

ASSIGNMENT-4.1

Name: G.charanya

Batch:23

Ht. No:2303A510G6

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

```
PS C:\Users\2303a\OneDrive\Desktop\AI> (C:\Users\2303a\miniconda3\scripts\activate) ; (conda activate base)
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
Enter the text to check for spam: "Congratulations! You have won a free lottery ticket"
The given text is spam.
PS C:\Users\2303a\OneDrive\Desktop\AI>
```

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

The screenshot shows a code editor interface with a dark theme. On the left, there are several small icons representing different file types or tools. The main area contains the following Python code:

```
13 |     print("The given text is not spam.")"""
14 |
15 # write a python program that detects emotion of a person
16 """ sentence: I am very joyful today
17 |     | output: happy"""
18 def detect_emotion(text):
19     text_lower = text.lower()
20     if "happy" in text_lower or "joyful" in text_lower or "excited" in text_lower:
21         return "happy"
22     elif "sad" in text_lower or "unhappy" in text_lower or "depressed" in text_lower:
23         return "sad"
24     elif "angry" in text_lower or "mad" in text_lower or "frustrated" in text_lower:
25         return "angry"
26     else:
27         return "neutral"
28
29 text = input("Enter a text: ")
30 emotion = detect_emotion(text)
31 print("Detected emotion:", emotion)
32
```

Below the code, there is a terminal window showing the execution of the script. The terminal tabs at the top are PROBLEMS (1), OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS. The terminal output is as follows:

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Enter the text to check for spam: "Congratulations! You have won a free lottery ticket
The given text is spam.
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
Enter a text: something feel of
Detected emotion: neutral
PS C:\Users\2303a\OneDrive\Desktop\AI>
```

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

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left, there are several icons for file operations like search, copy, paste, etc. The main area contains the following Python code:

```
ASS_4.1.py > ...
33     ''' marks-90-100
34     display "a"
35     marks-80-89
36     display "b"
37     marks-70-79
38     display "c"
39     marks-60-69
40     display "d"
41     marks-below 60
42     display "f" '''
43
44 def students_grade(marks):
45     if 90 <= marks <= 100:
46         return "a"
47     elif 80 <= marks <= 90:
48         return "b"
49     elif 70 <= marks <= 80:
50         return "c"
51     elif 60 <= marks <= 70:
52         return "d"
53     elif marks < 60:
54         return "f"
55     else:
56         return "Invalid marks"
57 marks = int(input("Enter the marks: "))
58 print("The grade is:", students_grade(marks))
```

Below the code, there are tabs for PROBLEMS (1), OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing command-line output:

```
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
Enter a text: something feel of
Detected emotion: neutral
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
Enter the marks: 52
The grade is: f
PS C:\Users\2303a\OneDrive\Desktop\AI>
```

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

The screenshot shows a code editor interface with several tabs at the top: 'day_3', 'ASS_1', 'Assignment_3.5.py', '28-1-26', and 'ASS_4'. The main area displays Python code. The code defines a dictionary 'zodiac_signs' that maps months to zodiac signs. It then uses this dictionary to find the zodiac sign for the month 'december'. Finally, it prints the result. The code is as follows:

```
59
60     ''' march -> mesha
61     april -> vrishabha
62     may -> mithuna
63     june -> karkata
64     july -> simha
65     august -> kanya
66     september -> tula
67     october -> vrischika
68     november -> dhanu
69     december -> makara
70     january -> kumbha
71     february -> meena'''

72
73 def month_of_zodiac(month):
74     month = month.lower()
75     zodiac_signs = {
76         'march': 'mesha',
77         'april': 'vrishabha',
78         'may': 'mithuna',
79         'june': 'karkata',
80         'july': 'simha',
81         'august': 'kanya',
82         'september': 'tula',
83         'october': 'vrischika',
84         'november': 'dhanu',
85         'december': 'makara',
86         'january': 'kumbha',
87         'february': 'meena'
88     }
89     return zodiac_signs.get(month, "Invalid month name")
90 month = input("Enter the month name: ")
91 print("The zodiac sign is:", month_of_zodiac(month))
```

Below the code editor, there are tabs for 'PROBLEMS' (with 1 error), 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined), and 'PORTS'. The terminal window shows the user entering 'december' and then receiving the output 'The zodiac sign is: makara'.

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']

```

94     """ read marks of students from range 0-100
95     check if marks are greater than 40
96     if yes display "pass"
97     if no display "fail" """
98
99     def check_student_marks(marks):
100        if 0 <= marks <= 100:
101            if marks > 40:
102                return "pass"
103            else:
104                return "fail"
105        else:
106            return "Invalid marks. Please enter a value between 0 and 100."
107 marks = int(input("Enter the marks of the student (0-100): "))
108 print(check_student_marks(marks))

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

$ C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
Enter the marks of the student (0-100): 75
pass
$ C:\Users\2303a\OneDrive\Desktop\AI>

```

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.

```

109
110     """ read the age of person from range 1-100
111     check if age is greater than or equal to 18
112     if yes print "You are eligible to vote"
113     otherwise print "not eligible to vote"
114     """
115
116     age = int(input("Enter your age: "))
117     if age >= 18:
118         print("You are eligible to vote")
119     else:
120         print("not eligible to vote")

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\2303d\OneDrive\Desktop\AI> & C:\Users\2303d\miniconda3\python.exe C:/Users/2303d/OneDrive/Desktop/AI/ASS_4.1.py
pass
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
● Enter your age: 20
○ You are eligible to vote
○ PS C:\Users\2303a\OneDrive\Desktop\AI>

```

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.

```

120
121     """ read student names from user
122     if name is palindrome store it in a list
123     handle case sensitivity
124     handle invalid inputs
125     display list of palindromic names
126     """
127     def is_palindrome(name):
128         name = name.strip()
129         if not name.isalpha():
130             return False
131         name_lower = name.lower()
132         return name_lower == name_lower[::-1]
133     palindrome_names = []
134     while True:
135         name = input("Enter student name (or type 'exit' to finish): ")
136         if name.lower() == 'exit':
137             break
138         if i (variable) palindrome_names: list
139             palindrome_names.append(name.strip())
140     print("Palindromic names:", palindrome_names)
141
142

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/0
You are eligible to vote
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/0
> Enter student name (or type 'exit' to finish): charanya
> Enter student name (or type 'exit' to finish): ganesh
> Enter student name (or type 'exit' to finish): exit
> Palindromic names: []
> PS C:\Users\2303a\OneDrive\Desktop\AI>

```

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.

```
41
42
43     """ read words from user
44     count the length of each and store it in a variable
45     if variable <5 display as short
46     otherwise display as long
47     display the result of each word
48     """
49
50     def classify_word_length(word):
51         if len(word) < 5:
52             return "short"
53         else:
54             return "long"
55     words = input("Enter words separated by spaces: ").split()
56     for word in words:
57         length_classification = classify_word_length(word)
58         print(f"The word '{word}' is {length_classification}")
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
S C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
S C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
Enter words separated by spaces: dad
the word 'dad' is short
S C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/ASS_4.1.py
Enter words separated by spaces: dad
the word 'dad' is short
S C:\Users\2303a\OneDrive\Desktop\AI>
```