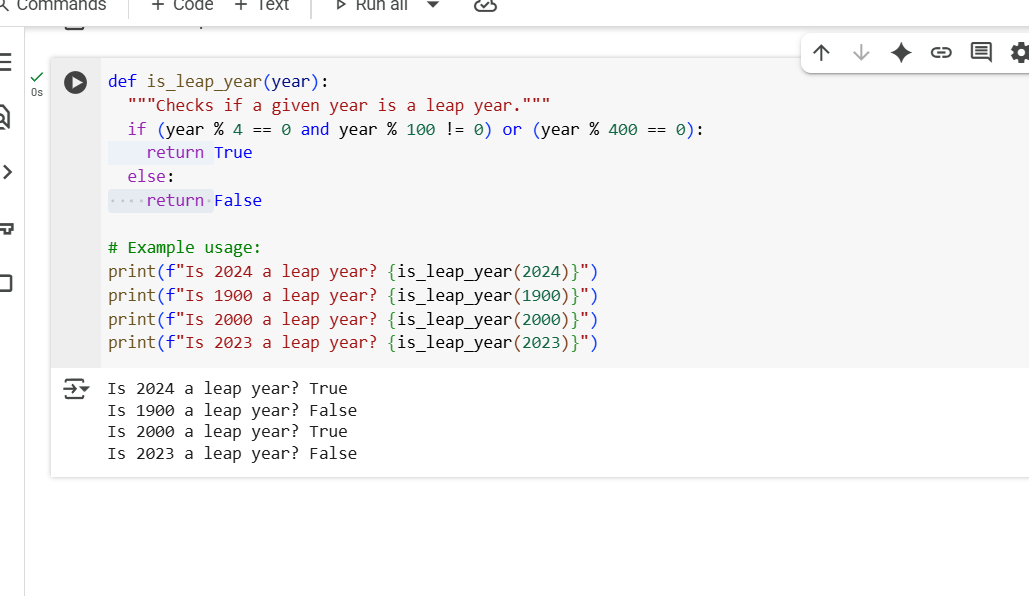
ASSINGMENT-4

Task-1:

 Prompt:

#AI to write a function that checks whether a given year is a leap year. 

Explaination:

def is\_leap\_year(year):

This line defines a function named is\_leap\_year that takes one argument, year.

  """Checks if a given year is a leap year."""

This is a docstring, which explains what the function does. It's good practice to include these in your functions.

  if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

This is the core logic for checking a leap year. It uses a conditional statement (if) with two main conditions combined by or:

1. year % 4 == 0 and year % 100 != 0: This checks if the year is divisible by 4 but not divisible by 100. This is the most common rule for a leap year.
2. year % 400 == 0: This checks if the year is divisible by 400. This handles the exception to the previous rule (years divisible by 100 are usually not leap years, but those divisible by 400 are).

    return True

If either of the conditions in the if statement is true, the function returns True, indicating that the year is a leap year.

  else:

If neither of the conditions in the if statement is true, the code proceeds to the else block.

    return False

In the else block, the function returns False, indicating that the year is not a leap year.

# Example usage:

This is a comment indicating that the following lines demonstrate how to use the function.

print(f"Is 2024 a leap year? {is\_leap\_year(2024)}")

This line calls the is\_leap\_year function with the year 2024 and prints a formatted string showing the result. The f"" creates an f-string, which allows you to embed the result of the function call directly into the string.

print(f"Is 1900 a leap year? {is\_leap\_year(1900)}")

This line calls the is\_leap\_year function with the year 1900 and prints the result in a formatted string.

print(f"Is 2000 a leap year? {is\_leap\_year(2000)}")

This line calls the is\_leap\_year function with the year 2000 and prints the result in a formatted string.

print(f"Is 2023 a leap year? {is\_leap\_year(2023)}")

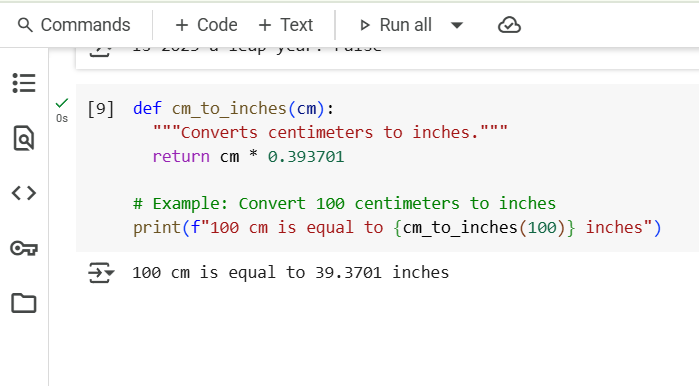
This line calls the is\_leap\_year function with the year 2023 and prints the result in a formatted string

Task-2:

#Let me know if you'd like a more detailed explanation of any part!

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

def cm\_to\_inches(cm):



Explaination:

This line defines a function named cm\_to\_inches that takes one argument, cm, which represents the value in centimeters you want to convert.

  """Converts centimeters to inches."""

This is a docstring, which is a brief explanation of what the function does. It's a good practice for making your code understandable.

  return cm \* 0.393701

This line performs the actual conversion. It multiplies the input value cm by the conversion factor 0.393701 (since 1 centimeter is approximately equal to 0.393701 inches) and then returns the calculated value.

# Example: Convert 100 centimeters to inches

This is a comment indicating that the following line is an example of how to use the function.

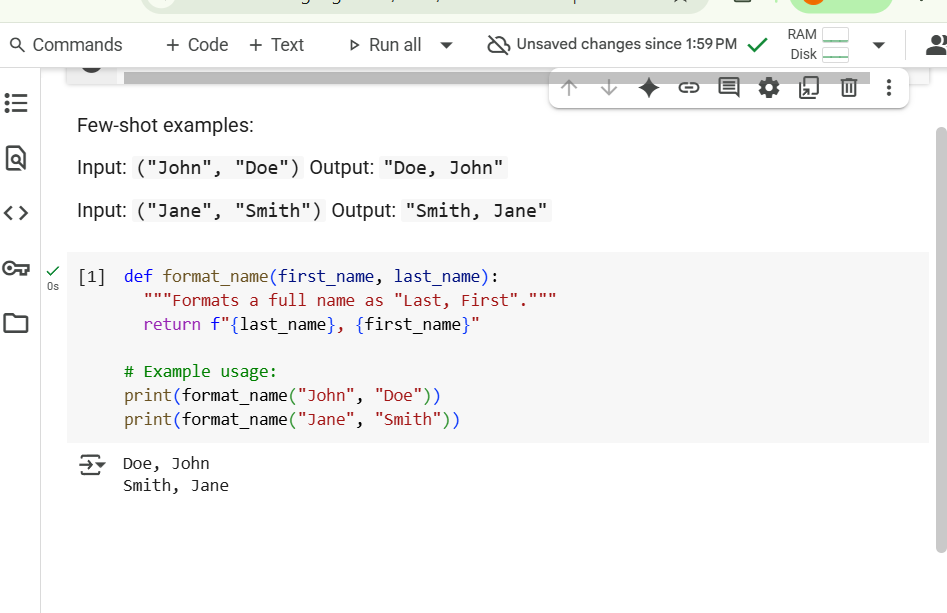
print(f"100 cm is equal to {cm\_to\_inches(100)} inches")

This line calls the cm\_to\_inches function with the value 100 as input. The result of this function call (the converted value in inches) is then embedded into a formatted string (using an f-string, indicated by the f before the opening quote) and printed to the console.

In summary, the code defines a function to convert centimeters to inches and then demonstrates how to use it with an example of converting 100 centimeters.

Let me know if anything is unclear!

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



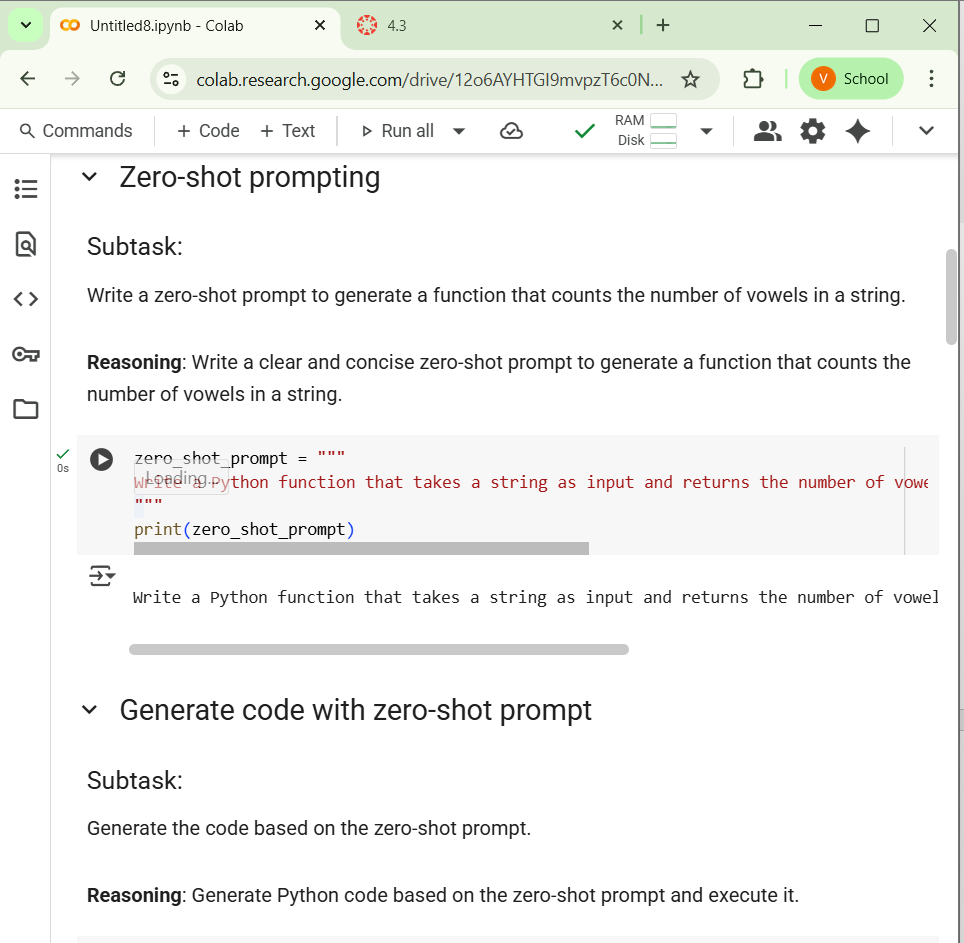
Explaination:

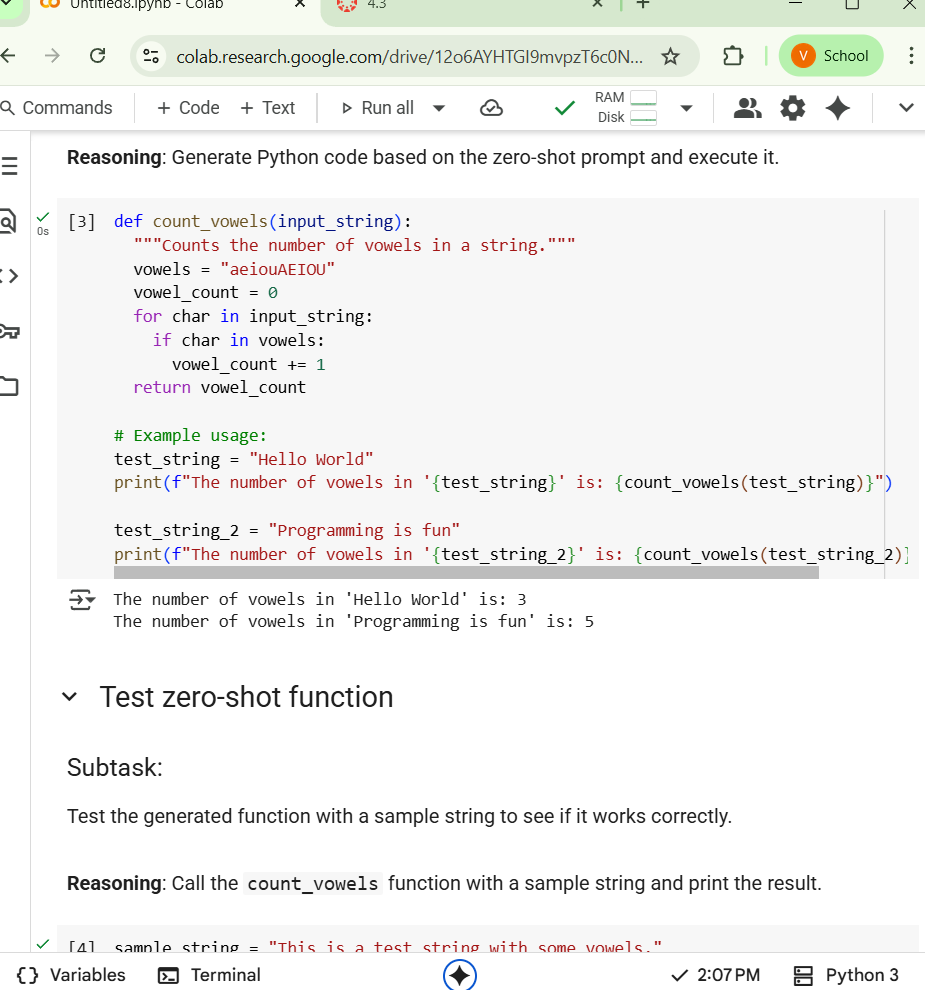
def format\_name(first\_name, last\_name):  
  """Formats a full name as "Last, First"."""  
  return f"{last\_name}, {first\_name}"  
  
# Example usage:  
print(format\_name("John", "Doe"))  
print(format\_name("Jane", "Smith"))

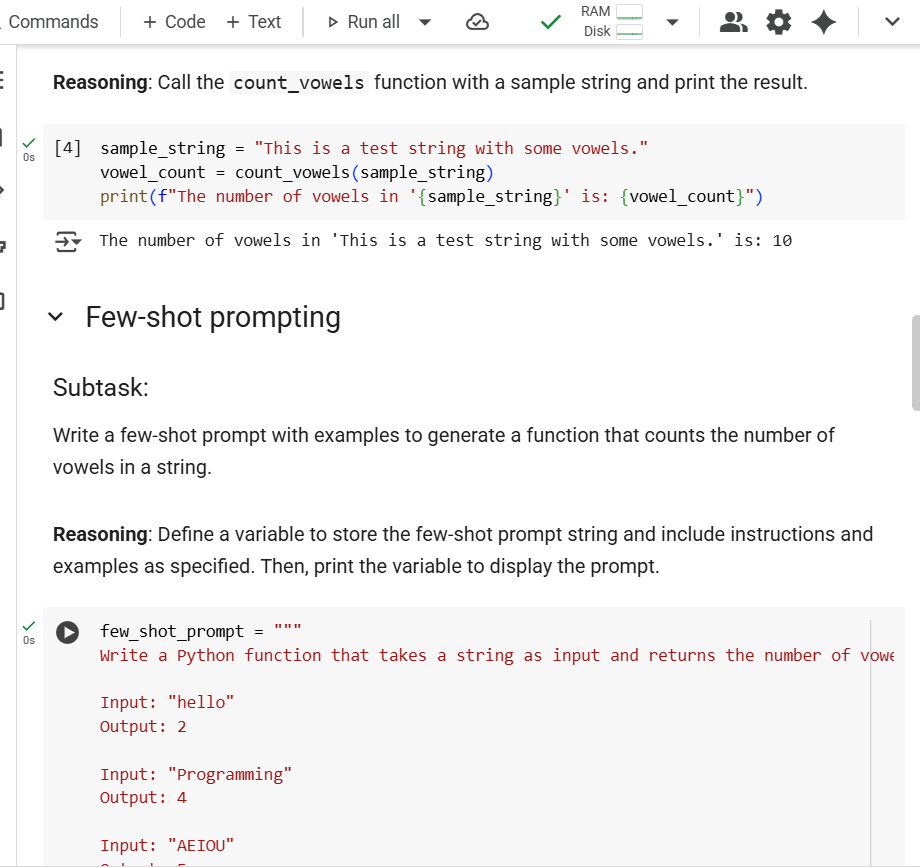
1. def format\_name(first\_name, last\_name):: This line defines a function named format\_name. It takes two parameters: first\_name and last\_name.
2. """Formats a full name as "Last, First".""": This is a docstring. It explains what the function does. It's good practice to include docstrings in your functions.
3. return f"{last\_name}, {first\_name}": This line is the core of the function.
   * f"..." indicates an f-string (formatted string literal). This allows you to embed expressions inside string literals.
   * {last\_name}: This inserts the value of the last\_name parameter into the string.
   * ,: This adds a comma and a space after the last name.
   * {first\_name}: This inserts the value of the first\_name parameter into the string.
   * return: This keyword sends the resulting formatted string back as the output of the function.
4. # Example usage:: This is a comment line indicating that the following lines demonstrate how to use the function. Comments are ignored by the Python interpreter.
5. print(format\_name("John", "Doe")): This line calls the format\_name function with "John" as the first name and "Doe" as the last name. The function returns "Doe, John", which is then printed to the console.
6. print(format\_name("Jane", "Smith")): This line calls the format\_name function with "Jane" as the first name and "Smith" as the last name. The function returns "Smith, Jane", which is then printed to the console.

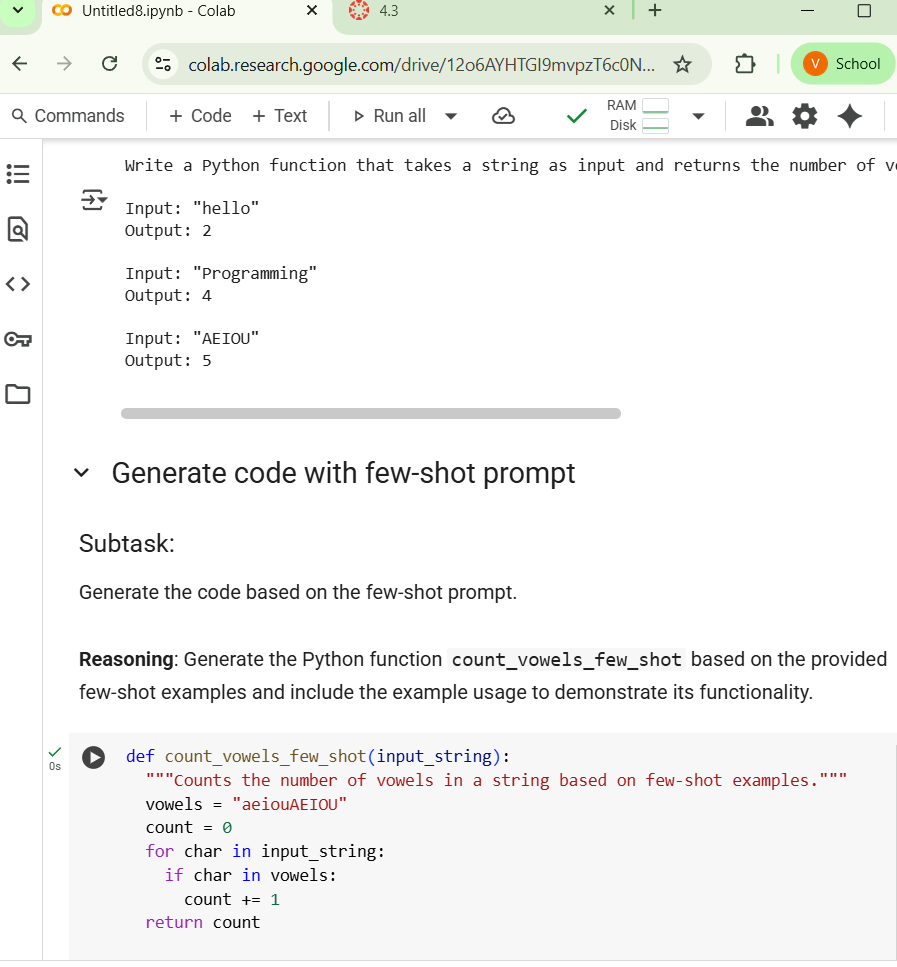
Task-4

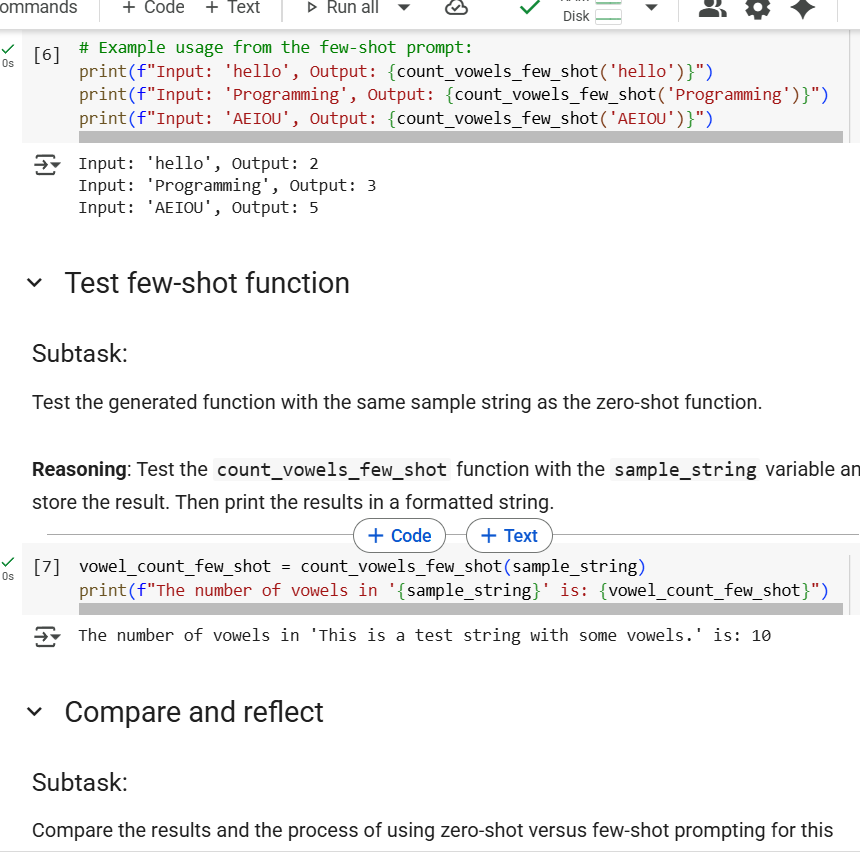
#Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string. Provide the functional output and a comparative reflection.

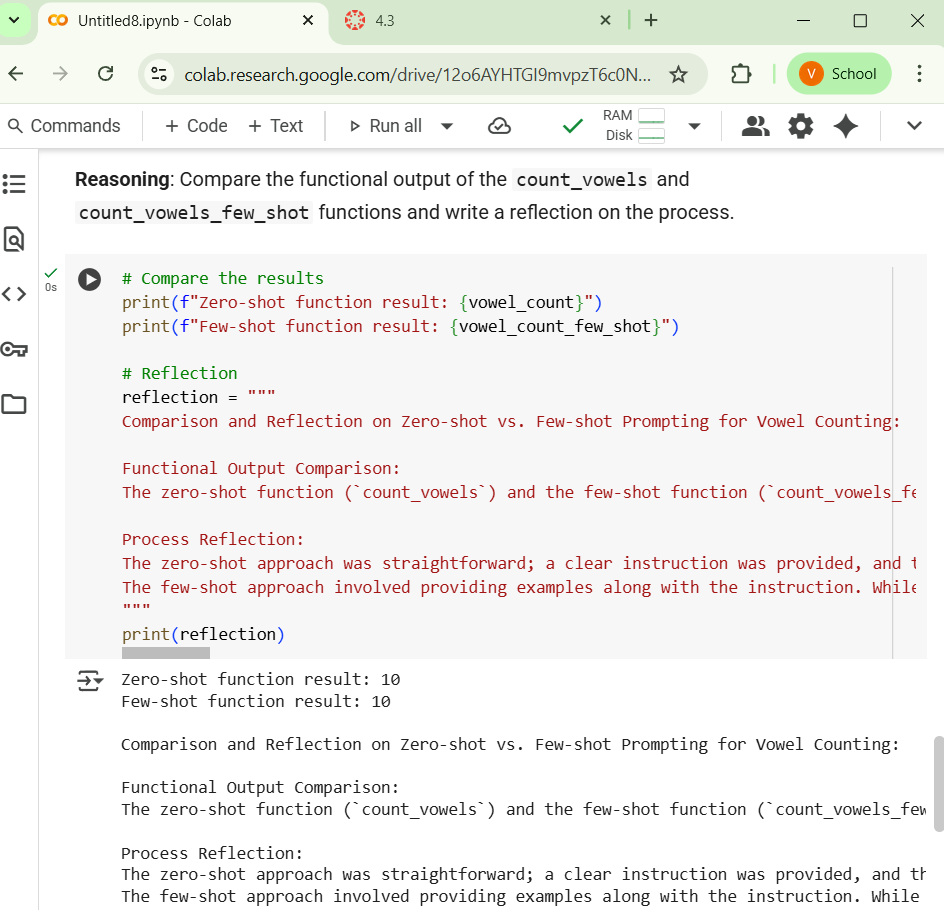












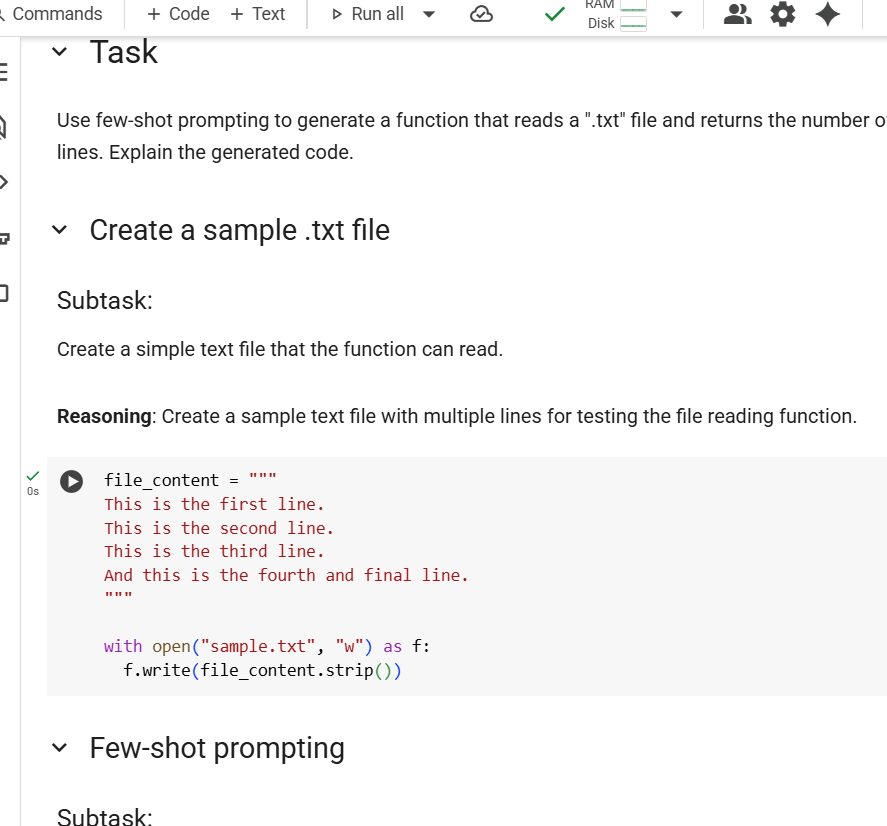
Explaination:

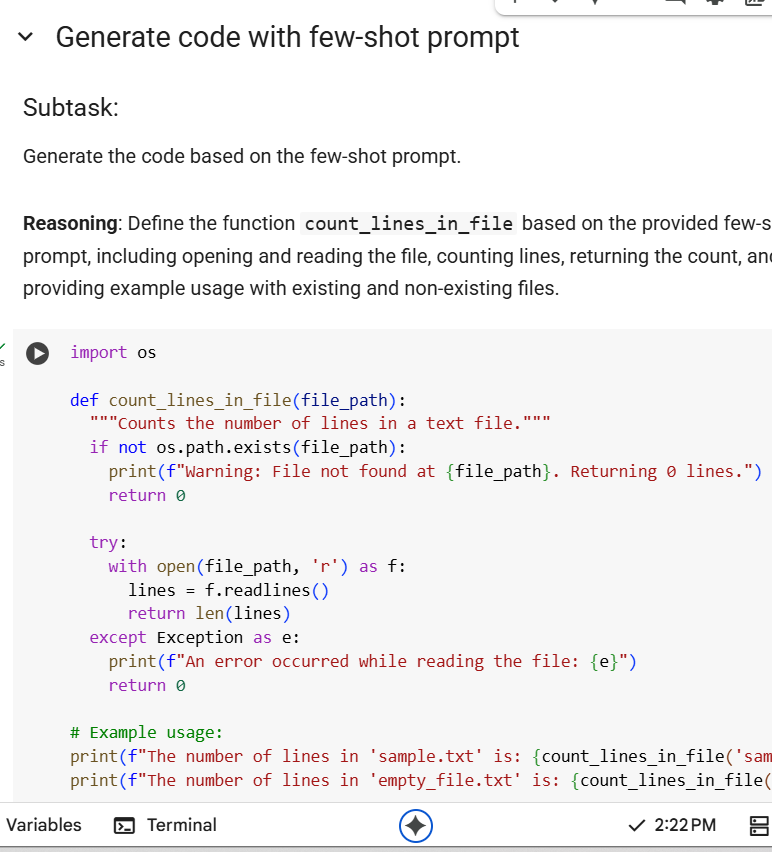
def count\_vowels(input\_string):  
  """Counts the number of vowels in a string."""  
  vowels = "aeiouAEIOU"  
  vowel\_count = 0  
  for char in input\_string:  
    if char in vowels:  
      vowel\_count += 1  
  return vowel\_count

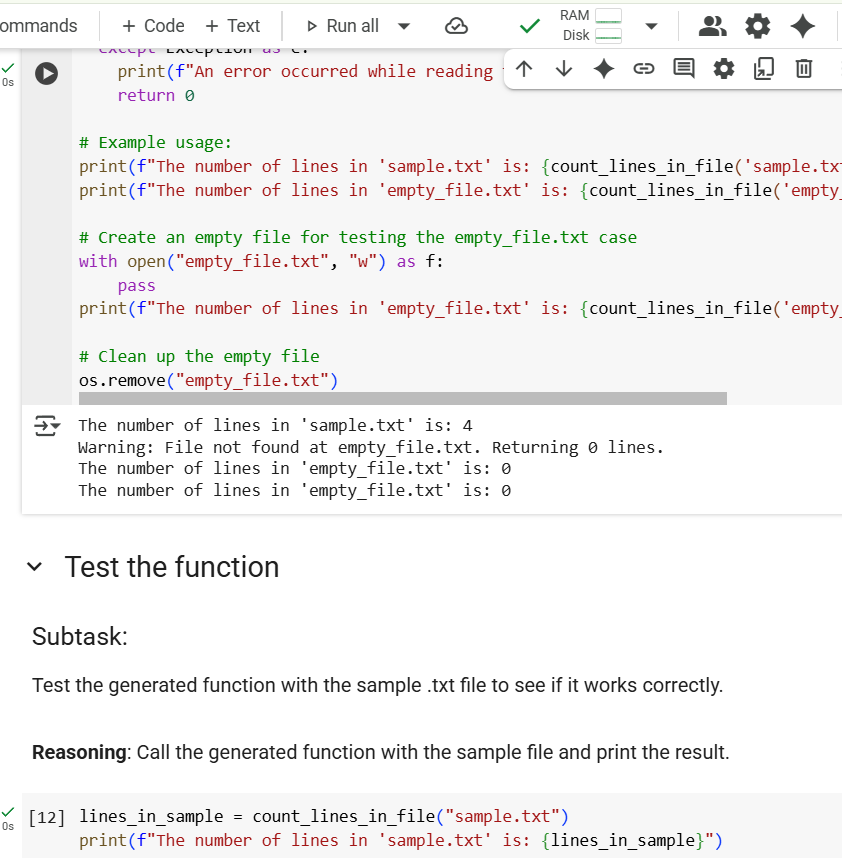
1. def count\_vowels(input\_string):: This line defines a function named count\_vowels that takes one argument: input\_string.
2. """Counts the number of vowels in a string.""": This is a docstring explaining what the function does.
3. vowels = "aeiouAEIOU": This line creates a string variable named vowels and assigns it the value containing all lowercase and uppercase vowels. This string will be used to check if a character is a vowel.
4. vowel\_count = 0: This line initializes an integer variable named vowel\_count to 0. This variable will keep track of the number of vowels found in the input string.
5. for char in input\_string:: This line starts a for loop that iterates through each character in the input\_string. In each iteration, the current character is assigned to the variable char.
6. if char in vowels:: This line checks if the current character (char) is present in the vowels string (which contains all lowercase and uppercase vowels).
7. vowel\_count += 1: If the condition in the if statement is true (i.e., the character is a vowel), this line increments the vowel\_count by 1.
8. return vowel\_count: After the loop has finished iterating through all the characters in the input string, this line returns the final value of vowel\_count, which is the total number of vowels found in the string.

Task-5

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







Explaination:

import os  
  
def count\_lines\_in\_file(file\_path):  
  """Counts the number of lines in a text file."""  
  if not os.path.exists(file\_path):  
    print(f"Warning: File not found at {file\_path}. Returning 0 lines.")  
    return 0  
  
  try:  
    with open(file\_path, 'r') as f:  
      lines = f.readlines()  
      return len(lines)  
  except Exception as e:  
    print(f"An error occurred while reading the file: {e}")  
    return 0

1. import os: This line imports the os module, which provides a way to interact with the operating system. We need this to check if a file exists.
2. def count\_lines\_in\_file(file\_path):: This line defines a function named count\_lines\_in\_file that takes one argument: file\_path, which is expected to be a string representing the path to the text file.
3. """Counts the number of lines in a text file.""": This is a docstring explaining what the function does.
4. if not os.path.exists(file\_path):: This line checks if the file specified by file\_path does *not* exist using the os.path.exists() function.
5. print(f"Warning: File not found at {file\_path}. Returning 0 lines."): If the file does not exist, this line prints a warning message indicating that the file was not found and that the function will return 0.
6. return 0: If the file does not exist, this line returns 0, as there are no lines in a non-existent file.
7. try:: This line starts a try block. Code within a try block is monitored for exceptions (errors).
8. with open(file\_path, 'r') as f:: This line opens the file specified by file\_path in read mode ('r'). The with statement ensures that the file is automatically closed even if errors occur. The opened file object is assigned to the variable f.
9. lines = f.readlines(): This line reads all the lines from the opened file f and stores them as a list of strings in the variable lines. Each string in the list represents a line from the file, including the newline character (\n) at the end of most lines.
10. return len(lines): This line calculates the number of elements (which are lines) in the lines list using the len() function and returns this count.
11. except Exception as e:: This line starts an except block, which catches any exceptions that might occur within the preceding try block. Exception as e catches any type of exception and assigns the exception object to the variable e.
12. print(f"An error occurred while reading the file: {e}"): If an exception occurs during file reading, this line prints an error message including the specific error information stored in e.
13. return 0: If an error occurs while reading the file, this line returns 0.

This function is designed to be robust by checking for file existence and handling potential errors during file operations.