AI ASSISTED CODING

Assignment-4 : Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

**2403A52107**

**Pavani Voddepalli**

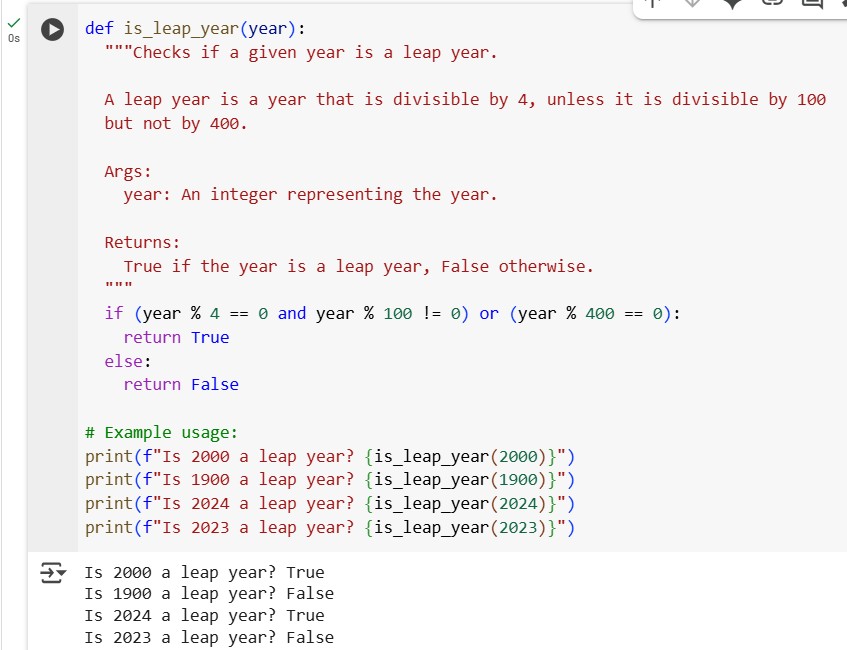
**Batch : 24BTCAIAI05**

**TASK 1 :**

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

PROMPT : Generate a python function that checks a given year is a leapyear.

**CODE & OUTPUT:**



**EXPLANATION :**

The Python code defines a function called is\_leap\_year that takes a single input, the year you want to check. The purpose of this function is to determine if that year follows the rules of a leap year in the Gregorian calendar.

The logic inside the function is based on the standard definition of a leap year:

* A year is generally a leap year if it is perfectly divisible by 4.
* However, there's an exception for years that are divisible by 100. These century years are *not* leap years unless...
* ...they are also perfectly divisible by 400.

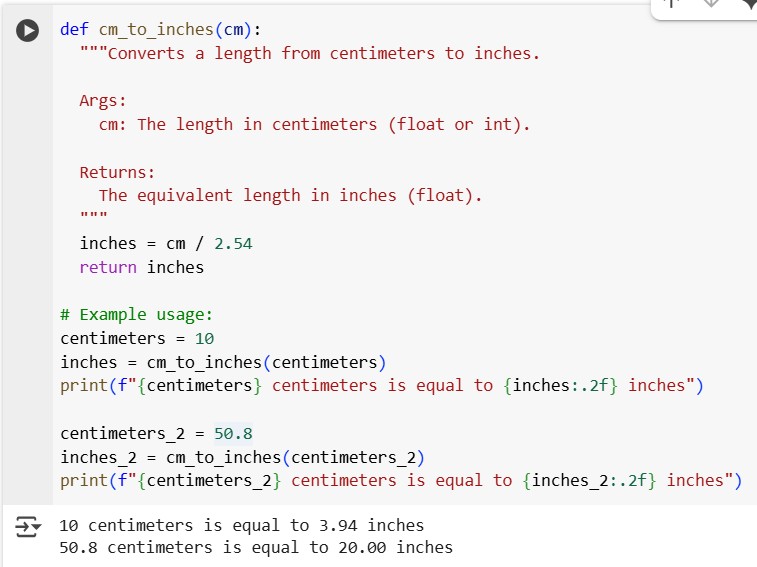
The code translates these rules into a concise conditional statement. It first checks if the year is divisible by 4 AND not divisible by 100. If this condition is true, the year is a leap year. Alternatively, it checks if the year is divisible by 400. If either of these main conditions is met, the function concludes that the year is a leap year and returns True. If neither condition is met, the year is not a leap year, and the function returns False.

**TASK 2 :**

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

PROMPT : Write a Python function that converts a length from centimeters to inches. 1 inch = 2.54 centimeters.

**CODE & OUTPUT :**



**EXPLANATION :**

1. **Function Definition:** The code defines a function named cm\_to\_inches that is designed to accept one input value, which represents a measurement in centimeters.
2. **Receive Input:** The function receives the length measurement in centimeters as the argument cm.
3. **Perform Conversion:** Inside the function, it takes the value of cm and divides it by the conversion factor 2.54. This calculation converts the measurement from centimeters to inches.
4. **Store Result:** The result of the division (the length in inches) is stored in a variable named inches.
5. **Return Value:** The function then provides the calculated value of inches back as its output.

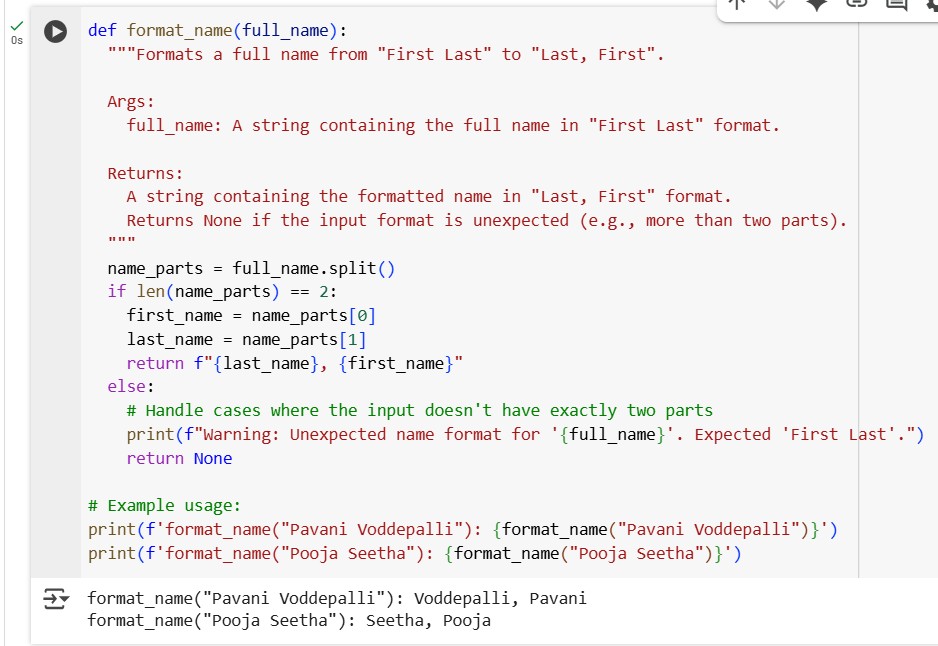
The example usage then demonstrates these steps:

1. A variable centimeters is set to a value (e.g., 10).
2. The cm\_to\_inches function is called with centimeters as the input.
3. The function performs the conversion (10 / 2.54).
4. The returned value (approximately 3.937) is stored in the inches variable.
5. Finally, a formatted string is printed showing the original centimeter value and the calculated inch value.
6. These steps are repeated for a second example value (50.8 centimeters).

**TASK 3 :**

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

PROMPT : Write a Python function that takes a full name in the format "First Last" and returns it as "Last, First". Use examples like format\_name("Pavani Voddepalli") # "Voddepalli , Pavani" format\_name("Pooja Seetha) # "Seetha , Pooja" **CODE & OUTPUT :**



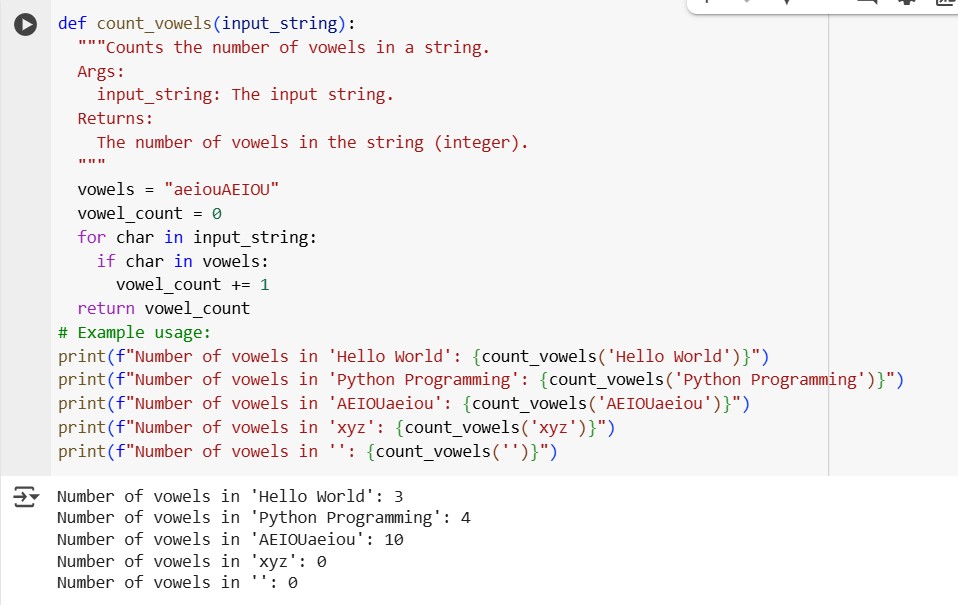
**EXPLANATION :**

* **Splits the name:** The function takes a full name as a single string and splits it into a list of parts (words) based on spaces.
* **Checks the number of parts:** It checks if the split resulted in exactly two parts (assuming a "First Last" format).
* **Formats the name (if two parts):** If there are two parts, it takes the second part (Last Name), followed by a comma and a space, and then the first part (First Name), and returns this new formatted string.

**TASK 4 : Compare zero-shot and few-shot prompts** for writing a function that counts the number of vowels in a string.

**PROMPT FOR ZERO SHOT :** Write a Python function that takes a string as input and returns the number of vowels in it.Consider both uppercase and lowercase vowels: a, e, i, o, u.

**CODE & OUTPUT :**



**PROMPT FOR FEW SHOT :** Write a Python function that takes a string and returns the number of vowels it contains. Vowels include both uppercase and lowercase: a, e, i, o, u. Use the examples: count\_vowels("hello") # 2 count\_vowels("HELLO") # 2 count\_vowels("Python") # 1

count\_vowels("AEIOUaeiou") # 10

**CODE & OUTPUT :**



**COMPARISION:**

**Zero-shot:** The first prompt is a **zero-shot** prompt because it provides the task description without any explicit input-output examples. It relies solely on the AI's understanding of the request and its general knowledge of Python and vowel counting.

• **Few-shot:** The second prompt is a **few-shot** prompt because it includes several input-output examples (count\_vowels("hello") # ➞ 2, etc.) to demonstrate the desired behavior of the function. These examples provide the AI with specific instances of what the function should do, helping to guide the generated code towards the intended output format and logic.

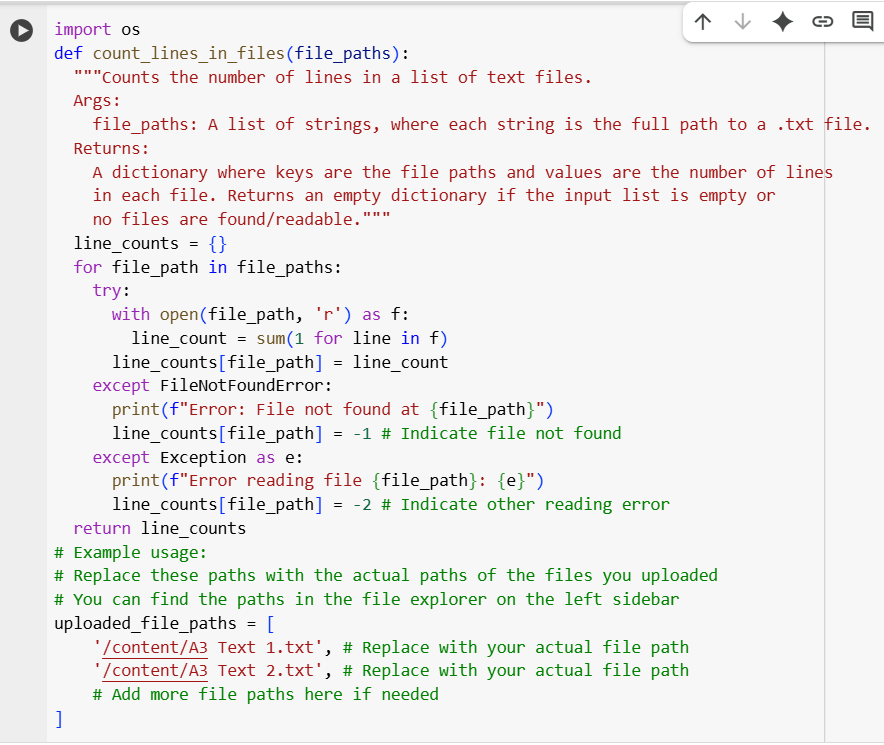
**Potential Impact:**

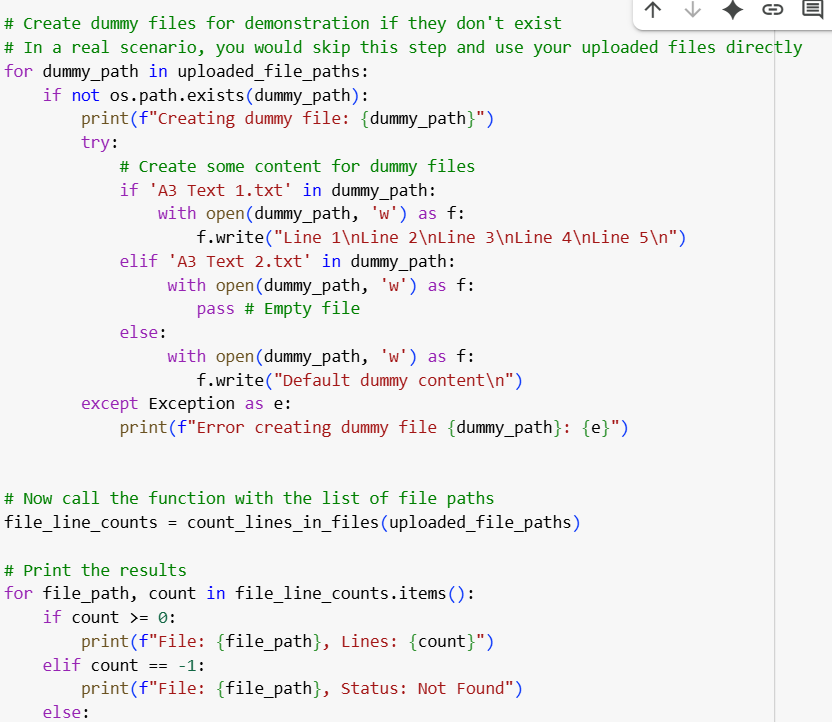
While both prompts successfully led to the generation of a correct count\_vowels function in this case, including examples in the few-shot prompt can be particularly helpful for more complex or ambiguous tasks. The examples clarify the expected input format, output format, and edge cases, reducing the likelihood of the AI misinterpreting the request. In simpler cases like this, the impact on the core logic might be minimal, but the few-shot prompt explicitly guided the inclusion of those specific examples in the generated code's usage section.

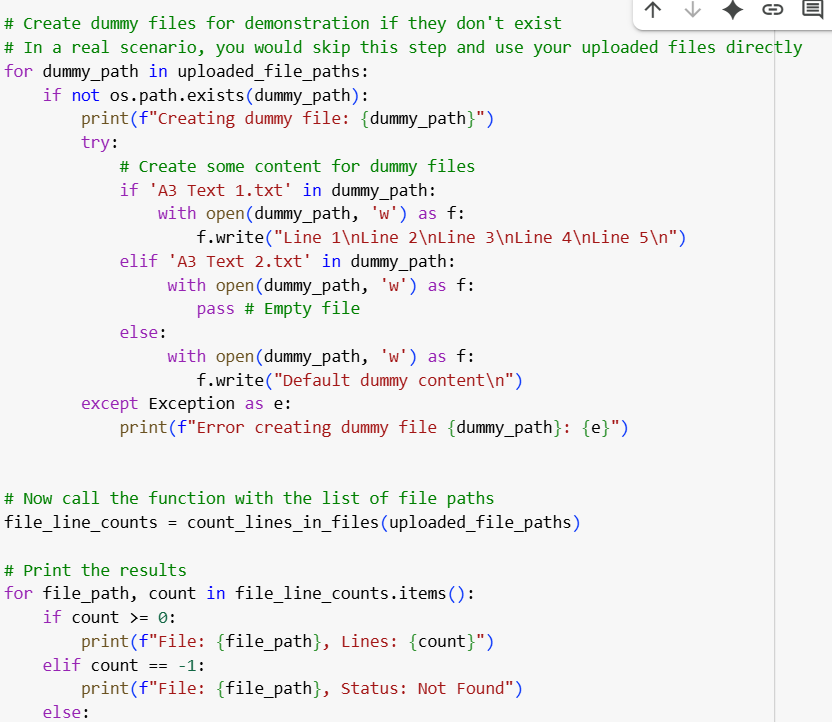
**TASK 5 :** Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines

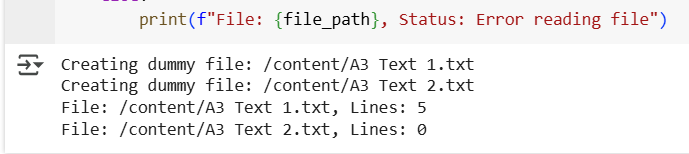
PROMPT : Write a Python function that takes the path to a .txt file and returns the number of lines in the file.Use the examples: count\_lines(r"D:\pavani 2-1\AI Assisted Coding\AI assigments\A3 Text 1.txt") # 5 count\_lines(r"D:\pavani 2-1\AI Assisted Coding\AI assigments\A3 Text 2.txt") # 0

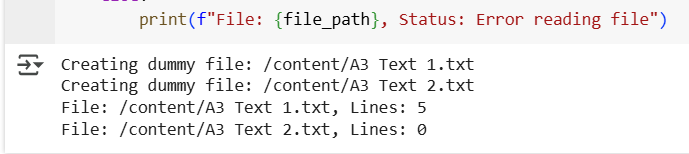
**CODE & OUTPUT :**











**EXPLANATION :**

This Python function, count\_lines\_in\_files, is designed to count the lines in a list of text files.

1. **It takes a list of file paths** as input.
2. **It processes each file path:** For every file path in the list, the function attempts to open and read the file.
3. **It counts lines efficiently:** Using a generator expression and the sum() function, it counts the number of lines within each file.
4. **It handles errors:** The function includes error handling to catch cases where a file is not found or other issues occur during reading.
5. **It stores results:** The count for each file (or an error indicator) is stored in a dictionary, where the file path is the key and the count is the value.
6. **It returns the results:** Finally, the dictionary containing all the file paths and their line counts is returned.