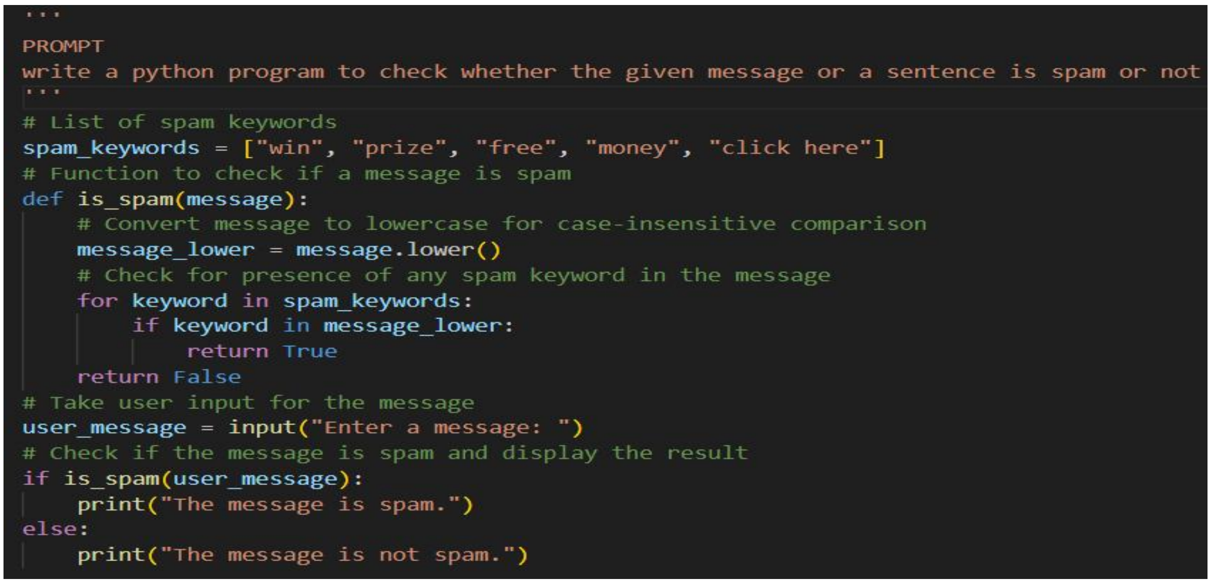
AI ASSISTED CODING
ASSIGNMENT- 4

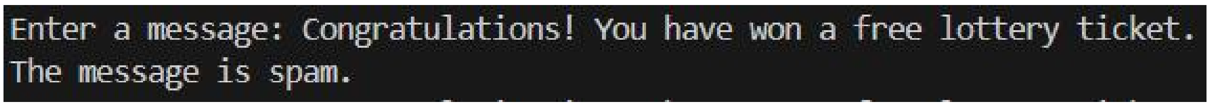
Name: A.Niteesh Kumar
HT No: 2303A52341
Batch: 32
Lab Objectives:
To explore and apply different levels of prompt examples in AI-assisted code generation.
To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality.
To evaluate the impact of context richness and example quantity on AI performance.
Question 1: Zero-Shot Prompting (Basic Lab Task)
Task 1:
write a python program to check whether the given message or a sentence is spam or not
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 & Code:



```python
'''
PROMPT
write a python program to check whether the given message or a sentence is spam or not
'''
# List of spam keywords
spam_keywords = ["win", "prize", "free", "money", "click here"]
# Function to check if a message is spam
def is_spam(message):
    # Convert message to lowercase for case-insensitive comparison
    message_lower = message.lower()
    # Check for presence of any spam keyword in the message
    for keyword in spam_keywords:
        if keyword in message_lower:
            return True
    return False
# Take user input for the message
user_message = input("Enter a message: ")
# Check if the message is spam and display the result
if is_spam(user_message):
    print("The message is spam.")
else:
    print("The message is not spam.")
```

Output(O/P)

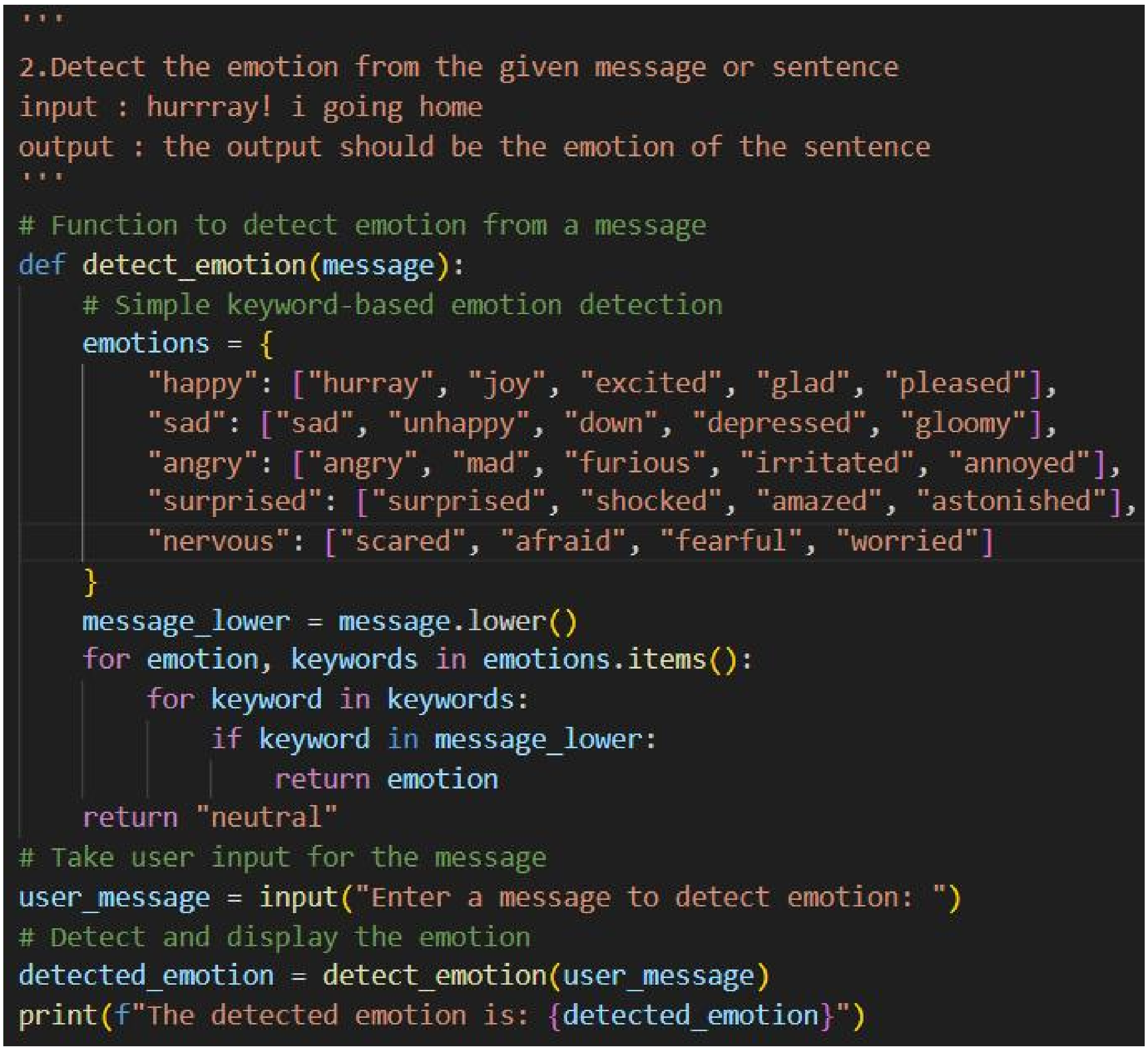


```
Enter a message: Congratulations! You have won a free lottery ticket.
The message is spam.
```

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"

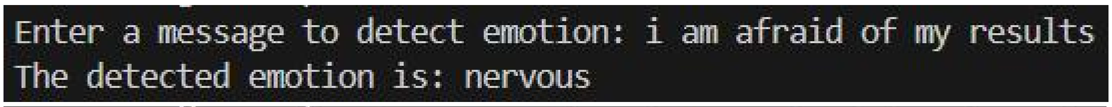
Question 2: One-Shot Prompting (Emotion detection)

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 & Code:

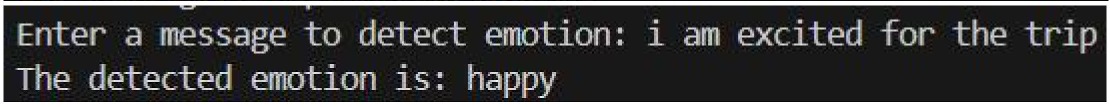


```python
'''
2.Detect the emotion from the given message or sentence
input : hurrray! i going home
output : the output should be the emotion of the sentence
'''
# Function to detect emotion from a message
def detect_emotion(message):
    # Simple keyword-based emotion detection
    emotions = {
        "happy": ["hurray", "joy", "excited", "glad", "pleased"],
        "sad": ["sad", "unhappy", "down", "depressed", "gloomy"],
        "angry": ["angry", "mad", "furious", "irritated", "annoyed"],
        "surprised": ["surprised", "shocked", "amazed", "astonished"],
        "nervous": ["scared", "afraid", "fearful", "worried"]
    }
    message_lower = message.lower()
    for emotion, keywords in emotions.items():
        for keyword in keywords:
            if keyword in message_lower:
                return emotion
    return "neutral"
# Take user input for the message
user_message = input("Enter a message to detect emotion: ")
# Detect and display the emotion
detected_emotion = detect_emotion(user_message)
print(f"The detected emotion is: {detected_emotion}")
```

Output:



```
Enter a message to detect emotion: i am afraid of my results
The detected emotion is: nervous
```



```
Enter a message to detect emotion: i am excited for the trip
The detected emotion is: 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

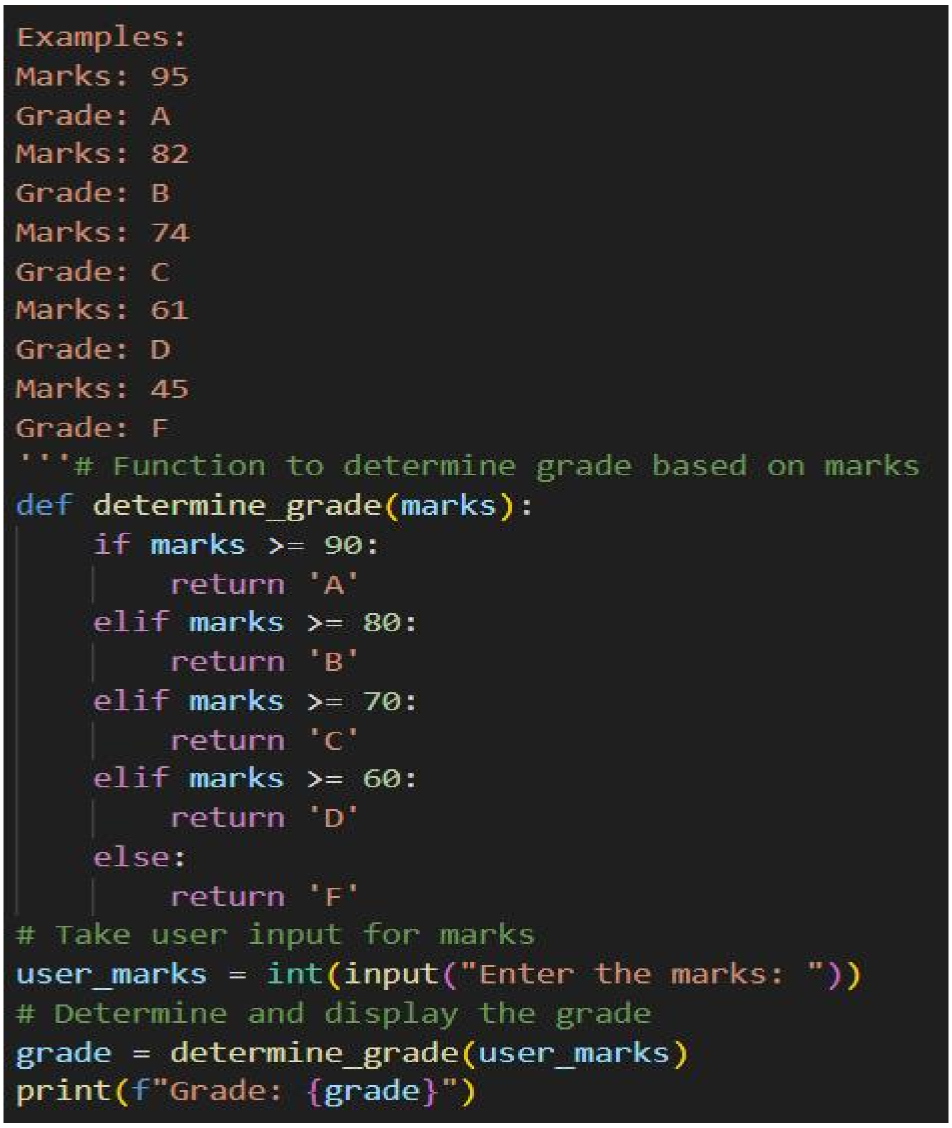
• 80–89 → B

• 70–79 → C

• 60–69 → D

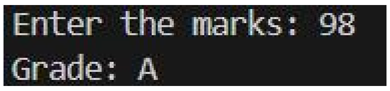
• Below 60 → F

Prompt & Code:

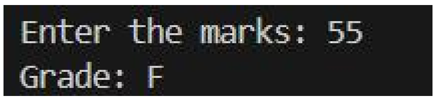


```
Examples:
Marks: 95
Grade: A
Marks: 82
Grade: B
Marks: 74
Grade: C
Marks: 61
Grade: D
Marks: 45
Grade: F
'''# Function to determine grade based on marks
def determine_grade(marks):
    if marks >= 90:
        return 'A'
    elif marks >= 80:
        return 'B'
    elif marks >= 70:
        return 'C'
    elif marks >= 60:
        return 'D'
    else:
        return 'F'
# Take user input for marks
user_marks = int(input("Enter the marks: "))
# Determine and display the grade
grade = determine_grade(user_marks)
print(f"Grade: {grade}")
```

Output:



```
Enter the marks: 98
Grade: A
```



```
Enter the marks: 55
Grade: F
```

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):

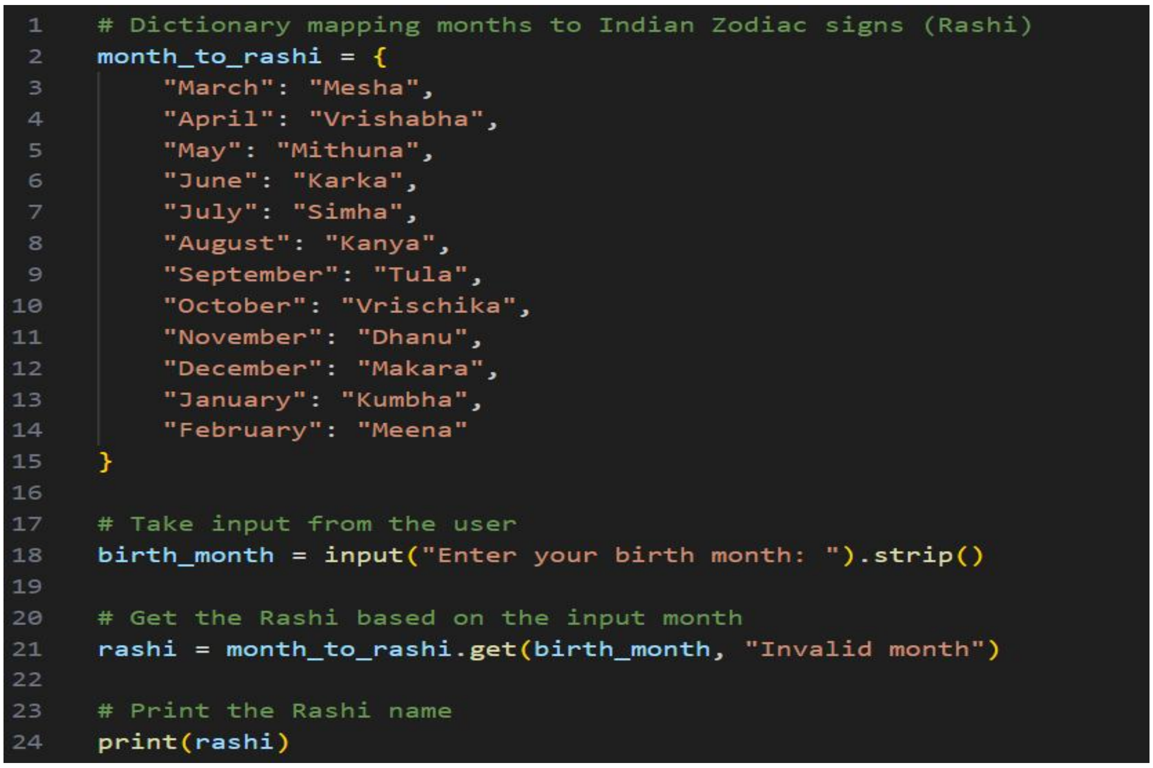
The Indian Zodiac cycle starts in March with Mesha and follows this order:

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 :
Examples:
March → Mesha
April → Vrishabha
May → Mithuna
June → Karka
July → Simha
August → Kanya
September → Tula
October → Vrischika
November → Dhanu
December → Makara
January → Kumbha
February → Meena
Print only the Rashi name.
Take Input From User.

Code:



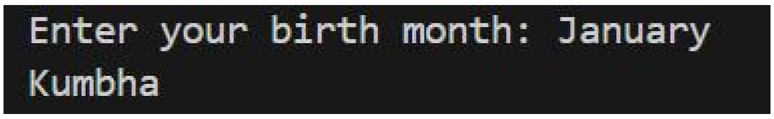
```python
# Dictionary mapping months to Indian Zodiac signs (Rashi)
month_to_rashi = {
    "March": "Mesha",
    "April": "Vrishabha",
    "May": "Mithuna",
    "June": "Karka",
    "July": "Simha",
    "August": "Kanya",
    "September": "Tula",
    "October": "Vrischika",
    "November": "Dhanu",
    "December": "Makara",
    "January": "Kumbha",
    "February": "Meena"
}

# Take input from the user
birth_month = input("Enter your birth month: ").strip()

# Get the Rashi based on the input month
rashi = month_to_rashi.get(birth_month, "Invalid month")

# Print the Rashi name
print(rashi)
```

Output:



```
Enter your birth month: January
Kumbha
```

Explanation:

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

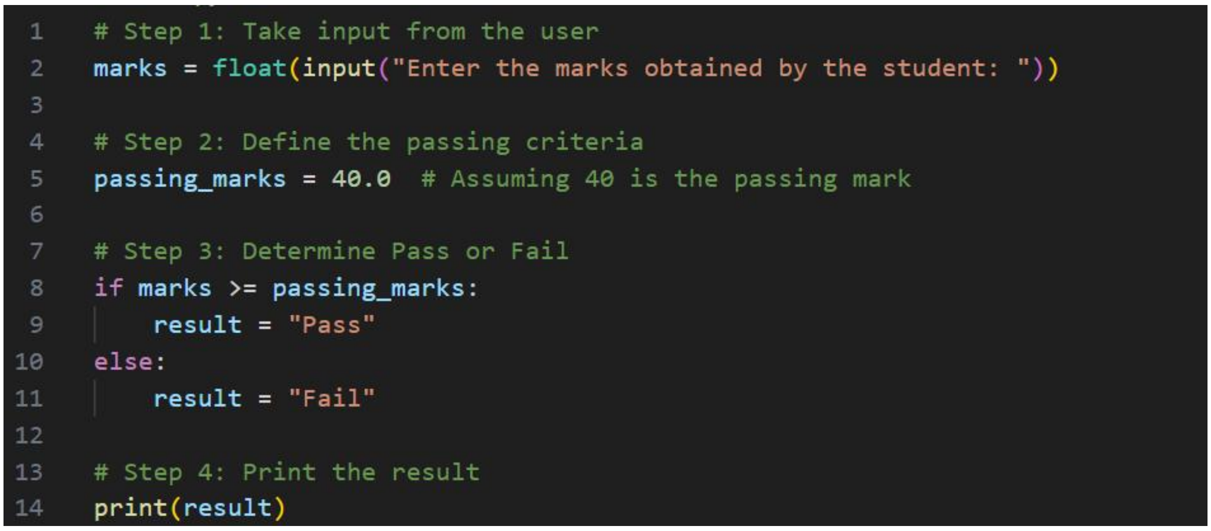
Question 5: Result Analysis Based on Marks

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:
Determine whether the student Passes or Fails based on marks.
Think step by step before giving the final answer.

At the end, print only Pass or Fail. Take Input From User.

Code:



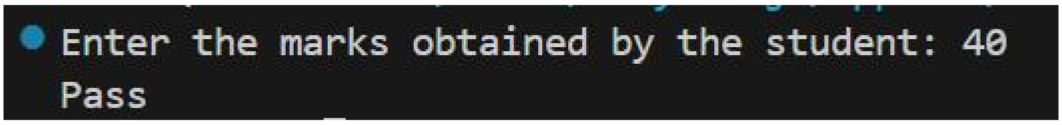
```python
# Step 1: Take input from the user
marks = float(input("Enter the marks obtained by the student: "))

# Step 2: Define the passing criteria
passing_marks = 40.0  # Assuming 40 is the passing mark

# Step 3: Determine Pass or Fail
if marks >= passing_marks:
    result = "Pass"
else:
    result = "Fail"

# Step 4: Print the result
print(result)
```

Output:



```
Enter the marks obtained by the student: 40
Pass
```

Explanation:

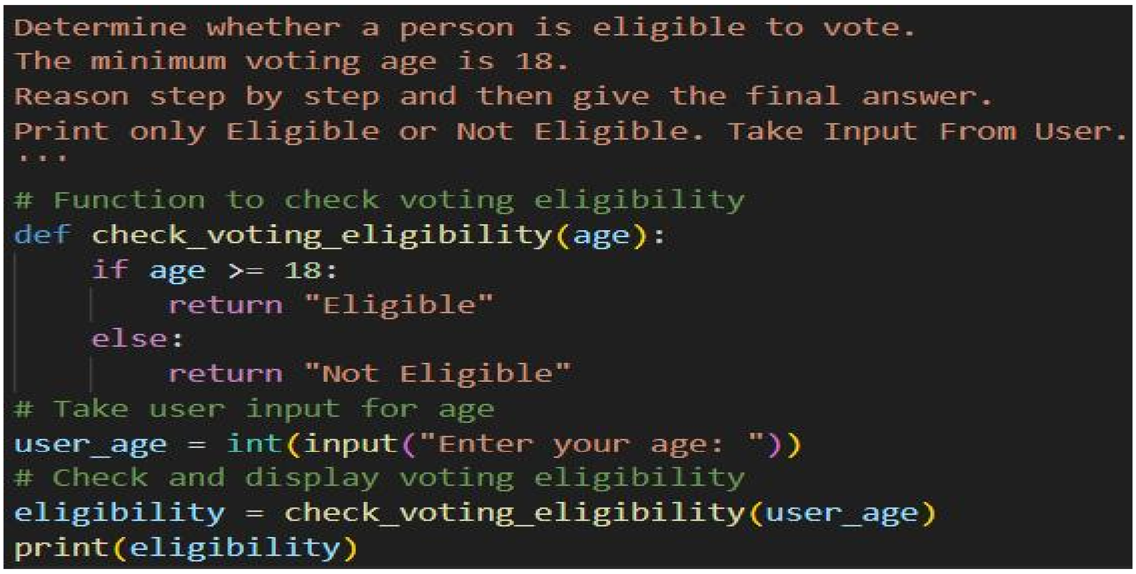
This code takes the marks obtained by a student as input and checks if the student has passed or failed based on a predefined passing criteria (40 marks). If the marks are greater than or equal to 40, it prints "Pass"; otherwise, it prints "Fail".

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

Task 6:

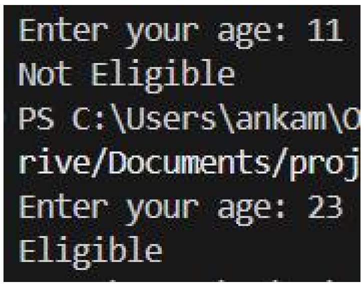
Write a Python program that determines whether a person is eligible to vote using Chain-of-Thought (CoT) prompting.

Prompt & Code:



```python
Determine whether a person is eligible to vote.
The minimum voting age is 18.
Reason step by step and then give the final answer.
Print only Eligible or Not Eligible. Take Input From User.
'''
# Function to check voting eligibility
def check_voting_eligibility(age):
    if age >= 18:
        return "Eligible"
    else:
        return "Not Eligible"
# Take user input for age
user_age = int(input("Enter your age: "))
# Check and display voting eligibility
eligibility = check_voting_eligibility(user_age)
print(eligibility)
```

Output:



```
Enter your age: 11
Not Eligible
PS C:\Users\ankam\O
rive/Documents/proj
Enter your age: 23
Eligible
```

Explanation:

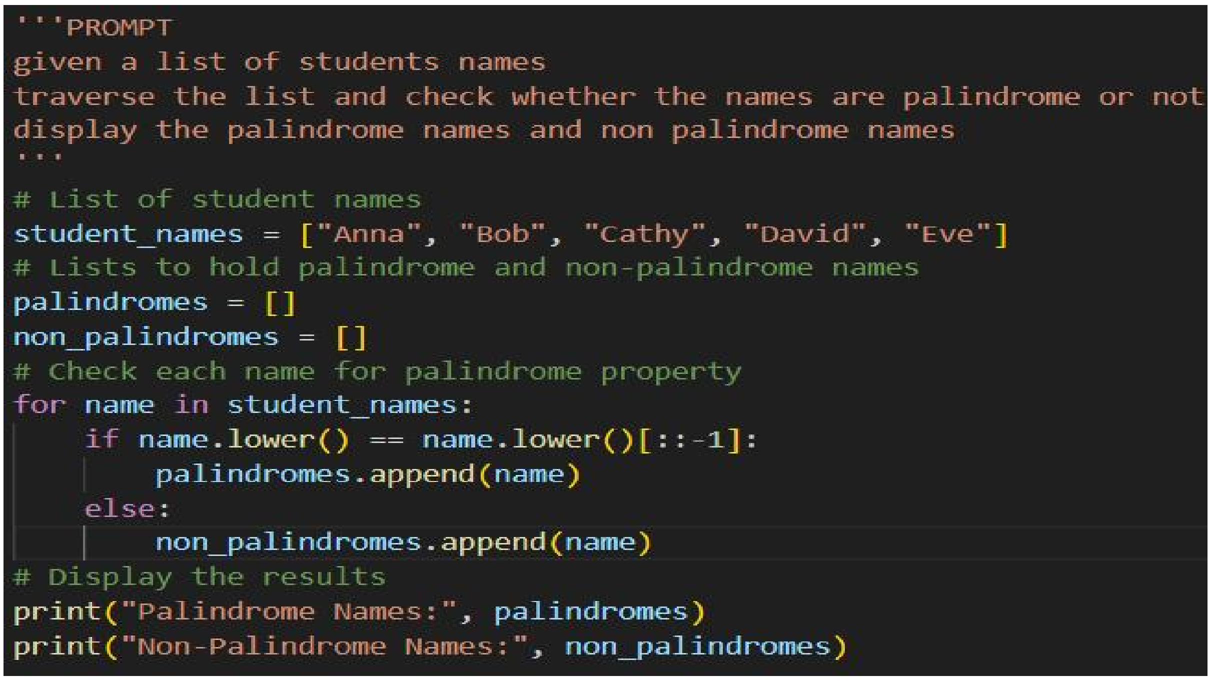
This code snippet checks if a user is eligible to vote based on their age. It prompts the user to enter their age and then evaluates whether the age is 18 or older. If the condition is met, it prints "Eligible"; otherwise, it prints "Not Eligible".

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

Task 7:

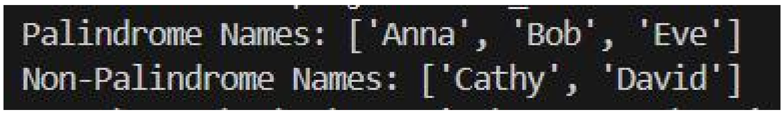
Write a Python program that uses the prompt chaining technique to identify palindrome names from a list of student names.

Prompt & Code:



```python
'''PROMPT
given a list of students names
traverse the list and check whether the names are palindrome or not
display the palindrome names and non palindrome names
'''
# List of student names
student_names = ["Anna", "Bob", "Cathy", "David", "Eve"]
# Lists to hold palindrome and non-palindrome names
palindromes = []
non_palindromes = []
# Check each name for palindrome property
for name in student_names:
    if name.lower() == name.lower()[::-1]:
        palindromes.append(name)
    else:
        non_palindromes.append(name)
# Display the results
print("Palindrome Names:", palindromes)
print("Non-Palindrome Names:", non_palindromes)
```

Output:



```
Palindrome Names: ['Anna', 'Bob', 'Eve']
Non-Palindrome Names: ['Cathy', 'David']
```

Explanation:

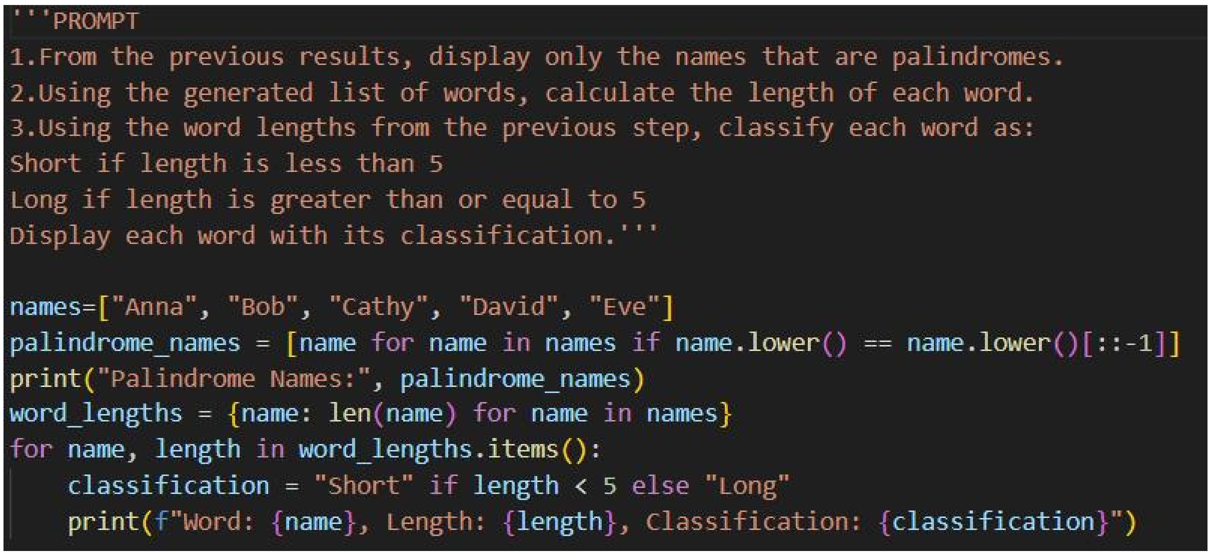
This code defines a list of student names and includes a function to check if a name is a palindrome (a word that reads the same backward as forward). It then filters the list to find all names that are palindromes and non palindromes and prints them. The use of list comprehension makes the filtering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

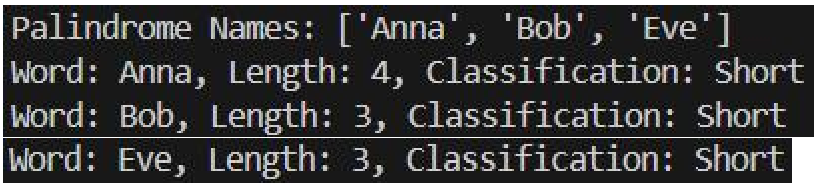
Prompt & Code:



```python
'''PROMPT
1.From the previous results, display only the names that are palindromes.
2.Using the generated list of words, calculate the length of each word.
3.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.'''

names=["Anna", "Bob", "Cathy", "David", "Eve"]
palindrome_names = [name for name in names if name.lower() == name.lower()[::-1]]
print("Palindrome Names:", palindrome_names)
word_lengths = {name: len(name) for name in names}
for name, length in word_lengths.items():
    classification = "Short" if length < 5 else "Long"
    print(f"Word: {name}, Length: {length}, Classification: {classification}")
```

Output:



```
Palindrome Names: ['Anna', 'Bob', 'Eve']
Word: Anna, Length: 4, Classification: Short
Word: Bob, Length: 3, Classification: Short
Word: Eve, Length: 3, 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.