

ASSIGNMENT-4.5

NAME-ARSHA VARDHINI

BATCH-29

ROLL-NO:2303A51600

ADVANCED PROMPT ENGINEERING: ZERO-SHOT, ONE-SHOT & FEW-SHOT

TASK-1:

ZERO-SHOT

A. Preparing Sample data:

```
test_emails = [  
    "My payment failed but money was deducted.",  
    "The app is not opening on my phone.",  
    "Great customer service, very satisfied.",  
    "What is your customer care number?",  
    "Invoice amount seems incorrect."  
]
```

Expected Labels (for evaluation):

```
true_labels = [  
    "Billing",  
    "Technical Support",
```

```
"Feedback",  
"Others",  
"Billing"  
]
```

PROMPT:

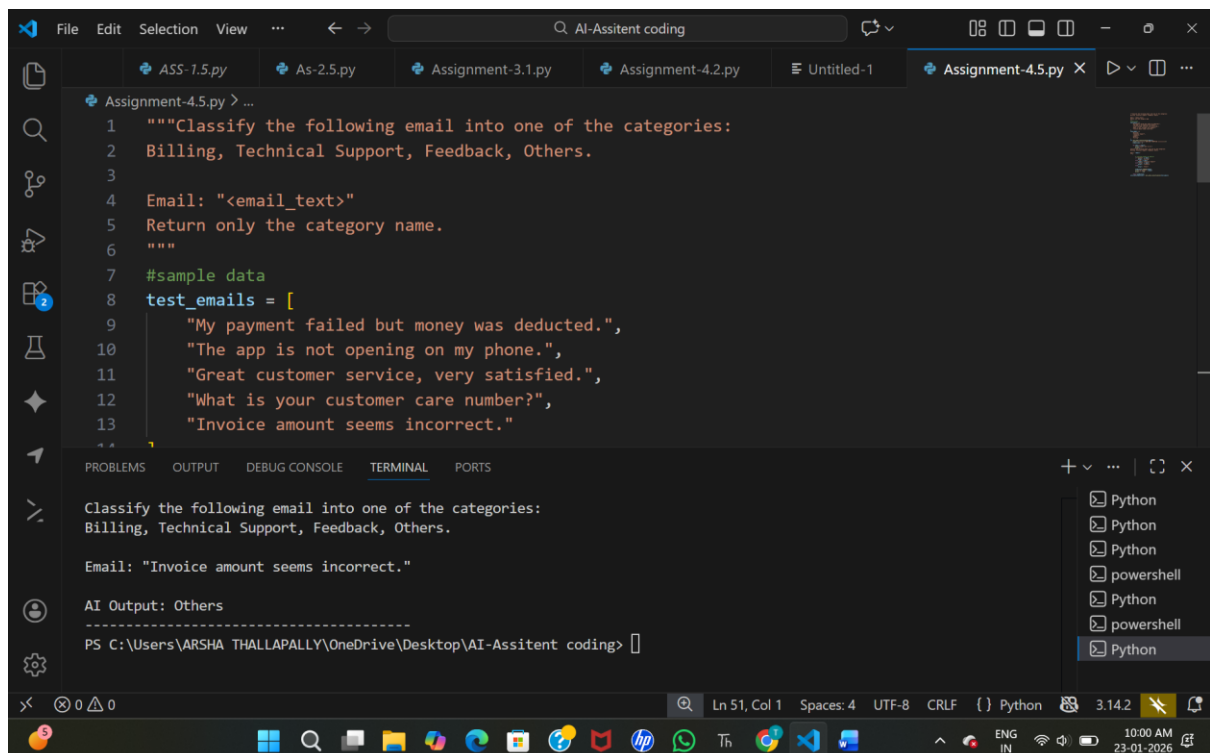
Classify the following email into one of the categories:

Billing, Technical Support, Feedback, Others.

Email: "<email_text>"

Return only the category name.

CODE:



The screenshot shows a Visual Studio Code editor window with a Python file named 'Assignment-4.5.py'. The code in the file is as follows:

```
1 """Classify the following email into one of the categories:  
2 Billing, Technical Support, Feedback, Others.  
3  
4 Email: "<email_text>"  
5 Return only the category name.  
6 """  
7 #sample data  
8 test_emails = [  
9     "My payment failed but money was deducted.",  
10    "The app is not opening on my phone.",  
11    "Great customer service, very satisfied.",  
12    "What is your customer care number?",  
13    "Invoice amount seems incorrect."  
14 ]
```

The terminal window at the bottom shows the output of the script:

```
Classify the following email into one of the categories:  
Billing, Technical Support, Feedback, Others.  
  
Email: "Invoice amount seems incorrect."  
  
AI Output: Others  
-----  
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

The status bar at the bottom indicates the file is at line 51, column 1, using UTF-8 encoding with CRLF line endings. The taskbar at the very bottom shows the system clock as 10:00 AM on 23-01-2026.

OBSERVATION:

Classifies emails using only instructions, without examples.

Works if keywords are clear, may misclassify ambiguous emails.

Quick and simple, but less accurate for complex cases.

ONE-SHOT :

PROMPT: Example:

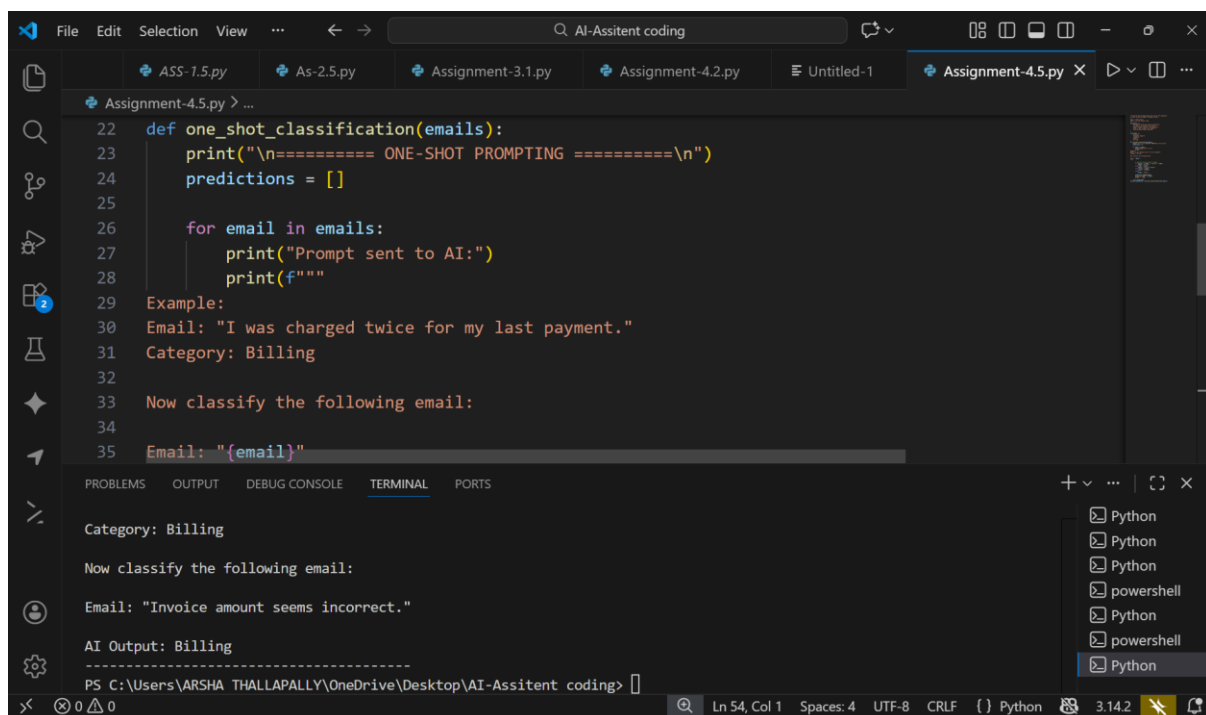
Email: "I was charged twice for my last payment."

Category: Billing

Now classify the following email:

Email: "<email_text>"

CODE:



```
22 def one_shot_classification(emails):
23     print("\n===== ONE-SHOT PROMPTING =====\n")
24     predictions = []
25
26     for email in emails:
27         print("Prompt sent to AI:")
28         print(f"""
29 Example:
30 Email: "I was charged twice for my last payment."
31 Category: Billing
32
33 Now classify the following email:
34
35 Email: "{email}"
36 """)
```

Category: Billing

Now classify the following email:

Email: "Invoice amount seems incorrect."

AI Output: Billing

PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides one example to guide the AI's reasoning.

Improves accuracy over zero-shot and handles slightly ambiguous emails better.

Still limited; accuracy depends on how representative the example is.

One example helps the AI understand the expected format and category mapping.

Classification accuracy improves compared to zero-shot, especially for similar issues.

Performance depends heavily on how relevant the single example is to the new email.

FEW SHOT:

PROMPT:

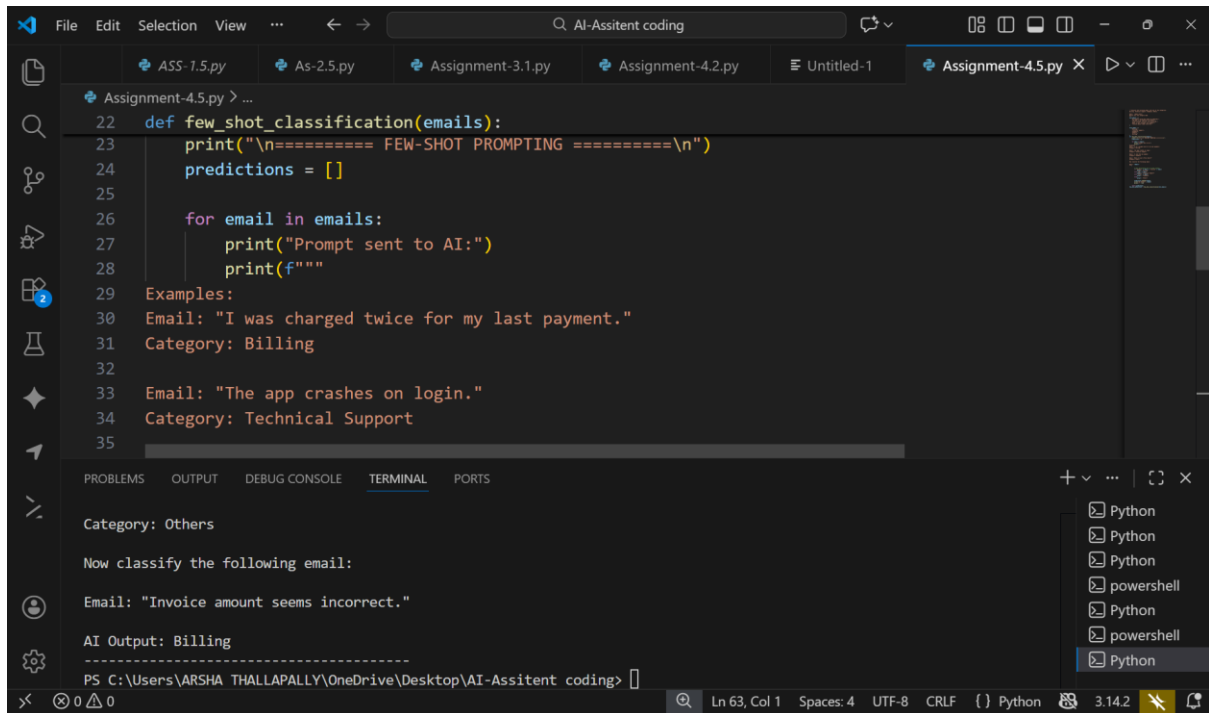
Email: "I was charged twice for my last payment." → Billing

Email: "The app crashes on login." → Technical Support

Email: "I love the new update." → Feedback

Email: "What are your office hours?" → Others

CODE:



```
File Edit Selection View ... AI-Assitent coding
Assignment-4.5.py > ...
22 def few_shot_classification(emails):
23     print("\n===== FEW-SHOT PROMPTING =====\n")
24     predictions = []
25
26     for email in emails:
27         print("Prompt sent to AI:")
28         print(f"""
29 Examples:
30 Email: "I was charged twice for my last payment."
31 Category: Billing
32
33 Email: "The app crashes on login."
34 Category: Technical Support
35
Category: Others

Now classify the following email:

Email: "Invoice amount seems incorrect."

AI Output: Billing
-----
PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Provides multiple examples to show patterns to the AI.

Highest accuracy; AI can generalize better for unseen emails.

Slightly longer prompts but most reliable for real-world use

TASK-2:

Sample travel queries (short & simple)

```
travel_queries = [  
    "Book a flight from Delhi to Mumbai.",  
    "Cancel my hotel reservation in Paris.",  
    "What is the baggage allowance?",  
    "I need a hotel in London for 2 nights.",  
    "Cancel my flight ticket to New York."  
]
```

True labels for evaluation

```
true_labels = [  
    "Flight Booking",  
    "Cancellation",  
    "General Travel Info",  
    "Hotel Booking",  
    "Cancellation"  
]
```

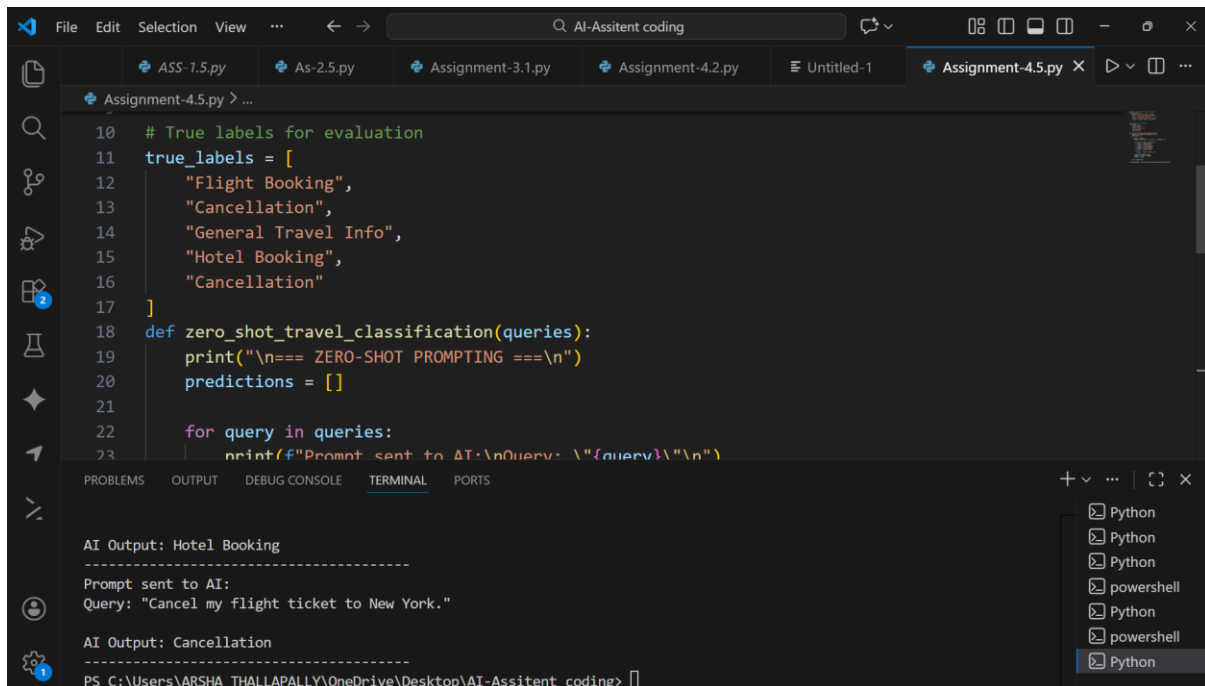
ZERO-SHOT:

PROMPT: Classify the following travel query into one of the categories:

Flight Booking, Hotel Booking, Cancellation, General Travel Info.

Query: "<travel_query>"

CODE:



```
10 # True labels for evaluation
11 true_labels = [
12     "Flight Booking",
13     "Cancellation",
14     "General Travel Info",
15     "Hotel Booking",
16     "Cancellation"
17 ]
18 def zero_shot_travel_classification(queries):
19     print("\n=== ZERO-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI:\nQuery: \"{query}\"")
24         # ... (AI interaction logic) ...
25         predictions.append("Cancellation")
26
27 # Example usage
28 queries = ["Cancel my flight ticket to New York."]
29 zero_shot_travel_classification(queries)
```

AI Output: Hotel Booking

Prompt sent to AI:
Query: "Cancel my flight ticket to New York."

AI Output: Cancellation

PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Classifies queries using only instructions, without examples.

Works for obvious keywords like “flight” or “cancel”, may misclassify tricky queries.

Fast and simple, but accuracy is lower for ambiguous cases.

ONE-SHOT:

PROMPT: Example:

Query: "Cancel my flight ticket."

Category: Cancellation

Now classify the following query:

Query: "<travel_query>"

CODE:

```
18 def one_shot_travel_classification(queries):
19     print("\n=== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with one example:\nQuery: \"{query}\"")
24
25         if "cancel" in query.lower():
26             output = "Cancellation"
27         elif "flight" in query.lower():
28             output = "Flight Booking"
29         elif "hotel" in query.lower():
30             output = "Hotel Booking"
31         else:
```

AI Output: Hotel Booking

Prompt sent to AI with one example:
Query: "Cancel my flight ticket to New York."

AI Output: Cancellation

PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides one example to guide AI's reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the example is.

FEW SHOT:

PROMPT:

Examples:

Query: "Book a flight to Mumbai."

Category: Flight Booking

Query: "Cancel my hotel reservation."

Category: Cancellation

Query: "I need a hotel in London."

Category: Hotel Booking

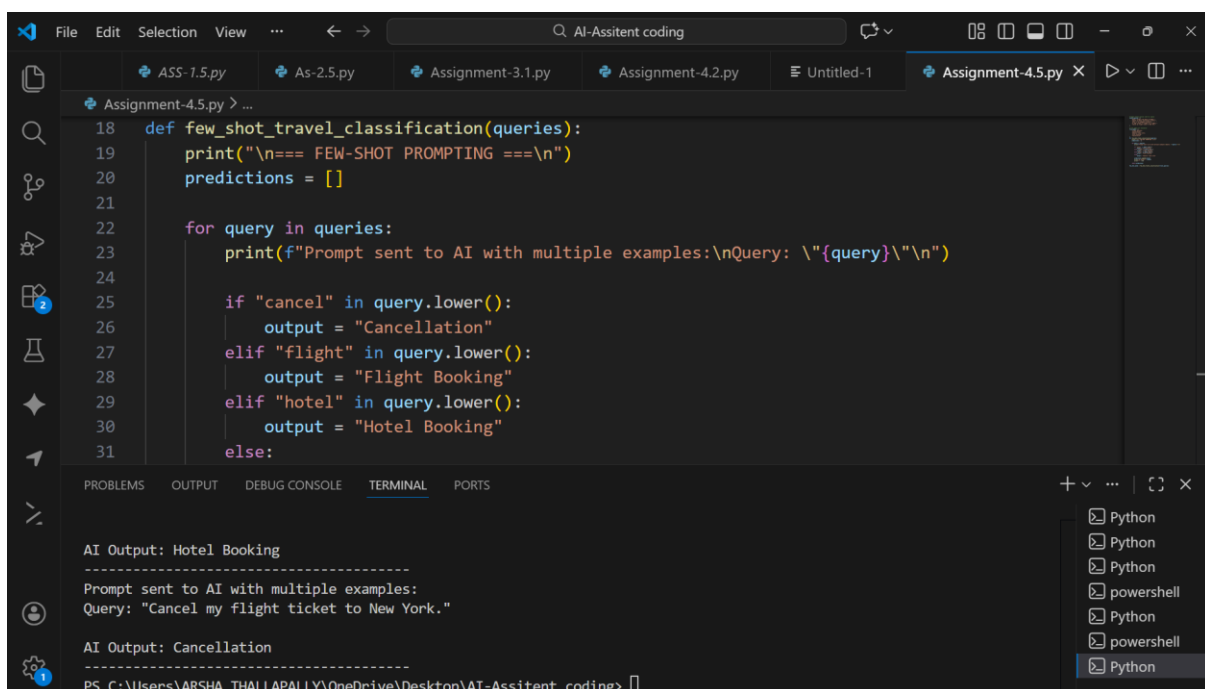
Query: "What is the baggage allowance?"

Category: General Travel Info

Now classify the following query:

Query: "<travel_query>"

CODE:



The screenshot shows a Visual Studio Code editor window with a Python file named 'Assignment-4.5.py'. The code defines a function 'few_shot_travel_classification' that takes a list of queries and returns a list of predicted categories. The function uses a series of 'if' and 'elif' statements to check for keywords like 'cancel', 'flight', and 'hotel' in the queries. The terminal output shows the function being called with a list of queries, and the predicted categories being printed.

```
18 def few_shot_travel_classification(queries):
19     print("\n=== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"")
24
25         if "cancel" in query.lower():
26             output = "Cancellation"
27         elif "flight" in query.lower():
28             output = "Flight Booking"
29         elif "hotel" in query.lower():
30             output = "Hotel Booking"
31         else:
```

AI Output: Hotel Booking

Prompt sent to AI with multiple examples:

Query: "Cancel my flight ticket to New York."

AI Output: Cancellation

PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides multiple examples to show patterns to AI.

Highest accuracy; AI generalizes better for unseen queries.

Slightly longer prompts but most reliable for real-world use.

TASK-3:

SAMPLE DATA:

Sample coding queries (short & simple)

```
coding_queries = [  
    "Why am I getting IndexError in my Python list?",  
    "My sorting algorithm is too slow for large inputs.",  
    "I wrote a function but it returns wrong results.",  
    "Explain the difference between list and tuple in Python.",  
    "How can I optimize my recursive Fibonacci function?"  
]
```

True labels for evaluation

```
true_labels = [  
    "Syntax Error",  
    "Optimization",  
    "Logic Error",  
    "Conceptual Question",  
    "Optimization"  
]
```

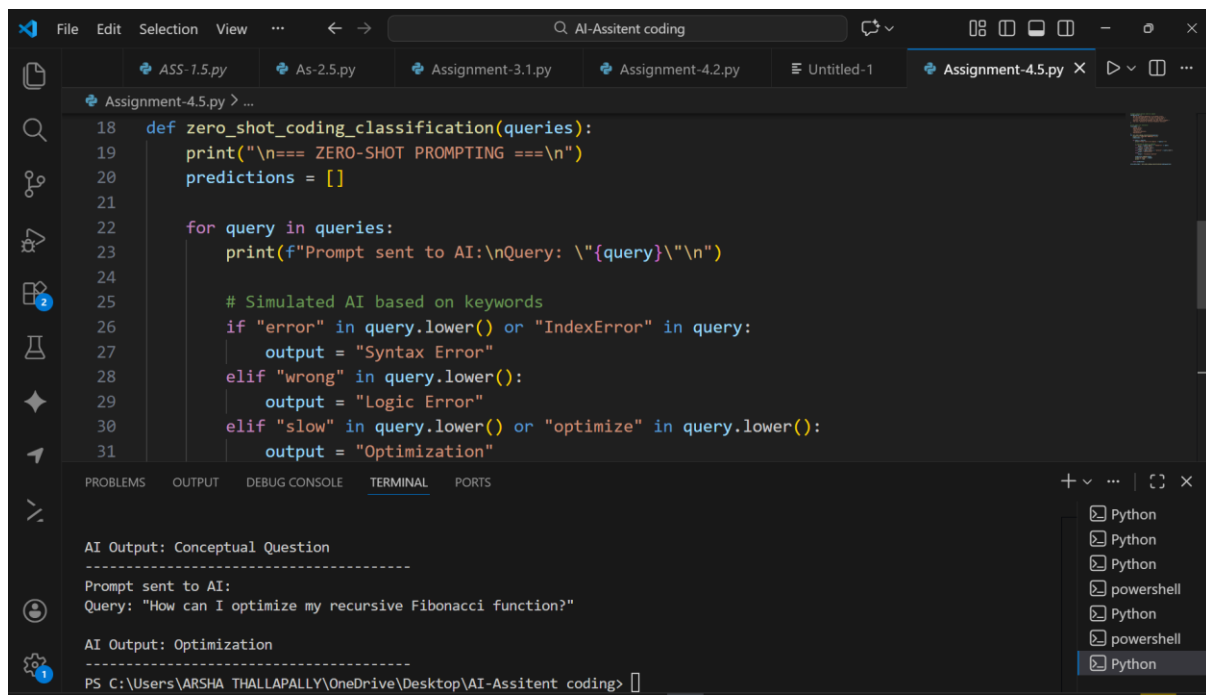
ZERO-SHOT

PROMPT: Classify the following coding query into one of the categories:

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "<coding_query>"

CODE:



```
18 def zero_shot_coding_classification(queries):
19     print("\n=== ZERO-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI:\nQuery: \"{query}\"")
24
25         # Simulated AI based on keywords
26         if "error" in query.lower() or "IndexError" in query:
27             output = "Syntax Error"
28         elif "wrong" in query.lower():
29             output = "Logic Error"
30         elif "slow" in query.lower() or "optimize" in query.lower():
31             output = "Optimization"
```

AI Output: Conceptual Question

Prompt sent to AI:

Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization

PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

ONE SHOT

PROMPT:

Example:

Query: "I want to cancel my Python function."

Category: Logic Error

Now classify the following coding query:

Query: "<coding_query>"

CODE:

```
18 def one_shot_coding_classification(queries):
19     print("\n=== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with one example:\nQuery: \"{query}\"")
24
25         if "error" in query.lower() or "IndexError" in query:
26             output = "Syntax Error"
27         elif "wrong" in query.lower():
28             output = "Logic Error"
29         elif "slow" in query.lower() or "optimize" in query.lower():
30             output = "Optimization"
31         else:
```

AI Output: Conceptual Question

Prompt sent to AI with one example:
Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides one example to guide AI's reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the single example is.

FEW SHOT

PROMPT:

Examples:

Query: "Why does my Python list give IndexError?"

Category: Syntax Error

Query: "My function returns wrong output."

Category: Logic Error

Query: "My loop is too slow for large data."

Category: Optimization

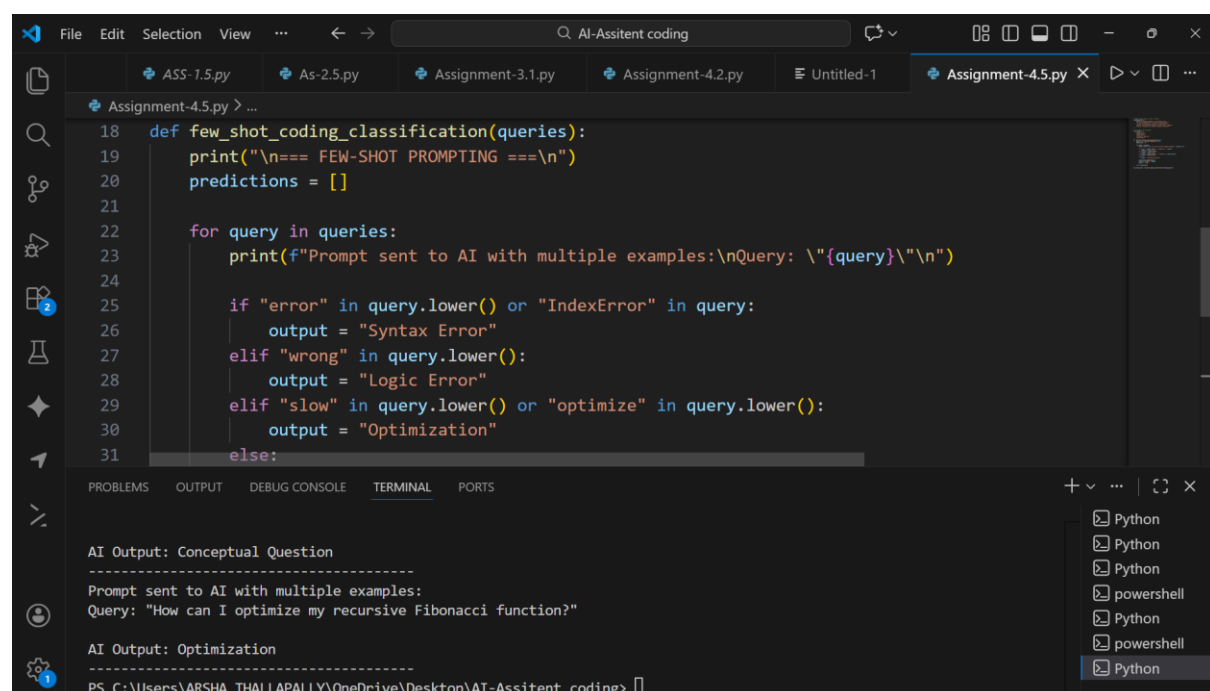
Query: "Explain Python variable scopes."

Category: Conceptual Question

Now classify the following coding query:

Query: "<coding_query>"

CODE:



```
File Edit Selection View ... AI-Assitent coding
Assignment-4.5.py > ...
18 def few_shot_coding_classification(queries):
19     print("\n=== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"")
24
25         if "error" in query.lower() or "IndexError" in query:
26             output = "Syntax Error"
27         elif "wrong" in query.lower():
28             output = "Logic Error"
29         elif "slow" in query.lower() or "optimize" in query.lower():
30             output = "Optimization"
31         else:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AI Output: Conceptual Question

Prompt sent to AI with multiple examples:
Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding> []

Python
Python
Python
powershell
Python
powershell
Python

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; AI generalizes better for unseen queries.

Slightly longer prompts but most reliable for technical classification.

TASK-4

ZERO-SHOT

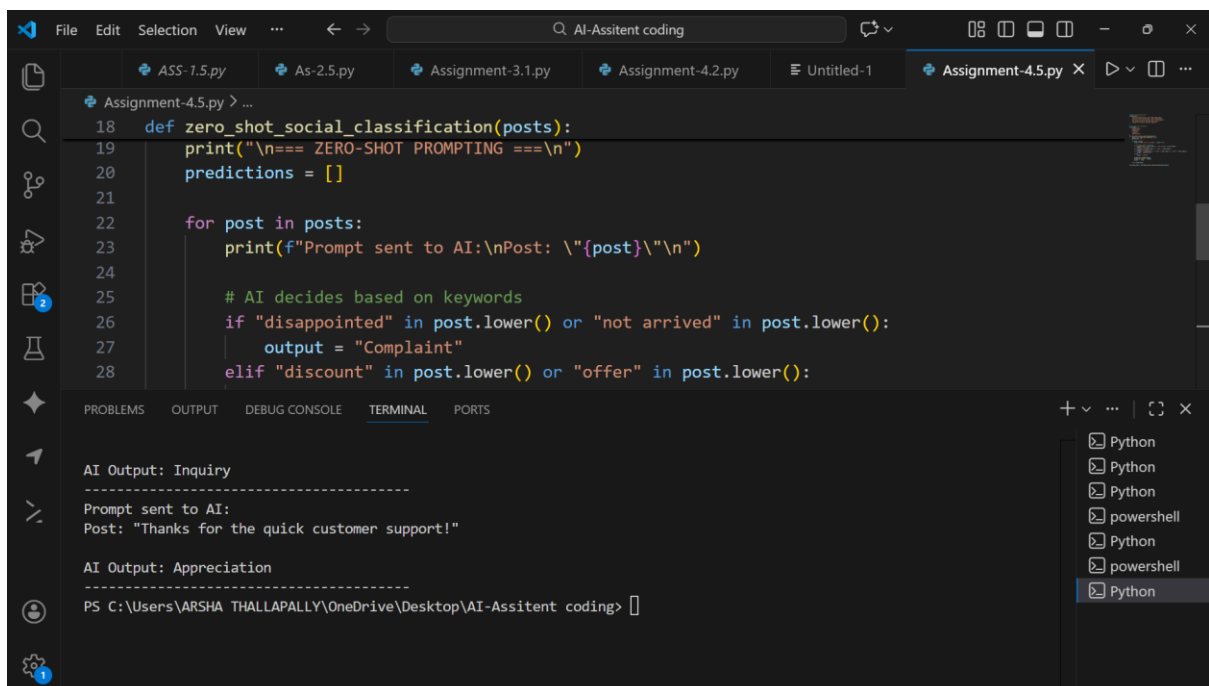
PROMPT:

Classify the following social media post into one of the categories:

Promotion, Complaint, Appreciation, Inquiry.

Post: "<social_post>"

CODE:



```
File Edit Selection View ... AI-Assitent coding
ASS-1.5.py As-2.5.py Assignment-3.1.py Assignment-4.2.py Untitled-1 Assignment-4.5.py
Assignment-4.5.py > ...
18 def zero_shot_social_classification(posts):
19     print("\n=== ZERO-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI:\nPost: \"{post}\"")
24
25         # AI decides based on keywords
26         if "disappointed" in post.lower() or "not arrived" in post.lower():
27             output = "Complaint"
28         elif "discount" in post.lower() or "offer" in post.lower():
29             output = "Promotion"
30         else:
31             output = "Inquiry"
32
33     return predictions

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
AI Output: Inquiry
-----
Prompt sent to AI:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation
-----
PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Classifies posts using only instructions, without examples.

Works for clear keywords but may misinterpret informal or slang language.

Fast and simple, lower accuracy for ambiguous or sarcastic posts.

ONE-SHOT

PROMPT:

Example:

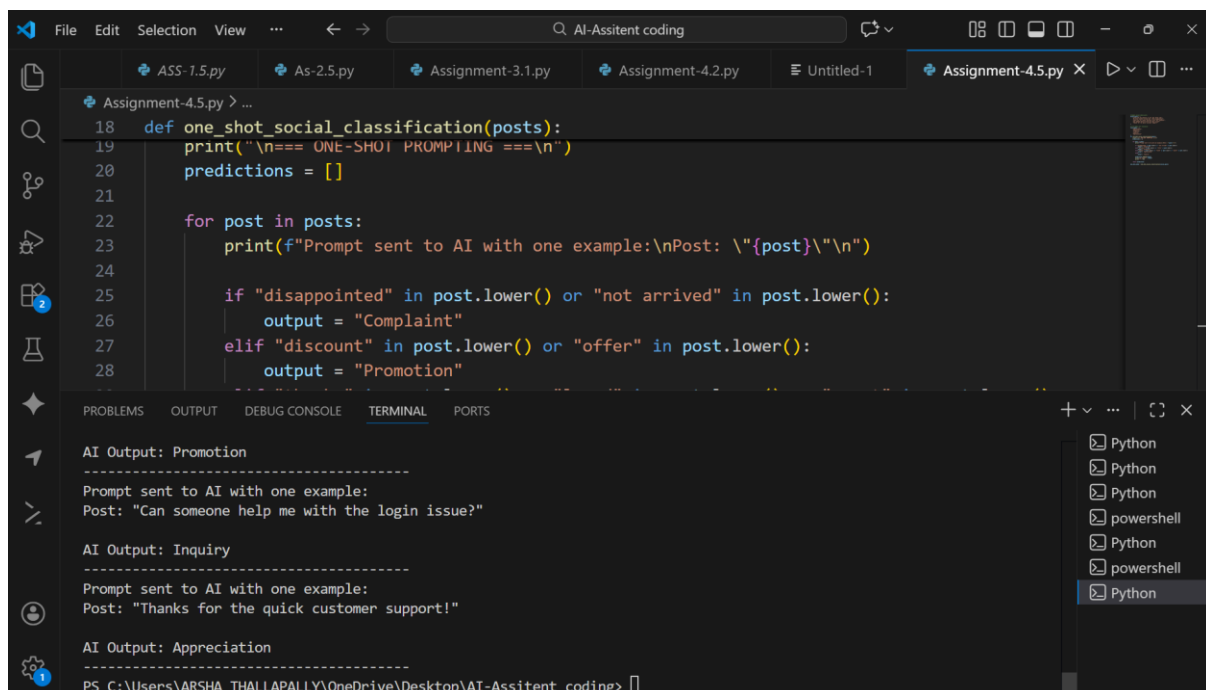
Post: "My order is late."

Category: Complaint

Now classify the following social media post:

Post: "<social_post>"

CODE:



```
18 def one_shot_social_classification(posts):
19     print("\n=== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI with one example:\nPost: \"{post}\"")
24
25         if "disappointed" in post.lower() or "not arrived" in post.lower():
26             output = "Complaint"
27         elif "discount" in post.lower() or "offer" in post.lower():
28             output = "Promotion"
```

AI Output: Promotion

Prompt sent to AI with one example:
Post: "Can someone help me with the login issue?"

AI Output: Inquiry

Prompt sent to AI with one example:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides one example to guide AI reasoning.

Improves accuracy and handles some informal expressions

better.

Depends on how representative the example is for informal language.

FEW-SHOT

PROMPT:

Examples:

Post: "Loved the new feature!" → Appreciation

Post: "My order hasn't arrived." → Complaint

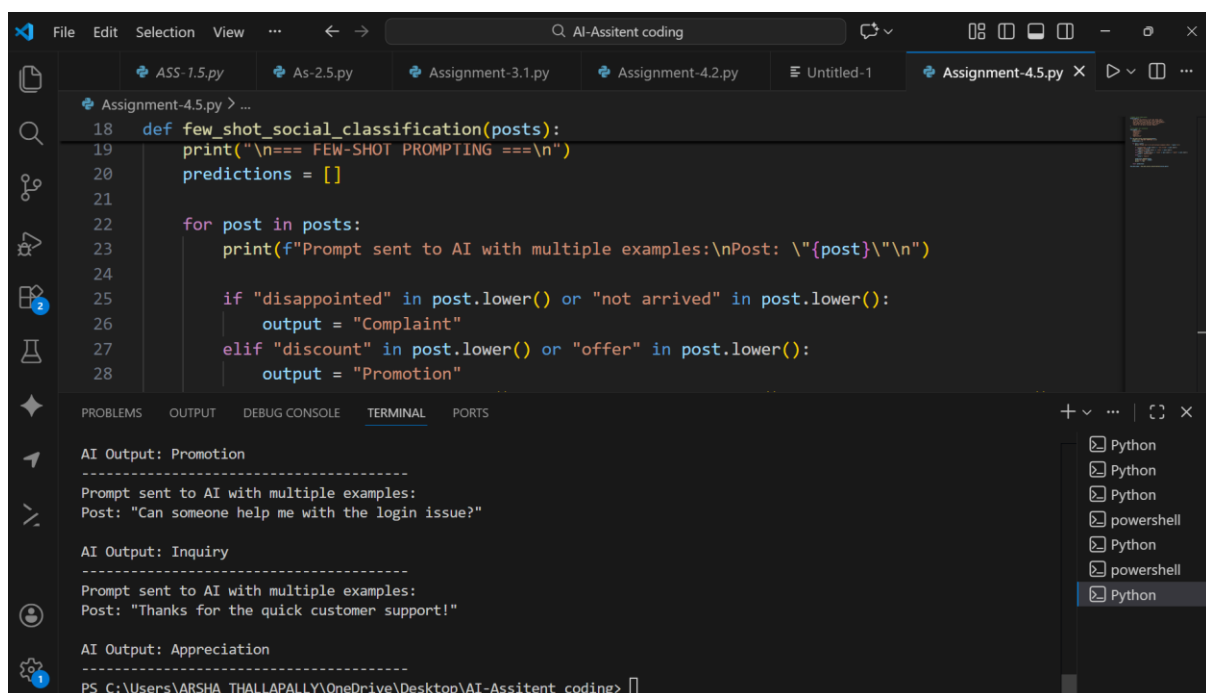
Post: "Check out our discount offer!" → Promotion

Post: "How can I reset my password?" → Inquiry

Now classify the following social media post:

Post: "<social_post>"

CODE:



```
File Edit Selection View ... AI-Assitent coding
ASS-1.5.py As-2.5.py Assignment-3.1.py Assignment-4.2.py Untitled-1 Assignment-4.5.py x
Assignment-4.5.py > ...
18 def few_shot_social_classification(posts):
19     print("\n=== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI with multiple examples:\nPost: \"{post}\"")
24
25         if "disappointed" in post.lower() or "not arrived" in post.lower():
26             output = "Complaint"
27         elif "discount" in post.lower() or "offer" in post.lower():
28             output = "Promotion"

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AI Output: Promotion

Prompt sent to AI with multiple examples:
Post: "Can someone help me with the login issue?"

AI Output: Inquiry

Prompt sent to AI with multiple examples:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; better handles informal, slang, or mixed-language posts.

Slightly longer prompts but most reliable for social media classification.