SR UNIVERSITY

School of Computer Science and Artificial Intelligence

# Lab Assignment - 2

**Name of Student** : **T. Sai Teja**
**Enrollment No.** : **2303A51535**
**Batch No.** : **22**

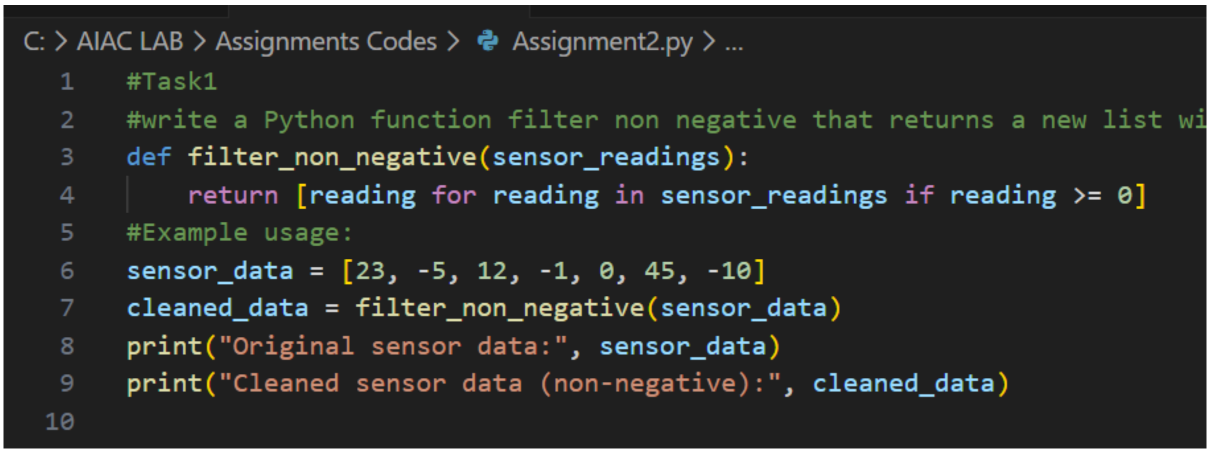
## Task 1: Cleaning Sensor Data

You are cleaning IoT sensor data where negative values are invalid.

**Prompt-**

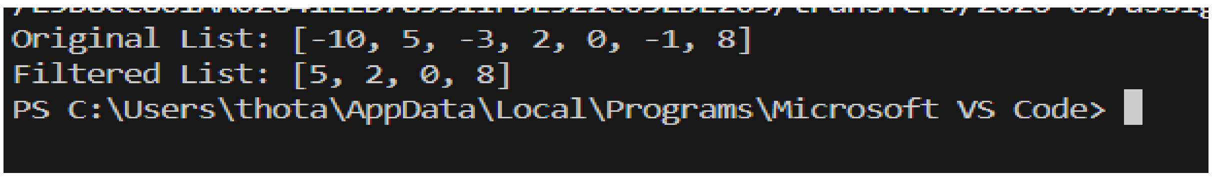
Write a Python function suitable for IoT data preprocessing that accepts a list of sensor readings and returns a new list containing only valid values (zero or positive numbers). The function should preserve the original list and demonstrate its working using a sample input and output.

**Code-**



```
C: > AIAC LAB > Assignments Codes > Assignment2.py > ...
#Task1
#write a Python function filter non negative that returns a new list wi
def filter_non_negative(sensor_readings):
    return [reading for reading in sensor_readings if reading >= 0]
#Example usage:
sensor_data = [23, -5, 12, -1, 0, 45, -10]
cleaned_data = filter_non_negative(sensor_data)
print("Original sensor data:", sensor_data)
print("Cleaned sensor data (non-negative):", cleaned_data)
```

**Output-**



```
Original List: [-10, 5, -3, 2, 0, -1, 8]
Filtered List: [5, 2, 0, 8]
PS C:\Users\thota\AppData\Local\Programs\Microsoft VS Code>
```

**Justification-**

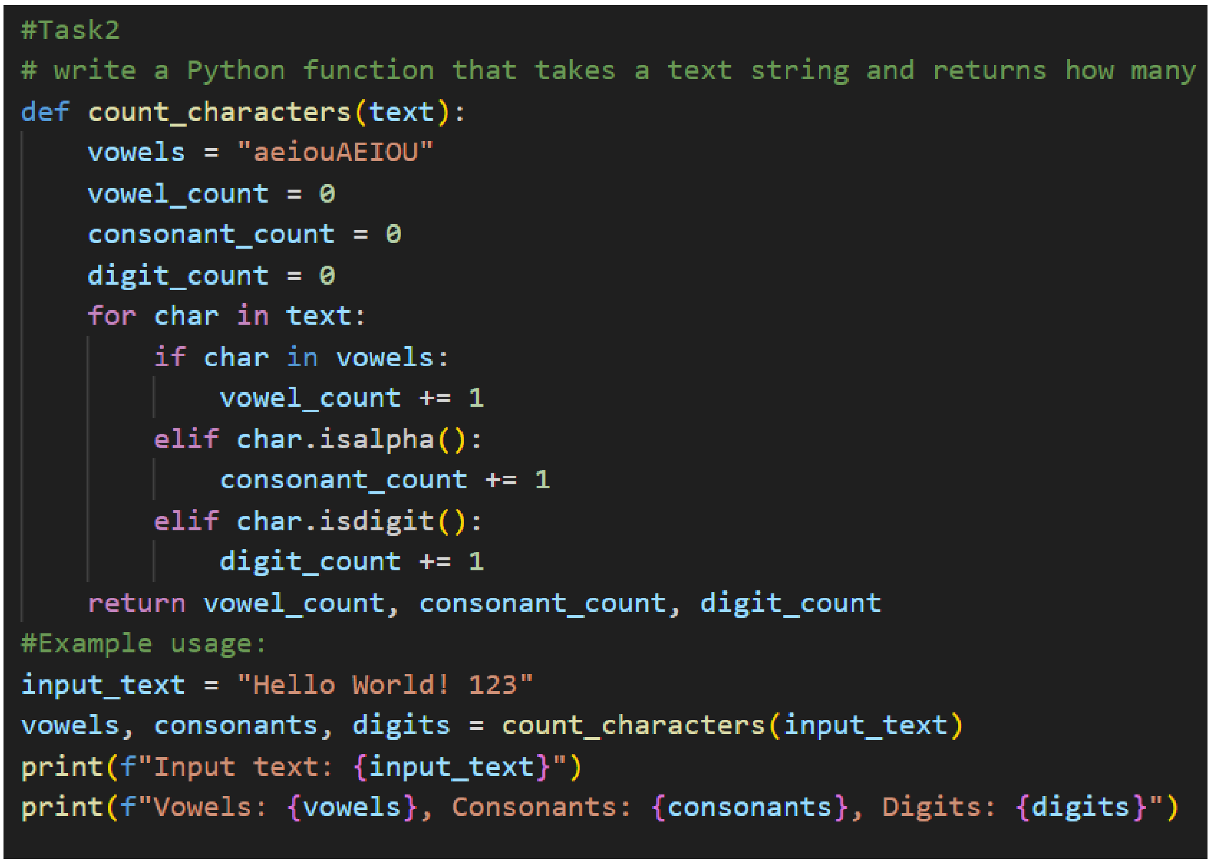
This function uses a list comprehension to create a new list that includes only the non-negative values from the original list. It does not modify the original list, ensuring that the raw sensor data remains intact for any further analysis or processing.

## Task 2: String Character Analysis

You are building a text-analysis feature.

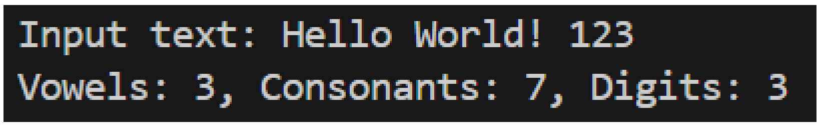
**Prompt-**
Design a Python function that processes an input string and counts the number of vowels, consonants, and digits present. The function should ignore case sensitivity and return the results clearly. Include an example input and its corresponding output.

**Code-**



```python
#Task2
# write a Python function that takes a text string and returns how many
def count_characters(text):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0
    digit_count = 0
    for char in text:
        if char in vowels:
            vowel_count += 1
        elif char.isalpha():
            consonant_count += 1
        elif char.isdigit():
            digit_count += 1
    return vowel_count, consonant_count, digit_count
#Example usage:
input_text = "Hello World! 123"
vowels, consonants, digits = count_characters(input_text)
print(f"Input text: {input_text}")
print(f"Vowels: {vowels}, Consonants: {consonants}, Digits: {digits}")
```

**Output-**



```
Input text: Hello World! 123
Vowels: 3, Consonants: 7, Digits: 3
```

**Justification-**This function iterates through each character in the input string, checking if it is a vowel, consonant, or digit, and increments the respective counters accordingly. It treats upper and lower case letters the same by including both in the vowels string and using isalpha() for consonants.

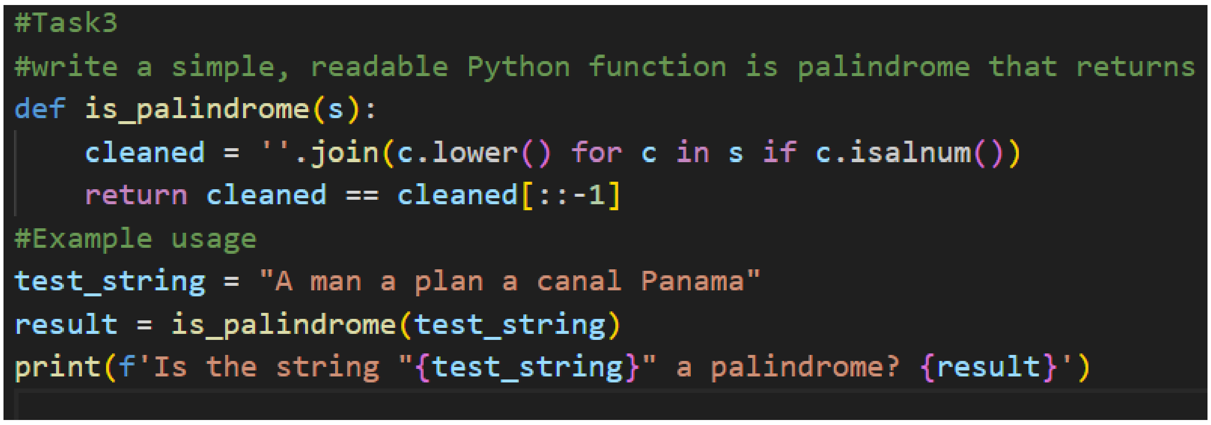
## Task 3: Palindrome Check – Tool Comparison

You must decide which AI tool is clearer for string logic.

**Prompt-**

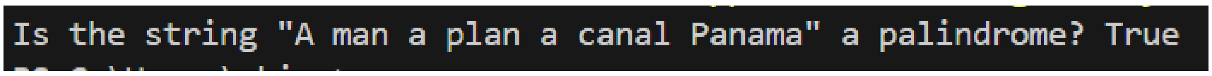
Create a beginner-friendly Python function that checks whether a given string is a palindrome by ignoring spaces, punctuation, and letter case. Implement this solution using both Gemini and GitHub Copilot, then compare the generated code based on clarity, readability, and structure.

**Code-**



```python
#Task3
#write a simple, readable Python function is palindrome that returns
def is_palindrome(s):
    cleaned = ''.join(c.lower() for c in s if c.isalnum())
    return cleaned == cleaned[::-1]
#Example usage
test_string = "A man a plan a canal Panama"
result = is_palindrome(test_string)
print(f'Is the string "{test_string}" a palindrome? {result}')
```

**Output-**



```
Is the string "A man a plan a canal Panama" a palindrome? True
```

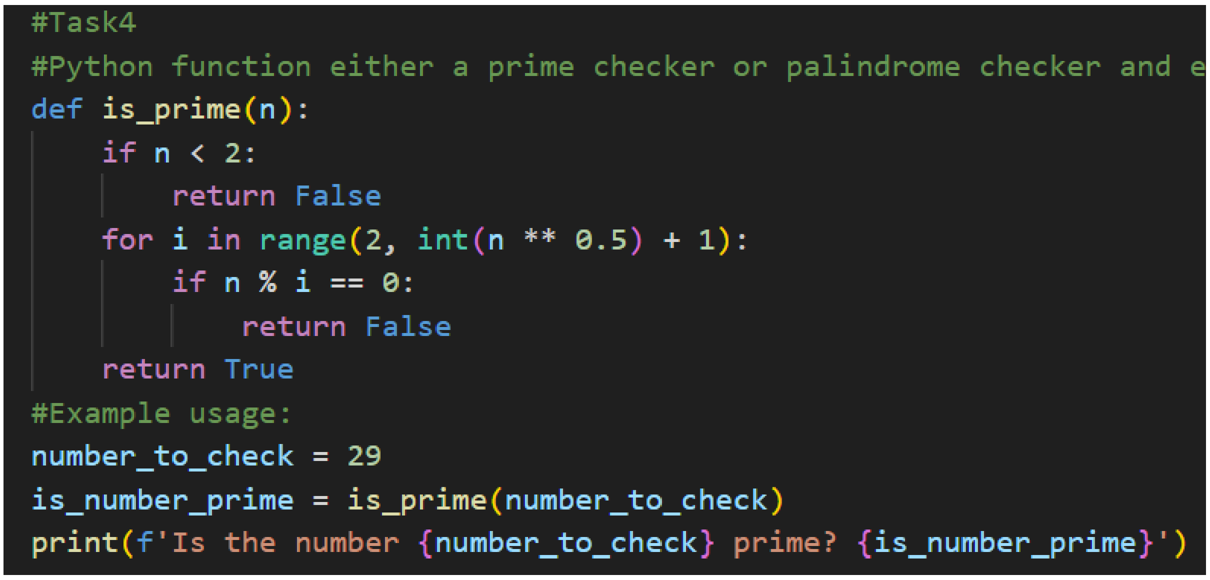
**Justification-**

This function is simple and readable because it breaks down the problem into clear steps: cleaning the string by removing non-alphanumeric characters and converting to lowercase, then checking if the cleaned string is equal to its reverse. This makes it easy for beginners to understand how palindromes work.

**Task 4: Code Explanation Using AI**
You are reviewing unfamiliar code written by another developer.

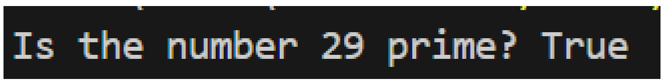
**Prompt-**
Provide a Python function that checks whether a number is prime (or alternatively, whether a string is a palindrome). Explain each line of the code in simple, step-by-step language suitable for a first-year computer science student.

**Code-**



```python
#Task4
#Python function either a prime checker or palindrome checker and e
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
#Example usage:
number_to_check = 29
is_number_prime = is_prime(number_to_check)
print(f'Is the number {number_to_check} prime? {is_number_prime}')
```

**Output-**



```
Is the number 29 prime? True
```

**Justification-**

The function checks if a number is prime by first handling numbers less than 2 (which are not prime). Then it iterates from 2 up to the square root of the number, checking for divisibility. If any divisor is found, it returns False; otherwise, it returns True.