaces) and provide sample input and output
def is_palindrome(s):
    # Remove spaces and convert to lowercase
    cleaned_string = ''.join(s.split()).lower()
    # Check if the cleaned string is equal to its reverse
    return cleaned_string == cleaned_string[::-1]

# Sample input and output
sample_input = "A man a plan a canal Panama"
print(f"Input: {sample_input}")
print(f"Is palindrome: {is_palindrome(sample_input)}")
# Output: True
sample_input2 = "Your mr World"
print(f"Input: {sample_input2}")
print(f"Is palindrome: {is_palindrome(sample_input2)}")
print(f"Output: {is_palindrome(sample_input2)}")
# Output: False

```

OUTPUT:

VS Code Python Debug Console output:

```

PS C:\Users\girug\Downloads\VAI> & 'c:\Users\girug\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\girug\.vscode\extensions\ms-python.python.debug-2025.18.0-win32-x64\bundled\l1bs\debug\launcher' '5975' '--' 'C:\Users\girug\Downloads\VAI\day.py'
Input: A man a plan a canal Panama
Is palindrome: True
Input: Your mr World
Is palindrome: False
PS C:\Users\girug\Downloads\VAI>

```

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:

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar titled 'RUN AND DEBUG' with a 'Run and Debug' button highlighted. Below it, instructions say 'To customize Run and Debug create a launch.json file.' and 'Debug using a terminal command or in an interactive chat.' The main area displays a Python script named 'ASSIGN-2-2.py'. The code is as follows:

```
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
53
```

To the right of the code, there's a 'BLACKBOX' section showing some AI-generated code. A sidebar on the right says 'Build with Agent' with a note that 'AI responses may be inaccurate.' and a link to 'Generate Agent Instructions to onboard AI onto your codebase.'

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.