

AI Assisted Coding

Assignment -4.3

Name :B.SUSHMA

RollNo:2303A54040

Batch:48

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.

Prompt used:#Write a Python function that takes a year as input and checks whether it is a leap year.

#The function should return an appropriate message indicating whether the year is a leap year or not.

#Do not include any input-output examples.

- **Generated code:**

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

- **Sample Input:**

Enter year: 2024

- **Sample Output:**

True

- **Short Explanation of Logic:**

the program checks the leap year conditions using logical operators . a year is a leap year if the it is divisible by 400 or divisible by 4 but not divisible by 100 . the logic is implemented directly without providing any prior .

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files: assignment 4.3.py, task1.py, assignment1st, assignment3.1.py, task 2.py, task2.py, task3.py, task4.py, task5.py.
- CODE EDITOR**: File: task1.py. Content:

```

assignment 4.3.py > task1.py > ...
1  #Write a python function that takes a year as input and checks whether it is a leap year.
2  #The function should return an appropriate message indicating whether the year is a leap year
3  #Do not include any input-output examples.
4
5  def is_leap_year(year):
6      if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
7          return True
8      else:
9          return False
10
11  year = int(input("Enter year: "))
12  print(is_leap_year(year))
13
14

```
- CHAT**: Task: LEAP YEAR CHECK FUNCTION IN PYTHON
 - The function should take year as input
 - Return whether it is a leap year or not
 - Do not provide any examples
- PROBLEMS**: No errors.
- OUTPUT**: Terminal output:

```

PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & 'c:\Users\bindu\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\bindu\vscode\extensions\ms-python.debugger-2025.19.2026011901\win32-x64\bundle\libs\debugpy\launcher' '51279' '--'
Enter year: 2024
True
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> assignment 4.3.py\task1.py

```
- TERMINAL**: Indexing completed.

Task2: One-Shot Prompting – Centimeters to Inches Conversion

❖ Scenario: One-shot prompting guides AI using a single example.

- **Prompt used:**
- # Write a Python function to convert centimeters to inches
- # Use the formula inches = centimeters / 2.54
- # Example:
- # Input: 10 cm
- # Output: 3.94 inches

• Generated code:

```

def cm_to_inches(centimeters):
    """
    Convert centimeters to inches.

    Args:
        centimeters: A number representing length in centimeters

    Returns:
        A float representing the length in inches
    """
    inches=centimeters/2.54
    return round(inches, 2)

# Example usage
if __name__=="__main__":
    cm_value=10
    result=cm_to_inches(cm_value)
    print(f"Input: {cm_value} cm")
    print(f"Output: {result} inches")

```

- **SampleInput:**
 - Enter value in cm: 25.4
 - **SampleOutput:**
- 10.0 inches

ShortExplanationofLogic:

The function takes centimeters as input and converts it into inches using the standard formula Cm % 2.54 one example in the prompt helps the ai to generate the logic accurately in one short

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows files in the workspace: assignment43.py, task1.py, task2.py, assignment1st, assignment3.1.py, task2.py, task3.py, task4.py, task5.py.
- CODE EDITOR:** Task2.py file open, containing the following Python code:

```

assignment43.py > task2.py > ...
1 def cm_to_inches(cm):
2     """
3         Convert centimeters to inches.
4     """
5     Args:
6         cm: A number representing length in centimeters
7     Returns:
8         A float representing the length in inches
9     """
10    inches = cm / 2.54
11    return round(inches, 2)
12
13
14
15 # Example usage
16 if __name__ == "__main__":
17     cm_value = 10
18     result = cm_to_inches(cm_value)
19     print(f"Input: {cm_value} cm")
20     print(f"Output: {result} inches")

```
- TERMINAL:** Shows the output of running the script:

```

PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment43.py/task2.py"
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> C
● PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment43.py/task2.py"
● Input: 10 cm
Output: 3.94 inches
○ PS C:\Users\bindu\OneDrive\Desktop\AI CODING>

```
- STATUS BAR:** Shows indexing completed, file statistics (0 errors, 0 warnings), and file encoding (UTF-8).

Task3: Few-shot prompting-Name Formatting

Scenario:

Few-shot prompting improves accuracy by providing multiple examples.



- **Prompt used:**

- # Write a Python function to format names as "Last, First"
- # Examples:
- # "John Smith" -> "Smith, John"
- # "Anita Rao" -> "Rao, Anita"

- **Generated code:**

```
"""  
  
def format_name(full_name):  
    """  
    Format a full name as "Last, First"  
  
    Args:  
        full_name (str): A full 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  
  
  
# Test examples  
print(format_name("John Smith"))      # Smith, John  
print(format_name("Anita Rao"))       # Rao, Anita
```

- **SampleInput:**

Enter full name: John Smith

- **SampleOutput:Smith, John**

ShortExplanationofLogic: The function splits the full name into first and last names and rearranges them in the required format it providing multiple examples improves few short

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files in the 'AI CODING' folder: assignment 4.3.py, task1.py, task2.py, task3.py, task4.py, task5.py, assignment1st.py, assignment3.1.py.
- Code Editor:** Task3.py file open, containing Python code for a 'format_name' function. The code uses string splitting and concatenation to reverse the order of names. It includes test cases for "John Smith" and "Anita Rao".
- Terminal:** Shows the command run in the terminal: PS C:\Users\bbindu\OneDrive\Desktop\AI CODING> & C:/Users/bbindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bbindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task3.py". The output shows the formatted names: "Smith, John" and "Rao, Anita".
- Status Bar:** Shows indexing completed and other system information.

Task4:Comparative Analysis – Zero-Shot vs Few-Shot

- **Scenario:**
Different prompt strategies may produce different code quality.
- **Prompt1:Zero-Shot Prompting**

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

The function should return the total count.

Do not provide any examples.

Generatedcode:

```
def count_vowels(string):
    vowels = "aeiouAEIOU"
    count = 0
    for char in string:
        if char in vowels:
```

- **SampleInput:**
Enter string: Hello World
- **SampleOutput:**

Number of vowels: 3

- **Prompt2: Few-Shot Prompting**

- Write a Python function to count vowels in a string.
- Examples:
- Input: "hello" → Output: 2
- Input: "AI Assisted Coding" → Output: 7

❖ **Generated code:**

```

❖ 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)
❖

❖ # Test cases
❖ print(count_vowels("hello")) # Output: 2
❖ print(count_vowels("AI Assisted Coding")) # Output: 7

```

- **Sample Input:**

Enter string: Hello World

- **Sample Output: 3**

Explanation (Few-Shot);

The function uses a predefined vowel set and python's sum with the generator expression to count vowels efficiently the logic is compact and easier to understand due to example .

The screenshot shows the Visual Studio Code interface. The code editor displays the following Python code in a file named task401.py:

```

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)

```

The file explorer on the left shows a folder structure under 'AI CODING' containing files like assignment 4.3.py, task1.py, task2.py, task3.py, and task401.py. The terminal at the bottom shows command-line history related to the assignment.

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows files in the "AI CODING" folder, including assignment 4.3.py, task1.py, task2.py, task3.py, task42.py, task401.py, and task401.py.
- Editor:** Task42.py is open, displaying Python code to count vowels in a string.
- Terminal:** Shows the output of running Python scripts assignment 4.3.py and task401.py.
- Bottom Status Bar:** Displays indexing completed, master*, Ln 9, Col 55, Spaces: 4, UTF-8, Python, and 3.14.2.

Comparison Table

| Criteria | Zero-Shot | Few-Shot |
|-----------------|-----------|------------|
| Accuracy | Correct | Correct |
| Readability | Medium | High |
| Logical Clarity | Simple | Very clear |
| Code Length | Longer | Shorter |

Task5: Few-Shot Prompting – File Handling (Line Count)

Scenario: You are building a text file analyzer that counts the number of lines in a file.

Clear examples help ensure correct file handling.

Prompt used:

```
# Write a Python function to read a text file
# The function should take the file name as input
# Open the file in read mode# Count the total number of lines in the file
#Return the line count as an integer
# Example:
# If "file.txt" contains 3 lines, the output should be 3
```

Generated code:

```
def count_lines(filename):
    """
    Read a text file and count the total number of lines.

    Args:
        filename (str): The name of the file to read

    Returns:
        int: The total number of lines in the file
    """
    try:
        with open(filename, 'r') as file:
            line_count = sum(1 for line in file)
        return line_count
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return 0

# Example usage
if __name__ == "__main__":
    result = count_lines("file.txt")
    print(f"Total lines: {result}")
```

- **Sample Input:**
- `print(count_lines("file.txt"))`
- **Sample Output:**

3

Short Explanation of Logic:

The program reads a text file in read mode and counts the number of lines present in it. A counter variable is used to keep track of the number of lines. Each time a line is read from the file, the counter increases by one. After reading the entire file, the function returns the total number of lines. This method ensures accurate line counting and is easy to understand.

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

- File Explorer:** Shows files task1.py, task2.py, task3.py, task401.py, task42.py, and task5.py.
- Code Editor:** Displays a Python script named assignment 4.3.py. The code defines a function count_lines that reads a file and counts the total number of lines. It includes error handling for file not found.
- Terminal:** Shows the command line output of running the script task5.py. It indicates that the file 'file.txt' was not found and prints 'Total lines: 0'.
- Breakpoints:** A sidebar shows a list of breakpoints with checkboxes for Raised Exception, Uncaught Exception, and User Uncaught.

```
assignment 4.3.py > task5.py > ...
1  def count_lines(filename):
2  """
3      Read a text file and count the total number of lines.
4
5      Args:
6          filename (str): The name of the file to read
7
8      Returns:
9          int: The total number of lines in the file
10     """
11
12     try:
13         with open(filename, 'r') as file:
14             line_count = sum(1 for line in file)
15         return line_count
16     except FileNotFoundError:
17         print(f"Error: File '{filename}' not found.")
18         return 0
19
20     # Example usage
21 if __name__ == "__main__":
22     result = count_lines("file.txt")
23     print(f"Total lines: {result}")
```

PROBLEMS OUTPUT TERMINAL PORTS

DEBUG CONSOLE

TERMINAL

Filter (e.g. text, exclude, \escape)

PS C:\Users\bindu\Desktop\AI CODING> & c:/Users/bindu/AppData/Local/Python/pythoncore-3.14-04/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task5.py"
Error: File 'file.txt' not found.
Total lines: 0
PS C:\Users\bindu\Desktop\AI CODING>

Indexing completed.

