

Lab Assignment-4.1

Name : G.Anoop Goud

Hallticket:2303A51085

Batch-02

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."

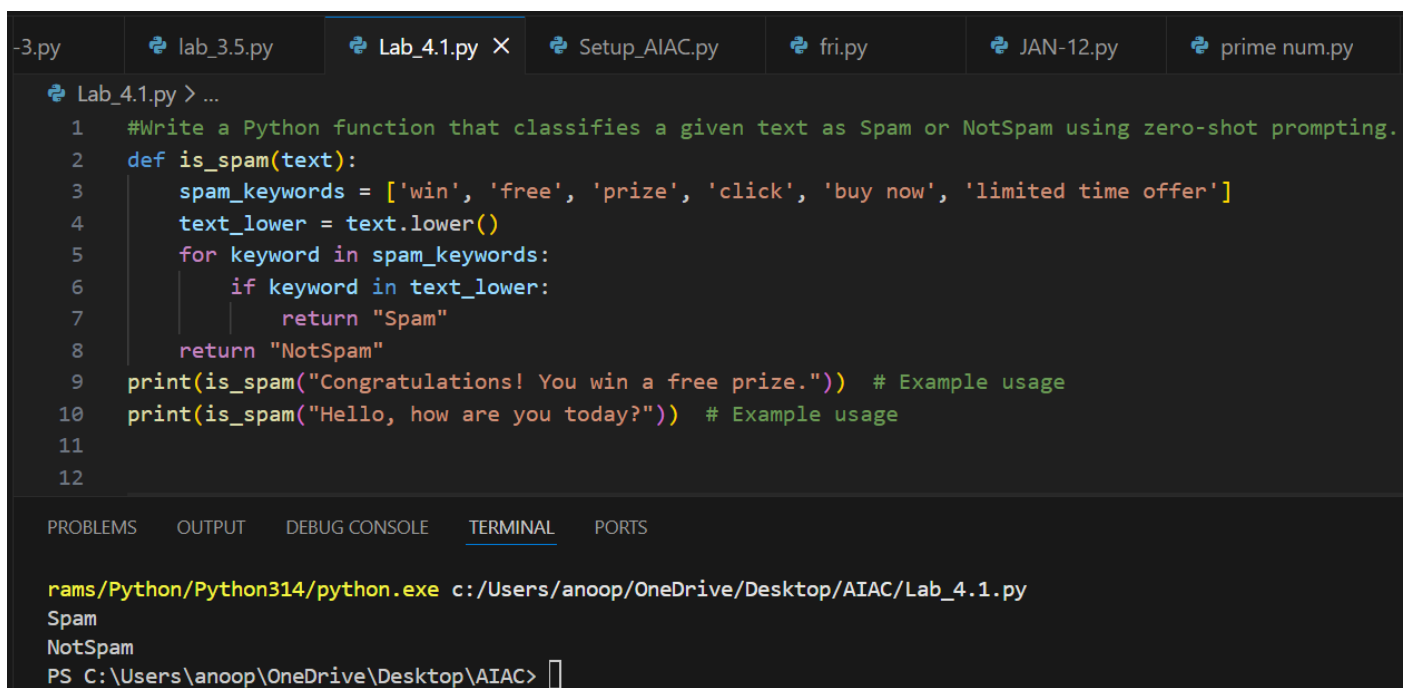
Expected Output:

Spam

Prompt:

#write a python function that classifies a given test as spam or not spam based on the presence of certain keywords

Code:



```
-3.py  lab_3.5.py  Lab_4.1.py X  Setup_AIAC.py  fri.py  JAN-12.py  prime num.py

Lab_4.1.py > ...
1  #Write a Python function that classifies a given text as Spam or NotSpam using zero-shot prompting.
2  def is_spam(text):
3      spam_keywords = ['win', 'free', 'prize', 'click', 'buy now', 'limited time offer']
4      text_lower = text.lower()
5      for keyword in spam_keywords:
6          if keyword in text_lower:
7              return "Spam"
8      return "NotSpam"
9  print(is_spam("Congratulations! You win a free prize.")) # Example usage
10 print(is_spam("Hello, how are you today?")) # Example usage
11
12

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

rams/Python/Python314/python.exe c:/Users/anoop/OneDrive/Desktop/AIAC/Lab_4.1.py
Spam
NotSpam
PS C:\Users\anoop\OneDrive\Desktop\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:

emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']

write a python program that detect the mood of the person and take the sentence from the user if any word from the emotions list is present in the sentence then display that emotion otherwise display no emotion detected

Code:

```

#write a python program that detect the mood of the person and take the sentence from
Emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
def detect_emotion(text):
    emotions = {
        'happy': ['joy', 'pleased', 'content', 'cheerful'],
        'sad': ['unhappy', 'sorrowful', 'dejected', 'downcast'],
        'angry': ['mad', 'furious', 'irate', 'enraged'],
        'excited': ['thrilled', 'elated', 'overjoyed', 'eager'],
        'nervous': ['anxious', 'tense', 'apprehensive', 'worried'],
        'neutral': ['calm', 'unemotional', 'indifferent', 'composed']
    }
    text_lower = text.lower()
    for emotion, keywords in emotions.items():
        for keyword in keywords:
            if keyword in text_lower:
                return emotion
    return "neutral"
print(detect_emotion("I am so thrilled about the trip!")) # Example usage
print(detect_emotion("I feel very anxious about the exam. ")) # Example usage

```

Python/Python314/python.exe c:/Users/anoop/OneDrive/Desktop/AIAC/Lab_4.1.py

```

am
\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AppData/Local/Programs/Python/Python3
ed
us
\Users\anoop\OneDrive\Desktop\AIAC> 

```

```

emotions = ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
user_input = input("Enter a sentence to detect emotion: ").lower()
detected_emotions = [emotion for emotion in emotions if emotion in user_input]
if detected_emotions:
    print(f"Detected emotions: {' '.join(detected_emotions)}")
else:
    print("No emotion detected.")

```

input/output:

```

Enter a sentence to detect emotion: iam happy today
Detected emotions: 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

Prompt:

90-100=A

80-89=B

70-79=C

60-69=D

below 60=F

1.we will give a list of marks

2.we will assign grades based on the marks

3.we will store the grades in a new list

4.finally we will print the list of grades

```
3.py lab_3.5.py Lab_4.1.py X Setup_AIAC.py fri.py
Lab_4.1.py > assign_grades
70
71 '''
72 1.we will give a list of marks
73 2.we will assign grades based on the marks
74 3.we will store the grades in a new list
75 4.finally we will print the list of grades
76 '''
77 def assign_grades(marks):
78     grades = []
79     for mark in marks:
80         if mark >= 90:
81             grades.append('A')
82         elif mark >= 80:
83             grades.append('B')
84         elif mark >= 70:
85             grades.append('C')
86         elif mark >= 60:
87             grades.append('D')
88         else:
89             grades.append('F')
90     return grades
91 marks_list = [95, 82, 67, 74, 58, 89]
92 grades_list = assign_grades(marks_list)
93 print(grades_list) # Example usage
94
95 '''
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AppData/Local/
File "c:\Users\anoop\OneDrive\Desktop\AIAC\Lab_4.1.py", line 77
    assign_grades(marks):
IndentationError: unexpected indent
PS C:\Users\anoop\OneDrive\Desktop\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:

march=mesha

april=vrishabha

may=mithuna

june=karka

july=simha

august=kanya

september=tula

october=vrischika

november=dhanus

december=makara

january=kumbha

february=meena

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

```
111
112     '''
113     march=mesha
114     april=vrishabha
115     may=mithuna
116     june=karka
117     july=simha
118     august=kanya
119     september=tula
120     october=vrishchika
121     november=dhanus
122     december=makara
123     january=kumbha
124     february=meena
125     write a python code to accept month from the user and display t
126
127     '''
128     '''def get_zodiac_sign(month):
```

Code:

```
py lab_3.5.py Lab_4.1.py X Setup_AIAC.py fri.py JAN-1
Lab_4.1.py > get_rashi
13
14 '''Write a Python program that predicts a person's Indian Zodiac sign
15 (Rashi) based on the month of birth (month name) using multi-shot
16 prompting'''
17 def get_rashi(month):
18     rashi_dict = {
19         'January': 'Makar (Capricorn)',
20         'February': 'Kumbh (Aquarius)',
21         'March': 'Meen (Pisces)',
22         'April': 'Mesh (Aries)',
23         'May': 'Vrishabh (Taurus)',
24         'June': 'Mithun (Gemini)',
25         'July': 'Kark (Cancer)',
26         'August': 'Singh (Leo)',
27         'September': 'Kanya (Virgo)',
28         'October': 'Tula (Libra)',
29         'November': 'Vrishchik (Scorpio)',
30         'December': 'Dhanu (Sagittarius)'
31     }
32     return rashi_dict.get(month, "Invalid month name")
33 print(get_rashi("March")) # Example usage
34 print(get_rashi("August")) # Example usage
35
36 '''#write a python program that detect the mood of the person and take the
37 Emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']'''

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

IndentationError: unexpected indent
PS C:\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AppData/Local/Programs/Python
Meen (Pisces)
Singh (Leo)
PS C:\Users\anoop\OneDrive\Desktop\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:

take marks as an input from the user

if marks is greater than or equal to 40 then print pass

if marks is less than 40 then print fail

```
93 """
94 take marks as an input from the user
95 if marks is greater than or equal to 40 then print pass
96 if marks is less than 40 then print fail
97
98 """
99 """try:
   marks = int(input("Enter the marks: "))
   if marks < 0:
       print("Invalid input. Please enter a positive integer for marks.")
   else:
       if marks >= 40:
           print("Pass")
       else:
           print("Fail")
except ValueError:
    print("Invalid input. Please enter a valid integer for marks.")"""
100
```

```
"""
take marks as an input from the user
if marks is greater than or equal to 40 then print pass
if marks is less than 40 then print fail

"""
try:
    marks = int(input("Enter the marks: "))
    if marks < 0:
        print("Invalid input. Please enter a positive integer.")
    else:
        if marks >= 40:
            print("Pass")
        else:
            print("Fail")
except ValueError:
    print("Invalid input. Please enter a valid integer.")
```

Code:

try:

```
marks = int(input("Enter the marks: "))
```

```
if marks < 0:
```

```
    print("Invalid input. Please enter a positive integer.")
```

else:

if marks >= 40:

print("Pass")

else:

print("Fail")

except ValueError:

print("Invalid input. Please enter a valid integer.")

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:

take age from the user

if age is equal or greater than 18

then print eligible to vote

if age is less than 18

print not eligible to vote

prompt:

```
'''
1.We will give input as age in years
2.check whether age is integer and positive
3.based on age we will decide he can vote or not
4.finally we will print the result'''
```

Code:

```

def can_vote(age):
    if not isinstance(age, int) or age < 0:
        return "Invalid input. Please enter a positive integer."
    if age >= 18:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"

print(can_vote(20)) # Example usage
print(can_vote(16)) # Example usage
print(can_vote(-5)) # Example usage

```

try:

```

age = int(input("Enter your age: "))

if age < 0:
    print("Invalid input. Please enter a positive integer.")
else:
    if age >= 18:
        print("Eligible to vote")
    else:
        print("Not eligible to vote")

except ValueError:
    print("Invalid input. Please enter a valid integer.")

```

input/output:

```

Enter your age: 18
Eligible to vote
P5 C:\Users\arell\Music\aiac>
_044.py"
Enter your age: 17
Not eligible to vote

```

Task: Write a Python program that uses the prompt chaining technique to identify palindrome names from a list of student names.

Prompt:

take list of names form the user

if student names raa palindrome

then print those names in the form of list

```
130 """
131 take lidt of names form the user
132 if student names raa palindrome
133 then print those names in the form of list
134
135 """
136 def is_palindrome(name):
137
138
```

```
129
130 """
131 take lidt of names form the user
132 if student names raa palindrome
133 then print those names in the form of list
134
135 """
136 def is_palindrome(name):
137     return name == name[::-1]
138 students = input("Enter student names separated by commas: ").split(",")
139 palindrome_students = [name.strip() for name in students if is_palindrome(name.strip())]
140 print("Palindrome names:", palindrome_students)
141
142 |
143
144
```

Code:

```
def is_palindrome(name):
```

```
    return name == name[::-1]
```

```
students = input("Enter student names separated by commas: ").split(",")
```

```
palindrome_students = [name.strip() for name in students if is_palindrome(name.strip())]
```

```
print("Palindrome names:", palindrome_students)
```

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:

take list of words from the user

if the length of the individual word is greater than 5 then the the word is longs word

else the word is short

print the longs words and short words in the form of list

```
142 """
143 take list of words from the user
144 if the length of the individual word is greater than 5 then the the word is longs word
145 else the word is short
146 print the longs words and short words in the form of list
147 """
148 """words = input("Enter words separated by commas: ").split(",")
149 long_words = [word.strip() for word in words if len(word.strip()) > 5]
150
151
```

```
142 """
143 take list of words from the user
144 if the length of the individual word is greater than 5 then the the word is longs word
145 else the word is short
146 print the longs words and short words in the form of list
147 """
148 words = input("Enter words separated by commas: ").split(",")
149 long_words = [word.strip() for word in words if len(word.strip()) > 5]
150 short_words = [word.strip() for word in words if len(word.strip()) <= 5]
151 print("Long words:", long_words)
152 print("Short words:", short_words)
153 |
154
```

Code:

```
words = input("Enter words separated by commas: ").split(",")
```

```
long_words = [word.strip() for word in words if len(word.strip()) > 5]
```

```
short_words = [word.strip() for word in words if len(word.strip()) <= 5]
```

```
print("Long words:", long_words)
```

```
print("Short words:", short_words)
```

