

Lab_Assignment_4.3

Id:2303A51940

Task 1: Zero-Shot Prompting – Leap Year Check

Scenario

Zero-shot prompting involves giving instructions without providing examples.

Task Description

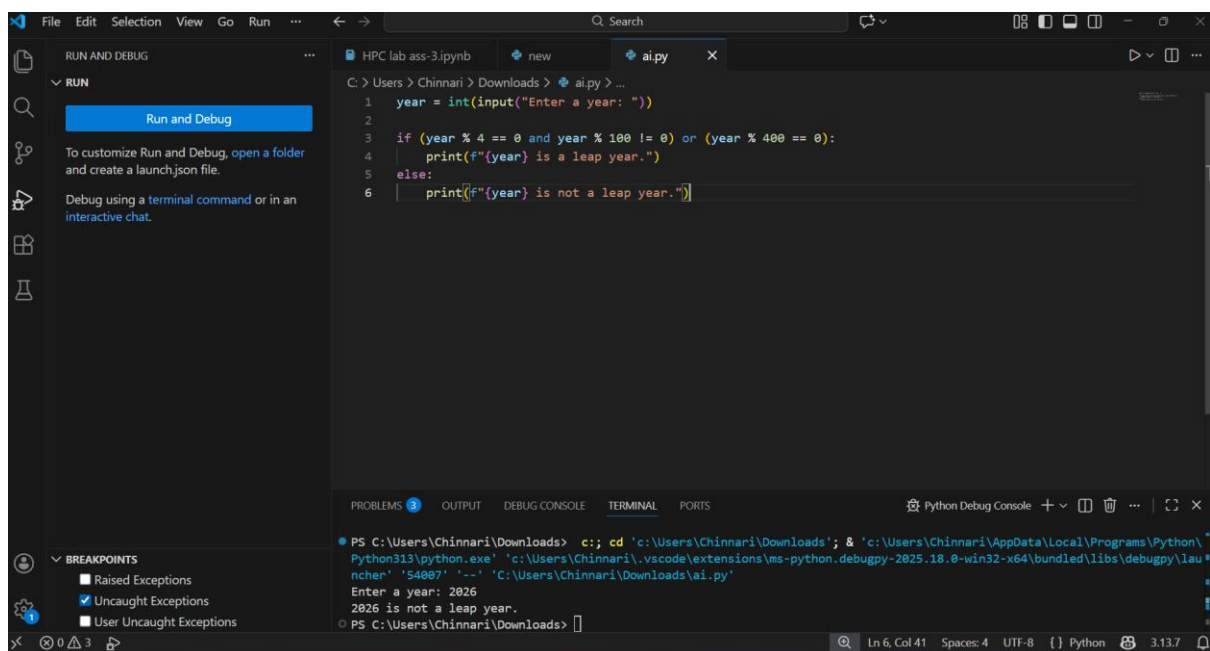
Use zero-shot prompting to instruct an AI tool to generate a Python function that:

- 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 a Visual Studio Code editor window with a Python file named `ai.py` open. The code defines a function to check if a year is a leap year. The terminal at the bottom shows the command to run the script, the input `2026`, and the output `2026 is not a leap year.`

```
C:\Users\Chinnari\Downloads> cd 'c:\Users\Chinnari\Downloads'; & 'c:\Users\Chinnari\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54007' '--' 'C:\Users\Chinnari\Downloads\ai.py'
Enter a year: 2026
2026 is not a leap year.
```

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:

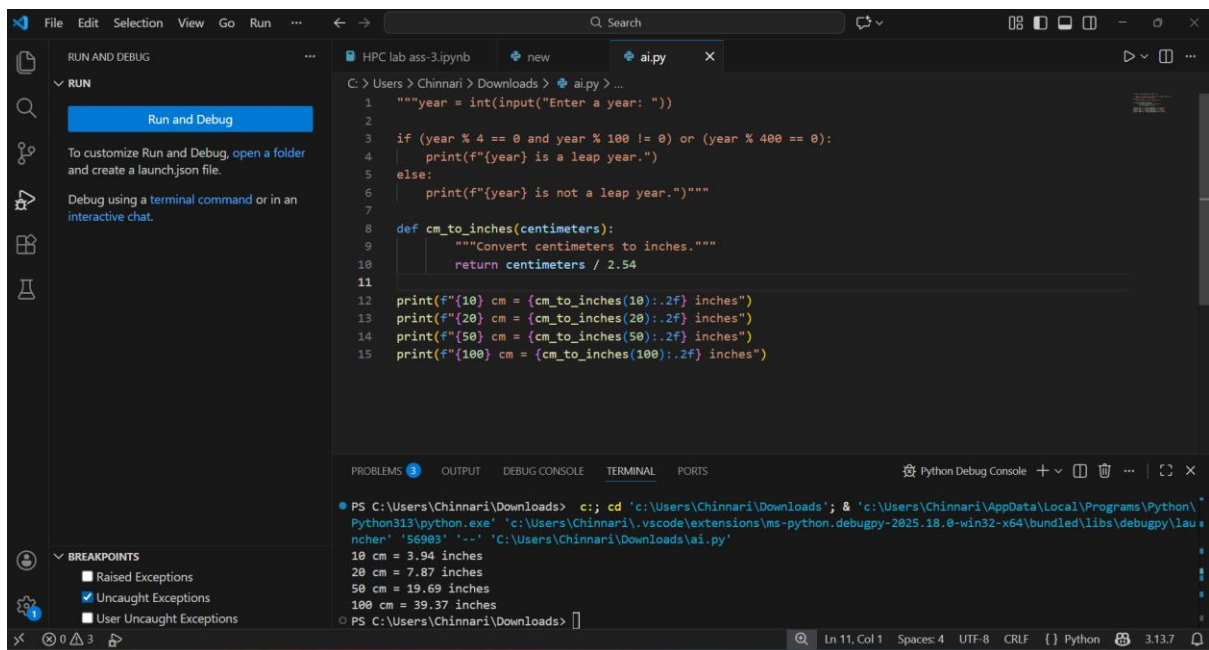
- Converts centimeters to inches
- Uses the correct mathematical formula

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



The screenshot shows a Visual Studio Code editor window with a Python file named 'ai.py'. The code defines a function 'cm_to_inches' that takes 'centimeters' as input and returns the value divided by 2.54. Below the function definition, there are four print statements that demonstrate the conversion for 10, 20, 50, and 100 centimeters. The output of the script is shown in the terminal at the bottom, displaying the results of these conversions: 10 cm = 3.94 inches, 20 cm = 7.87 inches, 50 cm = 19.69 inches, and 100 cm = 39.37 inches. The left sidebar shows the 'RUN AND DEBUG' panel with the 'Run and Debug' button highlighted. The bottom status bar indicates the file is at line 11, column 1, using UTF-8 encoding and CRLF line endings.

```
1 """year = int(input("Enter a year: "))
2
3 if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
4     print(f"{year} is a leap year.")
5 else:
6     print(f"{year} is not a leap year.")"""
7
8 def cm_to_inches(centimeters):
9     """Convert centimeters to inches."""
10    return centimeters / 2.54
11
12 print(f"10 cm = {cm_to_inches(10):.2f} inches")
13 print(f"20 cm = {cm_to_inches(20):.2f} inches")
14 print(f"50 cm = {cm_to_inches(50):.2f} inches")
15 print(f"100 cm = {cm_to_inches(100):.2f} inches")
```

```
PS C:\Users\Chinnari\Downloads> c:; cd 'c:\Users\Chinnari\Downloads'; & 'c:\Users\Chinnari\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56983' '-.-' 'C:\Users\Chinnari\Downloads\ai.py'
10 cm = 3.94 inches
20 cm = 7.87 inches
50 cm = 19.69 inches
100 cm = 39.37 inches
PS C:\Users\Chinnari\Downloads>
```

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

```

12 print(f"{10} cm = {cm_to_inches(10):.2f} inches")
13 print(f"{20} cm = {cm_to_inches(20):.2f} inches")
14 print(f"{50} cm = {cm_to_inches(50):.2f} inches")
15 print(f"{100} cm = {cm_to_inches(100):.2f} inches")
16 """
17
18 def format_name(full_name):
19     """Format a full name as 'Last, First'."""
20     parts = full_name.split()
21     if len(parts) == 2:
22         first, last = parts
23         return f"{last}, {first}"
24     return full_name
25
26 # Get user input and display formatted name
27 name = input("Enter a full name: ")
28 print(format_name(name))

```

Terminal Output:

```

PS C:\Users\Chinnari\Downloads> & 'c:\Users\Chinnari\AppData\Local\Programs\Python\Python313\python.exe'
c:\Users\Chinnari\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher
51954' '-.' 'c:\Users\Chinnari\Downloads\ai.py'
Enter a full name: John Smith
Smith, John
PS C:\Users\Chinnari\Downloads>

```

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

```

31
32 def count_vowels_zero_shot(text):
33     vowels = "aeiouAEIOU"
34     return sum(1 for char in text if char in vowels)
35
36
37 def count_vowels_few_shot(text):
38
39     vowels = "aeiouAEIOU"
40     return sum(1 for char in text if char in vowels)
41
42 print("Zero-shot result for 'hello':", count_vowels_zero_shot("hello"))
43 print("Few-shot result for 'hello':", count_vowels_few_shot("hello"))
44 print("Zero-shot result for 'Artificial Intelligence':", count_vowels_zero_shot("Artificial Intelligence"))
45 print("Few-shot result for 'Artificial Intelligence':", count_vowels_few_shot("Artificial Intelligence"))
46
47

```

Terminal Output:

```

PS C:\Users\Chinnari\Downloads> c:; cd 'c:\Users\Chinnari\Downloads'; & 'c:\Users\Chinnari\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-wi-n32-x64\bundle\libs\debugpy\launcher' '62696' '--' 'c:\Users\Chinnari\Downloads\ai.py'
Zero-shot result for 'hello': 2
Few-shot result for 'hello': 2
Zero-shot result for 'Artificial Intelligence': 10
Few-shot result for 'Artificial Intelligence': 10
PS C:\Users\Chinnari\Downloads>

```

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

```

48
49
50 def count_lines_in_file(filename):
51     try:
52         with open(filename, 'r') as file:
53             line_count = sum(1 for line in file)
54             return line_count
55     except FileNotFoundError:
56         print(f"Error: File '{filename}' not found.")
57         return -1
58     except Exception as e:
59         print(f"Error reading file: {e}")
60         return -1
61
62 if __name__ == "__main__":
63     with open("sample1.txt", 'w') as f:
64         f.write("Hello\nWorld")
65
66     with open("sample2.txt", 'w') as f:
67         f.write("Python\nFile\nHandling")

```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Chinnari\Downloads> c:; cd 'c:\Users\Chinnari\Downloads'; & 'c:\Users\Chinnari\AppData\Local\
Programs\Python\Python313\python.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-wi
n32-x64\bundled\libs\debugpy\launcher' '53650' '--' 'c:\Users\Chinnari\Downloads\ai.py' ...

```

Input file: sample3.txt
Output: 5 lines

```

60
61 if __name__ == "__main__":
62
63     with open("sample1.txt", 'w') as f:
64         f.write("Hello\nWorld")
65
66     with open("sample2.txt", 'w') as f:
67         f.write("Python\nFile\nHandling")
68
69     with open("sample3.txt", 'w') as f:
70         f.write("Line 1\nLine 2\nLine 3\nLine 4\nLine 5")
71
72     test_files = ["sample1.txt", "sample2.txt", "sample3.txt"]
73
74     for filename in test_files:
75         line_count = count_lines_in_file(filename)
76         print(f"Input file: {filename}")
77         print(f"Output: {line_count} lines")
78         print()

```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Chinnari\Downloads> c:; cd 'c:\Users\Chinnari\Downloads'; & 'c:\Users\Chinnari\AppData\Local\
Programs\Python\Python313\python.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-wi
n32-x64\bundled\libs\debugpy\launcher' '53650' '--' 'c:\Users\Chinnari\Downloads\ai.py' ...

```

Input file: sample3.txt
Output: 5 lines