

AI Assistant Coding

Assignment 4.3

Name :Sahasra HT. No : 2303A52499 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

```

1 def is_leap_year(year):
2     """
3     Check whether the given year is a leap year.
4
5     A year is a leap year if:
6     - It is divisible by 4 AND
7     - If it's divisible by 100, it must also be divisible by 400
8     """
9     if year % 400 == 0:
10        return True
11    if year % 100 == 0:
12        return False
13    if year % 4 == 0:
14        return True
15    return False
16
17
18 # Example usage
19 if __name__ == "__main__":
20     print(is_leap_year(2024)) # True
21     print(is_leap_year(2000)) # True
22     print(is_leap_year(1900)) # False
23     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

```
1 def cm_to_inches(centimeters):
2     """Convert centimeters to inches."""
3     return centimeters / 2.54
4
5 # Example usage
6 print(cm_to_inches(10)) # Output: 3.937007874015748
```

#OUTPUT

```
1 def cm_to_inches(centimeters):  
2     """Convert centimeters to inches."""  
3     return centimeters / 2.54  
4  
5 # Example usage  
6 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"

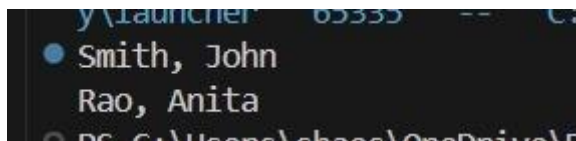
#CODE

```

1 def format_name(full_name):
2     """
3     Formats a full name as "Last, First".
4
5     Args:
6         full_name (str): A name in "First Last" format
7
8     Returns:
9         str: The name formatted as "Last, First"
10    """
11    parts = full_name.split()
12    if len(parts) >= 2:
13        first_name = parts[0]
14        last_name = parts[-1]
15        return f"{last_name}, {first_name}"
16    return full_name
17
18
19 # Examples
20 print(format_name("John Smith"))    # Output: Smith, John
21 print(format_name("Anita Rao"))    # Output: Rao, Anita

```

#OUTPUT



```

• Smith, John
  Rao, Anita

```

#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

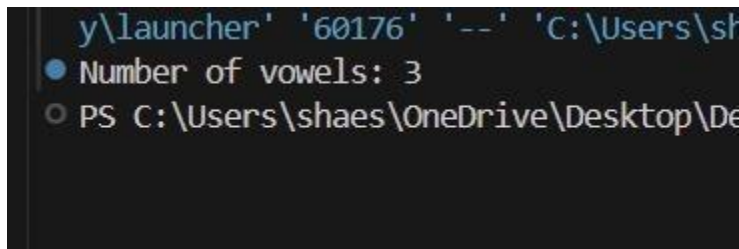
Zero-Shot Prompt

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

#CODE

```
1 def count_vowels(string):
2     """Count the number of vowels in a string."""
3     vowels = "aeiouAEIOU"
4     return sum(1 for char in string if char in vowels)
5
6
7 # Example usage
8 if __name__ == "__main__":
9     text = "Hello World"
10    print(f"Number of vowels: {count_vowels(text)}")
```

#OUTPUT

A terminal window showing a command prompt. The prompt is 'y\launcher' followed by a space and a quote, then '60176' followed by a space and a quote, then '--' followed by a space and a quote, then 'C:\Users\sh'. The output shows 'Number of vowels: 3' and 'PS C:\Users\shaes\OneDrive\Desktop\De'.

#PROMPT

Few-Shot Prompt

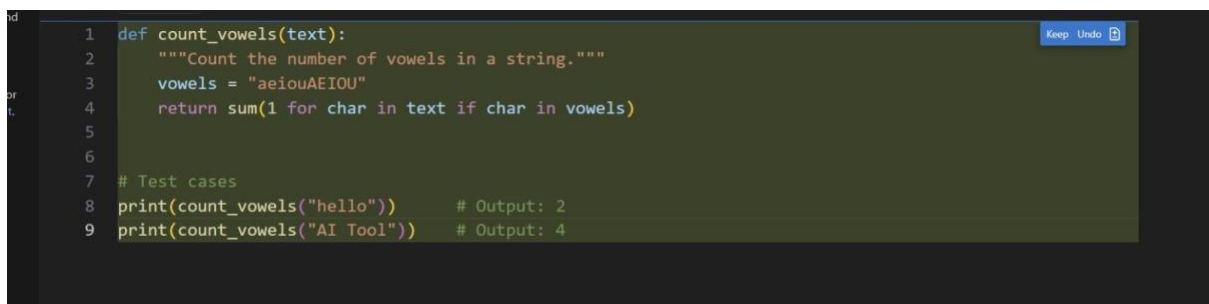
Write a Python function that counts vowels in a string.

Examples:

"hello" → 2

"AI Tool" → 4

#CODE

A code editor showing a Python function 'count_vowels(text)'. The function has a docstring 'Count the number of vowels in a string.', a list of vowels 'aeiouAEIOU', and a return statement 'return sum(1 for char in text if char in vowels)'. Below the function, there are test cases: 'print(count_vowels("hello"))' with a comment '# Output: 2' and 'print(count_vowels("AI Tool"))' with a comment '# Output: 4'.

#OUTPUT

A terminal window showing the output of the Python function. The prompt is 'y\launcher' followed by a space and a quote, then '2' followed by a space and a quote, then '4' followed by a space and a quote.

#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

```
1 def count_lines_in_file(file_path):
2     try:
3         with open(file_path, 'r') as file:
4             lines = file.readlines()
5             return len(lines)
6     except FileNotFoundError:
7         print("File not found.")
8         return 0
9
10 # Example usage
11 file_path = 'path/to/your/file.txt'
12 line_count = count_lines_in_file(file_path)
13 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.