ASSIGNMENT-4.3

M.PARAMESH

HNO: 2303A53021

Batch-46

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

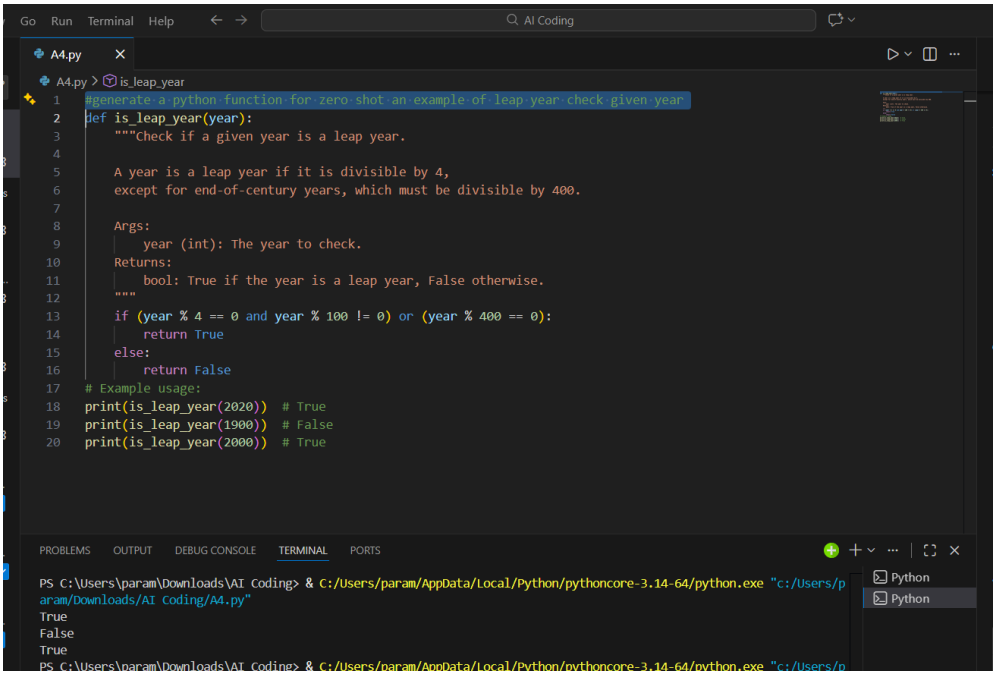
Prompt used :

#Generate a python function for zero shot an example of leap year check given year

Code: def is_leap_year(year):

```
    if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
        return True
    else:
        return False
```

Output:



Explaination : 1) The function is_leap_year(year) takes a **year** as input.

2) % (modulus operator) checks divisibility.

3) If the year is:

- divisible by **400**, it is a leap year
  OR
- divisible by **4** and **not divisible by 100**, it is a leap year

4) If the condition is satisfied, the function returns **True**.

5) Otherwise, it returns **False**.

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

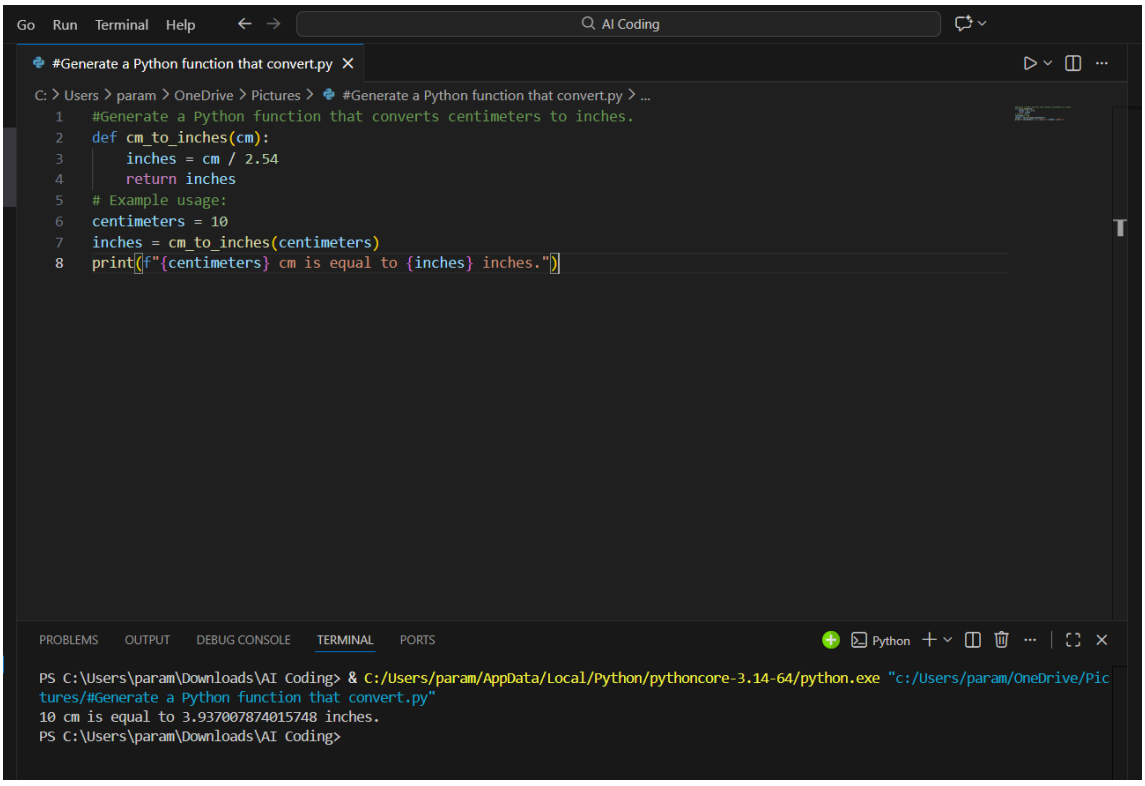
PROMPT USED :

#Generate a Python function that converts centimeters to inches.

Code :

```
def cm_to_inches(cm):
    # 1 inch = 2.54 centimeters
    inches = cm / 2.54
    return round(inches, 2)
```

Output :



Explaination:

- The function cm_to_inches(cm) takes a value in centimeters.
- The conversion formula used is:
- inches = centimeters / 2.54
- round(inches, 2) rounds the result to 2 decimal places.

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

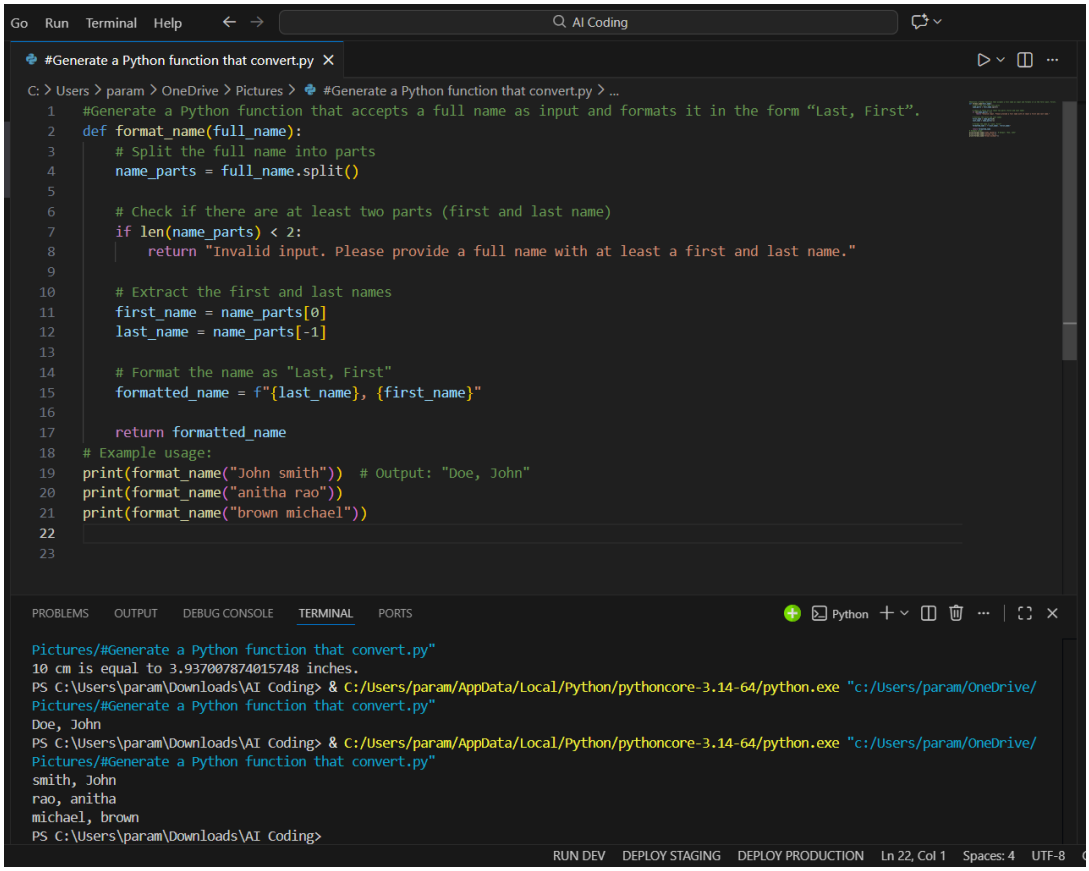
Code :

```
def format_name(full_name):
    # Split the full name into parts
    parts = full_name.strip().split()

    # First name is the first part, last name is the last part
    first_name = parts[0]
    last_name = parts[-1]

    # Format as "Last, First"
    return f"{last_name}, {first_name}"
```

output:



Explaination :

Name Formatting (“Last, First”)

1. The function format_name(full_name) takes a **full name string** as input.
2. strip() removes any extra spaces at the beginning or end of the name.
3. split() divides the name into individual words.
4. The **first word** is treated as the first name.
5. The **last word** is treated as the last name.
6. The function rearranges the name in the format **“Last, First”**.
7. The formatted name is returned as output.

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

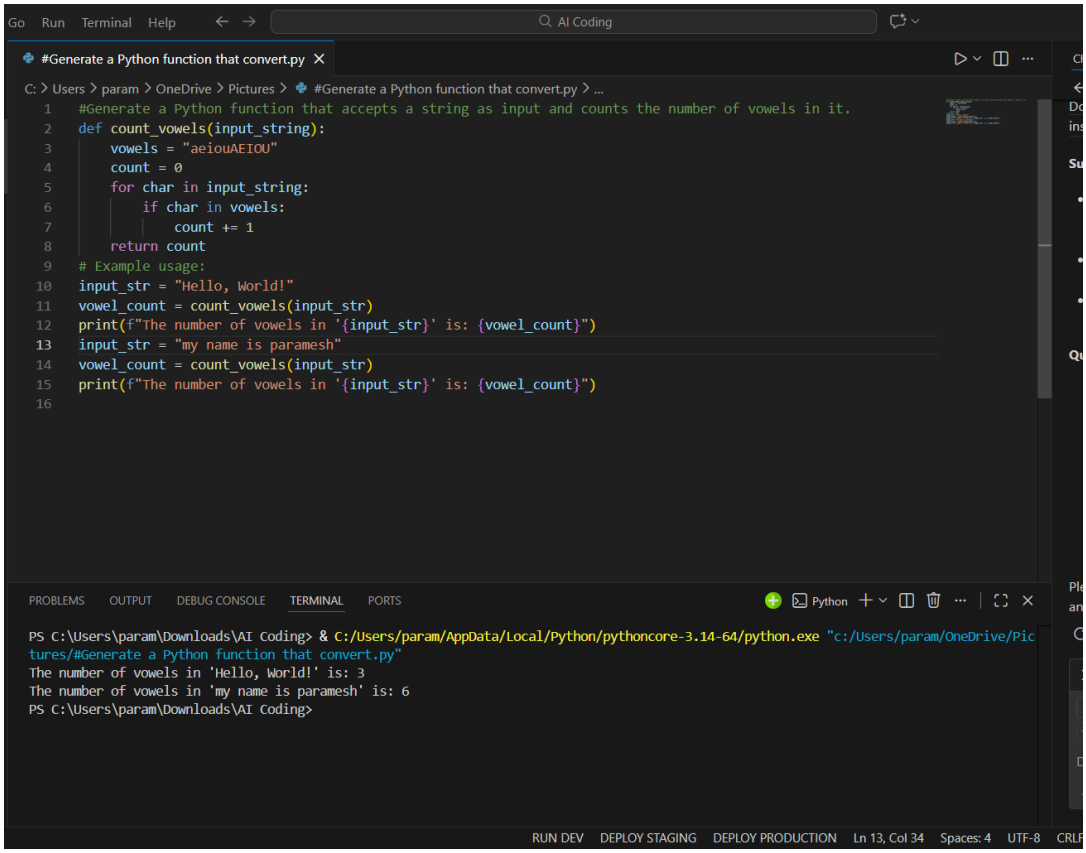
• Comparison table or short reflection paragraph

• Conclusion on prompt effectiveness

Code :

```
def count_vowels_zero_shot(text):
    vowels = "aeiouAEIOU"
    count = 0
    for ch in text:
        if ch in vowels:
            count += 1
    return count
```

output :



Explaination :

Comparison Table: Zero-Shot vs Few-Shot

Criteria	Zero-Shot Version	Few-Shot Version
Accuracy	Correct	Correct
Readability	Easy to understand (loop-based)	Concise and clean
Logical Clarity	Explicit step-by-step logic	Compact logic using Python features
Code Length	Slightly longer	Shorter

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.

Code: def count_lines_in_file(file_path):

```
  # Open the file in read mode
  with open(file_path, 'r') as file:
    # Read all lines from the file
    lines = file.readlines()


    # Return the number of lines
    return len(lines)



# Example usage
line_count = count_lines_in_file("python.txt")
print("The number of lines in the file is:", line_count)
```

explaination :

1) The function count_lines_in_file() opens a text file in read mode.

2)readlines() reads all lines from the file.

3) len() counts the total number of lines.

4) The function returns the line count as output.

Input: Hello World

File handling in Python

Counting number of lines

Output:

The number of lines in the file is: 3