

# **AI ASSISTANT CODING**

## **ASSIGNMENT-(4.3)**

**2303A52376**

**BT-45**

### **TASK-1:**

#### **PROMPT**

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 give a python code.

#### **CODE**

```
def is_leap_year(year):
```

```
    """
```

Check if a given year is a leap year.

A year is a leap year if:

- It is divisible by 4
- Except for years that are divisible by 100, unless they are also divisible by 400

Parameters:

year (int): The year to check

Returns:

bool: True if the year is a leap year, False otherwise

```
    """
```

```
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
```

```
    return True
```

```
else:
```

```
    return False
```

```
# Sample input and output

year = 2020

if is_leap_year(year):
    print(f"{year} is a leap year.")

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

## OUTPUT

```
SyntaxError: invalid character '\u2022' (at line 1)
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> & C:\Users\EDIT\AppData\Local\Programs\Python\Python313\python.exe c:/Users/EDIT/OneDrive/Desktop/CSE-4/AI-4.3.py
2020 is a leap year.
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4>
```

## EXPLANATION

Check whether a given year is leap year or not. For that we apply conditions like the year is divisible by 4 and Except for years that are divisible by 100, unless they are also divisible by 400.

## TASK-2:

### PROMPT

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 give a python code.

### CODE

```
def cm_to_inches(cm):
    """
    Convert centimeters to inches.
    """
```

Parameters:

cm (float): Length in centimeters

Returns:

float: Length in inches

.....

```
inches = cm / 2.54
```

```
return inches
```

```
# Sample test cases
```

```
test_values = [10, 25.4, 50, 100]
```

```
for cm in test_values:
```

```
    inches = cm_to_inches(cm)
```

```
    print(f"{cm} cm is equal to {inches:.2f} inches.")
```

## OUTPUT

```
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> & C:\Users\EDIT\AppData\Local\Programs\Python\Python313\python.exe c:/Users/EDIT/OneDrive/Desktop/CSE-4/AI-4.3.py
10 cm is equal to 3.94 inches.
25.4 cm is equal to 10.00 inches.
50 cm is equal to 19.69 inches.
100 cm is equal to 39.37 inches.
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> []
```

## EXPLANATION

Here we convert cm to inches by using formula inch=cm/2.54 and give some test values to convert it into inches and print the inches output.

## TASK-3

### PROMPT

se few-shot prompting with 2–3 examples to generate a Python function that:

- Accepts a full name as input
- Formats it as “Last, First”.Using few shot prompt write a python code.

### CODE

```

def format_name(full_name):
    """ Format a full name as "Last, First".
    Parameters:
        full_name (str): The full name in "First Last" format
    Returns:
        str: The formatted name in "Last, First" format
    """
    parts = full_name.split()
    if len(parts) != 2:
        raise ValueError("Input must be in 'First Last' format")
    first, last = parts
    return f"{last}, {first}"

# Sample inputs and outputs
test_names = ["John Smith", "Anita Rao", "Michael Jordan"]
for name in test_names:
    formatted_name = format_name(name)
    print(f"Original: {name} → Formatted: {formatted_name}")

```

## OUTPUT

```

s/EDIT/OneDrive/Desktop/CSE-4/AI-4.3.py
Original: John Smith → Formatted: Smith, John
Original: Anita Rao → Formatted: Rao, Anita
Original: Michael Jordan → Formatted: Jordan, Michael
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> 

```

## EXPLANATION

Format a full name as "Last, First". Parameters full\_name (str) The full name in "First Last" format .Returns str: The formatted name in "Last, First" format.

## **TASK-4**

### **PROMPT**

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

### **CODE**

```
def count_vowels_zero_shot(s):
```

```
    """
```

Count the number of vowels in a string.

Parameters:

s (str): Input string

Returns:

int: Number of vowels in the string

```
    """
```

```
vowels = "aeiouAEIOU"
```

```
count = 0
```

```
for char in s:
```

```
    if char in vowels:
```

```
        count += 1
```

```
return count
```

```
# Few-Shot Prompting Function
```

```
def count_vowels_few_shot(s):
```

.....

Count the number of vowels in a string.

Examples:

Input: "hello" → Output: 2

Input: "world" → Output: 1

Parameters:

s (str): Input string

Returns:

int: Number of vowels in the string

.....

```
vowels = "aeiouAEIOU"
```

```
count = sum(1 for char in s if char in vowels)
```

```
return count
```

```
# Sample Input and Output Testing
```

```
test_strings = ["hello", "world", "AI is fun", "Python programming"]
```

```
print("Zero-Shot Prompting Results:")
```

```
for s in test_strings:
```

```
    print(f"Input: '{s}' → Vowel Count: {count_vowels_zero_shot(s)}")
```

```
print("\nFew-Shot Prompting Results:")
```

```
for s in test_strings:
```

```
    print(f"Input: '{s}' → Vowel Count: {count_vowels_few_shot(s)}")
```

## OUTPUT

```
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> & C:\Users\EDIT\AppData\Local\Programs\Python\Python313\python.exe c:/users/EDIT/OneDrive/Desktop/CSE-4/AI-4-3.py
Zero-Shot Prompting Results:
Input: 'hello' → Vowel Count: 2
Input: 'world' → Vowel Count: 1
Input: 'AI is fun' → Vowel Count: 4
Input: 'Python programming' → Vowel Count: 4

Few-Shot Prompting Results:
Input: 'hello' → Vowel Count: 2
Input: 'world' → Vowel Count: 1
Input: 'AI is fun' → Vowel Count: 4
Input: 'Python programming' → Vowel Count: 4
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4>
```

## EXPLANATION

Using zero shot prompting and few shot prompting find number of vowels in a string.

## TASK-5

### PROMPT

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 GIVE A PYTHON CODE

### CODE

```
def count_lines_in_file(file_path):
```

```
    """
```

Count the number of lines in a text file.

Parameters:

file\_path (str): Path to the text file

Returns:

int: Number of lines in the file

```
    """
```

```
with open(file_path, 'r') as file:  
    lines = file.readlines()  
    return len(lines)  
  
# Sample Input and Output Testing  
  
if __name__ == "__main__":  
    sample_file = 'sample.txt'  
  
    # Create a sample text file for demonstration  
  
    with open(sample_file, 'w') as f:  
        f.write("Hello, World!\n")  
        f.write("This is a sample file.\n")  
        f.write("It has multiple lines.\n")  
  
    line_count = count_lines_in_file(sample_file)  
  
    print(f"The file '{sample_file}' has {line_count} lines.")
```

## OUTPUT

```
S/EDIT/OneDrive/Desktop/CSE-4/AI-4.3.py  
The file 'sample.txt' has 3 lines.  
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> []
```

## EXPLANATION

Count the number of lines in a text file.

