AI Assistant Coding

Assignment 4.1

Name : N. Eshwar **HT. No :** 2303A52072 **Batch:** 32

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.

Prompt:

```
#write a python function that classifies a text as spam or not
```

Code:

```
def classify_spam(text):
    spam_keywords = ['win', 'free', 'prize', 'click', 'buy now', 'limited time offer', 'winner', 'cash', 'urgent', 'act now']
    text_lower = text.lower()

    for keyword in spam_keywords:
        if keyword in text_lower:
            return True  # Classified as spam

    return False  # Not classified as spam
# Example usage
if __name__ == "__main__":
    test_texts = [
        "Congratulations! You are a winner of a free prize.",
        "Hello, how are you doing today?",
        "Limited time offer! Click here to claim your cash reward.",
        "Don't forget our meeting tomorrow."
    ]

    for text in test_texts:
        if classify_spam(text):
            print(f"Spam: {text}")
        else:
            print(f"Not Spam: {text}")
```

Output:

```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/spam.py
Spam: Congratulations! You are a winner of a free prize.
Not Spam: Hello, how are you doing today?
Spam: Limited time offer! Click here to claim your cash reward.
Not Spam: Don't forget our meeting tomorrow.
(AIAC_env) PS D:\Course\AIAC>
```

Q2. 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’]

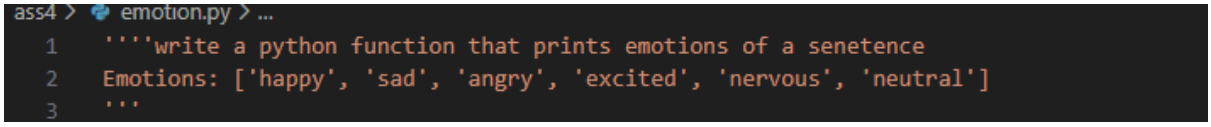
Steps:

1. Provide one labeled example inside the prompt.

2. Take a sentence as input.

3. Print the predicted emotion.

Prompt:



Code:

```python
def classify_emotion(text):
    emotion_keywords = {
        'happy': ['joy', 'pleased', 'delighted', 'cheerful', 'content'],
        'sad': ['unhappy', 'sorrowful', 'dejected', 'downcast', 'mournful'],
        'angry': ['mad', 'furious', 'irate', 'livid', 'outraged'],
        'excited': ['thrilled', 'elated', 'eager', 'enthusiastic', 'overjoyed'],
        'nervous': ['anxious', 'tense', 'apprehensive', 'worried', 'uneasy'],
        'neutral': ['calm', 'indifferent', 'unemotional', 'detached', 'impartial']
    }

    text_lower = text.lower()

    for emotion, keywords in emotion_keywords.items():
        for keyword in keywords:
            if keyword in text_lower:
                return emotion  # Classified emotion

    return 'neutral'  # Default to neutral if no keywords found
# Example usage
if __name__ == "__main__":
    test_texts = [
        "I am so delighted to see you!",
        "I feel very anxious about the exam.",
        "He was furious when he heard the news.",
        "It's just an ordinary day."
    ]

    for text in test_texts:
        emotion = classify_emotion(text)
        print(f"Emotion: {emotion} | Text: {text}")
```

Output:

```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/emotion.py
Emotion: happy | Text: I am so delighted to see you!
Emotion: nervous | Text: I feel very anxious about the exam.
Emotion: angry | Text: He was furious when he heard the news.
Emotion: neutral | Text: It's just an ordinary day.
(AIAC_env) PS D:\Course\AIAC>
```

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

Prompt:

```
'''
write a python program to grade students based on their marks
example
90-100 → A
• 80-89 → B
• 70-79 → C
• 60-69 → D
• Below 60 → F

'''
```

Code:

```
def grade_student(marks):
    if 90 <= marks <= 100:
        return 'A'
    elif 80 <= marks < 90:
        return 'B'
    elif 70 <= marks < 80:
        return 'C'
    elif 60 <= marks < 70:
        return 'D'
    elif 0 <= marks < 60:
        return 'F'
    else:
        return 'Invalid marks'  # Handle invalid input
# Example usage
if __name__ == "__main__":
    test_marks = [95, 82, 76, 64, 50, -10, 105]

    for marks in test_marks:
        grade = grade_student(marks)
        print(f"Marks: {marks} | Grade: {grade}")
```

Output:

```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/grade.
py
Marks: 95 | Grade: A
Marks: 82 | Grade: B
Marks: 76 | Grade: C
Marks: 64 | Grade: D
Marks: 50 | Grade: F
Marks: -10 | Grade: Invalid marks
Marks: 105 | Grade: Invalid marks
(AIAC_env) PS D:\Course\AIAC>
```

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

Prompt:

```
'''
write a python program that predicts a pesons zodiac sign beasd on birth
month

signs in 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
'''
```

Code:

```python
def predict_zodiac_sign(month):
    zodiac_signs = {
        'march': 'Mesha',
        'april': 'Vrishabha',
        'may': 'Mithuna',
        'june': 'Karka',
        'july': 'Simha',
        'august': 'Kanya',
        'september': 'Tula',
        'october': 'Vrischika',
        'november': 'Dhanu',
        'december': 'Makara',
        'january': 'Kumbha',
        'february': 'Meena'
    }

    month_lower = month.lower()
    return zodiac_signs.get(month_lower, 'Invalid month')  # Return sign or invalid message
# Example usage
if __name__ == "__main__":
    test_months = [
        "March",
        "July",
        "November",
        "January",
        "InvalidMonth"
    ]

    for month in test_months:
        sign = predict_zodiac_sign(month)
        print(f"Month: {month} | Zodiac Sign: {sign}")
```

Output:

```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/zodiac
.py
Month: March | Zodiac Sign: Mesha
Month: July | Zodiac Sign: Simha
Month: November | Zodiac Sign: Dhanu
Month: January | Zodiac Sign: Kumbha
Month: InvalidMonth | Zodiac Sign: Invalid month
(AIAC_env) PS D:\Course\AIAC>
```

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']

Prompt:

```
'''
write a python program to determine if a student has passed or failed
based on their marks.
The program should take the marks as input and print "Pass" if the marks
are 40 or above, and "Fail" if the marks are below 40.
'''
```

Code:

```
marks = float(input("Enter the marks: "))
if marks >= 40:
    print("Pass")
else:
    print("Fail")
```

Output:

```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/passOr
Fail.py
Enter the marks: 70
Pass
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/passOr
Fail.py
Enter the marks: 35
Fail
```

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.

Prompt:

```
'''
well commentd code
write a python program to determine if a person is eligible to vote based
on their age.
The program should take the age as input and print "Eligible to vote" if
the age is 18 or above, and "Not eligible to vote" if the age is below 18.
check for invalid inputs such as negative ages and non-integer values.
'''
```

Code:

```
try:
    age = int(input("Enter your age: "))
    if age < 0:
        print("Invalid input: Age cannot be negative.")
    elif age >= 18:
        print("Eligible to vote")
    else:
        print("Not eligible to vote")
except ValueError:
    print("Invalid input: Please enter a valid integer for age.")
```

Output:

```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/vote.py
Enter your age: 21
Eligible to vote
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/vote.py
Enter your age: 15
Not eligible to vote
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass4/vote.py
Enter your age: -21
Invalid input: Age cannot be negative.
(AIAC_env) PS D:\Course\AIAC>
```

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

Prompt:

```
palindrome.py > ...        D:\Course\AIAC\ass4\vote.py
'''
1. genarate a list of student names
2. check if the give name is plindrome or not
3. display the list of result

'''
```

Code:

```
def is_palindrome(name):
    # Function to check if a name is a palindrome
    return name == name[::-1]
student_names = []
n = int(input("Enter the number of students: "))
for _ in range(n):
    name = input("Enter student name: ")
    student_names.append(name)
for name in student_names:
    if is_palindrome(name):
        print(f"{name} is a palindrome")
    else:
        print(f"{name} is not a palindrome")
```

Output:

```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Script
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC
rome.py
Enter the number of students: 6
Enter student name: anna
Enter student name: bob
Enter student name: eshwar
Enter student name: naga
Enter student name: saras
Enter student name: otto
anna is a palindrome
bob is a palindrome
eshwar is not a palindrome
naga is not a palindrome
saras is a palindrome
otto is a palindrome
(AIAC_env) PS D:\Course\AIAC>
```

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

Prompt:

```
ass4 > length.py > ...
1   '''
2   1. generate a list of  random words
3   2. traverse words in the list and find the length
4   3. from above result check the word is short if less then 5 and long if
    more then 5 or eqaul to 5
5   4. display the result with each word
6   '''
```

D:\Course\AIAC\ass4\length.py

Code:

```
7   def word_length_classification(word):
8       # Function to classify word length
9       length = len(word)
0       if length < 5:
1           return "short"
2       else:
3           return "long"
4   words = ["apple", "bat", "carrot", "dog", "elephant"]
5   for word in words:
6       classification = word_length_classification(word)
7       print(f"The word '{word}' is {classification} (length: {len(word)})")
8
```

Output:

```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/
.py
The word 'apple' is long (length: 5)
The word 'bat' is short (length: 3)
The word 'carrot' is long (length: 6)
The word 'dog' is short (length: 3)
The word 'elephant' is long (length: 8)
(AIAC_env) PS D:\Course\AIAC>
```