

Name: P.Sai Venkatesh

H.No: 2303A53039 - B46

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:

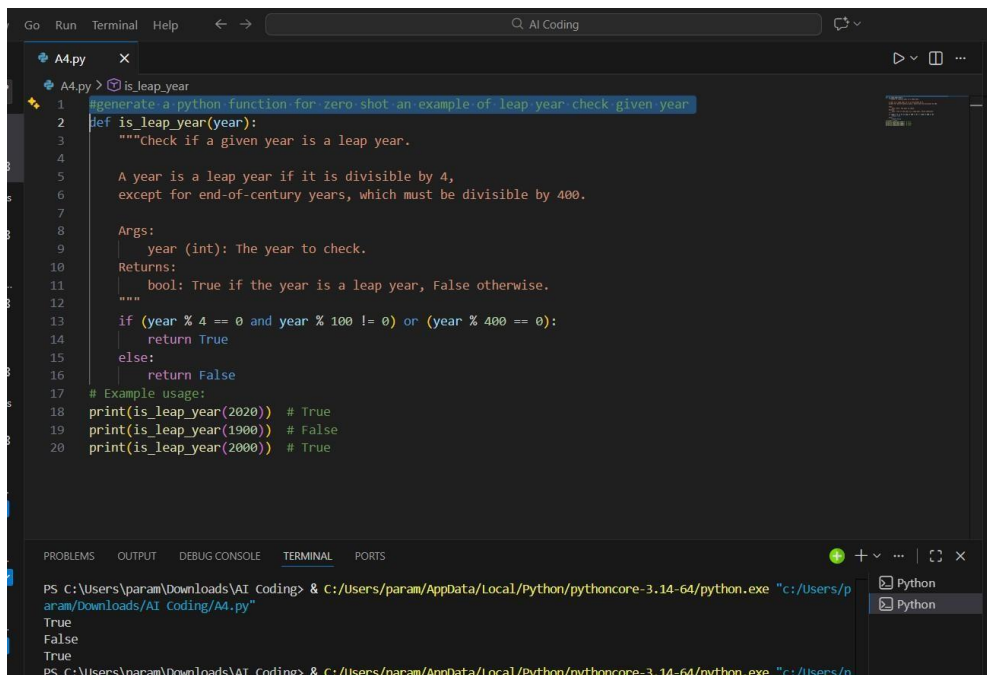
- Accepts a year as input
- Checks whether the given year is a leap year
- Returns an appropriate result Prompt used :

#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:



```
1 generate a python function for zero shot an example of leap year check given year
2 def is_leap_year(year):
3     """Check if a given year is a leap year.
4
5     A year is a leap year if it is divisible by 4,
6     except for end-of-century years, which must be divisible by 400.
7
8     Args:
9         year (int): The year to check.
10    Returns:
11        bool: True if the year is a leap year, False otherwise.
12    """
13    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
14        return True
15    else:
16        return False
17
18 # Example usage:
19 print(is_leap_year(2020)) # True
20 print(is_leap_year(1900)) # False
21 print(is_leap_year(2000)) # True
```

PS C:\Users\param\Downloads\AI_Coding> & C:\Users\param\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:/Users/p
aram/Downloads/AI_Coding/A4.py"

True
False
True

Explanation : 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.](#)

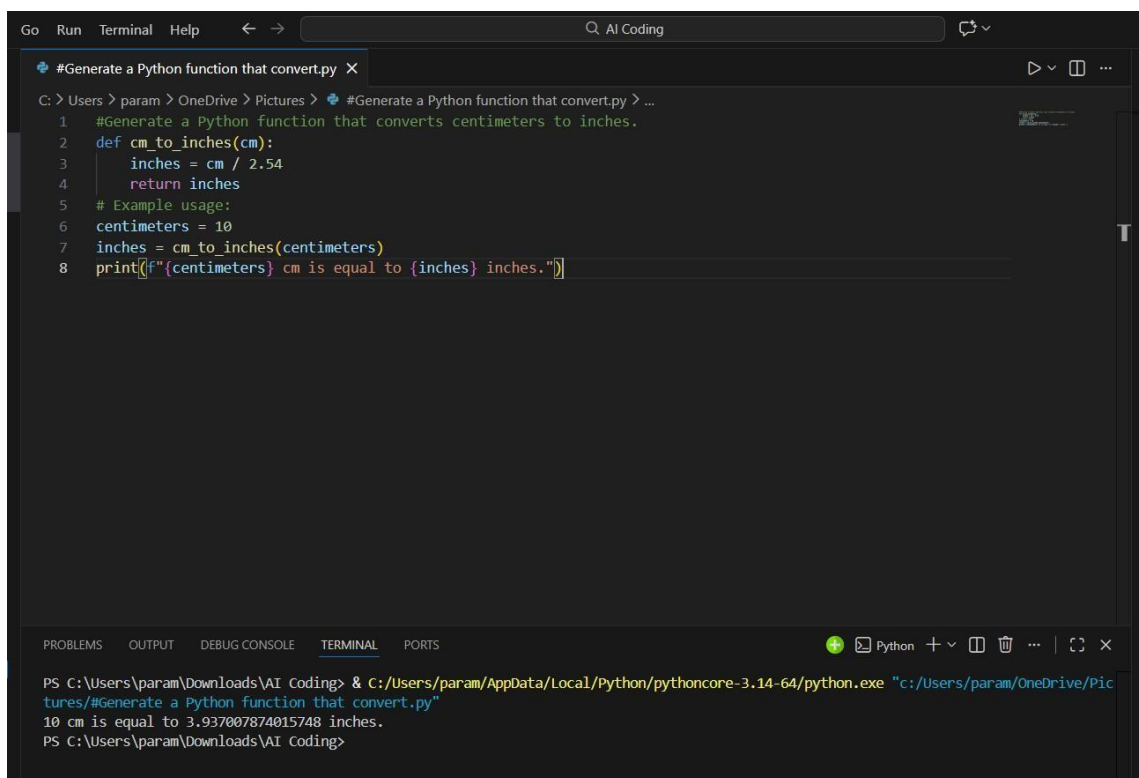
Code :

```
def cm_to_inches(cm):
```

```
    # 1 inch = 2.54 centimeters
```

```
    inches = cm / 2.54    return
```

round(inches, 2) Output :



The screenshot shows a Visual Studio Code editor window with a file named `#Generate a Python function that convert.py`. The code in the editor is as follows:

```
1 #Generate a Python function that converts centimeters to inches.
2 def cm_to_inches(cm):
3     inches = cm / 2.54
4     return inches
5 # Example usage:
6 centimeters = 10
7 inches = cm_to_inches(centimeters)
8 print(f"{centimeters} cm is equal to {inches} inches.")
```

Below the editor, the terminal window shows the command to run the script and its output:

```
PS C:\Users\param\Downloads\AI Coding> & C:/Users/param/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/param/OneDrive/Pictures/#Generate a Python function that convert.py"
10 cm is equal to 3.937007874015748 inches.
PS C:\Users\param\Downloads\AI Coding>
```

Explanation:

- The function `cm_to_inches(cm)` takes a value in centimeters.
- The conversion formula used is:
$$\text{inches} = \text{centimeters} / 2.54$$
- `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:

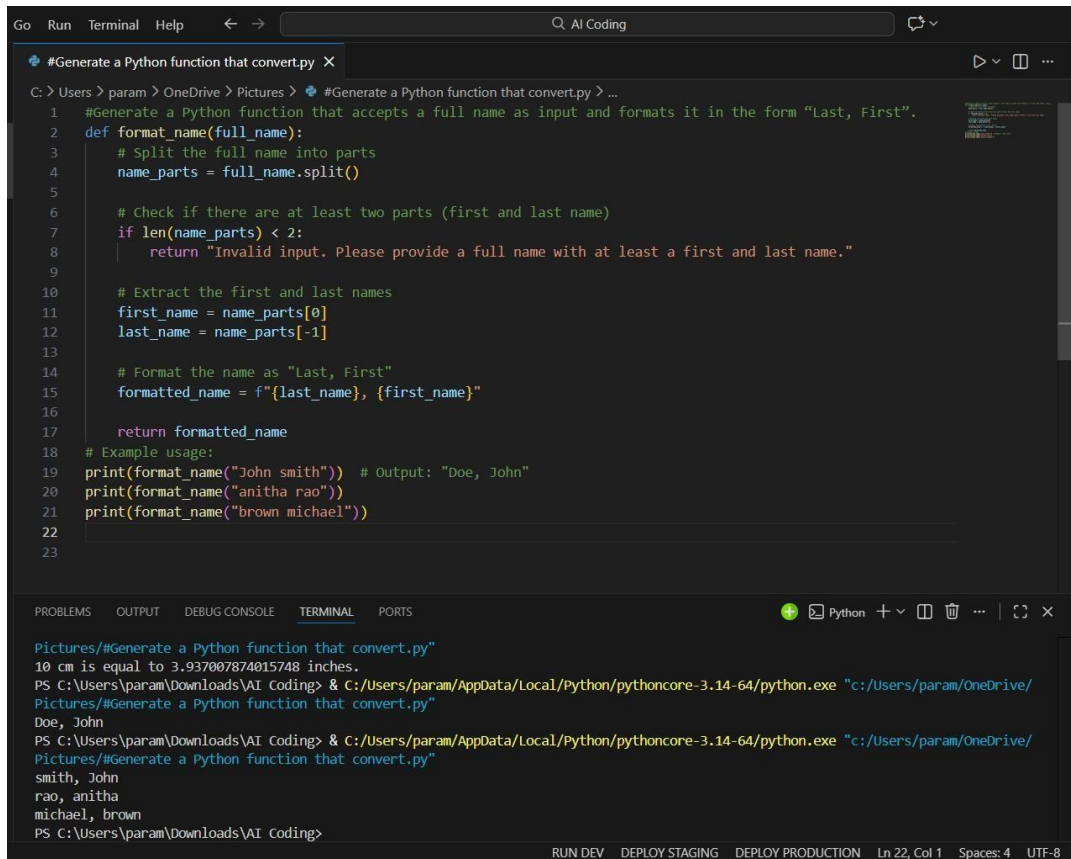
- Accepts a full name as input
- Formats it as “Last, First” Example formats:
- "John Smith" → "Smith, John"
- "Anita Rao" → "Rao, Anita"

Prompt used : Generate a Python function that accepts a full name as input and formats it in the form “**Last, First**”.

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:



The screenshot shows a VS Code editor window with a Python file named `#Generate a Python function that convert.py`. The code defines a function `format_name` that takes a full name as input and formats it as "Last, First". The function splits the name into parts, checks if there are at least two parts, and then formats the first and last names. Example usage is shown with `print` statements for "John smith", "aniha rao", and "brown michael". The terminal output shows the execution of the script, displaying the formatted names: "Doe, John", "smith, John", "rao, aniha", and "michael, brown".

```
C:\Users\param> OneDrive> Pictures> #Generate a Python function that convert.py> ...
1 #Generate a Python function that accepts a full name as input and formats it in the form "Last, First".
2 def format_name(full_name):
3     # Split the full name into parts
4     name_parts = full_name.split()
5
6     # Check if there are at least two parts (first and last name)
7     if len(name_parts) < 2:
8         return "Invalid input. Please provide a full name with at least a first and last name."
9
10    # Extract the first and last names
11    first_name = name_parts[0]
12    last_name = name_parts[-1]
13
14    # Format the name as "Last, First"
15    formatted_name = f"{last_name}, {first_name}"
16
17    return formatted_name
18
19 # Example usage:
20 print(format_name("John smith")) # Output: "Doe, John"
21 print(format_name("aniha rao"))
22 print(format_name("brown michael"))
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Pictures/#Generate a Python function that convert.py
10 cm is equal to 3.937007874015748 inches.
PS C:\Users\param\Downloads\AI Coding> & C:/Users/param/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/param/OneDrive/Pictures/#Generate a Python function that convert.py"
Doe, John
PS C:\Users\param\Downloads\AI Coding> & C:/Users/param/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/param/OneDrive/Pictures/#Generate a Python function that convert.py"
smith, John
rao, aniha
michael, brown
PS C:\Users\param\Downloads\AI Coding>

RUN DEV DEPLOY STAGING DEPLOY PRODUCTION Ln 22, Col 1 Spaces: 4 UTF-8

Explanation :

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

Prompt:

Generate a Python function that accepts a string as input and counts the number of vowels in it.

Code :

```
def count_vowels_zero_shot(text):
```

```
    vowels = "aeiouAEIOU"
```

```
    count = 0
```

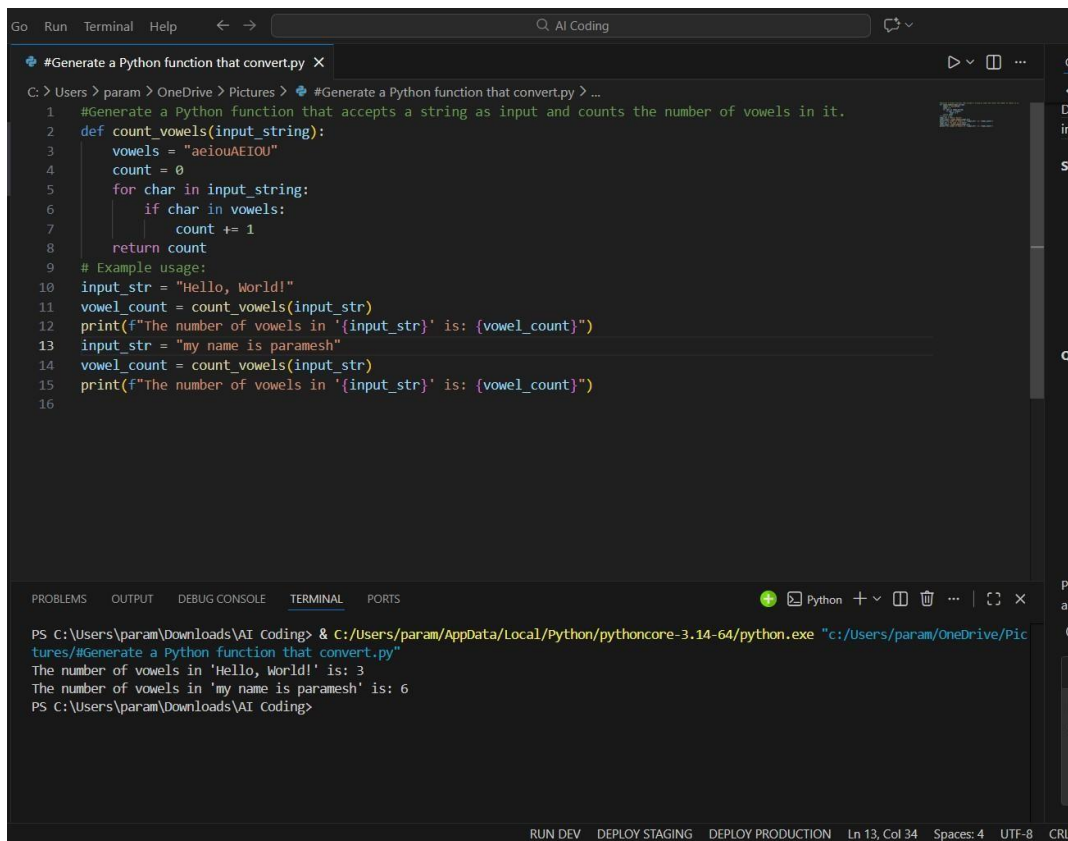
```
    for ch in text:
```

```
        if ch in vowels:
```

```
            count += 1
```

```
    return count
```

output :



The screenshot shows a VS Code editor window with a Python file named `#Generate a Python function that convert.py`. The code defines a function `count_vowels` that counts the number of vowels in a string. Below the function, there are example usages. The terminal at the bottom shows the command to run the script and the resulting output.

```
C:\> Users > param > OneDrive > Pictures > #Generate a Python function that convert.py > ...
1 #Generate a Python function that accepts a string as input and counts the number of vowels in it.
2 def count_vowels(input_string):
3     vowels = "aeiouAEIOU"
4     count = 0
5     for char in input_string:
6         if char in vowels:
7             count += 1
8     return count
9 # Example usage:
10 input_str = "Hello, World!"
11 vowel_count = count_vowels(input_str)
12 print(f"The number of vowels in '{input_str}' is: {vowel_count}")
13 input_str = "my name is paramesh"
14 vowel_count = count_vowels(input_str)
15 print(f"The number of vowels in '{input_str}' is: {vowel_count}")
16
```

Terminal Output:

```
PS C:\Users\param\Downloads\AI Coding> & C:/Users/param/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/param/OneDrive/Pictures/#Generate a Python function that convert.py"
The number of vowels in 'Hello, World!' is: 3
The number of vowels in 'my name is paramesh' is: 6
PS C:\Users\param\Downloads\AI Coding>
```

Explanation :

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)

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

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

explanation :

- 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

