Assignment-4

HT-NO: 2303A51963

Batch: 24

Q1. Zero-Shot Prompting (Basic Lab Task)

Write a Python function that classifies a given text as Spam or Not

Spam using zero-shot prompting.

Steps:

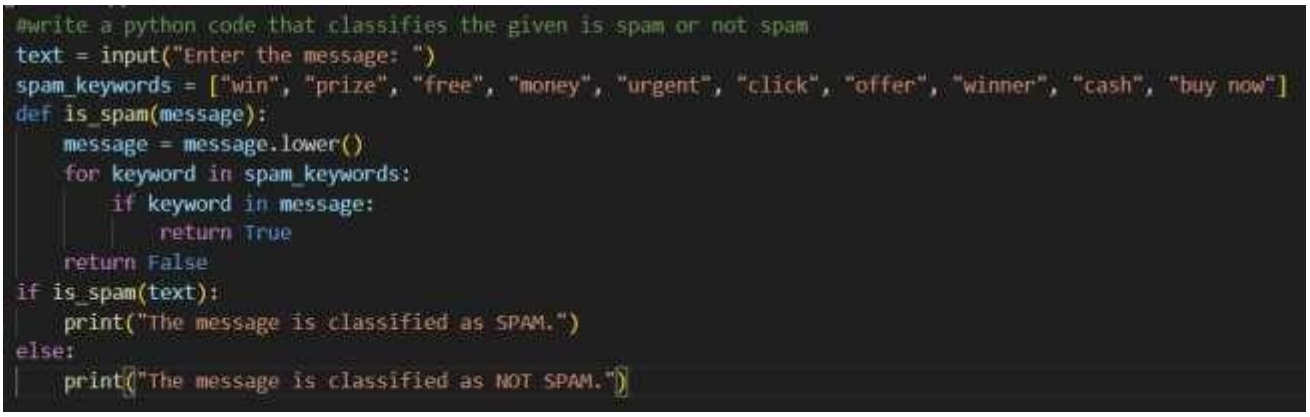
1. Construct a prompt without any examples.
2. Clearly specify the output labels.
3. Display only the predicted label.

Input:

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

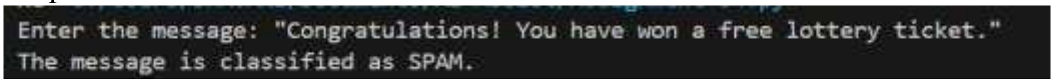
Expected Output:

Spam

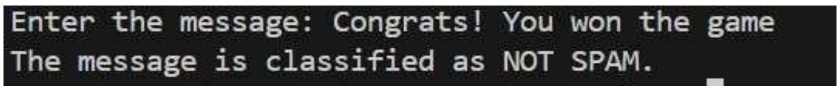


```python
#write a python code that classifies the given is spam or not spam
text = input("Enter the message: ")
spam_keywords = ["win", "prize", "free", "money", "urgent", "click", "offer", "winner", "cash", "buy now"]
def is_spam(message):
    message = message.lower()
    for keyword in spam_keywords:
        if keyword in message:
            return True
    return False
if is_spam(text):
    print("The message is classified as SPAM.")
else:
    print("The message is classified as NOT SPAM.")
```

Output:



```
Enter the message: "Congratulations! You have won a free lottery ticket."
The message is classified as SPAM.
```



```
Enter the message: Congrats! You won the game
The message is classified as NOT SPAM.
```

Q2. One-Shot Prompting (Emotion detection)

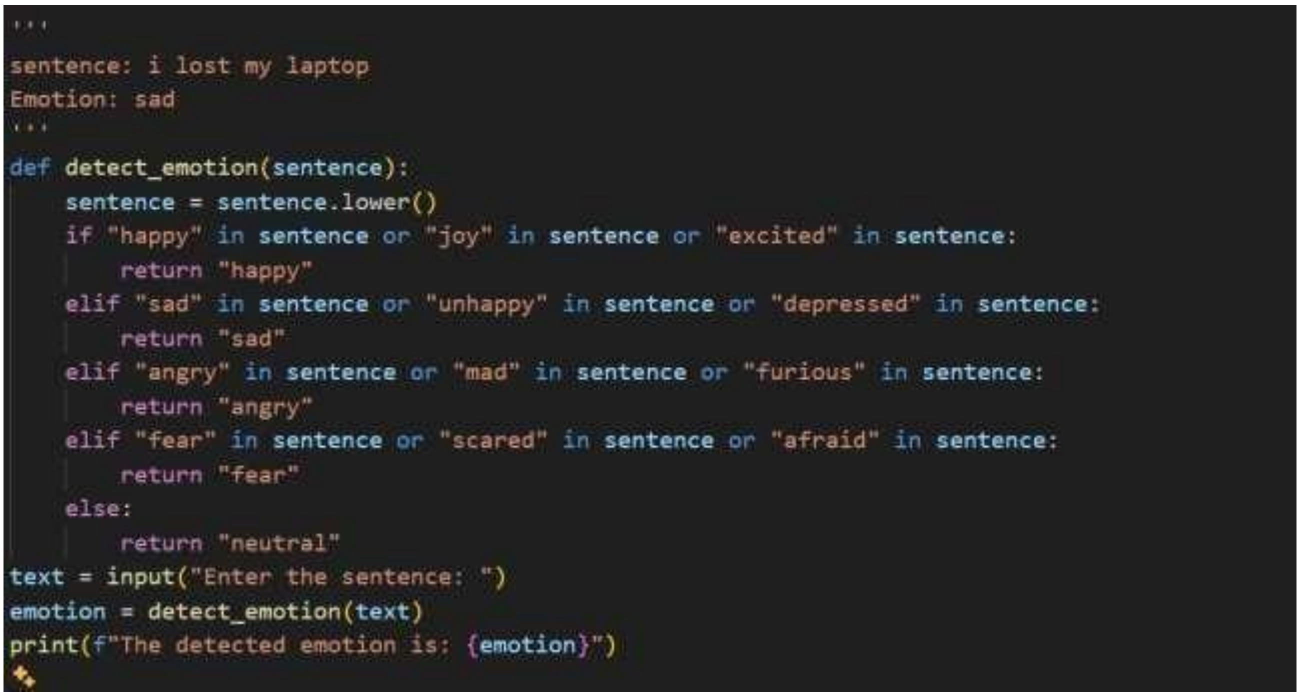
Task:

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

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

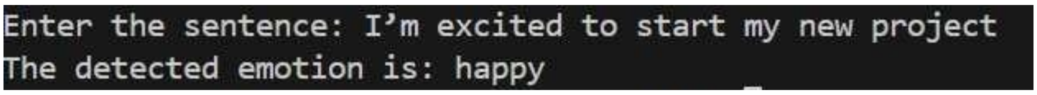
Steps:

1. Provide one labelled example inside the prompt.
2. Take a sentence as input.
3. Print the predicted emotion



```python
'''
sentence: i lost my laptop
Emotion: sad
'''
def detect_emotion(sentence):
    sentence = sentence.lower()
    if "happy" in sentence or "joy" in sentence or "excited" in sentence:
        return "happy"
    elif "sad" in sentence or "unhappy" in sentence or "depressed" in sentence:
        return "sad"
    elif "angry" in sentence or "mad" in sentence or "furious" in sentence:
        return "angry"
    elif "fear" in sentence or "scared" in sentence or "afraid" in sentence:
        return "fear"
    else:
        return "neutral"
text = input("Enter the sentence: ")
emotion = detect_emotion(text)
print(f"The detected emotion is: {emotion}")
```

Output:



```
Enter the sentence: I’m excited to start my new project
The detected emotion is: 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):

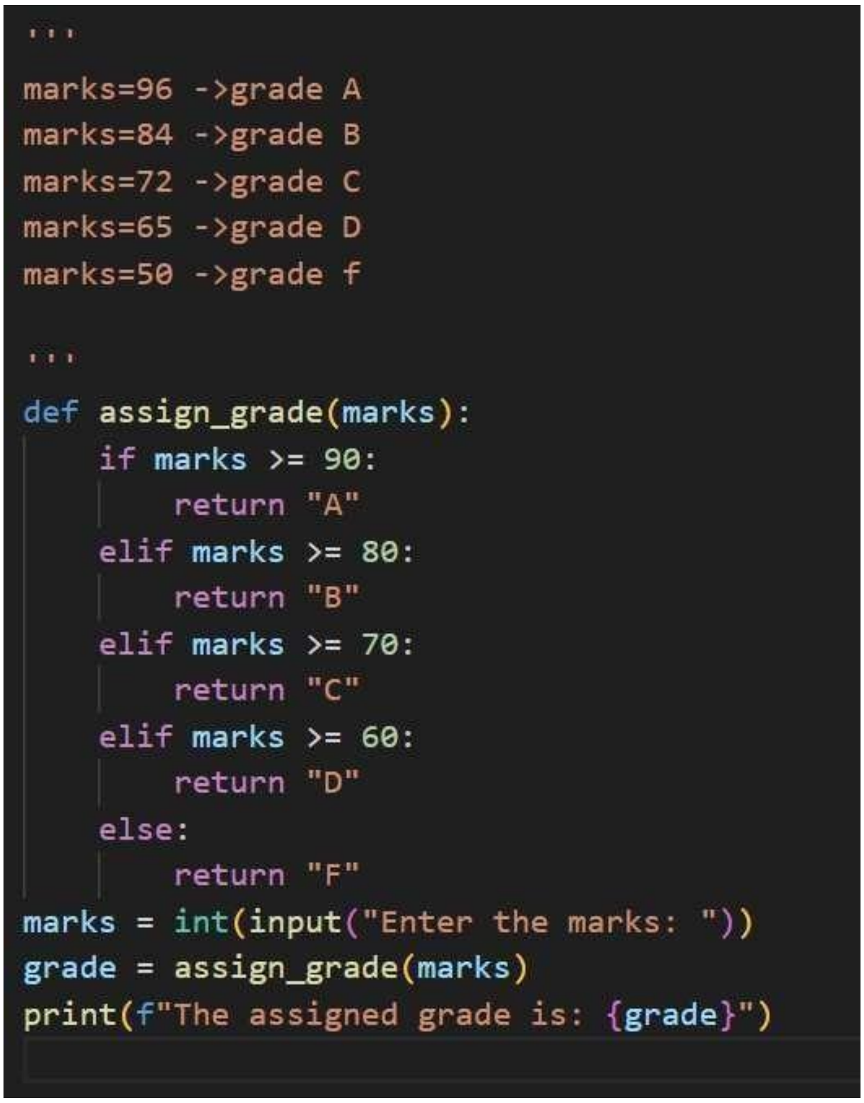
• 90–100 → A

• 80–89 → B

• 70–79 → C

• 60–69 → D

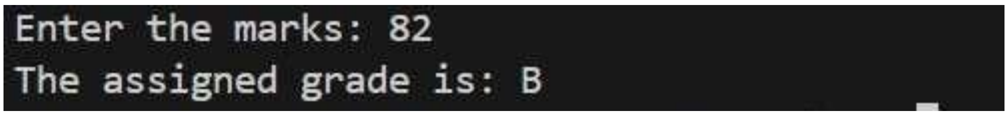
• Below 60 → F



```python
'''
marks=96 ->grade A
marks=84 ->grade B
marks=72 ->grade C
marks=65 ->grade D
marks=50 ->grade f

'''
def assign_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"
marks = int(input("Enter the marks: "))
grade = assign_grade(marks)
print(f"The assigned grade is: {grade}")
```

Output:



```
Enter the marks: 82
The assigned grade is: B
```

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

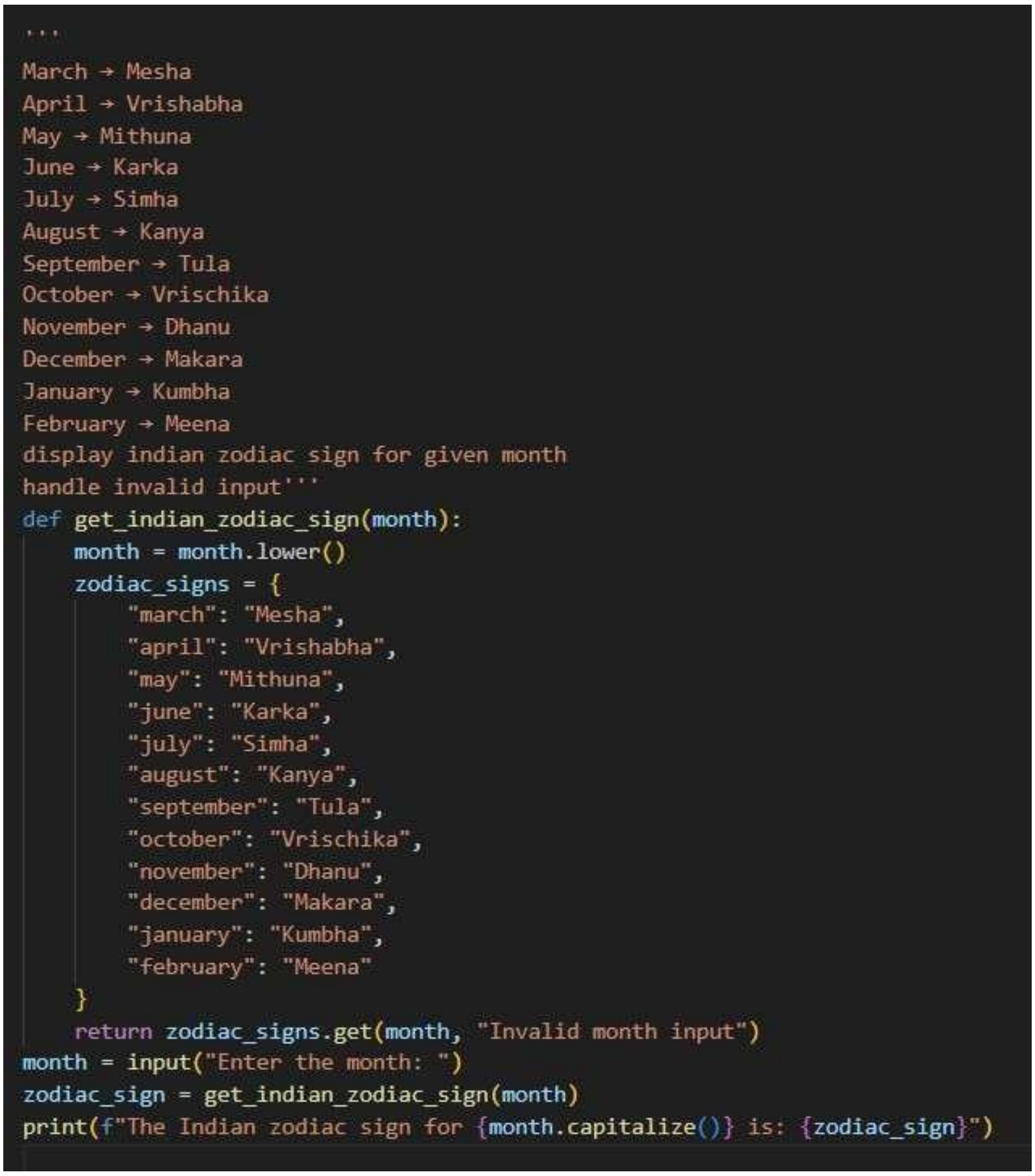
October → Vrischika

November → Dhanu

December → Makara

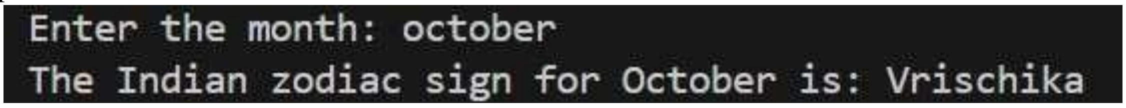
January → Kumbha

February → Meena



```python
'''
March → Mesha
April → Vrishabha
May → Mithuna
June → Karka
July → Simha
August → Kanya
September → Tula
October → Vrischika
November → Dhanu
December → Makara
January → Kumbha
February → Meena
display indian zodiac sign for given month
handle invalid input'''
def get_indian_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 input")
month = input("Enter the month: ")
zodiac_sign = get_indian_zodiac_sign(month)
print(f"The Indian zodiac sign for {month.capitalize()} is: {zodiac_sign}")
```

Output:



```
Enter the month: october
The Indian zodiac sign for October is: Vrischika
```

Q5. Result Analysis Based on Marks

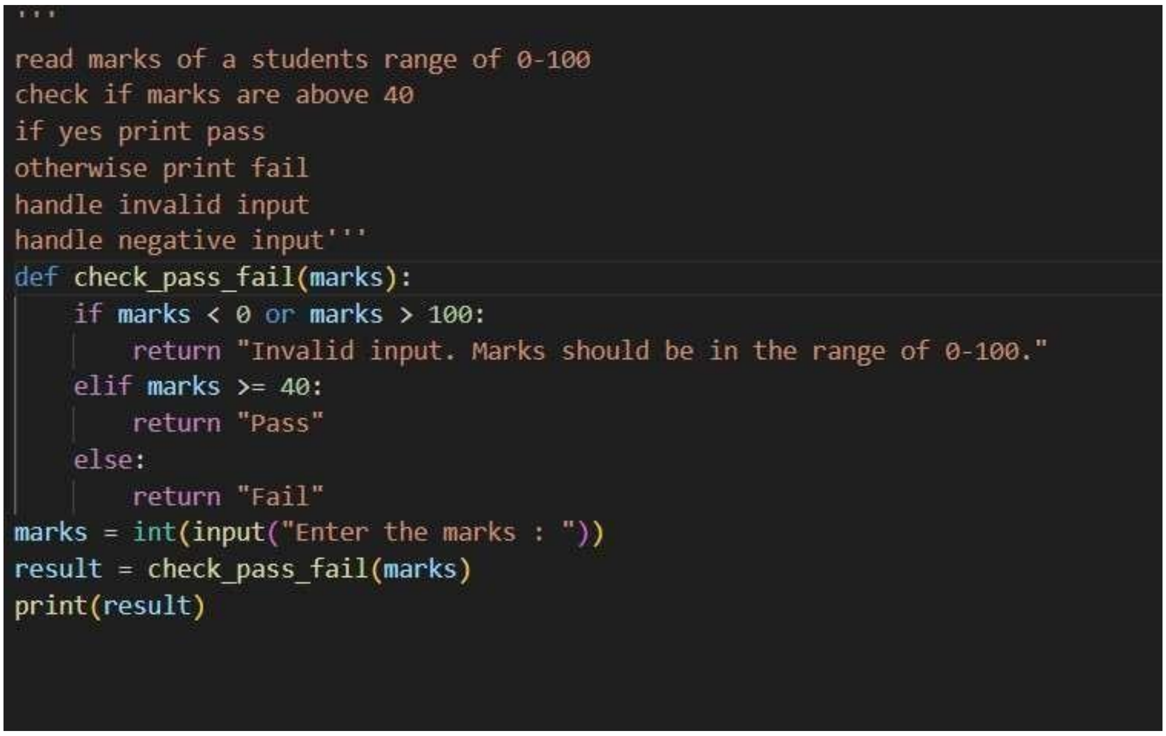
Task: Write a Python program that determines whether a student

Passes or Fails based on marks using Chain-of-Thought (CoT)

prompting.

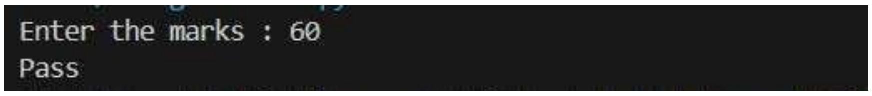
Result Categories:

['Pass', 'Fail']



```python
'''
read marks of a students range of 0-100
check if marks are above 40
if yes print pass
otherwise print fail
handle invalid input
handle negative input'''
def check_pass_fail(marks):
    if marks < 0 or marks > 100:
        return "Invalid input. Marks should be in the range of 0-100."
    elif marks >= 40:
        return "Pass"
    else:
        return "Fail"
marks = int(input("Enter the marks : "))
result = check_pass_fail(marks)
print(result)
```

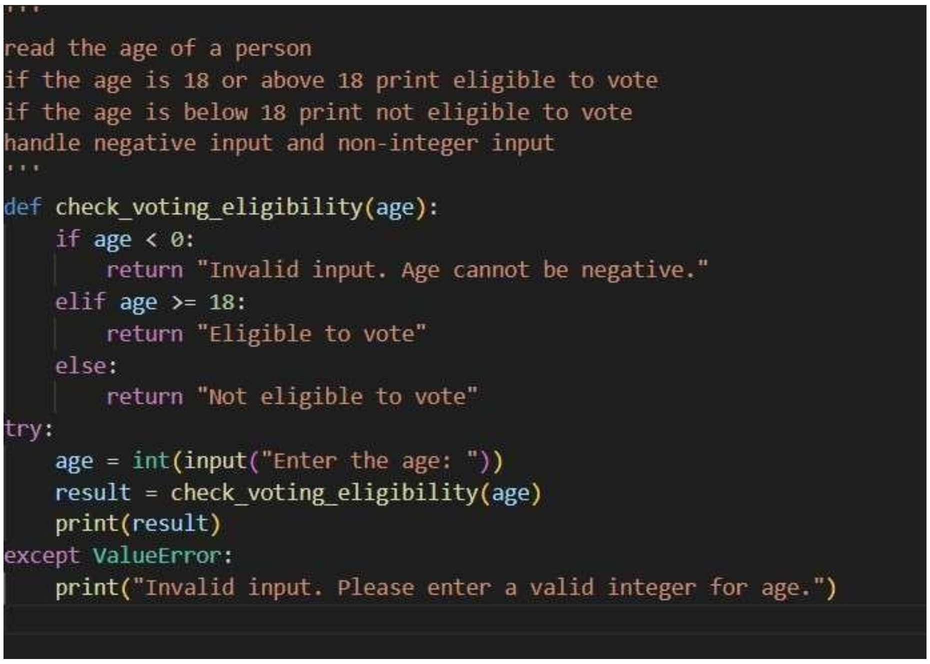
Output:



```
Enter the marks : 60
Pass
```

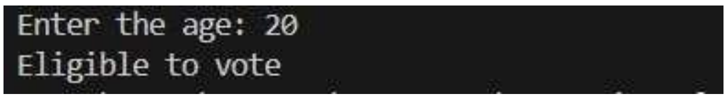
Q6 Voting Eligibility Check (Chain-of-Thought Prompting)

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



```
'''
read the age of a person
if the age is 18 or above 18 print eligible to vote
if the age is below 18 print not eligible to vote
handle negative input and non-integer input
'''
def check_voting_eligibility(age):
    if age < 0:
        return "Invalid input. Age cannot be negative."
    elif age >= 18:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"
try:
    age = int(input("Enter the age: "))
    result = check_voting_eligibility(age)
    print(result)
except ValueError:
    print("Invalid input. Please enter a valid integer for age.")
```

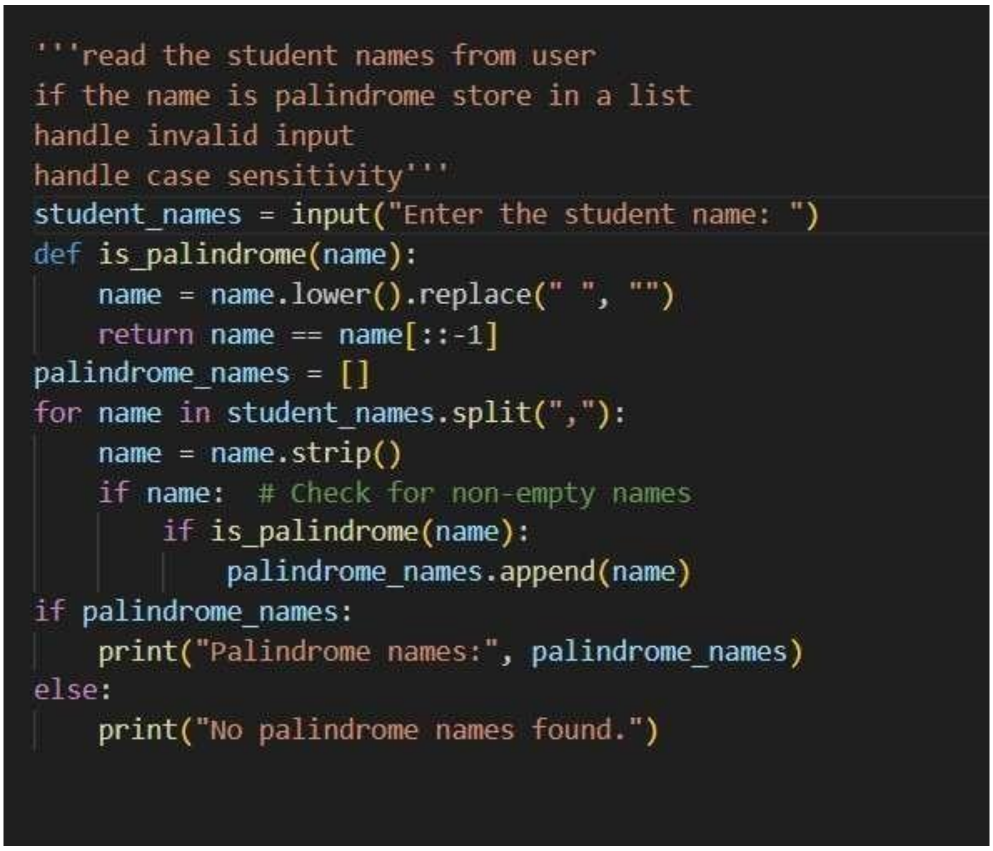
Output:



```
Enter the age: 20
Eligible to vote
```

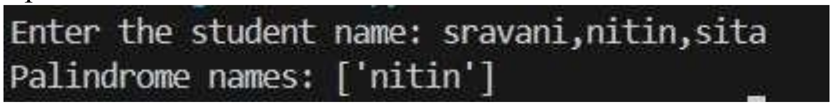
Q7 Prompt Chaining (String Processing – Palindrome Names)

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



```python
'''read the student names from user
if the name is palindrome store in a list
handle invalid input
handle case sensitivity'''
student_names = input("Enter the student name: ")
def is_palindrome(name):
    name = name.lower().replace(" ", "")
    return name == name[::-1]
palindrome_names = []
for name in student_names.split(","):
    name = name.strip()
    if name:  # Check for non-empty names
        if is_palindrome(name):
            palindrome_names.append(name)
if palindrome_names:
    print("Palindrome names:", palindrome_names)
else:
    print("No palindrome names found.")
```

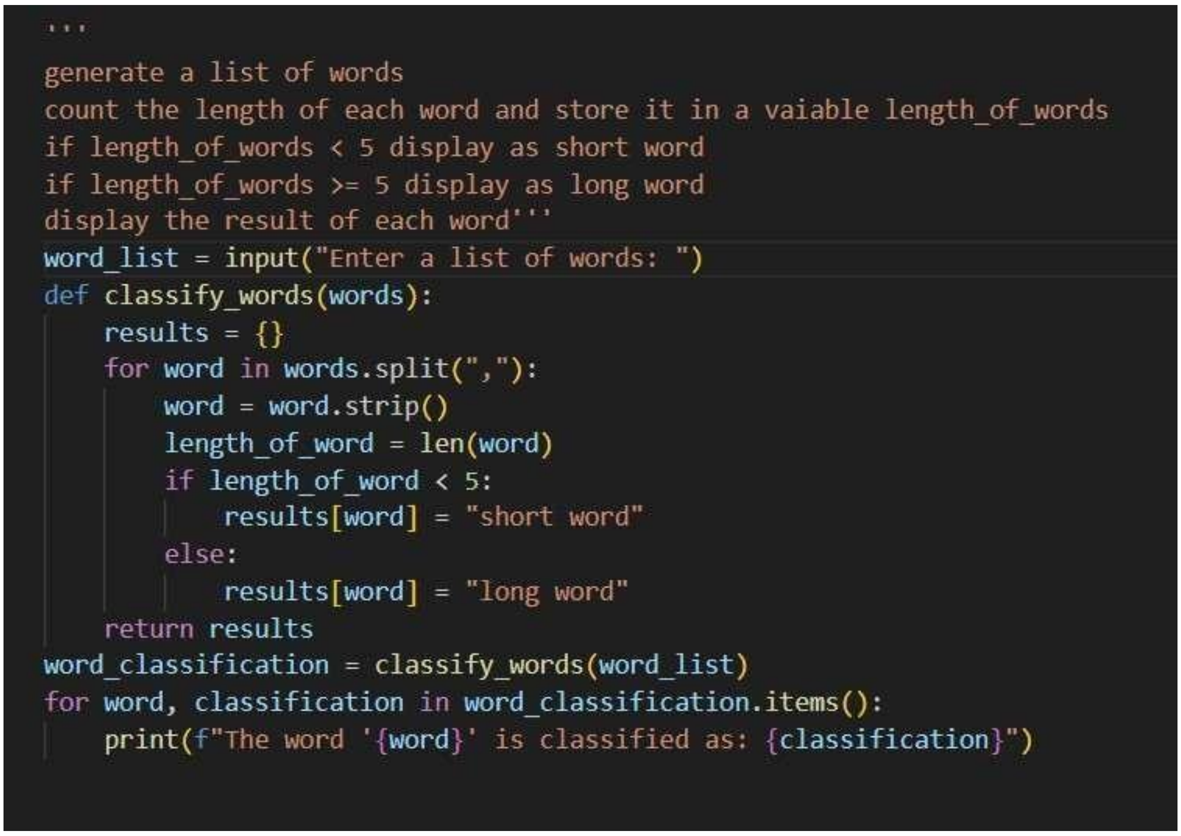
Output:



```
Enter the student name: sravani,nitin,sita
Palindrome names: ['nitin']
```

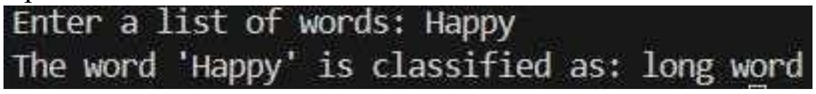
Q8 Prompt Chaining (String Processing – Word Length Analysis)

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 words
count the length of each word and store it in a vaiable length_of_words
if length_of_words < 5 display as short word
if length_of_words >= 5 display as long word
display the result of each word'''
word_list = input("Enter a list of words: ")
def classify_words(words):
    results = {}
    for word in words.split(","):
        word = word.strip()
        length_of_word = len(word)
        if length_of_word < 5:
            results[word] = "short word"
        else:
            results[word] = "long word"
    return results
word_classification = classify_words(word_list)
for word, classification in word_classification.items():
    print(f"The word '{word}' is classified as: {classification}")
```

Output:



```
Enter a list of words: Happy
The word 'Happy' is classified as: long word
```