# AI ASSISTED CODING

LAB ASSIGNMENT – 4.3
SATHWIK VEMULAWADA
2403A53002
24BTCAICYB01-2-1

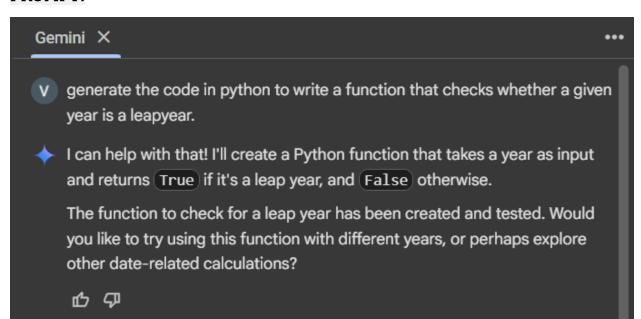
# Task Description#1

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

# Expected Output#1

• AI-generated function with no examples provided

### PROMPT:



### CODE:

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

Args:
    year: The year to check.

Returns:
    True if the year is a leap year, False otherwise.
    """
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

# Example usage:
    print(f"Is 2000 a leap year? {is_leap_year(2000)}")
    print(f"Is 1900 a leap year? {is_leap_year(1900)}")
    print(f"Is 2024 a leap year? {is_leap_year(2024)}")
    print(f"Is 2023 a leap year? {is_leap_year(2023)}")
```

#### **OUTPUT:**

```
Is 2000 a leap year? True
Is 1900 a leap year? False
Is 2024 a leap year? True
Is 2023 a leap year? False
```

## Task Description#2

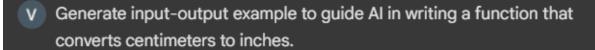
• One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

## **Expected Output#2**

• Function with correct conversion logic.

### PROMPT:

### Gemini X



Sure, I can provide an input-output example for a function that converts centimeters to inches. Here's an example:

Input: cm\_to\_inches(25.4)

Expected Output: 10.0

This example shows that if the input is 25.4 centimeters, the expected output is 10.0 inches (since 1 inch equals 2.54 centimeters).

Let me know if you'd like to see another example or want me to generate the code for this conversion!

### **CODE AND OUTPUT:**

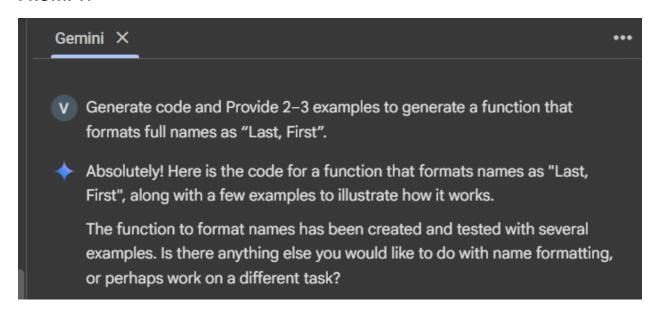
### **Task Description#3**

• 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 AND OUTPUT:**

```
→ C 25 colab.research.google.com/drive/1ak5ZioghyubCs-VaaowS6rN2WM1KFqjW#scrollTo=73b50ddb
↑ ↓ ♦ © 🗏 💠 🗓 🔟 ᠄
def format_name(full_name):
Q
             Formats a full name string as "Last, First".
               full_name: A string containing the full name (e.g., "John Doe").
೦ಾ
names = full_name.split()
               return f"{names[-1]}, {' '.join(names[:-1])}"
               return full_name
       Here are a few examples of how the format_name function works:
          1. Example 1: Standard two-part name Input: "John Doe" Output: "Doe, John" This shows a typical case where the last name is
            separated by a space from the first name.
          2. Example 2: Name with a middle name Input: "Mary Jane Smith" Output: "Smith, Mary Jane" This demonstrates how the function
            handles names with more than two parts, treating the last part as the last name and the rest as the first name(s).
          3. Example 3: Single name Input: "Alice" Output: "Alice" This shows how the function handles cases where the input is just a single
            name, returning the original input as formatting is not applicable in the "Last, First" style.
    √ [4] # Example usage:
           print(f"'{format_name('John Doe')}'")
           print(f"'{format_name('Mary Jane Smith')}'")
           print(f"'{format_name('Alice')}'")
            'Smith, Mary Jane'
            'Alice'
```

# Task Description#4

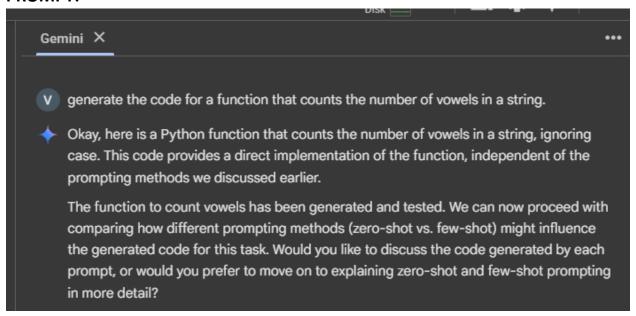
 Compare zero-shot and few-shot prompts for writing a function that counts the

number of vowels in a string

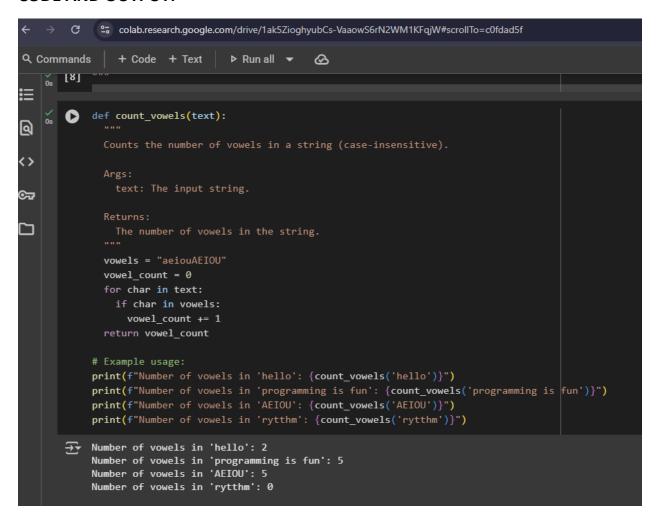
# **Expected Output#4**

• Functional output and comparative reflection.

#### **PROMPT:**



#### **CODE AND OUTPUT:**



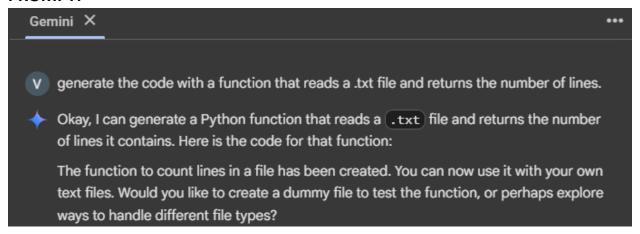
# **Task Description#5**

• 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 Al-guided logic.

### **PROMPT:**



### CODE:

```
colab.research.google.com/drive/1ak5ZioghyubCs-VaaowS6rN2WM1KFqjW#scrollTo=bbb0f169
&
Ħ
       0
           def count_lines_in_file(filepath):
Q
             Reads a text file and returns the number of lines.
<>
               filepath: The path to the text file.
☞
             Returns:
\Box
             try:
               with open(filepath, 'r') as f:
                 line_count = sum(1 for line in f)
               return line_count
             except FileNotFoundError:
               print(f"Error: File not found at {filepath}")
               return -1
           # Example usage (assuming you have a file named 'example.txt' in the same directory):
            # You might need to create a dummy file for testing
            # with open('example.txt', 'w') as f:
              f.write("Line 1\n")
              f.write("Line 2\n")
              f.write("Line 3\n")
           # file_path = 'example.txt'
            # num_lines = count_lines_in_file(file_path)
           # if num_lines != -1:
            # print(f"The file '{file_path}' has {num_lines} lines.")
```