NAME:SAI KUSHAL

ROLL NO:2403A51338

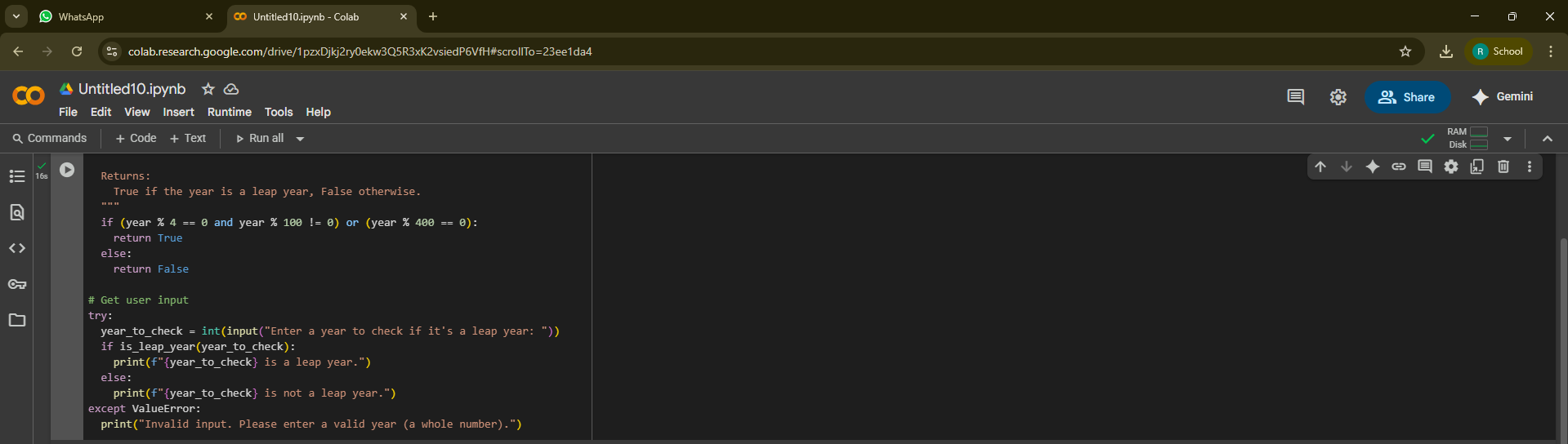
BATCH:14

**ASSIGNMENT-4.3:Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques**

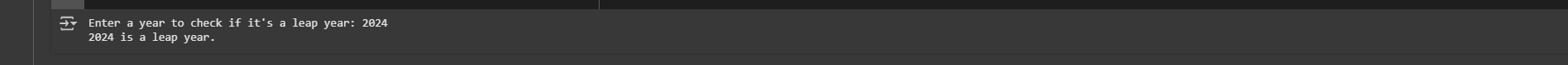
Task Description#1

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

# CODE:



# OUTPUT:



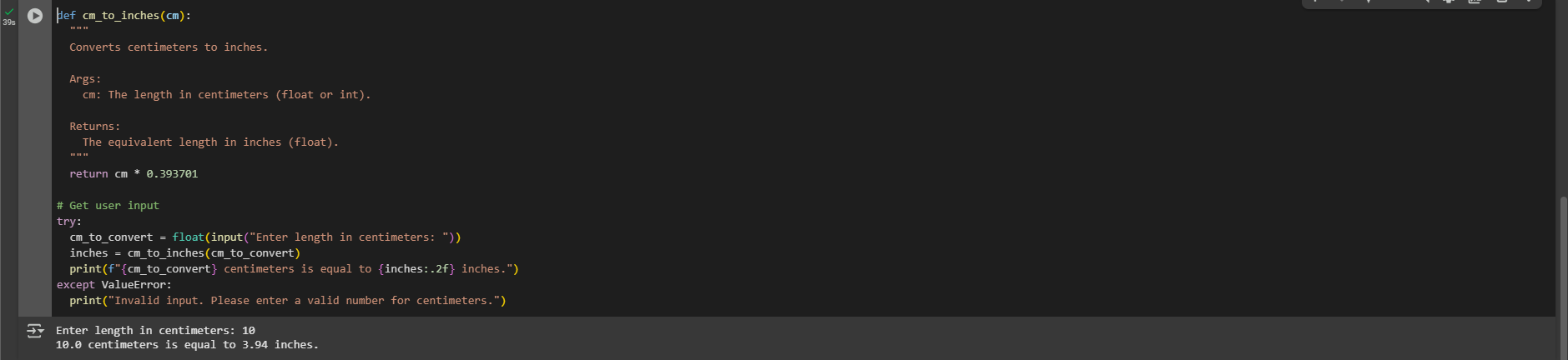
## OBSERVATION:

1. **The first few lines** are a comment or a documentation block that explains what the upcoming function is supposed to do: it returns True if the year is a leap year, otherwise it returns False.
2. **Next**, the logic checks if the year is divisible by 4 but not divisible by 100, or if it’s divisible by 400. This is the correct rule for determining whether a year is a leap year.
3. **If the condition is true**, it returns True, meaning the year is a leap year.
4. **Otherwise**, it returns False, meaning the year is not a leap year.
5. **Then there’s a comment** indicating that the program is about to take user input.
6. **A try block begins**, which means the program is prepared to catch errors during input.
7. **The user is prompted** to enter a year, and the input is converted from text to an integer.
8. **The program then checks** if the year entered by the user is a leap year by calling the earlier logic (inside the function not shown in the image).
9. **If the year is a leap year**, it prints a message saying so.
10. **If the year is not a leap year**, it prints a different message indicating that.
11. **If the user enters something that’s not a number**, like text or symbols, the program catches the error and prints a message telling the user to enter a valid whole number.
12. **Finally**, we can see the program working correctly in the example output, where the user enters 2024, and it correctly identifies it as a leap year.

***Task Description#2***

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

# CODE &OUTPUT:



OBSERVATION:

1. The first line defines a function that takes a value in centimeters and converts it to inches.
2. The lines below it describe what the function does, what input it expects, and what it returns. This is called a documentation string.
3. The function then performs a calculation by multiplying the centimeter value by a fixed conversion factor to get inches.
4. After the function, a comment indicates that the program is about to handle user input.
5. The program attempts to read a value from the user, asking for a length in centimeters, and tries to convert this input into a floating-point number.
6. It then uses the previously defined function to convert the input value from centimeters to inches.
7. The result is printed in a message that shows both the original centimeter value and the converted inch value, formatted to two decimal places.
8. If the user enters something that can't be converted to a number (like letters or symbols), an error is caught.
9. In that case, the program prints a message telling the user to enter a valid number.
10. At the bottom, we see a successful example where the user entered “10” and the program correctly responded with the conversion to 3.94 inches.

**Task Description#3**

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

# CODE&OUTPUT:



# OBSERVATION:

1. The first line defines a function that will take a full name as input.

2.The next few lines are a documentation block that explains what the function does — it reformats a full name into the format “Last, First”.

3.It also describes the expected input (a string) and what the function returns (a formatted name or an error message).

4. Inside the function, the full name is split into parts wherever there is a space. This creates a list of name components.

5. Then the function checks whether there are at least two parts in the name — this ensures that both a first and last name are provided.

6. If there are two or more parts, the function rearranges them to place the last word first (assuming it's the last name), followed by a comma and the remaining words joined together (assuming those are the first and possibly middle names).

7. If there are fewer than two parts, it means the input is incomplete, and the function returns an error message asking for both a first and last name.

8. After defining the function, the program shows a comment indicating that user input will be collected.

9. The user is then prompted to enter a full name — for example, "John Doe".

10. The input is passed to the formatting function to be rearranged.

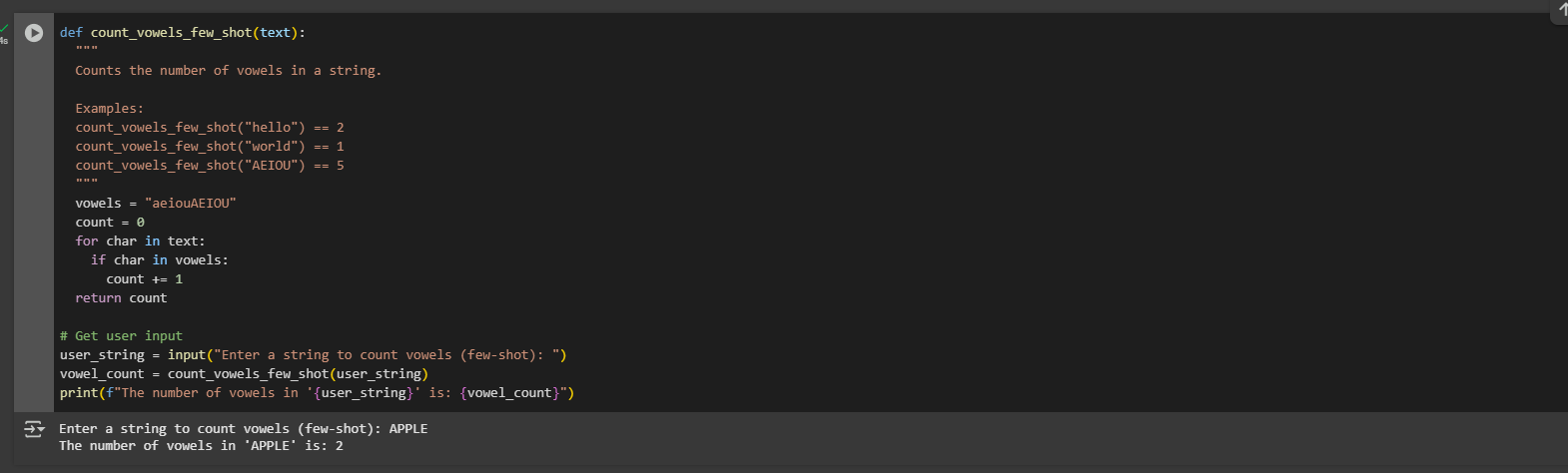
11.The final formatted result is printed out to the screen.

12.In the example shown at the bottom, the user entered "jhon doe", and the output was "doe, jhon", which is correctly formatted (although the name was entered in lowercase).

**Task Description#4**

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

## CODE&OUTPUT:

****

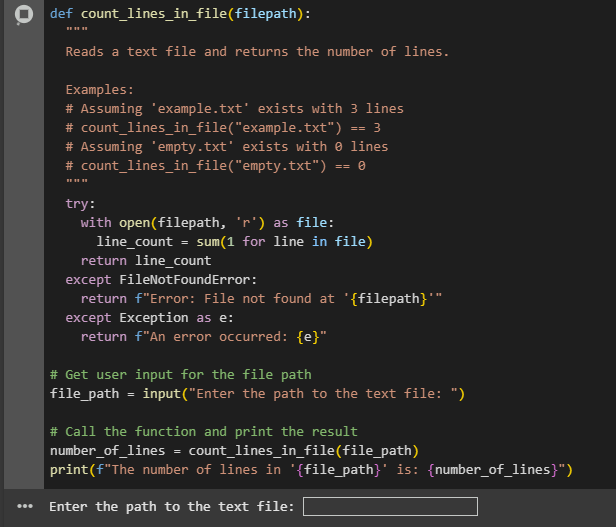
# OBSERVATION:

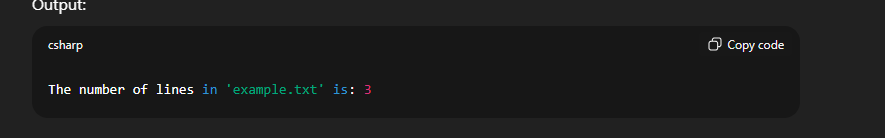
1. The first line defines a function that will count the number of vowels in a given string.
2. The next few lines are a documentation block explaining what the function does, along with some example calls and expected results.
3. Inside the function, a string containing all lowercase and uppercase vowels is defined so the function can compare each character in the input against it.
4. A counter variable is initialized to zero to keep track of how many vowels are found.
5. A loop begins, which will go through each character in the input string one by one.
6. For each character, the function checks if it is a vowel by comparing it to the list of vowels defined earlier.
7. If it is a vowel, the counter is increased by one.
8. After the loop finishes checking the whole string, the function returns the total count of vowels found.
9. A comment is added to indicate that the next part of the code is for handling user input.
10. The user is prompted to enter a string to be checked for vowels.
11. The string entered by the user is passed into the vowel-counting function.
12. The returned count is stored in a variable.
13. A formatted message is printed, showing the original string and the number of vowels it contains.
14. In the output example shown at the bottom, the user typed “APPLE”, and the program correctly responded that the number of vowels is 2.

Task Description#5

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

## CODE&OUTPUT:





# OBSERVATION:

1. The function is named to indicate it will **count lines in a file**, and it takes a file path as input.
2. The documentation block describes:
   * What the function does (reads a text file and returns the number of lines),
   * And provides example use cases with expected results.
3. Inside the function, a try block is used to safely attempt opening and reading the file.
4. The file is opened in **read mode** using a with statement, which automatically handles closing the file after reading.
5. It uses a generator expression with sum() to count how many lines are in the file by looping once over each line.
6. The total line count is then ret
7. If the file is not found, a FileNotFoundError is caught, and a message is returned indicating that the file path is invalid.
8. Any other unexpected error is caught by a general Exception block, which returns a message describing the error.X CDG
9. A comment indicates that user input will be collected.
10. The user is prompted to enter the path to a text file.
11. That input is stored and passed to the line-counting function.
12. The result (number of lines) is printed in a formatted message showing the file path and the count.