📚 Technical Documentation: Groq AI Interaction Scripts

Table of Contents

1. Introduction

This project provides three Python scripts that interact with the **Groq Language Model API** using the **Groq Python SDK**.

The scripts demonstrate how to configure, send prompts, handle streaming responses, and optionally terminate responses based on stop conditions.

These examples use the **Llama-3.3-70B-Versatile** model to respond to a sample user query.

---

2. System Requirements

- Python 3.8+
- `groq` Python package
- Internet connection
- Access to a valid Groq API key

---

3. Project Structure

```
project/
├── grocaistop.py    # Streams responses, with a stop sequence
├── grocstream.py    # Streams responses, without a stop sequence
├── groqai.py        # Non-streaming response
```

---

4. Setup Instructions

Install the Groq client library:

```
pip install groq
```

1.
2. Ensure your API Key is active and valid:
    - Replace `api_key="YOUR_API_KEY_HERE"` with your actual Groq API key.

Run any of the scripts:

```
python grocaistop.py
python grocstream.py
python groqai.py
```

3.

---

5. Detailed Code Walkthrough

5.1 Authentication

Each script first initializes a **Groq client** using an API key:

```
from groq import Groq
client = Groq(api_key="YOUR_API_KEY")
```

- **Purpose**: Authenticate and authorize access to Groq's services.

5.2 Creating Chat Completions

The `client.chat.completions.create()` method is used to create a conversation with the AI model.

Example structure:

```
chat_completion = client.chat.completions.create(
    messages=[
        {"role": "system", "content": "you are a helpful assistant."},
        {"role": "user", "content": "Explain the importance of fast language models"},
    ],
    model="llama-3.3-70b-versatile",
    temperature=0.5,
    max_completion_tokens=1024,
    top_p=1,
    stream=True or False,
    stop="some_phrase" or None,
)
```

Key roles in the `messages` array:

- `system`: Configures how the AI behaves throughout the conversation.
- `user`: Represents the actual input question/request from the user.

5.3 Streaming vs Non-Streaming

Streaming Mode (`stream=True`):
The AI sends the output piece-by-piece (called "chunks"). Useful for real-time applications.

In `grocaistop.py` and `grocstream.py`, streaming is enabled:

```
stream=True
```

The script loops through the response chunks and prints them as they arrive:

```
for chunk in chat_completion:
  print(chunk.choices[0].delta.content, end="", flush=True)
```

-

Non-Streaming Mode (`stream=False`):
The AI sends the full response at once when it finishes processing.

In `groqai.py`, non-streaming is used:

```
stream=False
```

After completion:

```
print(chat_completion.choices[0].message.content)
```

-

5.4 Stop Sequences

- A **stop sequence** tells the AI when to automatically end its response generation.

In `grocaistop.py`, a stop condition is used:

```
stop="real-time applications"
```

- The model will stop generating once it mentions *"real-time applications"*.
- In `grocstream.py` and `groqai.py`, no stop sequence is set (`stop=None`).

---

6. Key Parameters Explained

Parameter	Type	Description

`messages`	List	Ordered sequence of system, user, and assistant messages to define the conversation context.
`model`	String	Specifies the AI model used. (Here: "llama-3.3-70b-versatile")
`temperature`	Float	Controls randomness: 0 = more predictable, 1 = more creative.
`max_completion_tokens`	Integer	Sets the limit for the response length.
`top_p`	Float	Controls diversity: 1.0 considers all tokens; lower values limit diversity.
`stream`	Boolean	Determines whether to receive the output as a real-time stream or all at once.
`stop`	String or None	Optional string that, if found, ends the response generation.

---

7. Differences Between Scripts

Script	Streaming	Stop Sequence	Response Mode
`grocaistop.py`	✅ Yes	✅ Yes ("real-time applications")	Stream and stop
`grocstream.py`	✅ Yes	🚫 No	Stream fully
`groqai.py`	🚫 No	🚫 No	Full output after generation

---

8. Security Considerations

- **Protect your API keys**:
  Never hardcode your keys in production code. Use environment variables or secure storage.
- **Rate Limiting and Quotas**:
  Be aware of your API limits based on your Groq account tier.

- **User Input Validation**:
  If extending these scripts to accept external input, validate to prevent prompt injection attacks.

---

9. References

- Groq Python SDK Documentation
- Understanding Temperature and Top_p Sampling

✅ End of Technical Document

Created by Jason Mbugua on 27/4/2025