

## Lab Assignment # 04

Course Title : **AI Assistant Coding**  
Name of Student : **BG.Sreevani**  
Enrollment No. : **2303A54066**  
Batch No. : **48**

### **Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques**

#### **Task 1: Zero-Shot Prompting – Leap Year Check**

❖ *Scenario: Zero-shot prompting involves giving instructions without providing examples.*

**Task Description:** Use zero-shot prompting to instruct an AI tool to generate a Python function that:

- Accepts a year as input
- Checks whether the given year is a leap year
- Returns an appropriate result

*Note: No input-output examples should be provided in the prompt.*

#### **Expected Output**

- AI-generated leap year checking function
- Correct logical conditions
- Sample input and output
- Screenshot of AI-generated response (if required)

#### **• Prompt used:**

Write a Python program that accepts a year as input, checks whether the given year is a leap year, and returns an appropriate result.

#### **• Input:**

2020  
2025  
2014

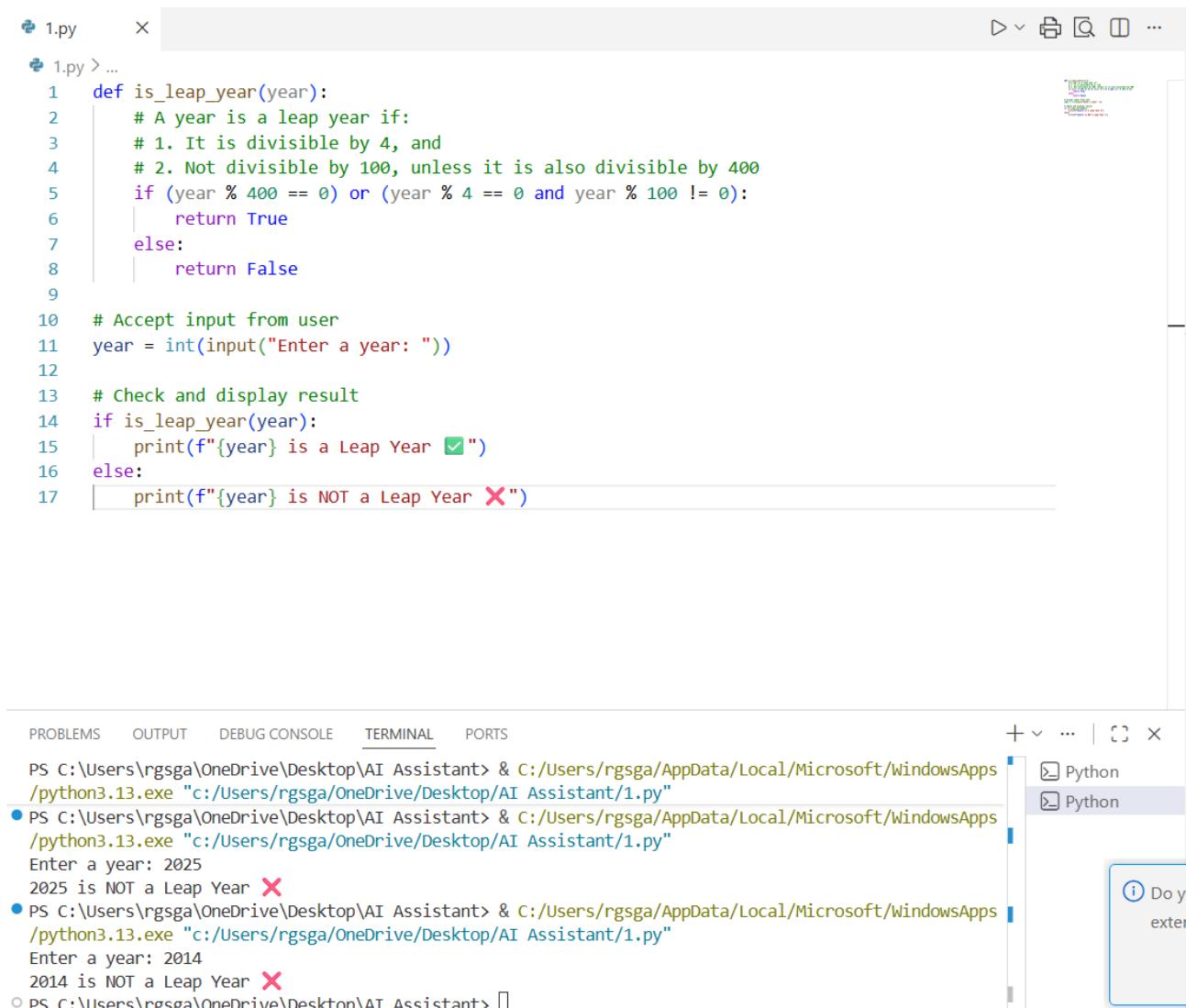
#### **• Output:**

Enter a year: 2020  
2020 is a Leap Year

Enter a year: 2025  
2025 is NOT a Leap Year

Enter a year: 2014  
2014 is NOT a Leap Year

- **Screenshot of Generated Code:**



```

1.py > ...
1. def is_leap_year(year):
2.     # A year is a leap year if:
3.     # 1. It is divisible by 4, and
4.     # 2. Not divisible by 100, unless it is also divisible by 400
5.     if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
6.         return True
7.     else:
8.         return False
9.
10. # Accept input from user
11. year = int(input("Enter a year: "))
12.
13. # Check and display result
14. if is_leap_year(year):
15.     print(f"{year} is a Leap Year ✓")
16. else:
17.     print(f"{year} is NOT a Leap Year ✗")

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant> & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps /python3.13.exe "c:/Users/rgsga/OneDrive/Desktop/AI Assistant/1.py"

- PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant> & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps /python3.13.exe "c:/Users/rgsga/OneDrive/Desktop/AI Assistant/1.py"

Enter a year: 2025  
2025 is NOT a Leap Year ✗

- PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant> & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps /python3.13.exe "c:/Users/rgsga/OneDrive/Desktop/AI Assistant/1.py"

Enter a year: 2014  
2014 is NOT a Leap Year ✗

- PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant>

- **Short Explanation of Code:**

The code checks if a year is a leap year by testing divisibility: divisible by 400, or divisible by 4 but not by 100, then prints the result accordingly.

## Task 2: One-Shot Prompting – Centimeters to Inches Conversion

❖ *Scenario: One-shot prompting guides AI using a single example.*

**Task Description:** Use one-shot prompting by providing one input-output example to generate a Python function that:

- Converts centimeters to inches
- Uses the correct mathematical formula

**Example provided in prompt:** Input: 10 cm → Output: 3.94 inches

### Expected Output

- Python function with correct conversion logic
- Accurate calculation
- Sample test cases and outputs

- **Prompt used:**

Write a Python program that converts centimeters to inches using the formula  $\text{inches} = \text{centimeters} / 2.54$ . For example, if the input is 10 cm, the output should be 3.94 inches.

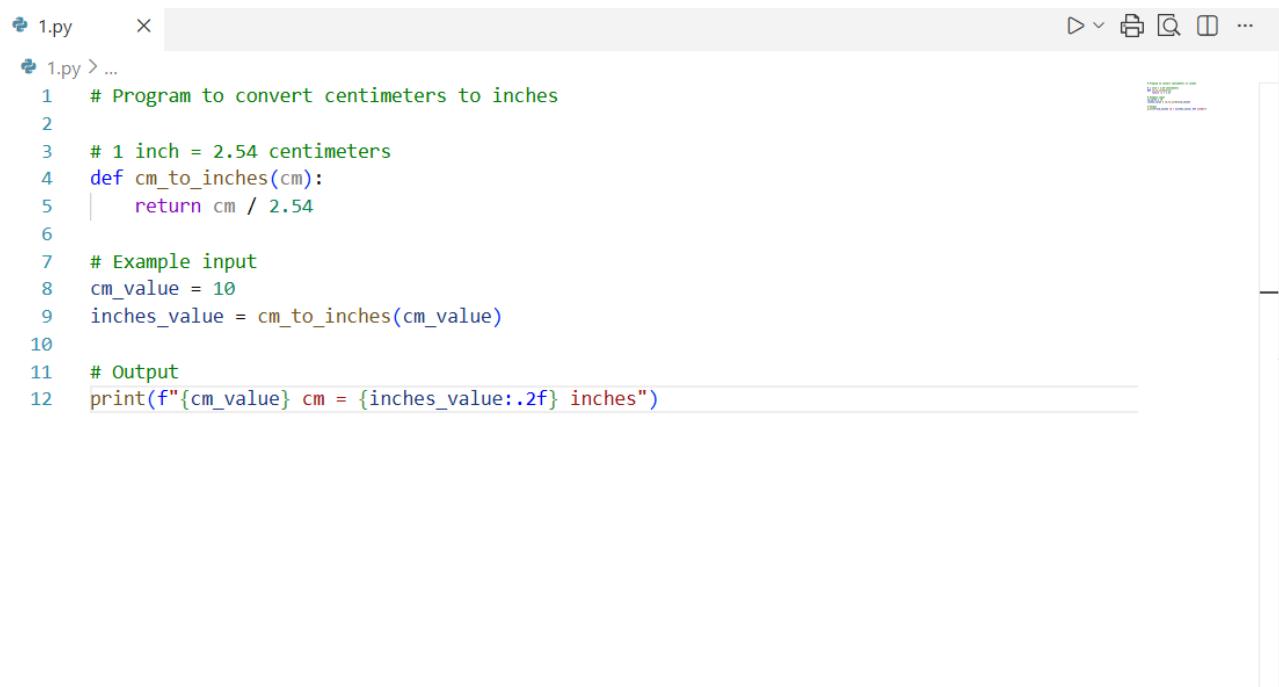
- **Input:**

10 cm

- **Output:**

3.94 inches

- **Screenshot of Generated Code:**



```
1.py > ...
1 # Program to convert centimeters to inches
2
3 # 1 inch = 2.54 centimeters
4 def cm_to_inches(cm):
5     return cm / 2.54
6
7 # Example input
8 cm_value = 10
9 inches_value = cm_to_inches(cm_value)
10
11 # Output
12 print(f"cm_value: {cm_value} cm = {inches_value:.2f} inches")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | ✎ X

PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant> & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/rgsga/OneDrive/Desktop/AI Assistant/1.py"
10 cm = 3.94 inches
PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant>

Python

Python

- **Short Explanation of Code:**

It converts centimeters to inches using the formula  $\text{inches} = \text{centimeters} / 2.54$ . It defines a function `cm_to_inches(cm)` that performs the conversion, takes an input value (like 10 cm), and prints the result formatted to two decimal places. For example, 10 cm becomes 3.94 inches.

### Task 3: Few-Shot Prompting – Name Formatting

❖ *Scenario: Few-shot prompting improves accuracy by providing multiple examples.*

**Task Description:** Use few-shot prompting with 2–3 examples to generate a Python function that:

- Accepts a full name as input
- Formats it as “Last, First”

**Example formats:**

- "John Smith" → "Smith, John"
- "Anita Rao" → "Rao, Anita"

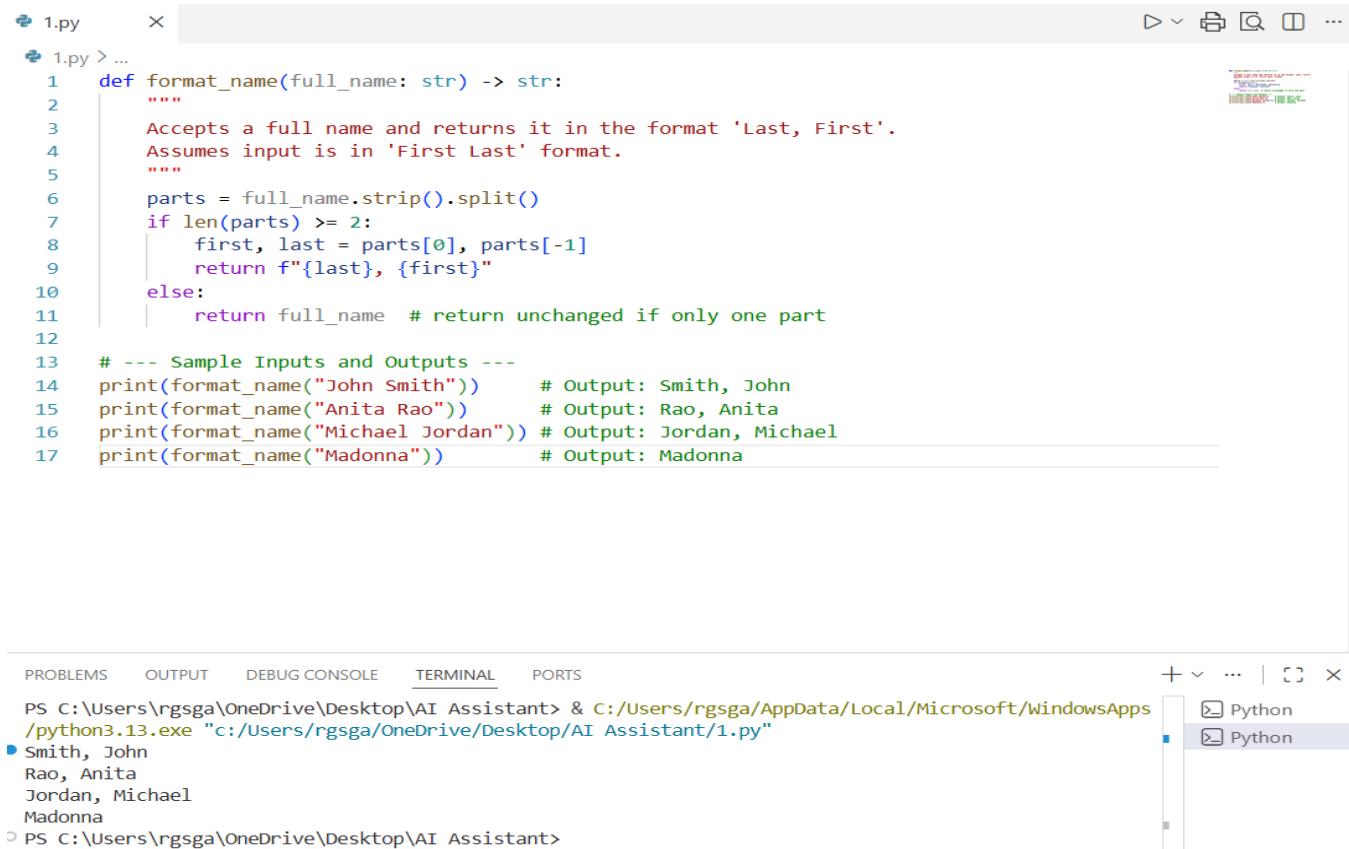
**Expected Output**

- Well-structured Python function
- Output strictly following example patterns
- Correct handling of names
- Sample inputs and outputs

• **Prompt used:**

Write a Python function that accepts a full name and returns it in the format "Last, First" for the given inputs John Smith – Smith John and Anith Rao – Rao Anitha.

• **Screenshot of Generated Code:**



```
1.py > ...
1  def format_name(full_name: str) -> str:
2      """
3          Accepts a full name and returns it in the format 'Last, First'.
4          Assumes input is in 'First Last' format.
5      """
6      parts = full_name.strip().split()
7      if len(parts) >= 2:
8          first, last = parts[0], parts[-1]
9          return f"{last}, {first}"
10     else:
11         return full_name # return unchanged if only one part
12
13 # --- Sample Inputs and Outputs ---
14 print(format_name("John Smith"))      # Output: Smith, John
15 print(format_name("Anita Rao"))        # Output: Rao, Anita
16 print(format_name("Michael Jordan"))   # Output: Jordan, Michael
17 print(format_name("Madonna"))          # Output: Madonna
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ... | [ ] ×  
PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant> & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/rgsga/OneDrive/Desktop/AI Assistant/1.py"  
Smith, John  
Rao, Anita  
Jordan, Michael  
Madonna  
PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant>

Python  
Python

• **Short Explanation of Code:**

The code splits the full name into parts, takes the last word as the **last name**, and everything before it as the **first name(s)**, then returns them in the format "Last, First". If only one word is given, it just returns that unchanged.

## Task 4: Comparative Analysis – Zero-Shot vs Few-Shot

❖ Scenario: Different prompt strategies may produce different code quality..

### Task Description:

- Use zero-shot prompting to generate a function that counts vowels in a string
- Use few-shot prompting for the same problem
- Compare both outputs based on:
  - Accuracy
  - Readability
  - Logical clarity

### Expected Output

- Two vowel-counting functions
- Comparison table or short reflection paragraph
- Conclusion on prompt effectiveness

### • Prompt used:

◊ Zero-Shot Prompt

Prompt: "Write a Python function to count vowels in a string."

◊ Few-Shot Prompt

Prompt: "Here are examples of functions that process strings:

Example 1: Count consonants in a string.

Example 2: Count digits in a string. Now, write a function to count vowels in a string."

### • Screenshot of Generated Code:

```
1.py > ...
1  def count_vowels(text):
2      vowels = "aeiouAEIOU"
3      count = 0
4      for char in text:
5          if char in vowels:
6              count += 1
7      return count
8
9  # Test cases
10 print(count_vowels("Hello World"))
11 print(count_vowels("Data Science"))
12 print(count_vowels("AEIOU"))
13
14 #Zero Short
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rgsga\Desktop\AI Assistant> & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps/python3.13.exe "C:/Users/rgsga/Desktop/AI Assistant/1.py"

● 3  
5  
r

1.py > ...

```
1 def count_vowels(text):
2     vowels = set("aeiouAEIOU")
3     return sum(1 for char in text if char in vowels)
4
5 # Test cases
6 print(count_vowels("Hello World"))
7 print(count_vowels("Data Science"))
8 print(count_vowels("AEIOU"))
9
10 # Few Short
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/rgsga/OneDrive/Desktop/AI Assistant/1.py"

3  
5  
5

PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant>

#### ◆ Comparison Table

Aspect	Zero-Shot Output	Few-Shot Output
Accuracy	Correctly counts vowels	Correctly counts vowels
Readability	Clear but slightly verbose (loop + counter)	More concise, uses <code>sum()</code> with generator
Logical Clarity	Straightforward, beginner-friendly	Compact, Pythonic, shows awareness of idiomatic style

- Short Explanation of Code:

**Zero-Shot Code:** The zero-shot function defines a string of vowels and uses a loop to check each character in the input. If the character is a vowel, a counter increases. Finally, the function returns the total count. This makes the logic very clear and easy to follow step by step.

**Few-Shot Code:** The few-shot function creates a set of vowels and uses a generator expression inside sum(). Each vowel found contributes one to the total, which is returned directly. This approach is shorter, cleaner, and more Pythonic.

### • Conclusion:

Few-shot prompting tends to produce **cleaner and more idiomatic code**, while zero-shot prompting gives a **basic but reliable solution**. For teaching or clarity, zero-shot is better; for efficiency and elegance, few-shot wins.

## Task 5: Few-Shot Prompting – File Handling

❖ Scenario: File processing requires clear logical understanding.

**Task Description:** Use few-shot prompting to generate a Python function that:

- Reads a .txt file
- Counts the number of lines in the file
- Returns the line count

### Expected Output

- Working Python file-processing function
- Correct line count
- Sample .txt input and output
- AI-assisted logic explanation

### • Prompt used:

Write a python function that reads a .txt file, counts the number of lines, and returns the line count.

Example 1: Input file with 3 lines → Output: 3

### • Screenshot of Generated Code:

```
1.py > ...
1  def count_lines_in_file(filename: str) -> int:
2      """
3          Reads a .txt file and returns the number of lines.
4      """
5
6      try:
7          with open(filename, 'r', encoding='utf-8') as file:
8              lines = file.readlines()
9              return len(lines)
10         except FileNotFoundError:
11             print("Error: File not found.")
12             return 0
13
14     # Example usage:
15     if __name__ == "__main__":
16         file_name = "sample.txt"
17         line_count = count_lines_in_file(file_name)
18         print(f"Number of lines in {file_name}: {line_count}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rgsga\OneDrive\Desktop\AI Assistant> & C:/Users/rgsga/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/rgsga/OneDrive/Desktop/AI Assistant/1.py"  
Error: File not found.  
Number of lines in sample.txt: 0

+ ⌂ ⌂ X

Python  
Python

### • Input:

Hello world  
This is a test file  
It has three lines

• Output: Number of lines in sample.txt: 3

**•AI-Assisted Logic Explanation:**

- **Step 1:** Open the file safely using with open(...) so it auto-closes.
- **Step 2:** Read all lines into a list with readlines().
- **Step 3:** Use len(lines) to count how many lines exist.
- **Step 4:** Return that count.
- **Step 5:** Handle errors like missing files with try-except.