

Lab Assignment-4.1

Name: M. Hasini

Ht.no: 2303A51109

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:

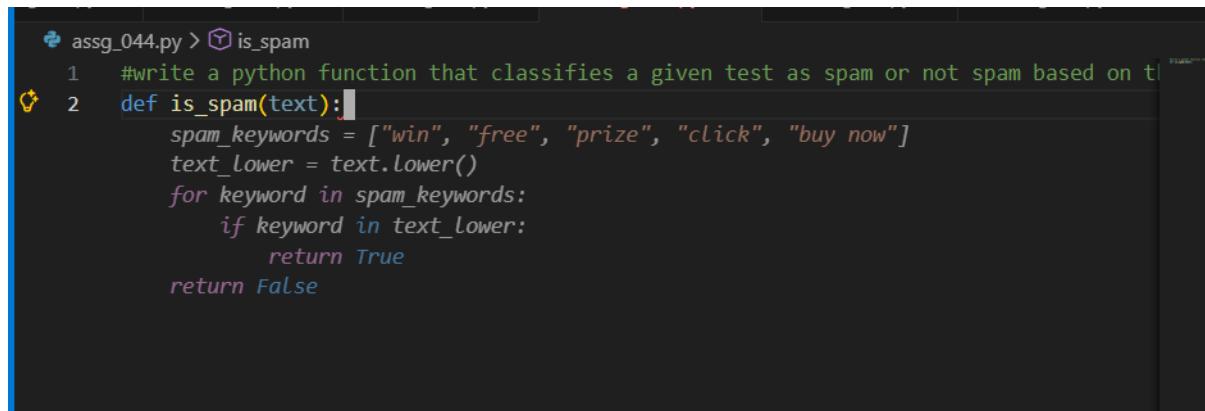
"Congratulations! You have won a free lottery ticket."

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
```



The screenshot shows a terminal window with the following content:

```
assg_044.py > is_spam
1 #write a python function that classifies a given test as spam or not spam based on the presence of certain keywords
2 def is_spam(text):
    spam_keywords = ["win", "free", "prize", "click", "buy now"]
    text_lower = text.lower()
    for keyword in spam_keywords:
        if keyword in text_lower:
            return True
    return False
```

```
assg_044.py > ...
Editors      #write a python function that classifies a given test as spam or not spam based on
             #the presence of certain keywords.
1  def is_spam(text):
2      spam_keywords = ["win", "free", "prize", "click", "buy now"]
3      text_lower = text.lower()
4      for keyword in spam_keywords:
5          if keyword in text_lower:
6              return True
7      return False
8
9 user_input = input("Enter the text to classify: ")
10 if is_spam(user_input):
11     print("The text is classified as spam.")
12 else:
13     print("The text is not spam.")
```

Code:

```
def is_spam(text):
    spam_keywords = ["win", "free", "prize", "click", "buy now"]
    text_lower = text.lower()
    for keyword in spam_keywords:
        if keyword in text_lower:
            return True
    return False

user_input = input("Enter the text to classify: ")

if is_spam(user_input):
    print("The text is classified as spam.")

else:
    print("The text is not spam.")
```

input/output:

```
PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/hasin/Untitled-2.py
Enter the text to classify: hello,click the link below to claim the prizes
The text is classified as spam.
PS C:\Users\hasin>
```

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 sentance from the user if any word from the emotions list is present in the sentance then display that emotion otherwise display no emotion detected.

```
15 """
16 emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
17 write a python program that detects the emotion of sentance using the above list of emotions
18 """
19 emotions = ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
20 user_emotion = input("Enter an emotion to check: ").strip().lower()
21 if user_emotion in emotions:
22     print(f"The emotion '{user_emotion}' is present in the list.")
23 else:
24     print(f"The emotion '{user_emotion}' is not present in the list.")"""
```

```
15 """
16 emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
17 write a python program that detect the mood of the person and take the sentance from the user
18 """
19 emotions = ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']
20 user_input = input("Enter a sentence to detect emotion: ").lower()
21 detected_emotions = [emotion for emotion in emotions if emotion in user_input]
22 if detected_emotions:
23     print(f"Detected emotions: {', '.join(detected_emotions)}")
24 else:
25     print("No emotion detected.")
```

Code:

```
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:

```
| PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users  
/hasin/Untitled-2.py  
Enter a sentence to detect emotion: I am happy yesterday.  
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

write a python program that predicts a student's grade based on marks obtained and only accept positive integer values from the user otherwise display invalid input.

```

32 below 60=F
33 write a " grade based on marks obtained and o
34 """ < > Accept Tab Accept Word Ctrl + RightArrow ...
35 """
36 """
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53

```

```

27 """
28 90-100=A
29 80-89=B
30 70-79=C
31 60-69=D
32 below 60=F
33 write a python program that predicts a student's grade based on marks obtained and o
34 """
35 """
36 """
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53

```

Code:

try:

```
marks = int(input("Enter the marks (0-100): "))
```

```
if marks < 0 or marks > 100:
```

```

print("Invalid input. Please enter a positive integer between 0 and 100.")

else:
    if 90 <= marks <= 100:
        grade = 'A'
    elif 80 <= marks <= 89:
        grade = 'B'
    elif 70 <= marks <= 69:
        grade = 'C'
    elif 60 <= marks <= 59:
        grade = 'D'
    else:
        grade = 'F'

    print(f"The grade for {marks} is: {grade}")

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

```

Inputs/outputs:

```

PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users
/hasin/Untitled-2.py
Enter the marks (0-100): 90
The grade for 90 is: A
PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users
/hasin/Untitled-2.py
Enter the marks (0-100): 70
The grade for 70 is: C
PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users
/hasin/Untitled-2.py
Enter the marks (0-100): 40
The grade for 40 is: F
PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users
/hasin/Untitled-2.py
Enter the marks (0-100): 60
The grade for 60 is: D
PS C:\Users\hasin>

```

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 code to accept month from the user and display the corresponding zodiac sign and only accept valid month names otherwise display invalid input

```

54
55     """
56
57     march=mesha
58     april=vrishabha
59     may=mithuna
60     june=karka
61     july=simha
62     august=kanya
63     september=tula
64     october=vrischika
65     november=dhanus
66     december=makara
67     january=kumbha
68     february=meena
69     write a python code to accept month from the user and display the corresponding zod
70
71     """
72     month_to_zodiac = {
73         "march": "mesha",
74         "april": "vrishabha",
75         "may": "mithuna",
76         "june": "karka",
77         "july": "simha",
78         "august": "kanya",
79         "september": "tula",
80         "october": "vrischika",
81         "november": "dhanus",
82         "december": "makara",
83         "january": "kumbha",
84         "february": "meena"
85     }
86     user_month = input("Enter a month: ").strip().lower()
87     zodiac_sign = month_to_zodiac.get(user_month)
88     if zodiac_sign:
89         print(f"The zodiac sign for {user_month.capitalize()} is {zodiac_sign}.")
90     else:
91         print("Invalid input. Please enter a valid month name.")
92

```

Code:

```

month_to_zodiac = {

    "march": "mesha",
    "april": "vrishabha",
    "may": "mithuna",
    "june": "karka",
    "july": "simha",
    "august": "kanya",

```

```

    "september": "tula",
    "october": "vrischika",
    "november": "dhanus",
    "december": "makara",
    "january": "kumbha",
    "february": "meena"
}

user_month = input("Enter a month: ").strip().lower()
zodiac_sign = month_to_zodiac.get(user_month)

if zodiac_sign:
    print(f"The zodiac sign for {user_month.capitalize()} is {zodiac_sign}.")
else:
    print("Invalid input. Please enter a valid month name.")

```

Inputs/outputs:

- PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/hasin/Untitled-2.py
Enter a month: january
The zodiac sign for January is kumbha.
- PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/hasin/Untitled-2.py
Enter a month: february
The zodiac sign for February is meena.
- PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/hasin/Untitled-2.py
Enter a month: april
The zodiac sign for April is vrishabha.
- PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/hasin/Untitled-2.py
Enter a month: december
The zodiac sign for December is makara.
- PS C:\Users\hasin>

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:
100     marks = int(input("Enter the marks: "))
101     if marks < 0:
102         print("Invalid input. Please enter a positive integer for marks.")
103     else:
104         if marks >= 40:
105             print("Pass")
106         else:
107             print("Fail")
108 except ValueError:
109     print("Invalid input. Please enter a valid integer for marks.")"""
110 ♦
```

```
"""
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.")

```

Inputs/outputs:

```

PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/users
/hasin/Untitled-2.py
Enter the marks: 88
Pass
● PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/users
/hasin/Untitled-2.py
Enter the marks: 93
Pass
● PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/users
/hasin/Untitled-2.py
Enter the marks: 26
Fail

```

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:

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

```
111  """
112  take age from the user
113  if age is equal or greater than 18
114  then print eligible to vote
115  if age is less than 18
116  print not eligible to vote
117  """
118  """try:
119      age = int(input("Enter your age: "))
120      if age < 0:
121          print("Invalid input. Please enter a positive integer.")
122      else:
123          if age >= 18:
124              print("Eligible to vote")
125          else:
126              print("Not eligible to vote")
127  except ValueError:
128      print("Invalid input. Please enter a valid integer.")"""
129
130
```

```
111  """
112  take age from the user
113  if age is equal or greater than 18
114  then print eligible to vote
115  if age is less than 18
116  print not eligible to vote
117  """
118  try:
119      age = int(input("Enter your age: "))
120      if age < 0:
121          print("Invalid input. Please enter a positive integer.")
122      else:
123          if age >= 18:
124              print("Eligible to vote")
125          else:
126              print("Not eligible to vote")
127  except ValueError:
128      print("Invalid input. Please enter a valid integer.")
129
130
```

Code:

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.")
```

Inputs/outputs:

```
PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/users/hasin/Untitled-2.py
Enter your age: 20
Eligible to vote
PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/users/hasin/Untitled-2.py
Enter your age: 14
Not 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.

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)

```

Inputs/outputs:

```

● PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/Python312/python.exe c:/Users
/hasin/Untitled-2.py
Enter student names separated by commas: hasini,bob,anvitha,anjali& C:/Users/hasin/AppData/Loca
l/Programs/Python/Python312/python.exe c:/Users/hasin/Untitled-2.py
Palindrome names: ['bob']

```

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 """
149 words = input("Enter words separated by commas: ").split(",")
150 long_words = [word.strip() for word in words if len(word.strip()) > 5]
151 """
152
```

```
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
```

Inputs/outputs:

```
PS C:\Users\hasin> & C:/Users/hasin/AppData/Local/Programs/Python/3.8.5/python hasin/Untitled-2.py
Enter words separated by commas: eagle,jackle,lion
Long words: ['jackle']
Short words: ['eagle', 'lion']
```