SR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Course Title – AI Assisted Coding

Course Code : 24CS002PC215

Assignment Type : Lab

Academic Year : 2025–2026

Name : Soumith Sripathi

Enroll Number : 2403A51340

Batch Number : 24BTCAICSB14

Week – 4 Date : 20 AUGUST

Task Description –1

Question : Zero–shot: Prompt AI to write a function that checks whether a given year is a leap year

PROGRAM AND OUTPUT :

```
def is_leap(year):
  """
  Checks if a given year is a leap year.
  Args:
    year: An integer representing the year.

  Returns:
    True if the year is a leap year, False otherwise.
  """
  return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

# Example usage
print(f"2000 is a leap year: {is_leap(2000)}")
print(f"1900 is a leap year: {is_leap(1900)}")
print(f"2024 is a leap year: {is_leap(2024)}")
print(f"2023 is a leap year: {is_leap(2023)}")
```

```
2000 is a leap year: True
1900 is a leap year: False
2024 is a leap year: True
2023 is a leap year: False
```

Explanation :

A function is_leap is defined.

It takes one argument: year (an integer).This is a docstring explaining:

What the function does. The input parameter (year). What it returns

This line contains the leap year logic:

A year is a leap year if It is divisible by 4, but not divisible by 100 .It is divisible by 400

The function is called with different years to test the output.
The result is printed using formatted strings (f-strings).
OUTPUT EXPECTATION :

2000: Divisible by 400 → Leap year → ✅ True
1900: Divisible by 100 but not 400 → Not a leap year → ❌ False
2024: Divisible by 4 and not by 100 → Leap year → ✅ True
2023: Not divisible by 4 → Not a leap year → ❌ False

Task Description –2

Question : One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

PROGRAM AND OUTPUT :

```python
def centimeters_to_inches(cm):
  """
  Converts a measurement in centimeters to inches.

  Args:
    cm: A float representing the measurement in centimeters.

  Returns:
    A float representing the equivalent measurement in inches.
  """
  return cm * 0.393701

# Example usage based on the provided example:
input_cm = 10
output_inches = centimeters_to_inches(input_cm)
print(f"Input: {input_cm} centimeters")
print(f"Output: {output_inches} inches")
```

```
Input: 10 centimeters
Output: 3.9370100000000003 inches
```

Explanation :

A function named centimeters_to_inches is defined.

It takes one parameter cm, which is expected to be a float or integer representing centimeters.

This is a multi–line docstring that explains:

Purpose of the function (conversion).

Argument type: cm should be a float.

Return value: a float (result in inches)

This is the core conversion formula.

1 centimeter = 0.393701 inches.

The function multiplies the input by this factor to convert cm .

A variable input_cm is assigned the value 10, which represents 10 centimeters.

The function centimeters_to_inches is called with

the argument 10.
The result is stored in output_inches.
These print statements use f–strings to format and display the input and output values.
The program correctly converts 10 centimeters to inches:
10 × 0.393701 = 3.93701
The result includes floating–point precision, which is why the output shows: 3.9370100000000003 inches.

Conclusion :

The program successfully:
Defines a conversion function.
Applies it to an example input.
Displays accurate and precise output.

Task Description –3

Question :

Few–shot: Provide 2–3 examples to generate a function that formats full names as “Last, First

PROGRAM AND OUTPUT

```python
def format_name(full_name):
  # Split the full name into parts
  name_parts = full_name.split()
  # Assuming the full name has at least a first and last name
  if len(name_parts) >= 2:
    last_name = name_parts[-1]
    first_name = " ".join(name_parts[:-1])
    return f"{last_name}, {first_name}"
  else:
    # Handle cases with less than two parts (e.g., just a first name)
    return full_name
# Example usage based on the provided examples:
print(f"Input: John Doe, Output: {format_name('John Doe')}")
print(f"Input: Jane Smith, Output: {format_name('Jane Smith')}")
print(f"Input: Peter Jones, Output: {format_name('Peter Jones')}")
print(f"Input: Cher, Output: {format_name('Cher')}") # Example with one name
```

```
Input: John Doe, Output: Doe, John
Input: Jane Smith, Output: Smith, Jane
Input: Peter Jones, Output: Jones, Peter
Input: Cher, Output: Cher
```

Explanation ：

Defines a function called format_name.

Takes one argument: full_name (a string input).

Uses .split() to break the name into a list of words.

Example: "John Doe" → ["John", "Doe"]

Verifies that the name has at least two parts (e.g., a first and last name).

last_name: Gets the last element from the list.

first_name: Joins all preceding elements (could include middle names).

Returns the formatted name in "Last, First" format using an f–string

If the name has less than two parts (e.g., just "Cher"), return it unchanged

Task Description –4

Question :

Compare zero–shot and few–shot prompts for writing a function that counts the number of vowels in a string

PROGRAM AND OUTPUT :

```
def count_vowels_few_shot(text):
  """
  Counts the number of vowels in a string.

  Examples:
  count_vowels_few_shot("hello") == 2
  count_vowels_few_shot("world") == 1
  count_vowels_few_shot("AEIOU") == 5
  """
  vowels = "aeiouAEIOU"
  count = 0
  for char in text:
    if char in vowels:
      count += 1
  return count

# Get user input
user_string = input("Enter a string to count vowels (few-shot): ")
vowel_count = count_vowels_few_shot(user_string)
print(f"The number of vowels in '{user_string}' is: {vowel_count}")
```

```
Enter a string to count vowels (few-shot): APPLE
The number of vowels in 'APPLE' is: 2
```

Explanation :

The first line defines a function that will count the number of vowels in a given string.

The next few lines are a documentation block explaining what the function does, along with some example calls and expected results.

Inside the function, a string containing all

lowercase and uppercase vowels is defined so the function can compare each character in the input against it.
A counter variable is initialized to zero to keep track of how many vowels are found.
A loop begins, which will go through each character in the input string one by one.
For each character, the function checks if it is a vowel by comparing it to the list of vowels defined earlier.
If it is a vowel, the counter is increased by one.
After the loop finishes checking the whole string, the function returns the total count of vowels found.
A comment is added to indicate that the next part of the code is for handling user input.
The user is prompted to enter a string to be checked for vowels.
The string entered by the user is passed into the vowel–counting function.
The returned count is stored in a variable.
A formatted message is printed, showing the original string and the number of vowels it contains.
In the output example shown at the bottom, the user typed “APPLE”, and the program correctly responded that the number of vowels is 2.

Task Description –5

Question :

Use few–shot prompting to generate a function that reads a .txt file and returns the number of lines.

PROGRAM AND OUTPUT :

```python
def count_lines_in_file(filepath):
    """
    Reads a text file and returns the number of lines.

    Examples:
    # Assuming 'example.txt' exists with 3 lines
    # count_lines_in_file("example.txt") == 3
    # Assuming 'empty.txt' exists with 0 lines
    # count_lines_in_file("empty.txt") == 0
    """
    try:
        with open(filepath, 'r') as file:
            line_count = sum(1 for line in file)
        return line_count
    except FileNotFoundError:
        return f"Error: File not found at '{filepath}'"
    except Exception as e:
        return f"An error occurred: {e}"

# Get user input for the file path
file_path = input("Enter the path to the text file: ")

# Call the function and print the result
number_of_lines = count_lines_in_file(file_path)
print(f"The number of lines in '{file_path}' is: {number_of_lines}")
```

```
Enter the path to the text file:
```

Output:

```csharp
The number of lines in 'example.txt' is: 3
```

Copy code

Explanation :

The function is named to indicate it will count lines in a file, and it takes a file path as input.

The documentation block describes:
What the function does (reads a text file and returns the number of lines),
And provides example use cases with expected results.
Inside the function, a try block is used to safely attempt opening and reading the file.
The file is opened in read mode using a with statement, which automatically handles closing the file after reading.
It uses a generator expression with sum() to count how many lines are in the file by looping once over each line.
The total line count is then ret
If the file is not found, a FileNotFoundError is caught, and a message is returned indicating that the file path is invalid.
Any other unexpected error is caught by a general Exception block, which returns a message describing the error.X CDG
A comment indicates that user input will be collected.
The user is prompted to enter the path to a text file.
That input is stored and passed to the line–counting function.

The result (number of lines) is printed in a formatted message showing the file path and the count.