AI Assisted Coding

ASSIGNMENT 4

Name: M.Goutham.

HT No: 2303A52010

Batch: 31

Task 1:

Write a Python function that classifies a given text as Spam or Not Spam using zero-shot prompting.

Steps:

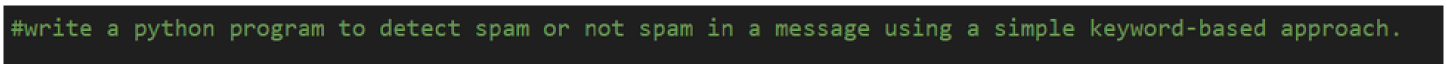
1. Construct a prompt without any examples.

2. Clearly specify the output labels.

3. Display only the predicted label.

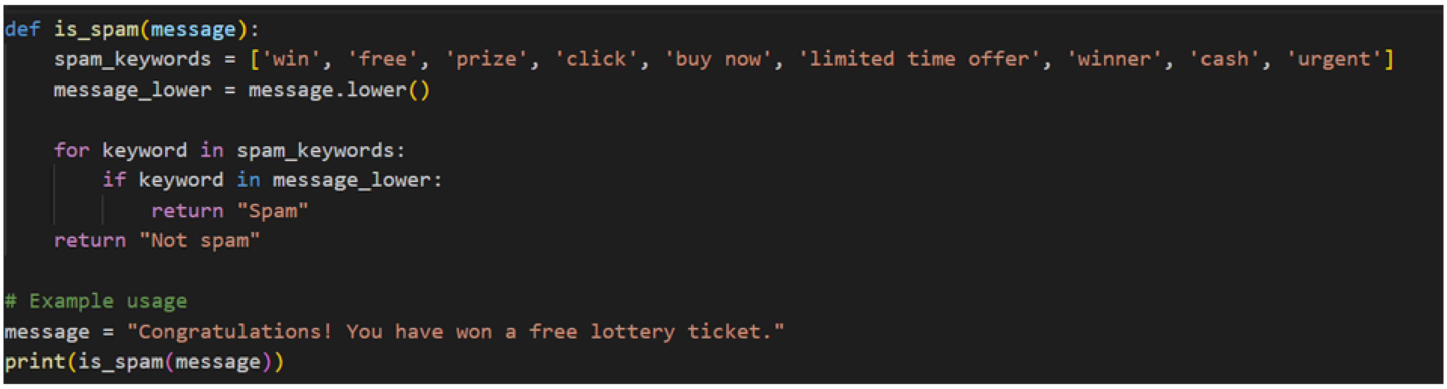
Input: "Congratulations! You have won a free lottery ticket." Expected Output: Spam

prompt:



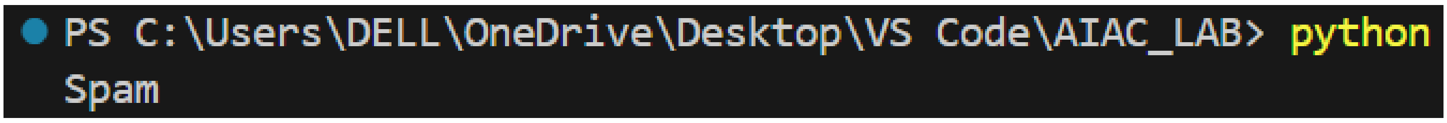
```
#write a python program to detect spam or not spam in a message using a simple keyword-based approach.
```

code:



```python
def is_spam(message):
    spam_keywords = ['win', 'free', 'prize', 'click', 'buy now', 'limited time offer', 'winner', 'cash', 'urgent']
    message_lower = message.lower()

    for keyword in spam_keywords:
        if keyword in message_lower:
            return "Spam"
    return "Not spam"

# Example usage
message = "Congratulations! You have won a free lottery ticket."
print(is_spam(message))
```

output:



```
● PS C:\Users\DELL\OneDrive\Desktop\VS Code\AIAC_LAB> python
  Spam
```

Explanation:

This function checks if any of the predefined spam keywords are present in the input text.

If any keyword is found, it classifies the text as "Spam"; otherwise, it classifies it as "Not Spam".

Task 2:

Write a Python program that detects the emotion of a sentence using one-shot prompting.

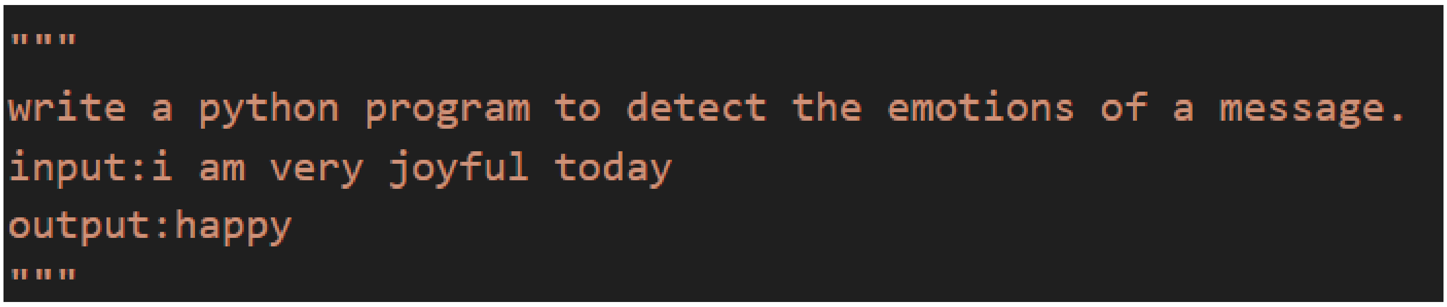
Emotions: [‘happy', 'sad', 'angry', 'excited', 'nervous', ’neutral’] Steps:

1. Provide one labeled example inside the prompt.

2. Take a sentence as input.

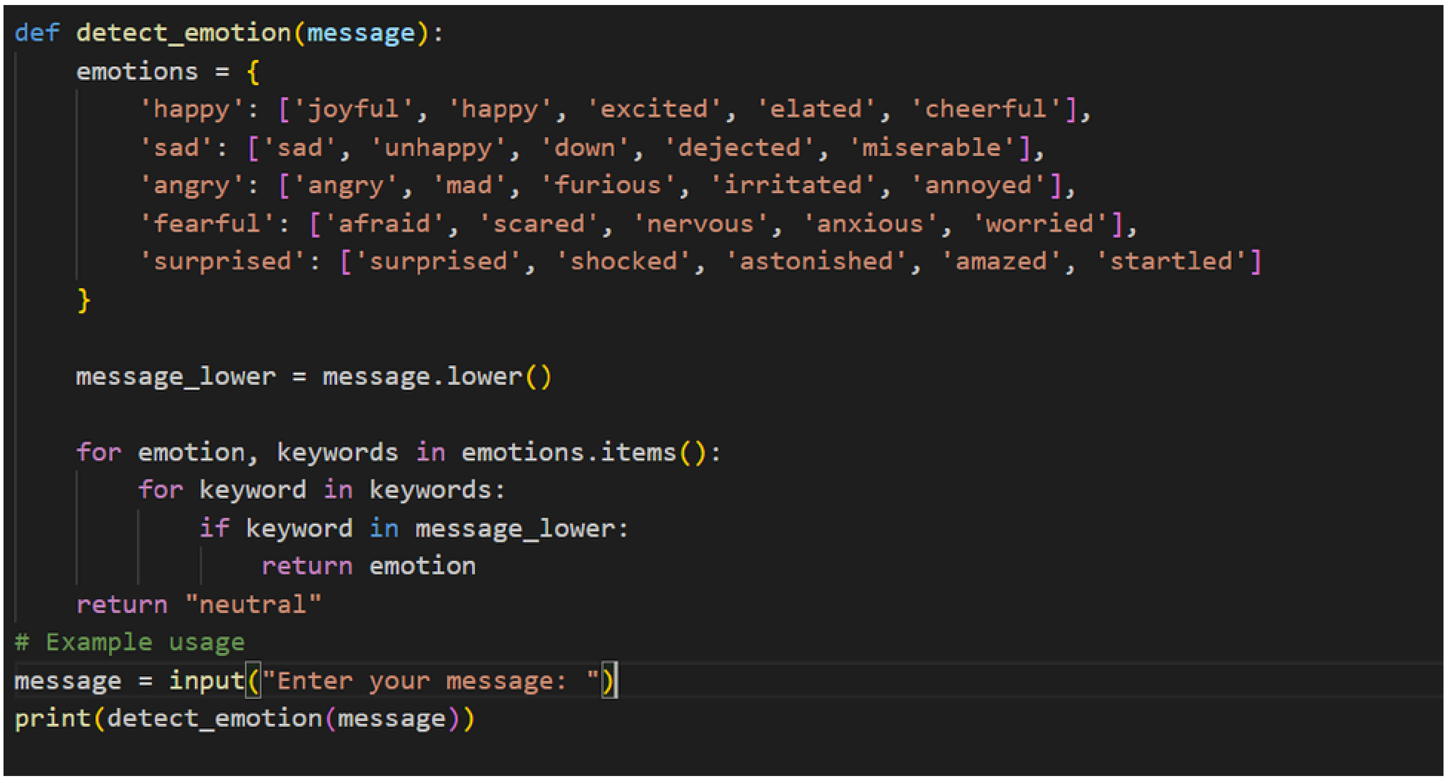
3. Print the predicted emotion

prompt:



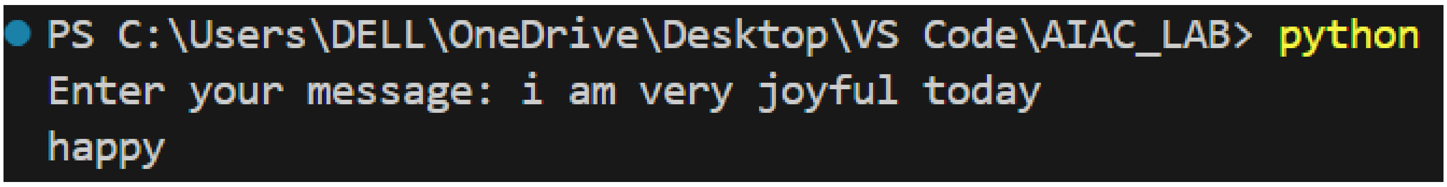
```
"""
write a python program to detect the emotions of a message.
input:i am very joyful today
output:happy
"""
```

code:



```python
def detect_emotion(message):
    emotions = {
        'happy': ['joyful', 'happy', 'excited', 'elated', 'cheerful'],
        'sad': ['sad', 'unhappy', 'down', 'dejected', 'miserable'],
        'angry': ['angry', 'mad', 'furious', 'irritated', 'annoyed'],
        'fearful': ['afraid', 'scared', 'nervous', 'anxious', 'worried'],
        'surprised': ['surprised', 'shocked', 'astonished', 'amazed', 'startled']
    }

    message_lower = message.lower()

    for emotion, keywords in emotions.items():
        for keyword in keywords:
            if keyword in message_lower:
                return emotion
    return "neutral"
# Example usage
message = input("Enter your message: ")
print(detect_emotion(message))
```

output:



```
PS C:\Users\DELL\OneDrive\Desktop\VS Code\AIAC_LAB> python
Enter your message: i am very joyful today
happy
```

Explanation:

The code defines a function detect_emotion that takes a sentence as input and returns the detected emotion based on specific keywords. It converts the sentence to lowercase and checks for the presence of keywords associated with different emotions such as happy, sad, angry, excited, and nervous. If none of these keywords are found, it returns "neutral". The user is prompted to enter a sentence, and the detected emotion is printed.

Question 3: Few-Shot Prompting (Student Grading Based on Marks)

Task 3:

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

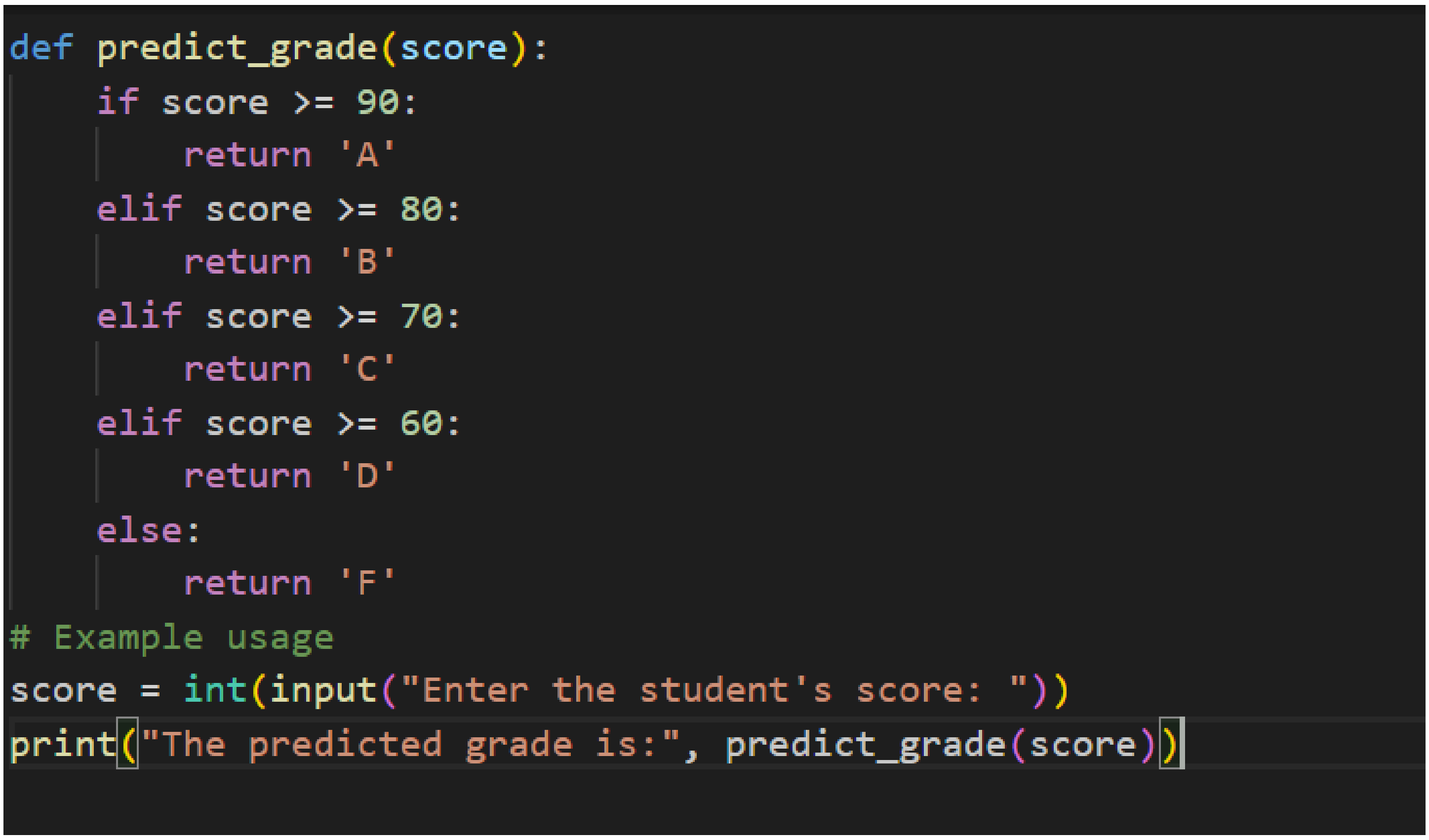
• 60–69 → D

• Below 60 → F

prompt:

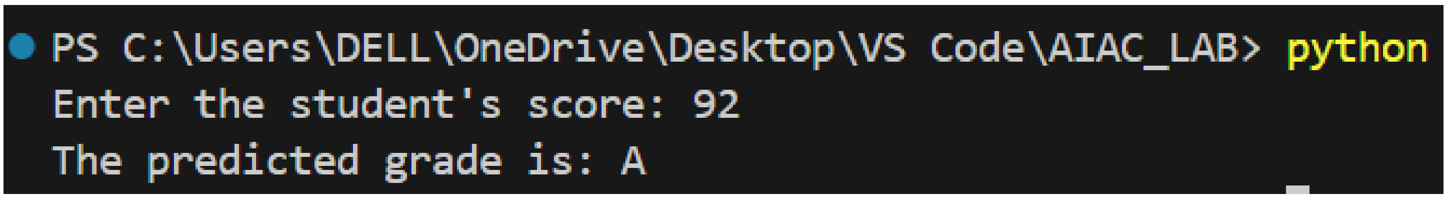
```
"""
write a python program to predict the student grades based on their scores
input:95
output:A
input:85
output:B
input:75
output:C
input:65
output:D
"""
```

code:



```python
def predict_grade(score):
    if score >= 90:
        return 'A'
    elif score >= 80:
        return 'B'
    elif score >= 70:
        return 'C'
    elif score >= 60:
        return 'D'
    else:
        return 'F'
# Example usage
score = int(input("Enter the student's score: "))
print("The predicted grade is:", predict_grade(score))
```

output:



```
PS C:\Users\DELL\OneDrive\Desktop\VS Code\AIAC_LAB> python
Enter the student's score: 92
The predicted grade is: A
```

Explanation:

This code defines a function get_grade that takes an integer input marks and returns a corresponding grade based on predefined thresholds.

Question 4:Multi-Shot Prompting (Indian Zodiac Sign Prediction using Month Name)

Task 4:

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

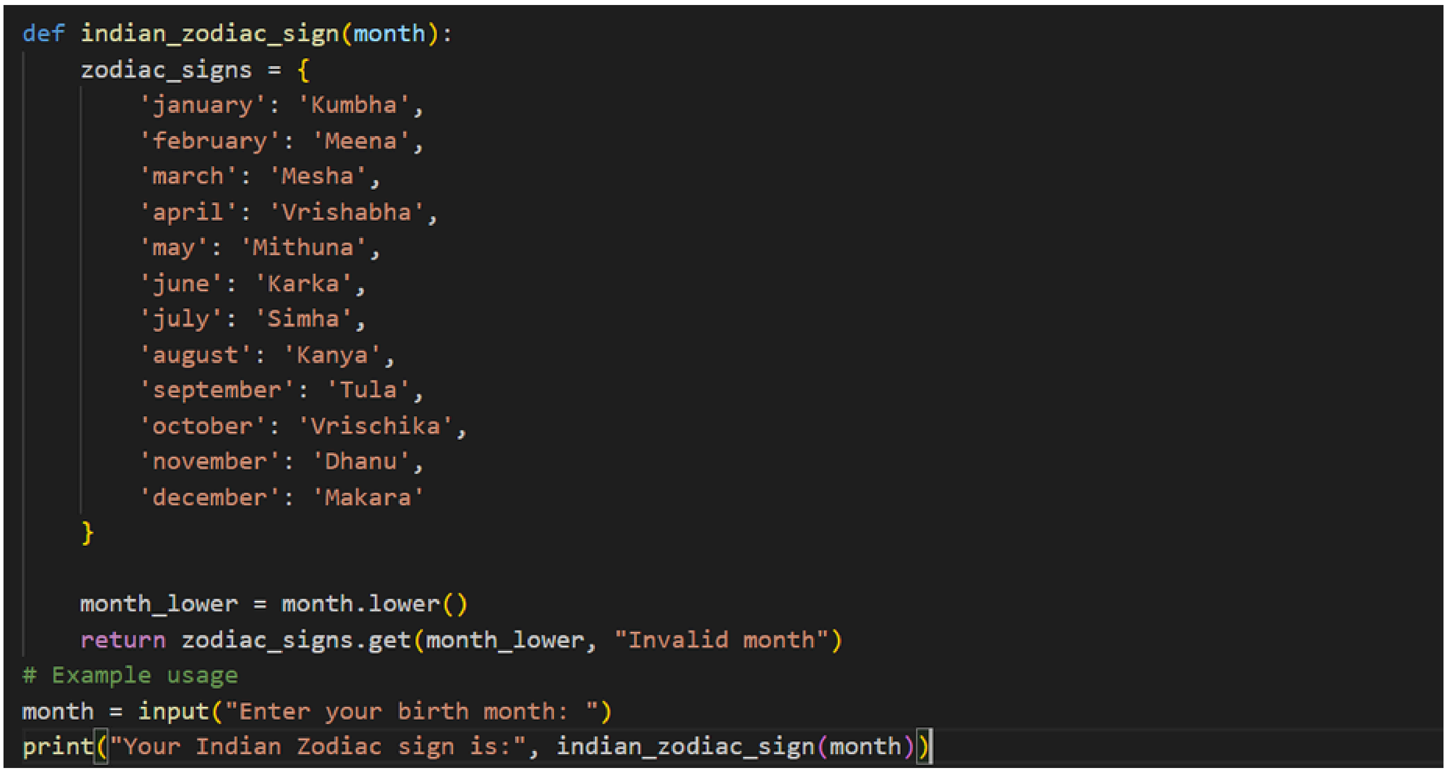
January → Kumbha

February → Meena

prompt:

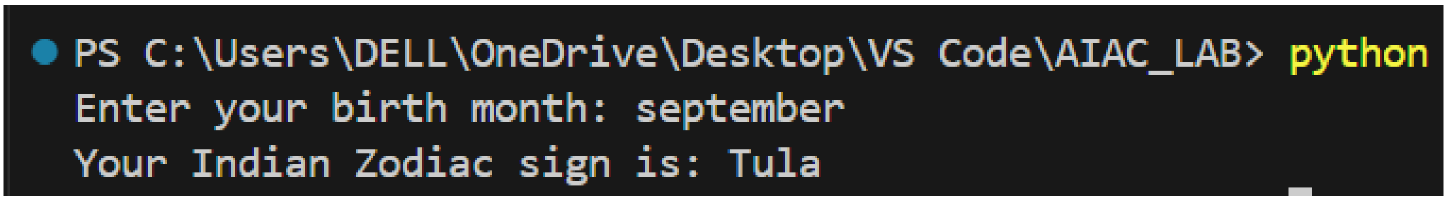
```
"""
write a python program that predicts a person's indian Zodiac sign based on their birth month.
March → Mesha
April → Vrishabha
May → Mithuna
June → Karka
July → Simha
August → Kanya
September → Tula
October → Vrischika
November → Dhanu
December → Makara
January → Kumbha
February → Meena
"""
```

code:



```python
def indian_zodiac_sign(month):
    zodiac_signs = {
        'january': 'Kumbha',
        'february': 'Meena',
        '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")
# Example usage
month = input("Enter your birth month: ")
print("Your Indian Zodiac sign is:", indian_zodiac_sign(month))
```

output:



```
● PS C:\Users\DELL\OneDrive\Desktop\VS Code\AIAC_LAB> python
  Enter your birth month: september
  Your Indian Zodiac sign is: Tula
```

Explanation:

This code defines a dictionary that maps each month of the year to its corresponding Indian Zodiac sign . It then prompts the user to input their birth month, retrieves the associated from the dictionary, and prints it. If the user enters an invalid month, it returns "Invalid month".

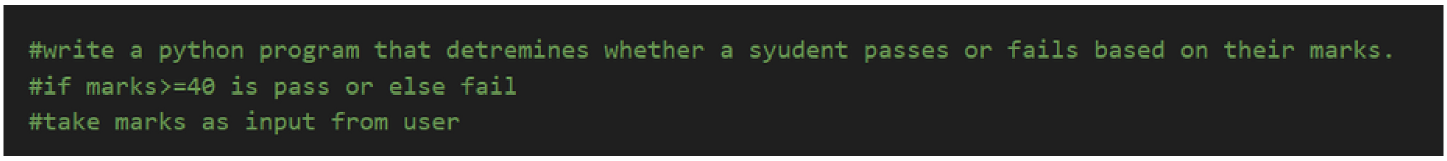
Question 5: Result Analysis Based on Marks

Task 5:

Write a Python program that determines whether a student Passes or Fails based on marks using Chain-of-Thought (CoT) prompting.

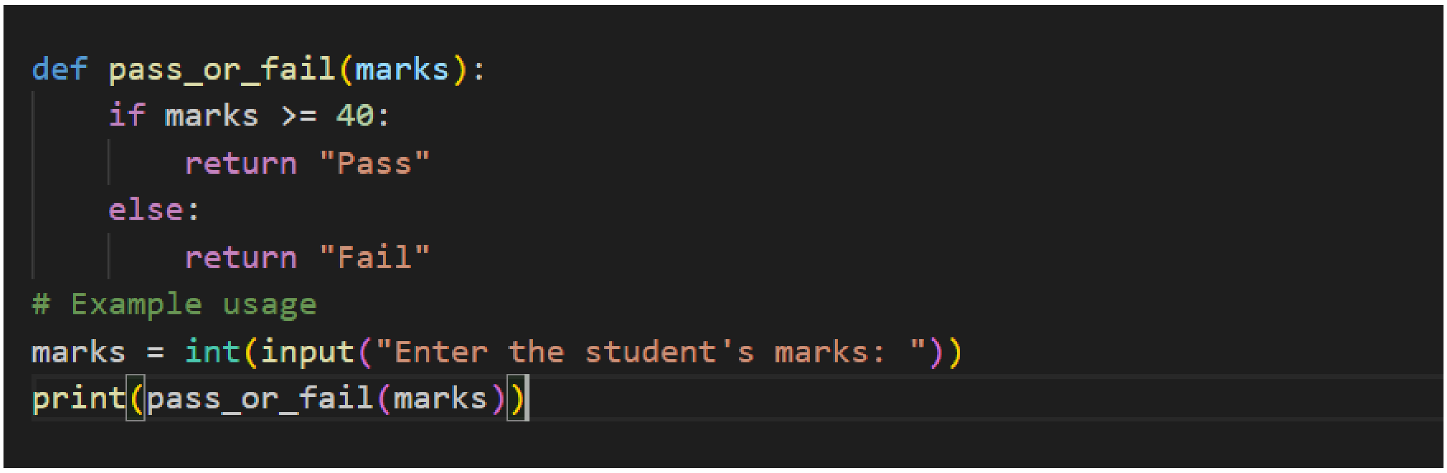
Result Categories: ['Pass', 'Fail']

prompt:



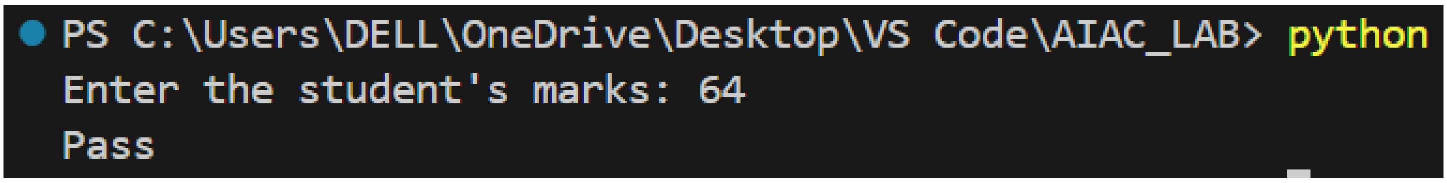
```
#write a python program that detremines whether a syudent passes or fails based on their marks.
#if marks>=40 is pass or else fail
#take marks as input from user
```

code:



```python
def pass_or_fail(marks):
    if marks >= 40:
        return "Pass"
    else:
        return "Fail"
# Example usage
marks = int(input("Enter the student's marks: "))
print(pass_or_fail(marks))
```

output:



```
PS C:\Users\DELL\OneDrive\Desktop\VS Code\AIAC_LAB> python
Enter the student's marks: 64
Pass
```

Explanation:

This code takes the marks obtained by a student as input and checks if the student has passed or failed based on a predefined passing criteria (40 marks). If the marks are greater than or equal to 40, it prints "Pass"; otherwise, it prints "Fail".

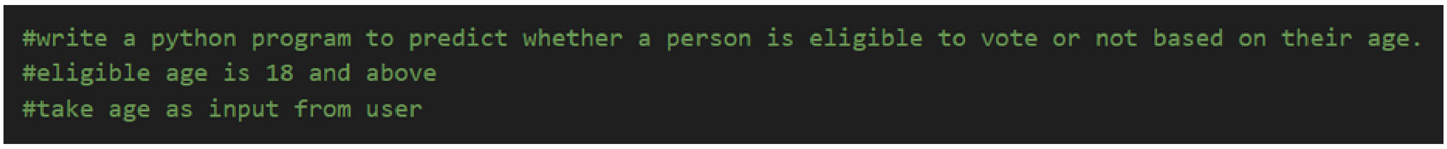
Question 6: Voting Eligibility Check (Chain-of-Thought Prompting)

Task 6:

Write a Python program that determines whether a person is

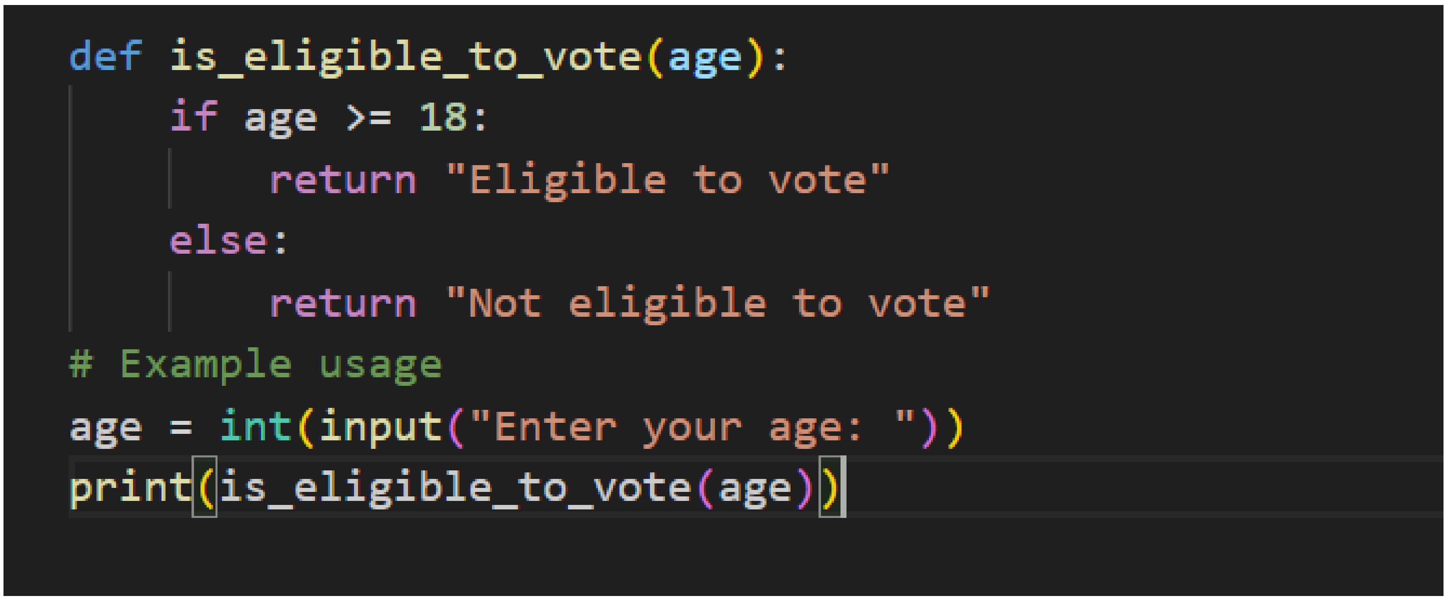
eligible to vote using Chain-of-Thought (CoT) prompting.

prompt:



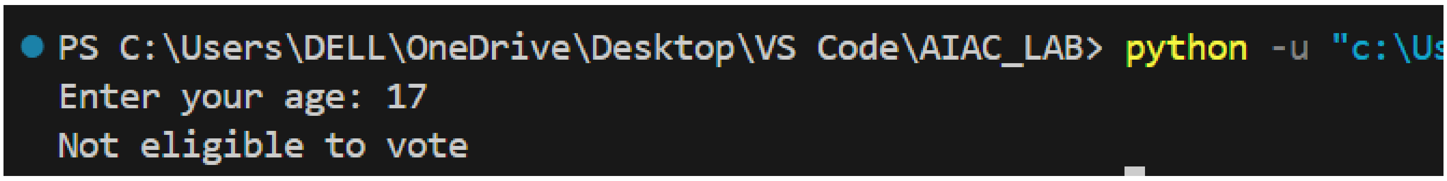
```
#write a python program to predict whether a person is eligible to vote or not based on their age.
#eligible age is 18 and above
#take age as input from user
```

code:



```python
def is_eligible_to_vote(age):
    if age >= 18:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"
# Example usage
age = int(input("Enter your age: "))
print(is_eligible_to_vote(age))
```

output:



```
● PS C:\Users\DELL\OneDrive\Desktop\VS Code\AIAC_LAB> python -u "c:\Us
  Enter your age: 17
  Not eligible to vote
```

Explanation:

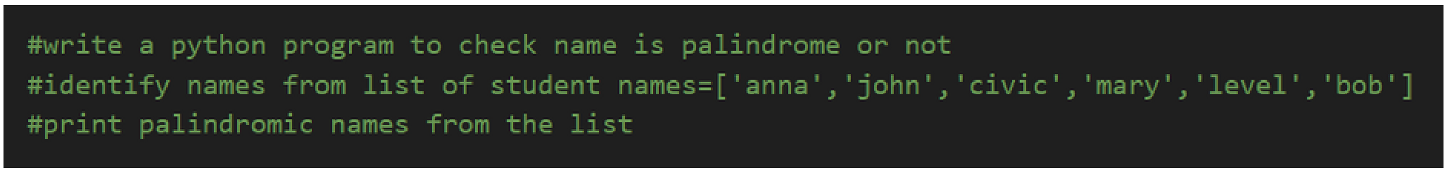
This code snippet checks if a user is eligible to vote based on their age. It prompts the user to enter their age and then evaluates whether the age is 18 or older. If the condition is met, it prints "Eligible"; otherwise, it prints "Not Eligible".

Question 7: Prompt Chaining (String Processing – Palindrome Names)

Task 7:

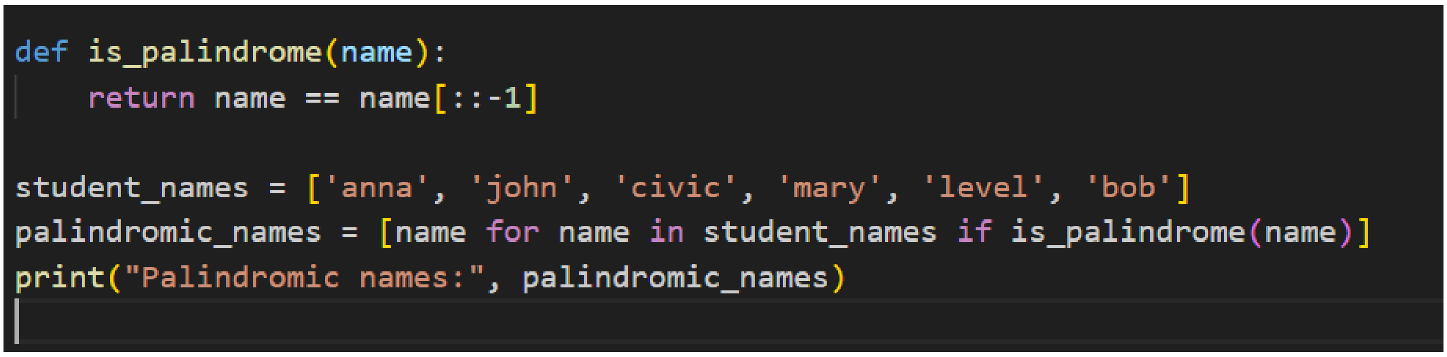
Write a Python program that uses the prompt chaining technique to identify palindrome names from a list of student names.

prompt:



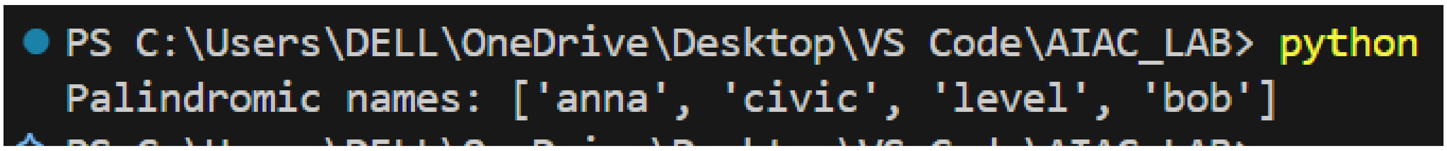
```
#write a python program to check name is palindrome or not
#identify names from list of student names=['anna','john','civic','mary','level','bob']
#print palindromic names from the list
```

code:



```python
def is_palindrome(name):
    return name == name[::-1]

student_names = ['anna', 'john', 'civic', 'mary', 'level', 'bob']
palindromic_names = [name for name in student_names if is_palindrome(name)]
print("Palindromic names:", palindromic_names)
```

output:



```
PS C:\Users\DELL\OneDrive\Desktop\VS Code\AIAC_LAB> python
Palindromic names: ['anna', 'civic', 'level', 'bob']
```

Explanation:

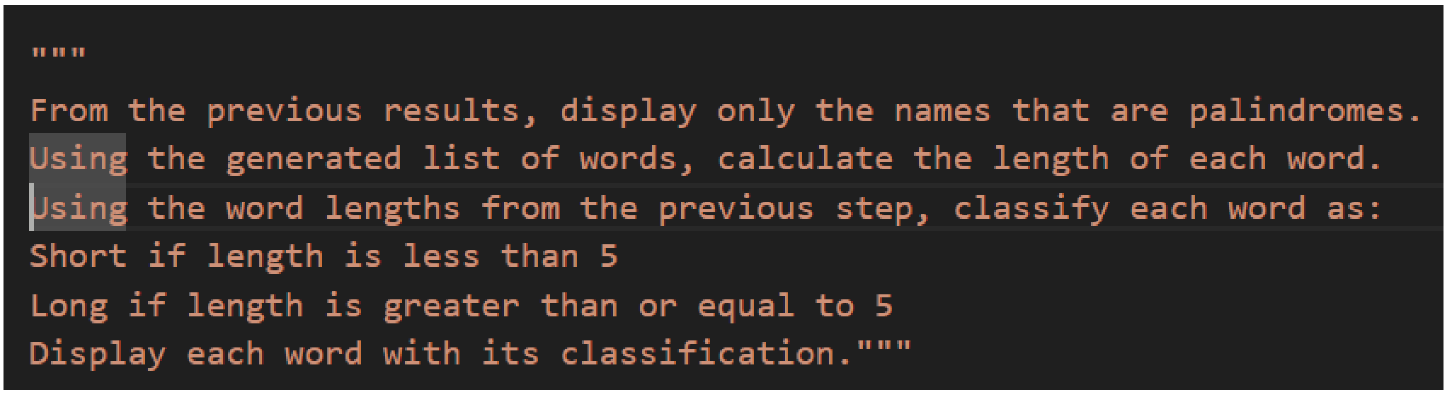
This code defines a list of student names and includes a function to check if a name is a palindrome (a word that reads the same backward as forward). It then filters the list to find all names that are palindromes and prints them. The use of list comprehension makes the filtering process concise and efficient. Used prompt chaining technique.

Question 8: Prompt Chaining (String Processing – Word Length Analysis)

Task 8:

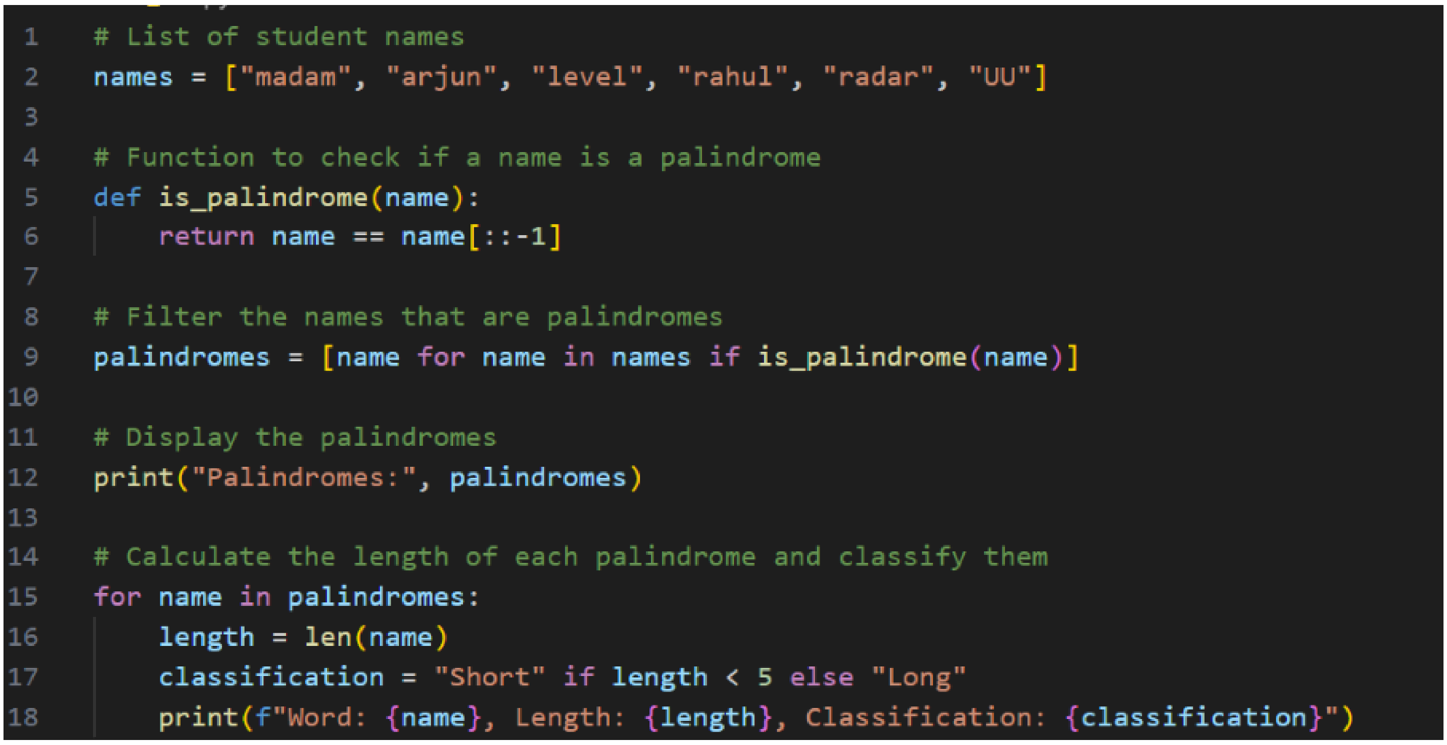
Write a Python program that uses prompt chaining to analyze a list of words. In the first prompt, generate a list of words. In the second prompt, traverse the list and calculate the length of each word. In the third prompt, use the output of the previous step to determine whether each word is Short (length less than 5) or Long (length greater than or equal to 5), and display the result for each word

prompt:



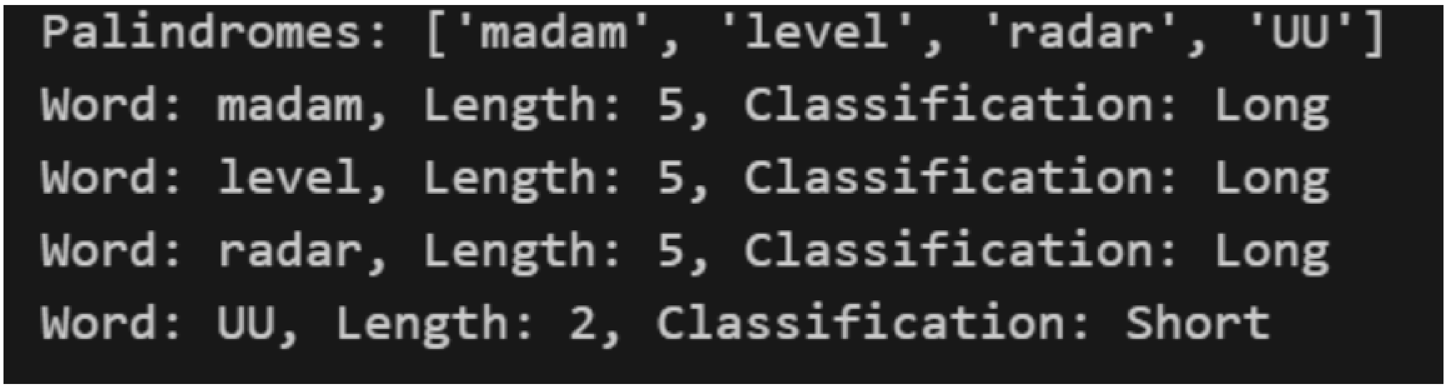
```
"""
From the previous results, display only the names that are palindromes.
Using the generated list of words, calculate the length of each word.
Using the word lengths from the previous step, classify each word as:
Short if length is less than 5
Long if length is greater than or equal to 5
Display each word with its classification."""
```

code:



```python
# List of student names
names = ["madam", "arjun", "level", "rahul", "radar", "UU"]

# Function to check if a name is a palindrome
def is_palindrome(name):
    return name == name[::-1]

# Filter the names that are palindromes
palindromes = [name for name in names if is_palindrome(name)]

# Display the palindromes
print("Palindromes:", palindromes)

# Calculate the length of each palindrome and classify them
for name in palindromes:
    length = len(name)
    classification = "Short" if length < 5 else "Long"
    print(f"Word: {name}, Length: {length}, Classification: {classification}")
```

output:



```
Palindromes: ['madam', 'level', 'radar', 'UU']
Word: madam, Length: 5, Classification: Long
Word: level, Length: 5, Classification: Long
Word: radar, Length: 5, Classification: Long
Word: UU, Length: 2, Classification: Short
```

Explanation:

This code defines a list of student names and checks which of those names are palindromes (words that read the same forwards and backwards). It uses a helper function is_palindrome to perform this check. The palindromic names are then filtered into a new list called palindromes. Used prompt chaining technique.