

AI Assisted Coding

Assignment-4.3

Name :

M.Anu Chandra

RollNo:2303A54070

Batch:48

Lab4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

Task1: Zero-Shot Prompting – Leap Year Check Scenario:
Zero-shot prompting involves giving instructions without providing examples.

Prompt used: #Write a Python function that takes a year as input and checks whether it is a leap year.
#The function should return an appropriate message indicating whether the year is a leap year or not.
#Do not include any input-output examples.

- **Generated code:**

```
def is_leap_year(year):  
    if(year%4==0 and year%100!=0) or (year%400==0):  
        returnTrue  
    else:  
        returnFalse  
  
year=int(input("Enter year:")) print(is_leap_year(year))
```

- **Sample Input:**

Enter year: 2024

- **Sample Output:**

True

- **Short Explanation of Logic:**

The program checks the leap year conditions using logical operators. A year is a leap year if it is divisible by 400 or divisible by 4 but not divisible by 100. The logic is implemented directly without providing any prior.

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a project named "AI CODING" with files: assignment 4.3.py, task1.py, assignment1.py, assignment3.1.py, task2.py, task2.py, task3.py, task4.py, and task5.py.
- CODE EDITOR**: The file "task1.py" is open, containing the following code:

```

assignment 4.3.py > task1.py > ...
1 #Write a Python function that takes a year as input and checks whether it is a leap year.
2 #The Function should return an appropriate message indicating whether the year is a leap year.
3 #Do not include any input-output examples.
4
5 def is_leap_year(year):
6     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
7         return True
8     else:
9         return False
10
11 year = int(input("Enter year: "))
12 print(is_leap_year(year))
13
14

```
- PROBLEMS**: No errors or warnings.
- OUTPUT**: Shows the terminal output of the code execution.
- TERMINAL**: Shows the command run: "python assignment 4.3.py task1.py". The output shows the user entering "2024" and the program outputting "True".
- CHAT**: A sidebar titled "LEAP YEAR CHECK FUNCTION IN PYTHON" with the following text:
 - The function should take year as input
 - Return whether it is a leap year or not
 - Do not provide any examples
- STATUS BAR**: Shows the file path as "C:\Users\bindu\OneDrive\Desktop\AI CODING> task1.py", line 14, column 1, and other status information.

Task2: One-Shot Prompting – Centimeters to Inches Conversion

❖ Scenario: One-shot prompting guides AI using a single example.

- **Prompt used:**
- #Write a Python function to convert centimeters to inches
- #Use the formula inches=centimeters/2.54
- #Example:
- #Input: 10cm
- #Output: 3.94inches
- **Generated code:**

```

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

    Args:
        centimeters: A number representing length in centimeters

    Returns:
        A float representing the length in inches
    """
    inches=centimeters/2.54
    return round(inches, 2)

# Example usage
if __name__=="__main__":
    cm_value=10
    result=cm_to_inches(cm_value)
    print(f"Input:{cm_value}cm")
    print(f"Output:{result}inches")

```

- **SampleInput:**
- Entervalueincm:25.4
- **SampleOutput:**
- 10.0inches

ShortExplanationofLogic:

The function takes centimeters as input and converts it into inches using the standard formula $Cm \times 0.3937 = inches$. The prompt helps the AI generate the logic accurately in one short example.

The screenshot shows the Visual Studio Code interface. The code editor displays a Python script named `task2.py` which contains a function `cm_to_inches` that converts centimeters to inches using the formula $cm \times 0.3937$. The terminal below shows the execution of the script and its output for an input of 10 cm.

```

File Edit Selection View Go Run Terminal Help < > Q AI CODING
EXPLORER task1.py task2.py
AI CODING assignment 4.3.py
task1.py task2.py
assignment1st assignment3.1.py
task 2.py task2.py
task3.py task4.py
task5.py
assignment 4.3.py > task2.py > ...
1 def cm_to_inches(cm):
2     """
3         Convert centimeters to inches.
4
5     Args:
6         cm: A number representing length in centimeters
7
8     Returns:
9         A float representing the length in inches
10    """
11    inches = cm / 2.54
12    return round(inches, 2)
13
14
15 # Example usage
16 if __name__ == "__main__":
17     cm_value = 10
18     result = cm_to_inches(cm_value)
19     print(f"Input: {cm_value} cm")
20     print(f"Output: {result} inches")

```

PROBLEMS OUTPUT TERMINAL PORTS

DEBUG CONSOLE Filter (e.g. text, \exclude, \escape)

TUTORIAL

PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task1.py"
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task2.py"
● PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task2.py"
● Input: 10 cm
Output: 3.94 inches
○ PS C:\Users\bindu\OneDrive\Desktop\AI CODING>

master* 0 1 0 0 ▲ 0 Indexing completed.

Ln 20, Col 38 Spaces:4 UTF-8 [] Python 3.14.2

Task3: Few-shot prompting-NameFormatting

Scenario:

Few-shot prompting improves accuracy by providing multiple examples.



- **Prompt used:**
 - #Write a Python function to format names as "Last, First"
 - #Examples:
 - # "JohnSmith" -> "Smith, John"
 - # "Anita Rao" -> "Rao, Anita"

- **Generated code:**

```
def format_name(full_name):
    """
    Format a full name as "Last, First"

    Args:
        full_name(str): A full name in "First Last" format

    Returns:
        str: The name formatted as "Last, First"
    """
    parts = full_name.split()
    if len(parts) >= 2:
        first_name = parts[0]
        last_name = parts[-1]
        return f'{last_name}, {first_name}'
    return full_name

# Test examples
print(format_name("JohnSmith"))      # Smith, John
print(format_name("AnitaRao"))       # Rao, Anita
```

- **SampleInput:**

Enterfullname:JohnSmith

- **SampleOutput:Smith,John**

ShortExplanationofLogic: The function splits the fullname into first and last names and rearranges them in the required format it providing multiple examples improves few short

```

File Edit Selection View Go Run Terminal Help <- > Q: AI CODING
EXPLORER ... task1.py task2.py task3.py
AI CODING assignment 4.3.py > task3.py > ...
1 def format_name(full_name):
2     """
3         Format a full name as "Last, First"
4
5     Args:
6         full_name (str): A full name in "First Last" format
7
8     Returns:
9         str: The name formatted as "Last, First"
10    ....
11    parts = full_name.split()
12    if len(parts) >= 2:
13        first_name = parts[0]
14        last_name = parts[-1]
15        return f'{last_name}, {first_name}'
16    return full_name
17
18
19 # Test examples
20 print(format_name("John Smith")) # Smith, John
21 print(format_name("Anita Rao")) # Rao, Anita

```

PROBLEMS OUTPUT TERMINAL PORTS

DEBUG CONSOLE Filter (e.g. text, exclude \escape)

TERMINAL

```

ers\baidu\OneDrive\Desktop\AI CODING\assignment 4.3.py\task2.py"
Input: 10 cm
Output: 3.94 inches
PS C:\Users\baidu\OneDrive\Desktop\AI CODING> & C:/Users/baidu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/baidu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task3.py"
Smith, John
Rao, Anita
PS C:\Users\baidu\OneDrive\Desktop\AI CODING>

```

LN 21, COL 51 SPACES: 4 UTF-8 {} Python 3.14.2

Task4:ComparativeAnalysis–Zero-ShotvsFew-Shot

- **Scenario:**
Different prompt strategies may produce different code quality.
- **Prompt1:Zero-ShotPrompting**

Write a Python function that counts the number of vowels in a given string. The function should return the total count.

Do not provide any examples.

Generated code:

```
def count_vowels(string):
    vowels="aeiouAEIOU" count
    = 0
    for char in string:
        if char in vowels:
```

- **SampleInput:**

Enterstring>HelloWorld

- **SampleOutput:**

Number of vowels:3

- **Prompt2:Few-ShotPrompting**

- Write a Python function to count vowels in a string.
- Examples:
- Input: "hello" → Output: 2
- Input: "AI Assisted Coding" → Output: 7

❖ **Generated code:**

```

❖ def count_vowels(string):
❖     """Count the number of vowels in a string."""
❖     vowels = "aeiouAEIOU"
❖     return sum(1 for char in string if char in vowels)
❖
❖ # Test cases
❖ print(count_vowels("hello")) # Output: 2
❖ print(count_vowels("AI Assisted Coding")) # Output: 7

```

- **Sample Input:**
Enter string: HelloWorld
- **Sample Output:** 3

Explanation(Few-Shot):

The function uses a predefined vowel set and Python's sum with the generator expression to count vowels efficiently. The logic is compact and easier to understand due to example.

```

task401.py
assignment 4.3.py > task401.py @ count_vowels
1 def count_vowels(string):
2     vowels = "aeiouAEIOU"
3     count = 0
4     for char in string:
5         if char in vowels:
6             count += 1
7     return count

```

TERMINAL

```

DING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task401.py"
Smith, John
● Rao, Anita
PS C:/Users/bindu/OneDrive/Desktop/AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task401.py"
PS C:/Users/bindu/OneDrive/Desktop/AI CODING>

```

ComparisonTable

Criteria	Zero-Shot	Few-Shot
Accuracy	Correct	Correct
Readability	Medium	High
LogicalClarity	Simple	Very clear
CodeLength	Longer	Shorter

Task5: Few-Shot Prompting – File Handling (Line Count)

Scenario: You are building a text file analyzer that counts the number of lines in a file. Clear examples help ensure correct file handling.

Prompt used:

```
# Write a Python function to read a text
```

```
file#The function should take the filename as input
```

```
# Open the file in read mode # Count the total number of lines in the file # Return the
line count as an integer
```

Example:

If 'file.txt' contains 3 lines, the output should be 3

Generated code:

```
def count_lines(filename):
    """
        Read a text file and count the total number of lines.

    Args:
        filename(str): The name of the file to read

    Returns:
        int: The total number of lines in the file
    """
    try:
        with open(filename, 'r') as file:
            line_count = sum(1 for line in file)
        return line_count
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.") return 0

# Example usage
if __name__ == "main": result = count_lines("file.txt")
    print(f"Total lines: {result}")
```

- **Sample Input:**
- `print(count_lines("file.txt"))`
- **Sample Output:**

3

Short Explanation of Logic:

The program reads a text file in read mode and counts the number of lines present in it. A counter variable is used to keep track of the number of lines. Each time a line is read from the file, the counter increases by one. After reading the entire file, the function returns the total number of lines. This method ensures accurate line counting and is easy to understand.

The screenshot shows the Visual Studio Code (VS Code) interface. The main editor area displays a Python script named `task5.py`. The script contains code to read a text file and count the total number of lines. A tooltip for the `task5.py` tab indicates it is the active file. The terminal below shows the execution of the script and an error message due to a missing file.

```
File Edit Selection View Go Run Terminal Help ⏪ ⏴ Q AI CODING RUN AND DEBUG ... task1.py task2.py task3.py task401.py task42.py task5py x RUN To customize Run and Debug create a launch.json file. Debug using a terminal command or in an interactive chat. Show automatic Python configurations
```

```
assignment 4.3.py > ...
1 def count_lines(filename):
2     """
3         Read a text file and count the total number of lines.
4     """
5     Args:
6         filename (str): The name of the file to read
7
8     Returns:
9         int: The total number of lines in the file
10    """
11    try:
12        with open(filename, 'r') as file:
13            line_count = sum(1 for line in file)
14        return line_count
15    except FileNotFoundError:
16        print(f"Error: File '{filename}' not found.")
17    return 0
18
19
20 # Example usage
21 if __name__ == "__main__":
22     result = count_lines("file.txt")
23     print(f"Total lines: {result}")
```

```
PROBLEMS OUTPUT TERMINAL PORTS ... | x
DEBUG CONSOLE Filter (e.g. text, exclude, \escape)
TERMINAL
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task5.py"
Error: File 'file.txt' not found.
Total lines: 0
PS C:\Users\bindu\OneDrive\Desktop\AI CODING>
```

BREAKPOINTS
Raised Excepti...
Uncaught Excepti...
User Uncaught...
Indexing completed.

Ln 23, Col 36 Spaces: 4 UTF-8 {} Python 3.142

