ASSIGNMENT 4.3

HTNO:2403A51319

TASK 1:

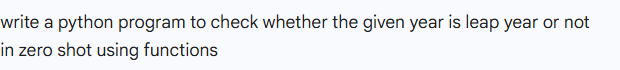
USING GOOGLE COLAB:

**Task Description#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
* PROMPT:



CODE:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

CODE EXPLANATION:

 The code defines a function called is\_leap that checks if a year is a leap year.

 A leap year happens every 4 years, **but** not every year that is divisible by 100.

 However, if a year **is divisible by 400**, it **is** still a leap year.

 The function uses this rule to return True (if it is a leap year) or False.

 A list of three years is created: 2024, 1900, and 2000.

 The code goes through each of these years one by one using a loop.

 It checks if the current year is a leap year by calling the is\_leap function.

 If the function returns True, it prints that the year **is** a leap year.

 If it returns False, it prints that the year **is not** a leap year.

 This helps us understand which years are leap years based on simple rules.

**Task Description#2**

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

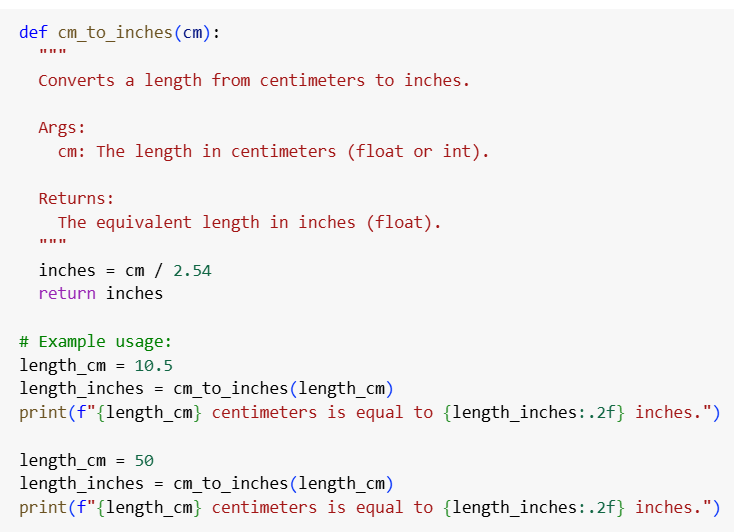
**Expected Output#2**

* Function with correct conversion logic

PROMPT:



CODE:



CODE EXPLANATION:

 **Purpose**:  
The code is written to convert a length from **centimeters to inches** using a function.

 **Function Use**:  
A function is created to do the conversion, so it can be reused easily for any value.

 **How It Works**:  
Inside the function, the given value in centimeters is divided by **2.54**, because **1 inch = 2.54 centimeters**. This gives the equivalent value in inches.

 **Return Value**:  
After doing the conversion, the function gives back (returns) the result in inches.

 **Examples**:  
The code then shows two examples:

* First, it converts **10.5 centimeters** into inches.
* Then, it converts **50 centimeters** into inches.

 **Output Format**:  
The results are printed in a user-friendly way, showing the original value in centimeters and the converted value in inches, rounded to two decimal places.

 **Result**:  
When the code runs, it clearly tells you how many inches each centimeter value equals.

OUTPUT:

A white background with black text

AI-generated content may be incorrect.

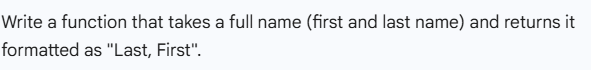
**Task Description#3**

* 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

PROMPT:



CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

CODE EXPLANATION:

 **Function Definition**  
A function called format\_name is created. It takes a single input: full\_name, which is expected to be a string like "John Doe".

 **Splitting the Name**  
Inside the function, it uses .split() to break the full name into parts using spaces.

* For "John Doe" → it becomes ["John", "Doe"]
* For "Peter Pan" → ["Peter", "Pan"]

 **Handling the Parts**

* If the name has **two or more parts**, the code:
  + Takes the **last word** as the **last name**
  + Joins all the previous words (usually the first name) as the **first part**
  + Returns a string formatted as "Last, First"

OUTPUT:

A close-up of a name

AI-generated content may be incorrect.

**Task Description#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

PROMPT:



CODE:

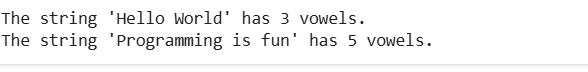
A screen shot of a computer code

AI-generated content may be incorrect.

CODE EXPLANATION:

1. **Takes a sentence or word as input**  
   You give the program a string of text (like "Hello World" or "Programming is fun").
2. **Looks at each letter one by one**  
   It goes through the text, checking each character in the string.
3. **Checks if the character is a vowel**  
   For every character, it asks:  
   *"Is this an A, E, I, O, or U (upper or lowercase)?"*
4. **Keeps track of how many vowels it finds**  
   If the character is a vowel, it adds 1 to a running total.
5. **Gives you the total number of vowels**  
   After checking the whole string, it tells you how many vowels it found.

OUTPUT:



**Task Description#5**

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

**Expected Output#5**

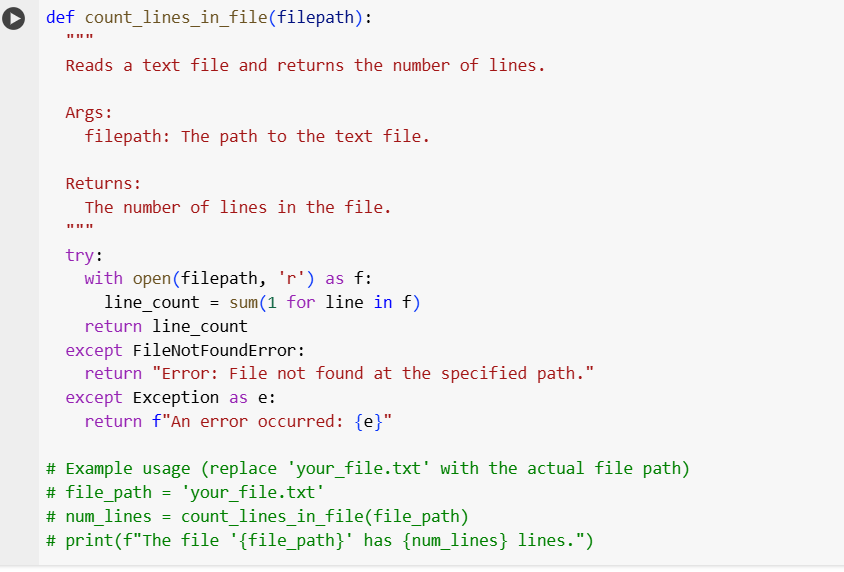
* Working file-processing function with AI-guided logic

PROMPT:

A close up of a text

AI-generated content may be incorrect.

CODE:



CODE EXPLANATION:

1. **def count\_lines\_in\_file(filepath):**: This line defines the function named count\_lines\_in\_file which accepts a single parameter filepath. This filepath should be a string representing the location of the text file you want to read.
2. **""" Docstring """**: This is a docstring, which explains what the function does, its arguments (Args), and what it returns (Returns). This is good practice for making your code understandable.
3. **try:**: This starts a try block. Code inside this block will be attempted, and if an error occurs, the code in the except block will be executed. This is used for error handling.
4. **with open(filepath, 'r') as f:**: This opens the file specified by filepath 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.
5. **line\_count = sum(1 for line in f)**: This is the core of the line counting.
   * for line in f: This iterates through each line in the opened file f.
   * 1 for line in f: This is a generator expression that yields 1 for every line it reads from the file.
   * sum(...): This sums up all the 1s generated, effectively counting the number of lines.
6. **return line\_count**: If the file is opened and read successfully, the function returns the calculated line\_count.
7. **except FileNotFoundError:**: This block is executed if the try block raises a FileNotFoundError, which happens if the file specified by filepath does not exist. It returns a specific error message.
8. **except Exception as e:**: This block catches any other type of error (Exception) that might occur during file processing (like permission errors, etc.). It returns a generic error message including the details of the error (e).
9. **# Example usage...**: The lines that are commented out show how you would call the function and print the result. You would replace 'your\_file.txt' with the actual path to your file.