Assignment Number: 4.1

NAME: N.Nanda Mukesh

H.T.NO: 2303A51410

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.

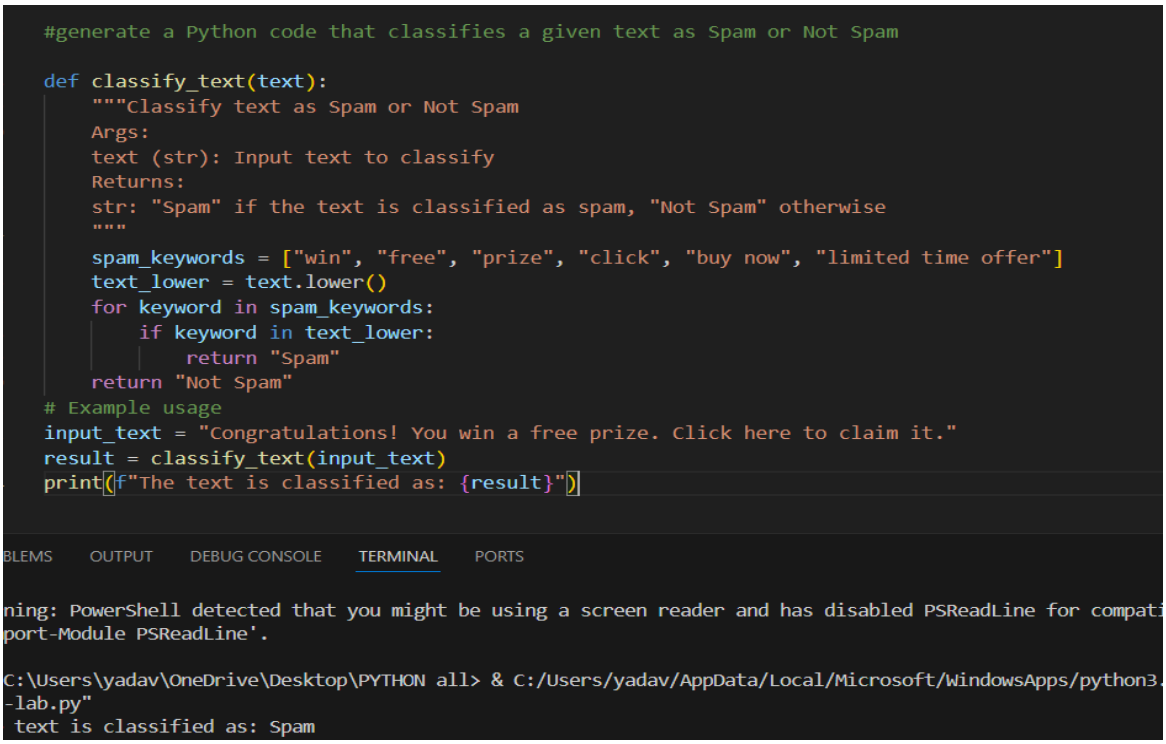
3. Display only the predicted label.

Input:

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

Expected Output:

Spam



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

```
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}")
```

```
OBLEMS   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
'''Write a Python program that predicts a student's grade based on marks
 90-100 → A
• 80-89 → B
• 70-79 → C
'''
def predict_grade(marks):
    """Predict the student's grade based on marks
    Args:
    marks (int): Marks obtained by the student
    Returns:
    str: Grade corresponding to the marks
    """
    if 90 <= marks <= 100:
        return 'A'
    elif 80 <= marks < 90:
        return 'B'
    elif 70 <= marks < 80:
        return 'C'
    else:
        return 'F'  # Assuming F for marks below 70
# Example usage
student_marks = [95, 82, 67, 74, 58]
for marks in student_marks:
    grade = predict_grade(marks)
    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

```
'''
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']

```
Read the marks inputs from the user
determine the grade based on the marks
hnadle invalid inputs and display error message
'''
def determine_grade():
    """Determine the grade based on user input marks
    Returns:
    str: Grade corresponding to the marks or an error message for invalid input
    """
    try:
        marks = float(input("Enter the marks (0-100): "))
        if marks < 0 or marks > 100:
            return "Error: Marks should be between 0 and 100."
        if marks >= 90:
            return 'A'
        elif marks >= 80:
            return 'B'
        elif marks >= 70:
            return 'C'
        elif marks >= 60:
            return 'D'
        else:
            return 'F'
    except ValueError:
        return "Error: Invalid input. Please enter a numeric value for marks."
# Example usage
grade = determine_grade()
print(f"The determined grade is: {grade}")

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/Windo
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.

```
'''
Write a Python program that determines whether a person is eligible to vote based on their age.
read the age input from the user
check if the age is 18 or above
if yes then display "Eligible to vote"
handle invalid inputs and display error message
'''
def check_voting_eligibility():
    """Check if a person is eligible to vote based on age
    Returns:
    str: "Eligible to vote" if age is 18 or above, or an error message for invalid input
    """
    try:
        age = int(input("Enter your age: "))
        if age < 0:
            return "Error: Age cannot be negative."
        if age >= 18:
            return "Eligible to vote"
        else:
            return "Not eligible to vote"
    except ValueError:
        return "Error: Invalid input. Please enter a valid age."
# Example usage
eligibility_status = check_voting_eligibility()
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
'''generate 10 names and store it in a list names
if the name is a palindrome then add it to palindrome_names list
display the palindrome_names list
handle invalid input and error message'''
def find_palindrome_names():
    """Find palindrome names from a predefined list
    Returns:
    list: List of palindrome names
    """
    names = ["Anna", "Bob", "rathore", "David", "Eve", "sundar", "John", "Kayak", "Liam", "Madam"]
    palindrome_names = []
    for name in names:
        if name.lower() == name.lower()[::-1]:
            palindrome_names.append(name)
    return palindrome_names
# Example usage
palindrome_names_list = find_palindrome_names()
print("Palindrome names:", palindrome_names_list)
```

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:/U
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:/U
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']
```