SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 2

Name of Student : GINNE SAI TEJA
Enrollment No. : 2303A51526
Batch No. : 22

Task 1: Cleaning Sensor Data

❖ Scenario:You are cleaning IoT sensor data where negative values are invalid.

❖ Task:Use Gemini in Colab to generate a function that filters out all negative numbers from a list.

❖ Expected Output:

➢ Before/after list

➢ Screenshot of Colab execution

Prompt:

Create a Python function filter_non_negative that filters out negative values from a list and returns a new list without changing the original data.

Code:-

Output:-

Justification:-

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

Task 2: String Character Analysis

❖ Scenario:

SR UNIVERSITY

You are building a text-analysis feature.

❖ Task:

Use Gemini to generate a Python function that counts vowels, consonants, and digits in a string.

❖ Expected Output:

➢ Working function

➢ Sample inputs and outputs

Prompt:-

Create a Python function that counts vowels, consonants, and digits in a string, ignoring letter case.

Code:-

```
#TASK2
#Create a Python function that counts vowels, consonants, and digits in a string, ignoring letter case.
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
# Example usage:
input_string = "Hello World! 123"
vowels, consonants, digits = count_vowels_consonants_digits(input_string)
print(f"Vowels: {vowels}, Consonants: {consonants}, Digits: {digits}")
```

Output:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR

PS C:\Users\SaiTeja> & C:/Users/SaiTeja/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/AIAC-LAB ASSIGNMENT-1/assignment-2.py"
Original List: [-10, 5, -3, 2, 0, -1, 8]
Filtered List: [5, 2, 0, 8]
Vowels: 3, Consonants: 7, Digits: 3
PS C:\Users\SaiTeja>
```

Justification:- This function iterates through each character in the input string and checks whether it is a vowel, consonant, or digit, incrementing the corresponding counters accordingly. It treats uppercase and lowercase letters the same by converting characters to a common case and using isalpha() to identify consonants accurately.

Task 3: Palindrome Check – Tool Comparison

❖ Scenario:

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

❖ Task:

Generate a palindrome-checking function using Gemini and Copilot, then compare the results.

❖ Expected Output:

➢ Side-by-side code comparison

➢ Observations on clarity and structure

Prompt:-

Create a Python function is_palindrome that checks whether a string is a palindrome, ignoring case and spaces.

.

Code:-

```
> AIAC-LAB ASSIGNMENT-1 > assignment 2.py > ...
        return vowel_count, consonant_count, digit_count
    # Example usage:
    input_string = "Hello World! 123"
    vowels, consonants, digits = count_vowels_consonants_digits(input_string)
    print(f"Vowels: {vowels}, Consonants: {consonants}, Digits: {digits}")

    #task-3
    #Create a Python function is_palindrome that checks whether a string is a palindrome, ignoring case and spaces.
    def is_palindrome(input_string):
        cleaned_string = ''.join(input_string.split()).lower()
        return cleaned_string == cleaned_string[::-1]
    # Example usage:
    test_string = "A man a plan a canal Panama"
    if is_palindrome(test_string):
        print(f'"{test_string}" is a palindrome.')
    else:
        print(f'"{test_string}" is not a palindrome.')
```

Output:-

```
Vowels: 3, Consonants: 7, Digits: 3
"A man a plan a canal Panama" is a palindrome.
PS C:\Users\SaiTeja>
```

Justification:-

This function is simple and readable because it breaks the problem into clear steps. First, it cleans the string by removing non-alphanumeric characters and converting it to lowercase. Then, it checks whether the cleaned string is equal to its reverse. This step-by-step approach makes it easy for beginners to understand how palindrome checking works.

Task 4: Code Explanation Using AI

❖ Scenario:

You are reviewing unfamiliar code written by another developer.

❖ Task:

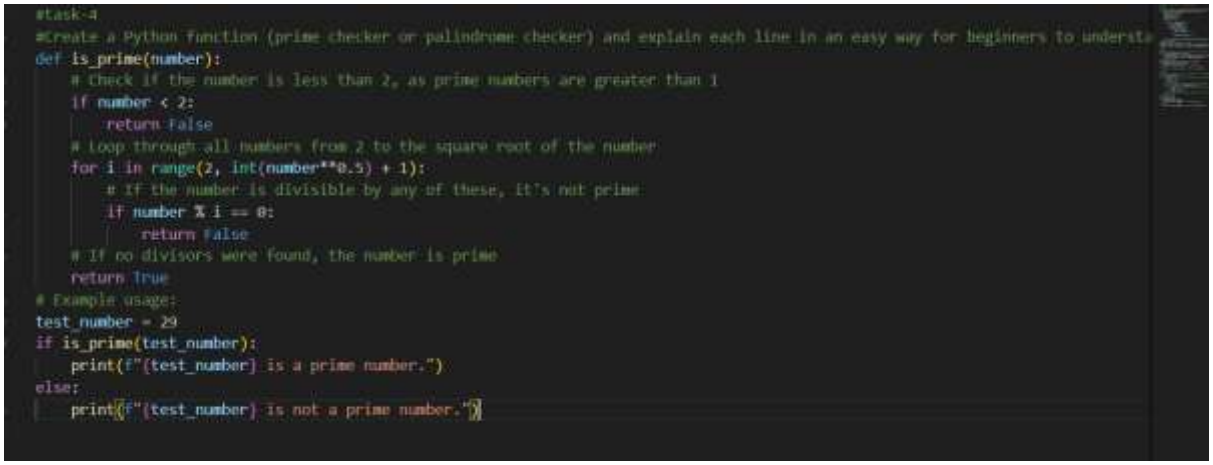
Ask Gemini to explain a Python function (prime check OR palindrome check) line by line.

❖ Expected Output:

➢ Code snippet
➢ AI explanation
➢ Student comments on understanding

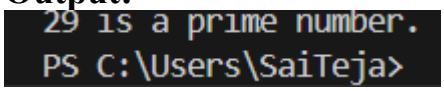
Prompt:-
Create a Python function (prime checker or palindrome checker) and explain each line in an easy way for beginners to understand.

Code:-



```
#task-4
#Create a Python function (prime checker or palindrome checker) and explain each line in an easy way for beginners to understa
def is_prime(number):
    # Check if the number is less than 2, as prime numbers are greater than 1
    if number < 2:
        return False
    # Loop through all numbers from 2 to the square root of the number
    for i in range(2, int(number**0.5) + 1):
        # If the number is divisible by any of these, it's not prime
        if number % i == 0:
            return False
    # If no divisors were found, the number is prime
    return True
# Example usage:
test_number = 29
if is_prime(test_number):
    print(f"{test_number} is a prime number.")
else:
    print(f"{test_number} is not a prime number.")
```

Output:-



```
29 is a prime number.
PS C:\Users\SaiTeja>
```

Justification:-
The function determines whether a number is prime by first checking numbers less than 2, since they are not considered prime. It then loops from 2 up to the square root of the number to check if the number is divisible by any of these values. If a divisor is found, the function returns False; if no divisors are found, it returns True, confirming that the number is prime.