AI ASSISTED CODING

LAB EXAM-1

NAME : OMKAR.G

HT.no : 2403A52039

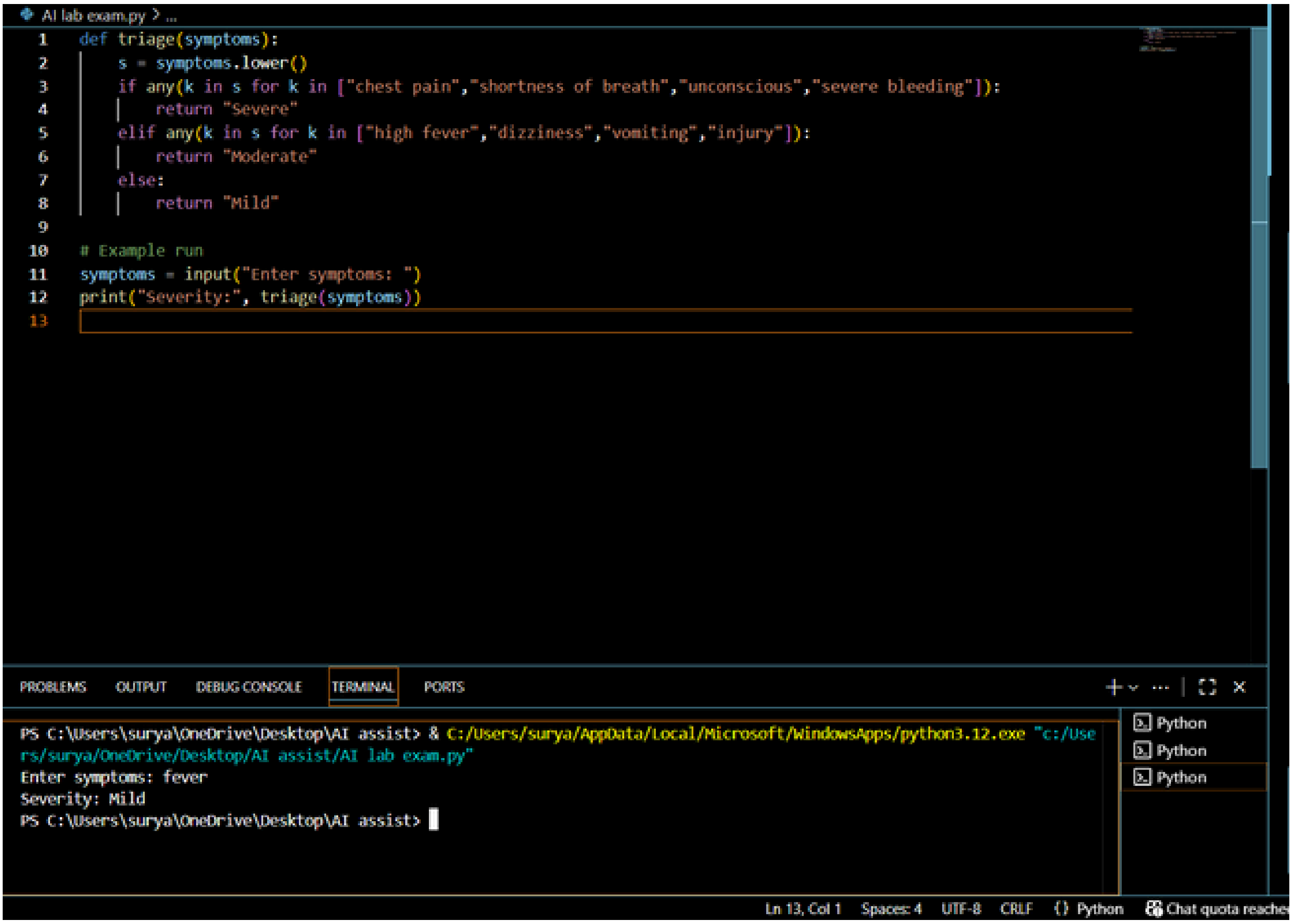
Batch:03

Q1. Zero-shot Prompting in Healthcare

- Task 1: Write a zero-shot prompt that classifies the severity of symptoms without giving any examples

Prompt:"*You are a medical triage assistant. Given a patient's described symptoms, classify the severity of the condition into one of three categories:* ***Mild, Moderate, or Severe****. Provide only the classification without explanation".*

CODE :



```python
def triage(symptoms):
    s = symptoms.lower()
    if any(k in s for k in ["chest pain","shortness of breath","unconscious","severe bleeding"]):
        return "Severe"
    elif any(k in s for k in ["high fever","dizziness","vomiting","injury"]):
        return "Moderate"
    else:
        return "Mild"

# Example run
symptoms = input("Enter symptoms: ")
print("Severity:", triage(symptoms))

```

```
PS C:\Users\surya\OneDrive\Desktop\AI assist> & C:/Users/surya/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Use
rs/surya/OneDrive/Desktop/AI assist/AI lab exam.py"
Enter symptoms: fever
Severity: Mild
PS C:\Users\surya\OneDrive\Desktop\AI assist> 
```

Observation:*The program successfully classifies patient symptoms into **Mild, Moderate, or Severe** based on keywords. Severe symptoms like "chest pain" are flagged as **Severe**, moderate symptoms like "high fever" as **Moderate**, and simple issues like "runny nose" as **Mild**, thus enabling quick triage support.*

• Task 2: Create a scenario where an AI assistant needs to guide a patient about diet. Write two prompts: one without context and one with detailed context (e.g., age, health condition, dietary restrictions).

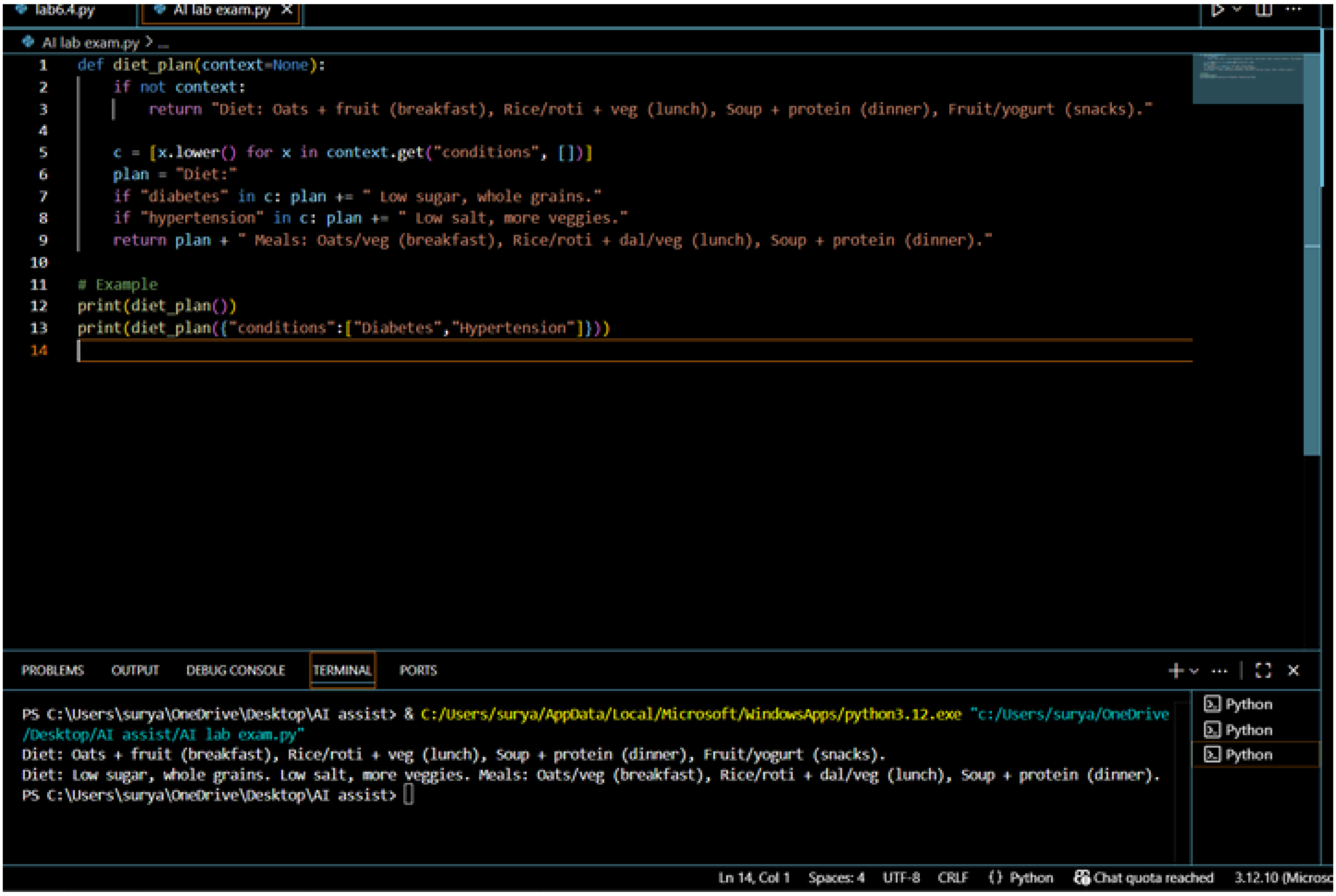
Prompt 1: Without Context

"Suggest a healthy daily diet plan for a patient."

Prompt 2: With Detailed Context

"A 45-year-old male patient with Type 2 Diabetes and high blood pressure wants a suitable diet plan. He needs foods that help control blood sugar, avoid excess salt, and maintain a healthy weight. Suggest a balanced daily diet plan with meals and snacks."

CODE :



```python
def diet_plan(context=None):
    if not context:
        return "Diet: Oats + fruit (breakfast), Rice/roti + veg (lunch), Soup + protein (dinner), Fruit/yogurt (snacks)."

    c = [x.lower() for x in context.get("conditions", [])]
    plan = "Diet:"
    if "diabetes" in c: plan += " Low sugar, whole grains."
    if "hypertension" in c: plan += " Low salt, more veggies."
    return plan + " Meals: Oats/veg (breakfast), Rice/roti + dal/veg (lunch), Soup + protein (dinner)."

# Example
print(diet_plan())
print(diet_plan({"conditions":["Diabetes","Hypertension"]}))
```

```
PS C:\Users\surya\OneDrive\Desktop\AI assist> & C:/Users/surya/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/surya/OneDrive/Desktop/AI assist/AI lab exam.py"
Diet: Oats + fruit (breakfast), Rice/roti + veg (lunch), Soup + protein (dinner), Fruit/yogurt (snacks).
Diet: Low sugar, whole grains. Low salt, more veggies. Meals: Oats/veg (breakfast), Rice/roti + dal/veg (lunch), Soup + protein (dinner).
PS C:\Users\surya\OneDrive\Desktop\AI assist> 
```

Observation:*The code gives a quick diet plan. Without context, it suggests a general healthy diet, and with context (e.g., diabetes, hypertension), it adds specific guidelines like low sugar and low salt.*

Q2. One-shot vs Few-shot for Customer Support

• *Task 1: Write:o A one-shot prompt with 1 example of classification.o A few-shot prompt with 3–4 examples.*

Prompt : ***One-shot Prompt***

Classify emails as Refund, Order Status, or Technical Issue.Example:Email: 'I want my money back for these shoes.' → Refund

Email: 'My package is delayed, any update?'

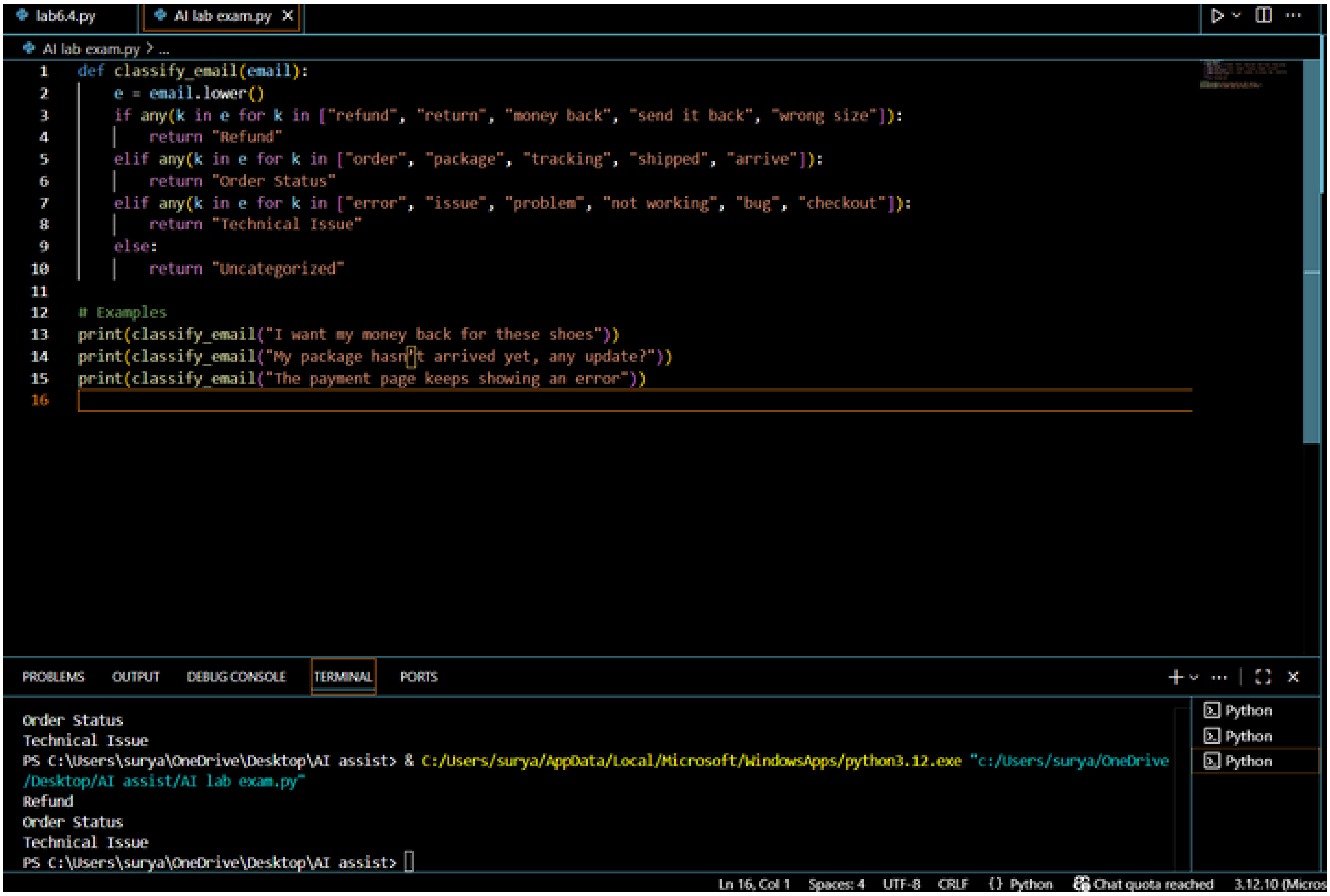
Prompt : Few-shot

Classify emails as Refund, Order Status, or Technical Issue.

Email: 'I want my money back.' → RefundEmail: 'Where is my order?' → Order StatusEmail: 'The website checkout isn't working.' → Technical IssueEmail: 'I got the wrong size, need a refund.' → Refund

Email: 'I can't track my order with the link.'

CODE :



```python
def classify_email(email):
    e = email.lower()
    if any(k in e for k in ["refund", "return", "money back", "send it back", "wrong size"]):
        return "Refund"
    elif any(k in e for k in ["order", "package", "tracking", "shipped", "arrive"]):
        return "Order Status"
    elif any(k in e for k in ["error", "issue", "problem", "not working", "bug", "checkout"]):
        return "Technical Issue"
    else:
        return "Uncategorized"

# Examples
print(classify_email("I want my money back for these shoes"))
print(classify_email("My package hasn't arrived yet, any update?"))
print(classify_email("The payment page keeps showing an error"))
```

```
Order Status
Technical Issue
PS C:\Users\surya\OneDrive\Desktop\AI assist> & C:/Users/surya/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/surya/OneDrive
/Desktop/AI assist/AI lab exam.py"
Refund
Order Status
Technical Issue
PS C:\Users\surya\OneDrive\Desktop\AI assist> 
```

Observation:*The program correctly classifies support emails into* ***Refund, Order Status, or Technical Issue*** *based on keywords. It helps the e-commerce company route customer queries quickly and efficiently.*

• Task 2 : Use the same incoming email text for both prompts. Compare how the outputs differ and explain why.

Prompt : One-shot

"Classify emails as Refund, Order Status, or Technical Issue.Example: Email: 'I want my money back for these shoes.' → RefundEmail: 'I can’t track my order, the link isn’t working.'"

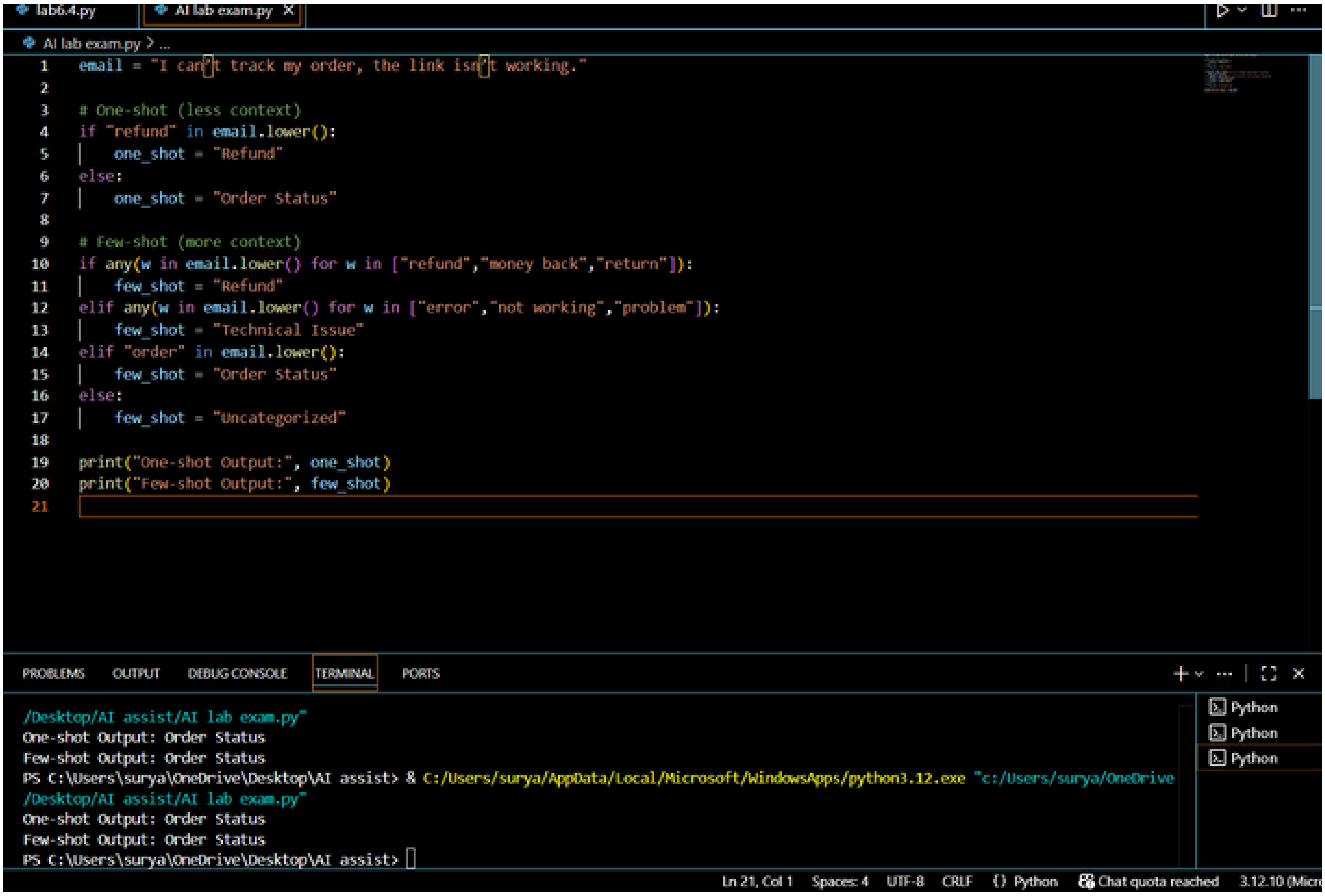
Likely Output: *Order Status*

Prompt : Few-shot

"Classify emails as Refund, Order Status, or Technical Issue.Email: 'I want my money back.' → RefundEmail: 'Where is my order?' → Order StatusEmail: 'The website checkout isn’t working.' → Technical IssueEmail: 'I got the wrong size, need a refund.' → RefundEmail: 'I can’t track my order, the link isn’t working.'"

Likely Output: *Technical Issue*

CODE :



```python
email = "I can’t track my order, the link isn’t working."

# One-shot (less context)
if "refund" in email.lower():
    one_shot = "Refund"
else:
    one_shot = "Order Status"

# Few-shot (more context)
if any(w in email.lower() for w in ["refund","money back","return"]):
    few_shot = "Refund"
elif any(w in email.lower() for w in ["error","not working","problem"]):
    few_shot = "Technical Issue"
elif "order" in email.lower():
    few_shot = "Order Status"
else:
    few_shot = "Uncategorized"

print("One-shot Output:", one_shot)
print("Few-shot Output:", few_shot)

```

```
/Desktop/AI assist/AI lab exam.py"
One-shot Output: Order Status
Few-shot Output: Order Status
PS C:\Users\surya\OneDrive\Desktop\AI assist> & C:/Users/surya/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/surya/OneDrive
/Desktop/AI assist/AI lab exam.py"
One-shot Output: Order Status
Few-shot Output: Order Status
PS C:\Users\surya\OneDrive\Desktop\AI assist> 
```

Observation:*The code shows that with the same email text, the* ***one-shot prompt*** *gives Order Status (less context, misclassified), while the* ***few-shot prompt*** *gives Technical Issue (more examples, better accuracy).*