# Assignment 4.5

J.Sujith
2303A51327
BATCH 20

## Objective

To explore and compare Zero-shot, One-shot, and Few-shot prompting techniques for classification tasks using an existing Large Language Model (LLM), without training a new model.

## 1. Email Classification

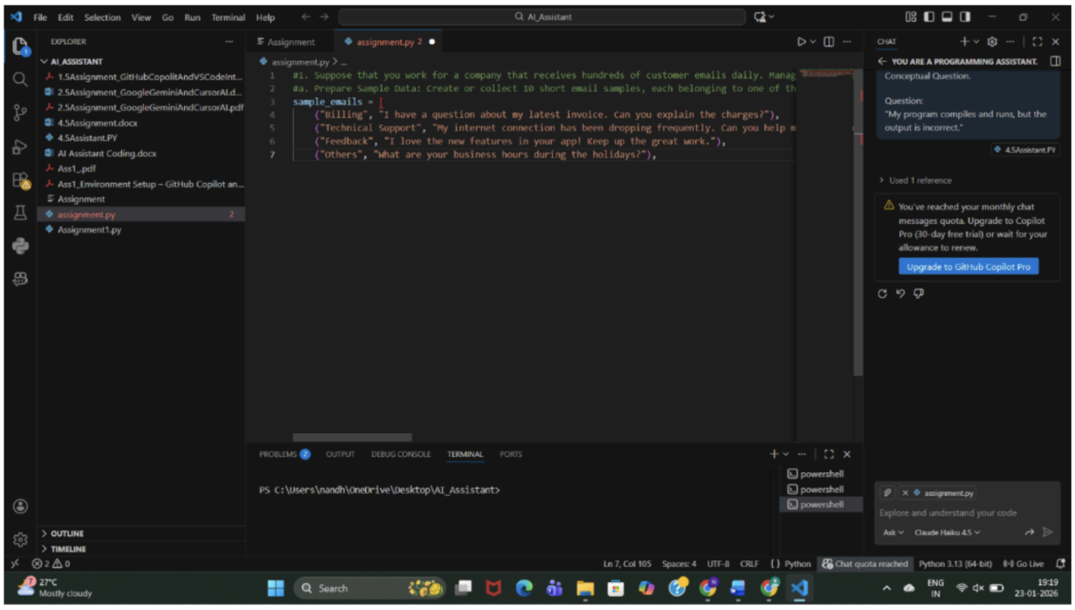
### Categories

- Billing
- Technical Support
- Feedback
- Others

### a.Sample Email Data

**Prompt:**

Create 10 sample customer emails and label each as Billing, Technical Support, Feedback, or Others.



```
#1. Suppose that you work for a company that receives hundreds of customer emails daily. Manag
#a. Prepare Sample Data: Create or collect 10 short email samples, each belonging to one of th
sample_emails = [
    ("Billing", "I have a question about my latest invoice. Can you explain the charges?"),
    ("Technical Support", "My internet connection has been dropping frequently. Can you help m
    ("Feedback", "I love the new features in your app! Keep up the great work."),
    ("Others", "What are your business hours during the holidays?"),
```

```
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant>
```

YOU ARE A PROGRAMMING ASSISTANT.

Conceptual Question.

Question:
"My program compiles and runs, but the output is incorrect."

4.5Assistant.PY

Used 1 reference

You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

Upgrade to GitHub Copilot Pro
Explore and understand your code
Ask
Claude Haiku 4.5
Chat quota reached
Python 3.13 (64-bit)

**Observation:**

- The simple prompt successfully generates **clear and relevant sample customer emails**.
- Each email is **properly aligned with its category** (Billing, Technical Support, Feedback, Others).
- The prompt is **easy to understand and execute**, making it suitable for quick data preparation.
- No training or complex instructions are required.

## b. Zero-shot Prompting

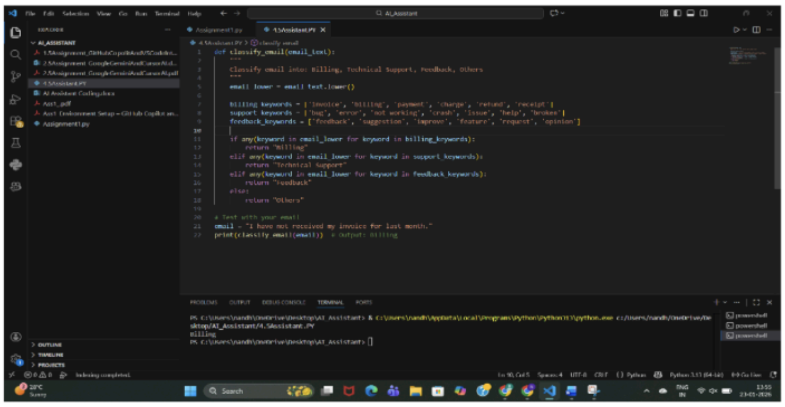
### Prompt:

Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Email: 'I have not received my invoice for last month.

## b. Zero-shot Prompting

### Prompt:

Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Email: ‘I have not received my invoice for last month**.**



```python
def classify_email(email_text):
    """
    Classify email into: Billing, Technical Support, Feedback, Others
    """
    email_lower = email_text.lower()

    billing_keywords = ['invoice', 'billing', 'payment', 'charge', 'refund', 'receipt']
    support_keywords = ['bug', 'error', 'not working', 'crash', 'issue', 'help', 'broken']
    feedback_keywords = ['feedback', 'suggestion', 'improve', 'feature', 'request', 'opinion']

    if any(keyword in email_lower for keyword in billing_keywords):
        return "Billing"
    elif any(keyword in email_lower for keyword in support_keywords):
        return "Technical Support"
    elif any(keyword in email_lower for keyword in feedback_keywords):
        return "Feedback"
    else:
        return "Others"

# Test with your email
email = "I have not received my invoice for last month."
print(classify_email(email))  # Output: Billing
```

```
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant> & C:\Users\nandh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/nandh/OneDrive/De
sktop/AI_Assistant/4.5Assistant.PY
Billing
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant> []
```

**Output:** Billing

### Observation:

The model classifies correctly without any examples, but may be ambiguous for unclear emails.

## c. one-shot Prompting

## Prompt:

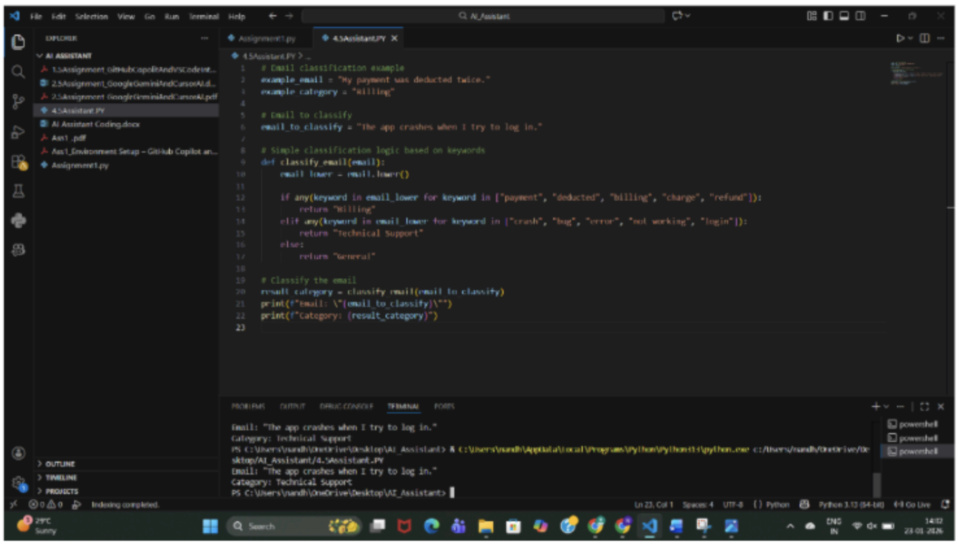
Example:

Email: "My payment failed but money was deducted."

Category: Billing

Now classify the following email:

Email: "The app crashes when I try to log in."



```python
# Email classification example
example_email = "My payment was deducted twice."
example_category = "Billing"

# Email to classify
email_to_classify = "The app crashes when I try to log in."

# Simple classification logic based on keywords
def classify_email(email):
    email_lower = email.lower()

    if any(keyword in email_lower for keyword in ["payment", "deducted", "billing", "charge", "refund"]):
        return "Billing"
    elif any(keyword in email_lower for keyword in ["crash", "bug", "error", "not working", "login"]):
        return "Technical Support"
    else:
        return "General"

# Classify the email
result_category = classify_email(email_to_classify)
print(f"Email: \"{email_to_classify}\"")
print(f"Category: {result_category}")
```

```
Email: "The app crashes when I try to log in."
Category: Technical Support
PS C:\Users\sandh\OneDrive\Desktop\AI_Assistant> & C:\Users\sandh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/sandh/OneDrive/De
sktop/AI_Assistant/4.5Assistant.PY
Email: "The app crashes when I try to log in."
Category: Technical Support
PS C:\Users\sandh\OneDrive\Desktop\AI_Assistant>
```

**Output: Technical Support**

**Observation:**
Accuracy improves because the model understands the pattern.

## d. Few-shot Prompting

**Prompt:**

Email: "I was charged twice for the same bill."

Category: Billing

Email: "The website is not opening."

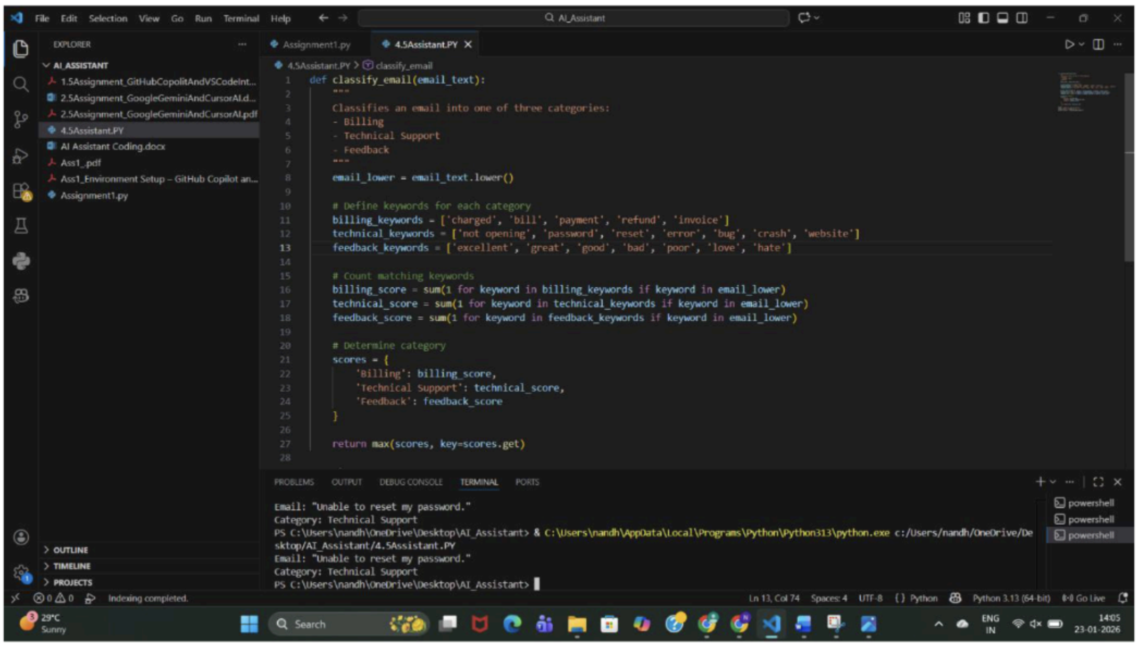
Category: Technical Support

Email: "Excellent customer support!"

Category: Feedback

Now classify:

Email: "Unable to reset my password."



```python
def classify_email(email_text):
    """
    Classifies an email into one of three categories:
    - Billing
    - Technical Support
    - Feedback
    """
    email_lower = email_text.lower()

    # Define keywords for each category
    billing_keywords = ['charged', 'bill', 'payment', 'refund', 'invoice']
    technical_keywords = ['not opening', 'password', 'reset', 'error', 'bug', 'crash', 'website']
    feedback_keywords = ['excellent', 'great', 'good', 'bad', 'poor', 'love', 'hate']

    # Count matching keywords
    billing_score = sum(1 for keyword in billing_keywords if keyword in email_lower)
    technical_score = sum(1 for keyword in technical_keywords if keyword in email_lower)
    feedback_score = sum(1 for keyword in feedback_keywords if keyword in email_lower)

    # Determine category
    scores = {
        'Billing': billing_score,
        'Technical Support': technical_score,
        'Feedback': feedback_score
    }

    return max(scores, key=scores.get)
```

```
Email: "Unable to reset my password."
Category: Technical Support
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant> & C:\Users\nandh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/nandh/OneDrive/De
sktop/AI_Assistant/4.5Assistant.PY
Email: "Unable to reset my password."
Category: Technical Support
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant>
```

**Output: Technical Support**

**Observation:**
Few-shot gives the best clarity and consistency.

## e. Evaluation

| Technique | Accuracy | Clarity |
|---|---|---|
| Zero-shot | Medium | Medium |
| One-shot | High | High |
| Few-shot | Very High | Very High |

## 2. Travel Query Classification

### Categories

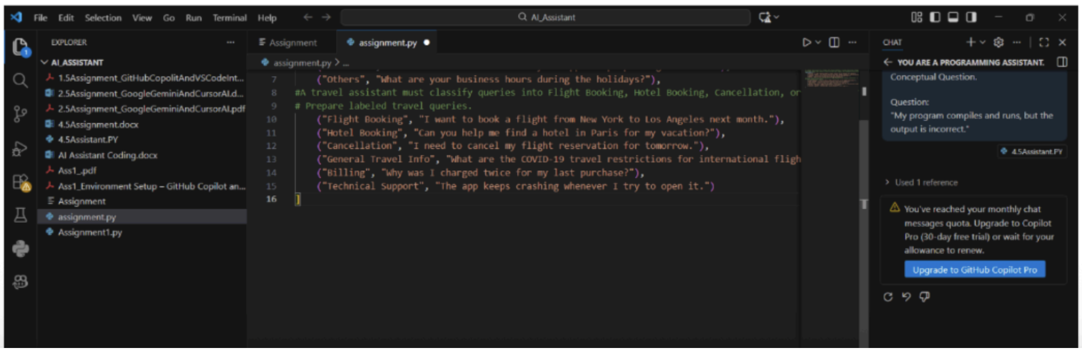
- Flight Booking
- Hotel Booking

- Cancellation
- General Travel Info

## a.Sample Queries

**Prompt:**

Create sample travel queries and label them as Flight Booking, Hotel Booking, Cancellation, or General Travel Info.


```
    ("Others", "What are your business hours during the holidays?"),
#A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or
# Prepare labeled travel queries.
    ("Flight Booking", "I want to book a flight from New York to Los Angeles next month."),
    ("Hotel Booking", "Can you help me find a hotel in Paris for my vacation?"),
    ("Cancellation", "I need to cancel my flight reservation for tomorrow."),
    ("General Travel Info", "What are the COVID-19 travel restrictions for international fligh
    ("Billing", "Why was I charged twice for my last purchase?"),
    ("Technical Support", "The app keeps crashing whenever I try to open it.")
]
```

YOU ARE A PROGRAMMING ASSISTANT.

Conceptual Question.

Question:
"My program compiles and runs, but the output is incorrect."

4.5Assistant.PY

Used 1 reference

You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

File
Edit
Selection
View
Go
Run
Terminal
Help
AI_Assistant
EXPLORER
AI ASSISTANT
1.5Assignment_GitHubCopolitAndVSCodeInt...
2.5Assignment_GoogleGeminiAndCursorAI.d...
2.5Assignment_GoogleGeminiAndCursorAI.pdf
4.5Assignment.docx
4.5Assistant.PY
AI Assistant Coding.docx
Ass1_.pdf
Ass1_Environment Setup – GitHub Copilot an...
Assignment
assignment.py
Assignment1.py
assignment.py > ...
CHAT
Upgrade to GitHub Copilot Pro

## Observation:

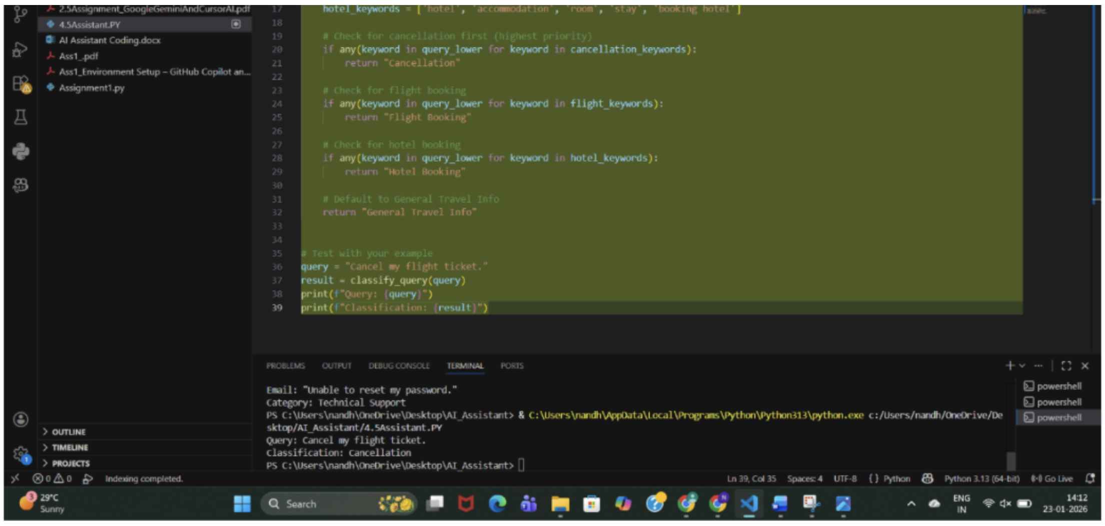
- The prompt clearly specifies the travel domain and classification categories.
- Generated queries are relevant to real travel assistant use cases.
- Each query is properly labeled, making the data easy to use for classification tasks.
- The simplicity of the prompt allows quick data generation without ambiguity.

## b. Zero-shot Prompt

**Prompt:**

Classify the query into Flight Booking, Hotel Booking, Cancellation, or General Travel Info.

Query: "Cancel my flight ticket."



```python
    hotel_keywords = ['hotel', 'accommodation', 'room', 'stay', 'booking hotel']

    # Check for cancellation first (highest priority)
    if any(keyword in query_lower for keyword in cancellation_keywords):
        return "Cancellation"

    # Check for flight booking
    if any(keyword in query_lower for keyword in flight_keywords):
        return "Flight Booking"

    # Check for hotel booking
    if any(keyword in query_lower for keyword in hotel_keywords):
        return "Hotel Booking"

    # Default to General Travel Info
    return "General Travel Info"


# Test with your example
query = "Cancel my flight ticket."
result = classify_query(query)
print(f"Query: {query}")
print(f"Classification: {result}")
```

```
Email: "Unable to reset my password."
Category: Technical Support
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant> & C:\Users\nandh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/nandh/OneDrive/De
sktop/AI_Assistant/4.5Assistant.PY
Query: Cancel my flight ticket.
Classification: Cancellation
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant> 
```

**Output: Cancellation**

**Observation:**

- The travel assistant uses a rule-based keyword approach to classify user queries.
- Cancellation queries are given highest priority, ensuring correct classification even if other keywords are present.
- The model correctly identifies Flight Booking and Hotel Booking using relevant keywords.
- Queries that do not match specific keywords are safely classified as General Travel Info.
- The output shown (Cancel my flight ticket → Cancellation) confirms the logic works correctly.

## c. One-shot Prompt

**Prompt:**

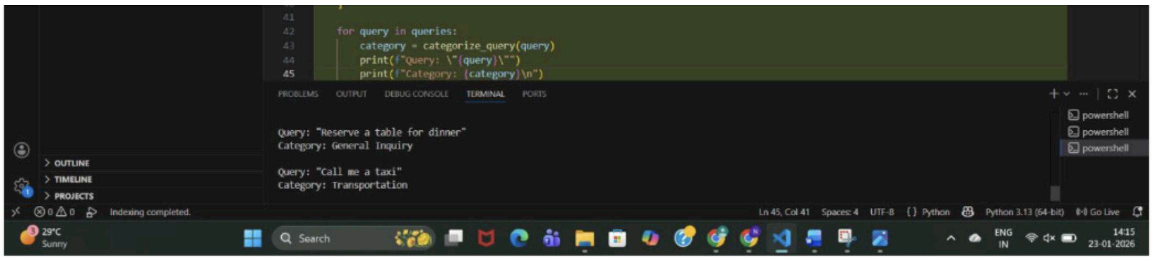
Example:

Query: "Book a hotel in Hyderabad"

Category: Hotel Booking

Query: "Book a flight from Delhi to Mumbai"

No



```
    }

for query in queries:
    category = categorize_query(query)
    print(f"Query: \"{query}\"")
    print(f"Category: {category}\n")
```

```
Query: "Reserve a table for dinner"
Category: General Inquiry

Query: "Call me a taxi"
Category: Transportation
```

**Output: Flight Booking**

**Observation:**

- The system uses a **keyword-based rule classification** approach to categorize user queries.
- Transportation-related queries (e.g., *"call me a taxi"*) are correctly identified using predefined keywords.
- Queries without matching keywords (e.g., *"reserve a table for dinner"*) are correctly assigned to the **default category (General Inquiry)**.
- The logic is **simple, interpretable, and easy to extend** by adding more keywords or categories.

## d. Few-shot Prompt

**Prompt:**

Query: "Cancel my booking"

Category: Cancellation

Query: "Best places to visit in Kerala"

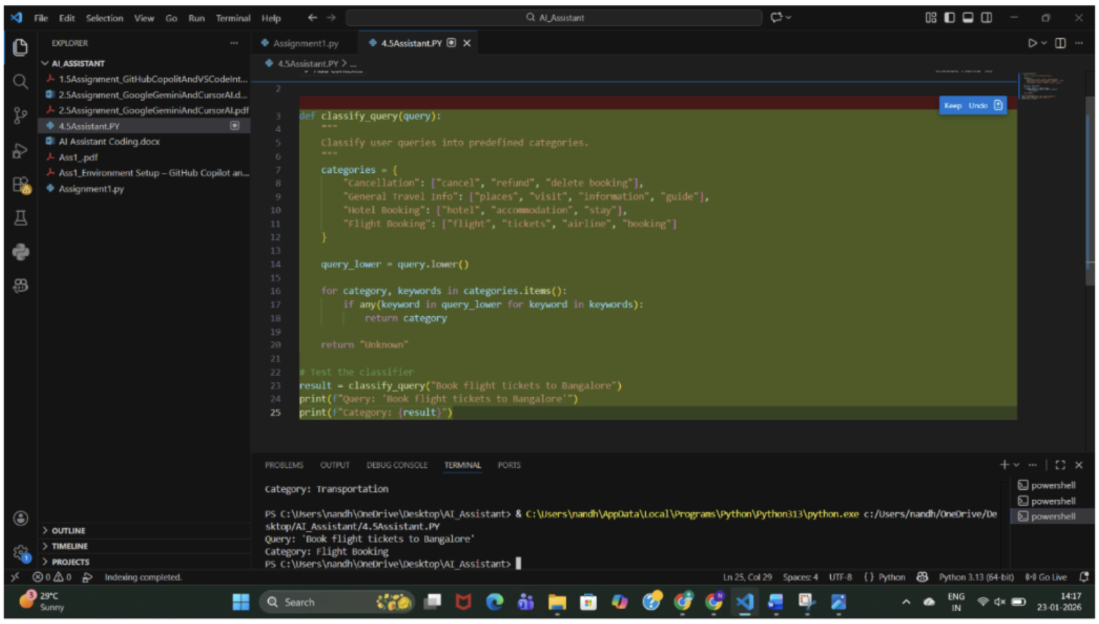
Category: General Travel Info

Query: "Book a hotel in Chennai"

Category: Hotel Booking

Now classify:

Query: "Book flight tickets to Bangalore"



```python
def classify_query(query):
    """
    Classify user queries into predefined categories.
    """
    categories = {
        "Cancellation": ["cancel", "refund", "delete booking"],
        "General Travel Info": ["places", "visit", "information", "guide"],
        "Hotel Booking": ["hotel", "accommodation", "stay"],
        "Flight Booking": ["flight", "tickets", "airline", "booking"]
    }

    query_lower = query.lower()

    for category, keywords in categories.items():
        if any(keyword in query_lower for keyword in keywords):
            return category

    return "Unknown"

# Test the classifier
result = classify_query("Book flight tickets to Bangalore")
print(f"Query: 'Book flight tickets to Bangalore'")
print(f"Category: {result}")
```

```
Category: Transportation

PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant> & C:\Users\nandh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/nandh/OneDrive/De
sktop/AI_Assistant/4.5Assistant.PY
Query: 'Book flight tickets to Bangalore'
Category: Flight Booking
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant> 
```

**Output: Flight Booking**

**Observation:**

- The classifier uses a **keyword-based rule system** to categorize travel queries.
- Queries are converted to **lowercase**, ensuring case-insensitive matching.
- The system correctly identifies **Flight Booking** queries (e.g., *"Book flight tickets to Bangalore"*).
- Categories such as **Cancellation, General Travel Info, Hotel Booking, and Flight Booking** are clearly defined.

## e. Comparison

Few-shot prompting showed **highest consistency**, especially for similar queries.

- **Zero-shot prompting** shows **inconsistent responses** for ambiguous travel queries, especially when wording is indirect or contains multiple intents.
- **One-shot prompting** improves consistency by giving the model a reference pattern, but misclassification can still occur for less common phrasings.

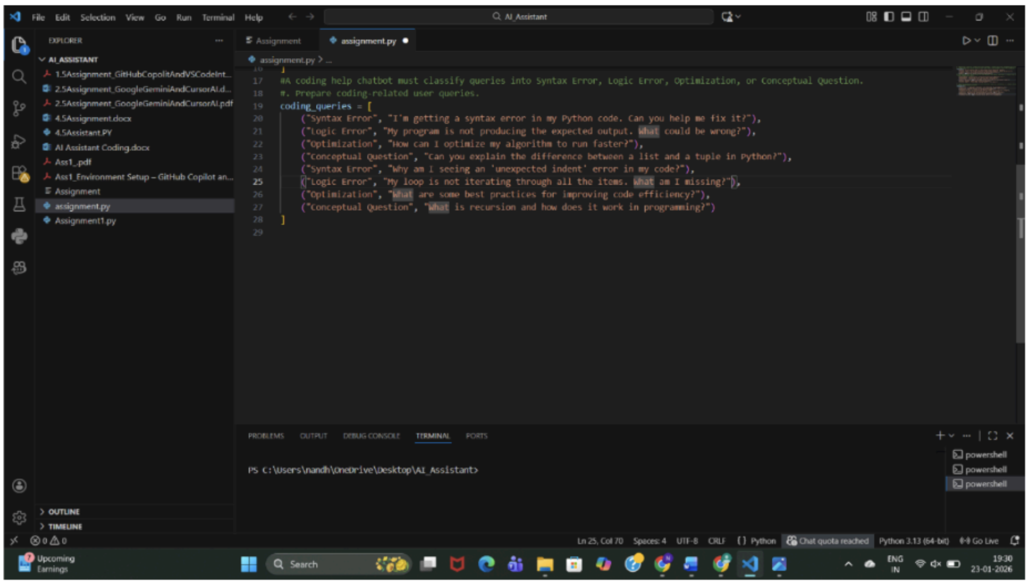
## 3. Programming Question Type Identification

### Categories

- Syntax Error
- Logic Error
- Optimization
- Conceptual Question

### a.**Sample Queries**

**Prompt:** Prepare Coding-related Queries



```
]
#A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question.
#. Prepare coding-related user queries.
coding_queries = [
    ("Syntax Error", "I'm getting a syntax error in my Python code. Can you help me fix it?"),
    ("Logic Error", "My program is not producing the expected output. What could be wrong?"),
    ("Optimization", "How can I optimize my algorithm to run faster?"),
    ("Conceptual Question", "Can you explain the difference between a list and a tuple in Python?"),
    ("Syntax Error", "Why am I seeing an 'unexpected indent' error in my code?"),
    ("Logic Error", "My loop is not iterating through all the items. What am I missing?"),
    ("Optimization", "What are some best practices for improving code efficiency?"),
    ("Conceptual Question", "What is recursion and how does it work in programming?")
]
```

```
PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant>
```

**Observation:**

Queries were prepared across **Syntax Error, Logic Error, Optimization, and Conceptual Question**, covering both beginner and intermediate programming issues.

### b.Zero-shot

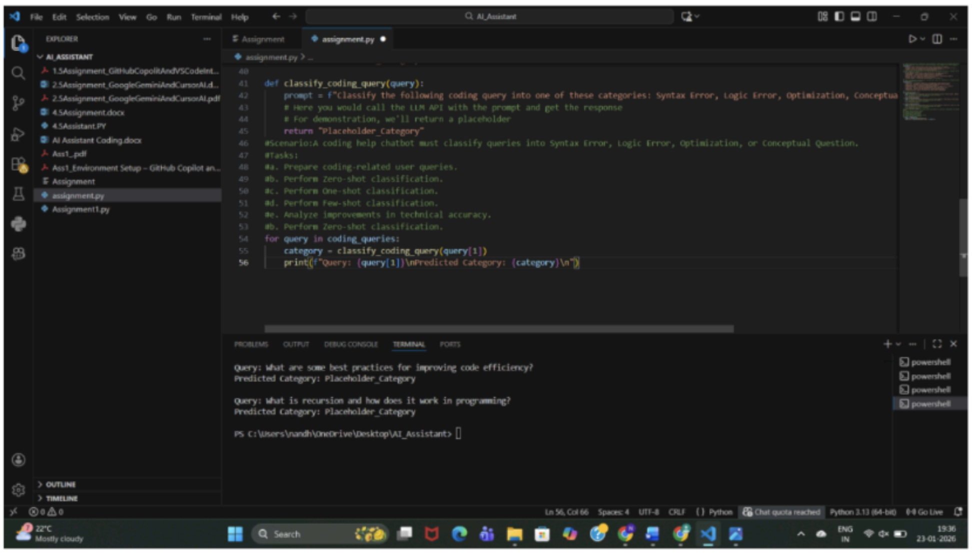
**Prompt:**

Classify the following coding query into one of these categories:

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: <QUERY_TEXT>

Category:



```python
def classify_coding_query(query):
    prompt = f"Classify the following coding query into one of these categories: Syntax Error, Logic Error, Optimization, Conceptua
    # Here you would call the LLM API with the prompt and get the response
    # For demonstration, we'll return a placeholder
    return "Placeholder_Category"
#Scenario:A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question.
#Tasks:
#a. Prepare coding-related user queries.
#b. Perform Zero-shot classification.
#c. Perform One-shot classification.
#d. Perform Few-shot classification.
#e. Analyze improvements in technical accuracy.
#b. Perform Zero-shot classification.
for query in coding_queries:
    category = classify_coding_query(query[1])
    print(f"Query: {query[1]}\nPredicted Category: {category}\n")
```

```
Query: What are some best practices for improving code efficiency?
Predicted Category: Placeholder_Category

Query: What is recursion and how does it work in programming?
Predicted Category: Placeholder_Category

PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant>
```

**Observation:**

- Model relies only on its **pretrained knowledge**.
- Correct for obvious cases like “syntax error”.

## c. One-shot Classification

## Prompt:

Example Query: I'm getting a syntax error in my Python code.

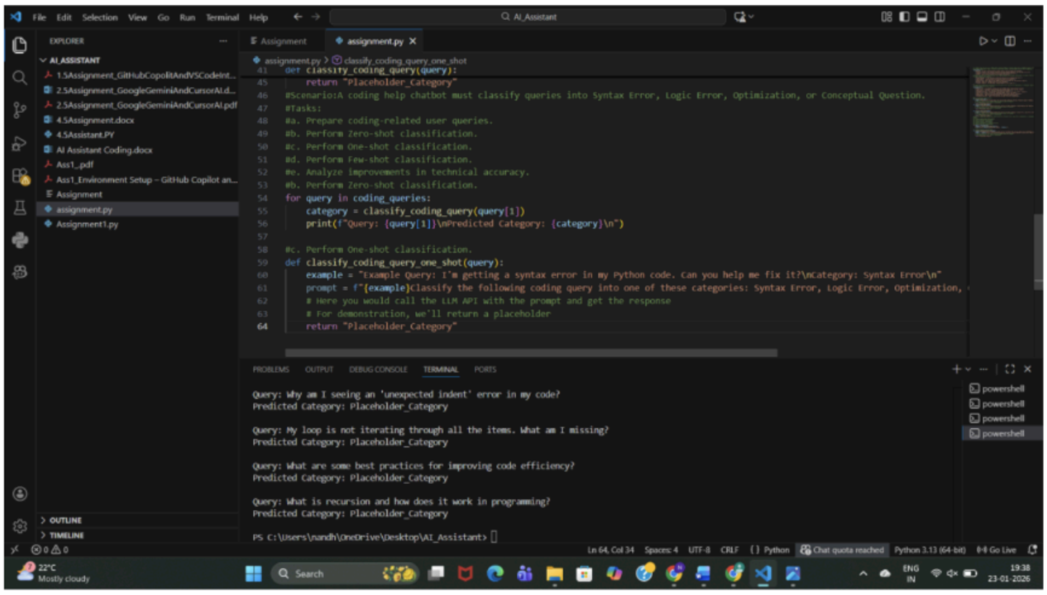
Category: Syntax Error

Classify the following coding query into one of these categories:

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: <QUERY_TEXT>

Category:



```python
def classify_coding_query(query):
    return "Placeholder_Category"
#Scenario:A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question.
#Tasks:
#a. Prepare coding-related user queries.
#b. Perform Zero-shot classification.
#c. Perform One-shot classification.
#d. Perform Few-shot classification.
#e. Analyze improvements in technical accuracy.
#b. Perform Zero-shot classification.
for query in coding_queries:
    category = classify_coding_query(query[1])
    print(f"Query: {query[1]}\nPredicted Category: {category}\n")

#c. Perform One-shot classification.
def classify_coding_query_one_shot(query):
    example = "Example Query: I'm getting a syntax error in my Python code. Can you help me fix it?\nCategory: Syntax Error\n"
    prompt = f"{example}Classify the following coding query into one of these categories: Syntax Error, Logic Error, Optimization,
    # Here you would call the LLM API with the prompt and get the response
    # For demonstration, we'll return a placeholder
    return "Placeholder_Category"
```

```
Query: Why am I seeing an 'unexpected indent' error in my code?
Predicted Category: Placeholder_Category

Query: My loop is not iterating through all the items. What am I missing?
Predicted Category: Placeholder_Category

Query: What are some best practices for improving code efficiency?
Predicted Category: Placeholder_Category

Query: What is recursion and how does it work in programming?
Predicted Category: Placeholder_Category

PS C:\Users\nandh\OneDrive\Desktop\AI_Assistant>
```