

# AI ASSISTED CODING

## LAB EXAM-1

NAME : SD.Nayaz HT.no : 2403A52076 BATCH : 04

+

### Q1. Zero-shot Prompting in Healthcare

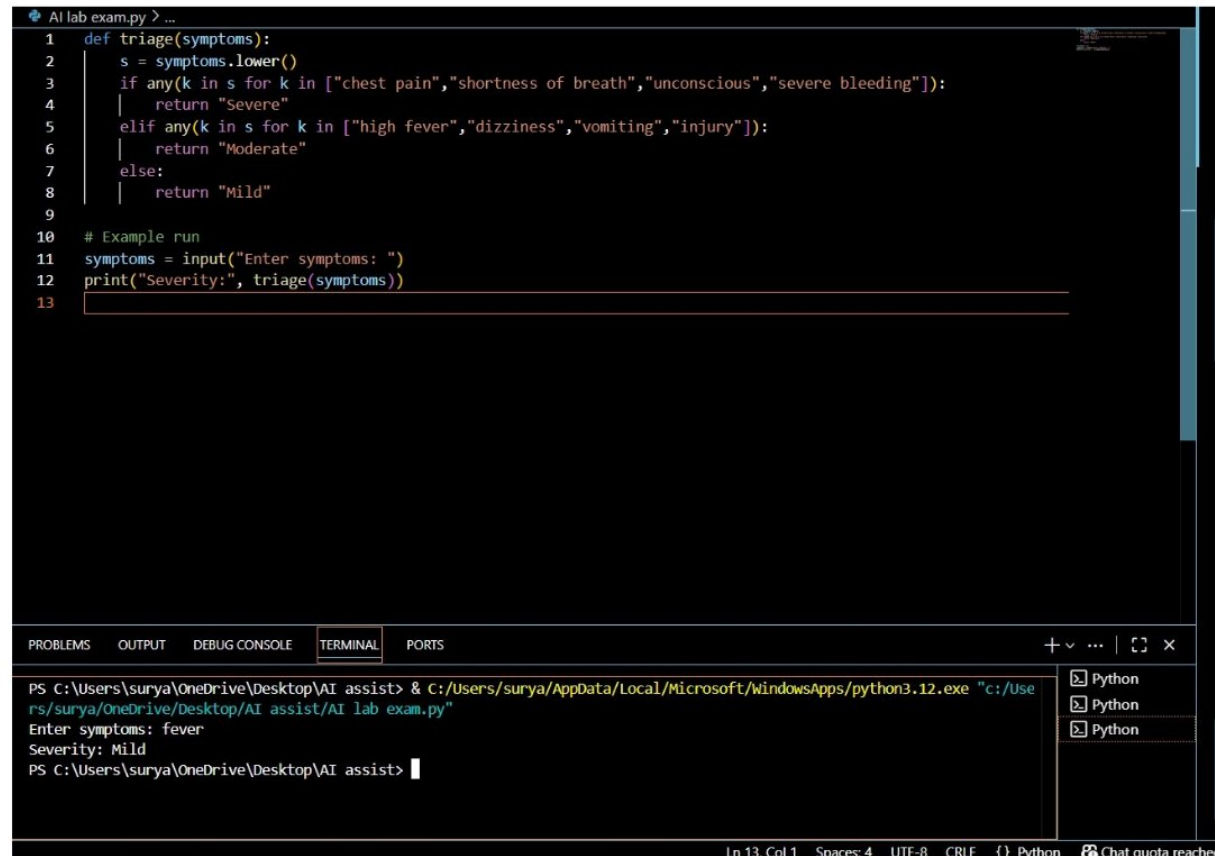
• **Task 1:** Write a zero-shot prompt that classifies the severity of symptoms without giving any examples

#### 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 :



```
AI lab exam.py > ...
1 def triage(symptoms):
2     s = symptoms.lower()
3     if any(k in s for k in ["chest pain", "shortness of breath", "unconscious", "severe bleeding"]):
4         return "Severe"
5     elif any(k in s for k in ["high fever", "dizziness", "vomiting", "injury"]):
6         return "Moderate"
7     else:
8         return "Mild"
9
10 # Example run
11 symptoms = input("Enter symptoms: ")
12 print("Severity:", triage(symptoms))
13
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS + v ... | [ ] x

```
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>
```

Ln 13, Col 1 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached

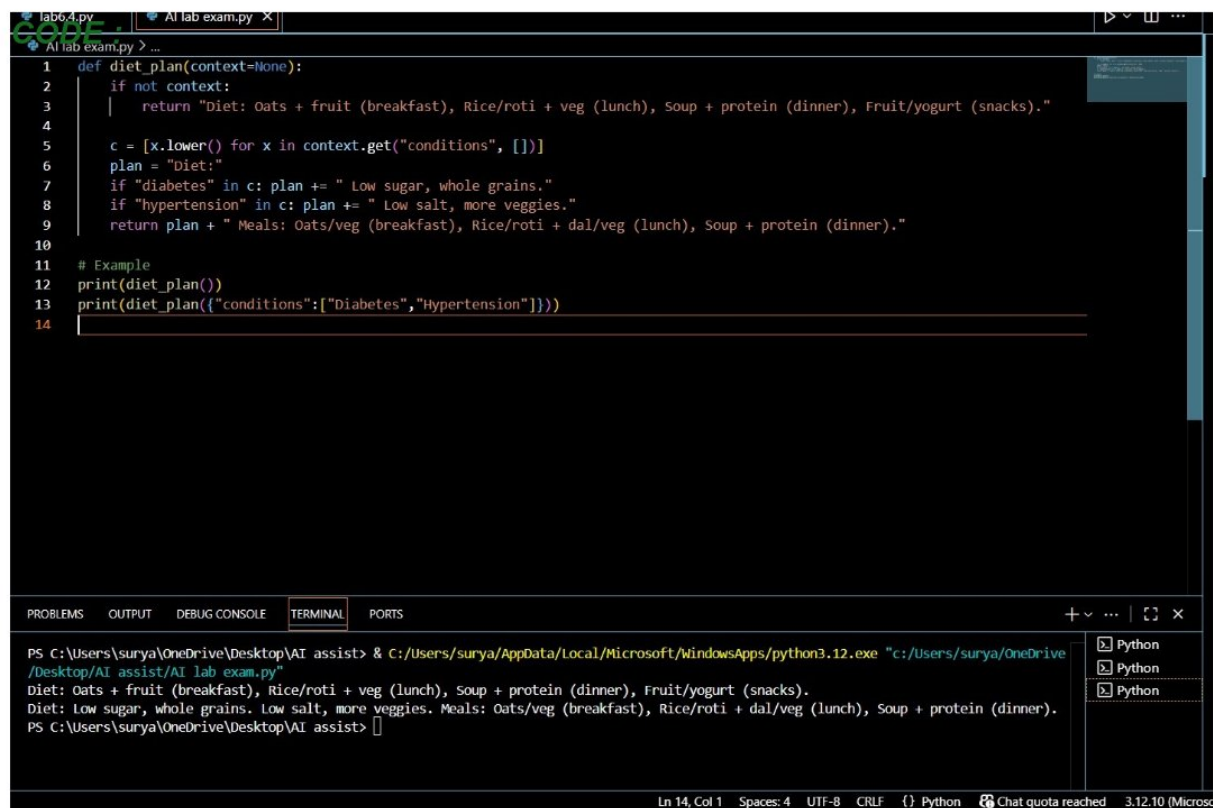
**Observation:** The program successfully classifies patient symptoms into Mild, Moderate, or Severe based on keywords. Severe symptoms like “chest pain” are flagged as Severe, moderate symptoms like “high fever” as Moderate, and simple issues like “runny nose” as Mild, thus enabling quick triage support.

•**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."



```
lab04.py AI lab exam.py X
AI lab exam.py > ...
1 def diet_plan(context=None):
2     if not context:
3         return "Diet: Oats + fruit (breakfast), Rice/roti + veg (lunch), Soup + protein (dinner), Fruit/yogurt (snacks)."
```

```
4
5     c = [x.lower() for x in context.get("conditions", [])]
6     plan = "Diet:"
7     if "diabetes" in c: plan += " Low sugar, whole grains."
8     if "hypertension" in c: plan += " Low salt, more veggies."
9     return plan + " Meals: Oats/veg (breakfast), Rice/roti + dal/veg (lunch), Soup + protein (dinner)."
```

```
10
11 # Example
12 print(diet_plan())
13 print(diet_plan({"conditions":["Diabetes","Hypertension"]}))
14
```

```
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>
```

Ln 14, Col 1 Spaces: 4 UTF-8 CRLF Python Chat quota reached 3.12.10 (Micros

### **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:**

*oA one-shot prompt with 1 example of classification.*

*oA 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.' → Refund*

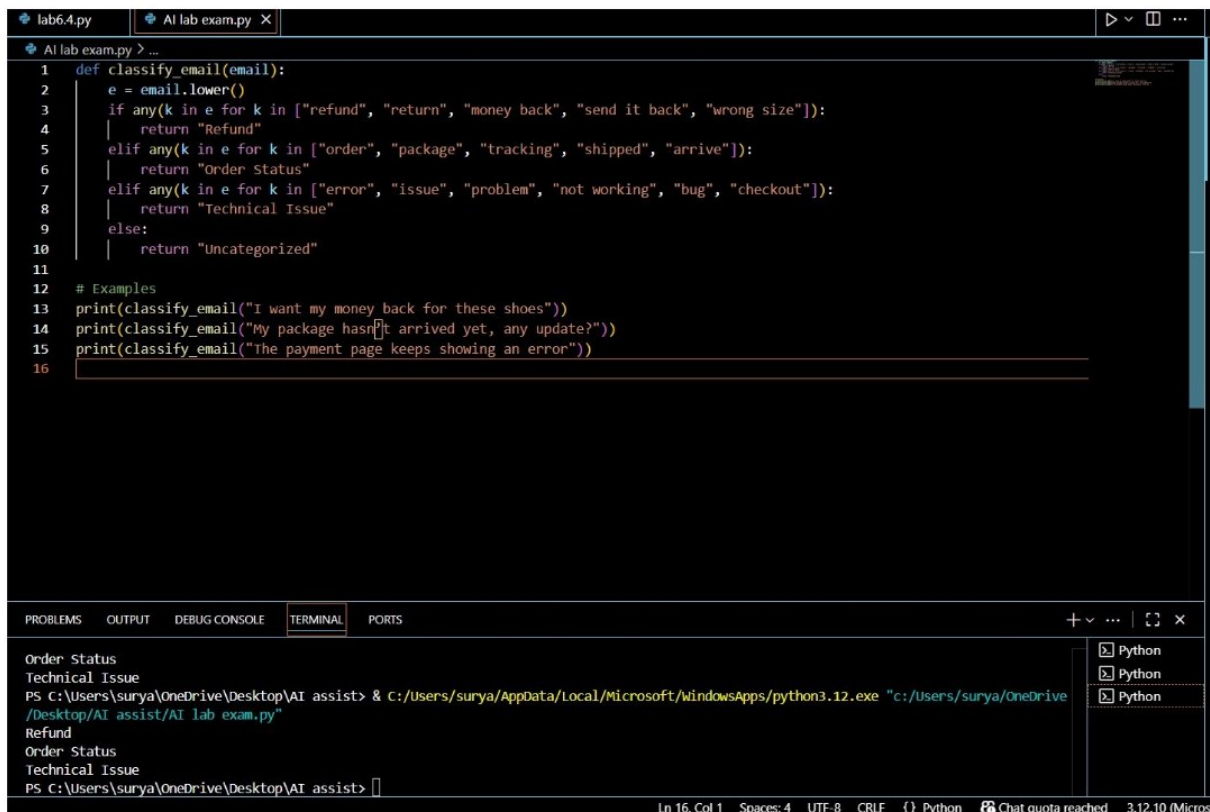
*Email: 'Where is my order?' → Order Status*

*Email: 'The website checkout isn't working.' → Technical Issue*

*Email: 'I got the wrong size, need a refund.' → Refund*

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

### **CODE :**



The screenshot shows a VS Code editor with a file named 'AI lab exam.py'. The code defines a function 'classify\_email(email)' that categorizes emails based on keywords. The keywords are grouped into three categories: 'Refund' (refund, return, money back, send it back, wrong size), 'Order Status' (order, package, tracking, shipped, arrive), and 'Technical Issue' (error, issue, problem, not working, bug, checkout). If no keywords are found, it returns 'Uncategorized'. Below the function, there are three example calls to the function with their corresponding outputs: 'I want my money back for these shoes' returns 'Refund', 'My package hasn't arrived yet, any update?' returns 'Order Status', and 'The payment page keeps showing an error' returns 'Technical Issue'.

```
1 def classify_email(email):
2     e = email.lower()
3     if any(k in e for k in ["refund", "return", "money back", "send it back", "wrong size"]):
4         return "Refund"
5     elif any(k in e for k in ["order", "package", "tracking", "shipped", "arrive"]):
6         return "Order Status"
7     elif any(k in e for k in ["error", "issue", "problem", "not working", "bug", "checkout"]):
8         return "Technical Issue"
9     else:
10        return "Uncategorized"
11
12 # Examples
13 print(classify_email("I want my money back for these shoes"))
14 print(classify_email("My package hasn't arrived yet, any update?"))
15 print(classify_email("The payment page keeps showing an error"))
16
```

The terminal output shows the results of running the script:

```
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.' → Refund

Email: '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.' → Refund

Email: 'Where is my order?' → Order Status

Email: 'The website checkout isn't working.' → Technical Issue

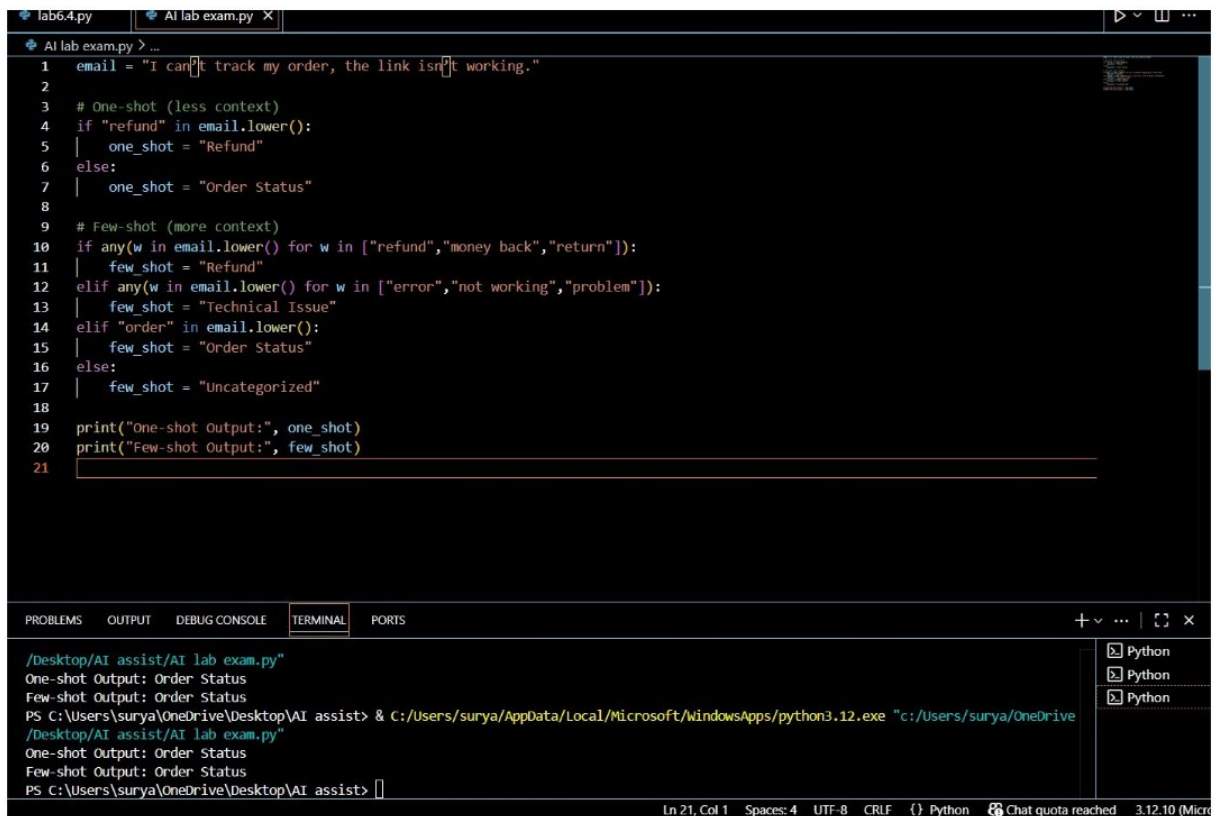
Email: 'I got the wrong size, need a refund.' → Refund

Email: 'I can't track my order, the link isn't working.'"

Likely Output: Technical Issue



## CODE:



The screenshot shows a VS Code editor with a file named 'AI lab exam.py'. The script defines two functions: 'one\_shot' and 'few\_shot', both using 'email.lower()' to check for keywords. The 'one\_shot' function has two conditions: 'refund' leads to 'Refund' and 'order' leads to 'Order Status'. The 'few\_shot' function has three conditions: 'refund', 'money back', or 'return' leads to 'Refund'; 'error', 'not working', or 'problem' leads to 'Technical Issue'; and 'order' leads to 'Order Status'. Both functions default to 'Uncategorized' if no keyword is found. The script prints the results for both functions using the same email text: 'I can't track my order, the link isn't working.'.

```
1 email = "I can't track my order, the link isn't working."
2
3 # One-shot (less context)
4 if "refund" in email.lower():
5     one_shot = "Refund"
6 else:
7     one_shot = "Order Status"
8
9 # Few-shot (more context)
10 if any(w in email.lower() for w in ["refund", "money back", "return"]):
11     few_shot = "Refund"
12 elif any(w in email.lower() for w in ["error", "not working", "problem"]):
13     few_shot = "Technical Issue"
14 elif "order" in email.lower():
15     few_shot = "Order Status"
16 else:
17     few_shot = "Uncategorized"
18
19 print("One-shot Output:", one_shot)
20 print("Few-shot Output:", few_shot)
21
```

The terminal output shows the results of running the script. The 'one-shot' output is 'Order Status' and the 'few-shot' output is 'Order Status'. The terminal also shows the command used to run the script: 'C:\Users\surya\AppData\Local\Microsoft\WindowsApps\python3.12.exe "c:\Users\surya\OneDrive\Desktop\AI assist\AI lab exam.py"'. The status bar at the bottom indicates 'Ln 21, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'.

## 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).*