

Lab Assignment -4

N. Soukya

Batch-24

2303A51401

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:

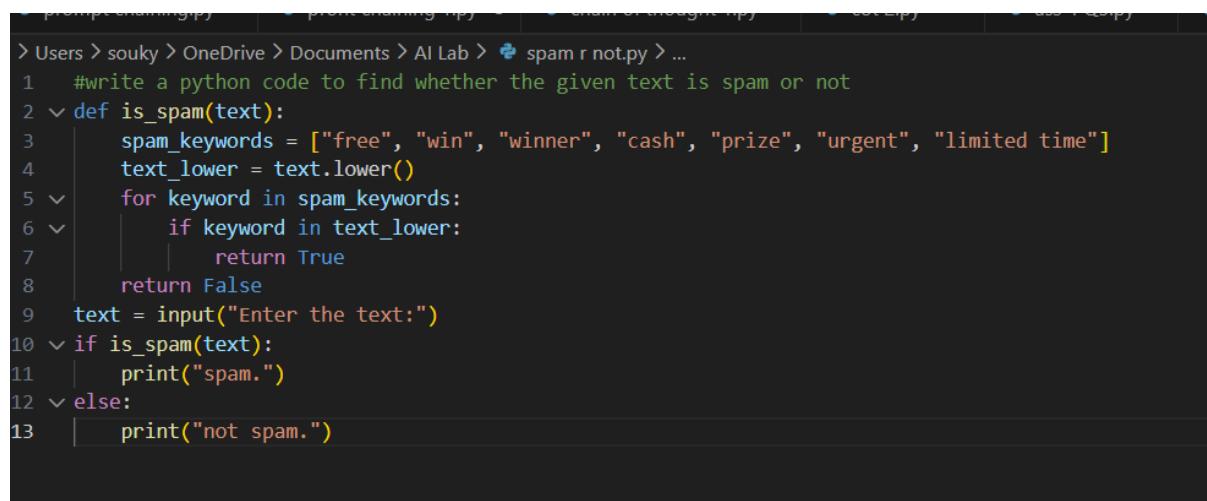
1. Construct a prompt without any examples.
2. Clearly specify the output labels.
3. Display only the predicted label.

Input:

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

Expected Output:

Spam



```
> Users > souky > OneDrive > Documents > AI Lab > spam_r_not.py > ...
1  #write a python code to find whether the given text is spam or not
2  def is_spam(text):
3      spam_keywords = ["free", "win", "winner", "cash", "prize", "urgent", "limited time"]
4      text_lower = text.lower()
5      for keyword in spam_keywords:
6          if keyword in text_lower:
7              return True
8      return False
9  text = input("Enter the text:")
10 if is_spam(text):
11     print("spam.")
12 else:
13     print("not spam.")
```

```
PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-6
Enter the text:Congratulations! You have won a free lottery ticket."
spam.
PS C:\Users\souky>
```

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

```
Users > souky > OneDrive > Documents > AI Lab > emtion.py > ...
#write a python code that detects emotion of a person
...
sentence: I am happy
output: happy
...

def detect_emotion(text):
    text_lower = text.lower()
    if "happy" in text_lower or "joyful" in text_lower or "excited" in text_lower:
        return "happy"
    elif "sad" in text_lower or "unhappy" in text_lower or "depressed" in text_lower:
        return "sad"
    elif "angry" in text_lower or "mad" in text_lower or "furious" in text_lower:
        return "angry"
    else:
        return "neutral"
input_text = "I am happy"
emotion = detect_emotion(input_text)
print(f"The detected emotion is: {emotion}")
```

```
PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-64/python.
The detected emotion is: happy
PS C:\Users\souky> 
```

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

```

Users > souky > OneDrive > Documents > AI Lab > grde.py > determine_grade
#write a python code displaying the grade based on the marks obtained.ex:marks=92, grade=A
#marks=85 -grade=B, marks=76 -grade=C
def determine_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'
try:
    user_input = int(input("Enter the marks obtained: "))
    grade = determine_grade(user_input)
    print(f"The grade based on the marks {user_input} is: {grade}")
except ValueError:
    print("Please enter a valid integer for marks.")
except Exception as e:
    print(f"An error occurred: {e}")

```

```

PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-6
Enter the marks obtained: 0
The grade based on the marks 0 is: F
PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-6
Enter the marks obtained: 45
The grade based on the marks 45 is: F
PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-6
Enter the marks obtained: 96
The grade based on the marks 96 is: A
PS C:\Users\souky>

```

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

```
Users > souky > OneDrive > Documents > AI Lab > zodiac.py > ...
#generate a python code that displays a person's Indian Zodiac sign(Rashi) based on the month of birth
'''ex:March → Mesha
April → Vrishabha
May → Mithuna
June → Karka
July → Simha
August → Kanya'''
def get_indian_zodiac_sign(month):
    zodiac_signs = {
        "january": "Makara",
        "february": "Kumbha",
        "march": " Mesha",
        "april": "Vrishabha",
        "may": "Mithuna",
        "june": "Karka",
        "july": "Simha",
        "august": "Kanya",
        "september": "Tula",
        "october": "Vrischika",
        "november": "Dhanu",
        "december": "Makara"
    }
    month_lower = month.lower()
    return zodiac_signs.get(month_lower, "Invalid month name")
try:
    user_input = input("Enter the month of birth:")
    zodiac_sign = get_indian_zodiac_sign(user_input)
    print(f"The Indian zodiac sign for the month {user_input} is: {zodiac_sign}")
except Exception as e:
    print(f"An error occurred: {e}")
```

- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-64/python.exe
 Enter the month of birth:august
 The Indian zodiac sign for the month {user_input} is: {zodiac_sign}
- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-64/python.exe
 Enter the month of birth:february
 The Indian zodiac sign for the month {user_input} is: {zodiac_sign}
- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore-3.14-64/python.exe
 Enter the month of birth:january
 The Indian zodiac sign for the month {user_input} is: {zodiac_sign}
- PS C:\Users\souky> []
 0 ▲ 0

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']

```
Binary to decimal.py   F shot D to G.py   Harshad.py   student grade.py
C: > Users > souky > OneDrive > Documents > AI Lab > ass 4 Q5.py > ...
1 ...
2 read marks of students from range 0-100
3 check if marks are greater than 40
4 if yes print pass otherwise print fail
5 ...
6 def check_pass_fail(marks):
7     if 0 <= marks <= 100:
8         if marks > 40:
9             return "Pass"
10        else:
11            return "Fail"
12    else:
13        return "Invalid marks"
14 marks = int(input("Enter marks (0-100): "))
15 print(check_pass_fail(marks))
```

- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Programs/Python/Python310/python.exe student grade.py
Enter marks (0-100): 60
Pass
- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Programs/Python/Python310/python.exe student grade.py
Enter marks (0-100): 35
Fail
- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Programs/Python/Python310/python.exe student grade.py
Enter marks (0-100): 0
Fail
- PS C:\Users\souky>

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.

```
> Users > souky > OneDrive > Documents > AI Lab > cot age.py > ...
1 ...
2 read the age of person from range 1-100
3 check if age is greater than or equal to 18
4 if yes,print eligible to vote
5 otherwise print not eligible
6 ...
7
8 age = int(input("Enter the age of the person (1-100): "))
9 if age >= 18:
10    print("Eligible to vote")
11 else:
12    print("Not eligible to vote")
```

- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore
Enter the age of the person (1-100): 16
Not eligible to vote
- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore
Enter the age of the person (1-100): 20
Eligible to vote
- PS C:\Users\souky>

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.

```
C: > Users > souky > OneDrive > Documents > AI Lab > ✎ palindromic names.py > ...
1  ''
2  read student names from user and store in a list of student names
3  if name is palindrome store it in a list
4  handle case sensitivity
5  handle invalid inputs
6  display list of palindrome names
7  ''
8  def is_palindrome(name):
9      name = name.strip()
10     if not name.isalpha():
11         return False
12     name_lower = name.lower()
13     return name_lower == name_lower[::-1]
14 palindrome_names = []
15 while True:
16     name = input("Enter a student name (or 'quit' to stop): ")
17     if name.lower() == 'quit':
18         break
19     if is_palindrome(name):
20         palindrome_names.append(name)
21 print("Palindrome names:", palindrome_names)
```

- PS C:\Users\souky> & C:/Users/souky/AppData/Local/Python/pythoncore
Enter a student name (or 'quit' to stop): soukya
Enter a student name (or 'quit' to stop): madam
Enter a student name (or 'quit' to stop): harini
Enter a student name (or 'quit' to stop): navya
Enter a student name (or 'quit' to stop): mom
Enter a student name (or 'quit' to stop): quit
Palindrome names: ['madam', 'mom']
- PS C:\Users\souky>

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

```
Users > souky > OneDrive > Documents > AI Lab > Ass4 q8.py > ...
...
read words from user
count the length of each and store it in a variable
if variables <5 display as short
otherwise display as long
display the result of each word
...
def classify_word_length(word):
    length = len(word)
    if length < 5:
        return "short"
    else:
        return "long"
words = input("Enter words separated by spaces: ").split()
for word in words:
    classification = classify_word_length(word)
    print(f"{word}: {classification}")
```

- ▶ PS C:\Users\souky> & c:/Users/souky/AppData/Local/Python/pyt
Enter words separated by spaces: apple
apple: long
- ▶ PS C:\Users\souky> & c:/Users/souky/AppData/Local/Python/pyt
Enter words separated by spaces: rice
rice: short
- ▶ PS C:\Users\souky>