

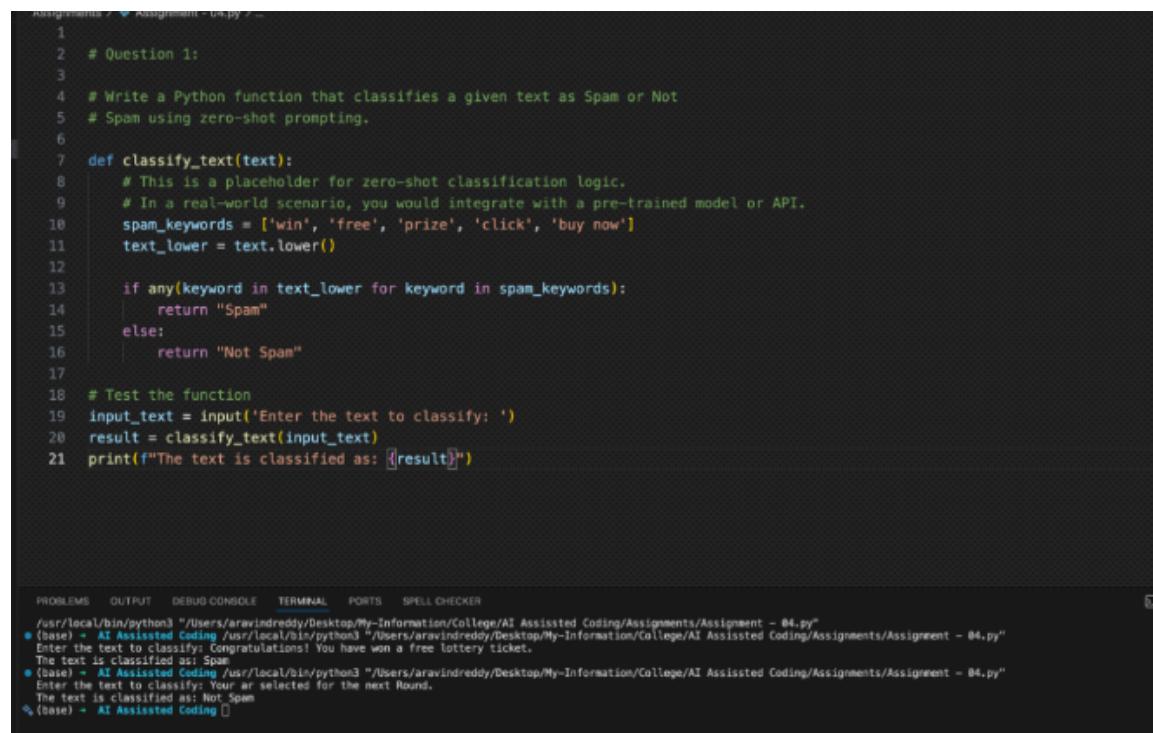
# Assignment-4.1

Name:Naga Rishik Reddy

Hall ticket:2303A51089

Batch - 02

## Q1. Zero-Shot Prompting (Basic Lab Task)



```
Assignment / Assignment - 04.py / ...
1
2 # Question 1:
3
4 # Write a Python function that classifies a given text as Spam or Not
5 # Spam using zero-shot prompting.
6
7 def classify_text(text):
8     # This is a placeholder for zero-shot classification logic.
9     # In a real-world scenario, you would integrate with a pre-trained model or API.
10    spam_keywords = ['win', 'free', 'prize', 'click', 'buy now']
11    text_lower = text.lower()
12
13    if any(keyword in text_lower for keyword in spam_keywords):
14        return "Spam"
15    else:
16        return "Not Spam"
17
18 # Test the function
19 input_text = input('Enter the text to classify: ')
20 result = classify_text(input_text)
21 print(f"The text is classified as: {result}")
```

The screenshot shows a code editor window with the file 'Assignment - 04.py' open. The code defines a function 'classify\_text' that checks if a given text contains any keywords from a list ('win', 'free', 'prize', 'click', 'buy now'). If it does, it returns 'Spam'; otherwise, it returns 'Not Spam'. The code then tests this function by taking user input and printing the result. The terminal output at the bottom shows two runs of the program. In the first run, the user enters 'The text is classified as: Spam'. In the second run, the user enters 'The text is classified as: Not Spam'.

## Q2. One-Shot Prompting (Emotion detection)

```
Assignment-1.py > ...
2 #write a python program that detects the emotion of a sentence using one-shot prompting
3 #emotions: ['happy', 'sad', 'angry', 'neutral']
4 def detect_emotion(sentence):
5     #this is a placeholder for one-shot prompting logic
6     #in a real-world scenario, you would integrate with a pre-trained model or API
7     emotion_keywords = {
8         'happy': ['joy', 'glad', 'excited', 'pleased'],
9         'sad': ['unhappy', 'sorrow', 'depressed', 'down'],
10        'angry': ['mad', 'furious', 'irritated', 'annoyed'],
11        'neutral': []
12    }
13
14    sentence_lower = sentence.lower()
15    for emotion, keywords in emotion_keywords.items():
16        if any(keyword in sentence_lower for keyword in keywords):
17            return emotion.capitalize()
18
19    return "Neutral"
20 #test the function
21 input_sentence = input('Enter a sentence to detect emotion: ')
22 emotion_result = detect_emotion(input_sentence)
23 print(f"The detected emotion is: {emotion_result}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | X
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
Enter a sentence to detect emotion: mad
The detected emotion is: Angry
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
Enter a sentence to detect emotion: excited
The detected emotion is: Happy
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> 
```

### Q3. Few-Shot Prompting (Student Grading Based on Marks)

```
Assignment-1.py > ...
1 #write a python program that predicts that predicts a student's based on marks using few-shot prompting
2 #grades: ['A', 'B', 'C', 'D', 'F']
3 #grading criteria(to be inferred from examples):
4 #90-100: A
5 #80-89: B
6 #70-79: C
7 #60-69: D
8 #below 60: F
9 def predict_grade(marks):
10    if marks >= 90:
11        return 'A'
12    elif marks >= 80:
13        return 'B'
14    elif marks >= 70:
15        return 'C'
16    elif marks >= 60:
17        return 'D'
18    else:
19        return 'F'
20 #test the function
21 marks = float(input("Enter the student's marks: "))
22 grade = predict_grade(marks)
23 print(f"The predicted grade for the student is: {grade}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | X
Enter the student's marks: 41
The predicted grade for the student is: F
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
Enter the student's marks: 95
The predicted grade for the student is: A
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> 
```

## Q4. Multi-Shot Prompting (Indian Zodiac Sign Prediction using Month Name)

The screenshot shows a Jupyter Notebook interface with the following content:

```
Assignment-1.py > ...
1 #write a python program that predicts a persons's Indian zodiac sign(Rashi) based on the month of birth (month)
2 #Indian zodiac order(simplified month-based model): the indian
3 #zodiac cycle starts in march with mesha and follows this order:
4 #march - mesha (aries)
5 #april - vrishabha (taurus)
6 #may - mithuna (gemini)
7 #june - karka (cancer)
8 #july - simha (leo)
9 #august - kanya (virgo)
10 #september - tula (libra)
11 #october - vrishchika (scorpio)
12 #november - dhanu (sagittarius)
13 #december - makara (capricorn)
14 #january - kumbha (aquarius)
15 #february - meena (pisces)
16 def predict_rashi(month):
17     month = month.lower()
18     if month == 'march':
19         return 'Mesha (Aries)'
20     elif month == 'april':
21         return 'Vrishabha (Taurus)'
22     elif month == 'may':
23         return 'Mithuna (Gemini)'
24     elif month == 'june':
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | []

```
Enter the month of birth to predict the Indian zodiac sign (Rashi): january
The predicted Indian zodiac sign (Rashi) for the month of january is: Kumbha (Aquarius)
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
Enter the month of birth to predict the Indian zodiac sign (Rashi): november
The predicted Indian zodiac sign (Rashi) for the month of november is: Dhanu (Sagittarius)
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> | powershell | Python | Activate Windows | Go to Settings to activate Windows.
```

## Q5. Result Analysis Based on Marks

```
Assignment-1.py > ...
1
2 step 1: read input from user and store it in a variable marks
3 step 2: the marks variable value should be in between 0-100
4 step 3: based on the marks value, display the grade using the following criteria
5 if marks>=35 and marks<=100 display "pass"
6 if marks<35 and marks>=0 display "fail"
7 if student absent display "apply for re-exam"
8
9 ...
10 marks = input("Enter the marks (or type 'absent' if the student is absent): ")
11 if marks.lower() == 'absent':
12     print("Apply for re-exam")
13 else:
14     try:
15         marks = float(marks)
16         if 0 <= marks <= 100:
17             if marks >= 35:
18                 print("Pass")
19             else:
20                 print("Fail")
21         else:
22             print("Marks should be between 0 and 100.")
23     except ValueError:
24         print("Invalid input. Please enter a number or 'absent'.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | 🔍 x

```
Enter the marks (or type 'absent' if the student is absent): 99
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
Enter the marks (or type 'absent' if the student is absent): absent
Apply for re-exam
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> 🔍
```

Activate Windows  
Go to Settings to activate Windows.

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

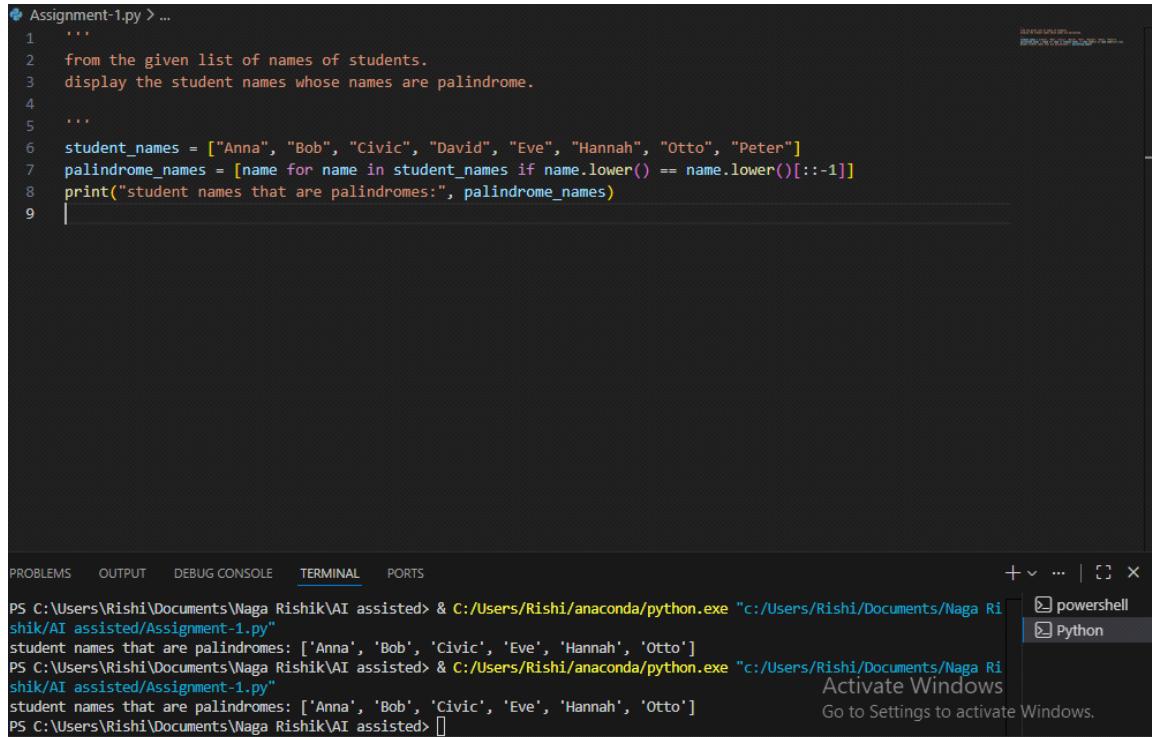
```
Assignment-1.py > ...
1
2 step 1: read input user and store it in a variable age
3 step 2: the age variable should be in between 0 - 120
4 step 3: based on the age value, display the life stage using the following criteria:
5 if age>=18 you are eligible to vote
6 if age<18 and age>=0 you are not eligible to vote
7
8 age = int(input("Enter your age: "))
9 if age >= 18 and age <= 120:
10     print("You are eligible to vote.")
11 elif age >= 0 and age < 18:
12     print("You are not eligible to vote.")
13 else:
14     print("Invalid age. Please enter a value between 0 and 120.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | 🔍 x

```
Enter your age: 13
You are not eligible to vote.
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
Enter your age: 25
You are eligible to vote.
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> 🔍
```

Activate Windows  
Go to Settings to activate Windows.

## Q7 Prompt Chaining (String Processing – Palindrome Names)



The screenshot shows a code editor window with a dark theme. A Python script named 'Assignment-1.py' is open. The code defines a list of student names and filters it to find palindromes. The terminal below shows the script running and outputting the list of palindromes.

```
Assignment-1.py > ...
1 ...
2 from the given list of names of students.
3 display the student names whose names are palindrome.
4 ...
5 ...
6 student_names = ["Anna", "Bob", "Civic", "David", "Eve", "Hannah", "Otto", "Peter"]
7 palindrome_names = [name for name in student_names if name.lower() == name.lower()[::-1]]
8 print("student names that are palindromes:", palindrome_names)
9 
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
student names that are palindromes: ['Anna', 'Bob', 'Civic', 'Eve', 'Hannah', 'Otto']
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> & C:/Users/Rishi/anaconda/python.exe "c:/Users/Rishi/Documents/Naga Rishik/AI assisted/Assignment-1.py"
student names that are palindromes: ['Anna', 'Bob', 'Civic', 'Eve', 'Hannah', 'Otto']
PS C:\Users\Rishi\Documents\Naga Rishik\AI assisted> 
```

+ v ... | [] x

powershell  
Python

Activate Windows  
Go to Settings to activate Windows.

## Q8 Prompt Chaining (String Processing – Word Length Analysis)

```
Assignment-1.py > ...
1  ...
2  generate a list of words in a list.
3  traverse through the list and find the length of the word.
4  finally, display the words as 'long' whose length is 5 or more else display as 'short'
5
6  ...
7  words = ["hello", "world", "python", "is", "great", "mathematics"]
8  for word in words:
9      if len(word) >= 5:
10         print(f"{word} is long")
11     else:
12         print(f"{word} is short")
13

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    + v ... | [ ]
powershell
Python
Activate Windows
Go to Settings to activate Windows.
In 13 Col 9 Spaces: 4 UFT-8 CR LF { } Python 3.13.5 (base) (6) Go Live
```