

Assignment-4.1

HT-NO: 2303A51272

Batch: 23

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

```
#write a python code that classifies a given text is spam or not spam
text = input("Enter the text: ")
spam_keywords = ["win", "prize", "free", "money", "urgent", "click", "offer"]
text_lower = text.lower()
if any(keyword in text_lower for keyword in spam_keywords):
    print("The text is classified as SPAM.")
else:
    print("The text is classified as NOT SPAM.")

sisted code/xyz.py
Enter the text: Congratulations! You have won a free lottery ticket.
The text is classified as SPAM.
○ PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code>
```

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 labelled example inside the prompt.
2. Take a sentence as input.
3. Print the predicted emotion

```
"""
Sentence: "I miss my friends and feel really down."
Emotion: sad
"""

def analyze_emotion(sentence):
    if "miss" in sentence or "down" in sentence or "lonely" in sentence:
        return "sad"
    elif "happy" in sentence or "joy" in sentence or "excited" in sentence:
        return "happy"
    elif "angry" in sentence or "mad" in sentence or "furious" in sentence:
        return "angry"
    else:
        return "neutral"
emotion = analyze_emotion("I am excited about my new job!")
print(f"The emotion of the sentence is: {emotion}")

PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code & C:/Users/sriva/AppData/Local/Programs/Python/3.9.10/xyz.py
The emotion of the sentence 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

```

"""
Marks: 95
Grade: A
Marks: 82
Grade: B
Marks: 76
Grade: C
Marks: 63
Grade: D
Marks: 50
Grade: F
"""

def get_grade(marks):
    if marks >= 90:
        return 'A'
    elif marks >= 80:
        return 'B'
    elif marks >= 70:
        return 'C'
    elif marks >= 60:
        return 'D'
    else:
        return 'F'
marks = int(input("Enter marks: "))
grade = get_grade(marks)
print(f'Marks: {marks}\nGrade: {grade}')

```

```

Enter marks: 55
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

```
1  """
2  Month: March
3  Rashi: Mesha
4  Month: April
5  Rashi: Vrishabha
6  Month: May
7  Rashi: Mithuna
8  Month: June
9  Rashi: Karka
10 Month: July
11 Rashi: Simha
12 Month: August
13 Rashi: Kanya
14 Month: September
15 Rashi: Tula
16 display indian zodiac sign for given month
17 handle invalid input
18 """
19 def get_indian_zodiac_sign(month):
20     month = month.lower()
21     zodiac_signs = {
22         "march": "Mesha",
23         "april": "Vrishabha",
24         "may": "Mithuna",
25         "june": "Karka",
26         "july": "Simha",
27         "august": "Kanya",
28         "september": "Tula",
29         "october": "Vrischika",
30         "november": "Dhanu",
31         "december": "Makara",
32         "january": "Kumbha",
33         "february": "Meena"
34     }
35
36     return zodiac_signs.get(month, "Invalid month name")
37 # Example usage
38 month_input = input("Enter month name: ")
39 print(f"Month: {month_input}")
40 print(f"Rashi: {get_indian_zodiac_sign(month_input)}")
```

```
Enter month name: january
```

```
Month: january
```

```
Rashi: Kumbha
```

```
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> █
```

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']

```
1  """
2  read marks of students(range 0-100)
3  check if marks are above 40
4  if yes display "pass"
5  else display "fail"
6  handle invalid input only take positive integers within range
7  """
8
9  marks=int(input("Enter the marks of the student : "))
10 def check_pass_fail(marks):
11     if 0 <= marks <= 100:
12         if marks > 40:
13             return "pass"
14         else:
15             return "fail"
16     else:
17         return "Invalid input. Please enter marks in the range 0-100."
18 result = check_pass_fail(marks)
19 print(result)
```

- PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code & C:/Users/sriva/Documents/AI Assisted Code/xyz.py"
Enter the marks of the student : 99
pass
- PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code & C:/Users/sriva/Documents/AI Assisted Code/xyz.py"
Enter the marks of the student : -99
Invalid input. Please enter marks in the range 0-100.
- PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code

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.

```
1  ...
2  Read the age.
3  If the age is 18 or older, print "Eligible".
4  If the age is less than 18, print "Not Eligible".
5  Take care of invalid input (negative age and non-integer input).
6  ...
7  age_input = input("Enter your age: ")
8  try:
9      age = int(age_input)
10     if age < 0:
11         |   print("Invalid input. Age cannot be negative.")
12     elif age >= 18:
13         |   print("Eligible")
14     else:
15         |   print("Not Eligible")
16 except ValueError:
17     |   print("Invalid input. Please enter a valid integer.")
```

Enter your age: 20

Eligible

- PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> & /Documents/AI Assisted Code/xyz.py"

Enter your age: -9

Invalid input. Age cannot be negative.

- PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> & /Documents/AI Assisted Code/xyz.py"

Enter your age: abbb

Invalid input. Please enter a valid integer.

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.

```
1  ...
2  Read student names from user
3  If name is palindrome store it in a list
4  Handle case sensitivity
5  Handle invalid inputs
6  ...
7  student_names = input("Enter student names : ").split()
8  palindrome_names = []
9  for name in student_names:
10     name = name.strip()
11     if not name.isalpha():
12         print(f"Invalid input: {name}. Please enter valid names.")
13         continue
14     lower_name = name.lower()
15     if lower_name == lower_name[::-1]:
16         palindrome_names.append(name)
17 print("Palindrome names:", palindrome_names)
```

```
Enter student names : anna chitti srivani
```

```
Palindrome names: ['anna']
```

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

```
1  ...
2  Generate a list of words
3  Count the length of each word and store it in a variable word_lengths
4  If word_length < 5 display as short
5  If word_length ≥ 5 display as long
6  Display the result for each word
7  ...
8  words = ["apple", "bat", "elephant", "cat", "dolphin", "ant", "giraffe"]
9  for word in words:
10     word_length = len(word)
11     if word_length < 5:
12         print(f"{word}: short")
13     else:
14         print(f"{word}: long")
15
```

apple: long
bat: short
elephant: long
cat: short
dolphin: long
ant: short
giraffe: long