AI ASSISTED CODING

ASSIGNMENT-4.1

Name: Harshitha Guda

H.T.No:2303A51102

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

```python
#Write a python function which reads a text and classifies the text as spam or not spam based on the presence of certain keywords like 'win',
def is_spam(text):
    spam_keywords = ['win', 'prize', 'free', 'money','click', 'buy now', 'subscribe', 'limited time offer']
    text_lower = text.lower()
    for keyword in spam_keywords:
        if keyword in text_lower:
            return True
    return False
try:
    user_input = input("Enter the text message: ")
    if is_spam(user_input):
        print("The message is classified as spam.")
    else:
        print("The message is not spam.")
except Exception as e:
    print("An error occurred:", e)

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Python

PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the text message: free lottery
The message is classified as spam.
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the text message: buy now offers are limited
The message is classified as spam.
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the text message: Expirey date for your subscription
The message is not spam.
PS C:\Users\gudah>
```

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

```
'''
s=Ohh! what a surprise
display excitement
'''
def classify_emotion(text):
    excitement_keywords = ['wow', 'amazing', 'surprise', 'exciting', 'fantastic', 'incredible']
    happiness_keywords = ['happy', 'joy', 'pleased', 'delighted', 'content', 'cheerful']
    sad_keywords = ['sad', 'unhappy', 'sorrow', 'depressed', 'downcast', 'gloomy']
    anger_keywords = ['angry', 'mad', 'furious', 'irate', 'outraged', 'annoyed']
    text_lower = text.lower()
    if any(keyword in text_lower for keyword in excitement_keywords):
        return "excitement"
    elif any(keyword in text_lower for keyword in happiness_keywords):
        return "happiness"
    elif any(keyword in text_lower for keyword in sad_keywords):
        return "sadness"
    elif any(keyword in text_lower for keyword in anger_keywords):
        return "anger"
    else:
        return "neutral"
try:
    user_input = input("Enter the text message: ")
    emotion = classify_emotion(user_input)
    print(emotion)
except Exception as e:
    print("An error occurred:", e)
```

```
def classify_emotion(text):
    excitement_keywords = ['wow', 'amazing', 'surprise', 'exciting', 'fantastic', 'incredible']
    happiness_keywords = ['happy', 'joy', 'pleased', 'delighted', 'content', 'cheerful']
    sad_keywords = ['sad', 'unhappy', 'sorrow', 'depressed', 'downcast', 'gloomy']
    anger_keywords = ['angry', 'mad', 'furious', 'irate', 'outraged', 'annoyed']
    text_lower = text.lower()
    if any(keyword in text_lower for keyword in excitement_keywords):
        return "excitement"
    elif any(keyword in text_lower for keyword in happiness_keywords):
        return "happiness"
    elif any(keyword in text_lower for keyword in sad_keywords):
        return "sadness"
    elif any(keyword in text_lower for keyword in anger_keywords):
        return "anger"
    else:
        return "neutral"
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the text message: She is good
neutral
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the text message: Wow!
excitement
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the text message: I am so happy
happiness
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the text message: so annoyed
anger
```

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

```
'''
m=85
display B
m=55
display F
m=78
display C
m=61
display D'''
def classify_grade(marks):
    if marks < 0 or marks > 100:
        return "Invalid marks"
    elif 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 student marks: "))
    grade = classify_grade(user_input)
    print(grade)
except ValueError:
    print("Invalid marks")
```

```
'''
m=85
display B
m=55
display F
m=78
display C
m=61
display D'''
def classify_grade(marks):
    if marks < 0 or marks > 100:
        return "Invalid marks"
    elif marks >= 90:
        return "A"
    elif marks >= 80:
        return "B"
    elif marks >= 70:
        return "C"
    elif marks >= 60:
        return "D"
    else:
        return "F"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: 91
A
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: 55
F
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: 89
B
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: -1
Invalid marks
```

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

```
'''March
display Mesha
April
display Vrishabha
May
display Mithuna
June
display Karka
July
display Simha
August
display Kanya
September
display Tula
October
display Vrischika
November
display Dhanu
December
display Makara
January
display Kumbha
February
display Meena
'''
def get_zodiac_sign(month):
    month = month.lower()
    zodiac_signs = {
        'march': 'Mesha',
        'april': 'Vrishabha',
        'may': 'Mithuna',
        'june': 'Karka',
        'july': 'Simha',
        'august': 'Kanya',
        'september': 'Tula',
        'october': 'Vrischika',
```

```python
def get_zodiac_sign(month):
    month = month.lower()
    zodiac_signs = {
        'march': 'Mesha',
        'april': 'Vrishabha',
        'may': 'Mithuna',
        'june': 'Karka',
        'july': 'Simha',
        'august': 'Kanya',
        'september': 'Tula',
        'october': 'Vrischika',
        'november': 'Dhanu',
        'december': 'Makara',
        'january': 'Kumbha',
        'february': 'Meena'
    }
    return zodiac_signs.get(month, "Invalid month")
try:
    user_input = input("Enter the month: ")
    zodiac_sign = get_zodiac_sign(user_input)
    print(zodiac_sign)
except Exception as e:
    print("An error occurred:", e)
```

```python
    return zodiac_signs.get(month, "Invalid month")
try:
    user_input = input("Enter the month: ")
    zodiac_sign = get_zodiac_sign(user_input)
    print(zodiac_sign)
except Exception as e:
    print("An error occurred:", e)

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the month: february
Meena
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the month: october
Vrischika
```

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']

```python
'''
Read student marks
validate that the marks are between 0 and 100
display appropriate message if invalid
if marks are greater than 40 display pass else display fail
'''
def classify_marks(marks):
    if marks < 0 or marks > 100:
        return "Invalid marks"
    elif marks > 40:
        return "pass"
    else:
        return "fail"
try:
    user_input = int(input("Enter the student marks: "))
    result = classify_marks(user_input)
    print(result)
except ValueError:
    print("Invalid marks")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: 35
fail
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: 50
pass
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: -1
Invalid marks
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the student marks: 111
Invalid marks
```

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.

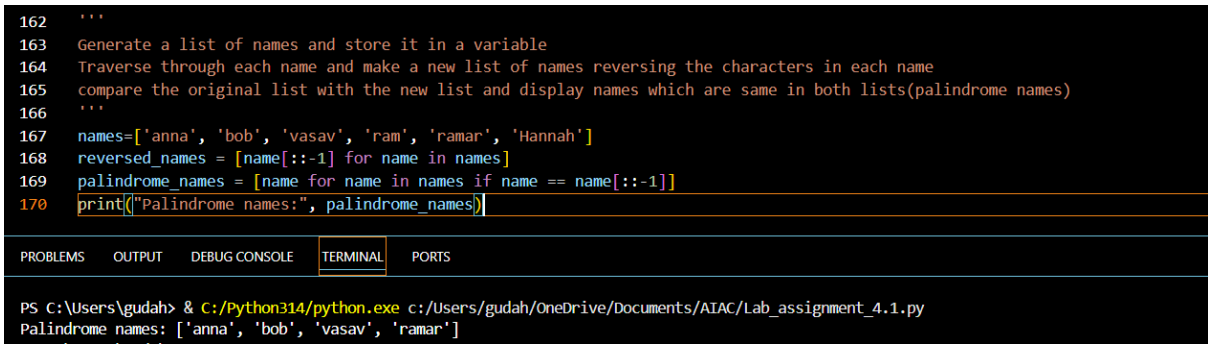
```python
'''
read the age of a person
display eligible to vote if age is greater than or equal to 18
'''
def check_voting_eligibility(age):
    if age < 0:
        return "Invalid age"
    elif age >= 18:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"
try:
    user_input = int(input("Enter the age of the person: "))
    eligibility = check_voting_eligibility(user_input)
    print(eligibility)
except ValueError:
    print("Invalid age")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the age of the person: 18
Eligible to vote
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the age of the person: 16
Not eligible to vote
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Enter the age of the person: -8
Invalid age
```

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.



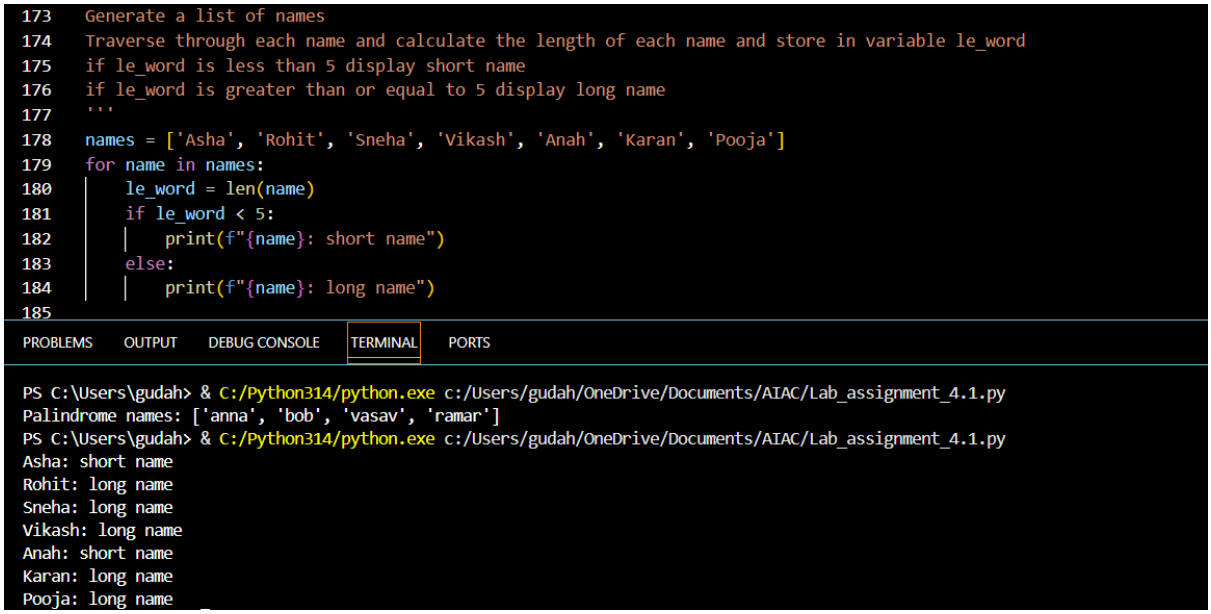
```
'''
Generate a list of names and store it in a variable
Traverse through each name and make a new list of names reversing the characters in each name
compare the original list with the new list and display names which are same in both lists(palindrome names)
'''
names=['anna', 'bob', 'vasav', 'ram', 'ramar', 'Hannah']
reversed_names = [name[::-1] for name in names]
palindrome_names = [name for name in names if name == name[::-1]]
print("Palindrome names:", palindrome_names)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Palindrome names: ['anna', 'bob', 'vasav', 'ramar']
```

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



```
Generate a list of names
Traverse through each name and calculate the length of each name and store in variable le_word
if le_word is less than 5 display short name
if le_word is greater than or equal to 5 display long name
'''
names = ['Asha', 'Rohit', 'Sneha', 'Vikash', 'Anah', 'Karan', 'Pooja']
for name in names:
    le_word = len(name)
    if le_word < 5:
        print(f"{name}: short name")
    else:
        print(f"{name}: long name")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Palindrome names: ['anna', 'bob', 'vasav', 'ramar']
PS C:\Users\gudah> & C:/Python314/python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_4.1.py
Asha: short name
Rohit: long name
Sneha: long name
Vikash: long name
Anah: short name
Karan: long name
Pooja: long name
```