AI Assisted Coding
Lab Assignment 4.3

M.Abhiram
2303A51667
Batch-18

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)

```
#use zero short prompting to instruct an AI tool to generate a python
function that :
#accepts year as input
#checks whether the given year is a leap year
#results an appropriate result
#No input-output examples should be provided in the prompt
def is_leap_year(year):
    """
    Check if the given year is a leap year.
    A year is a leap year if it is divisible by 4,
    except for end-of-century years, which must be divisible by 400.

    Args:
    year (int): The year to check.

    Returns:
```

```
    bool: True if the year is a leap year, False otherwise.
    """
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
# Example usage:
year = int(input())
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Input:
2010
Output:
2010 is not a leap year

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

```
#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
def cm_to_inches(cm):
    """
```

```
    Convert centimeters to inches.

    Args:
    cm (float): The length in centimeters.

    Returns:
    float: The length in inches.
    """
    inches = cm / 2.54
    return round(inches, 2)
# Example usage:
cm = float(input("Enter length in centimeters: "))
inches = cm_to_inches(cm)
print(f"{cm} cm is equal to {inches} inches.")
```

Input:
Enter length in centimeters: 10
Output:
10.0 cm is equal to 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

```
#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"
```

```python
def format_name(full_name):
    """
    Format a full name as "Last, First".

    Args:
    full_name (str): The full name in "First Last" format.

    Returns:
    str: The formatted name in "Last, First" format.
    """
    parts = full_name.split()
    if len(parts) >= 2:
        first_name = parts[0]
        last_name = parts[-1]
        return f"{last_name}, {first_name}"
    else:
        return full_name  # Return as is if not enough parts
# Example usage:
full_name = input("Enter full name (First Last): ")
formatted_name = format_name(full_name)
print(f"Formatted name: {formatted_name}")
```

Input:
Enter full name (First Last): John Smith
Output:
Formatted name: Smith, John

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:
o Accuracy
o Readability
o Logical clarity
Expected Output
• Two vowel-counting functions
• Comparison table or short reflection paragraph
• Conclusion on prompt effectiveness

```python
# 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
def count_vowels(s):
    """
    Count the number of vowels in a given string.

    Args:
    s (str): The input string.

    Returns:
    int: The count of vowels in the string.
    """
    vowels = 'aeiouAEIOU'
    count = sum(1 for char in s if char in vowels)
    return count
# Example usage:
input_string = input("Enter a string: ")
vowel_count = count_vowels(input_string)
print(f"Number of vowels in the string: {vowel_count}")
```

Input:
Enter a string: Mudumba Abhiram
Output:
Number of vowels in the string: 6

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

```
#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
def count_lines_in_file(file_path):
    """
    Count the number of lines in a text file.

    Args:
    file_path (str): The path to the text file.

    Returns:
    int: The number of lines in the file.
    """
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
            return len(lines)
    except FileNotFoundError:
        print("File not found.")
        return 0
# Example usage:
file_path = input("Enter the path to the text file: ")
line_count = count_lines_in_file(file_path)
```

Input:
Enter the path to the text file: C:\Users\Geervani\Documents\notes.txt
Output:
Number of lines in the file: 37