ASSIGNMENT-4

NAME: Nithin Kumar Kondabattini

Batch-23 2303A51630

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.

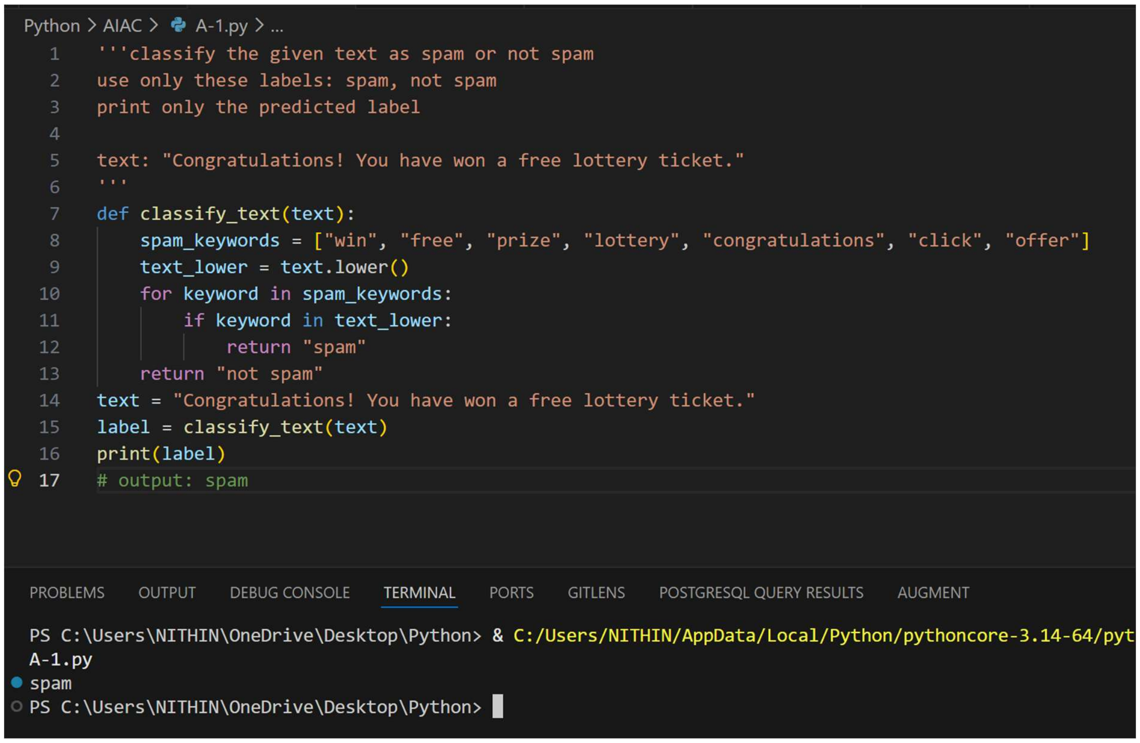
3. Display only the predicted label.

Input:

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

Expected Output:

Spam



Python > AIAC > A-1.py > ...

```python
'''classify the given text as spam or not spam
use only these labels: spam, not spam
print only the predicted label

text: "Congratulations! You have won a free lottery ticket."
'''
def classify_text(text):
    spam_keywords = ["win", "free", "prize", "lottery", "congratulations", "click", "offer"]
    text_lower = text.lower()
    for keyword in spam_keywords:
        if keyword in text_lower:
            return "spam"
    return "not spam"
text = "Congratulations! You have won a free lottery ticket."
label = classify_text(text)
print(label)
# output: spam
```

```
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Local/Python/pythoncore-3.14-64/pyt
A-1.py
spam
PS C:\Users\NITHIN\OneDrive\Desktop\Python>
```

Q2. One-Shot Prompting (Emotion detection)

Task:

Write a Python program that detects the emotion of a sentence

using one-shot prompting.

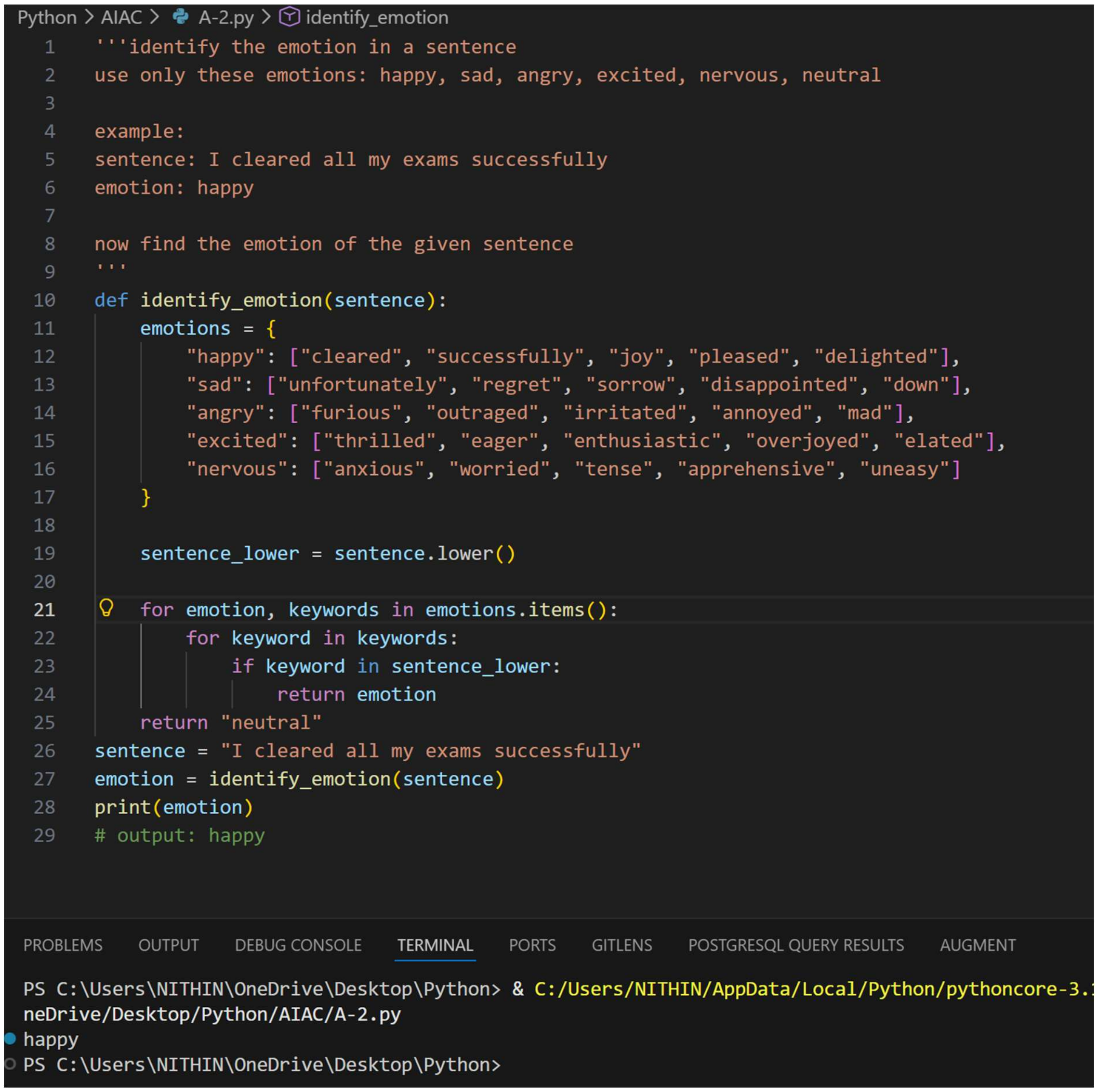
Emotions: [‘happy', 'sad', 'angry', 'excited', 'nervous', ’neutral’]

Steps:

1. Provide one labeled example inside the prompt.

2. Take a sentence as input.

3. Print the predicted emotion



```python
'''identify the emotion in a sentence
use only these emotions: happy, sad, angry, excited, nervous, neutral

example:
sentence: I cleared all my exams successfully
emotion: happy

now find the emotion of the given sentence
'''
def identify_emotion(sentence):
    emotions = {
        "happy": ["cleared", "successfully", "joy", "pleased", "delighted"],
        "sad": ["unfortunately", "regret", "sorrow", "disappointed", "down"],
        "angry": ["furious", "outraged", "irritated", "annoyed", "mad"],
        "excited": ["thrilled", "eager", "enthusiastic", "overjoyed", "elated"],
        "nervous": ["anxious", "worried", "tense", "apprehensive", "uneasy"]
    }

    sentence_lower = sentence.lower()

    for emotion, keywords in emotions.items():
        for keyword in keywords:
            if keyword in sentence_lower:
                return emotion
    return "neutral"
sentence = "I cleared all my exams successfully"
emotion = identify_emotion(sentence)
print(emotion)
# output: happy
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRESQL QUERY RESULTS AUGMENT

```
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Local/Python/pythoncore-3.
neDrive/Desktop/Python/AIAC/A-2.py
happy
PS C:\Users\NITHIN\OneDrive\Desktop\Python>
```

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):

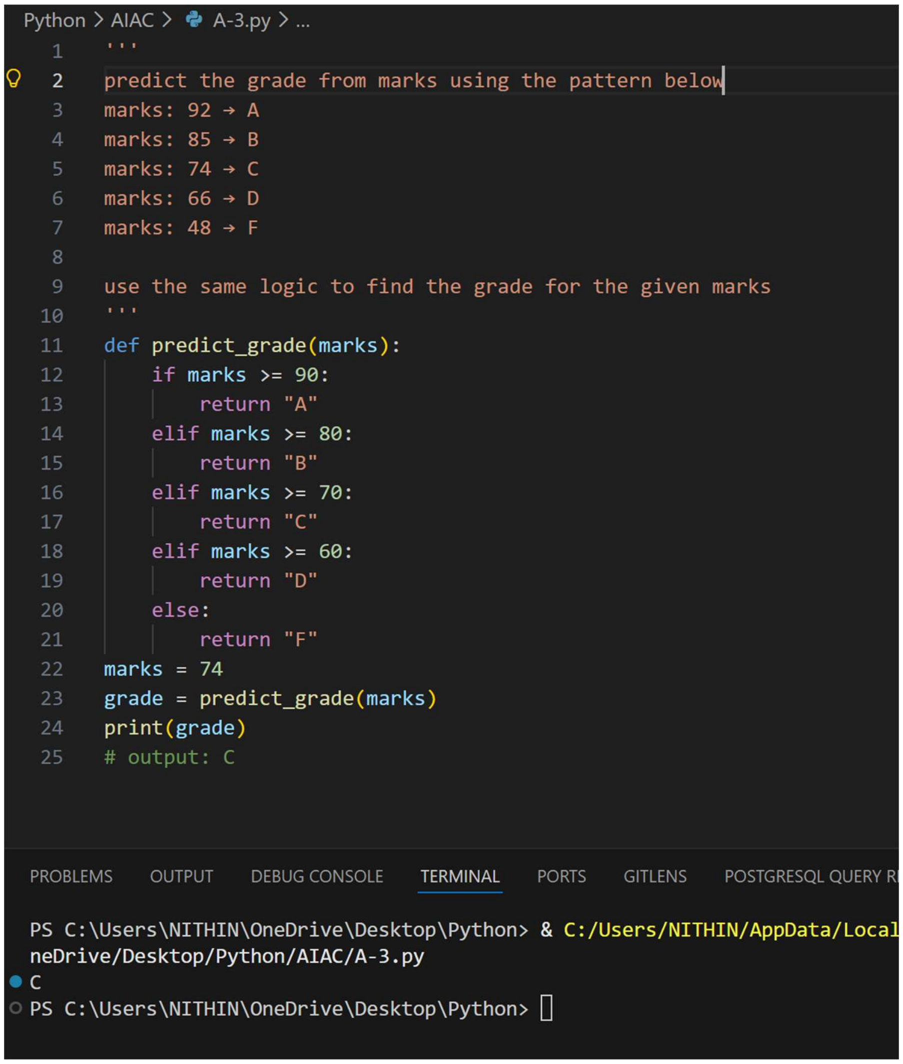
• 90–100 → A

• 80–89 → B

• 70–79 → C

• 60–69 → D

• Below 60 → F



```python
'''
predict the grade from marks using the pattern below
marks: 92 → A
marks: 85 → B
marks: 74 → C
marks: 66 → D
marks: 48 → F

use the same logic to find the grade for the given marks
'''
def predict_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 = 74
grade = predict_grade(marks)
print(grade)
# output: C
```

```
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Local
neDrive/Desktop/Python/AIAC/A-3.py
C
PS C:\Users\NITHIN\OneDrive\Desktop\Python>
```

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

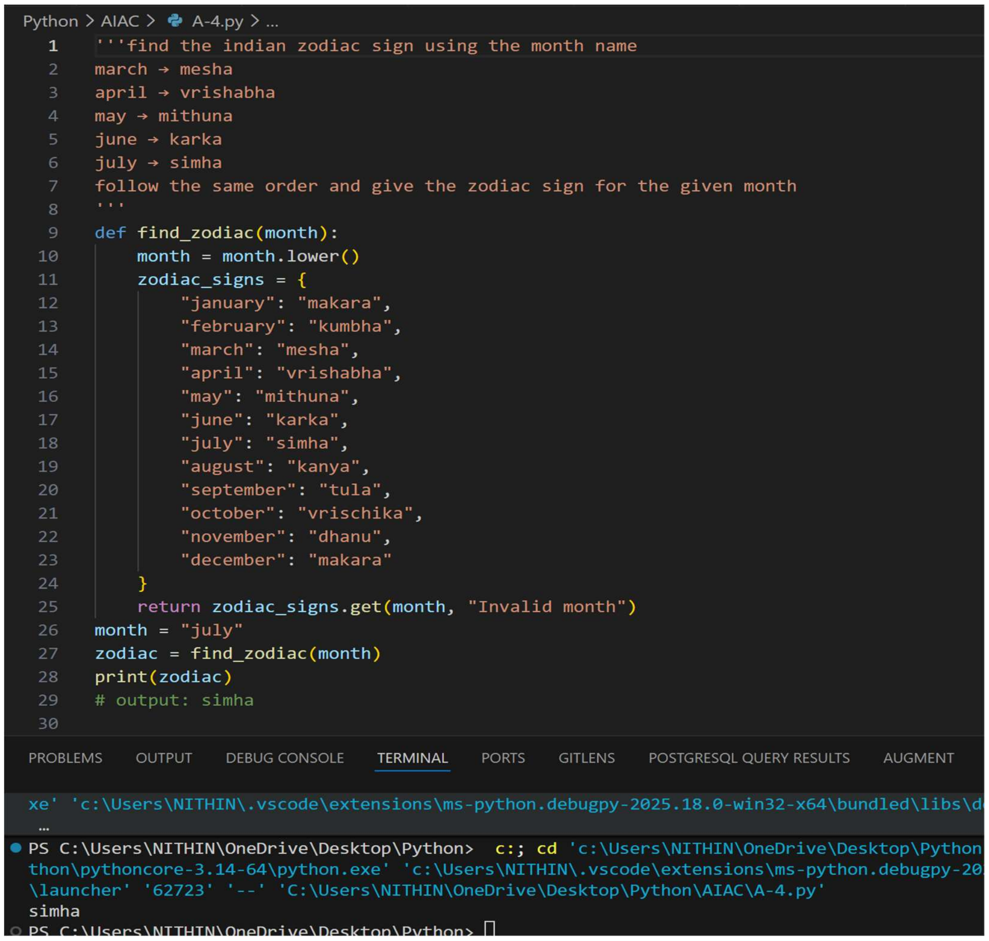
October → Vrischika

November → Dhanu

December → Makara

January → Kumbha

February → Meena



```python
'''find the indian zodiac sign using the month name
march → mesha
april → vrishabha
may → mithuna
june → karka
july → simha
follow the same order and give the zodiac sign for the given month
'''
def find_zodiac(month):
    month = month.lower()
    zodiac_signs = {
        "january": "makara",
        "february": "kumbha",
        "march": "mesha",
        "april": "vrishabha",
        "may": "mithuna",
        "june": "karka",
        "july": "simha",
        "august": "kanya",
        "september": "tula",
        "october": "vrischika",
        "november": "dhanu",
        "december": "makara"
    }
    return zodiac_signs.get(month, "Invalid month")
month = "july"
zodiac = find_zodiac(month)
print(zodiac)
# output: simha

```

```
xe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\d
 …
PS C:\Users\NITHIN\OneDrive\Desktop\Python>  c:; cd 'c:\Users\NITHIN\OneDrive\Desktop\Python
thon\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-20
\launcher' '62723' '--' 'C:\Users\NITHIN\OneDrive\Desktop\Python\AIAC\A-4.py'
simha
PS C:\Users\NITHIN\OneDrive\Desktop\Python> 
```

Q5. Result Analysis Based on Marks

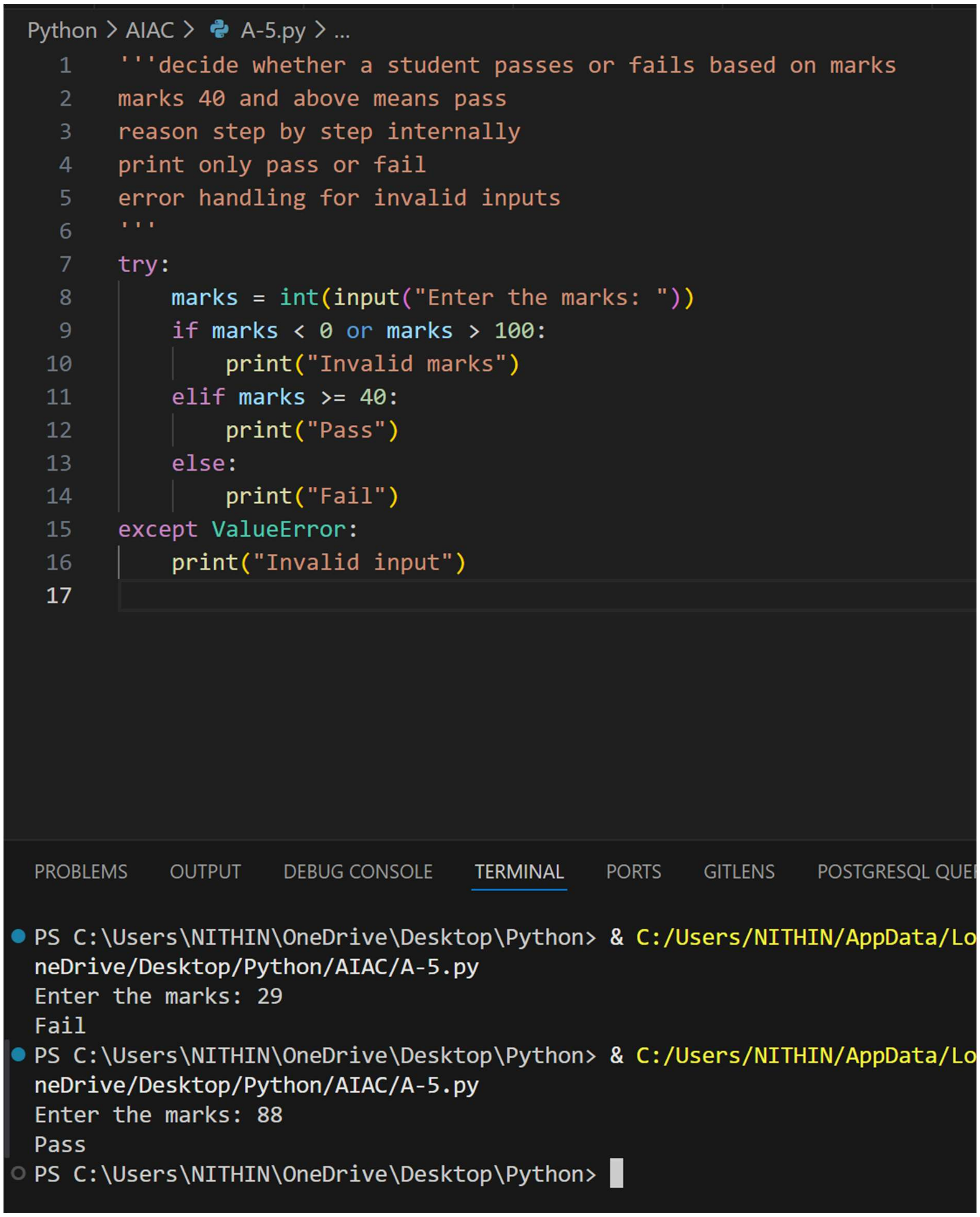
Task: Write a Python program that determines whether a student

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

prompting.

Result Categories:

['Pass', 'Fail']



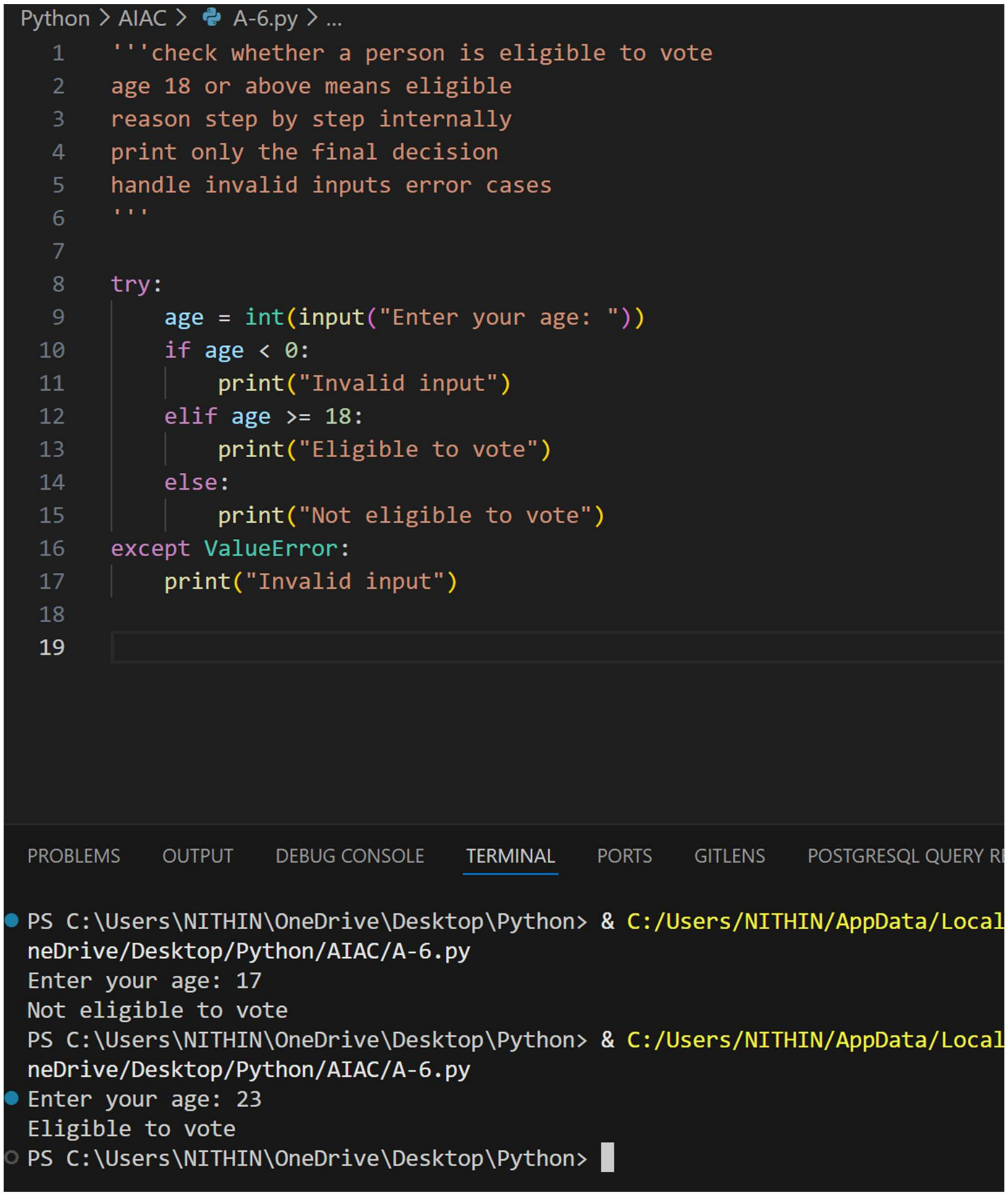
```python
'''decide whether a student passes or fails based on marks
marks 40 and above means pass
reason step by step internally
print only pass or fail
error handling for invalid inputs
'''
try:
    marks = int(input("Enter the marks: "))
    if marks < 0 or marks > 100:
        print("Invalid marks")
    elif marks >= 40:
        print("Pass")
    else:
        print("Fail")
except ValueError:
    print("Invalid input")

```

```
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Lo
neDrive/Desktop/Python/AIAC/A-5.py
Enter the marks: 29
Fail
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Lo
neDrive/Desktop/Python/AIAC/A-5.py
Enter the marks: 88
Pass
PS C:\Users\NITHIN\OneDrive\Desktop\Python>
```

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.



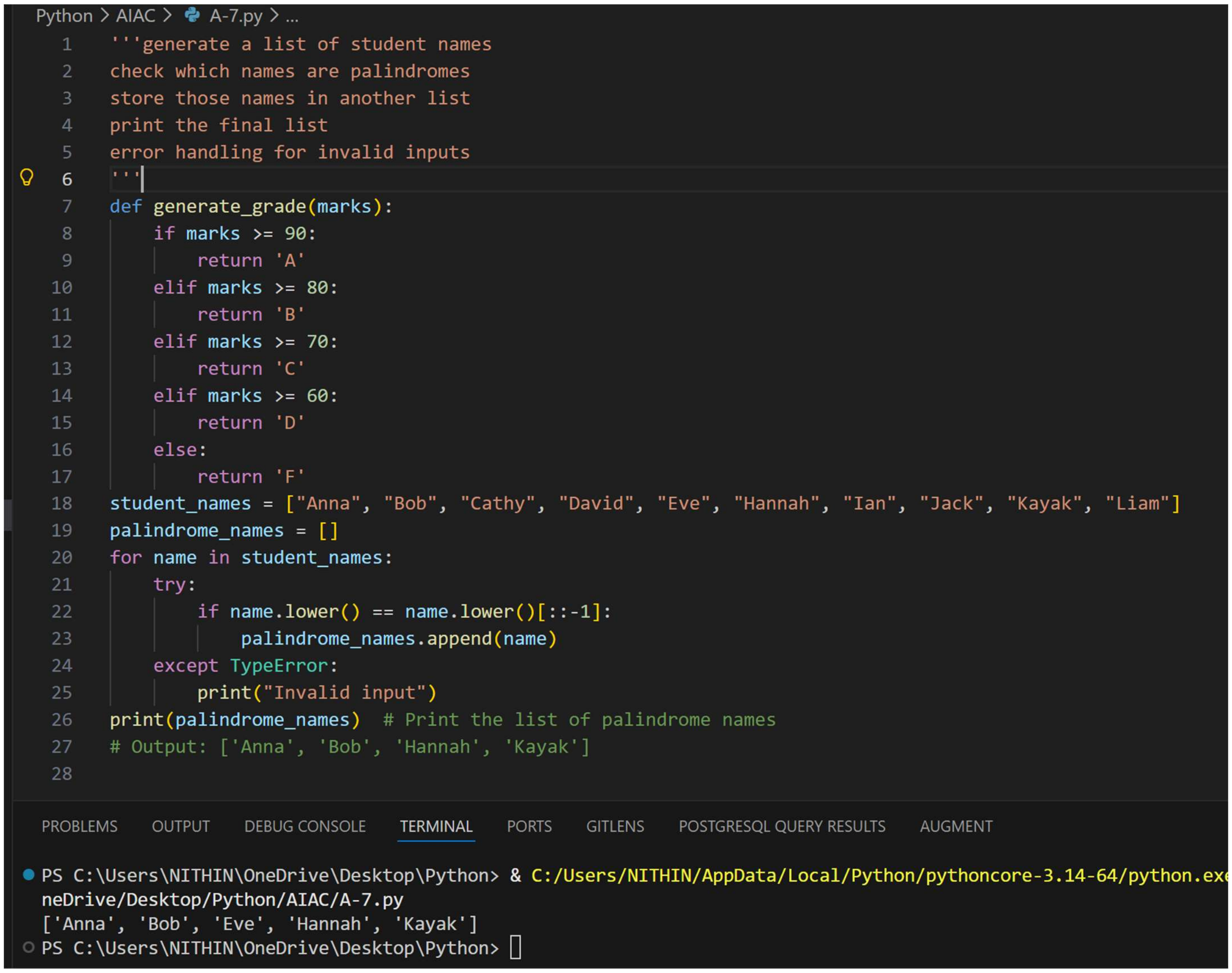
Python > AIAC > A-6.py > ...

```python
'''check whether a person is eligible to vote
age 18 or above means eligible
reason step by step internally
print only the final decision
handle invalid inputs error cases
'''

try:
    age = int(input("Enter your age: "))
    if age < 0:
        print("Invalid input")
    elif age >= 18:
        print("Eligible to vote")
    else:
        print("Not eligible to vote")
except ValueError:
    print("Invalid input")

```

```
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Local
neDrive/Desktop/Python/AIAC/A-6.py
Enter your age: 17
Not eligible to vote
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Local
neDrive/Desktop/Python/AIAC/A-6.py
Enter your age: 23
Eligible to vote
PS C:\Users\NITHIN\OneDrive\Desktop\Python>
```

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 > AIAC > A-7.py > ...

```python
'''generate a list of student names
check which names are palindromes
store those names in another list
print the final list
error handling for invalid inputs
'''
def generate_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'
student_names = ["Anna", "Bob", "Cathy", "David", "Eve", "Hannah", "Ian", "Jack", "Kayak", "Liam"]
palindrome_names = []
for name in student_names:
    try:
        if name.lower() == name.lower()[::-1]:
            palindrome_names.append(name)
    except TypeError:
        print("Invalid input")
print(palindrome_names)  # Print the list of palindrome names
# Output: ['Anna', 'Bob', 'Hannah', 'Kayak']

```

```
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Local/Python/pythoncore-3.14-64/python.exe
neDrive/Desktop/Python/AIAC/A-7.py
['Anna', 'Bob', 'Eve', 'Hannah', 'Kayak']
PS C:\Users\NITHIN\OneDrive\Desktop\Python>
```

Q8 Prompt Chaining (String Processing – Word Length

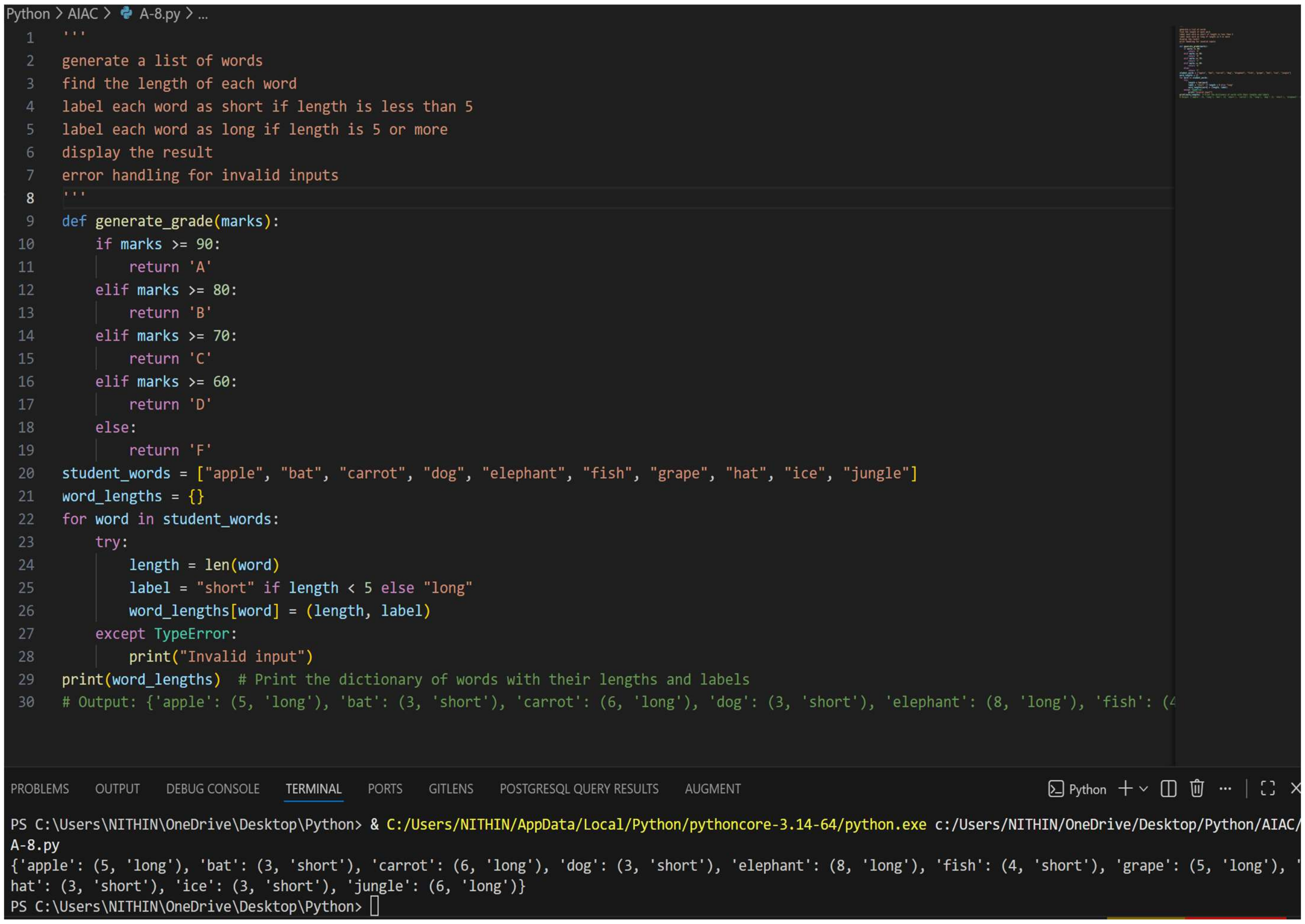
Analysis)

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

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



```python
'''
generate a list of words
find the length of each word
label each word as short if length is less than 5
label each word as long if length is 5 or more
display the result
error handling for invalid inputs
'''
def generate_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'
student_words = ["apple", "bat", "carrot", "dog", "elephant", "fish", "grape", "hat", "ice", "jungle"]
word_lengths = {}
for word in student_words:
    try:
        length = len(word)
        label = "short" if length < 5 else "long"
        word_lengths[word] = (length, label)
    except TypeError:
        print("Invalid input")
print(word_lengths)  # Print the dictionary of words with their lengths and labels
# Output: {'apple': (5, 'long'), 'bat': (3, 'short'), 'carrot': (6, 'long'), 'dog': (3, 'short'), 'elephant': (8, 'long'), 'fish': (4
```

```
PS C:\Users\NITHIN\OneDrive\Desktop\Python> & C:/Users/NITHIN/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/NITHIN/OneDrive/Desktop/Python/AIAC/
A-8.py
{'apple': (5, 'long'), 'bat': (3, 'short'), 'carrot': (6, 'long'), 'dog': (3, 'short'), 'elephant': (8, 'long'), 'fish': (4, 'short'), 'grape': (5, 'long'), '
hat': (3, 'short'), 'ice': (3, 'short'), 'jungle': (6, 'long')}
PS C:\Users\NITHIN\OneDrive\Desktop\Python> 
```