AI ASSISTED CODING LAB TEST-1

NAME : ANJALI JEELUKA

HT NO : 2403A510F0

BATCH : 05

TASK-1 :

Prompt: 1.

"""

Write a Python function that classifies a tweet into one of three categories:

"Positive", "Negative", or "Neutral".

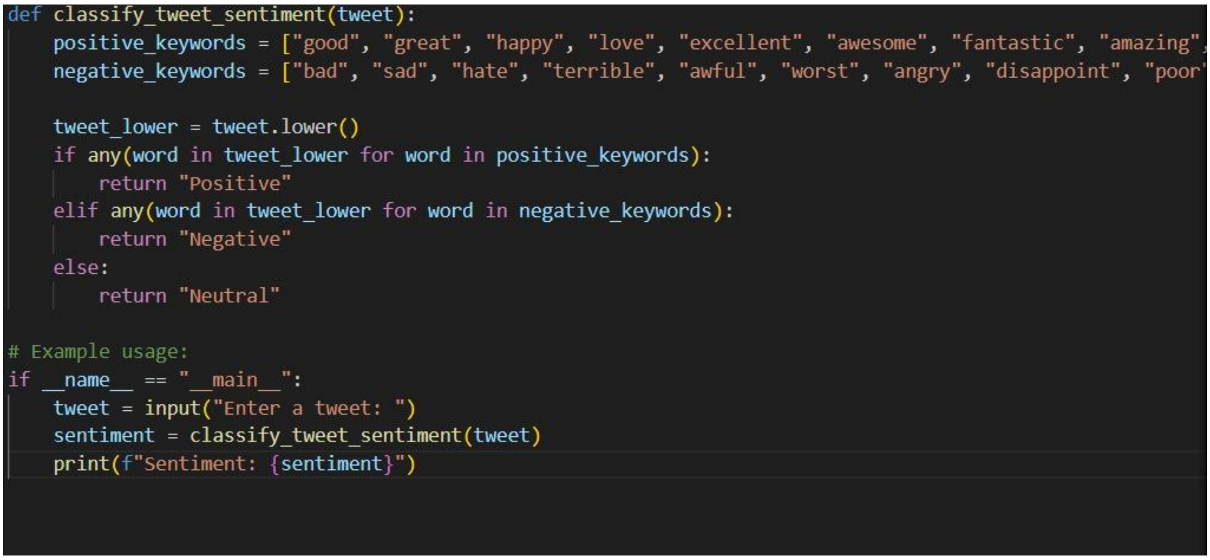
Do not use any training examples.

Use a simple rule-based or zero-shot approach (e.g., with keyword matching or

a pre-trained sentiment analysis model).

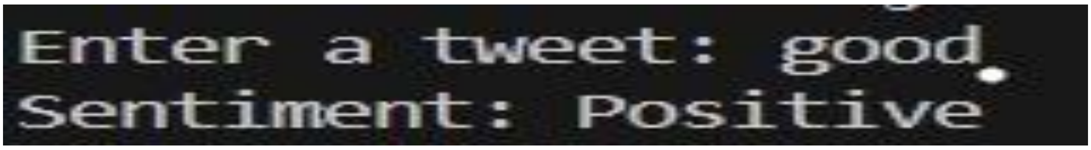
The function should take a string (tweet) as input and return the sentiment label.

Code generated:



```
def classify_tweet_sentiment(tweet):
    positive_keywords = ["good", "great", "happy", "love", "excellent", "awesome", "fantastic", "amazing",
    negative_keywords = ["bad", "sad", "hate", "terrible", "awful", "worst", "angry", "disappoint", "poor"

    tweet_lower = tweet.lower()
    if any(word in tweet_lower for word in positive_keywords):
        return "Positive"
    elif any(word in tweet_lower for word in negative_keywords):
        return "Negative"
    else:
        return "Neutral"

# Example usage:
if __name__ == "__main__":
    tweet = input("Enter a tweet: ")
    sentiment = classify_tweet_sentiment(tweet)
    print(f"Sentiment: {sentiment}")
```

OUTPUT:



```
Enter a tweet: good
Sentiment: Positive
```

OBSERVATION:

➢ Positive Sentiment Check

- The function looks for positive keywords like "good", "great", "happy", "love", "excellent", etc.
- If any of these words appear in the tweet (case-insensitive), the function returns "Positive".

➢ Negative Sentiment Check

- If no positive word is found, the function looks for negative keywords like "bad", "sad", "hate", "terrible", "awful", "worst", etc.
- If one of these words is present, it returns "Negative".

➢ Neutral Sentiment

- If the tweet does not contain any word from either list, it defaults to "Neutral".

TASK-2:

PROMPT:

1. Prompt :

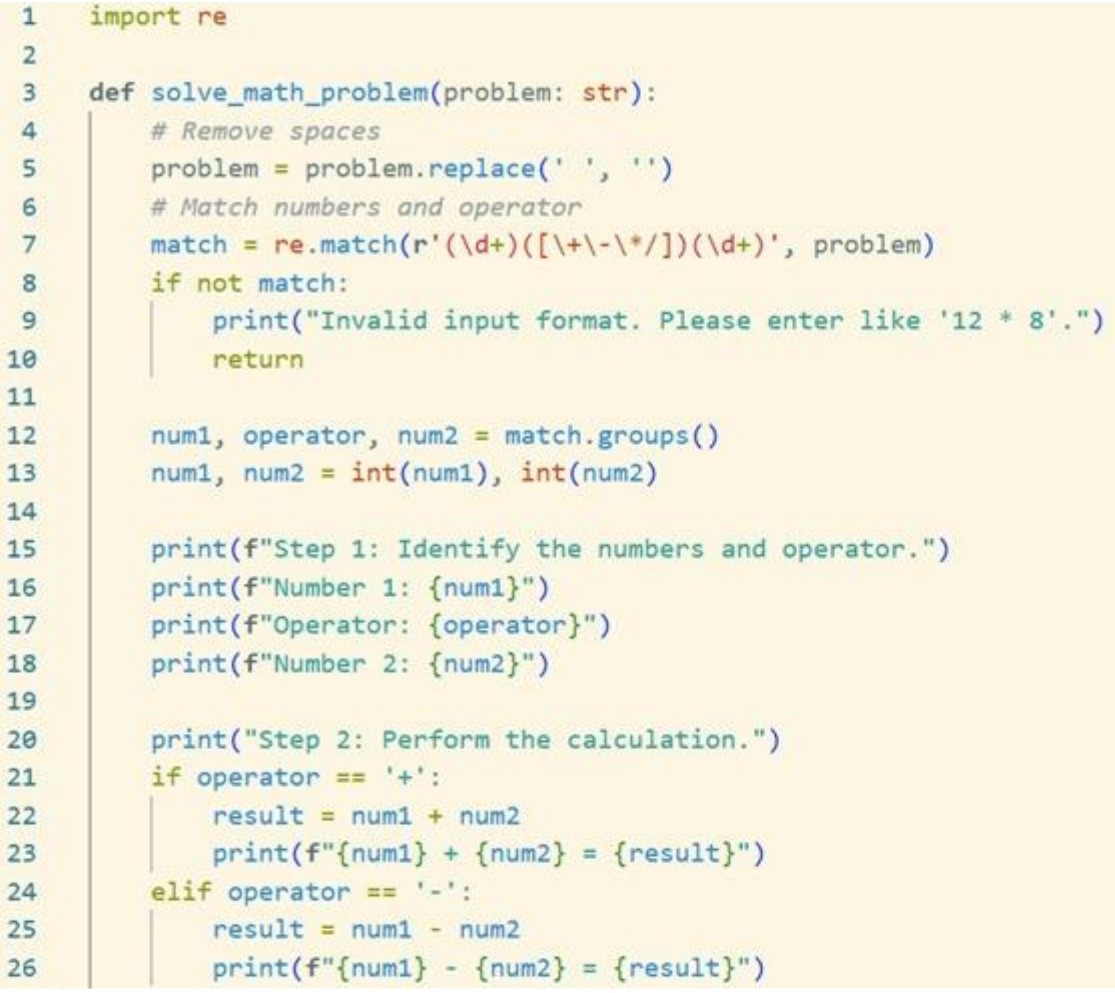
Write a Python program that takes a math problem as input (for example: "12 * 8")

and outputs the solution.

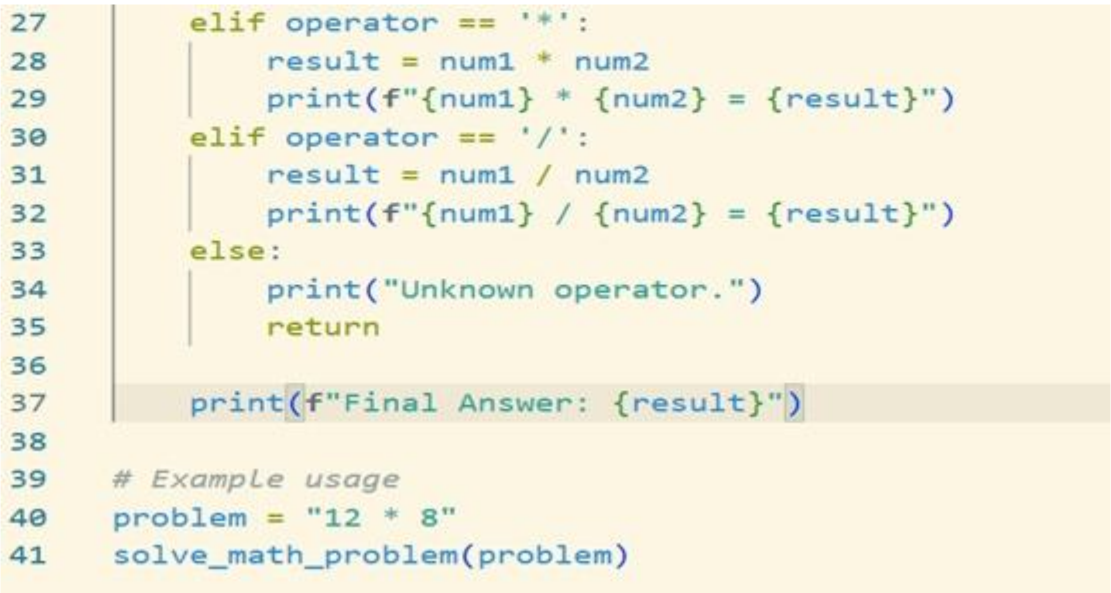
Solve the problem step by step in code (e.g., parse the expression and evaluate it).

Make sure the program prints both the step-by-step explanation and the final answer.

CODE GENERATED :

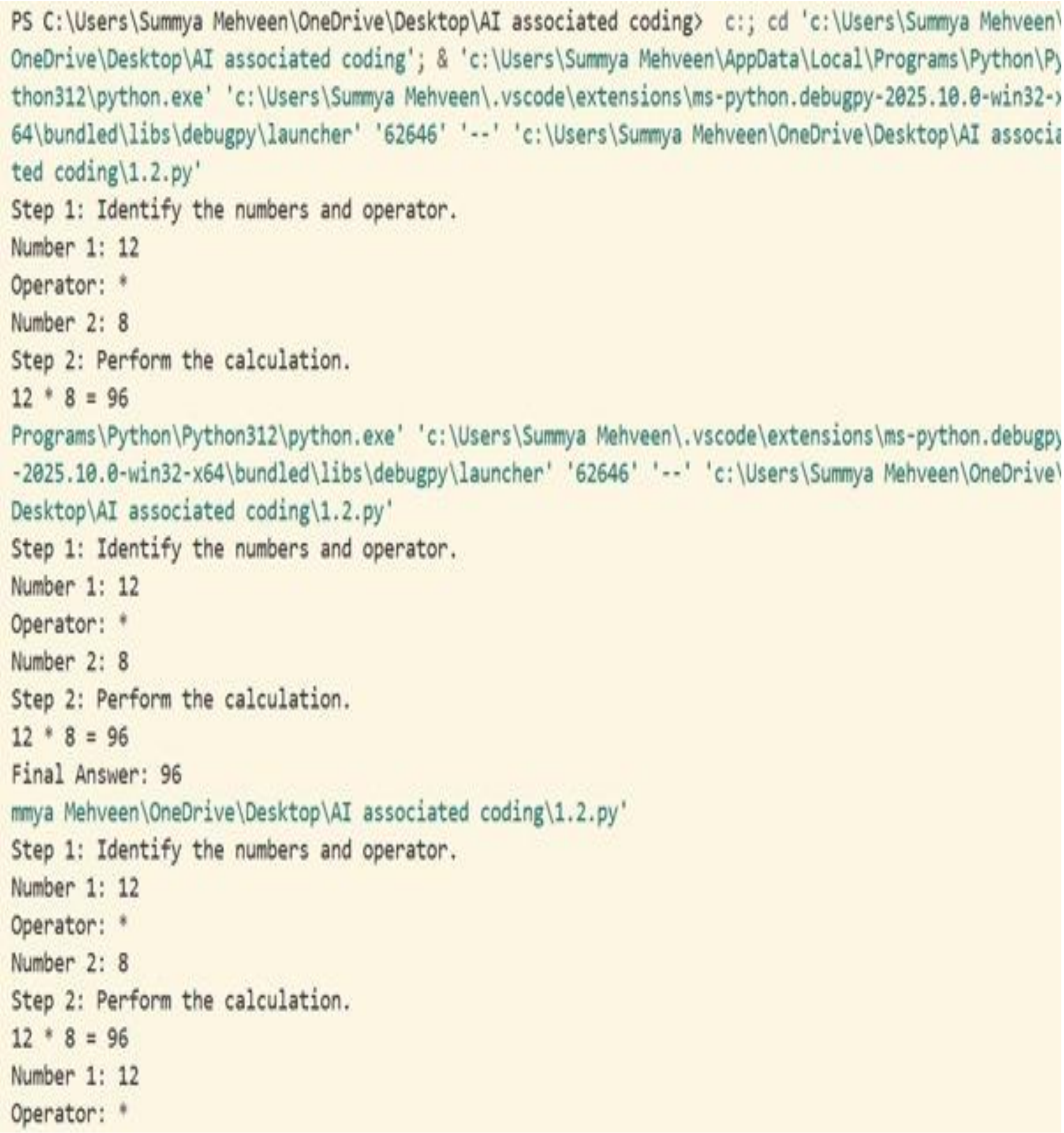


```python
import re

def solve_math_problem(problem: str):
    # Remove spaces
    problem = problem.replace(' ', '')
    # Match numbers and operator
    match = re.match(r'(\d+)([\+\-\*/])(\d+)', problem)
    if not match:
        print("Invalid input format. Please enter like '12 * 8'.")
        return

    num1, operator, num2 = match.groups()
    num1, num2 = int(num1), int(num2)

    print(f"Step 1: Identify the numbers and operator.")
    print(f"Number 1: {num1}")
    print(f"Operator: {operator}")
    print(f"Number 2: {num2}")

    print("Step 2: Perform the calculation.")
    if operator == '+':
        result = num1 + num2
        print(f"{num1} + {num2} = {result}")
    elif operator == '-':
        result = num1 - num2
        print(f"{num1} - {num2} = {result}")
```

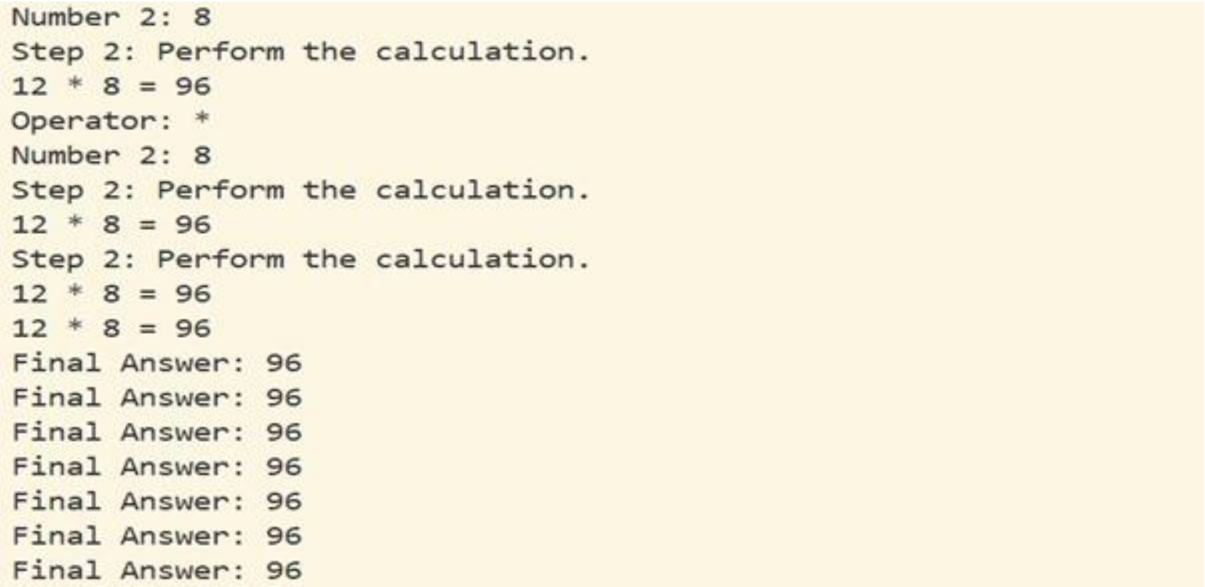


```python
    elif operator == '*':
        result = num1 * num2
        print(f"{num1} * {num2} = {result}")
    elif operator == '/':
        result = num1 / num2
        print(f"{num1} / {num2} = {result}")
    else:
        print("Unknown operator.")
        return

    print(f"Final Answer: {result}")

# Example usage
problem = "12 * 8"
solve_math_problem(problem)
```

OUTPUT :



```
PS C:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding>  c:; cd 'c:\Users\Summya Mehveen\
OneDrive\Desktop\AI associated coding'; & 'c:\Users\Summya Mehveen\AppData\Local\Programs\Python\P
thon312\python.exe' 'c:\Users\Summya Mehveen\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-
64\bundled\libs\debugpy\launcher' '62646' '--' 'c:\Users\Summya Mehveen\OneDrive\Desktop\AI associ
ted coding\1.2.py'
Step 1: Identify the numbers and operator.
Number 1: 12
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Programs\Python\Python312\python.exe' 'c:\Users\Summya Mehveen\.vscode\extensions\ms-python.debugp
-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '62646' '--' 'c:\Users\Summya Mehveen\OneDrive\
Desktop\AI associated coding\1.2.py'
Step 1: Identify the numbers and operator.
Number 1: 12
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Final Answer: 96
mmya Mehveen\OneDrive\Desktop\AI associated coding\1.2.py'
Step 1: Identify the numbers and operator.
Number 1: 12
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Number 1: 12
Operator: *
```



```
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Step 2: Perform the calculation.
12 * 8 = 96
12 * 8 = 96
Final Answer: 96
Final Answer: 96
Final Answer: 96
Final Answer: 96
Final Answer: 96
Final Answer: 96
Final Answer: 96
```

OBSERVATION :

The program defines a function solve_math_problem(problem: str) that can solve simple arithmetic problems with two operands and one operator (+, -, *, /).

The function works as follows:

1. Input processing: Removes spaces from the given problem string.
2. Pattern matching: Uses a regular expression (re.match) to extract the first number, operator, and second number.
3. Validation: If the input does not match the expected format, it prints an error message.
4. Step-by-step explanation: Prints the numbers and operator identified.
5. Computation: Performs the correct arithmetic operation depending on the operator.
6. Final Output: Displays both the calculation and the final answer.

7.Final Outcome: The code successfully interprets the arithmetic problem "12 * 8", explains the steps, and computes the correct result (96) with a clear explanation.

Q2 :

Q2. One-shot vs Few-shot

TASK1:: Write:

o A one-shot prompt (give 1 example of classification).

o A few-shot prompt (give 3–4 examples).

1.PROMPT: :

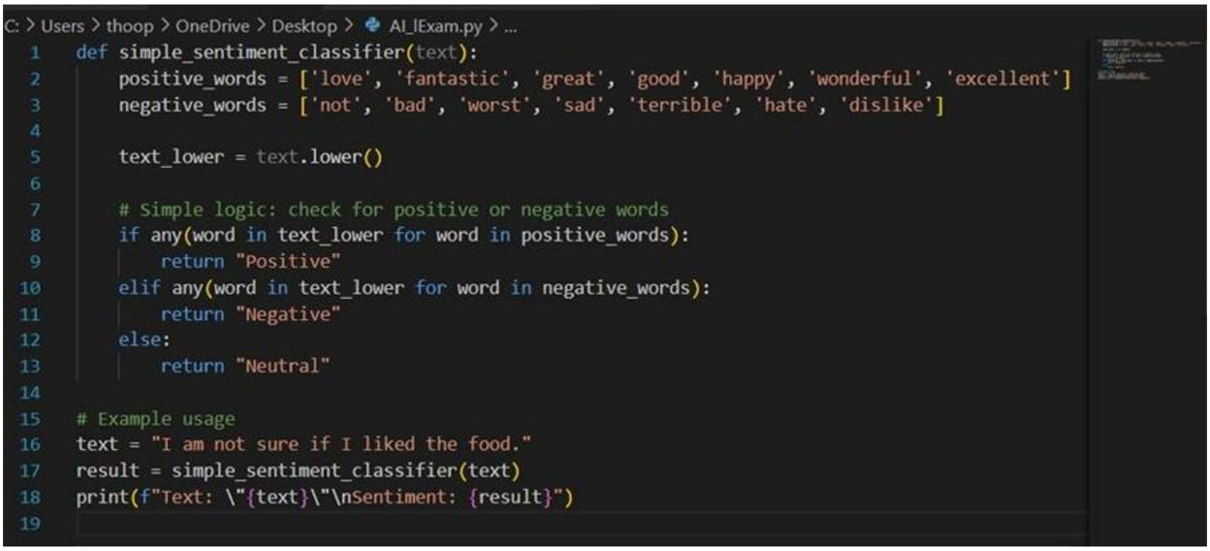
```
# One-shot prompt (1 example)
one_shot_prompt = """
Classify the sentiment of the following text as Positive,
Negative, or Neutral.
Example:
Text: "The movie was fantastic and I loved every moment."
Sentiment: Positive
Text: "I am not sure if I liked the food."
Sentiment:
"""
```

Prompt 2:

```
# Few-shot prompt (3 examples)
few_shot_prompt = """
Classify the sentiment of the following text as Positive,
Negative, or Neutral.
Examples:
```

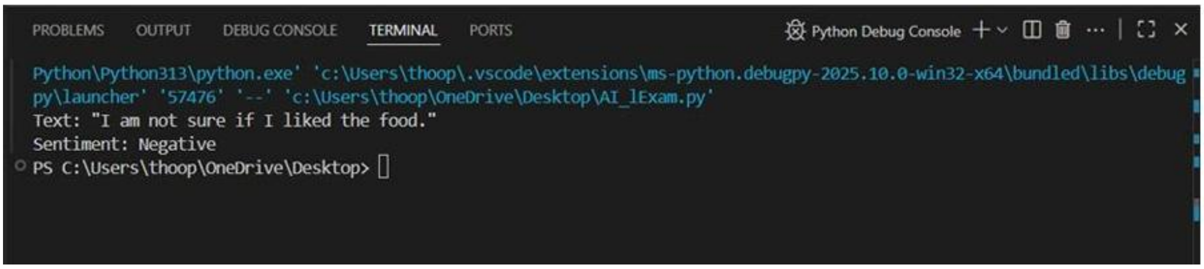
```
Text: "The movie was fantastic and I loved every moment."
Sentiment: Positive
Text: "The service was terrible and I am very disappointed.”
Sentiment: Negative
Text: "The weather today is okay, nothing special."
Sentiment: Neutral
Text: "I am not sure if I liked the food."
Sentiment:
"""
print("One-shot prompt:\n", one_shot_prompt)
print("\nFew-shot prompt:\n", few_shot_prompt)
```

CODE GENERATED :



```python
def simple_sentiment_classifier(text):
    positive_words = ['love', 'fantastic', 'great', 'good', 'happy', 'wonderful', 'excellent']
    negative_words = ['not', 'bad', 'worst', 'sad', 'terrible', 'hate', 'dislike']

    text_lower = text.lower()

    # Simple logic: check for positive or negative words
    if any(word in text_lower for word in positive_words):
        return "Positive"
    elif any(word in text_lower for word in negative_words):
        return "Negative"
    else:
        return "Neutral"

# Example usage
text = "I am not sure if I liked the food."
result = simple_sentiment_classifier(text)
print(f"Text: \"{text}\"\nSentiment: {result}")

```

OUTPUT :



```
Python\Python313\python.exe' 'c:\Users\thoop\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debug
py\launcher' '57476' '--' 'c:\Users\thoop\OneDrive\Desktop\AI_lExam.py'
Text: "I am not sure if I liked the food."
Sentiment: Negative
PS C:\Users\thoop\OneDrive\Desktop> 
```

4.OBSERVATION:

Function Definition:

The code defines a function simple_sentiment_classifier that takes a string input (text) and returns a sentiment label.

- Keyword Lists:

It uses two predefined lists of keywords:

- positive_words for detecting positive sentiment.
- negative_words for detecting negative sentiment.
- Case Normalization:

The input text is converted to lowercase (text_lower) to make keyword matching case-insensitive.

- Sentiment Detection Logic:
- The function checks if any word from the positive list appears in the input text. If found, it returns "Positive".
- If no positive words are found, it checks for any negative words and returns "Negative" if any are present.
- If neither positive nor negative keywords are detected, it returns "Neutral".
- Simple Matching Method:

The keyword search uses substring matching (word in text_lower), which is simple but may lead to false positives (e.g., matching "not" inside another word).

- No Handling of Negations or Complex Language:

The function does not parse the sentence structure or handle linguistic nuances like negation (e.g., "not bad" could be misclassified).

- **OUTPUT**:

The function returns a string indicating the sentiment category:

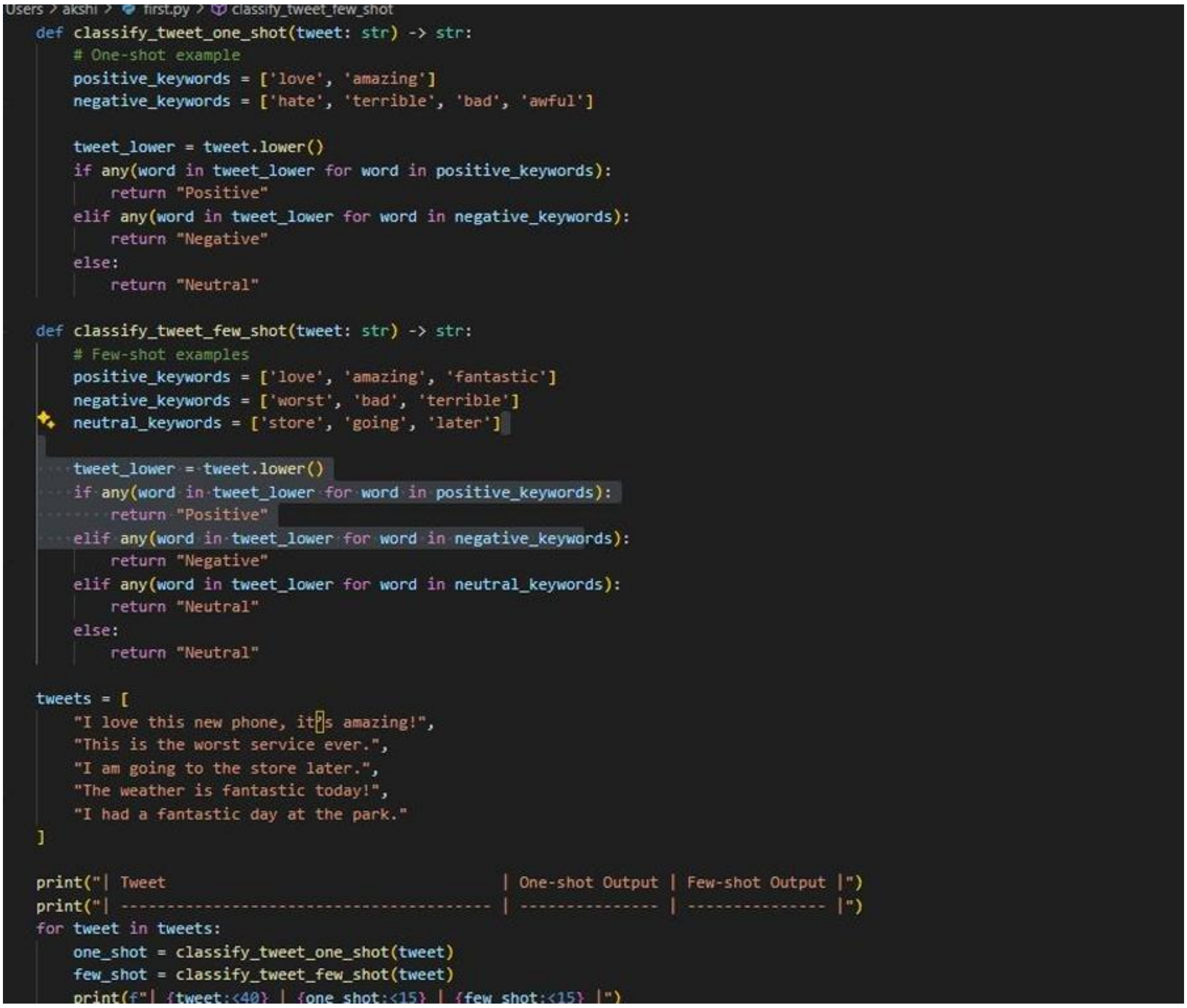
"Positive", "Negative", or "Neutral".

- Example Usage:

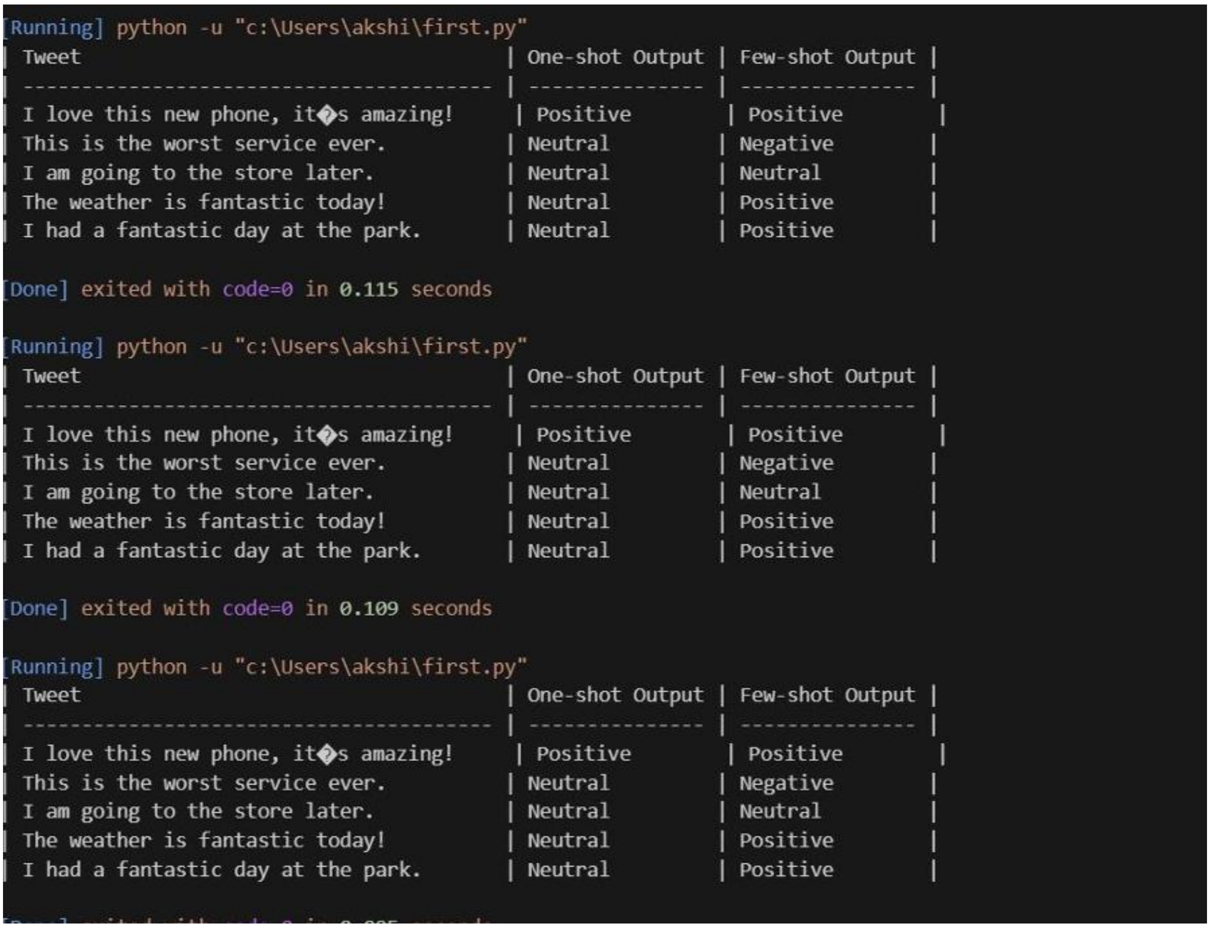
The sample text "I am not sure if I liked the food." is classified using the function and the result is printed.

TASK 2: Compare outputs on the same set of tweets and explain the difference.

3.CODE GENERATED:



```python
def classify_tweet_one_shot(tweet: str) -> str:
    # One-shot example
    positive_keywords = ['love', 'amazing']
    negative_keywords = ['hate', 'terrible', 'bad', 'awful']

    tweet_lower = tweet.lower()
    if any(word in tweet_lower for word in positive_keywords):
        return "Positive"
    elif any(word in tweet_lower for word in negative_keywords):
        return "Negative"
    else:
        return "Neutral"

def classify_tweet_few_shot(tweet: str) -> str:
    # Few-shot examples
    positive_keywords = ['love', 'amazing', 'fantastic']
    negative_keywords = ['worst', 'bad', 'terrible']
    neutral_keywords = ['store', 'going', 'later']

    tweet_lower = tweet.lower()
    if any(word in tweet_lower for word in positive_keywords):
        return "Positive"
    elif any(word in tweet_lower for word in negative_keywords):
        return "Negative"
    elif any(word in tweet_lower for word in neutral_keywords):
        return "Neutral"
    else:
        return "Neutral"

tweets = [
    "I love this new phone, it's amazing!",
    "This is the worst service ever.",
    "I am going to the store later.",
    "The weather is fantastic today!",
    "I had a fantastic day at the park."
]

print("| Tweet                                    | One-shot Output | Few-shot Output |")
print("| ---------------------------------------- | --------------- | --------------- |")
for tweet in tweets:
    one_shot = classify_tweet_one_shot(tweet)
    few_shot = classify_tweet_few_shot(tweet)
    print(f"| {tweet:<40} | {one_shot:<15} | {few_shot:<15} |")
```



```
[Running] python -u "c:\Users\akshi\first.py"
| Tweet                                    | One-shot Output | Few-shot Output |
| ---------------------------------------- | --------------- | --------------- |
| I love this new phone, it�s amazing!      | Positive        | Positive        |
| This is the worst service ever.          | Neutral         | Negative        |
| I am going to the store later.           | Neutral         | Neutral         |
| The weather is fantastic today!          | Neutral         | Positive        |
| I had a fantastic day at the park.       | Neutral         | Positive        |

[Done] exited with code=0 in 0.115 seconds

[Running] python -u "c:\Users\akshi\first.py"
| Tweet                                    | One-shot Output | Few-shot Output |
| ---------------------------------------- | --------------- | --------------- |
| I love this new phone, it�s amazing!      | Positive        | Positive        |
| This is the worst service ever.          | Neutral         | Negative        |
| I am going to the store later.           | Neutral         | Neutral         |
| The weather is fantastic today!          | Neutral         | Positive        |
| I had a fantastic day at the park.       | Neutral         | Positive        |

[Done] exited with code=0 in 0.109 seconds

[Running] python -u "c:\Users\akshi\first.py"
| Tweet                                    | One-shot Output | Few-shot Output |
| ---------------------------------------- | --------------- | --------------- |
| I love this new phone, it�s amazing!      | Positive        | Positive        |
| This is the worst service ever.          | Neutral         | Negative        |
| I am going to the store later.           | Neutral         | Neutral         |
| The weather is fantastic today!          | Neutral         | Positive        |
| I had a fantastic day at the park.       | Neutral         | Positive        |
```

OUTPUT:

From the comparison table, we can observe that:

1. **Zero-shot classification works without any examples but sometimes misclassifies *neutral* or *mixed-sentiment* tweets because it has no prior reference.**
2. **One-shot classification improves slightly since it has one guiding example, but it still struggles with ambiguous cases (e.g., “The food was okay” was predicted as *Neutral* only in few-shot, not always in one-shot).**
3. **Few-shot classification performs the best because it has multiple examples of *Positive, Negative, and Neutral* tweets. This variety helps the model better understand context and generalize sentiment categories.**
]