

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

```
#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 compatibility.
port-Module PSReadLine'.

C:\Users\yadav\OneDrive\Desktop\PYTHON all> & C:/Users/yadav/AppData/Local/Microsoft/WindowsApps/python3.8-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

```
7 Write a Python program that detects the emotion of a sentence
8 Emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
9 example:
10 Input: "I am so happy today!
11 Output: "happy"
12 ...
13
14 def detect_emotion(sentence):
15     """Detect the emotion of a sentence
16     Args:
17         sentence (str): Input sentence to analyze
18     Returns:
19         str: Detected emotion from the sentence
20     """
21
22     emotions = {
23         'happy': ['happy', 'joyful', 'elated', 'pleased', 'content'],
24         'sad': ['sad', 'unhappy', 'downcast', 'dejected', 'sorrowful'],
25         'angry': ['angry', 'mad', 'furious', 'irate', 'annoyed'],
26         'excited': ['excited', 'thrilled', 'eager', 'enthusiastic', 'overjoyed'],
27         'nervous': ['nervous', 'anxious', 'uneasy', 'apprehensive', 'tense'],
28         'neutral': ['neutral', 'indifferent', 'unemotional', 'calm', 'composed']
29     }
30     sentence_lower = sentence.lower()
31     for emotion, keywords in emotions.items():
32         for keyword in keywords:
33             if keyword in sentence_lower:
34                 return emotion
35     return "neutral" # Default to neutral if no emotion is detected
36
37 # Example usage
38 input_sentence = "I am so happy today!"
39 detected_emotion = detect_emotion(input_sentence)
40 print(f"The detected emotion is: {detected_emotion}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

!-lab.py"
? 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

```
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

```
284 """
285 Write a Python program that predicts a person's Indian Zodiac sign
286 (Rashi) based on the month of birth (month name) using multi-shot
287 prompting.
288 Indian Zodiac Order (Simplified Month-Based Model)
289
290 March → Mesha
291 April → Vrishabha
292 May → Mithuna
293 June → Karka
294 July → Simha
295 August → Kanya
296 September → Tula
297 October → Vrischika
298 November → Dhanu
299 December → Makara
300 January → Kumbha
301 February → Meena
302 ...
303 def predict_rashi(month):
304     """Predict the Indian Zodiac sign (Rashi) based on the month of birth
305     Args:
306         month (str): Month of birth
307     Returns:
308         str: Corresponding Rashi
309     """
310     rashi_dict = {
311         'March': 'Mesha',
312         'April': 'Vrishabha',
313         'May': 'Mithuna',
314         'June': 'Karka',
315         'July': 'Simha',
316         'August': 'Kanya',
317         'September': 'Tula',
318         'October': 'Vrischika',
319         'November': 'Dhanu',
320         'December': 'Makara',
321         'January': 'Kumbha',
322         'February': 'Meena'
323     }
324     return rashi_dict.get(month, "Invalid month name")
325 # Example usage
326 input_month = input("Enter the month of birth: ")
327 predicted_rashi = predict_rashi(input_month)
328 print(f"The predicted Rashi is: {predicted_rashi}")
```

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     handle 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/Windows/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.

```
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.

```

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
Palindrom 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
Palindrom 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

```
405     '''generate a list of words named words_list
406     traverse the list and calculate the length of each word.
407     if length is less than 5 make a list and add as "short" to list
408     if length greater than 7 add as "long" to list '''
409     def classify_word_lengths():
410         """Classify words based on their lengths
411         Returns:
412             list: List containing "short" or "long" based on word lengths
413         """
414         words_list = ["apple", "banana", "kiwi", "strawberry", "fig", "watermelon", "grape", "blueberry", "pear", "pineapple"]
415         length_classification = []
416         for word in words_list:
417             if len(word) < 5:
418                 length_classification.append("short")
419             elif len(word) > 7:
420                 length_classification.append("long")
421         return length_classification
422     # Example usage
423     classification_result = classify_word_lengths()
424     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']"
```