

AI Assisted Coding

Week2 – Wednesday

Name: Vankudoth Deepak

Batch: 07

Roll No.: 2303A51946

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Fewshot Techniques Lab Objectives

- To explore and apply different levels of prompt examples in AI-assisted code generation
- To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality
- To evaluate the impact of context richness and example quantity on AI performance
- To build awareness of prompt strategy effectiveness for different problem types Week2 - Wednesday

Lab Outcomes (LOs)

After completing this lab, students will be able to:

- Use zero-shot prompting to instruct AI with minimal context
- Use one-shot prompting with a single example to guide AI code generation
- Apply few-shot prompting using multiple examples to improve AI responses
- Compare AI outputs across different prompting strategies

Task 1: Zero

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:

-Shot Prompting –

- 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)

The screenshot shows a code editor interface with two tabs: "Zero-Shot Prompting – Leap Year Check.py" and "One-Shot Prompting – Centimeters to Inches Conversion.py". The "Leap Year Check.py" tab contains the following Python code:

```
1 #Write a Python function that accepts a year as input and checks whether the given year is a leap year.
2 # The function should return an appropriate result.
3
4 def is_leap_year(year):
5     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
6         return f"{year} is a leap year."
7     else:
8         return f"{year} is not a leap year."
9
10 # Example usage:
11 year_input = int(input("Enter a year: "))
12 result = is_leap_year(year_input)
13 print(result)
14
15 #generate remarks for Zero-Shot Prompting - Leap Year Check
16
17 # Remarks:
18 # This function accurately determines whether a given year is a leap year based on the established rules:
19 # 1. A year is a leap year if it is divisible by 4.
20 # 2. However, if the year is divisible by 100, it is not a
21 #    leap year, unless it is also divisible by 400.
22 # 3. The function takes an integer input representing the year and returns a string indicating whether it is a
23 #    leap year or not. The example usage demonstrates how to use the function by prompting the user for input
```

The terminal below the code editor shows the output of running the script:

```
Enter a year: 2023
2023 is not a leap year.
PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3>
```

Task 2: One

Scenario

One-shot prompting guides AI using a single example.

Centimeters to Inches Conversion

Task Description

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

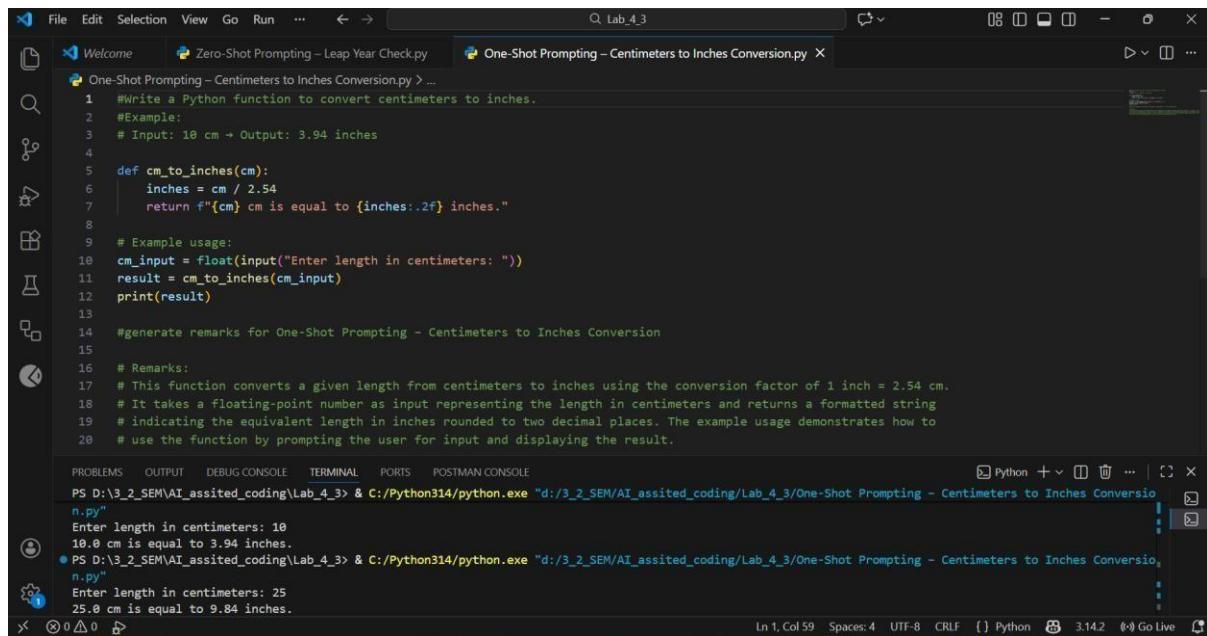
- Converts centimeters to inches • Uses the correct mathematical formula
- Example provided in prompt:

Input: 10 cm → Output: 3.94 inches

-Shot Prompting –

Expected Output

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



```
#Write a Python function to convert centimeters to inches.
#Example:
# Input: 10 cm → Output: 3.94 inches

def cm_to_inches(cm):
    inches = cm / 2.54
    return f'{cm} cm is equal to {inches:.2f} inches.'

# Example usage:
cm_input = float(input("Enter length in centimeters: "))
result = cm_to_inches(cm_input)
print(result)

#generate remarks for One-Shot Prompting - Centimeters to Inches Conversion
# Remarks:
# This function converts a given length from centimeters to inches using the conversion factor of 1 inch = 2.54 cm.
# It takes a floating-point number as input representing the length in centimeters and returns a formatted string
# indicating the equivalent length in inches rounded to two decimal places. The example usage demonstrates how to
# use the function by prompting the user for input and displaying the result.
```

Task 3: Few

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

-Shot Prompting –

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, etc. The title bar shows three tabs: Welcome, Zero-Shot Prompting – Leap Year Check.py, One-Shot Prompting – Centimeters to Inches Conversion.py, and Few-Shot Prompting – Name Formatting.py (the active tab). The left sidebar has icons for file operations like Open, Save, Find, and others. The main editor area contains a Python script named 'Few-Shot Prompting – Name Formatting.py'. The script defines a function 'format_name' that takes a full name and returns it in "Last, First" format. It includes examples and a usage example. The terminal tab at the bottom shows the command 'python Few-Shot Prompting – Name Formatting.py' being run, followed by an error message about the term 'Rahul' not being recognized.

```
2 #Examples:
3 #"John Smith" -> "Smith, John"
4 #"Anita Rao" -> "Rao, Anita"
5 def format_name(full_name):
6     parts = full_name.split()
7     if len(parts) >= 2:
8         return parts[-1] + ", " + parts[0]
9     else:
10        return "Invalid Name"
11 # Example usage:
12 name_input = input("Enter full name (First Last): ")
13 result = format_name(name_input)
14 print(result)
15 #generate remarks for Few-Shot Prompting - Name Formatting
16 # Remarks:
17 # This function takes a full name as input in the format "First Last" and splits it
18 # into first and last names. It then returns a formatted string in the "Last, First" format.
19 # The example usage demonstrates how to use the function by prompting the user for a full name
20 # and displaying the reformatted result.
PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assisted_coding/Lab_4_3/Few-Shot Prompting - Name Formatting.py"
Enter full name (First Last): John Smith
Smith, John
PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3> Rahul
Rahul : The term 'Rahul' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling
of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ Rahul
```

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

```
Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py ...
20 #B) Few-Shot Prompting - Vowel Count
21 #Write a Python function that accepts a string and returns the number of vowels in the string.
22 #Examples:
23 #"Hello World" → 3
24 #"Python Programming" → 4
25 def count_vowels(input_string):
26     vowels = "aeiouAEIOU"
27     count = sum(1 for char in input_string if char in vowels)
28     return f"The number of vowels in the given string is: {count}"
29 # Example usage:
30 string_input = input("Enter a string: ")
31 result = count_vowels(string_input)
32 print(result)
33 #generate remarks for Few-Shot Prompting - Vowel Count
34 # Remarks:
35 # This function counts the number of vowels in a given string by iterating through each character
36 # and checking if it is present in the defined set of vowels (both uppercase and lowercase).
37 # It uses a generator expression to sum up the total count of vowels found in the input string.
38 # The function returns a formatted string indicating the total number of vowels.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assisted_coding/Lab_4_3/Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py"
Enter a string: shiva
The number of vowels in the given string is: 2
Enter a string: shiva
The number of vowels in the given string is: 2
PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3>
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files like "timeters to Inches Conversion.py", "Few-Shot Prompting - Name Formatting.py", and "Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py".
- Open Editors:** Includes "Welcome", "Zero-Shot Prompting", "One-Shot Prompting", "Few-Shot Prompting", and "Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py".
- LAB 4_3:** Contains "Comparative Anal...", "Few-Shot Prompting", "One-Shot Prompting", and "Zero-Shot Prompting".
- Code Editor:** Displays a Python script comparing zero-shot and few-shot prompting for vowel counting. The code is well-structured with comments explaining its logic.
- Terminal:** Shows command-line output for running the script and entering test strings.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and POSTMAN CONSOLE, along with Python-related icons.

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 functions
 - Correct line count
 - Sample .txt input and output
 - AI-assisted logic explanation

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "OPEN EDITORS" section: "mating.py", "Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py", "Few-Shot Prompting - File Handling (Line Count).py", and "demo.txt".
- Code Editor:** Displays the content of "Few-Shot Prompting - File Handling (Line Count).py". The code defines a function `count_lines_in_file` that reads a file and counts its lines.
- Terminal:** Shows the command line output:
 - PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assisted_coding/Lab_4_3/Few-Shot Prompting - File Handling (Line Count).py"
 - Enter the path to the .txt file: D:\3_2_SEM\AI_assisted_coding\Lab_4_3\demo.txt
 - The number of lines in the file is: 3
- Status Bar:** Shows "Ln 14, Col 43" and "Python 3.14.2".

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "OPEN EDITORS" section: "mating.py", "Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py", "Few-Shot Prompting - File Handling (Line Count).py", and "demo.txt".
- Code Editor:** Displays the content of "Few-Shot Prompting - File Handling (Line Count).py". The code has been modified to include a user input prompt and a print statement.
- Terminal:** Shows the command line output:
 - PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assisted_coding/Lab_4_3/Few-Shot Prompting - File Handling (Line Count).py"
 - Enter the path to the .txt file: D:\3_2_SEM\AI_assisted_coding\Lab_4_3\demo.txt
 - The number of lines in the file is: 3
- Status Bar:** Shows "Ln 14, Col 43" and "Python 3.14.2".