Vyshnavi Parisha

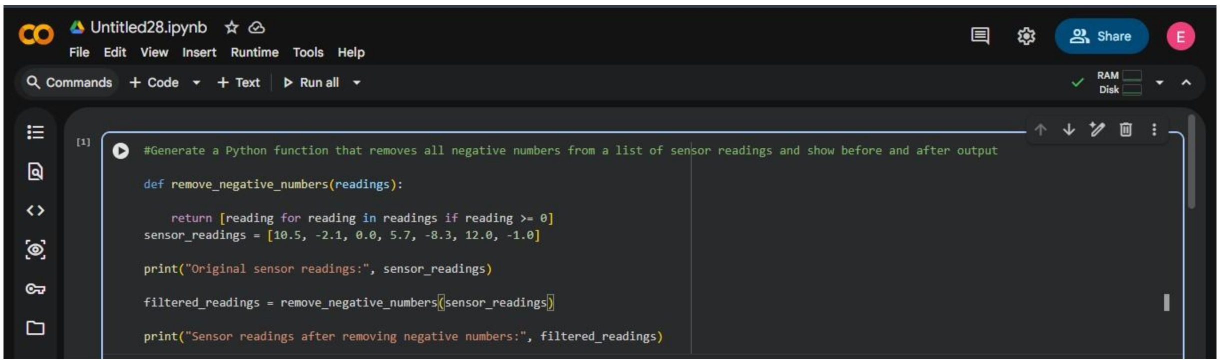
2403A51L34

B-52

ASSIGNMENT -2.2

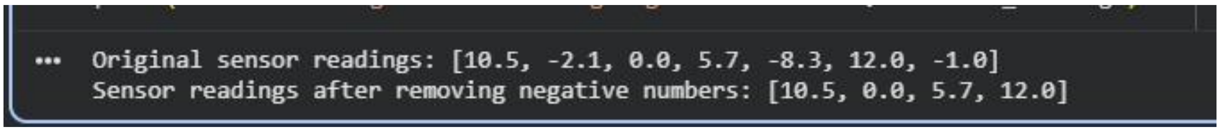
Task 1: Cleaning Sensor Data

PROMPT: Generate a Python function that removes all negative numbers from a list of sensor readings and show before and after output


Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM
Disk
[1]

```python
#Generate a Python function that removes all negative numbers from a list of sensor readings and show before and after output

def remove_negative_numbers(readings):

    return [reading for reading in readings if reading >= 0]
sensor_readings = [10.5, -2.1, 0.0, 5.7, -8.3, 12.0, -1.0]

print("Original sensor readings:", sensor_readings)

filtered_readings = remove_negative_numbers(sensor_readings)

print("Sensor readings after removing negative numbers:", filtered_readings)
```

OUTPUT:



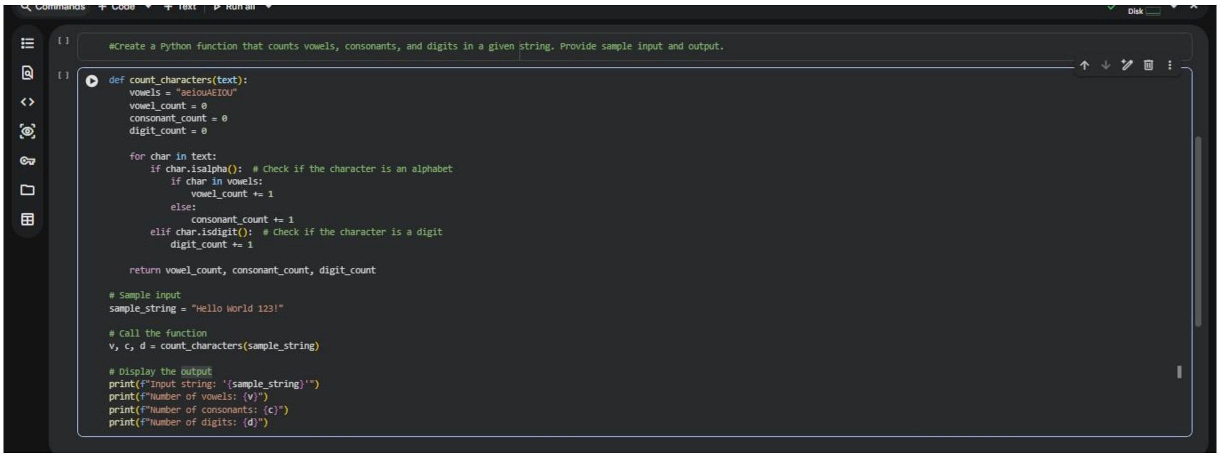
```
Original sensor readings: [10.5, -2.1, 0.0, 5.7, -8.3, 12.0, -1.0]
Sensor readings after removing negative numbers: [10.5, 0.0, 5.7, 12.0]
```

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.

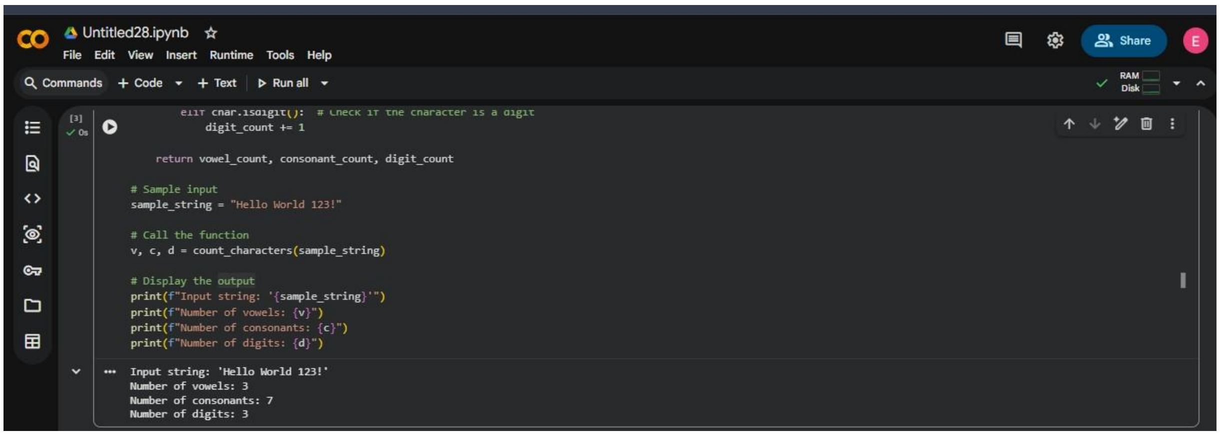
Task 2: String Character Analysis

PROMPT: Create a Python function that counts vowels, consonants, and digits in a given string. Provide sample input and output.



```
#Create a Python function that counts vowels, consonants, and digits in a given string. Provide sample input and output.
```

```python
def count_characters(text):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0
    digit_count = 0

    for char in text:
        if char.isalpha():  # Check if the character is an alphabet
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1
        elif char.isdigit():  # Check if the character is a digit
            digit_count += 1

    return vowel_count, consonant_count, digit_count

# Sample input
sample_string = "Hello World 123!"

# Call the function
v, c, d = count_characters(sample_string)

# Display the output
print(f"Input string: '{sample_string}'")
print(f"Number of vowels: {v}")
print(f"Number of consonants: {c}")
print(f"Number of digits: {d}")
```



```python
        elif char.isdigit():  # Check if the character is a digit
            digit_count += 1

    return vowel_count, consonant_count, digit_count

# Sample input
sample_string = "Hello World 123!"

# Call the function
v, c, d = count_characters(sample_string)

# Display the output
print(f"Input string: '{sample_string}'")
print(f"Number of vowels: {v}")
print(f"Number of consonants: {c}")
print(f"Number of digits: {d}")
```

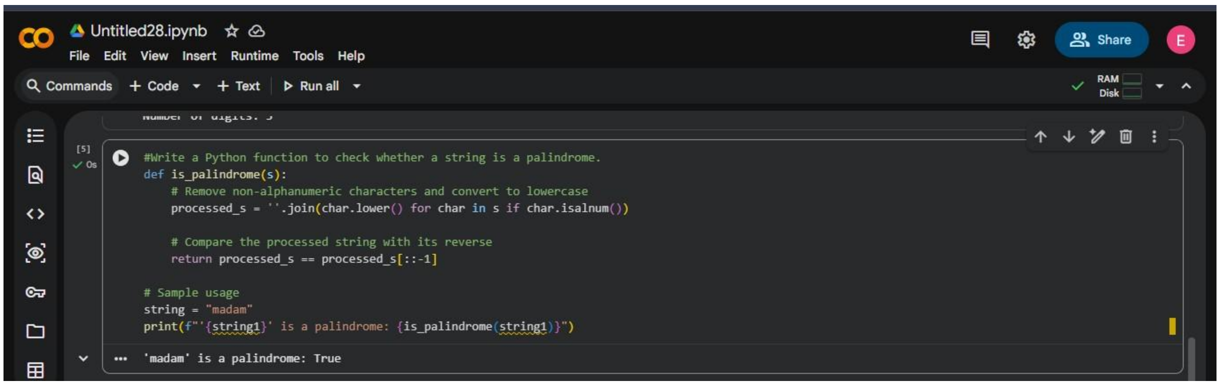
```
Input string: 'Hello World 123!'
Number of vowels: 3
Number of consonants: 7
Number of digits: 3
```

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.

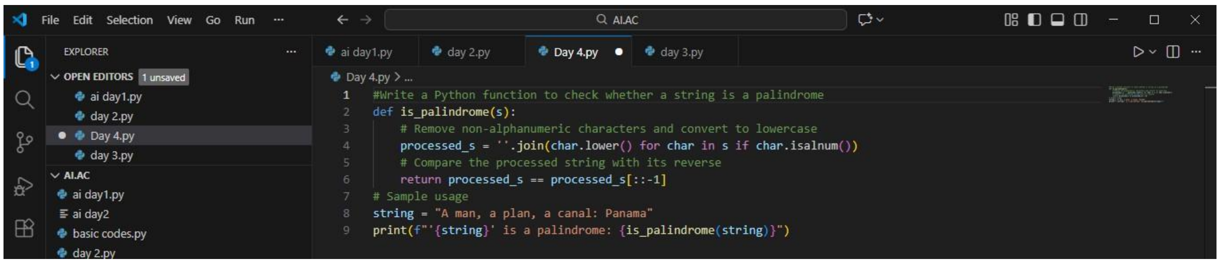

Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk

```
Number of digits: 5
```

```python
#Write a Python function to check whether a string is a palindrome.
def is_palindrome(s):
    # Remove non-alphanumeric characters and convert to lowercase
    processed_s = ''.join(char.lower() for char in s if char.isalnum())

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

# Sample usage
string = "madam"
print(f"'{string1}' is a palindrome: {is_palindrome(string1)}")
```

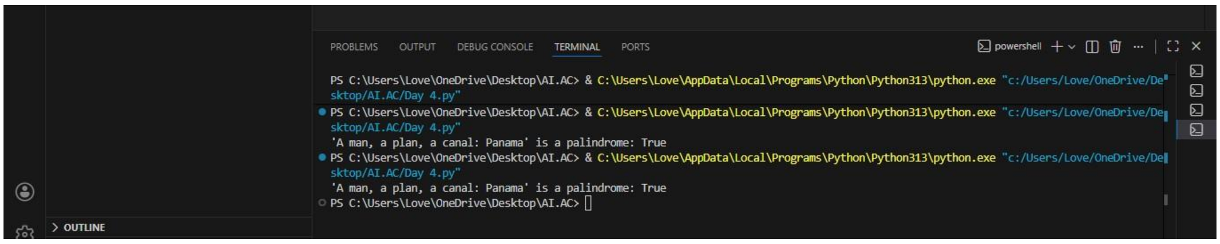
```
'madam' is a palindrome: True
```

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



```python
#Write a Python function to check whether a string is a palindrome
def is_palindrome(s):
    # Remove non-alphanumeric characters and convert to lowercase
    processed_s = ''.join(char.lower() for char in s if char.isalnum())
    # Compare the processed string with its reverse
    return processed_s == processed_s[::-1]
# Sample usage
string = "A man, a plan, a canal: Panama"
print(f"'{string}' is a palindrome: {is_palindrome(string)}")
```

OUTPUT:



```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/De
sktop/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/De
sktop/AI.AC/Day 4.py"
'A man, a plan, a canal: Panama' is a palindrome: True
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/De
sktop/AI.AC/Day 4.py"
'A man, a plan, a canal: Panama' is a palindrome: True
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

Comparison Table:

Feature	**Gemini**	**Copilot**

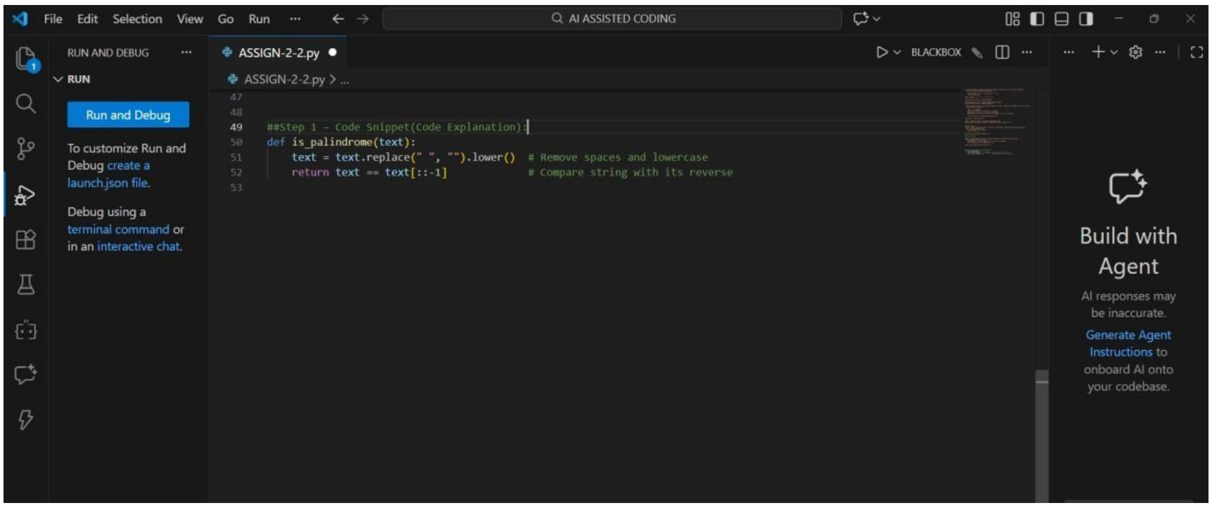
Clarity	Simple, minimal code	Slightly longer, more robust
Handling spaces/case	Ignores spaces, converts to lowercase	Ignores spaces and punctuation, lowercase
Readability	Very clear	Clear, slightly more detailed
Efficiency	Uses string slicing	Uses string comprehension

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:



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

Debug using a terminal command or in an interactive chat.

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

AI responses may be inaccurate.

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.