NAME :- KAUSHAL KUMAR.

HALLTICKET NO :-2403A51317

BATCH NO :- 13

DATE :- 20.08.2025

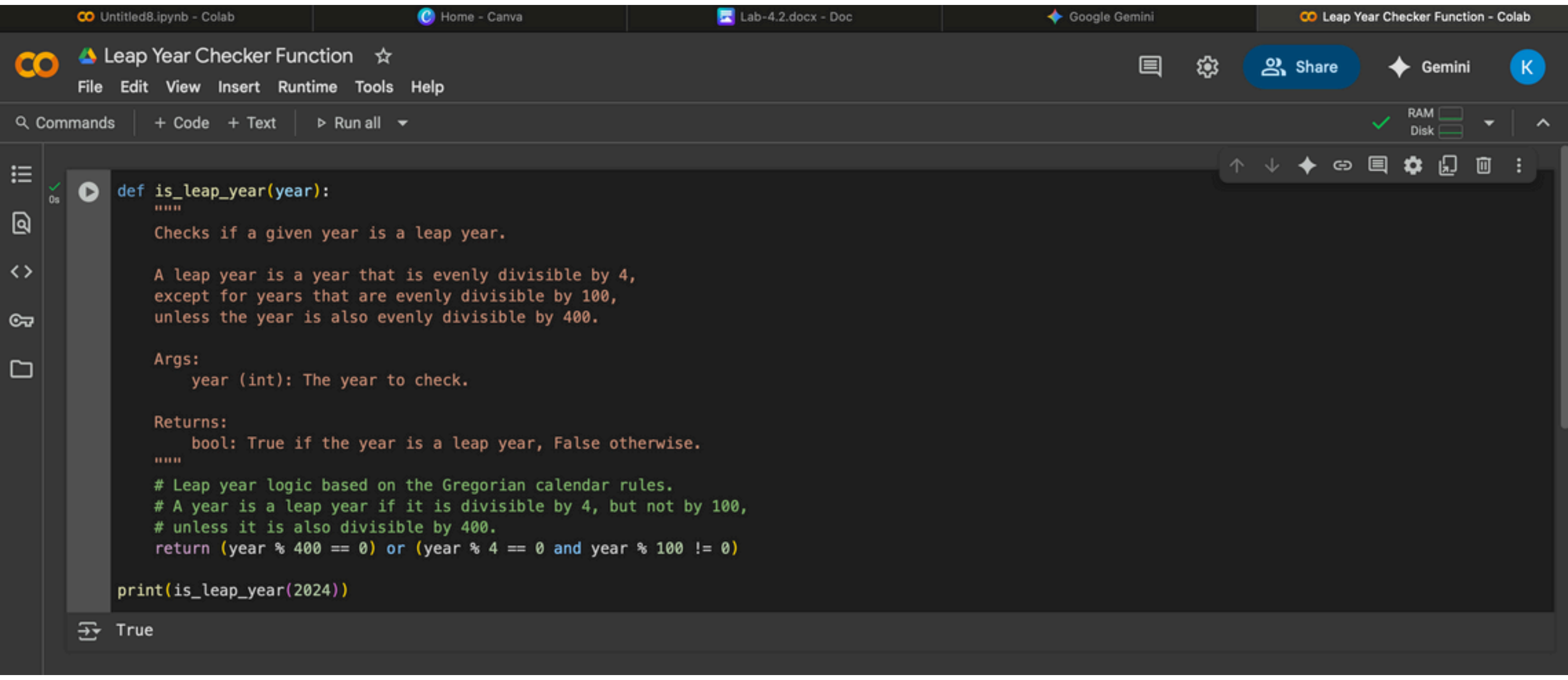
ASSIGNMENT :-4.3

TASK 1 :-

Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

Expected Output#1

● AI-generated function with no examples provided


Untitled8.ipynb - Colab
Home - Canva
Lab-4.2.docx - Doc
Google Gemini
Leap Year Checker Function - Colab
Leap Year Checker Function
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM Disk

```
def is_leap_year(year):
    """
    Checks if a given year is a leap year.

    A leap year is a year that is evenly divisible by 4,
    except for years that are evenly divisible by 100,
    unless the year is also evenly divisible by 400.

    Args:
        year (int): The year to check.

    Returns:
        bool: True if the year is a leap year, False otherwise.
    """
    # Leap year logic based on the Gregorian calendar rules.
    # A year is a leap year if it is divisible by 4, but not by 100,
    # unless it is also divisible by 400.
    return (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0)

print(is_leap_year(2024))
```

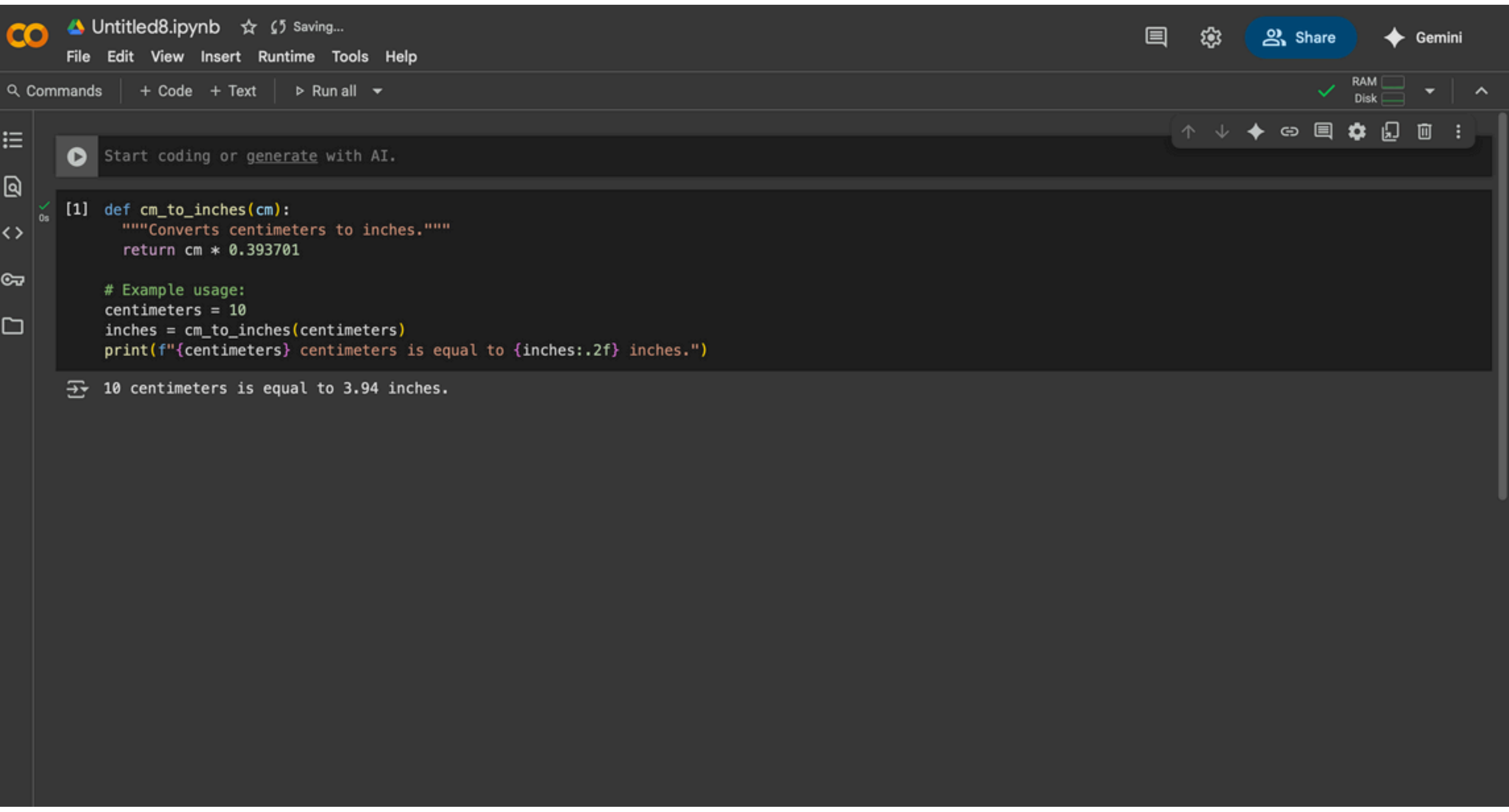
```
True
```

TASK 2:-

One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

Expected Output#2

● Function with correct conversion logic


Untitled8.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
RAM
Disk
Start coding or generate with AI.

```
[1] def cm_to_inches(cm):
      """Converts centimeters to inches."""
      return cm * 0.393701

    # Example usage:
    centimeters = 10
    inches = cm_to_inches(centimeters)
    print(f"{centimeters} centimeters is equal to {inches:.2f} inches.")
```

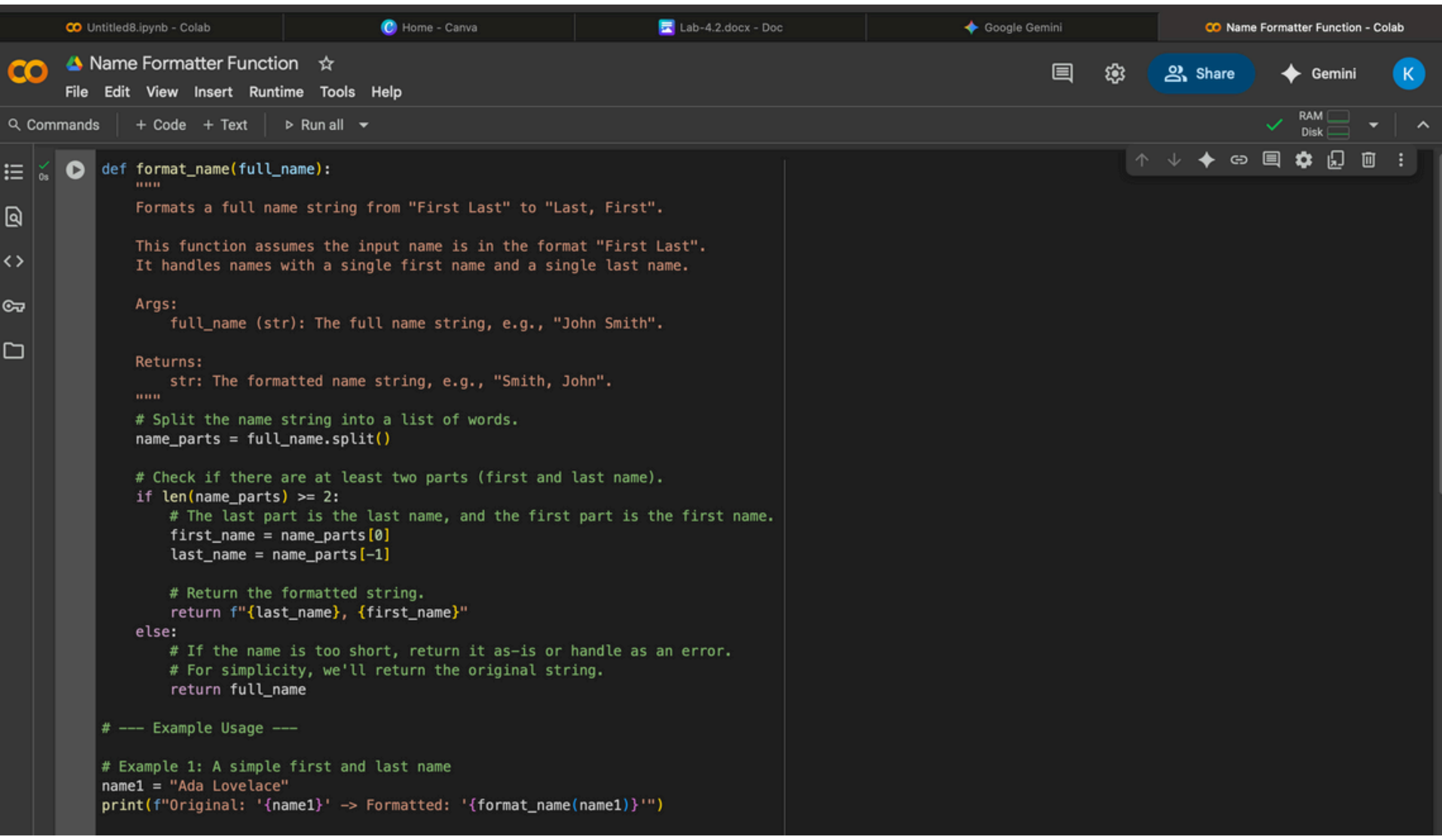
```
10 centimeters is equal to 3.94 inches.
```

TASK 3:-

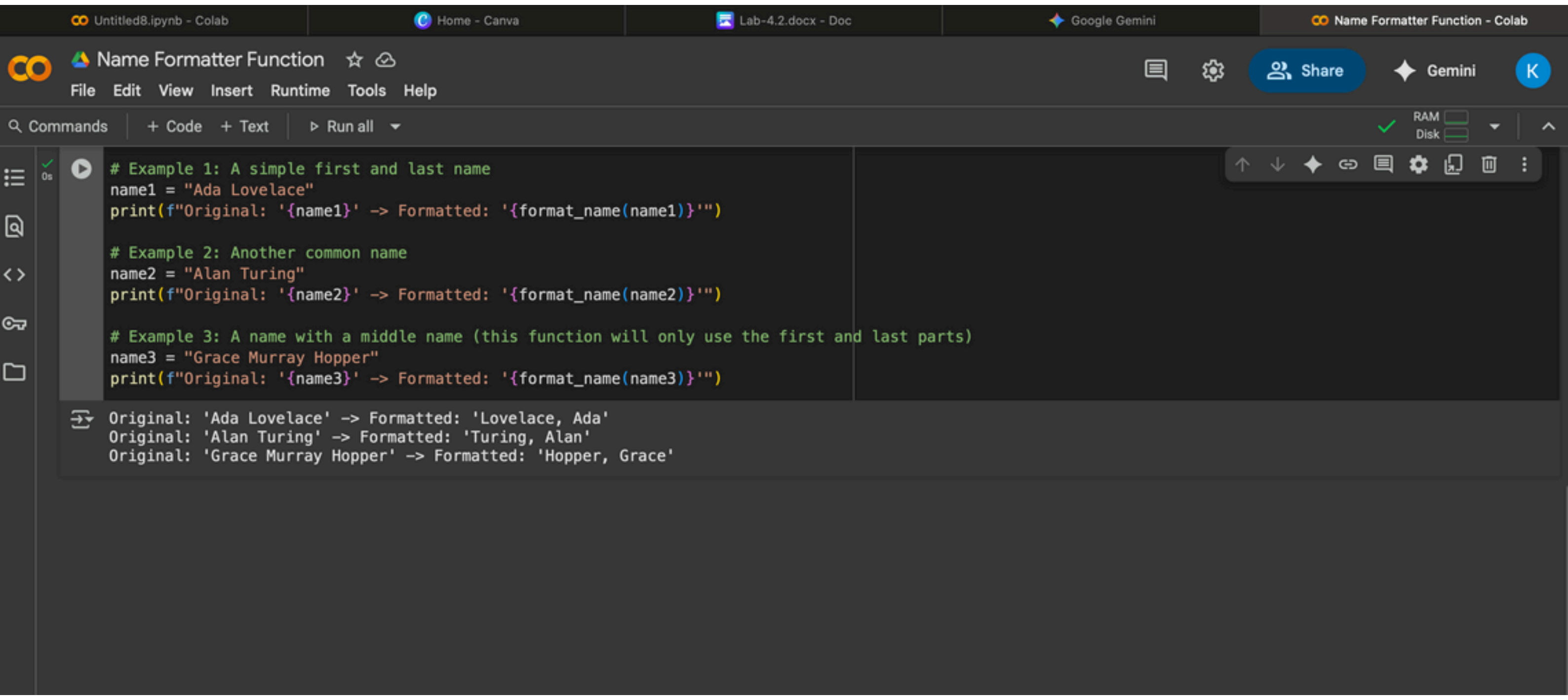
Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”.

Expected Output#3

- **Well-structured function respecting the examples**



```python
def format_name(full_name):
    """
    Formats a full name string from "First Last" to "Last, First".

    This function assumes the input name is in the format "First Last".
    It handles names with a single first name and a single last name.

    Args:
        full_name (str): The full name string, e.g., "John Smith".

    Returns:
        str: The formatted name string, e.g., "Smith, John".
    """
    # Split the name string into a list of words.
    name_parts = full_name.split()

    # Check if there are at least two parts (first and last name).
    if len(name_parts) >= 2:
        # The last part is the last name, and the first part is the first name.
        first_name = name_parts[0]
        last_name = name_parts[-1]

        # Return the formatted string.
        return f"{last_name}, {first_name}"
    else:
        # If the name is too short, return it as-is or handle as an error.
        # For simplicity, we'll return the original string.
        return full_name

# --- Example Usage ---

# Example 1: A simple first and last name
name1 = "Ada Lovelace"
print(f"Original: '{name1}' -> Formatted: '{format_name(name1)}'")
```



```python
# Example 1: A simple first and last name
name1 = "Ada Lovelace"
print(f"Original: '{name1}' -> Formatted: '{format_name(name1)}'")

# Example 2: Another common name
name2 = "Alan Turing"
print(f"Original: '{name2}' -> Formatted: '{format_name(name2)}'")

# Example 3: A name with a middle name (this function will only use the first and last parts)
name3 = "Grace Murray Hopper"
print(f"Original: '{name3}' -> Formatted: '{format_name(name3)}'")
```

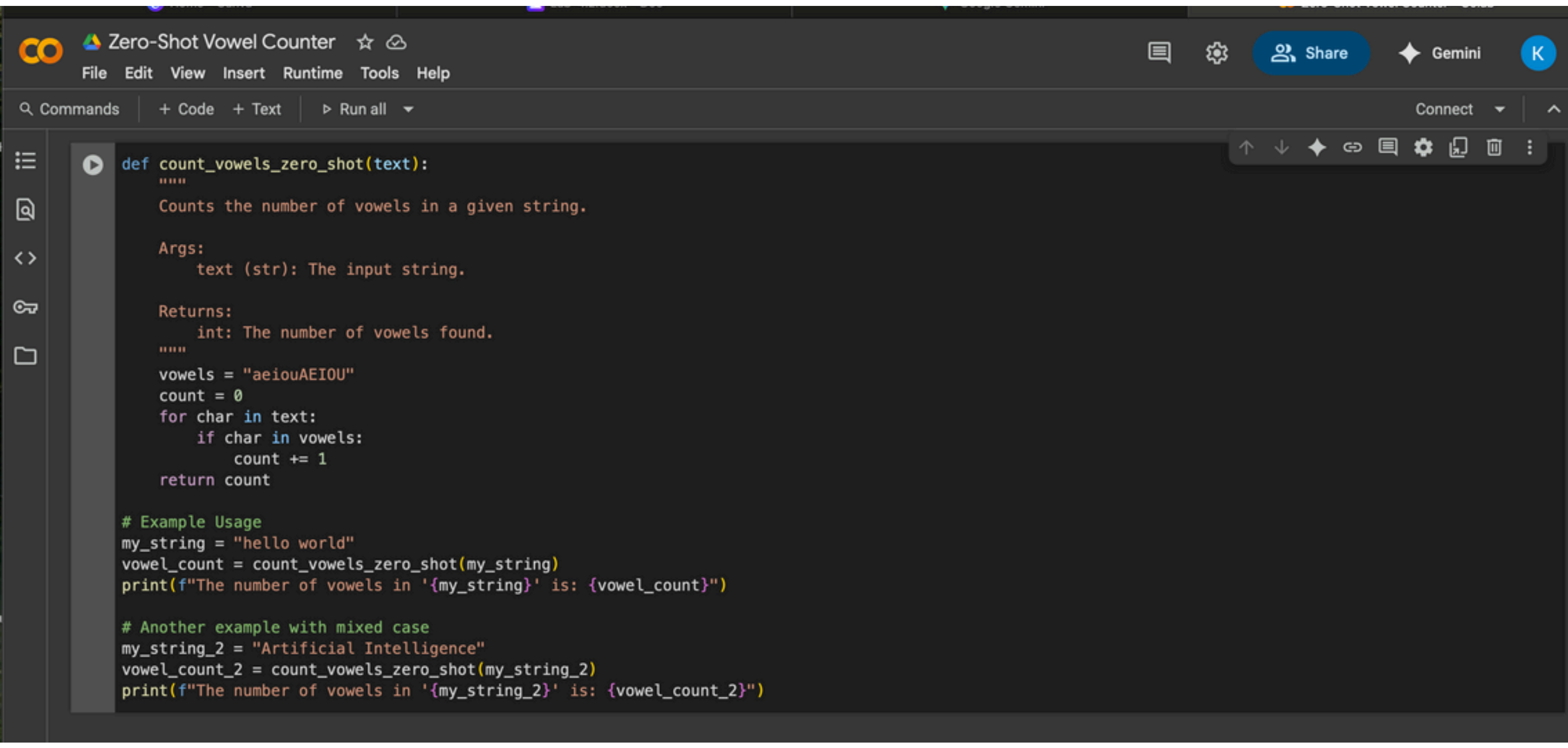
```
Original: 'Ada Lovelace' -> Formatted: 'Lovelace, Ada'
Original: 'Alan Turing' -> Formatted: 'Turing, Alan'
Original: 'Grace Murray Hopper' -> Formatted: 'Hopper, Grace'
```

TASK 4:-

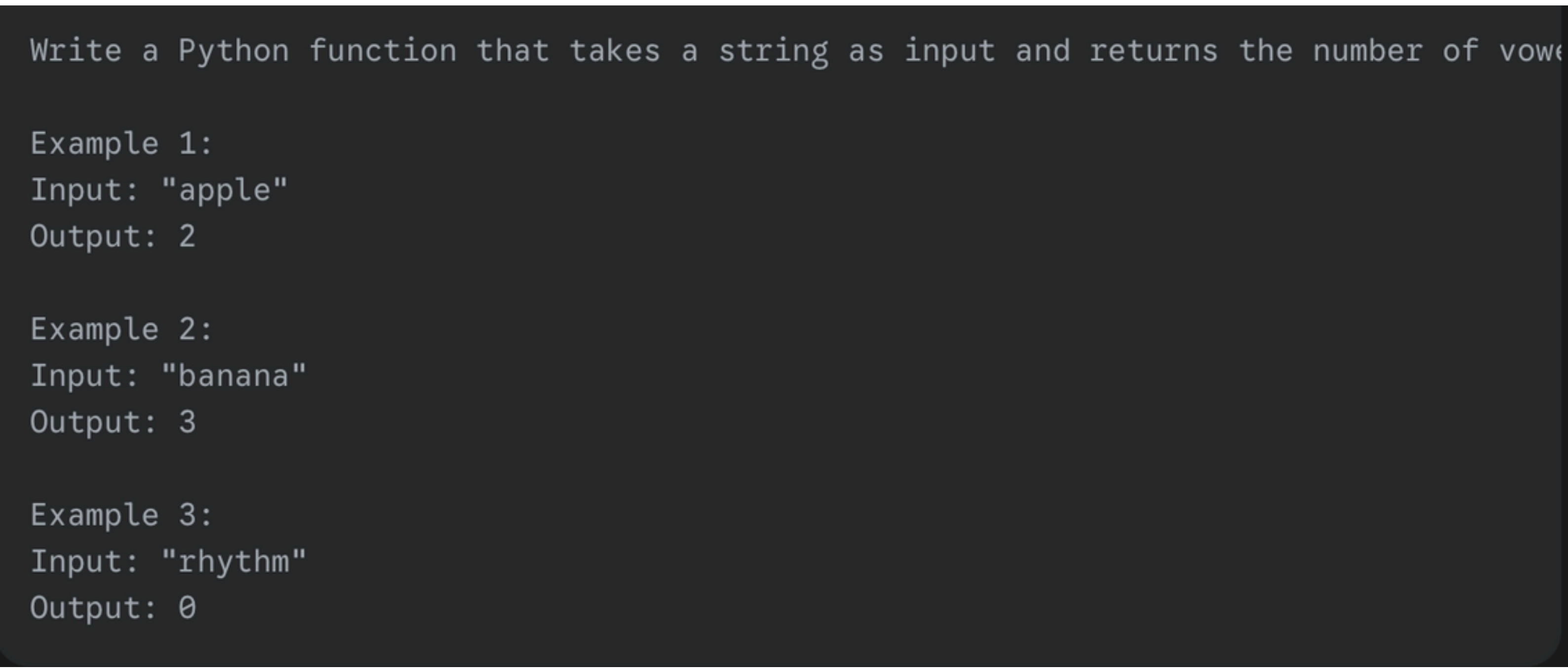
Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.

Expected Output#4

- **Functional output and comparative reflection**


Zero-Shot Vowel Counter
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
Connect

```python
def count_vowels_zero_shot(text):
    """
    Counts the number of vowels in a given string.

    Args:
        text (str): The input string.

    Returns:
        int: The number of vowels found.
    """
    vowels = "aeiouAEIOU"
    count = 0
    for char in text:
        if char in vowels:
            count += 1
    return count

# Example Usage
my_string = "hello world"
vowel_count = count_vowels_zero_shot(my_string)
print(f"The number of vowels in '{my_string}' is: {vowel_count}")

# Another example with mixed case
my_string_2 = "Artificial Intelligence"
vowel_count_2 = count_vowels_zero_shot(my_string_2)
print(f"The number of vowels in '{my_string_2}' is: {vowel_count_2}")
```



```
Write a Python function that takes a string as input and returns the number of vowe

Example 1:
Input: "apple"
Output: 2

Example 2:
Input: "banana"
Output: 3

Example 3:
Input: "rhythm"
Output: 0
```

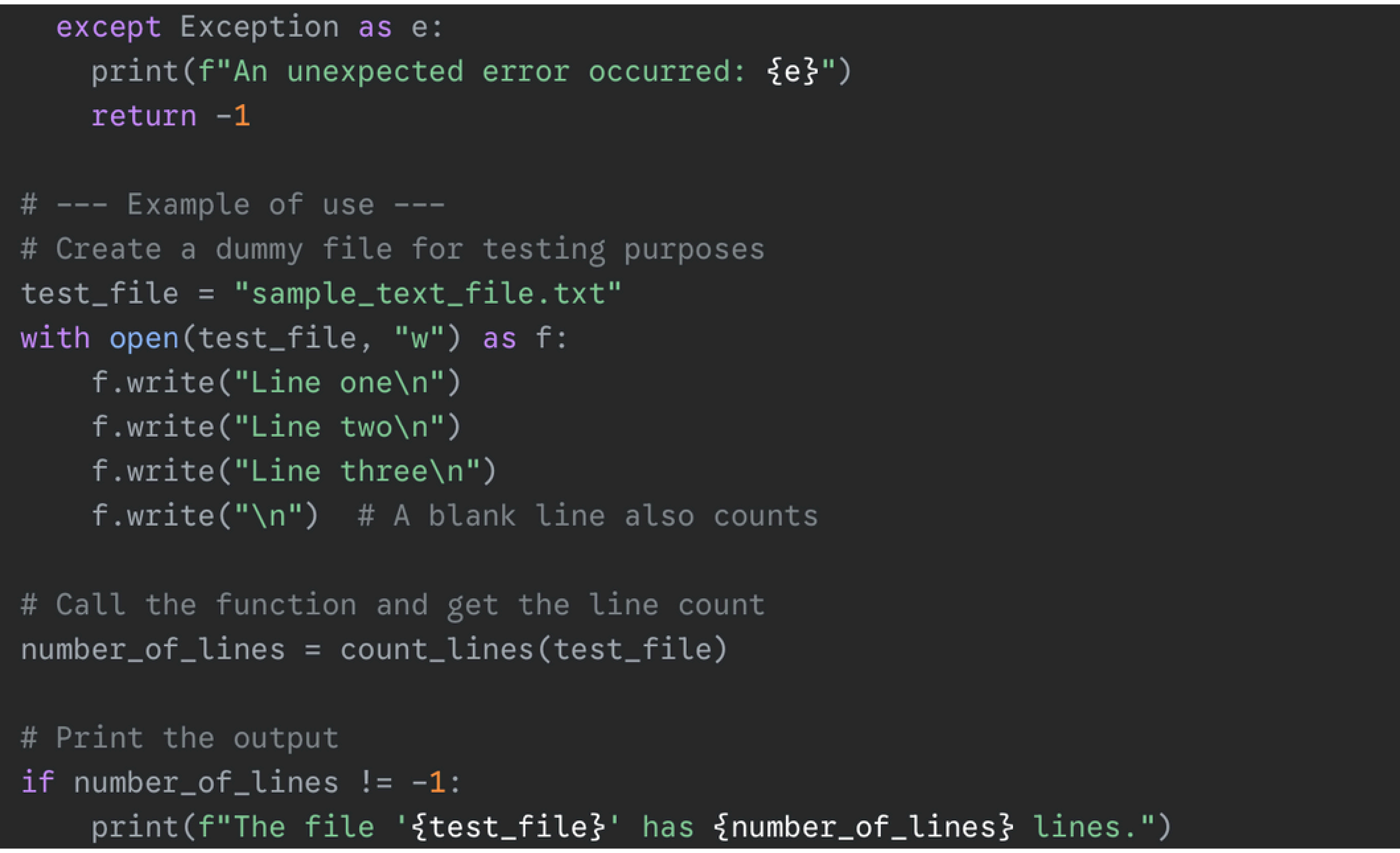
TASK 5:-

Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.

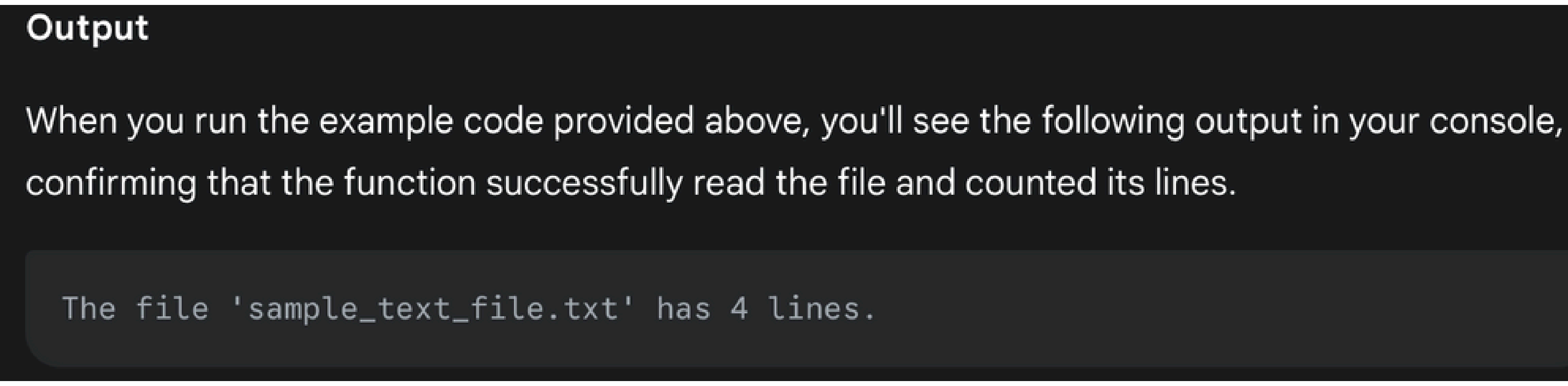
Expected Output#5

- **Working file-processing function with AI-guided logic**

```
def count_lines(filepath):
  """
  Reads a text file and returns the total number of lines.

  Args:
    filepath (str): The path to the text file.

  Returns:
    int: The number of lines in the file, or -1 if an error occurs.
  """
  try:
    with open(filepath, 'r') as file:
      # This generator expression efficiently counts lines
      # without loading the entire file into memory.
      return sum(1 for line in file)
  except FileNotFoundError:
    print(f"Error: The file at '{filepath}' was not found.")
    return -1
  except Exception as e:
    print(f"An unexpected error occurred: {e}")
    return -1

# --- Example of use ---
# Create a dummy file for testing purposes
test_file = "sample_text_file.txt"
with open(test_file, "w") as f:
```



```python
  except Exception as e:
    print(f"An unexpected error occurred: {e}")
    return -1

# --- Example of use ---
# Create a dummy file for testing purposes
test_file = "sample_text_file.txt"
with open(test_file, "w") as f:
    f.write("Line one\n")
    f.write("Line two\n")
    f.write("Line three\n")
    f.write("\n")  # A blank line also counts

# Call the function and get the line count
number_of_lines = count_lines(test_file)

# Print the output
if number_of_lines != -1:
    print(f"The file '{test_file}' has {number_of_lines} lines.")
```



Output

When you run the example code provided above, you'll see the following output in your console, confirming that the function successfully read the file and counted its lines.

```
The file 'sample_text_file.txt' has 4 lines.
```