Assignment Number: 4.1

NAME: K. ANIRUDH

H.T.NO: 2303A51478

BATCH: 24

Q1. Zero-Shot Prompting (Basic Lab Task)

Task:

Write a Python function that classifies a given text as Spam or Not

Spam using zero-shot prompting.

Steps:

1. Construct a prompt without any examples.

2. Clearly specify the output labels.

3. Display only the predicted label.

Input:

"Congratulations! You have won a free lottery ticket."

Expected Output:

Spam

```python
#generate a Python code that classifies a given text as Spam or Not Spam

def classify_text(text):
    """Classify text as Spam or Not Spam
    Args:
    text (str): Input text to classify
    Returns:
    str: "Spam" if the text is classified as spam, "Not Spam" otherwise
    """
    spam_keywords = ["win", "free", "prize", "click", "buy now", "limited time offer"]
    text_lower = text.lower()
    for keyword in spam_keywords:
        if keyword in text_lower:
            return "Spam"
    return "Not Spam"
# Example usage
input_text = "Congratulations! You win a free prize. Click here to claim it."
result = classify_text(input_text)
print(f"The text is classified as: {result}")
```

```
BLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

ning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compati
port-Module PSReadLine'.

C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.
-lab.py"
 text is classified as: Spam
```

2. One-Shot Prompting (Emotion detection)

Task:

Write a Python program that detects the emotion of a sentence

using one-shot prompting.

Emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']

Steps:

1. Provide one labeled example inside the prompt.

2. Take a sentence as input.

3. Print the predicted emotion

```python
Write a Python program that detects the emotion of a sentence
Emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
example:
Input: "I am so happy today!
Output: "happy"
'''
def detect_emotion(sentence):
    """Detect the emotion of a sentence
    Args:
    sentence (str): Input sentence to analyze
    Returns:
    str: Detected emotion from the sentence
    """
    emotions = {
        'happy': ['happy', 'joyful', 'elated', 'pleased', 'content'],
        'sad': ['sad', 'unhappy', 'downcast', 'dejected', 'sorrowful'],
        'angry': ['angry', 'mad', 'furious', 'irate', 'annoyed'],
        'excited': ['excited', 'thrilled', 'eager', 'enthusiastic', 'overjoyed'],
        'nervous': ['nervous', 'anxious', 'uneasy', 'apprehensive', 'tense'],
        'neutral': ['neutral', 'indifferent', 'unemotional', 'calm', 'composed']
    }
    sentence_lower = sentence.lower()
    for emotion, keywords in emotions.items():
        for keyword in keywords:
            if keyword in sentence_lower:
                return emotion
    return "neutral"  # Default to neutral if no emotion is detected
# Example usage
input_sentence = "I am so happy today!"
detected_emotion = detect_emotion(input_sentence)
print(f"The detected emotion is: {detected_emotion}")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

I-lab.py"
e detected emotion is: happy

Q3. Few-Shot Prompting (Student Grading Based on Marks)

Task:

Write a Python program that predicts a student's grade based on

marks using few-shot prompting.

Grades:

['A', 'B', 'C', 'D', 'F']

Grading Criteria (to be inferred from examples):

• 90–100 → A

• 80–89 → B

• 70–79 → C

• 60–69 → D

• Below 60 → F

```python
259     '''Write a Python program that predicts a student's grade based on marks
260      90-100 → A
261     • 80-89 → B
262     • 70-79 → C
263     '''
264     def predict_grade(marks):
265         """Predict the student's grade based on marks
266         Args:
267         marks (int): Marks obtained by the student
268         Returns:
269         str: Grade corresponding to the marks
270         """
271         if 90 <= marks <= 100:
272             return 'A'
273         elif 80 <= marks < 90:
274             return 'B'
275         elif 70 <= marks < 80:
276             return 'C'
277         else:
278             return 'F'   # Assuming F for marks below 70
279     # Example usage
280     student_marks = [95, 82, 67, 74, 58]
281     for marks in student_marks:
282         grade = predict_grade(marks)
283         print(f"Marks: {marks}, Grade: {grade}")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
/AI-lab.py"
Marks: 95, Grade: A
Marks: 82, Grade: B
Marks: 67, Grade: F
Marks: 74, Grade: C
Marks: 58, Grade: F
```

Q4. Multi-Shot Prompting (Indian Zodiac Sign Prediction using Month Name)

Task:

Write a Python program that predicts a person's Indian Zodiac sign (Rashi) based on the month of birth (month name) using multi-shot prompting.

Indian Zodiac Order (Simplified Month-Based Model): The Indian Zodiac cycle starts in March with Mesha and follows this order:

March → Mesha

April → Vrishabha

May → Mithuna

June → Karka

July → Simha

August → Kanya

September → Tula

October → Vrischika

November → Dhanu

December → Makara

January → Kumbha

February → Meena

```python
'''
Write a Python program that predicts a person's Indian Zodiac sign
(Rashi) based on the month of birth (month name) using multi-shot
prompting.
Indian Zodiac Order (Simplified Month-Based Model)

March → Mesha
April → Vrishabha
May → Mithuna
June → Karka
July → Simha
August → Kanya
September → Tula
October → Vrischika
November → Dhanu
December → Makara
January → Kumbha
February → Meena
'''
def predict_rashi(month):
    """Predict the Indian Zodiac sign (Rashi) based on the month of birth
    Args:
        month (str): Month of birth
    Returns:
        str: Corresponding Rashi
    """
    rashi_dict = {
        'March': 'Mesha',
        'April': 'Vrishabha',
        'May': 'Mithuna',
        'June': 'Karka',
        'July': 'Simha',
        'August': 'Kanya',
        'September': 'Tula',
        'October': 'Vrischika',
        'November': 'Dhanu',
        'December': 'Makara',
        'January': 'Kumbha',
        'February': 'Meena'
    }
    return rashi_dict.get(month, "Invalid month name")
# Example usage
input_month = input("Enter the month of birth: ")
predicted_rashi = predict_rashi(input_month)
print(f"The predicted Rashi is: {predicted_rashi}")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/yadav/OneDrive/Desktop/PYTHON all/AI-lab.py"
Enter the month of birth: May
The predicted Rashi is: Mithuna
```

Q5. Result Analysis Based on Marks

Task: Write a Python program that determines whether a student

Passes or Fails based on marks using Chain-of-Thought (CoT)

prompting.

Result Categories:

['Pass', 'Fail']

```
331    Read the marks inputs from the user
332    determine the grade based on the marks
333    hnadle invalid inputs and display error message
334    '''
335    def determine_grade():
336        """Determine the grade based on user input marks
337        Returns:
338        str: Grade corresponding to the marks or an error message for invalid input
339        """
340        try:
341            marks = float(input("Enter the marks (0-100): "))
342            if marks < 0 or marks > 100:
343                return "Error: Marks should be between 0 and 100."
344            if marks >= 90:
345                return 'A'
346            elif marks >= 80:
347                return 'B'
348            elif marks >= 70:
349                return 'C'
350            elif marks >= 60:
351                return 'D'
352            else:
353                return 'F'
354        except ValueError:
355            return "Error: Invalid input. Please enter a numeric value for marks."
356    # Example usage
357    grade = determine_grade()
358    print(f"The determined grade is: {grade}")
359
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
Enter the marks (0-100): 0
The determined grade is: F
PS C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/Window
Enter the marks (0-100): 77
The determined grade is: C
```

Q6 Voting Eligibility Check (Chain-of-Thought Prompting)

Task: Write a Python program that determines whether a person is

eligible to vote using Chain-of-Thought (CoT) prompting.

```python
360    '''
361    Write a Python program that determines whether a person is eligible to vote based on their age.
362    read the age input from the user
363    check if the age is 18 or above
364    if yes then display "Eligible to vote"
365    handle invalid inputs and display error message
366    '''
367    def check_voting_eligibility():
368        """Check if a person is eligible to vote based on age
369        Returns:
370        str: "Eligible to vote" if age is 18 or above, or an error message for invalid input
371        """
372        try:
373            age = int(input("Enter your age: "))
374            if age < 0:
375                return "Error: Age cannot be negative."
376            if age >= 18:
377                return "Eligible to vote"
378            else:
379                return "Not eligible to vote"
380        except ValueError:
381            return "Error: Invalid input. Please enter a valid age."
382    # Example usage
383    eligibility_status = check_voting_eligibility()
384    print(eligibility_status)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
PS C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.
Enter your age: 8
Not eligible to vote
PS C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.
Enter your age: 20
Eligible to vote
```

Q7 Prompt Chaining (String Processing – Palindrome Names)

Task: Write a Python program that uses the prompt chaining

technique to identify palindrome names from a list of student

names.

```python
386  '''generate 10 names and store it in a list names
387  if the name is a palindrome then add it to palindrome_names list
388  display the palindrome_names list
389  handle invalid input and error message'''
390  def find_palindrome_names():
391      """Find palindrome names from a predefined list
392      Returns:
393      list: List of palindrome names
394      """
395      names = ["Anna", "Bob", "rathore", "David", "Eve", "sundar", "John", "Kayak", "Liam", "Madam"]
396      palindrome_names = []
397      for name in names:
398          if name.lower() == name.lower()[::-1]:
399              palindrome_names.append(name)
400      return palindrome_names
401  # Example usage
402  palindrome_names_list = find_palindrome_names()
403  print("Palindrome names:", palindrome_names_list)
404
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Us
Palindrome names: ['Anna', 'Bob', 'Civic', 'Eve', 'Hannah', 'Kayak', 'Madam']
PS C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Us
Palindrome names: ['Anna', 'Bob', 'Eve', 'Kayak', 'Madam']
```

Q8 Prompt Chaining (String Processing – Word Length

Analysis)

Task: Write a Python program that uses prompt chaining to

analyze a list of words. In the first prompt, generate a list of words.

In the second prompt, traverse the list and calculate the length of

each word. In the third prompt, use the output of the previous step

to determine whether each word is Short (length less than 5) or

Long (length greater than or equal to 5), and display the result for

each word

```python
'''generate a list of words named words_list
traverse  the list and calculate the length of each word.
if length is less than 5 make a list and add as "short" to list
if length greater than 7 add as "long" to list  '''
def classify_word_lengths():
    """Classify words based on their lengths
    Returns:
    list: List containing "short" or "long" based on word lengths
    """
    words_list = ["apple", "banana", "kiwi", "strawberry", "fig", "watermelon", "grape", "blueberry", "pear", "pineapple"]
    length_classification = []
    for word in words_list:
        if len(word) < 5:
            length_classification.append("short")
        elif len(word) > 7:
            length_classification.append("long")
    return length_classification
# Example usage
classification_result = classify_word_lengths()
print("Word length classification:", classification_result)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
PS C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/yadav/OneDrive
Word length classification: ['short', 'long', 'short', 'long', 'long', 'short', 'long']
```