AI ASSISTED CODING

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

T.SNUHITH REDDY **2303A510H9**

BATCH – 03 **13 – 01 – 2026**

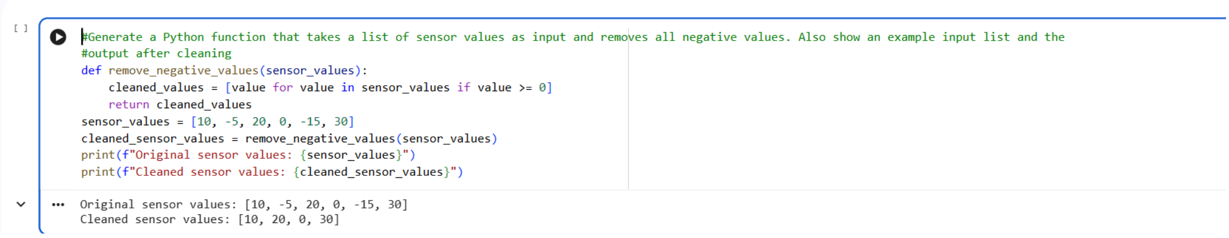
ASSIGNMENT – 2.2

Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI.

Task 1: Cleaning Sensor Data

Prompt: I am working with IoT sensor data where negative values are invalid. Generate a Python function that removes all negative numbers from a given list. Also print the list before cleaning and after cleaning. Provide a sample input and output.

Code:



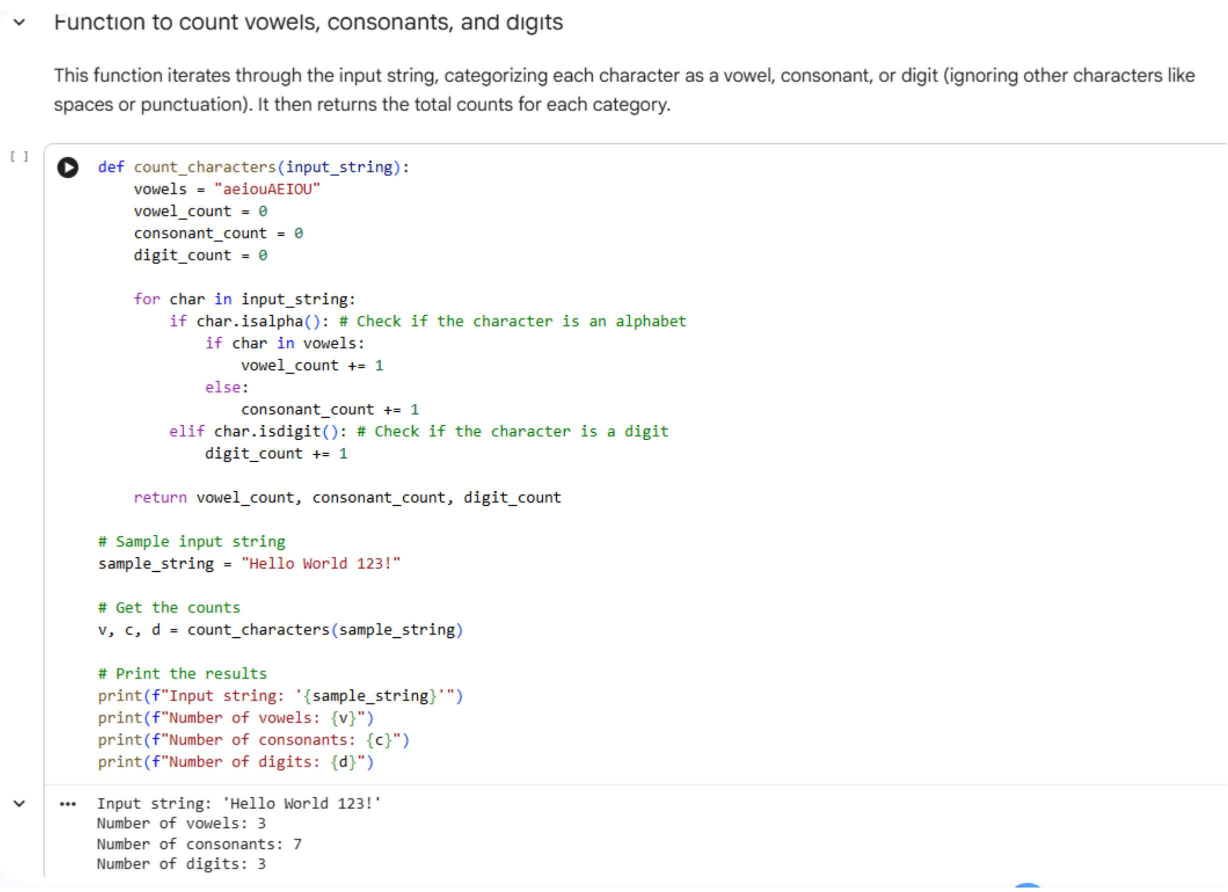
```
#Generate a Python function that takes a list of sensor values as input and removes all negative values. Also show an example input list and the
#output after cleaning
def remove_negative_values(sensor_values):
    cleaned_values = [value for value in sensor_values if value >= 0]
    return cleaned_values
sensor_values = [10, -5, 20, 0, -15, 30]
cleaned_sensor_values = remove_negative_values(sensor_values)
print(f"Original sensor values: {sensor_values}")
print(f"Cleaned sensor values: {cleaned_sensor_values}")
```

```
Original sensor values: [10, -5, 20, 0, -15, 30]
Cleaned sensor values: [10, 20, 0, 30]
```

Task 2: String Character Analysis.

Prompt: Create a Python function that analyzes a given string and counts the number of vowels, consonants, and digits present in it.
Show the function, explain briefly how it works, and provide sample inputs with corresponding outputs.

Code:



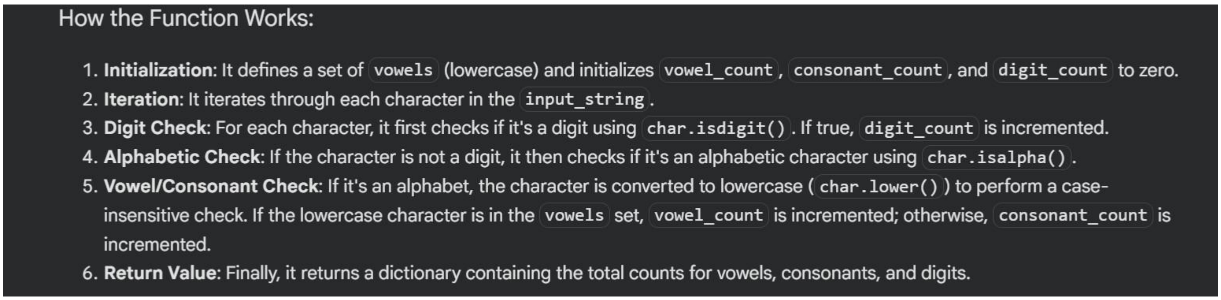
Function to count vowels, consonants, and digits

This function iterates through the input string, categorizing each character as a vowel, consonant, or digit (ignoring other characters like spaces or punctuation). It then returns the total counts for each category.

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

    for char in input_string:
        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 string
sample_string = "Hello World 123!"

# Get the counts
v, c, d = count_characters(sample_string)

# Print the results
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
```

Working Function:



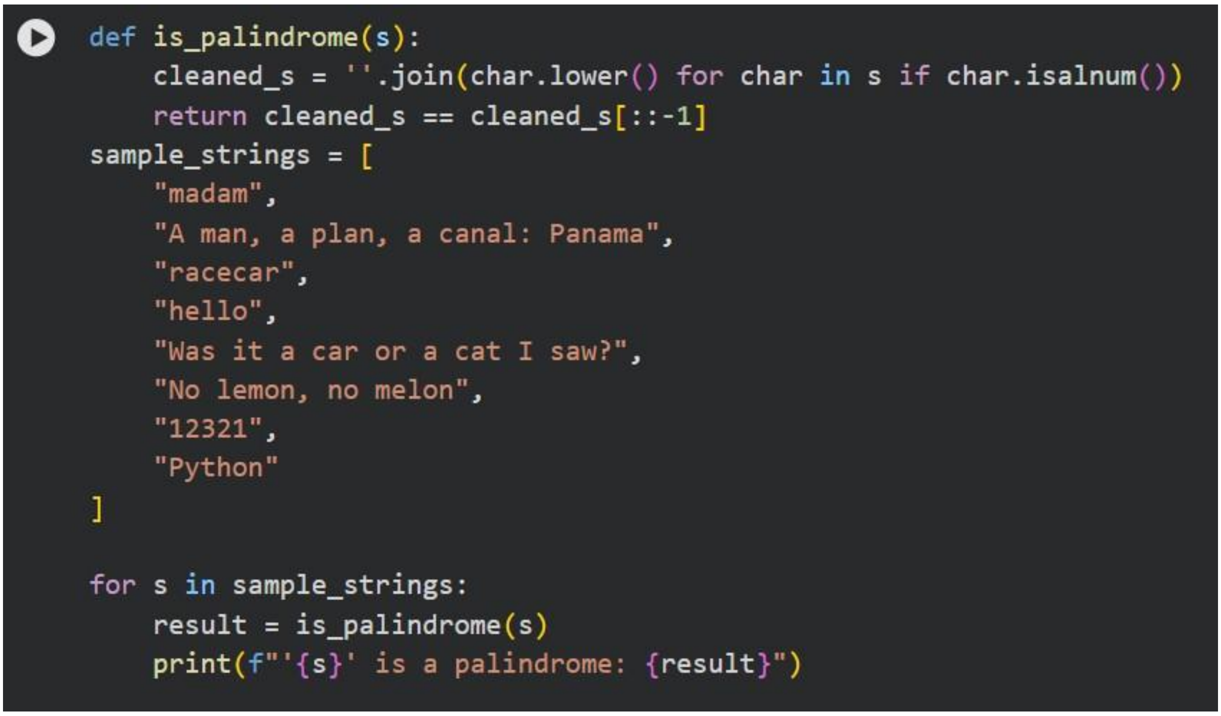
How the Function Works:

1. **Initialization**: It defines a set of `vowels` (lowercase) and initializes `vowel_count`, `consonant_count`, and `digit_count` to zero.
2. **Iteration**: It iterates through each character in the `input_string`.
3. **Digit Check**: For each character, it first checks if it's a digit using `char.isdigit()`. If true, `digit_count` is incremented.
4. **Alphabetic Check**: If the character is not a digit, it then checks if it's an alphabetic character using `char.isalpha()`.
5. **Vowel/Consonant Check**: If it's an alphabet, the character is converted to lowercase (`char.lower()`) to perform a case-insensitive check. If the lowercase character is in the `vowels` set, `vowel_count` is incremented; otherwise, `consonant_count` is incremented.
6. **Return Value**: Finally, it returns a dictionary containing the total counts for vowels, consonants, and digits.

Task 3: Palindrome Check.

Prompt (Gemini Colab): Write a Python function to check whether a given string is a palindrome. Keep the code simple and easy to understand.

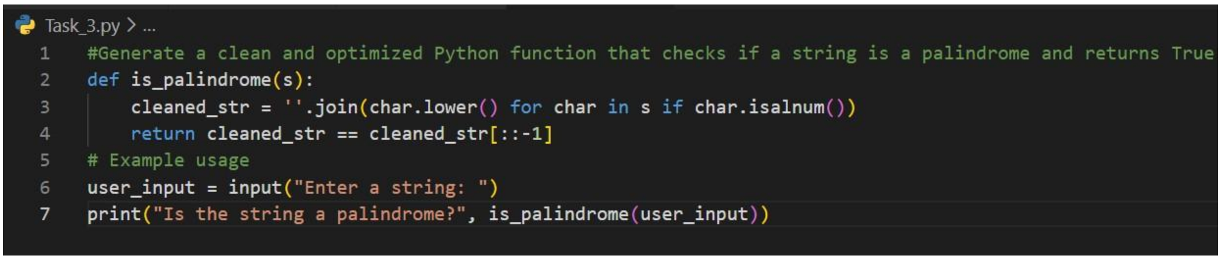
Code:



```python
def is_palindrome(s):
    cleaned_s = ''.join(char.lower() for char in s if char.isalnum())
    return cleaned_s == cleaned_s[::-1]
sample_strings = [
    "madam",
    "A man, a plan, a canal: Panama",
    "racecar",
    "hello",
    "Was it a car or a cat I saw?",
    "No lemon, no melon",
    "12321",
    "Python"
]

for s in sample_strings:
    result = is_palindrome(s)
    print(f"'{s}' is a palindrome: {result}")
```

Prompt (Copilot): Generate a clean and optimized Python function that checks if a string is a palindrome and returns True or False.

Code:



Task_3.py > ...

```python
#Generate a clean and optimized Python function that checks if a string is a palindrome and returns True
def is_palindrome(s):
    cleaned_str = ''.join(char.lower() for char in s if char.isalnum())
    return cleaned_str == cleaned_str[::-1]
# Example usage
user_input = input("Enter a string: ")
print("Is the string a palindrome?", is_palindrome(user_input))
```

Comparison: (Gemini vs GitHub Copilot)

Feature	Gemini	GitHub Copilot
Clarity	Very clear and beginner-friendly. Uses simple logic and explains steps well.	Clear logic but provides little explanation; assumes basic coding knowledge.
Structure	Step-by-step approach focusing on understanding before optimization.	Compact and efficient structure following professional coding standards.
Feature	**Gemini**	**GitHub Copilot**

Readability	Highly readable for students and beginners.	Readable for experienced programmers but slightly complex for beginners.
Explanation Quality	Provides detailed and easy-to-understand explanations.	Minimal explanation; focuses mainly on code generation.
Overall Strength	Best for learning and concept understanding.	Best for real-world coding and optimized solutions.

Task 4: Code Explanation Using AI.

Prompt: Explain the following Python function line by line in simple language so that a beginner can understand it clearly.(**prime check OR palindrome function)**

Line	**Explanation**
def is_palindrome(word):	This line creates a function named is_palindrome that takes one input called word.
word = word.lower()	Converts the word to lowercase so that capital and small letters are treated the same.
reversed_word = word[::-1]	Reverses the string using slicing. Example: "madam" becomes "madam", "hello" becomes "olleh".
if word == reversed_word:	Checks whether the original word and the reversed word are the same.
return True	If they are the same, the function returns True meaning it **is a palindrome**.
else:	Runs when the condition is false.
return False	Returns False meaning it **is not a palindrome**

This explanation helped me understand how strings can be reversed and compared in Python. The step-by-step logic made it easy to learn how palindrome checking works.

THANK YOU !!