LAB ASSIGNMENT - 2.2

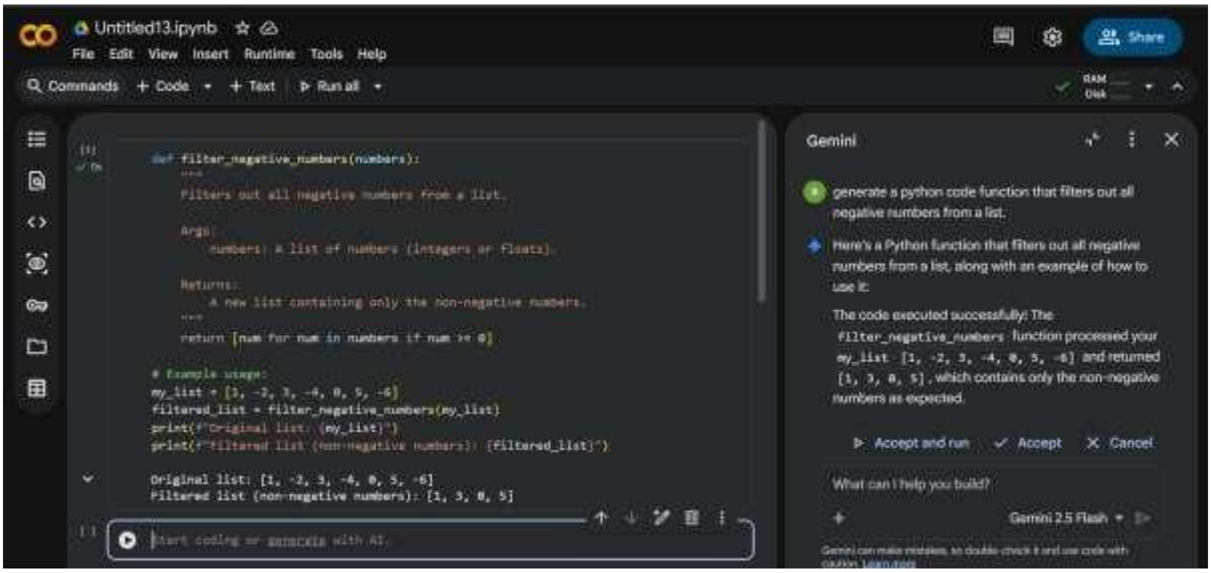
NAME : Ramya Sri

2303A510I9

BATCH 30

TASK-1:

Prompt : Generate a python code for function that filters out all negative numbers from a list



```python
def filter_negative_numbers(numbers):
    """
    Filters out all negative numbers from a list.

    Args:
        numbers: A list of numbers (integers or floats).

    Returns:
        A new list containing only the non-negative numbers.
    """
    return [num for num in numbers if num >= 0]

# Example usage:
my_list = [1, -2, 3, -4, 0, 5, -6]
filtered_list = filter_negative_numbers(my_list)
print(f"Original list: {my_list}")
print(f"Filtered list (non-negative numbers): {filtered_list}")
```

```
Original list: [1, -2, 3, -4, 0, 5, -6]
Filtered list (non-negative numbers): [1, 3, 0, 5]
```

generate a python code function that filters out all negative numbers from a list.

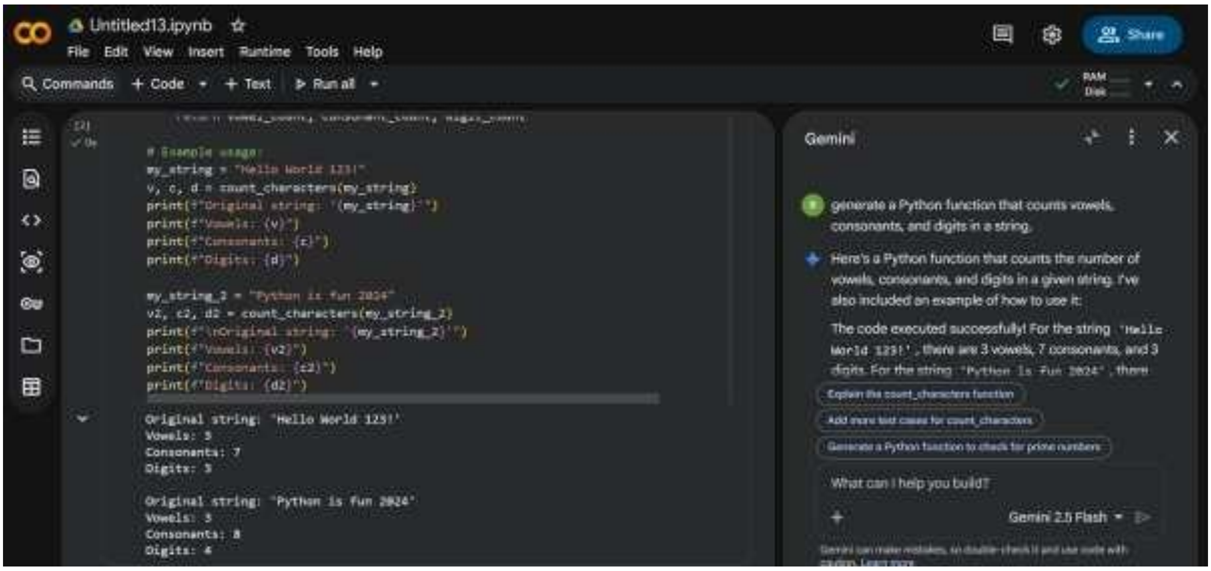
Here's a Python function that filters out all negative numbers from a list, along with an example of how to use it:

The code executed successfully! The `filter_negative_numbers` function processed your `my_list` `[1, -2, 3, -4, 0, 5, -6]` and returned `[1, 3, 0, 5]`, which contains only the non-negative numbers as expected.

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

TASK-2:

Prompt : generate a Python function that counts vowels, consonants, and digits in a string.


Untitled13.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
RAM
Disk
Gemini
Explain the count_characters function
Add more test cases for count_characters
Generate a Python function to check for prime numbers
What can I help you build?
Gemini 2.5 Flash

```python
# Example usage:
my_string = "Hello World 123!"
v, c, d = count_characters(my_string)
print(f"Original string: '{my_string}'")
print(f"Vowels: {v}")
print(f"Consonants: {c}")
print(f"Digits: {d}")

my_string_2 = "Python is fun 2024"
v2, c2, d2 = count_characters(my_string_2)
print(f"\nOriginal string: '{my_string_2}'")
print(f"Vowels: {v2}")
print(f"Consonants: {c2}")
print(f"Digits: {d2}")
```

```
Original string: 'Hello World 123!'
Vowels: 3
Consonants: 7
Digits: 3

Original string: 'Python is fun 2024'
Vowels: 3
Consonants: 8
Digits: 4
```

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

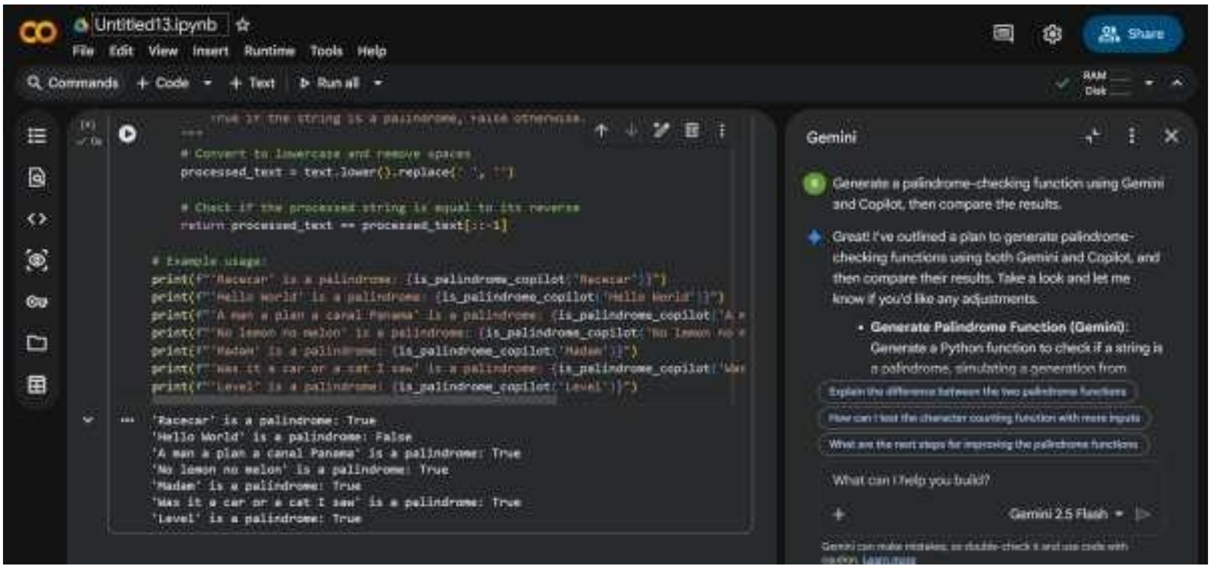
Here's a Python function that counts the number of vowels, consonants, and digits in a given string. I've also included an example of how to use it:

The code executed successfully! For the string `'Hello World 123!'`, there are 3 vowels, 7 consonants, and 3 digits. For the string `'Python is fun 2024'`, there

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

TASK-3:

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



```
        True if the string is a palindrome, False otherwise.
    """
    # Convert to lowercase and remove spaces
    processed_text = text.lower().replace(' ', '')

    # Check if the processed string is equal to its reverse
    return processed_text == processed_text[::-1]

# Example usage:
print(f"'Racecar' is a palindrome: {is_palindrome_copilot('Racecar')}")
print(f"'Hello World' is a palindrome: {is_palindrome_copilot('Hello World')}")
print(f"'A man a plan a canal Panama' is a palindrome: {is_palindrome_copilot('A
print(f"'No lemon no melon' is a palindrome: {is_palindrome_copilot('No lemon no
print(f"'Madam' is a palindrome: {is_palindrome_copilot('Madam')}")
print(f"'Was it a car or a cat I saw' is a palindrome: {is_palindrome_copilot('Was
print(f"'Level' is a palindrome: {is_palindrome_copilot('Level')}")
```

```
'Racecar' is a palindrome: True
'Hello World' is a palindrome: False
'A man a plan a canal Panama' is a palindrome: True
'No lemon no melon' is a palindrome: True
'Madam' is a palindrome: True
'Was it a car or a cat I saw' is a palindrome: True
'Level' is a palindrome: True
```

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

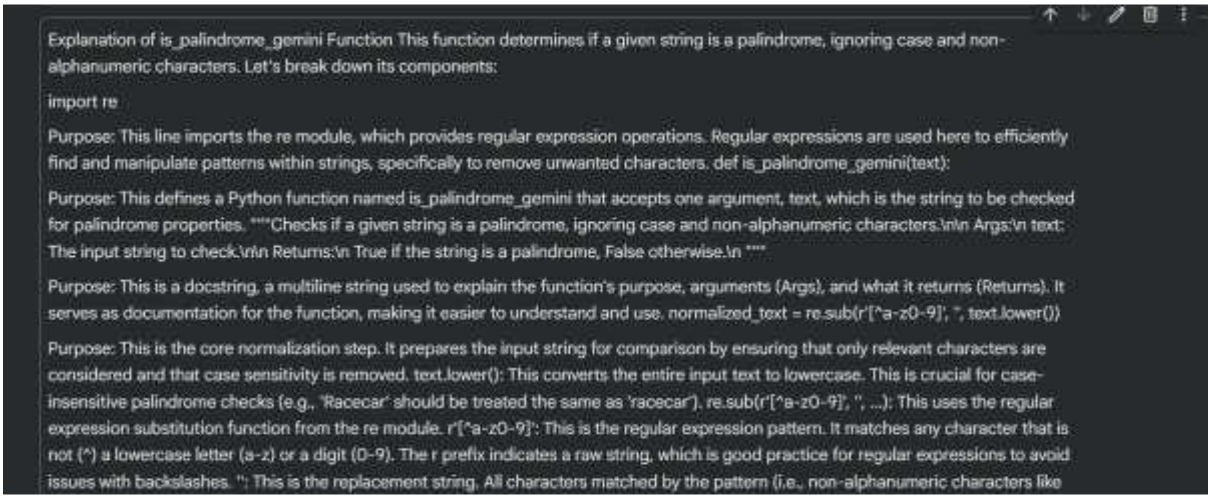
Great! I've outlined a plan to generate palindrome-checking functions using both Gemini and Copilot, and then compare their results. Take a look and let me know if you'd like any adjustments.

- **Generate Palindrome Function (Gemini):** Generate a Python function to check if a string is a palindrome, simulating a generation from

Explain the difference between the two palindrome functions
How can I test the character counting function with more inputs
What are the next steps for improving the palindrome functions
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

TASK-4:

Prompt: explain a Python function (prime check OR palindrome check) line by line.



Explanation of is_palindrome_gemini Function This function determines if a given string is a palindrome, ignoring case and non-alphanumeric characters. Let's break down its components:

import re

Purpose: This line imports the re module, which provides regular expression operations. Regular expressions are used here to efficiently find and manipulate patterns within strings, specifically to remove unwanted characters. def is_palindrome_gemini(text):

Purpose: This defines a Python function named is_palindrome_gemini that accepts one argument, text, which is the string to be checked for palindrome properties. """Checks if a given string is a palindrome, ignoring case and non-alphanumeric characters.\n\n Args:\n text: The input string to check.\n\n Returns:\n True if the string is a palindrome, False otherwise.\n """

Purpose: This is a docstring, a multiline string used to explain the function's purpose, arguments (Args), and what it returns (Returns). It serves as documentation for the function, making it easier to understand and use. normalized_text = re.sub(r'[^a-z0-9]', '', text.lower())

Purpose: This is the core normalization step. It prepares the input string for comparison by ensuring that only relevant characters are considered and that case sensitivity is removed. text.lower(): This converts the entire input text to lowercase. This is crucial for case-insensitive palindrome checks (e.g., 'Racecar' should be treated the same as 'racecar'). re.sub(r'[^a-z0-9]', '', ...): This uses the regular expression substitution function from the re module. r'[^a-z0-9]': This is the regular expression pattern. It matches any character that is not (^) a lowercase letter (a-z) or a digit (0-9). The r prefix indicates a raw string, which is good practice for regular expressions to avoid issues with backslashes. '': This is the replacement string. All characters matched by the pattern (i.e., non-alphanumeric characters like