AI ASSISTED CODING

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

NASANI SAI PAVAN **2303A51117**

BATCH – 03 **13 – 01 – 2026**

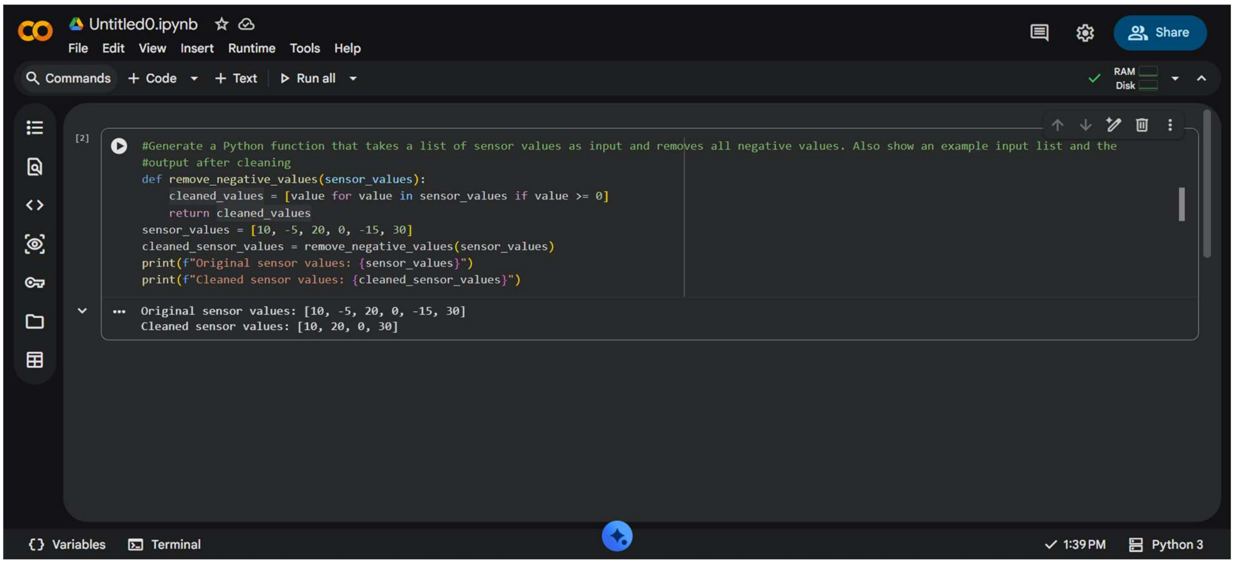
ASSIGNMENT – 2.2

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:



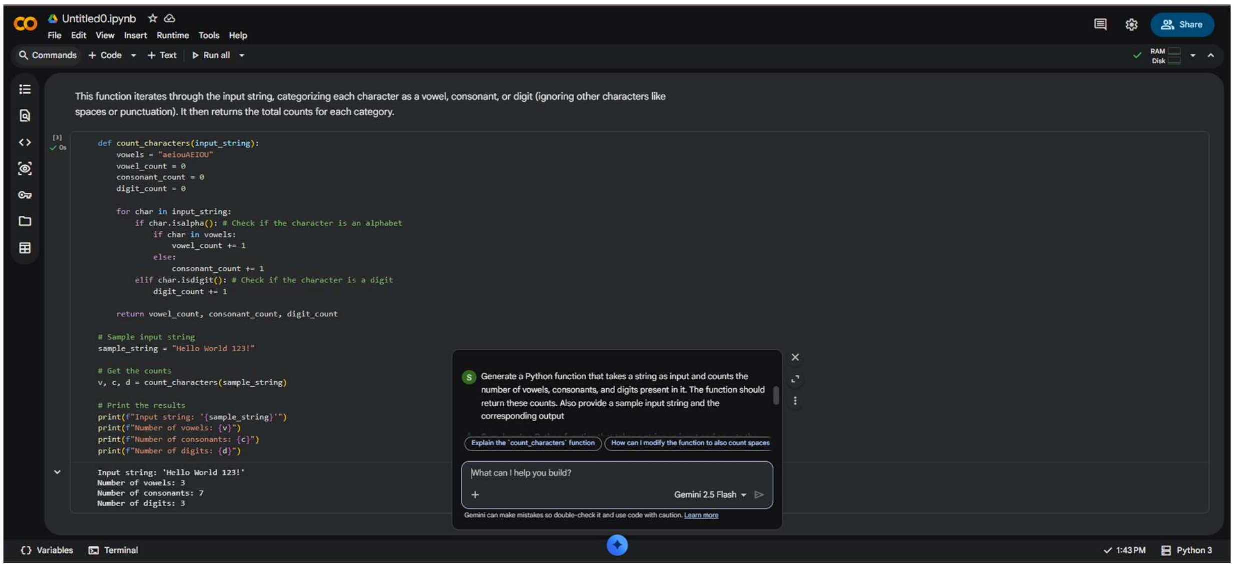
```
#Generate a Python function that takes a list of sensor values as input and removes all negative values. Also show an example input list and the
#output after cleaning
def remove_negative_values(sensor_values):
    cleaned_values = [value for value in sensor_values if value >= 0]
    return cleaned_values
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:


Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM
Disk
[3]
0s
Variables
Terminal
1:43 PM
Python 3

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}")
```

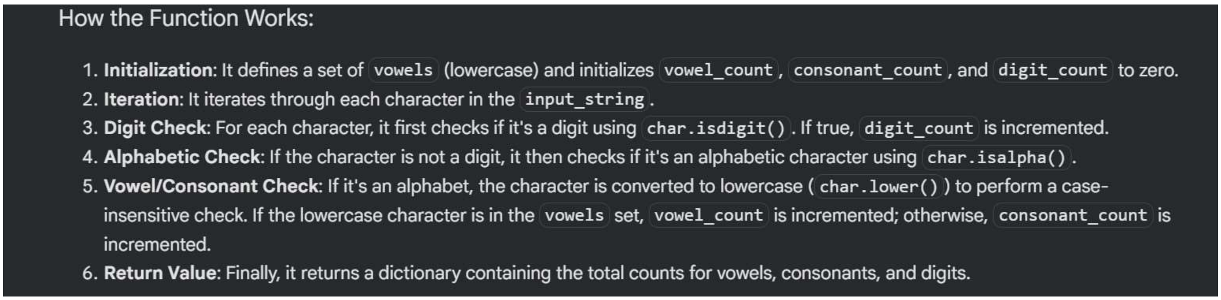
```
Input string: 'Hello World 123!'
Number of vowels: 3
Number of consonants: 7
Number of digits: 3
```

Generate a Python function that takes a string as input and counts the number of vowels, consonants, and digits present in it. The function should return these counts. Also provide a sample input string and the corresponding output

Explain the `count_characters` function
How can I modify the function to also count spaces
What can I help you build?
Gemini 2.5 Flash

Gemini can make mistakes so double-check it and use code with caution. Learn more

Working Function:



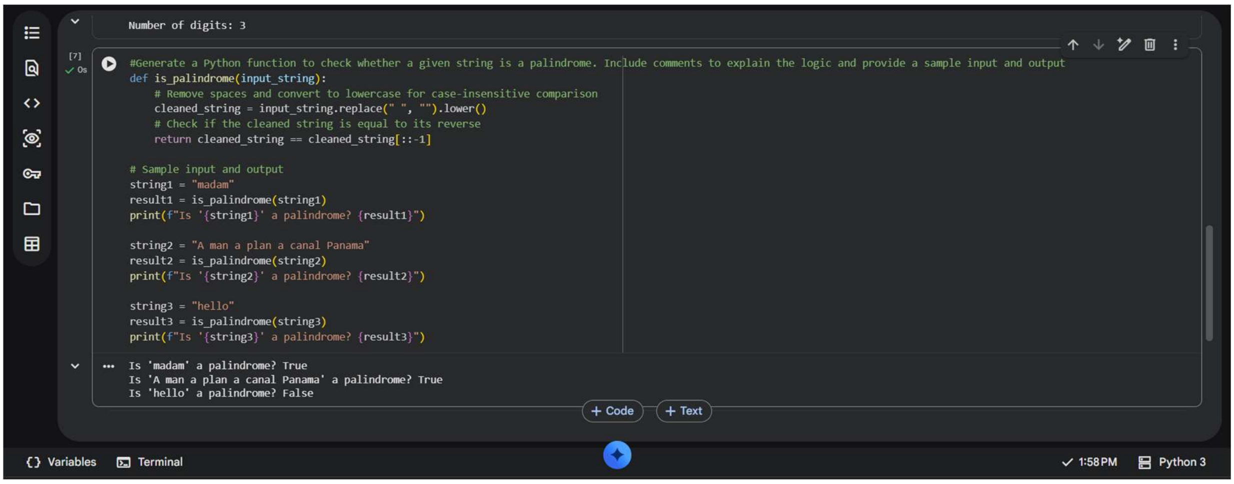
How the Function Works:

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:



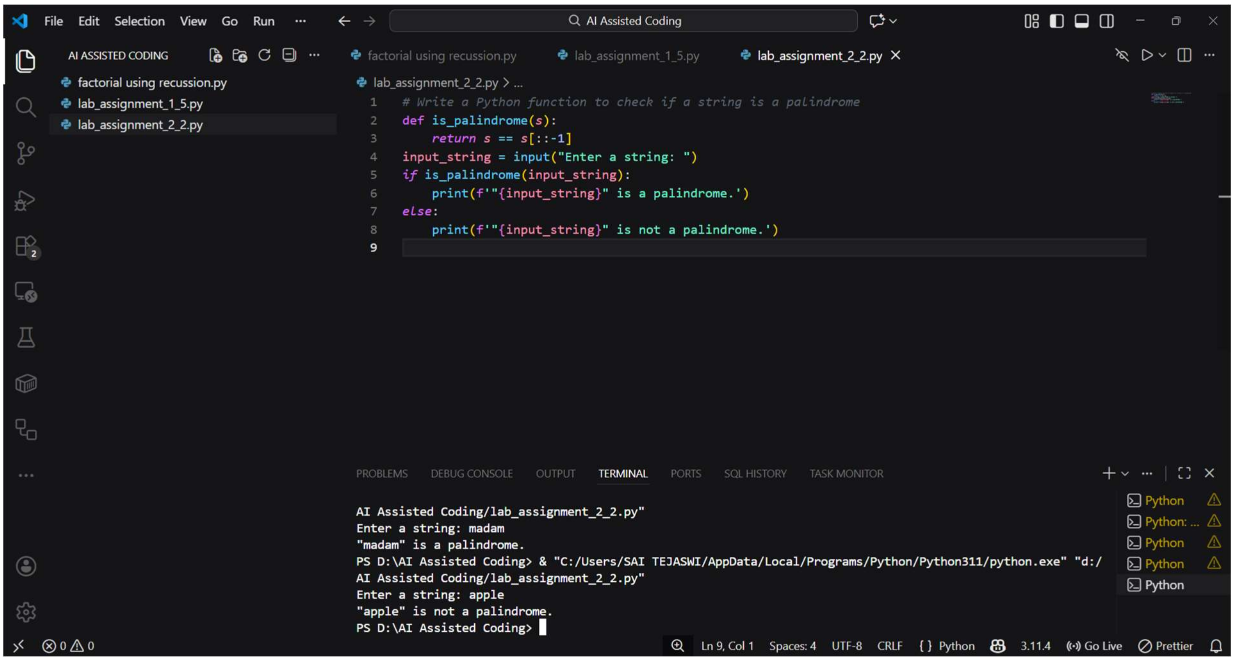
```
Number of digits: 3
```

```python
#Generate a Python function to check whether a given string is a palindrome. Include comments to explain the logic and provide a sample input and output
def is_palindrome(input_string):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    cleaned_string = input_string.replace(" ", "").lower()
    # Check if the cleaned string is equal to its reverse
    return cleaned_string == cleaned_string[::-1]

# Sample input and output
string1 = "madam"
result1 = is_palindrome(string1)
print(f"Is '{string1}' a palindrome? {result1}")

string2 = "A man a plan a canal Panama"
result2 = is_palindrome(string2)
print(f"Is '{string2}' a palindrome? {result2}")

string3 = "hello"
result3 = is_palindrome(string3)
print(f"Is '{string3}' a palindrome? {result3}")
```

```
Is 'madam' a palindrome? True
Is 'A man a plan a canal Panama' a palindrome? True
Is 'hello' a palindrome? False
```

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

Code:



```python
# Write a Python function to check if a string is a palindrome
def is_palindrome(s):
    return s == s[::-1]
input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print(f'"{input_string}" is a palindrome.')
else:
    print(f'"{input_string}" is not a palindrome.')
```

```
AI Assisted Coding/lab_assignment_2_2.py"
Enter a string: madam
"madam" is a palindrome.
PS D:\AI Assisted Coding> & "C:/Users/SAI TEJASWI/AppData/Local/Programs/Python/Python311/python.exe" "d:/
AI Assisted Coding/lab_assignment_2_2.py"
Enter a string: apple
"apple" is not a palindrome.
PS D:\AI Assisted Coding> █
```

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)**

Code:

```
def is_prime(n):
  if n <= 1:
    return False
  for i in range(2, int(n**0.5) + 1):
    if n % i == 0:
      return False
  return True
```

Explanation:

Certainly! Here's a line-by-line explanation of the is_prime function:

```
def is_prime(n):
```

This line defines a Python function named is_prime that accepts one input, n. The purpose of this function is to determine whether the number n is a prime number.

if n <= 1:

Prime numbers are defined as natural numbers greater than 1. So, if the input number n is 1 or less (e.g., 1, 0, or negative numbers), it cannot be prime.
This line checks for this initial condition.
return False

If the condition n <= 1 is true, the function immediately stops and returns False, indicating that n is not a prime number.
for i in range(2, int(n**0.5) + 1):

This line starts a for loop to check for potential divisors of n.
range(2, ...): We start checking for divisors from 2 because 1 divides every number, and we've already handled numbers less than or equal to 1.
int(n**0.5) + 1: This is an optimization. To check if a number n is prime, you only need to test for divisibility by numbers up to its square root. If n has a divisor larger than its square root, it must also have a divisor smaller than its square root. n**0.5 calculates the square root, int(...) converts it to an integer, and + 1 ensures the loop includes the integer part of the square root itself.
if n % i == 0:

Inside the loop, this line checks if n is perfectly divisible by the current number i.
The modulo operator (%) returns the remainder of the division. If the remainder is 0, it means i is a divisor of n.
return False

If n is found to be divisible by any i in the loop (meaning it has a divisor other than 1 and itself), it cannot be a prime number.
The function immediately returns False and stops.
return True

If the loop finishes without finding any divisors for n (i.e., n was not perfectly divisible by any i from 2 up to its square root), it means n is a prime number.
In this case, the function returns True.