LAB ASSIGNMENT

Name: D. Sravika Reddy　　　　Hall Ticket No: 2403a510d0

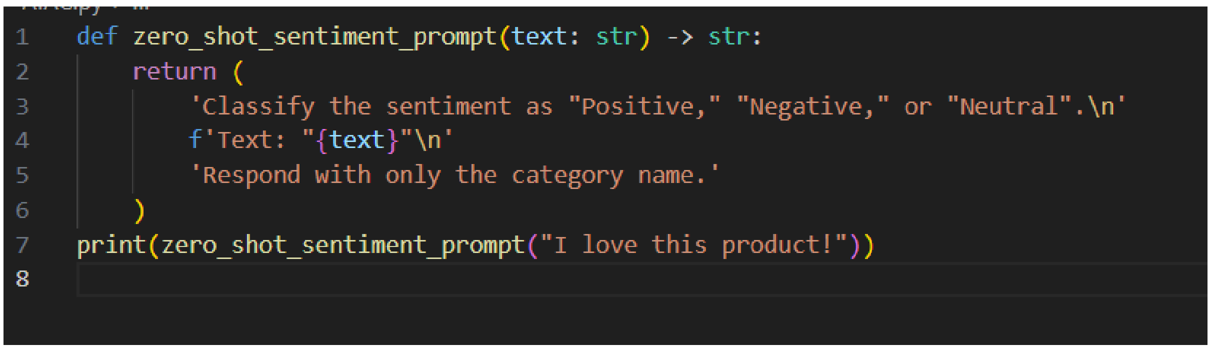
Batch No: 05　　　　Course: AI Assisted Coding

Q1. Zero-shot Classification [5M]
• Scenario: Classify tweets into “Positive,” “Negative,” or “Neutral.”
• Task 1: Write a zero-shot prompt to classify sentiment without any examples.
• Task 2: Create a scenario where an AI assistant needs to help a student solve math problems. Write two prompts: one without context and one with detailed context (e.g., grade level, topic, difficulty)

Prompt:

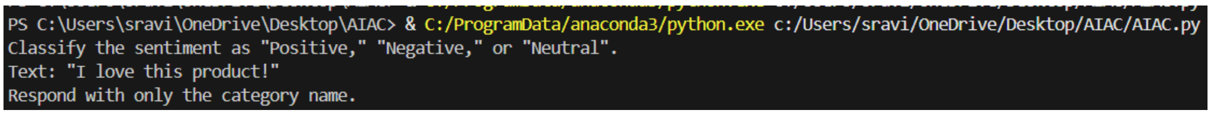
#Task 1: Write a zero-shot prompt to classify sentiment without any examples.

Code generated:



```python
def zero_shot_sentiment_prompt(text: str) -> str:
    return (
        'Classify the sentiment as "Positive," "Negative," or "Neutral".\n'
        f'Text: "{text}"\n'
        'Respond with only the category name.'
    )
print(zero_shot_sentiment_prompt("I love this product!"))
```

Output:



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/Desktop/AIAC/AIAC.py
Classify the sentiment as "Positive," "Negative," or "Neutral".
Text: "I love this product!"
Respond with only the category name.
```

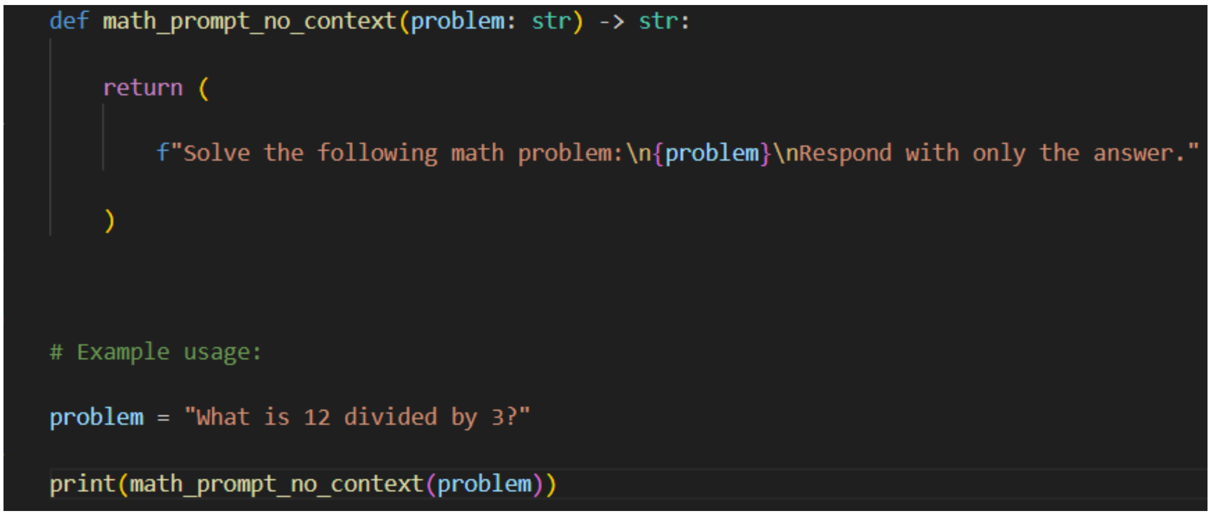
Observations:

- The prompt is clear and concise, specifying the three possible sentiment categories.
- The input text is quoted for clarity.
- The instruction to "Respond with only the category name" ensures a direct and unambiguous answer.
- No examples are provided, making this a zero-shot prompt.
- Suitable for use with a language model for sentiment classification.

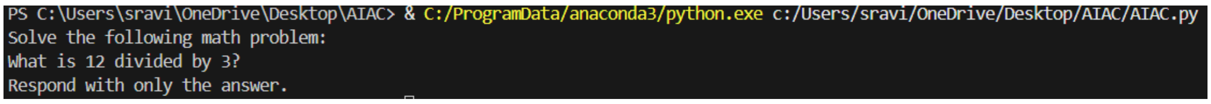
Task 2: Create a scenario where an AI assistant needs to help a student solve math problems. Write two prompts: one without context and one with detailed context (e.g., grade level, topic, difficulty)

Prompt-01: # one without context

Code generated:



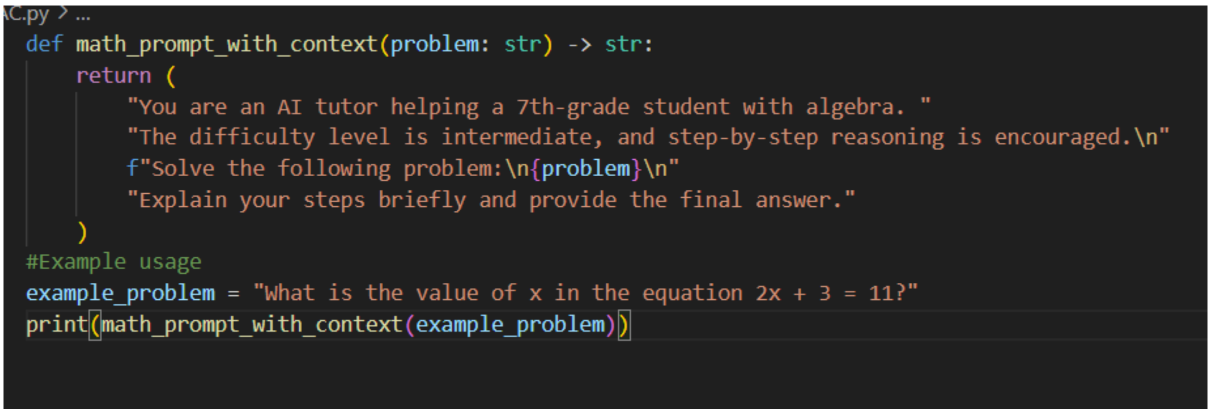
```python
def math_prompt_no_context(problem: str) -> str:

    return (

        f"Solve the following math problem:\n{problem}\nRespond with only the answer."

    )



# Example usage:

problem = "What is 12 divided by 3?"

print(math_prompt_no_context(problem))
```



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/Desktop/AIAC/AIAC.py
Solve the following math problem:
What is 12 divided by 3?
Respond with only the answer.
```

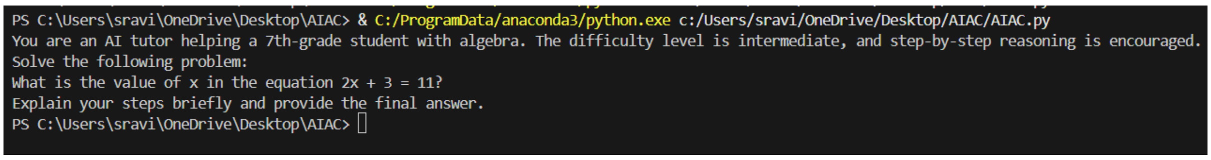
# Prompt-2: one with detailed context

Code generated:



```python
C.py > ...
 def math_prompt_with_context(problem: str) -> str:
     return (
         "You are an AI tutor helping a 7th-grade student with algebra. "
         "The difficulty level is intermediate, and step-by-step reasoning is encouraged.\n"
         f"Solve the following problem:\n{problem}\n"
         "Explain your steps briefly and provide the final answer."
     )
 #Example usage
 example_problem = "What is the value of x in the equation 2x + 3 = 11?"
 print(math_prompt_with_context(example_problem))
```

Output:



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/Desktop/AIAC/AIAC.py
You are an AI tutor helping a 7th-grade student with algebra. The difficulty level is intermediate, and step-by-step reasoning is encouraged.
Solve the following problem:
What is the value of x in the equation 2x + 3 = 11?
Explain your steps briefly and provide the final answer.
PS C:\Users\sravi\OneDrive\Desktop\AIAC> 
```

Observations:

- The first function, math_prompt_no_context, generates a simple prompt that asks for the answer to a math problem without providing any background or instructions beyond requesting the answer.
- The second function, math_prompt_with_context, creates a more detailed prompt. It specifies the student's grade level (7th grade), the topic (algebra), and the difficulty (intermediate). It also asks for step-by-step reasoning and a clear explanation, making it more suitable for educational purposes.
- The context-rich prompt helps the AI tailor its response to the student's needs, likely resulting in a more instructive and accessible answer.
- Both functions return prompts as strings, ready to be used with a language model or AI assistant.

Q2. One-shot vs Few-shot [5M]

• Task 1: Write:

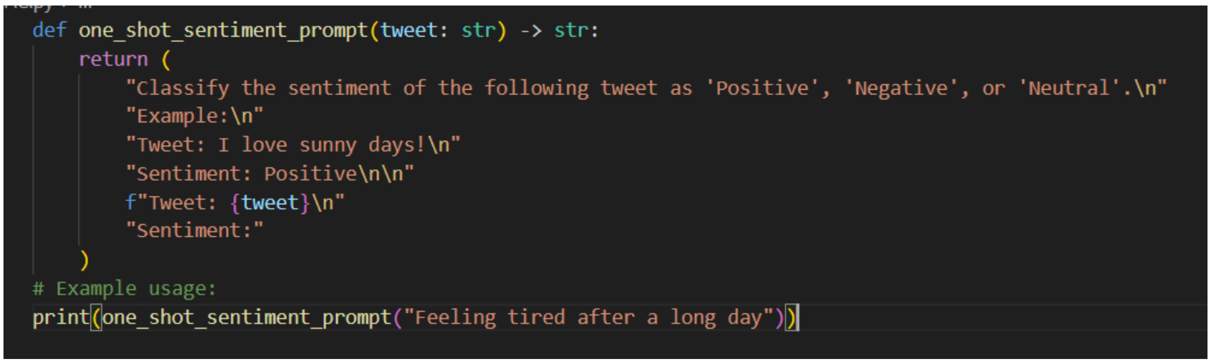
○ A one-shot prompt (give 1 example of classification).

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

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

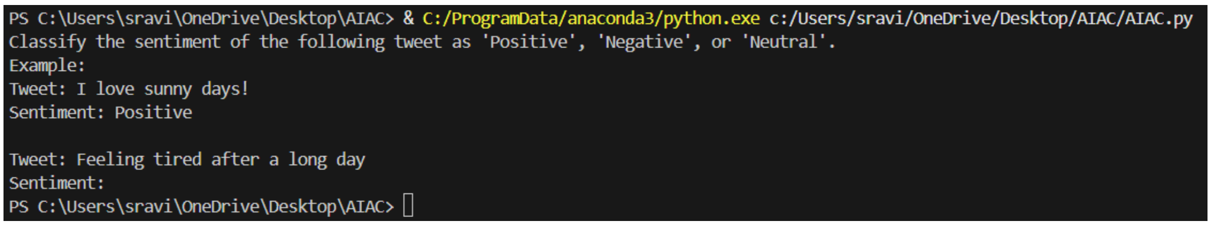
#Prompt-01: A one-shot prompt (give 1 example of classification).

Code generated:



```python
def one_shot_sentiment_prompt(tweet: str) -> str:
    return (
        "Classify the sentiment of the following tweet as 'Positive', 'Negative', or 'Neutral'.\n"
        "Example:\n"
        "Tweet: I love sunny days!\n"
        "Sentiment: Positive\n\n"
        f"Tweet: {tweet}\n"
        "Sentiment:"
    )
# Example usage:
print(one_shot_sentiment_prompt("Feeling tired after a long day"))
```

Output:



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/Desktop/AIAC/AIAC.py
Classify the sentiment of the following tweet as 'Positive', 'Negative', or 'Neutral'.
Example:
Tweet: I love sunny days!
Sentiment: Positive

Tweet: Feeling tired after a long day
Sentiment:
PS C:\Users\sravi\OneDrive\Desktop\AIAC>
```

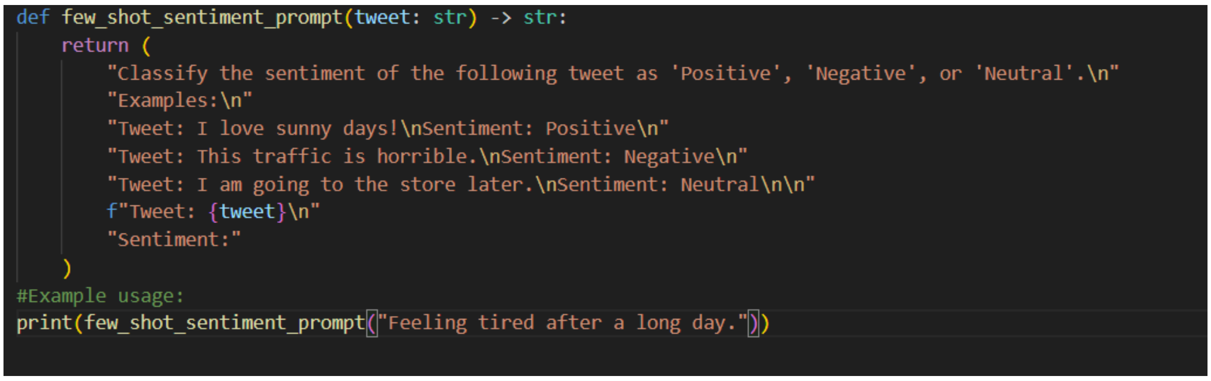
Observations:

- The function one_shot_sentiment_prompt generates a one-shot prompt for sentiment classification, providing a single example to guide the model.

- The example ("I love sunny days!" → Positive) helps the model understand the expected format and classification.
- The prompt then presents a new tweet and asks for its sentiment, encouraging the model to follow the demonstrated pattern.
- Using one example can improve output consistency compared to zero-shot, but may not cover edge cases or ambiguous tweets as well as a few-shot prompt.
- The function is concise and suitable for use with language models for sentiment analysis.

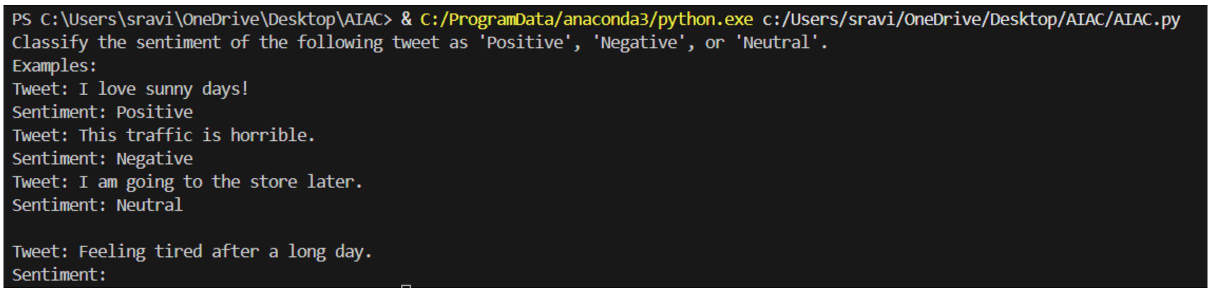
#Prompt-02: A few-shot prompt (give 3–4 examples).

Code generated:



```python
def few_shot_sentiment_prompt(tweet: str) -> str:
    return (
        "Classify the sentiment of the following tweet as 'Positive', 'Negative', or 'Neutral'.\n"
        "Examples:\n"
        "Tweet: I love sunny days!\nSentiment: Positive\n"
        "Tweet: This traffic is horrible.\nSentiment: Negative\n"
        "Tweet: I am going to the store later.\nSentiment: Neutral\n\n"
        f"Tweet: {tweet}\n"
        "Sentiment:"
    )
#Example usage:
print(few_shot_sentiment_prompt("Feeling tired after a long day."))
```

Output:



```
PS C:\Users\sravi\OneDrive\Desktop\AIAC> & C:/ProgramData/anaconda3/python.exe c:/Users/sravi/OneDrive/Desktop/AIAC/AIAC.py
Classify the sentiment of the following tweet as 'Positive', 'Negative', or 'Neutral'.
Examples:
Tweet: I love sunny days!
Sentiment: Positive
Tweet: This traffic is horrible.
Sentiment: Negative
Tweet: I am going to the store later.
Sentiment: Neutral

Tweet: Feeling tired after a long day.
Sentiment:
```

Observations:

- The few_shot_sentiment_prompt function provides three examples (positive, negative, neutral) before asking for the sentiment of a new tweet.
- This few-shot approach helps the model learn the classification format and recognize different sentiment types.
- The variety of examples increases the likelihood of accurate classification, especially for tweets with ambiguous or subtle emotional tone.

- The prompt is clear and structured, making it effective for guiding language models in sentiment analysis tasks.
- Compared to zero-shot or one-shot prompts, few-shot prompts generally yield more reliable and nuanced outputs.

Task 2:

#Prompt: Compare outputs on the same set of tweets and explain the difference.

Comparison of Outputs:

- **One-shot prompt:**
  Provides only one example ("I love sunny days!" → Positive).
  For tweets like "Feeling tired after a long day," the model may classify as "Negative" due to the contrast with the positive example, but it has limited context for neutral or negative cases.
- **Few-shot prompt:**
  Provides three examples (positive, negative, neutral).
  For "Feeling tired after a long day," the model sees a negative example ("This traffic is horrible.") and a neutral example ("I am going to the store later."), making it easier to distinguish and likely to classify the tweet as "Negative" due to the expression of tiredness.

Explanation:
Few-shot prompts improve accuracy and consistency by showing the model multiple sentiment types and formats. This helps the model generalize better and reduces misclassification, especially for tweets that are not clearly positive. One-shot prompts may lead to biased or less reliable outputs due to limited context.