Okay, I can help you structure the Groq API reference for Python from the text file you provided. While I can't create a PDF file directly, I can organize the information in an understandable format that you can use as context for your AI project.

Here’s a breakdown of the Groq Python API based on the document:

Groq API Overview

The Groq API allows interaction with various services, including chat completions, audio processing (transcription, translation, speech generation), model management, batch processing, and file handling. Authentication is typically done using an API key, often retrieved from environment variables.

---

1. Chat Completions

- **Endpoint:** POST https://api.groq.com/openai/v1/chat/completions
- **Purpose:** Creates a model response for a given chat conversation.
- **Key Request Parameters:**
  - messages (array, required): A list of message objects representing the conversation history.
  - model (string, required): The ID of the model to use (e.g., llama-3.3-70b-versatile, llama3-8b-8192).
  - temperature (number, optional, default: 1): Controls randomness. Lower values (~0.2) are more deterministic, higher values (~0.8) more random. It's recommended to alter either temperature or top_p, not both.
  - max_completion_tokens (integer, optional): Maximum number of tokens to generate in the completion. (max_tokens is deprecated).
  - top_p (number, optional, default: 1): Nucleus sampling parameter. Considers tokens comprising the top 'p' probability mass. Recommended to alter either top_p or temperature, not both.
  - stream (boolean, optional, default: false): If true, sends partial message deltas as server-sent events.
  - stop (string/array, optional): Sequences where the API will stop generating tokens.
  - response_format (object, optional): Can enable JSON mode by setting {"type": "json_object"}. Requires instructing the model to produce JSON via messages.
  - tools (array, optional): A list of tools (currently only functions) the model may call. Max 128 functions supported. (functions parameter is deprecated).
  - tool_choice (string/object, optional): Controls which tool (if any) is called.

    Options: none, auto, required, or specifying a function. (function_call is deprecated).
  - frequency_penalty (number, optional, default: 0): Penalizes tokens based on existing frequency (-2.0 to 2.0).
  - presence_penalty (number, optional, default: 0): Penalizes tokens based on presence so far (-2.0 to 2.0).
  - seed (integer, optional): For attempting deterministic outputs.
- **Returns:** A chat completion object or a stream of chunks if stream=true.
- **Python Example:**

```Python
# [cite: 55]
import os
from groq import Groq

client = Groq(
    api_key=os.environ.get("GROQ_API_KEY"), # This is the default and can be omitted
)

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", # [cite: 56]
        }
    ],
    model="llama-3.3-70b-versatile", # Example model ID used in provided snippet[cite: 56],
although the response example shows llama3-8b-8192 [cite: 56]
)

print(chat_completion.choices[0].message.content)
# [cite: 55]
```

---

2. Audio Processing

a. Create Transcription

- **Endpoint:** POST https://api.groq.com/openai/v1/audio/transcriptions
- **Purpose:** Transcribes audio into the input language.
- **Key Request Parameters:**
  - model (string, required): Only whisper-large-v3 is currently available.
  - file (string, optional): The audio file object (formats: flac, mp3, mp4, mpeg, mpga, m4a, ogg, wav, webm). Either file or url required. Not supported in Batch API.
  - url (string, optional): Audio URL. Required for Batch API.
  - language (string, optional): Input audio language (ISO-639-1 format) improves accuracy.
  - response_format (string, optional, default: json): Output format (json, text, verbose_json).
  - timestamp_granularities[] (array, optional, default: segment): Specify word or segment timestamps. Requires response_format to be verbose_json. Word timestamps add latency.
- **Returns:** Audio transcription object.
- **Python Example:**

```python
# [cite: 83]
import os
from groq import Groq

client = Groq()
# Assuming sample_audio.m4a is in the same directory as the script
# filename = os.path.dirname(__file__) + "/sample_audio.m4a" # Original line might need adjustment based on actual file location
filename = "sample_audio.m4a" # Simplified path

with open(filename, "rb") as file:
  transcription = client.audio.transcriptions.create(
    file=(filename, file.read()),
    model="whisper-large-v3",
    prompt="Specify context or spelling",  # Optional
    response_format="json",  # Optional
    language="en",  # Optional
    temperature=0.0  # Optional
  )
  print(transcription.text)
# [cite: 83]
```

b. Create Translation

- **Endpoint:** POST https://api.groq.com/openai/v1/audio/translations
- **Purpose:** Translates audio into English.
- **Key Request Parameters:** Similar to transcription, but prompt should be in English.
  - model (string, required): Only whisper-large-v3 currently available.
  - file (string, optional) or url (string, optional). Only url supported in Batch API.
- **Returns:** Audio translation object.
- **Python Example:** Provided, similar structure to transcription.

c. Create Speech (Text-to-Speech)

- **Endpoint:** POST https://api.groq.com/openai/v1/audio/speech
- **Purpose:** Generates audio from input text.
- **Key Request Parameters:**
  - input (string, required): Text to generate audio for.
  - model (string, required): TTS model ID (e.g., playai-tts).
  - voice (string, required): Voice ID (e.g., Fritz-PlayAI).
  - response_format (string, optional, default: wav): Only wav is supported.
  - speed (number, optional, default: 1): Only 1.0 is supported.
- **Returns:** Audio file in wav format.
- **Python Example:**

  ```python
  # [cite: 97]
  import os
  from groq import Groq

  client = Groq(api_key=os.environ.get("GROQ_API_KEY"))

  speech_file_path = "speech.wav"
  model = "playai-tts"
  voice = "Fritz-PlayAI"
  text = "I love building and shipping new features for our users!"
  response_format = "wav"

  response = client.audio.speech.create(
     model=model,
     voice=voice,
  ```

```
    input=text,
    response_format=response_format
)

response.write_to_file(speech_file_path)
# [cite: 97]
```

---

3. Models

- **List Models:**
  - **Endpoint:** GET https://api.groq.com/openai/v1/models
  - **Purpose:** Lists all available models.
  - **Returns:** A list of model objects, including IDs like gemma2-9b-it, llama3-8b-8192, llama3-70b-8192, whisper-large-v3, etc..
  - **Python Example:** Provided.
- **Retrieve Model:**
  - **Endpoint:** GET https://api.groq.com/openai/v1/models/{model}
  - **Purpose:** Gets detailed information about a specific model.
  - **Returns:** A model object.
  - **Python Example:** Provided.

---

4. Batches

- **Purpose:** Allows asynchronous processing of multiple requests from an uploaded file.
- **Create Batch:**
  - **Endpoint:** POST https://api.groq.com/openai/v1/batches
  - **Key Request Parameters:**
    - input_file_id (string, required): ID of the uploaded JSONL file (max 100 MB, purpose 'batch').
    - endpoint (string, required): Target endpoint (e.g., /v1/chat/completions).
    - completion_window (string, required): Time frame for processing (e.g., "24h").
  - **Returns:** A created batch object.
  - **Python Example:** Provided.
- **Retrieve Batch:**
  - **Endpoint:** GET https://api.groq.com/openai/v1/batches/{batch_id}
  - **Purpose:** Retrieves the status and details of a batch.
  - **Returns:** A batch object.

    - **Python Example:** Provided.
- **List Batches:**
    - **Endpoint:** GET https://api.groq.com/openai/v1/batches
    - **Purpose:** Lists your organization's batches.
    - **Returns:** A list of batch objects.
    - **Python Example:** Provided.

---

5. Files

- **Purpose:** Manage files for use with endpoints like the Batch API.
- **Upload File:**
    - **Endpoint:** POST https://api.groq.com/openai/v1/files
    - **Key Request Parameters:**
        - file (string, required): The file object to upload.
        - purpose (string, required): Intended use (e.g., "batch"). Batch API requires JSONL format, max 100MB.
    - **Returns:** The uploaded File object.
    - **Python Example:** Uses requests library for upload.
- **List Files:**
    - **Endpoint:** GET https://api.groq.com/openai/v1/files
    - **Purpose:** Returns a list of your files.
    - **Returns:** A list of File objects.
    - **Python Example:** Provided.
- **Delete File:**
    - **Endpoint:** DELETE https://api.groq.com/openai/v1/files/{file_id}
    - **Purpose:** Deletes a specific file.
    - **Returns:** A deleted file response object.
    - **Python Example:** Provided.
- **Retrieve File:**
    - **Endpoint:** GET https://api.groq.com/openai/v1/files/{file_id}
    - **Purpose:** Returns information about a specific file.
    - **Returns:** A file object.
    - **Python Example:** Provided.
- **Download File:**
    - **Endpoint:** GET https://api.groq.com/openai/v1/files/{file_id}/content
    - **Purpose:** Returns the content of a specified file.
    - **Returns:** The file content.
    - **Python Example:** Provided.

This summary covers the main functionalities described in the provided Groq API reference. You can use this structured information as context for your AI project.