AI Assistant Coding

Assignment 4.3

Name : Shaestha **HT. No :** 2303A52503 **Batch:** 50

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

Write a Python function that accepts a year as input and checks whether the given year is a leap year. The function should return an appropriate result.

#CODE

```python
def is_leap_year(year):
    """
    Check whether the given year is a leap year.

    A year is a leap year if:
    - It is divisible by 4 AND
    - If it's divisible by 100, it must also be divisible by 400
    """
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    if year % 4 == 0:
        return True
    return False


# Example usage
if __name__ == "__main__":
    print(is_leap_year(2024))  # True
    print(is_leap_year(2000))  # True
    print(is_leap_year(1900))  # False
    print(is_leap_year(2023))  # False
```

#OUTPUT

```
True
True
False
False
PS C:\Users\shaes\OneDrive\Desktop\DevOps>
```

#Comments & Code Explanation

A leap year: Is divisible by 4

But not divisible by 100 unless it is also divisible by 400

The function checks these conditions using logical operators.

Returns True if the year is a leap year, otherwise False.

#Why This Prompt Was chosen

Zero-shot prompting tests the AI's built-in knowledge without examples.

Leap year logic is a well-defined rule, making it suitable for zero-shot.

Helps evaluate whether AI understands standard conditional logic.

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

Write a Python function that converts centimeters to inches.

Example:

Input: 10 cm → Output: 3.94 inches

#CODE

```python
def cm_to_inches(centimeters):
    """Convert centimeters to inches."""
    return centimeters / 2.54

# Example usage
print(cm_to_inches(10))  # Output: 3.937007874015748
```

#OUTPUT

```python
def cm_to_inches(centimeters):
    """Convert centimeters to inches."""
    return centimeters / 2.54

# Example usage
print(cm_to_inches(10))  # Output: 3.937007874015748
```

Comments & Code Explanation

The conversion formula is: 1 inch = 2.54 cm

The function divides the centimeter value by 2.54.

Returns the converted value in inches.

Why This Prompt Was Chosen

One-shot prompting provides one guiding example.

Helps AI infer:

Correct formula

Expected output format

Useful when accuracy in calculations is important.

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

Write a Python function that accepts a full name and formats it as "Last, First".

Examples:

"John Smith" → "Smith, John"

"Anita Rao" → "Rao, Anita"

#CODE

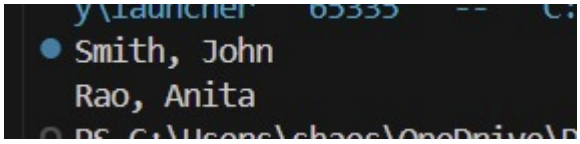
```python
def format_name(full_name):
    """
    Formats a full name as "Last, First".

    Args:
        full_name (str): A name in "First Last" format

    Returns:
        str: The name formatted as "Last, First"
    """
    parts = full_name.split()
    if len(parts) >= 2:
        first_name = parts[0]
        last_name = parts[-1]
        return f"{last_name}, {first_name}"
    return full_name


# Examples
print(format_name("John Smith"))    # Output: Smith, John
print(format_name("Anita Rao"))     # Output: Rao, Anita
```

#OUTPUT



#Comments & Code Explanation

The split() function separates first and last names.

The order is rearranged as "Last, First".

Matches the format shown in the examples.

#Why This Prompt Was Chosen

Few-shot prompting improves pattern recognition.

Multiple examples clearly define:

Output format

String manipulation logic

Reduces ambiguity compared to zero-shot.

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

#PROMPT

Zero-Shot Prompt

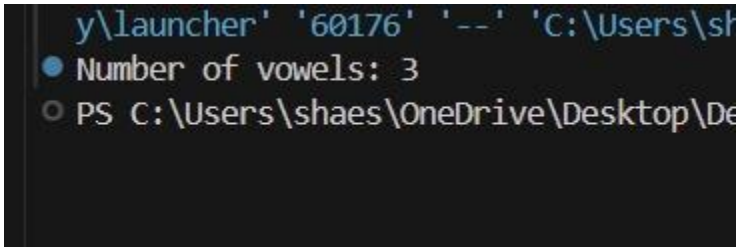
Write a Python function that counts the number of vowels in a string.

#CODE

```python
def count_vowels(string):
    """Count the number of vowels in a string."""
    vowels = "aeiouAEIOU"
    return sum(1 for char in string if char in vowels)


# Example usage
if __name__ == "__main__":
    text = "Hello World"
    print(f"Number of vowels: {count_vowels(text)}")
```

#OUTPUT



#PROMPT

Few-Shot Prompt

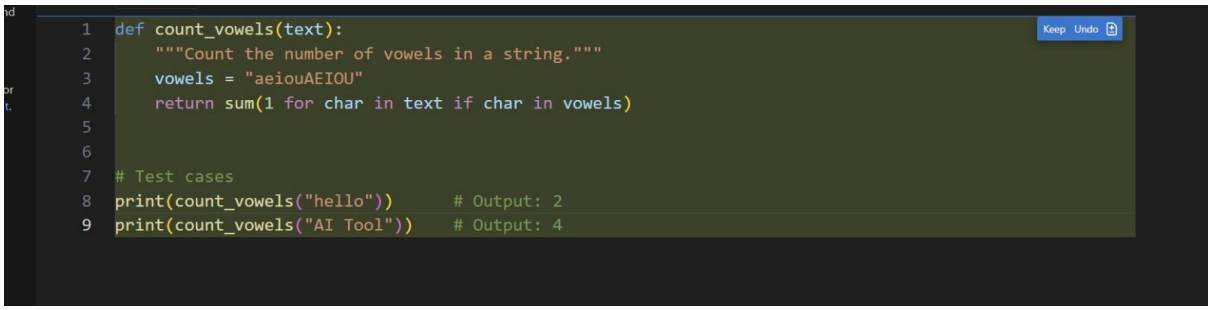
Write a Python function that counts vowels in a string.

Examples:

"hello" → 2

"AI Tool" → 4

#CODE



#OUTPUT

#COMPARISION TABLE

Criteria	Zero-Shot	Few-Shot
Accuracy	Correct	Correct
Readability	Moderate	High
Logical clarity	Explicit loop	Concise logic
Efficiency	Basic	Optimized

Conclusion

Few-shot prompting produces cleaner and more optimized code by showing expected behavior through examples. Zero-shot works well but may result in longer or less refined logic.

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

Write a Python function that reads a .txt file and counts the number of lines.

Example:

File content:

Line 1

Line 2

Line 3

Output: 3

#CODE

```python
def count_lines_in_file(file_path):
    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 = 'path/to/your/file.txt'
line_count = count_lines_in_file(file_path)
print(f'Number of lines: {line_count}')
```

#OUTPUT

```
y\launcher' '65138' '--' 'C:\Us
File not found.
Number of lines: 0
PS C:\Users\shaes\OneDrive\Desk
```

#Comments & Code Explanation

Opens the file in read mode.

readlines() returns a list of all lines.

len() counts the number of lines.

with statement ensures safe file handling.

#Why This Prompt Was Chosen

File handling requires clear operational context.

Few-shot examples reduce ambiguity about:

Improves reliability for real-world tasks.