Assignment-4.3

Aditya Raj

2303A53031

Batch-46

Task 1: Zero-Shot Prompting – Leap Year

Check Scenario: Zero-shot prompting involves giving instructions without providing any 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

Prompt Used: Generate a Python function that accepts a year as input and checks whether the given year is a leap year.

AI-Generated Python Code:

```
def is_leap_year(year):
 if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
 return True
 else:
return False
```

Sample Input and Output:

Input: 2024 -> Output: True

Input: 2023 -> Output: False

Explanation:

A year is a leap year if it is divisible by 400 or divisible by 4 but not divisible by 100. Result:

The AI successfully generated a correct leap year checking function using zero-shot prompting

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

Scenario

One-shot prompting guides the AI by providing **one input–output example** along with the instruction. This example helps the AI understand the expected logic and output format.

Task Description

In this task, one-shot prompting is used to generate a Python function that:

- Converts centimeters to inches
- Uses the correct mathematical conversion formula

A single example is provided in the prompt to guide the AI.

Prompt Used (One-Shot Prompt)

Generate a Python function that converts centimeters to inches.
Example:
Input: 10 cm → Output: 3.94 inches

Code:

```
def cm_to_inches(cm):

   return cm * 0.394
```

Sample Input and Output

```
  print(cm_to_inches(10))

print(cm_to_inches(25))
```

Output:

```
3.94

9.85
```

Explanation

- The conversion formula used is:
  1 centimeter = 0.394 inches
- The function multiplies the given centimeter value by 0.394
- The provided example helps the AI select the correct formula and output precision

Result

Using one-shot prompting, the AI successfully generated an accurate Python function with correct conversion logic and precise output.

Task 3: Few-Shot Prompting – Name Formatting

Scenario

Few-shot prompting improves AI accuracy by providing **multiple input–output examples**. These examples help the AI clearly understand the expected pattern and formatting.

Task Description

In this task, few-shot prompting is used to generate a Python function that:

- Accepts a full name as input
- Formats the name in the form **"Last, First"**

Multiple examples are provided in the prompt to guide the AI.

Prompt Used (Few-Shot Prompt)

Generate a Python function that accepts a full name and formats it as "Last, First".
Examples:
"John Smith" → "Smith, John"
"Anita Rao" → "Rao, Anita"

Python Code

```
def format_name(full_name):

  first, last = full_name.split()

  return f"{last}, {first}"
```

Sample Input and Output

```
print(format_name("John Smith"))

print(format_name("Anita Rao"))
```

Output:

```
Smith, John

Rao, Anita
```

Explanation

- The input string containing the full name is split into first name and last name
- The function rearranges the order based on the given examples
- The output strictly follows the **"Last, First"** format demonstrated in the prompt

Result

Using few-shot prompting with multiple examples, the AI generated a well-structured Python function that correctly formats names according to the specified pattern.

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

Scenario

Different prompting strategies can produce variations in code quality, structure, and readability. This task compares **zero-shot** and **few-shot** prompting for the same problem.

Task Description

In this task:

- Zero-shot prompting is used to generate a Python function that counts vowels in a string
- Few-shot prompting is used for the same problem with example guidance
- The outputs are compared based on:
    - Accuracy
    - Readability
    - Logical clarity

Part A: Zero-Shot Prompting

Prompt Used (Zero-Shot Prompt)

Generate a Python function that counts the number of vowels in a string.

AI-Generated Python Code (Zero-Shot)

```
def count_vowels(text):
    vowels = "aeiouAEIOU"
    count = 0
    for ch in text:
        if ch in vowels:
            count += 1
    return count
```

Sample Input and Output

```
print(count_vowels("hello"))
```

Output:

```
2
```

Explanation

- The function iterates through each character in the string
- Checks if the character is a vowel
- Increments the counter for every vowel found

Part B: Few-Shot Prompting

Prompt Used (Few-Shot Prompt)

Generate a Python function to count vowels in a string.
Examples:
"hello" → 2
"AI Tools" → 4

AI-Generated Python Code (Few-Shot)

```
def count_vowels(text):

   return sum(1 for ch in text.lower() if ch in "aeiou")
```

Sample Input and Output

```
print(count_vowels("AI Tools"))
```

Output:

```
4
```

Explanation

- The string is converted to lowercase
- A generator expression efficiently counts vowel occurrences
- The code is concise and optimized

Comparison Table

Criteria	**Zero-Shot Prompting**	**Few-Shot Prompting**
Accuracy	Correct	Correct
Readability	Moderate	High
Logical Clarity	Step-by-step logic	Compact and clear
Code Length	Longer	Shorter

Conclusion

Both prompting techniques produce correct results. However, **few-shot prompting** results in more concise, readable, and optimized code. Providing examples helps the AI better understand expected behavior and improve output quality.

Task 5: Few-Shot Prompting – File Handling

Scenario

File handling operations require a clear understanding of program logic and structure. Few-shot prompting helps the AI generate accurate and reliable code by providing example-based guidance.

Task Description

In this task, few-shot prompting is used to generate a Python function that:

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

Examples are used in the prompt to guide the AI in understanding the expected behavior.

Prompt Used (Few-Shot Prompt)

Generate a Python function that reads a .txt file and counts the number of lines in it.
Examples:
File with 3 lines → Output: 3
File with 5 lines → Output: 5

AI-Generated Python Code

```
def count_lines(filename):
  with open(filename, 'r') as file:
    return len(file.readlines())
```

Sample Input File (sample.txt)

```
Welcome
to
AI
Lab
```

Sample Input and Output

```
print(count_lines("sample.txt"))
```

Output:

```
4
```

Explanation

- The file is opened in **read mode**
- readlines() reads all lines from the file into a list

- len() returns the total number of lines in the file
- The with statement ensures the file is properly closed after reading

Result

Using few-shot prompting, the AI successfully generated a working Python function for file handling that accurately counts the number of lines in a text file.