

## **AI ASSISTED CODING**

### **ASSIGNMENT-4.3**

2303A52315

M.AKHIL REDDY

BATCH-45

ASSIGNMENT-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)

PROMPT:

Write a Python function that takes an integer representing a year as input and determines if it is a leap year. The function should return True if the year is a leap year and False otherwise. Use the standard rules for leap years: a year is a leap year if it is divisible by 4, but not by 100, unless it is also divisible by 400. Do not include any examples or test cases in the function code.

## EXPECTED OUTPUT:

The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there's a folder named 'AKHILS AI CODING' containing files 1.5.py, 2.5.py, 3.5.py, 4.3.py, 5.5.py, and user.activity.log. The main editor tab is 4.3.py, which contains the following code:

```
1 def is_leap_year(year):
2     if year % 400 == 0:
3         return True
4     elif year % 100 == 0:
5         return False
6     elif year % 4 == 0:
7         return True
8     else:
9         return False
10 n=int(input())
11 print(is_leap_year(n))
```

Below the editor, the terminal tab is active, showing the command PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py". The output shows the function being tested with the input 2004, resulting in 'True'. The terminal also shows the command PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>.

## 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

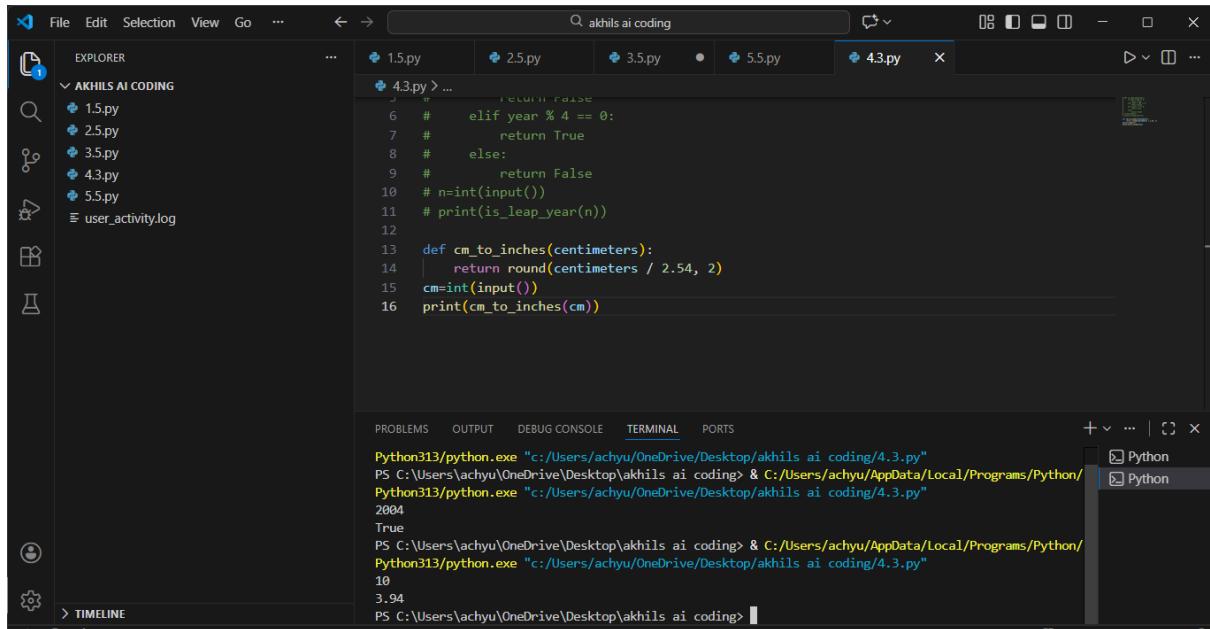
### PROMPT:

Write a Python function that takes a float representing centimeters as input and converts it to inches. The function should return the value in inches rounded to two decimal places. Use the conversion formula: inches = centimeters / 2.54.

Example: Input: 10 Output: 3.94

Do not include any additional examples or test cases in the function code.

#### EXPECTED OUTPUT:



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files: 1.5.py, 2.5.py, 3.5.py, 4.3.py, 5.5.py, and user\_activity.log. The 4.3.py file is open in the editor, displaying the following code:

```
def is_leap_year(year):
    if year % 4 == 0:
        return True
    else:
        return False

def cm_to_inches(cm):
    return round(cm / 2.54, 2)

cm=int(input())
print(cm_to_inches(cm))
```

The Terminal tab at the bottom shows the output of running the script:

```
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py"
2004
True
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py"
10
3.94
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

#### 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 takes a string representing a full name as input and formats it as "Last, First". The function should return the formatted string. Assume the input consists of exactly two names: first and last.

Examples: Input: "John Smith" Output: "Smith, John"

Input: "Anita Rao" Output: "Rao, Anita"

Input: "Michael Jordan" Output: "Jordan, Michael"

Do not include any additional examples or test cases in the function code.

## EXPECTED OUTPUT:

The screenshot shows a Visual Studio Code (VS Code) interface. The top bar has the title 'akhils ai coding'. The left sidebar ('EXPLORER') shows a folder named 'AKHILS AI CODING' containing files: 1.5.py, 2.5.py, 3.5.py, 4.3.py (which is selected), and 5.5.py. It also lists 'user\_activity.log'. The main area displays the content of '4.3.py':

```
1.5.py | 2.5.py | 3.5.py | 5.5.py | 4.3.py | x

4.3.py > ...
11   " print(%s_leap_year)" %j
12
13 # def cm_to_inches(centimeters):
14 #     return round(centimeters / 2.54, 2)
15 # cm=int(input())
16 # print(cm_to_inches(cm))
17
18 def format_name(full_name):
19     first, last = full_name.split()
20     return last + ", " + first
21 nm=input()
22 print(format_name(nm))
```

The bottom right corner shows a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | x
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py"
akhil reddy
reddy, akhil
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

A vertical sidebar on the right lists three Python environments: 'Python', 'Python', and 'Python'.

## 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

o Logical clarity

Expected Output

- Two vowel-counting functions
- Comparison table or short reflection paragraph
- Conclusion on prompt effectiveness

PROMPT:

Write a Python function that takes a string as input and returns the count of vowels in it. Vowels are a, e, i, o, u, considering both lowercase and uppercase. Do not include any examples or test cases in the function code.

---

Write a Python function that takes a string as input and returns the count of vowels in it. Vowels are a, e, i, o, u, considering both lowercase and uppercase.

Examples: Input: "hello" Output: 2

Input: "Python" Output: 1

Input: "AEIOU" Output: 5

Do not include any additional examples or test cases in the function code.

EXPECTED OUTPUT:

```
EXPLORER          1.5.py  2.5.py  3.5.py  5.5.py  4.3.py
AKHILS AI CODING
4.3.py> ...
22     "print(count_vowels(str))"
23
24 def count_vowels(input_string):
25
26     vowels = set('aeiouAEIOU')
27     count = 0
28     for char in input_string:
29         if char in vowels:
30             count += 1
31     return count
32 str=input()
33 print(count_vowels(str))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py"
akhil reddy
reddy, akhil
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py"
akhilreddy
3
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

```

1.5.py 2.5.py 3.5.py 5.5.py 4.3.py
30     "    count -= 1
31     #     return count
32     # str=input()
33     # print(count_vowels(str))
34
35 def count_vowels(input_string):
36
37     vowels = 'aeiouAEIOU'
38     return sum(1 for char in input_string if char in vowels)
39 str=input() (variable) str: str
40 print(count_vowels(str))

```

The screenshot shows a code editor interface with multiple tabs open. The current tab is '4.3.py'. The code in the editor is as follows:

```

30     "    count -= 1
31     #     return count
32     # str=input()
33     # print(count_vowels(str))
34
35 def count_vowels(input_string):
36
37     vowels = 'aeiouAEIOU'
38     return sum(1 for char in input_string if char in vowels)
39 str=input() (variable) str: str
40 print(count_vowels(str))

```

The terminal below shows the output of running the script:

```

PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py"
akhilreddy
3
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.3.py"
makhilreddy
4
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>

```

## Comparison

Aspect	Zero-Shot Function	Few-Shot Function
Accuracy	Fully accurate; correctly handles case insensitivity and counts only vowels.	Fully accurate; identical logic ensures correct vowel counting with case insensitivity.
Readability	Readable with explicit loop and counter, but slightly verbose.	Highly readable; uses concise generator expression for counting, which is more Pythonic.
Logical Clarity	Clear step-by-step logic: initialize counter, iterate, check membership, increment.	Clear and efficient; leverages built-in sum with comprehension for straightforward logic.

## Conclusion on Prompt Effectiveness

Both prompting strategies produced accurate and functional code, but few-shot prompting led to a more concise and idiomatic implementation, likely because the examples guided the AI toward optimized patterns. Zero-shot was effective for basic tasks but may require more refinement for elegance, while few-shot enhances reliability in capturing nuanced best practices. Overall, few-shot appears more effective for improving code quality in this scenario.

### 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 takes a string representing the filename of a .txt file as input, reads the file, counts the number of lines in it, and returns the integer line count.

Examples: Input: "example1.txt" (assuming content with 3 lines: "Hello\nWorld\n!") Output: 3

Input: "example2.txt" (assuming content with 1 line: "Single line") Output: 1

Input: "example3.txt" (assuming empty file) Output: 0

Do not include any additional examples or test cases in the function code.

#### EXPECTED OUTPUT:

A screenshot of the Visual Studio Code (VS Code) interface. The main area shows a code editor with a dark theme containing Python code. The code defines a function `count\_vowels` that takes a string and returns the count of vowels ('aeiouAEIOU'). It then defines a function `count\_lines` that reads a file and returns the number of lines. Finally, it prompts for a filename and prints the line count. The code editor has tabs for 1.5.py, 2.5.py, 3.5.py, 5.5.py, and 4.3.py, with 4.3.py currently selected.

```
36
37     #     vowels = 'aeiouAEIOU'
38     #     return sum(1 for char in input_string if char in vowels)
39     # str=input()
40     # print(count_vowels(str))
41
42 def count_lines(filename):
43     with open(filename, "r") as file:
44         return sum(1 for _ in file)
45
46 filename = input()
47 print(count_lines(filename))
```

The Explorer sidebar on the left shows files 1.5.py, 2.5.py, 3.5.py, 5.5.py, and user\_activity.log under the folder AKHILS AI CODING. The Timeline sidebar on the right shows a history of terminal sessions with command-line inputs and outputs. The bottom status bar indicates the file is 48 lines long, has 4 spaces, and is using Python 3.13.7.