

2303a51602

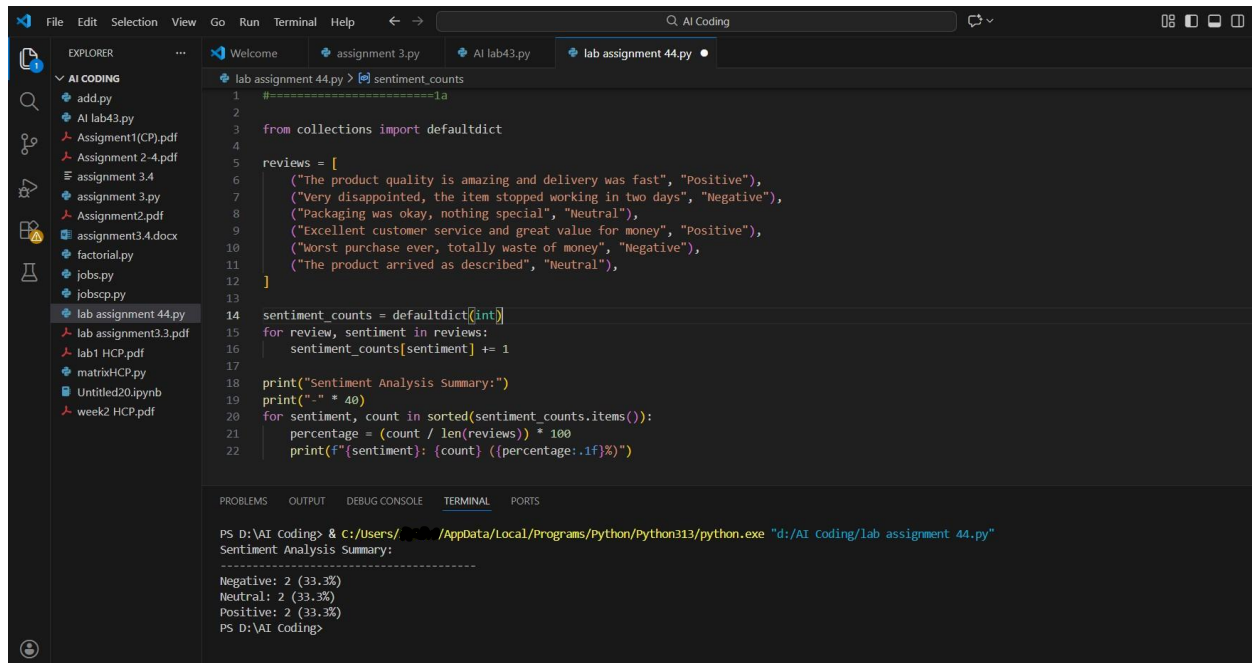
Lab assignment-4.4

1. Sentiment Classification for Customer Reviews Scenario:

An e-commerce platform wants to analyze customer reviews and classify them into Positive, Negative, or Neutral sentiments using prompt engineering

Tasks:

a) Prepare 6 short customer reviews mapped to sentiment labels.



The screenshot shows a VS Code editor with a Python file named 'lab assignment 44.py'. The script defines a list of six reviews with their corresponding sentiment labels and uses a defaultdict to count the occurrences of each sentiment. The terminal output shows the results of the script execution.

```
1 #=====1a
2
3 from collections import defaultdict
4
5 reviews = [
6     ("The product quality is amazing and delivery was fast", "Positive"),
7     ("Very disappointed, the item stopped working in two days", "Negative"),
8     ("Packaging was okay, nothing special", "Neutral"),
9     ("Excellent customer service and great value for money", "Positive"),
10    ("Worst purchase ever, totally waste of money", "Negative"),
11    ("The product arrived as described", "Neutral"),
12 ]
13
14 sentiment_counts = defaultdict(int)
15 for review, sentiment in reviews:
16     sentiment_counts[sentiment] += 1
17
18 print("Sentiment Analysis Summary:")
19 print("-" * 40)
20 for sentiment, count in sorted(sentiment_counts.items()):
21     percentage = (count / len(reviews)) * 100
22     print(f"{sentiment}: {count} ({percentage:.1f}%)")
```

Terminal Output:

```
PS D:\AI Coding> & C:/Users/.../AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Sentiment Analysis Summary:
-----
Negative: 2 (33.3%)
Neutral: 2 (33.3%)
Positive: 2 (33.3%)
PS D:\AI Coding>
```

b) Design a Zero-shot prompt to classify sentiment.

The screenshot shows a Visual Studio Code editor with a file explorer on the left containing various files like 'add.py', 'AI lab43.py', and 'lab assignment 44.py'. The main editor window displays a Python script for sentiment analysis. The script defines positive and negative word lists, processes a review string, and prints the sentiment result. The terminal at the bottom shows the command to run the script and the resulting output, which indicates a 'Positive' sentiment for the given review.

```
1 #
2
3 #####1#####
4 review = "The product quality is amazing and delivery was fast"
5
6 # Simple sentiment classification based on keywords
7 positive_words = ["amazing", "great", "excellent", "good", "fast", "love", "best"]
8 negative_words = ["bad", "terrible", "poor", "slow", "worst", "hate", "awful"]
9
10 review_lower = review.lower()
11 pos_count = sum(1 for word in positive_words if word in review_lower)
12 neg_count = sum(1 for word in negative_words if word in review_lower)
13
14 if pos_count > neg_count:
15     sentiment = "Positive"
16 elif neg_count > pos_count:
17     sentiment = "Negative"
18 else:
19     sentiment = "Neutral"
20
21 print(f"Review: \'{review}\'")
22 print(f"Sentiment: {sentiment}")
```

PS D:\AI Coding> & C:/Users/.../AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"

Sentiment Analysis Summary:

Negative: 2 (33.3%)
Neutral: 2 (33.3%)
Positive: 2 (33.3%)

PS D:\AI Coding> & C:/Users/.../AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"

Review: "The product quality is amazing and delivery was fast"
Sentiment: Positive

PS D:\AI Coding>

1c) Design a One-shot prompt with one labeled example.

The screenshot shows a Visual Studio Code editor with a file explorer on the left. The main editor window displays a Python script for sentiment analysis. The script defines positive and negative word lists, processes a review string, and prints the sentiment result. The terminal at the bottom shows the command to run the script and the resulting output, which indicates a 'Positive' sentiment for the given review.

```
4 #####1#####
5 # Sentiment Classification
6 review = "Excellent customer service and great value for money"
7
8 # Simple sentiment classification
9 positive_words = ['excellent', 'great', 'good', 'amazing', 'wonderful', 'best', 'love']
10 negative_words = ['hate', 'bad', 'poor', 'awful', 'terrible', 'worst', 'useless']
11
12 review_lower = review.lower()
13 positive_count = sum(1 for word in positive_words if word in review_lower)
14 negative_count = sum(1 for word in negative_words if word in review_lower)
15
16 if positive_count > negative_count:
17     sentiment = "Positive"
18 elif negative_count > positive_count:
19     sentiment = "Negative"
20 else:
21     sentiment = "Neutral"
22
23 print(f"Review: \'{review}\'")
24 print(f"Sentiment: {sentiment}")
```

Sentiment Analysis Summary:

Negative: 2 (33.3%)
Neutral: 2 (33.3%)
Positive: 2 (33.3%)

PS D:\AI Coding> & C:/Users/.../AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"

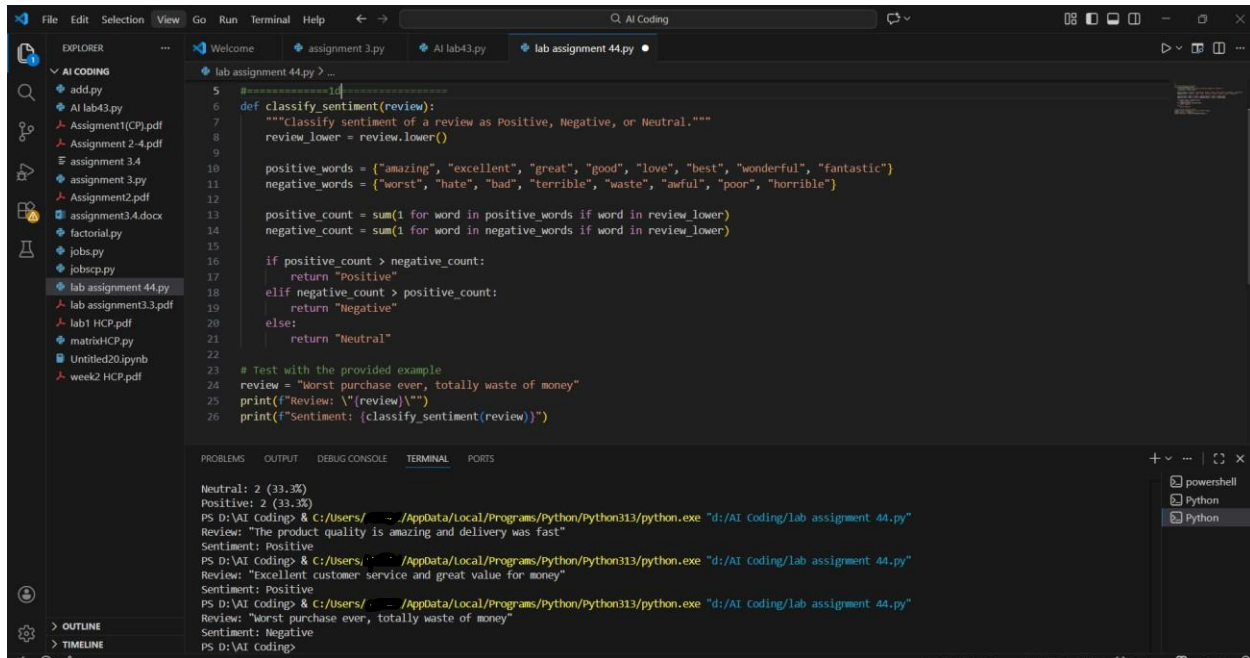
Review: "The product quality is amazing and delivery was fast"
Sentiment: Positive

PS D:\AI Coding> & C:/Users/.../AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"

Review: "Excellent customer service and great value for money"
Sentiment: Positive

PS D:\AI Coding>

1d) Design a Few-shot prompt with 3–5 labeled examples.

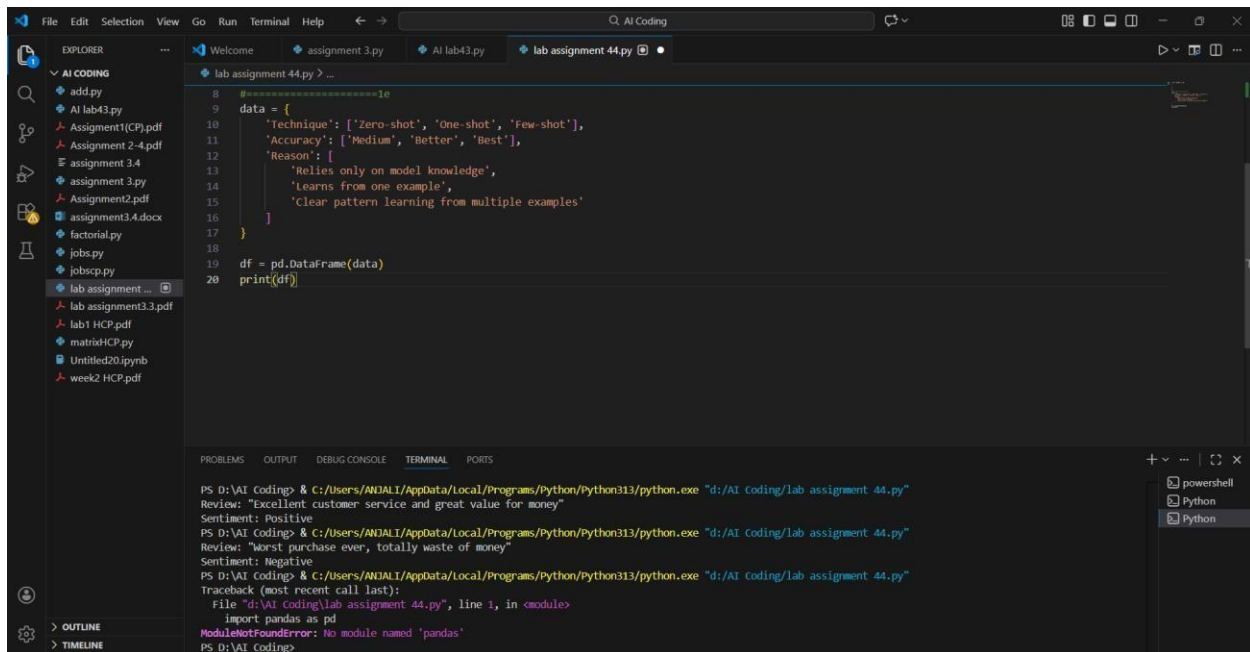


```
5 #=====1e=====
6 def classify_sentiment(review):
7     """classify sentiment of a review as Positive, Negative, or Neutral."""
8     review_lower = review.lower()
9
10    positive_words = ("amazing", "excellent", "great", "good", "love", "best", "wonderful", "fantastic")
11    negative_words = ("worst", "hate", "bad", "terrible", "waste", "awful", "poor", "horrible")
12
13    positive_count = sum(1 for word in positive_words if word in review_lower)
14    negative_count = sum(1 for word in negative_words if word in review_lower)
15
16    if positive_count > negative_count:
17        return "Positive"
18    elif negative_count > positive_count:
19        return "Negative"
20    else:
21        return "Neutral"
22
23 # Test with the provided example
24 review = "Worst purchase ever, totally waste of money"
25 print(f"Review: \"{review}\"")
26 print(f"Sentiment: {classify_sentiment(review)}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Neutral: 2 (33.3%)
Positive: 2 (33.3%)
PS D:\AI Coding\ & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Review: "The product quality is amazing and delivery was fast"
Sentiment: Positive
PS D:\AI Coding\ & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Review: "Excellent customer service and great value for money"
Sentiment: Positive
PS D:\AI Coding\ & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Review: "Worst purchase ever, totally waste of money"
Sentiment: Negative
PS D:\AI Coding\

1e) Compare the outputs and discuss accuracy differences.



```
8 #=====1e=====
9 data = {
10     'Technique': ['Zero-shot', 'One-shot', 'Few-shot'],
11     'Accuracy': ['Medium', 'Better', 'Best'],
12     'Reason': [
13         'Relies only on model knowledge',
14         'Learns from one example',
15         'Clear pattern learning from multiple examples'
16     ]
17 }
18
19 df = pd.DataFrame(data)
20 print(df)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

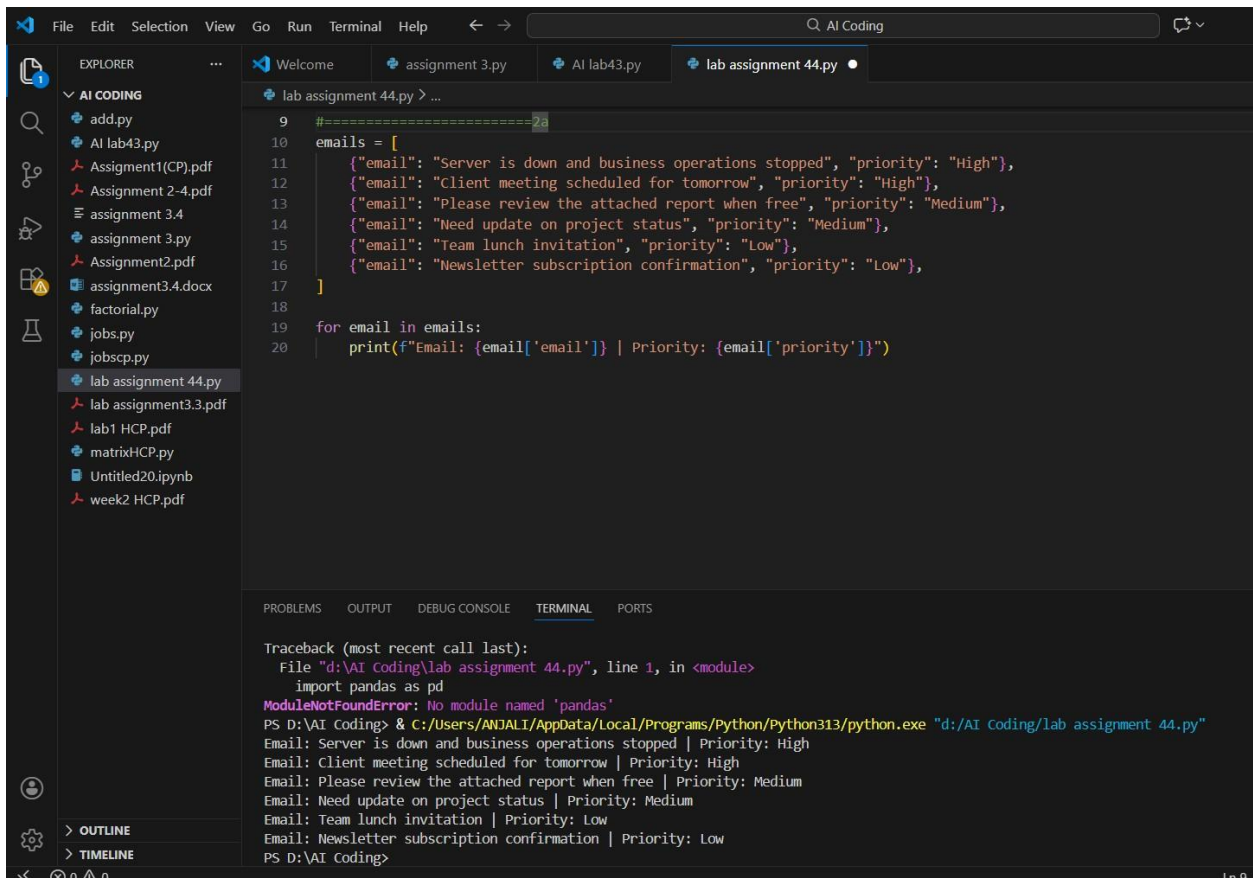
PS D:\AI Coding\ & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Review: "Excellent customer service and great value for money"
Sentiment: Positive
PS D:\AI Coding\ & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Review: "Worst purchase ever, totally waste of money"
Sentiment: Negative
PS D:\AI Coding\ & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
 File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
 import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding\

2. Email Priority Classification

Scenario:

A company wants to automatically prioritize incoming emails into High Priority, Medium Priority, or Low Priority

2a) Create 6 sample email messages with priority labels.



2b) Perform intent classification using Zero-shot prompting

```
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
  import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: Server is down and business operations stopped | Priority: High
Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding>
```

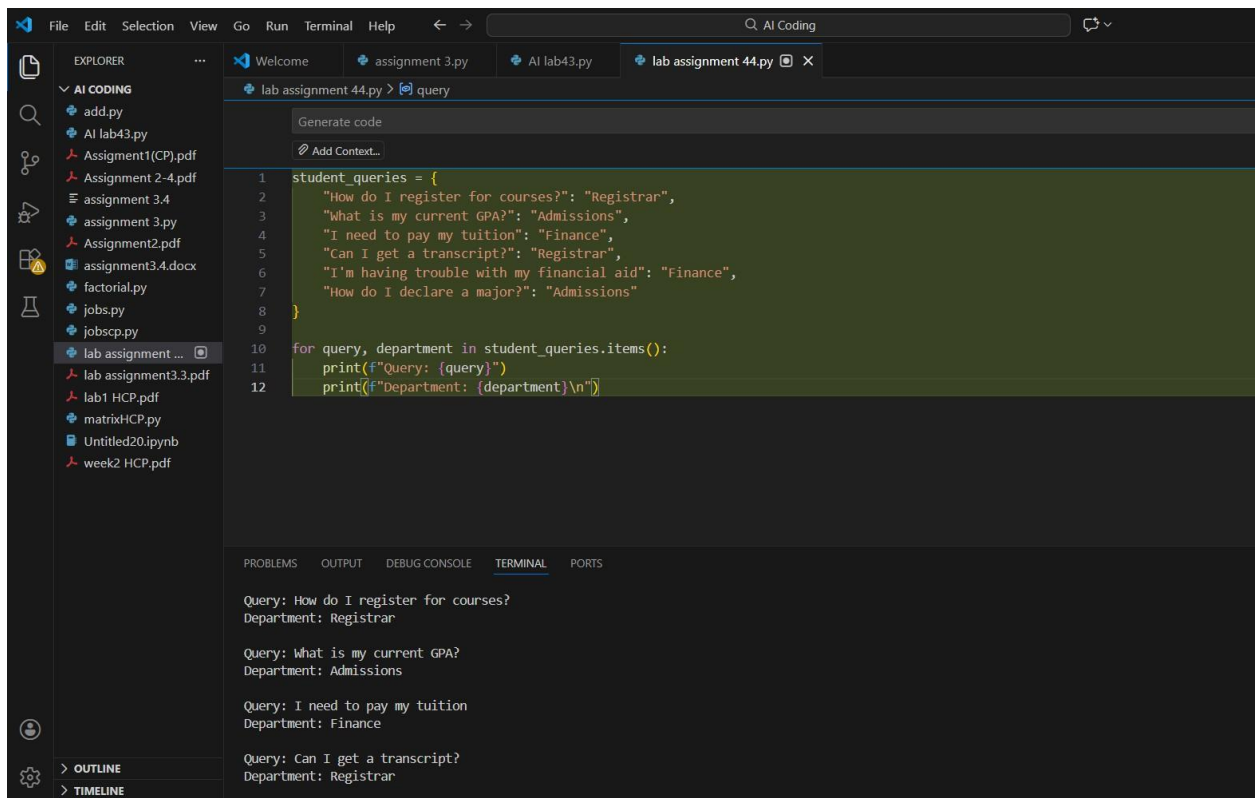
2c) Perform classification using One-shot prompting

```
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: Server is down and business operations stopped | Priority: High
Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding>
```

2d) Perform classification using Few-shot prompting.

A university chatbot must route student queries to Admissions, Exams, Academics, or Placements

3a) . Create 6 sample student queries mapped to departments.



The screenshot shows a Visual Studio Code editor with a Python script named `lab assignment 44.py` open. The script defines a dictionary `student_queries` mapping specific queries to departments. The queries are: "How do I register for courses?", "What is my current GPA?", "I need to pay my tuition", "Can I get a transcript?", "I'm having trouble with my financial aid", and "How do I declare a major?". The script then iterates over these queries and prints the query and its corresponding department.

```
1 student_queries = {
2     "How do I register for courses?": "Registrar",
3     "What is my current GPA?": "Admissions",
4     "I need to pay my tuition": "Finance",
5     "Can I get a transcript?": "Registrar",
6     "I'm having trouble with my financial aid": "Finance",
7     "How do I declare a major?": "Admissions"
8 }
9
10 for query, department in student_queries.items():
11     print(f"Query: {query}")
12     print(f"Department: {department}\n")
```

The terminal output shows the execution of the script, displaying the query and department for each item in the dictionary:

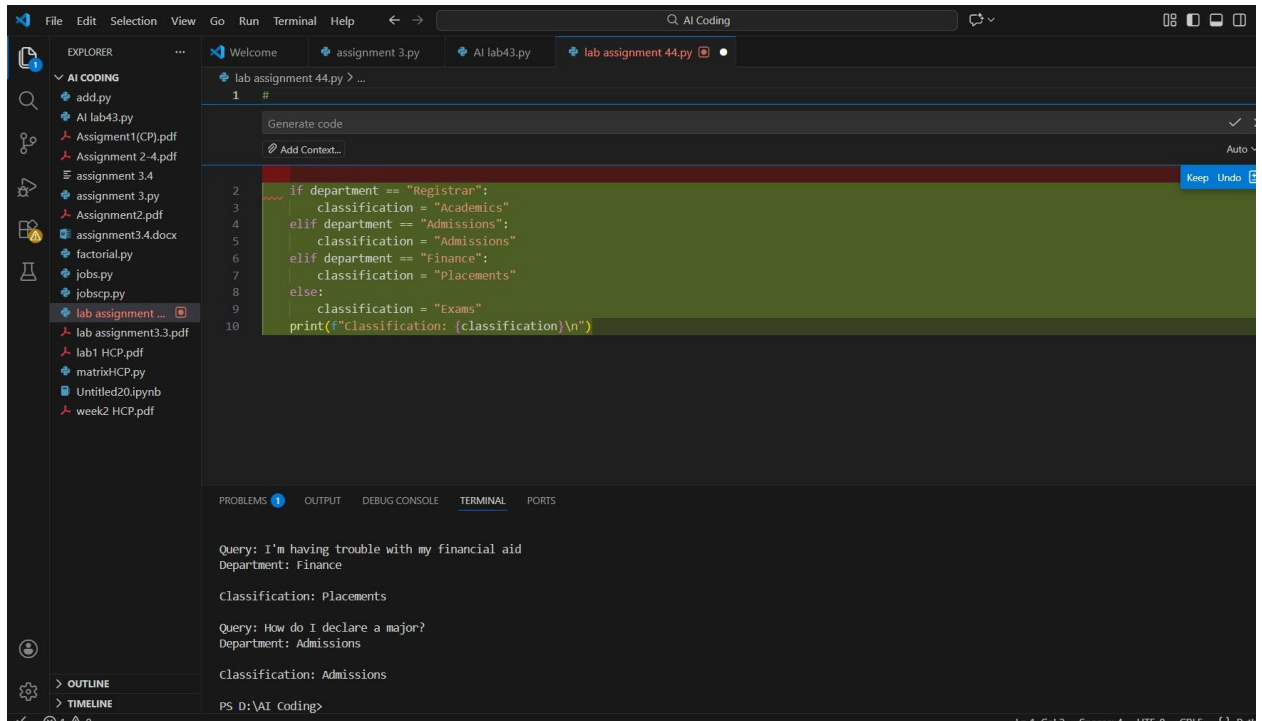
```
Query: How do I register for courses?
Department: Registrar

Query: What is my current GPA?
Department: Admissions

Query: I need to pay my tuition
Department: Finance

Query: Can I get a transcript?
Department: Registrar
```

3b) Implement Zero-shot intent classification using an LLM.



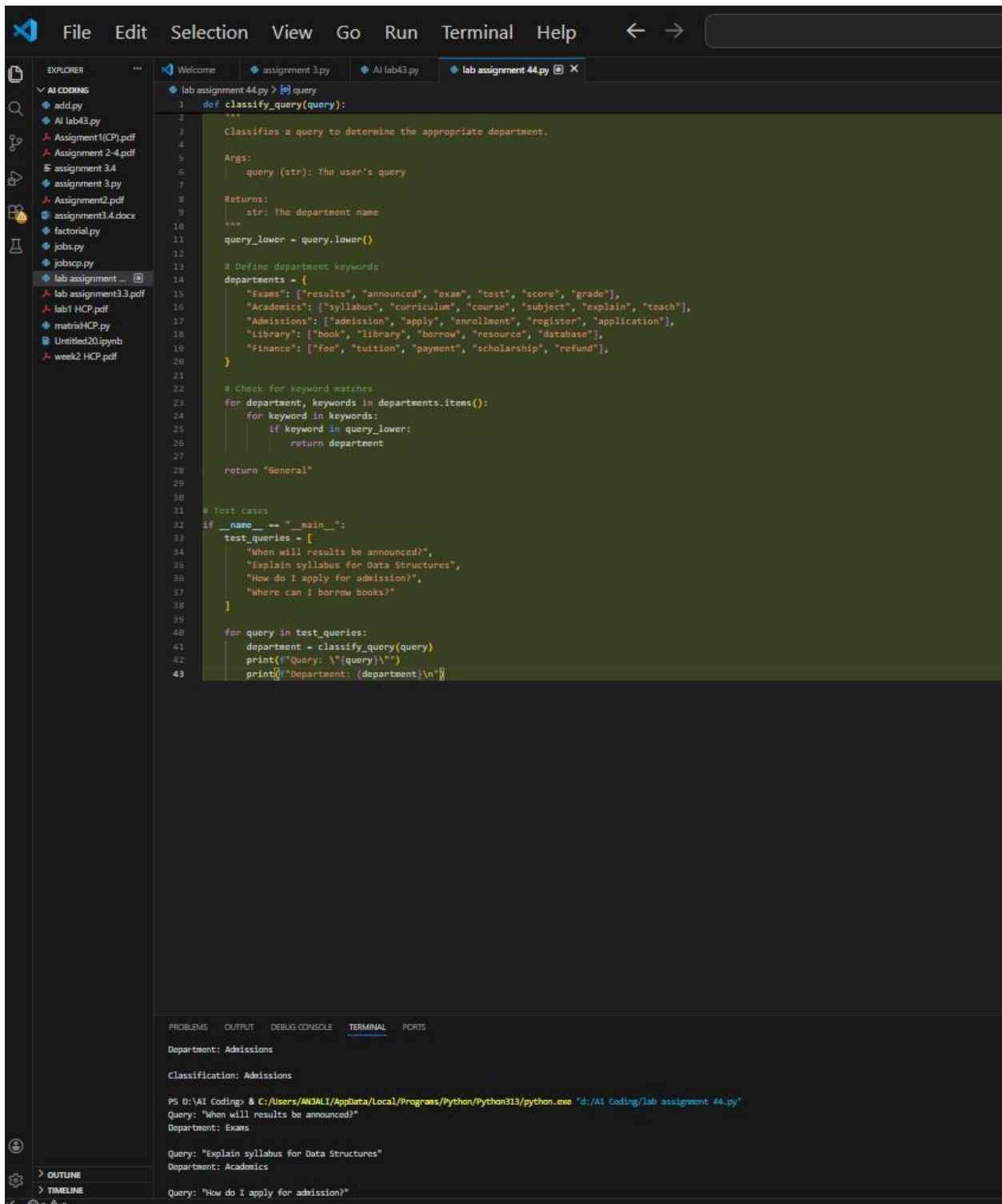
The screenshot displays a Visual Studio Code editor with a Python file named 'lab assignment 44.py'. The code implements a zero-shot intent classification system using a simple if-elif-else structure. The terminal window at the bottom shows two example queries and their corresponding classifications.

```
1 #  
2  
3 if department == "Registrar":  
4     classification = "Academics"  
5 elif department == "Admissions":  
6     classification = "Admissions"  
7 elif department == "Finance":  
8     classification = "Placements"  
9 else:  
10    classification = "Exams"  
11 print(f"Classification: {classification}\n")
```

Terminal Output:

```
Query: I'm having trouble with my financial aid  
Department: Finance  
Classification: Placements  
  
Query: How do I declare a major?  
Department: Admissions  
Classification: Admissions  
  
PS D:\AI Coding>
```


3c) mprove results using One-shot prompting.



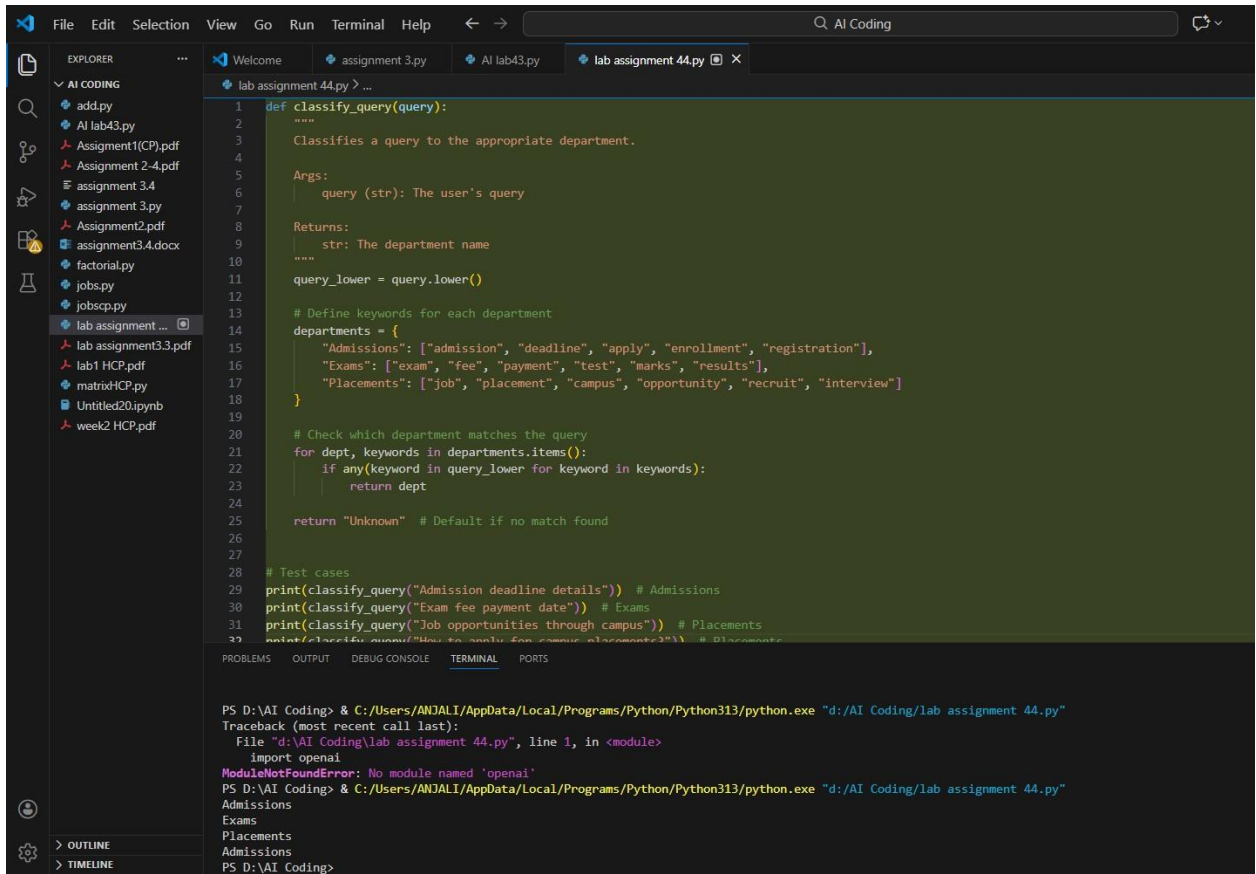
The screenshot shows a VS Code editor with a Python file named `lab assignment 44.py`. The script defines a function `classify_query` that takes a query string and returns a department name based on keyword matching. The keywords are organized into a dictionary: `Exams` (results, announced, exams, test, score, grade), `Academics` (syllabus, curriculum, course, subject, explain, teach), `Admissions` (admission, apply, enrollment, register, application), `Library` (book, library, borrow, resource, database), and `Finance` (fee, tuition, payment, scholarship, refund). The script includes test cases and prints the results for three queries.

```
1 def classify_query(query):
2     """
3     Classifies a query to determine the appropriate department.
4
5     Args:
6         query (str): The user's query
7
8     Returns:
9         str: The department name
10    """
11    query_lower = query.lower()
12
13    # Define department keywords
14    departments = {
15        "Exams": ["results", "announced", "exams", "test", "score", "grade"],
16        "Academics": ["syllabus", "curriculum", "course", "subject", "explain", "teach"],
17        "Admissions": ["admission", "apply", "enrollment", "register", "application"],
18        "Library": ["book", "library", "borrow", "resource", "database"],
19        "Finance": ["fee", "tuition", "payment", "scholarship", "refund"],
20    }
21
22    # Check for keyword matches
23    for department, keywords in departments.items():
24        for keyword in keywords:
25            if keyword in query_lower:
26                return department
27
28    return "General"
29
30
31 # Test cases
32 if __name__ == "__main__":
33     test_queries = [
34         "When will results be announced?",
35         "Explain syllabus for Data Structures",
36         "How do I apply for admission?",
37         "Where can I borrow books?"
38     ]
39
40     for query in test_queries:
41         department = classify_query(query)
42         print(f"Query: \"{query}\"")
43         print(f"Department: {department}\n")
```

The terminal output shows the results of the script execution:

```
Department: Admissions
Classification: Admissions
PS D:\AI Coding> & C:\Users\MSJALI\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Coding/lab assignment 44.py"
Query: "When will results be announced?"
Department: Exams
Query: "Explain syllabus for Data Structures"
Department: Academics
Query: "How do I apply for admission?"
```

3d) Further refine results using Few-shot prompting.



3e) Analyze how contextual examples affect classification accuracy.

The screenshot shows a VS Code editor with a file explorer on the left containing various files like 'add.py', 'AI lab43.py', and 'lab assignment 44.py'. The main editor window displays a Python script for a `ClassificationAnalyzer` class. The script includes methods for zero-shot and one-shot prompts, using `collections.defaultdict` and `json` modules. The terminal window at the bottom shows the following output:

```
Problems OUTPUT DEBUG CONSOLE TERMINAL PORTS

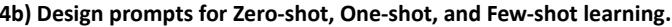
Ambiguity: Reduced - one example clarifies intent
Consistency: Improved - example sets pattern
Pros: Minimal overhead, Some context
Cons: Limited learning from one example

FEW_SHOT:
Accuracy: Higher - multiple references provided
Ambiguity: Significantly reduced - pattern clear
Consistency: High - multiple examples establish standard
Pros: Best accuracy, Clear patterns, Reduced errors
Cons: Requires manual examples, Prompt size
PS D:\AI Coding>
```

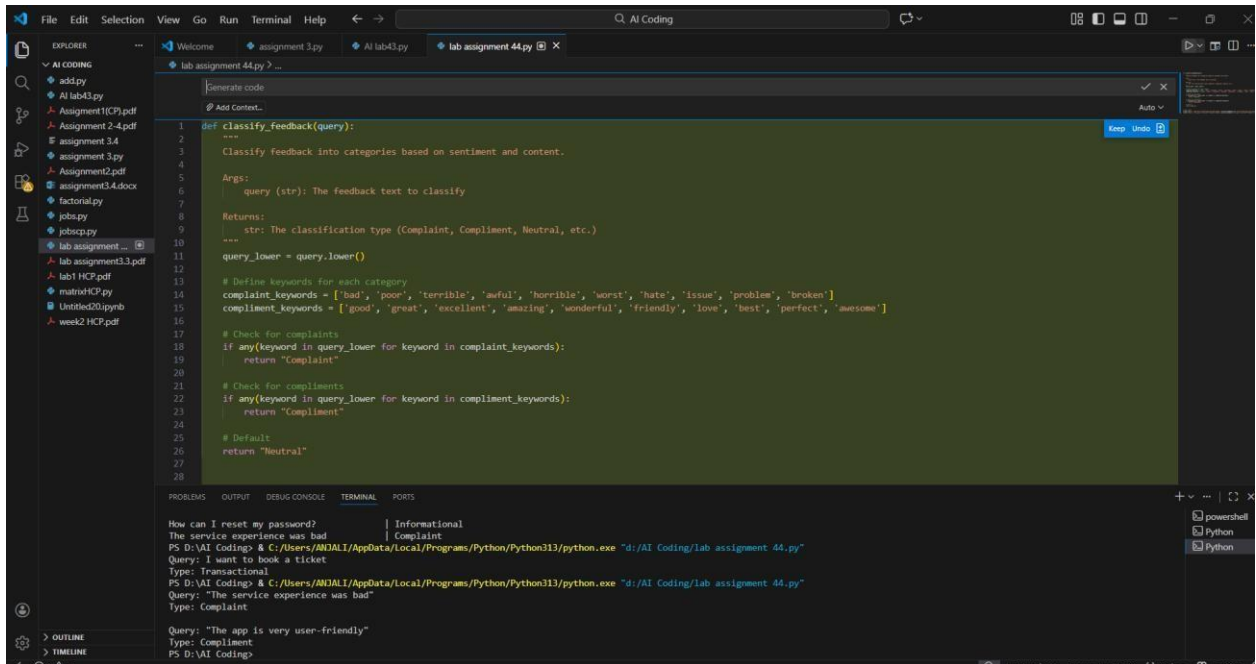
4) Chatbot Question Type Detection Scenario:

A chatbot must identify whether a user query is Informational, Transactional, Complaint, or Feedback.

4a) Prepare 6 chatbot queries mapped to question types.



4c) Test all prompts on the same unseen queries.



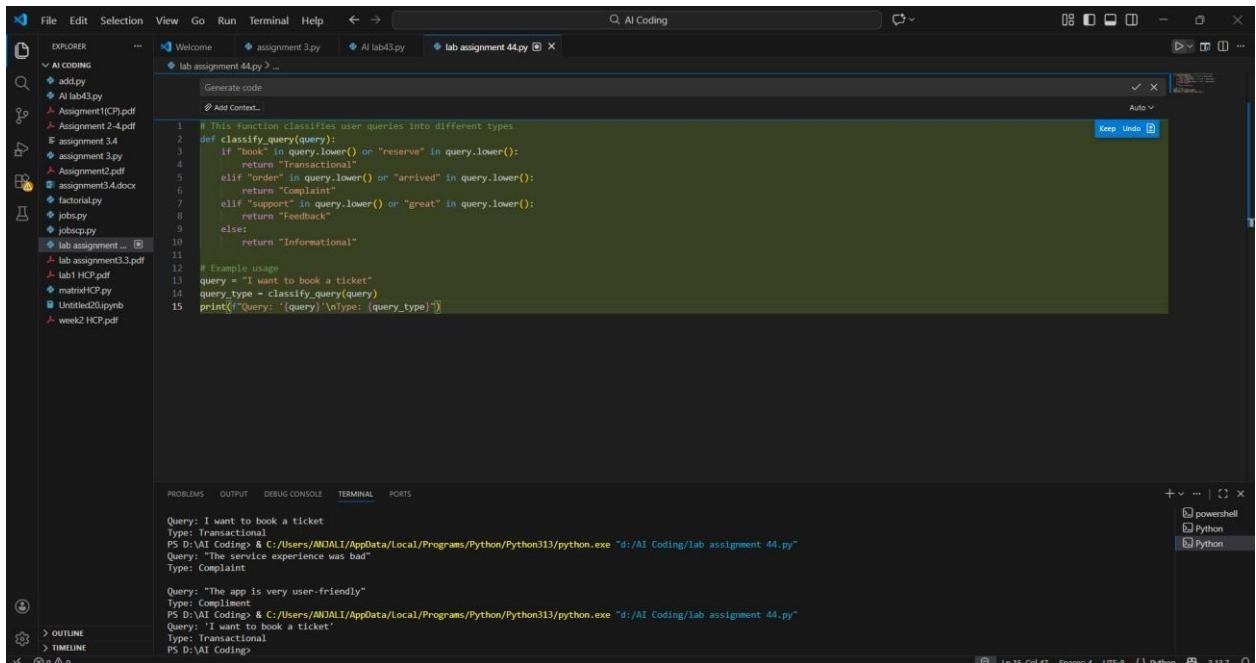
The screenshot shows a VS Code editor with a file explorer on the left containing various assignment files. The main editor displays a Python script named `lab_assignment_44.py`. The script defines a function `classify_feedback(query)` that classifies feedback into categories based on sentiment and content. It includes keywords for complaints and compliments and checks the input query against these keywords to return a classification type (Complaint, Compliment, Neutral, etc.). The terminal at the bottom shows the execution of the script with several test queries and their corresponding classifications.

```
def classify_feedback(query):  
    """  
    Classify feedback into categories based on sentiment and content.  
    """  
    Args:  
        query (str): The feedback text to classify  
    Returns:  
        str: The classification type (Complaint, Compliment, Neutral, etc.)  
    """  
    query_lower = query.lower()  
  
    # Define keywords for each category  
    complaint_keywords = ['bad', 'poor', 'terrible', 'awful', 'horrible', 'worst', 'hate', 'issue', 'problem', 'broken']  
    compliment_keywords = ['good', 'great', 'excellent', 'amazing', 'wonderful', 'friendly', 'love', 'best', 'perfect', 'awesome']  
  
    # Check for complaints  
    if any(keyword in query_lower for keyword in complaint_keywords):  
        return "Complaint"  
  
    # Check for compliments  
    if any(keyword in query_lower for keyword in compliment_keywords):  
        return "Compliment"  
  
    # Default  
    return "Neutral"
```

Terminal Output:

```
How can I reset my password?  
Type: Transactional  
The service experience was bad  
Type: Complaint  
Query: "The service experience was bad"  
Type: Complaint  
Query: "The app is very user-friendly"  
Type: Compliment  
PS D:\AI Coding>
```

4d) Compare response correctness and ambiguity handling.



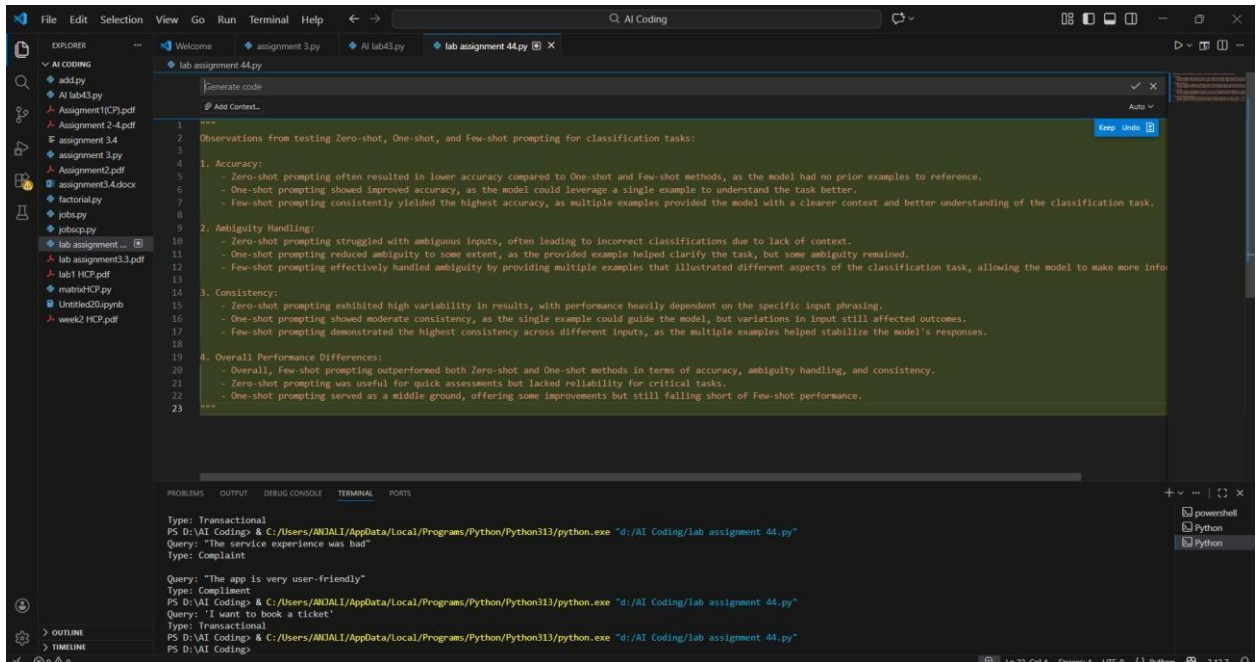
The screenshot shows a VS Code editor with a file explorer on the left. The main editor displays a Python script named `lab_assignment_44.py`. The script defines a function `classify_query(query)` that classifies user queries into different types based on specific keywords. It includes keywords for transactional, complaint, support, feedback, and informational queries. The terminal at the bottom shows the execution of the script with several test queries and their corresponding classifications.

```
# This function classifies user queries into different types  
def classify_query(query):  
    if "book" in query.lower() or "reserve" in query.lower():  
        return "Transactional"  
    elif "order" in query.lower() or "arrived" in query.lower():  
        return "Complaint"  
    elif "support" in query.lower() or "great" in query.lower():  
        return "Feedback"  
    else:  
        return "Informational"
```

Terminal Output:

```
Query: I want to book a ticket  
Type: Transactional  
PS D:\AI Coding>  
Query: "The service experience was bad"  
Type: Complaint  
Query: "The app is very user-friendly"  
Type: Compliment  
PS D:\AI Coding>  
Query: "I want to book a ticket"  
Type: Transactional  
PS D:\AI Coding>
```

4e) Document observations.

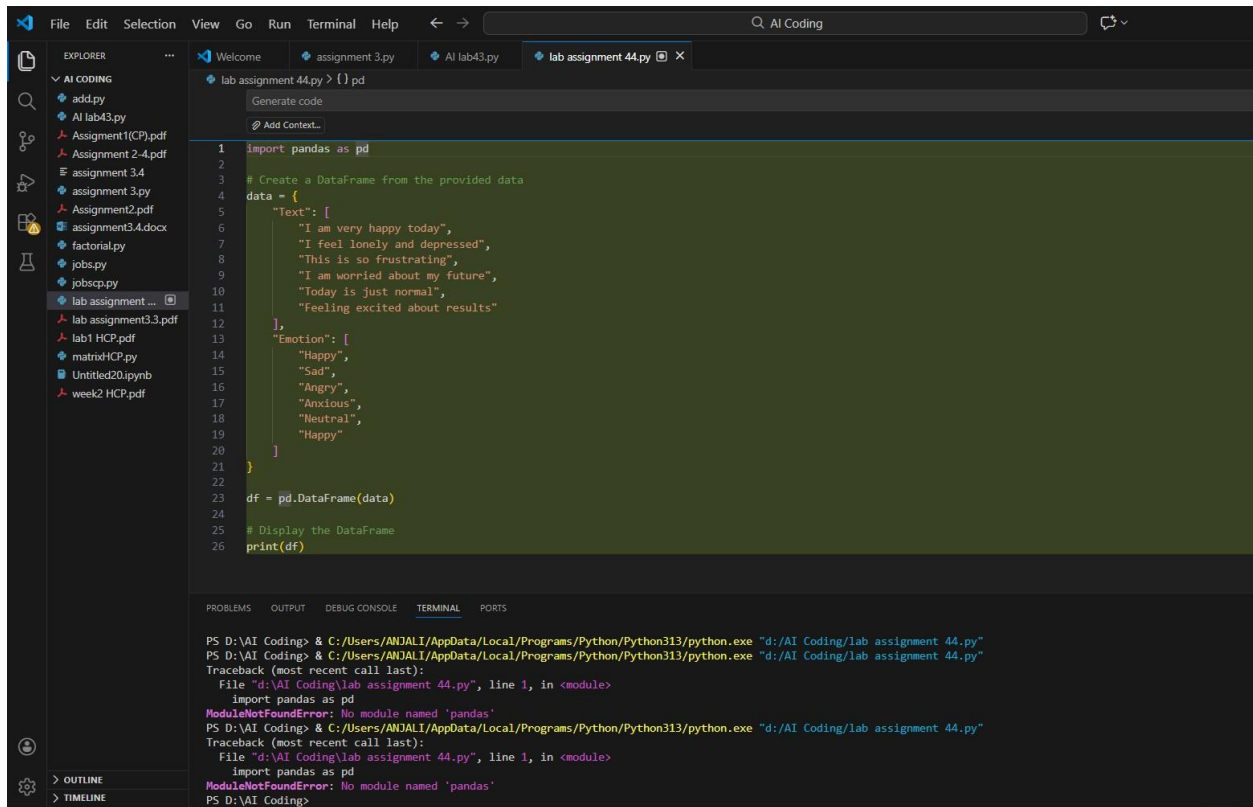


5)Emotion Detection in Text

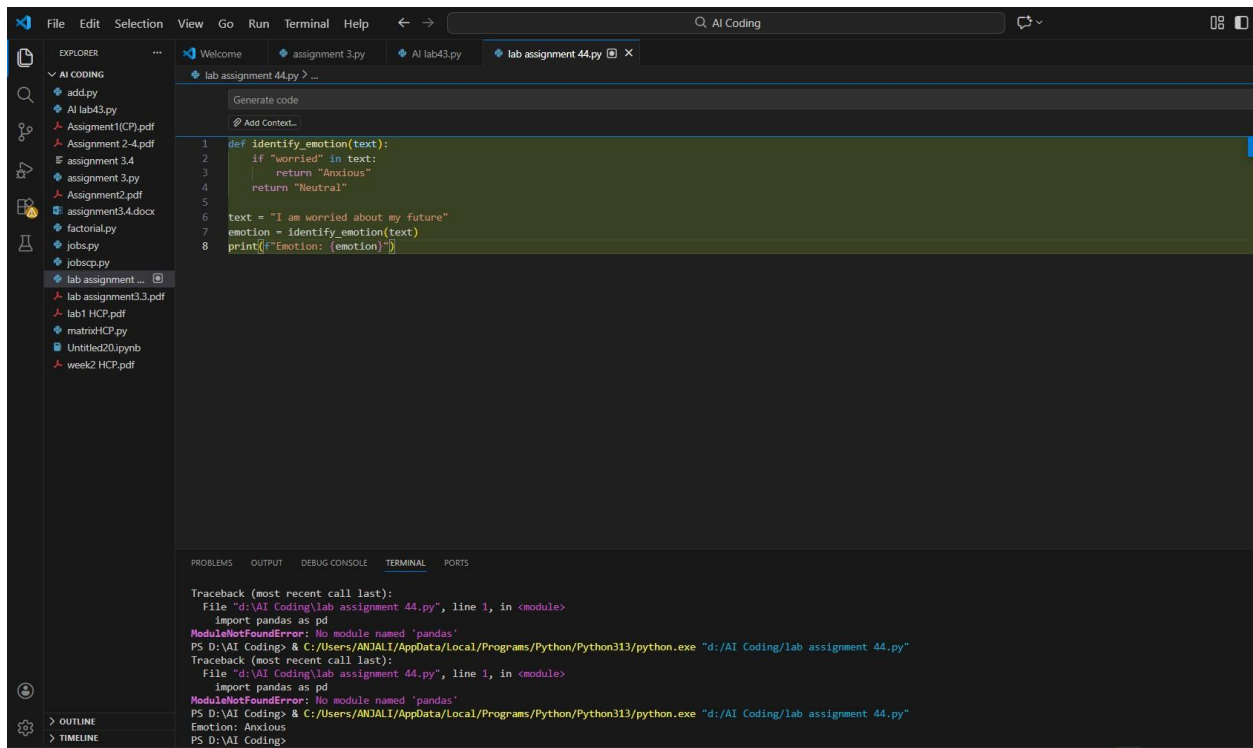
Scenario:

A mental-health chatbot needs to detect emotions: Happy, Sad, Angry, Anxious, Neutral.

5a)Create labeled emotion samples.



5b) Use Zero-shot prompting to identify emotions.

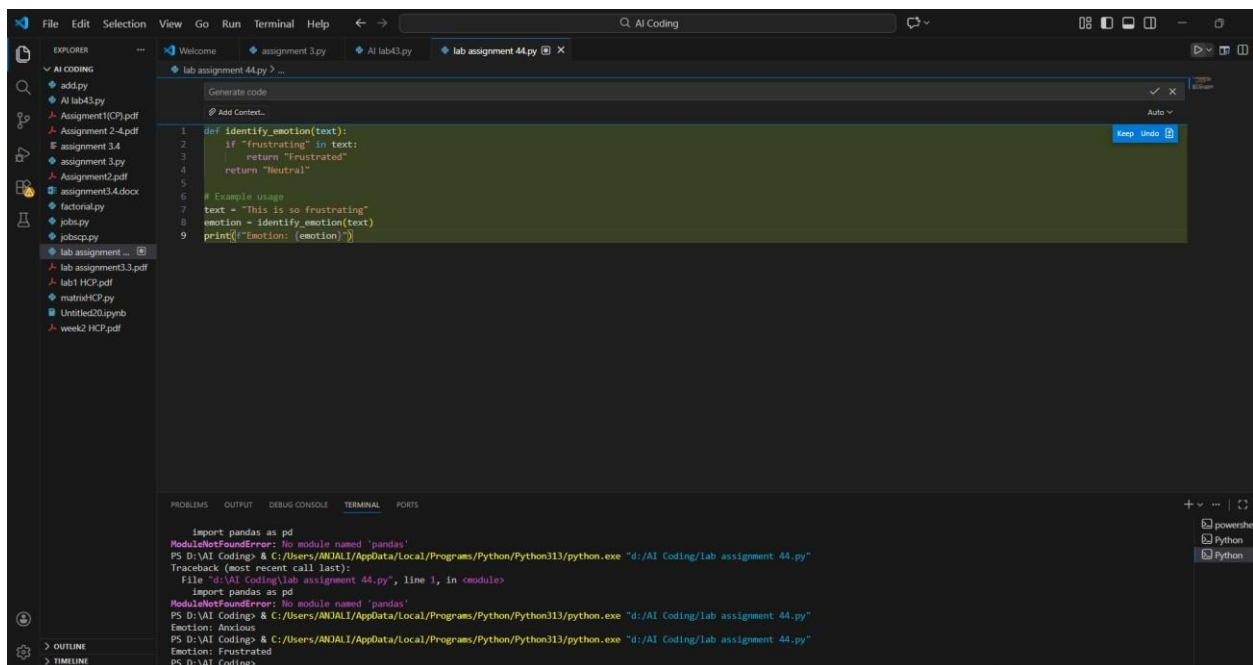


The screenshot shows the Visual Studio Code editor with a file explorer on the left containing various assignment files. The main editor window displays a Python script named 'lab assignment 44.py'. The script defines a function 'identify_emotion(text)' that checks for the word 'worried' and returns 'Anxious' or 'Neutral'. Below the function, there is a test case where 'text' is set to 'I am worried about my future', the function is called, and the result is printed. The terminal at the bottom shows a 'ModuleNotFoundError: No module named 'pandas'' error, which is unrelated to the script's logic.

```
1 def identify_emotion(text):
2     if "worried" in text:
3         return "Anxious"
4     return "Neutral"
5
6 text = "I am worried about my future"
7 emotion = identify_emotion(text)
8 print(f"Emotion: {emotion}")
```

Traceback (most recent call last):
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding

5c) Use One-shot prompting with an example.



The screenshot shows the Visual Studio Code editor with a file explorer on the left. The main editor window displays a Python script named 'lab assignment 44.py'. The script defines a function 'identify_emotion(text)' that checks for the word 'frustrating' and returns 'Frustrated' or 'Neutral'. Below the function, there is a test case where 'text' is set to 'This is so frustrating', the function is called, and the result is printed. The terminal at the bottom shows a 'ModuleNotFoundError: No module named 'pandas'' error, which is unrelated to the script's logic.

```
1 def identify_emotion(text):
2     if "frustrating" in text:
3         return "Frustrated"
4     return "Neutral"
5
6 # Example usage
7 text = "This is so frustrating"
8 emotion = identify_emotion(text)
9 print(f"Emotion: {emotion}")
```

Import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding

5d) Use Few-shot prompting with multiple emotions.

The screenshot shows the Visual Studio Code interface with a file explorer on the left containing various assignment files. The main editor displays a Python script named `lab assignment 44.py`. The script defines a `classify_emotion(text)` function that maps words to emotions like happy, sad, anxious, neutral, and frustrated. It includes an example usage section where the text "This is so frustrating" is processed, resulting in the emotion "Frustrated". The terminal at the bottom shows a `Traceback (most recent call last):` error, indicating a `ModuleNotFoundError: No module named 'pandas'` because the script attempts to `import pandas as pd` which is not installed in the current environment.

```
1 def classify_emotion(text):
2     emotions = {
3         "happy": ["happy", "joyful", "excited", "pleased"],
4         "sad": ["lonely", "depressed", "sad", "down"],
5         "anxious": ["worried", "anxious", "nervous", "stressed"],
6         "neutral": ["normal", "fine", "okay", "average"],
7         "frustrated": ["frustrating", "annoyed", "irritated"]
8     }
9
10    for emotion, keywords in emotions.items():
11        if any(keyword in text.lower() for keyword in keywords):
12            return emotion
13    return "Unknown"
14
15 # Example usage
16 text = "This is so frustrating"
17 emotion = classify_emotion(text)
18 print(f"Text: \"{text}\"\\nEmotion: {emotion}")
```

Traceback (most recent call last):
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Text: "This is so frustrating"
Emotion: frustrated
PS D:\AI Coding>

5e) Discuss ambiguity handling across techniques.

The screenshot shows the Visual Studio Code interface with the same file explorer. The main editor displays a Python script named `lab assignment 44.py` that implements a `handle_emotion(technique, input_text)` function. This function uses a `zero-shot` technique to handle ambiguity by returning specific messages for different techniques: "This technique struggles with ambiguity in understanding emotions." for `zero-shot`, "This technique provides better clarity in emotional interpretation." for `one-shot`, and "This technique achieves the best emotional accuracy by learning from examples." for `few-shot`. The terminal shows the output of running the script, demonstrating how these techniques handle the ambiguous text "This is so frustrating".

```
1 # Emotion Handling Techniques
2
3 def handle_emotion(technique, input_text):
4     if technique == "zero-shot":
5         return "This technique struggles with ambiguity in understanding emotions."
6     elif technique == "one-shot":
7         return "This technique provides better clarity in emotional interpretation."
8     elif technique == "few-shot":
9         return "This technique achieves the best emotional accuracy by learning from examples."
10    else:
11        return "Unknown technique."
12
13 # Example usage
14 techniques = ["zero-shot", "one-shot", "few-shot"]
15 for technique in techniques:
16     print(f"{technique.capitalize()}: {handle_emotion(technique, '')}")
```

PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Text: "This is so frustrating"
Emotion: frustrated
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Zero-shot: This technique struggles with ambiguity in understanding emotions.
One-shot: This technique provides better clarity in emotional interpretation.
Few-shot: This technique achieves the best emotional accuracy by learning from examples.
PS D:\AI Coding>