

2303A51504

Batch-25

Lab 4.3

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)

The screenshot shows the Visual Studio Code interface. In the center, there is a code editor window titled "lab 4.3.py" containing the following Python code:

```

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

# Main program
year = int(input("Enter a year: "))

if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")

```

Below the code editor is a terminal window showing the execution of the script:

```

PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/parva/OneDrive/Desktop/AI Assted/lab 4.3.py"
Enter a year: 206
206 is not a leap year.
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/parva/OneDrive/Desktop/AI Assted/lab 4.3.py"
Enter a year: 2006
2006 is not a leap year.
PS C:\Users\parva\OneDrive\Desktop\AI Assted>

```

To the right of the code editor is a "CHAT" pane with a message from AI Assted: "Generate a complete Python program that checks whether a given year is a leap year. The program must:". Below this is a list of requirements:

- Define a function that takes a year as input
- Use correct leap year conditions
- Take input from the user
- Print whether the year is a leap year or not
- Be directly runnable in VS Code

At the bottom right of the interface, there is a status bar showing "Ln 10, Col 41", "Spaces: 4", "UTF-8", "Python", "3.13.7", "9:30 AM", and "1/28/2026".

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

```

12
13 #task2
14 def cm_to_inches(cm):
15     inches = cm / 2.54
16     return inches
17
18
19 # Sample test cases
20 if __name__ == "__main__":
21     test_cases = [10, 25, 50, 100, 2.54]
22
23     for cm in test_cases:
24         inches = cm_to_inches(cm)
25         print(f"Input: {cm} cm")
26         print(f"Output: {inches:.2f} inches")
27         print()

```

AI Assisted

Write a Python function that converts centimeters to inches.

Example:

Input: 10 cm
Output: 3.94 inches

The function should:

- Accept centimeters as input
- Use the correct conversion formula
- Return the value in inches
- Include a few sample test cases with their outputs

Provide only the Python code.

Created new file for conversion function

```

def cm_to_inches(cm):
    """
    Convert centimeters to inches
    Args:
        cm: A numeric value in centimeters
    Returns:
        float: The equivalent value in inches
    """
    inches = cm / 2.54
    return inches

```

Sample test cases

If __name__ == "__main__":

2 files changed +44 -2

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

```

def format_name(full_name):
    parts = full_name.split()
    first_name = parts[0]
    last_name = parts[1]
    return f'{last_name}, {first_name}'

# Sample test cases
if __name__ == "__main__":
    names = ["John Smith", "Anita Rao", "Alice Johnson", "Bob Williams", "Emma Davis"]
    for name in names:
        formatted = format_name(name)
        print(f"Input: '{name}'")
        print(f"Output: '{formatted}'")
        print()

```

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

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

- File Explorer:** Shows files like `add.py`, `AI Assisted`, `ass 3.4.pdf`, `check_leap_year.py`, `lab 4.3.py`, `lab ass 3.4.py`, `lab assignment 3.3.pdf`, `lab assignment-1.4.pdf`, `lab assignment-2.3.pdf`, and `leap_year.py`.
- Code Editor:** The active file is `lab 4.3.py`. It contains Python code for counting vowels in a string. The code includes a docstring, argument description, and a detailed explanation of the logic.
- Terminal:** Shows command-line history related to the file.
- Output:** Shows the results of running the code with various inputs.
- Right Panel:** An AI-assisted panel titled "LEAP YEAR DETERMINATION FUNCTION..." provides a task description and generated code for a vowel counter.
- Bottom Status Bar:** Displays file changes (2 files changed), build status (Keep), and system information (Ln 48, Col 24, Spaces: 4, UTF-8, CRLF, Python, 9:39 AM, 1/28/2026).

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

The screenshot shows the Visual Studio Code (VS Code) interface. The main area displays a Python file named `lab 4.3.py` with the following code:

```
74 #task5
75 def count_lines_in_file(filename):
76     try:
77         with open(filename, 'r') as file:
78             line_count = sum(1 for line in file)
79         return line_count
80     except FileNotFoundError:
81         print(f"Error: File '{filename}' not found.")
82         return -1
83     except Exception as e:
84         print(f"Error reading file: {e}")
85         return -1
86
87 # Sample usage with test files
88 if __name__ == "__main__":
89
90     with open("sample1.txt", 'w') as f:
91         f.write("Hello\\nWorld")
92
93     with open("sample2.txt", 'w') as f:
94         f.write("Python\\nfile\\nhandling")
95
96     with open("sample3.txt", 'w') as f:
97         f.write("Line 1\\nline 2\\nline 3\\nline 4\\nline 5")
98
99     test_files = ["sample1.txt", "sample2.txt", "sample3.txt"]
100
101     for filename in test_files:
102         line_count = count_lines_in_file(filename)
103         print(f"Input file: {filename}")
104         print(f"Output: {line_count} lines")
105         print("\n")
```

The terminal at the bottom shows the execution of the script:

```
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/parva/OneDrive/Desktop\AI Assted\lab 4.3.py"
Input file: sample1.txt
Output: 2 lines

Input file: sample2.txt
Output: 3 lines

Input file: sample3.txt
Output: 5 lines
```

The right sidebar contains an AI-assisted task titled "LEAP YEAR DETERMINATION FUNCTION". It includes examples, input/output, and a code completion panel:

- Examples:
 - Input file contents: Hello\\nWorld
 - Output: 2
- Input file contents: Python\\nfile\\nhandling
- Output: 3
- The function should:
 - Accept a .txt file name as input.
 - Read the file safely.
 - Count the total number of lines.
 - Return the line count.
 - Also include sample input and output.
- Code completion panel:

```
def count_lines_in_file(filename):
    """
    Count the number of lines in a file.

    Args:
        filename: The name of the file to count lines in.

    Returns:
        int: The total number of lines in the file.
    """
```