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:

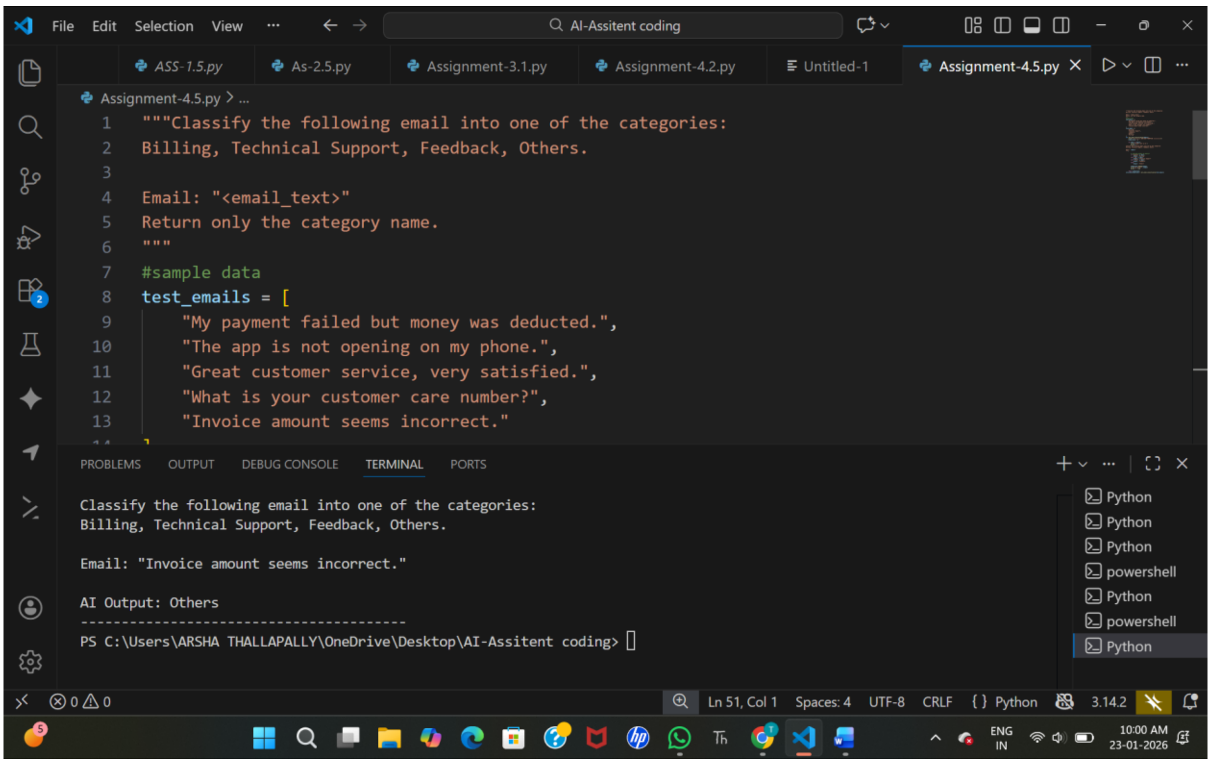
Classify the following email into one of the categories:

Billing, Technical Support, Feedback, Others.

Email: "<email_text>"

Return only the category name.

CODE:



```
"""Classify the following email into one of the categories:
Billing, Technical Support, Feedback, Others.

Email: "<email_text>"
Return only the category name.
"""
#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."
```

```
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> 
```

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:

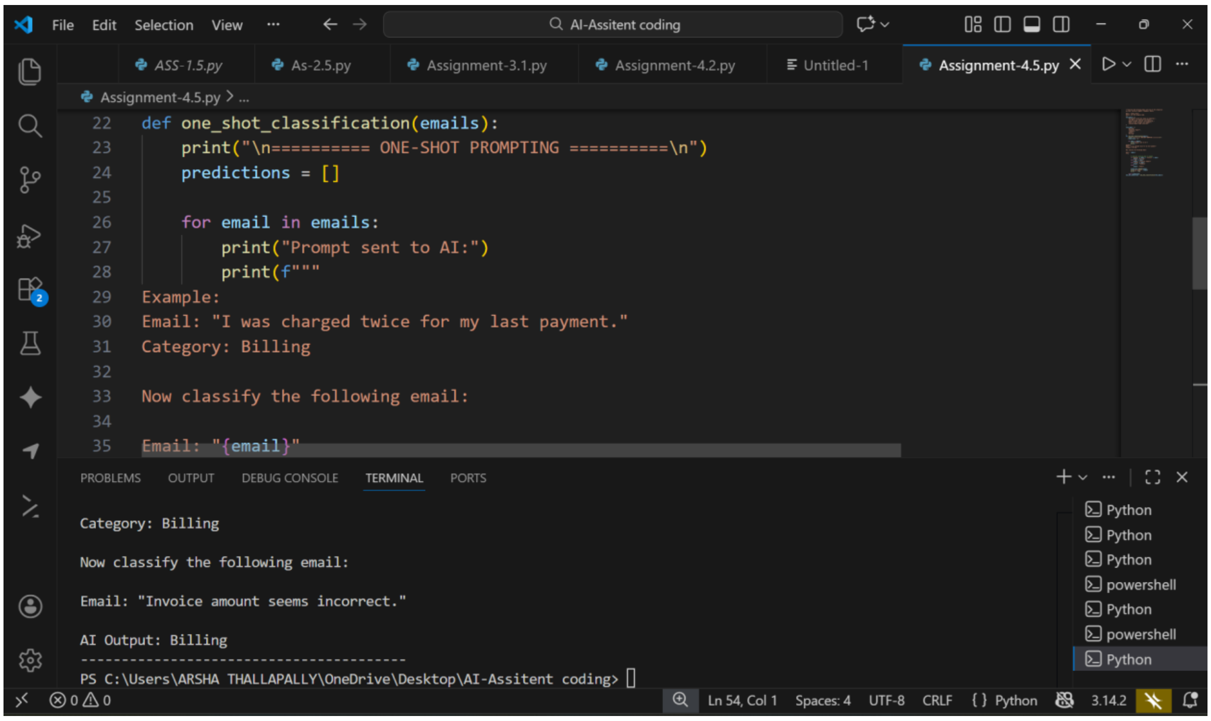
Email: "I was charged twice for my last payment."

Category: Billing

Now classify the following email:

Email: "<email_text>"

CODE:



```python
def one_shot_classification(emails):
    print("\n========== ONE-SHOT PROMPTING ==========\n")
    predictions = []

    for email in emails:
        print("Prompt sent to AI:")
        print(f"""
Example:
Email: "I was charged twice for my last payment."
Category: Billing

Now classify the following email:

Email: "{email}"
```

```
Category: Billing

Now classify the following email:

Email: "Invoice amount seems incorrect."

AI Output: Billing
----------------------------------------
PS C:\Users\ARSHA 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:

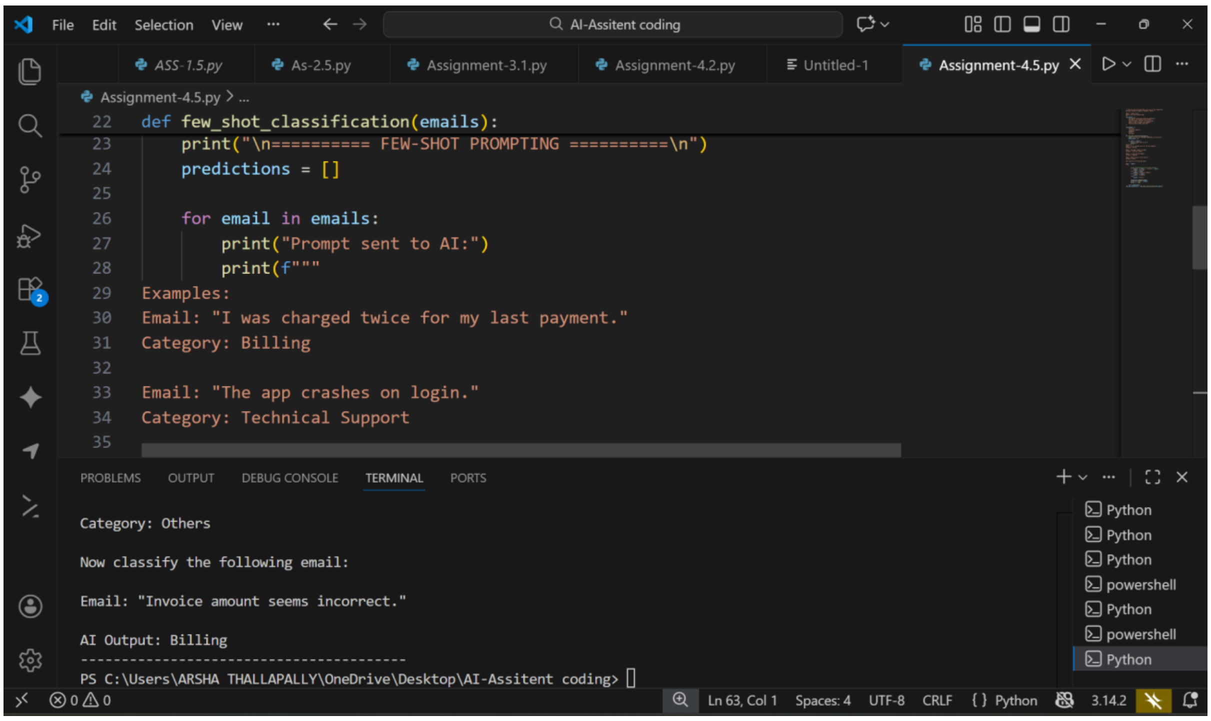
Email: "I was charged twice for my last payment." → Billing

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

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

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

CODE:



```
def few_shot_classification(emails):
    print("\n========== FEW-SHOT PROMPTING ==========\n")
    predictions = []

    for email in emails:
        print("Prompt sent to AI:")
        print(f"""
Examples:
Email: "I was charged twice for my last payment."
Category: Billing

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

```

```
Category: Others

Now classify the following email:

Email: "Invoice amount seems incorrect."

AI Output: Billing
----------------------------------------
PS C:\Users\ARSHA 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"
]
```

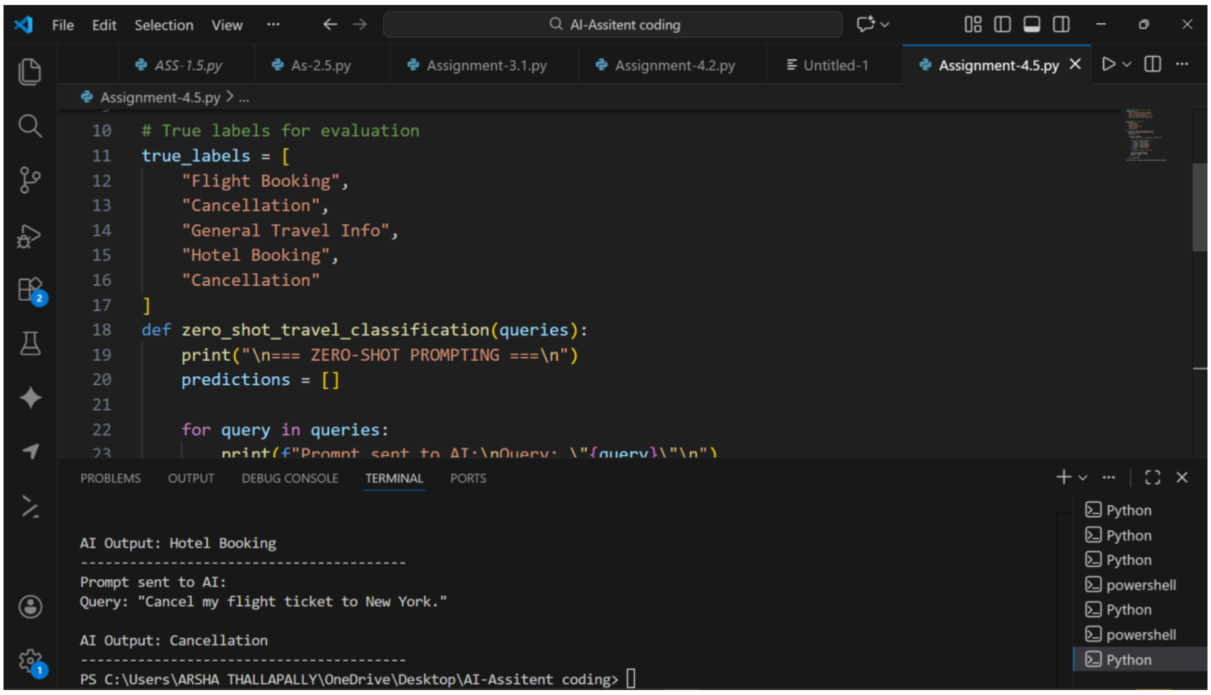
ZERO-SHOT:

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

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

Query: "<travel_query>"

CODE:



```python
# True labels for evaluation
true_labels = [
    "Flight Booking",
    "Cancellation",
    "General Travel Info",
    "Hotel Booking",
    "Cancellation"
]
def zero_shot_travel_classification(queries):
    print("\n=== ZERO-SHOT PROMPTING ===\n")
    predictions = []

    for query in queries:
        print(f"Prompt sent to AI:\nQuery: \"{query}\"\n")
```

```
AI Output: Hotel Booking
----------------------------------------
Prompt sent to AI:
Query: "Cancel my flight ticket to New York."

AI Output: Cancellation
----------------------------------------
PS C:\Users\ARSHA 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:

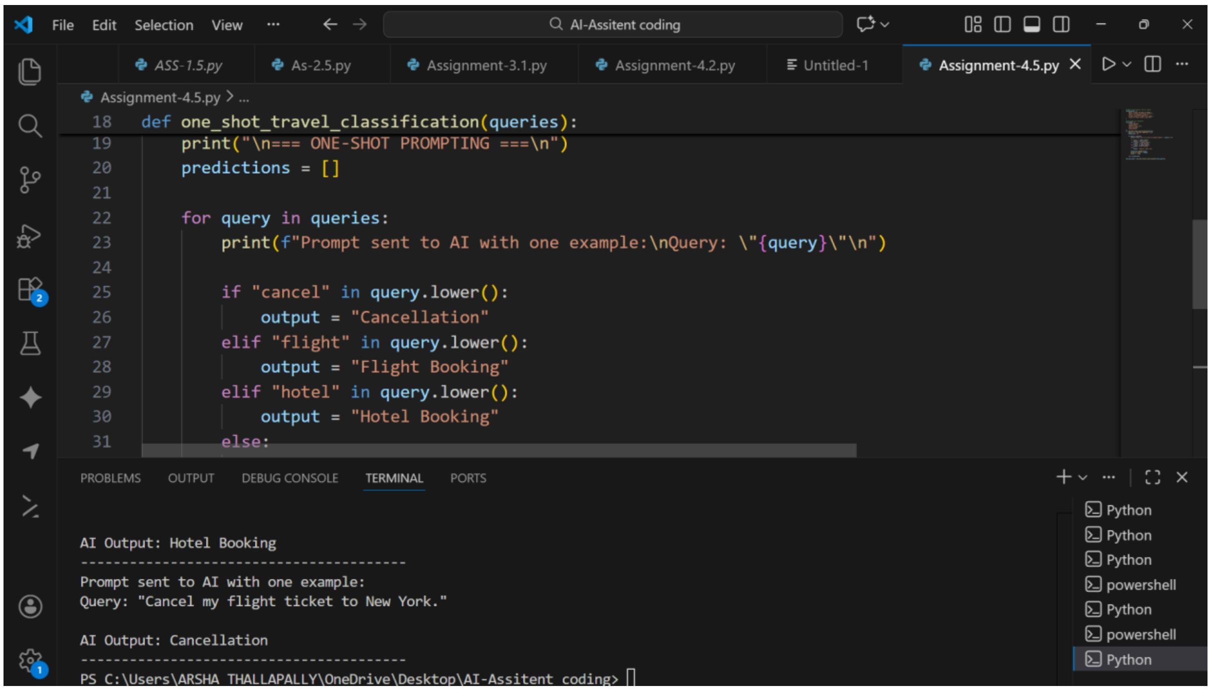
Query: "Cancel my flight ticket."

Category: Cancellation

Now classify the following query:

Query: "<travel_query>"

CODE:



```python
def one_shot_travel_classification(queries):
    print("\n=== ONE-SHOT PROMPTING ===\n")
    predictions = []

    for query in queries:
        print(f"Prompt sent to AI with one example:\nQuery: \"{query}\"\n")

        if "cancel" in query.lower():
            output = "Cancellation"
        elif "flight" in query.lower():
            output = "Flight Booking"
        elif "hotel" in query.lower():
            output = "Hotel Booking"
        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\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 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

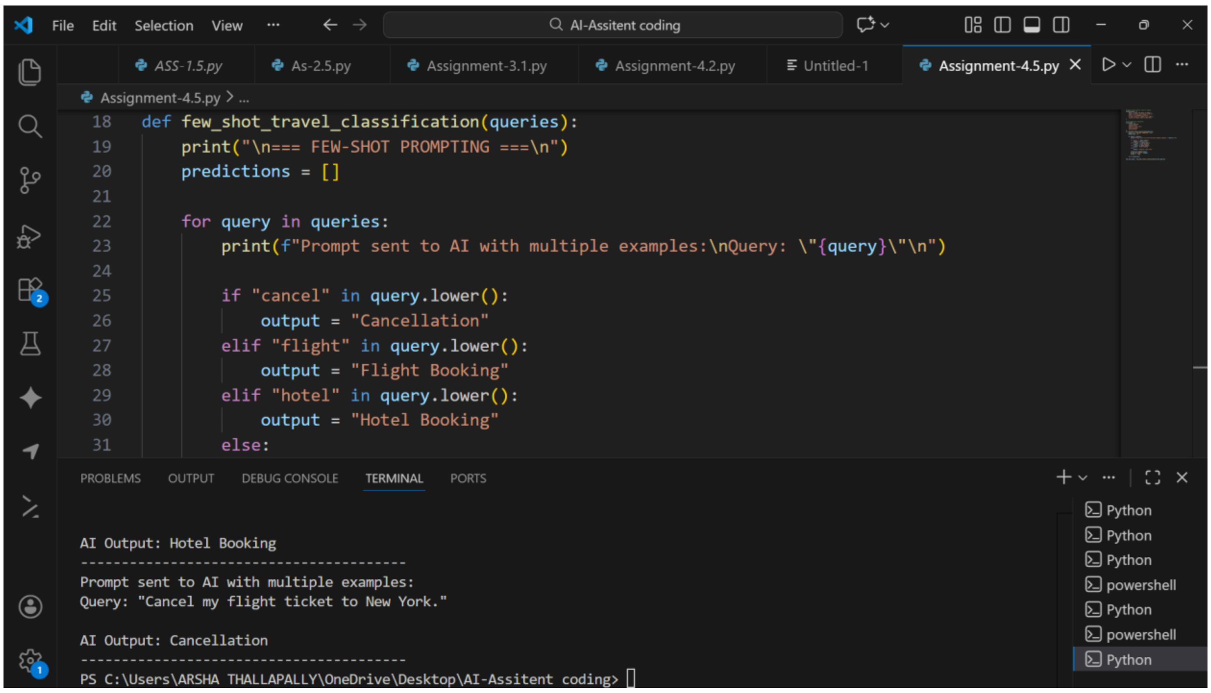
Query: "What is the baggage allowance?"

Category: General Travel Info

Now classify the following query:

Query: "<travel_query>"

CODE:



```python
def few_shot_travel_classification(queries):
    print("\n=== FEW-SHOT PROMPTING ===\n")
    predictions = []

    for query in queries:
        print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"\n")

        if "cancel" in query.lower():
            output = "Cancellation"
        elif "flight" in query.lower():
            output = "Flight Booking"
        elif "hotel" in query.lower():
            output = "Hotel Booking"
        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\ARSHA 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"

]
```

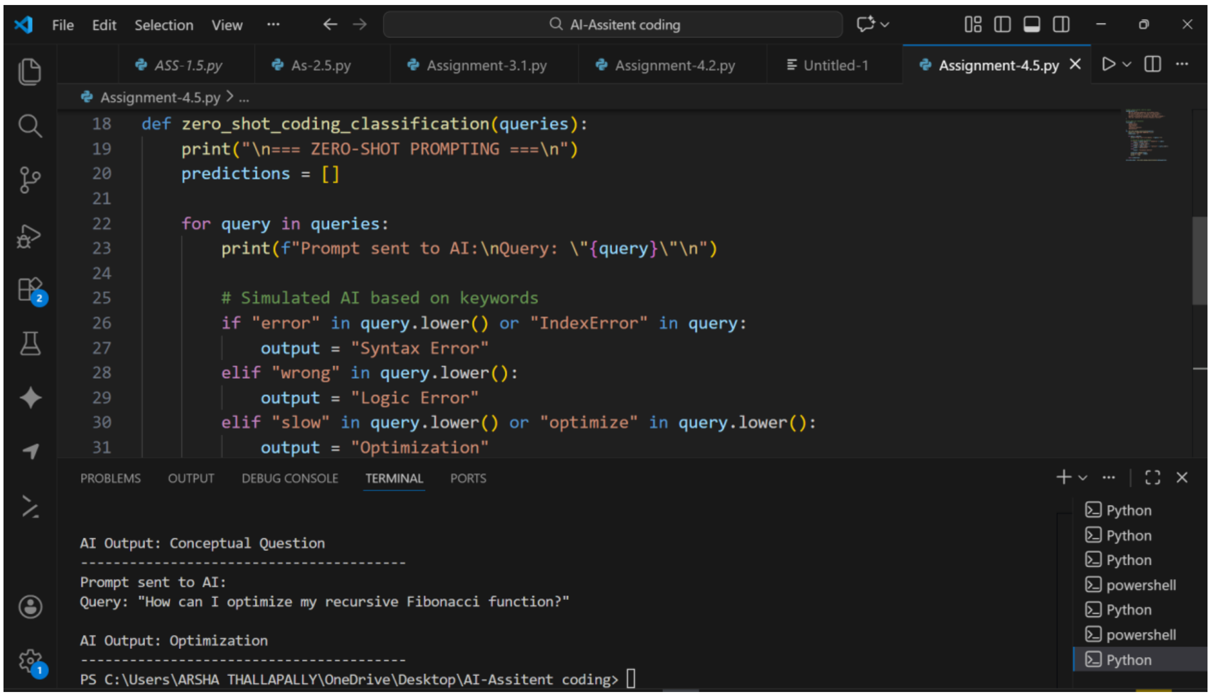
ZERO-SHOT

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

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "<coding_query>"

CODE:



```python
def zero_shot_coding_classification(queries):
    print("\n=== ZERO-SHOT PROMPTING ===\n")
    predictions = []

    for query in queries:
        print(f"Prompt sent to AI:\nQuery: \"{query}\"\n")

        # Simulated AI based on keywords
        if "error" in query.lower() or "IndexError" in query:
            output = "Syntax Error"
        elif "wrong" in query.lower():
            output = "Logic Error"
        elif "slow" in query.lower() or "optimize" in query.lower():
            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\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

ONE SHOT

PROMPT:

Example:

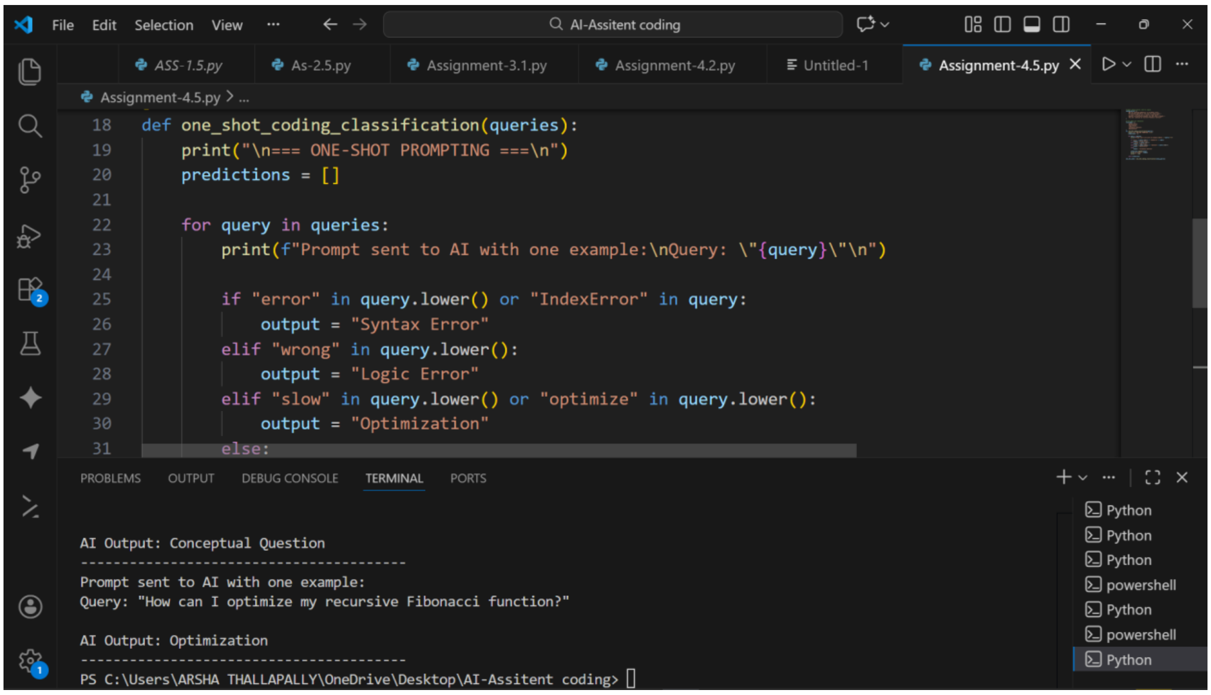
Query: "I want to cancel my Python function."

Category: Logic Error

Now classify the following coding query:

Query: "<coding_query>"

CODE:



```
def one_shot_coding_classification(queries):
    print("\n=== ONE-SHOT PROMPTING ===\n")
    predictions = []

    for query in queries:
        print(f"Prompt sent to AI with one example:\nQuery: \"{query}\"\n")

        if "error" in query.lower() or "IndexError" in query:
            output = "Syntax Error"
        elif "wrong" in query.lower():
            output = "Logic Error"
        elif "slow" in query.lower() or "optimize" in query.lower():
            output = "Optimization"
        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

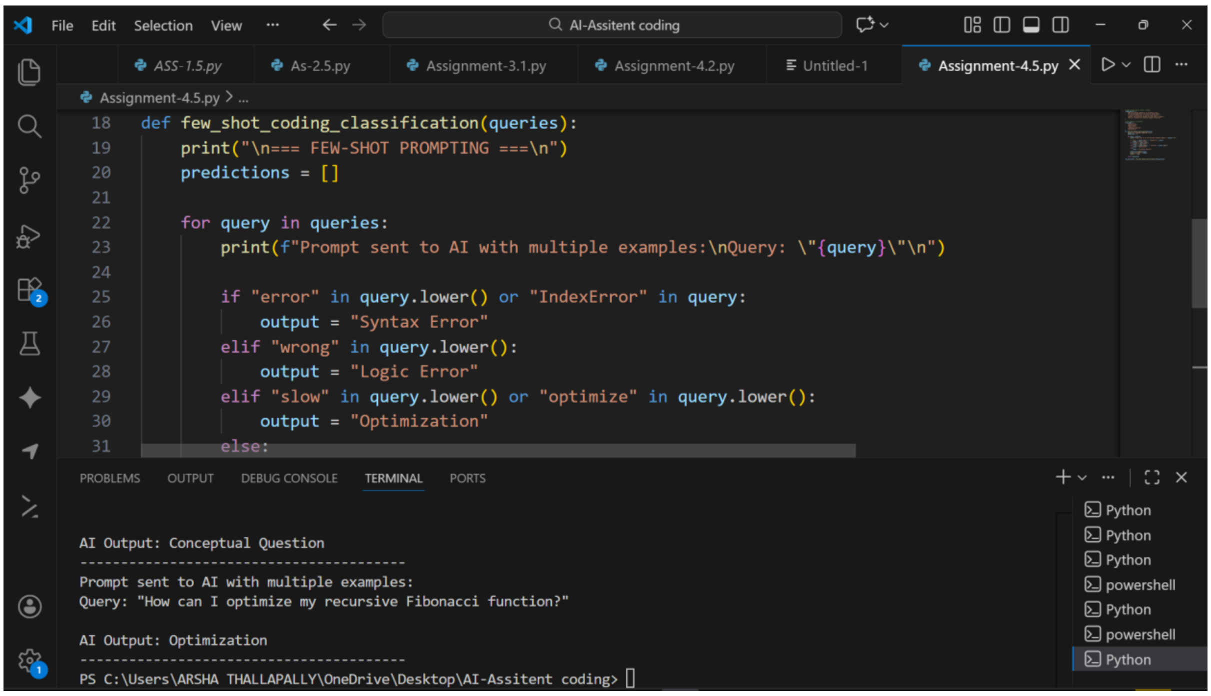
Query: "Explain Python variable scopes."

Category: Conceptual Question

Now classify the following coding query:

Query: "<coding_query>"

CODE:



```python
def few_shot_coding_classification(queries):
    print("\n=== FEW-SHOT PROMPTING ===\n")
    predictions = []

    for query in queries:
        print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"\n")

        if "error" in query.lower() or "IndexError" in query:
            output = "Syntax Error"
        elif "wrong" in query.lower():
            output = "Logic Error"
        elif "slow" in query.lower() or "optimize" in query.lower():
            output = "Optimization"
        else:
```

```
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> 
```

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

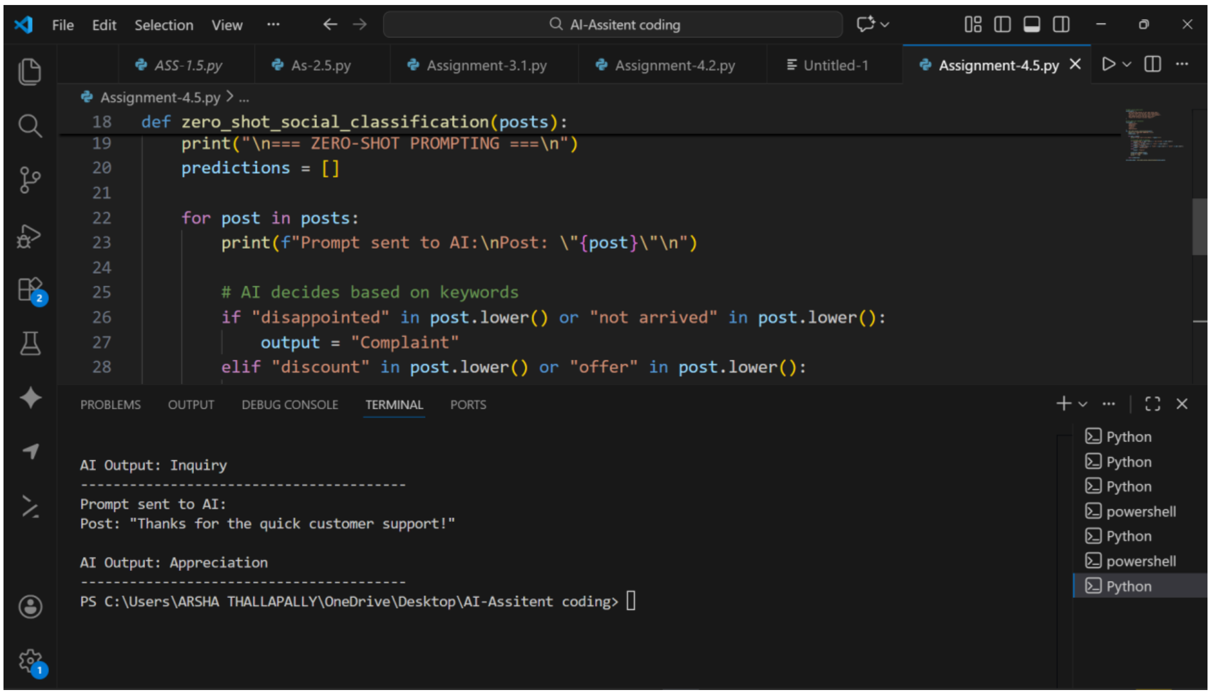
PROMPT:

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

Promotion, Complaint, Appreciation, Inquiry.

Post: "<social_post>"

CODE:



```python
def zero_shot_social_classification(posts):
    print("\n=== ZERO-SHOT PROMPTING ===\n")
    predictions = []

    for post in posts:
        print(f"Prompt sent to AI:\nPost: \"{post}\"\n")

        # AI decides based on keywords
        if "disappointed" in post.lower() or "not arrived" in post.lower():
            output = "Complaint"
        elif "discount" in post.lower() or "offer" in post.lower():
```

```
AI Output: Inquiry
----------------------------------------
Prompt sent to AI:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation
----------------------------------------
PS C:\Users\ARSHA 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:

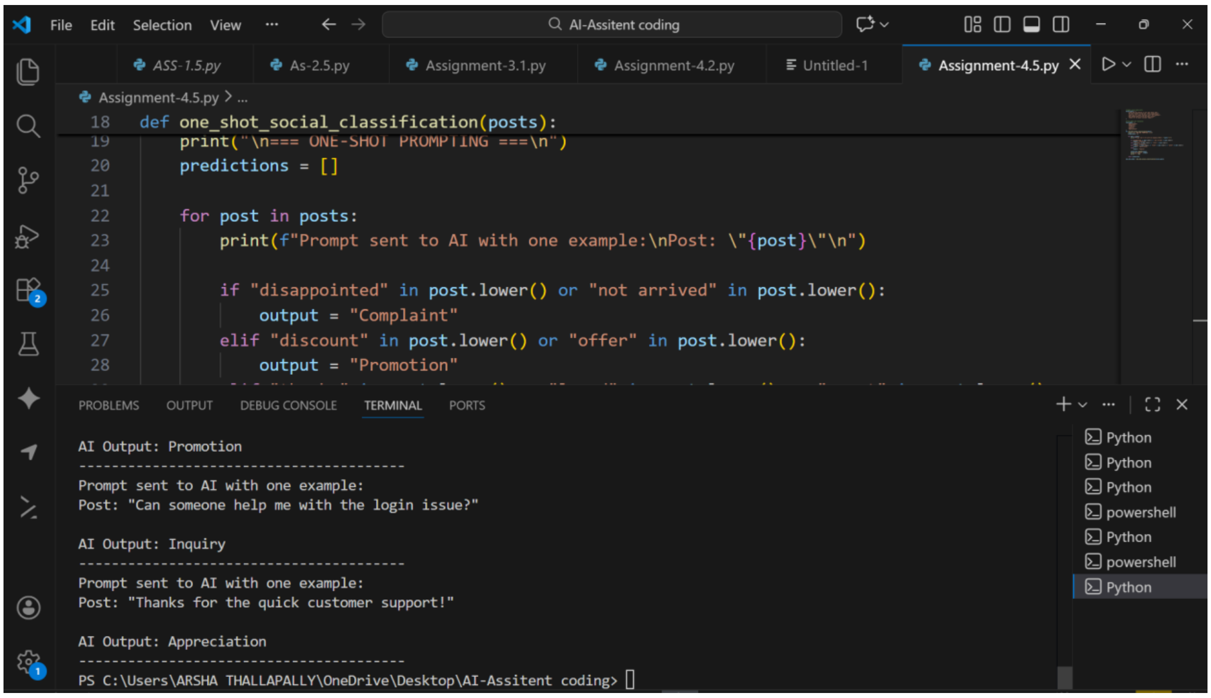
Post: "My order is late."

Category: Complaint

Now classify the following social media post:

Post: "<social_post>"

CODE:



```python
def one_shot_social_classification(posts):
    print("\n=== ONE-SHOT PROMPTING ===\n")
    predictions = []

    for post in posts:
        print(f"Prompt sent to AI with one example:\nPost: \"{post}\"\n")

        if "disappointed" in post.lower() or "not arrived" in post.lower():
            output = "Complaint"
        elif "discount" in post.lower() or "offer" in post.lower():
            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

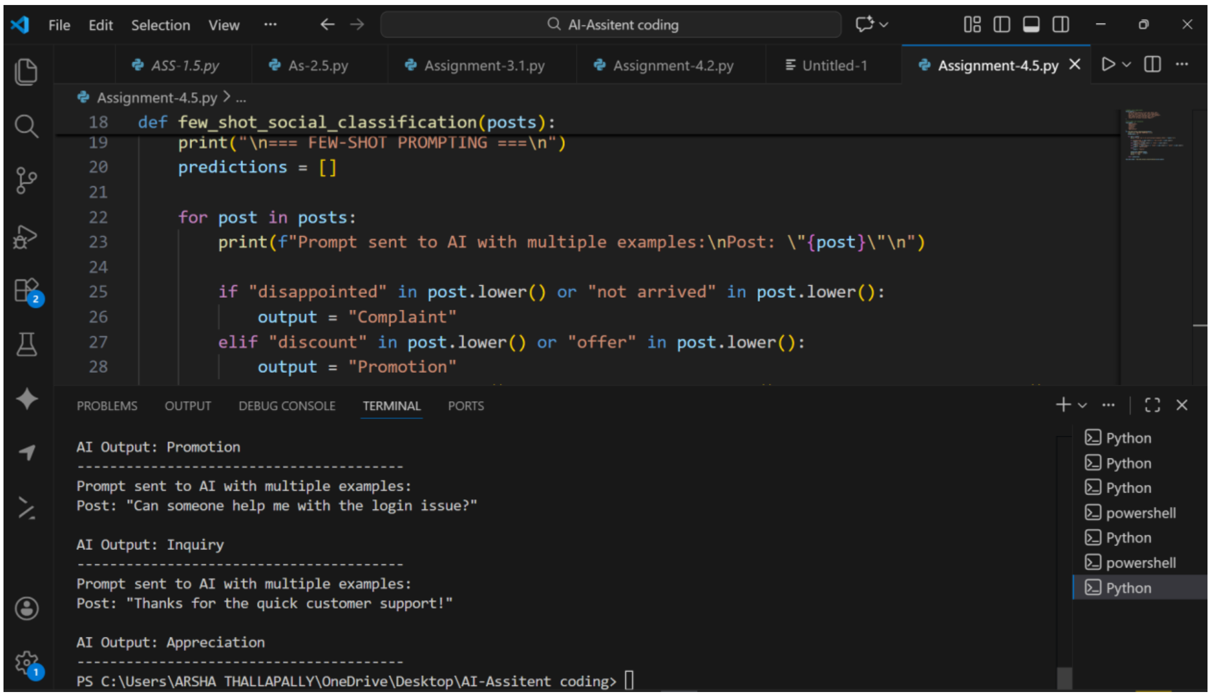
Post: "Check out our discount offer!" → Promotion

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

Now classify the following social media post:

Post: "<social_post>"

CODE:



```python
def few_shot_social_classification(posts):
    print("\n=== FEW-SHOT PROMPTING ===\n")
    predictions = []

    for post in posts:
        print(f"Prompt sent to AI with multiple examples:\nPost: \"{post}\"\n")

        if "disappointed" in post.lower() or "not arrived" in post.lower():
            output = "Complaint"
        elif "discount" in post.lower() or "offer" in post.lower():
            output = "Promotion"
```

```
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.