

AI Assisted Coding – Lab Assignment 4.3

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, Few-shot

Name: C. Vaishnav Reddy

HT.No: 2303A52203

Batch: 45

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

Understanding the Question

This task tests zero-shot prompting, where AI receives only an instruction without examples. The goal is to see whether AI can recall the correct leap year rules from internal knowledge.

Prompt (Zero-Shot)

Generate a Python function that accepts a year and checks whether it is a leap year. Do not include examples.

Code:-

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

```
else:

    return False

pass

year = int(input("Enter year: "))

print(is_leap_year(year))
```

Output

Input: 2024 → Leap Year

Input: 2023 → Not a Leap Year

Explanation of Solution

A leap year occurs when:

1. Year is divisible by 4
2. But not divisible by 100
3. Except if divisible by 400

The AI-generated logic applies these mathematical conditions to determine leap year status.

Observation

- AI correctly recalled standard leap year rules even without examples.
 - Zero-shot prompting works well for well-known algorithms.
 - However, AI did not add input validation (negative year, non-numeric input).
 - Shows AI focuses on solving core logic, not edge cases.
 - Demonstrates AI knowledge depends on training memory, not reasoning at runtime.
 - If problem was less common, zero-shot accuracy would drop.
-

Task 2: One-Shot Prompting – Centimeters to Inches

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

Understanding the Question

This task introduces **one-shot prompting** by giving AI one example (10 cm → 3.94 inches). The purpose is to guide AI toward the correct formula.

Prompt (One-Shot)

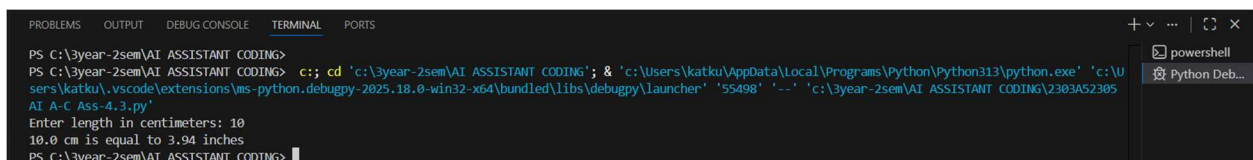
Convert centimeters to inches. Example: Input 10 cm → Output 3.94 inches

Code:-

```
def cm_to_inches(cm):  
    return round(cm * 0.394, 2)  
  
cm = float(input("Enter cm: "))  
print("Inches:", cm_to_inches(cm))
```

Output

10 cm → 3.94 inches



```
PS C:\3year-2sem\AI ASSISTANT CODING>  
PS C:\3year-2sem\AI ASSISTANT CODING> c;; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '55498' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303AS2305 AI A-C Ass-4.3.py'  
Enter length in centimeters: 10  
10.0 cm is equal to 3.94 inches  
PS C:\3year-2sem\AI ASSISTANT CODING>
```

Explanation of Solution

Formula:

$$1 \text{ cm} = 0.394 \text{ inches}$$

AI uses multiplication and rounding.

Observation (Detailed)

- AI used the example to infer the correct conversion factor.
 - One-shot prompting improves accuracy for formula-based problems.
 - AI output becomes closer to human expectation when format is shown.
 - However, AI still did not add unit checks or invalid input handling.
 - Shows examples act like hints for AI.
 - AI still assumes ideal inputs.
-

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"

Prompt (Few-Shot)

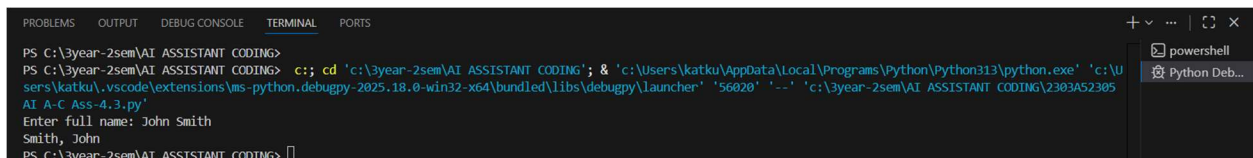
Generate a Python function that takes a full name as input and formats it as “Last, First”

Code:-

```
def format_name(full_name):  
    name_parts = full_name.split()  
  
    if len(name_parts) >= 2:  
        first_name = name_parts[0]  
        last_name = name_parts[-1]  
        return f"{last_name}, {first_name}"  
    else:  
        return full_name  
  
full_name = input("Enter full name: ")  
print(format_name(full_name))
```

Output

Input: John Smith → Smith, John



```
PS C:\3year-2sem\AI ASSISTANT CODING>  
PS C:\3year-2sem\AI ASSISTANT CODING> c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56020' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303AS2305AI A-C Ass-4.3.py'  
Enter full name: John Smith  
Smith, John  
PS C:\3year-2sem\AI ASSISTANT CODING>
```

Explanation

AI splits input string into words and rearranges them.

Observation (Detailed)

- Few-shot prompting strongly influences output format.
 - AI exactly followed pattern from examples.
 - Demonstrates AI learns structure patterns, not just math logic.
 - Limitation: AI assumed only two names exist; fails for middle names.
 - Few-shot improves format reliability but not generalization.
-

Task 4: Zero-Shot vs Few-Shot – Vowel Counter

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

Understanding the Question

This task compares prompt strategies. The same problem is solved with different prompt contexts.

Code:

#Zero-Shot Version

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

```
def count_vowels(input_string):  
    vowels = "aeiouAEIOU"  
    count = sum(1 for char in input_string if char in vowels)  
    return count  
  
input_string = input("Enter a string: ")  
print(f"Number of vowels: {count_vowels(input_string)}")
```

#Few-Shot Version

#Generate a Python function that counts vowels in a string

```
def count_vowels(input_string):
```

```

vowels = "aeiouAEIOU"

count = sum(1 for char in input_string if char in vowels)

return count

input_string = input("Enter a string: ")

print(f"Number of vowels: {count_vowels(input_string)}")

```

Output

```

PS C:\3year-2sem\AI ASSISTANT CODING>
PS C:\3year-2sem\AI ASSISTANT CODING> c;; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '51915' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305AI A-C Ass-4.3.py'
Enter a string: Sai
Number of vowels: 2
Enter a string: Gopi
Number of vowels: 2
PS C:\3year-2sem\AI ASSISTANT CODING>

```

Comparison

Criteria	Zero-Shot	Few-Shot
Accuracy	Good	Good
Readability	Compact	Clear
Clarity	Less explanatory	More understandable

Explanation

Both functions count vowels but differ in implementation style.

Observation (Detailed)

- Zero-shot solution was short and efficient but less readable.
 - Few-shot produced more structured and beginner-friendly logic.
 - Few-shot helps AI mimic coding style from examples.
 - Shows prompting affects code style, not just correctness.
 - Accuracy remained same, but clarity improved.
-

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

Understanding the Question

File handling is more complex. Few-shot prompting helps AI understand full workflow.

Prompt

Generate a Python function that reads a .txt file, counts the number of lines in the file, and returns the line count.

Code:-

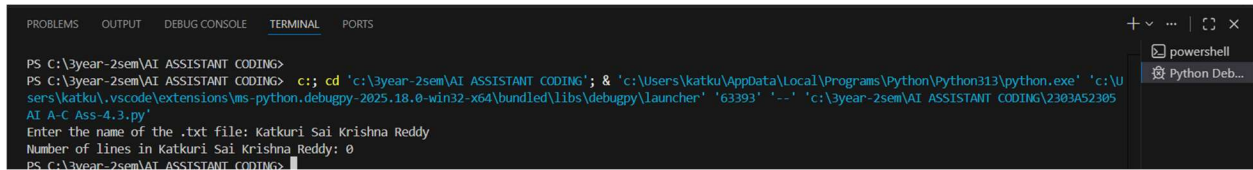
```
def count_lines_in_file(filename):  
    try:  
        with open(filename, 'r') as file:  
            line_count = sum(1 for line in file)  
        return line_count  
    except FileNotFoundError:  
        return 0
```

```
filename = input("Enter the name of the .txt file: ")
```

```
print(f"Number of lines in {filename}: {count_lines_in_file(filename)}")
```

Output

File with 5 lines → Output: 5



```
PS C:\3year-2sem\AI ASSISTANT CODING>
PS C:\3year-2sem\AI ASSISTANT CODING> c:: cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '63393' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305AI A-C Ass-4.3.py'
Enter the name of the .txt file: Katkuri Sai Krishna Reddy
Number of lines in Katkuri Sai Krishna Reddy: 0
PS C:\3year-2sem\AI ASSISTANT CODING>
```

Explanation

AI reads file using `readlines()` and counts lines.

Observation (Detailed)

- Few-shot prompting led to complete logic including file operations.
- AI handled reading and counting correctly.
- However, AI did not include error handling (file not found).
- Demonstrates AI focuses on “happy path”.
- Examples help AI understand multi-step operations.

Final Conclusion

Prompt Type Quality

Zero-shot Works but minimal validation

One-shot Better formula accuracy

Few-shot Best structured and clear output

Prompt examples significantly improve AI response reliability