# 1. Introduction

This document outlines the basic understanding to read and ask question to retrieve answer from document for a proof of concept. The study involves to build semantic relationship and summarize text from PDF documents using OpenAI's text-embedded models. One of the most powerful applications enabled by LLMs is sophisticated question-answering (Q&A) chatbots. These are applications that can answer questions about specific source information like PDF, websites, images etc. These applications use a technique known as Retrieval Augmented Generation, or RAG.

The goal is to efficiently extract, process, and summarize content from PDF documents, leveraging advanced natural language processing techniques.

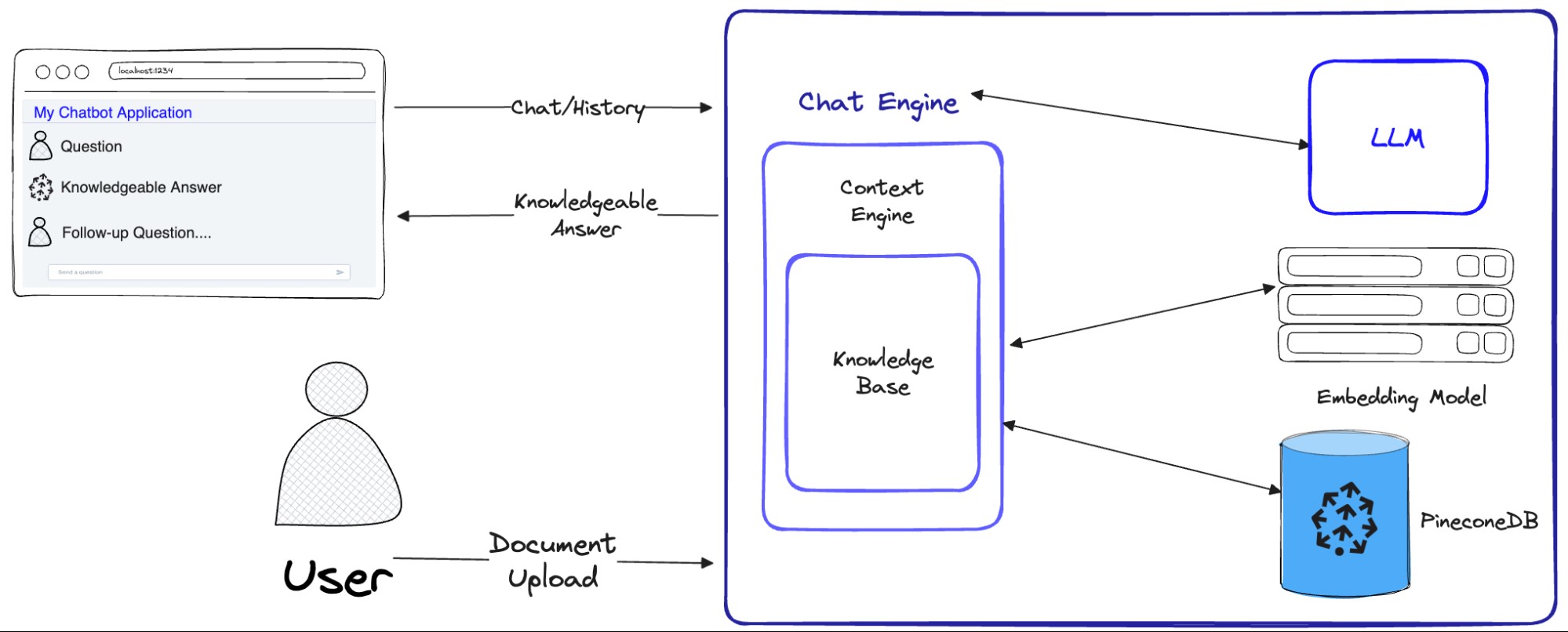
# 2. Pre-Requisites

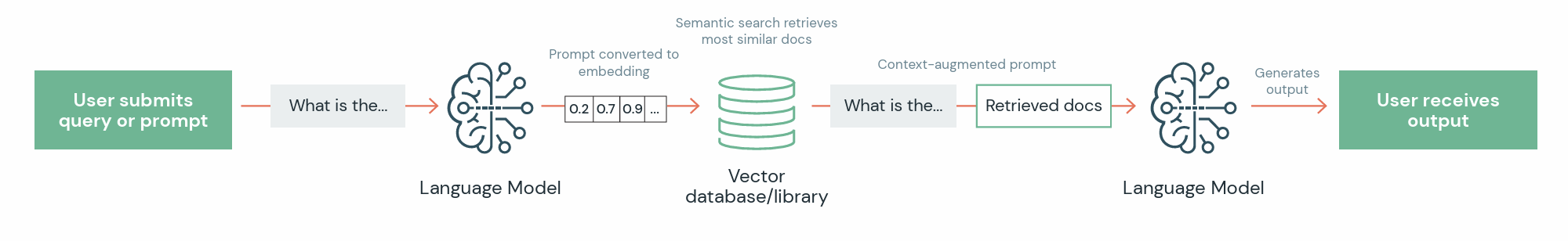
The project requires the installation of several software packages to handle PDF processing, text extraction, and interaction with OpenAI's APIs for text embedding and summarization.

For deployment we use amazon Ec2 and pinecone DB for storing vectors

**!pip install config  
!pip install langchain --upgrade  
!pip install pypdf  
pip install python-dotenv  
!pip install pinecone-client  
!pip install openai  
!pip install tiktoken**

# 3. Chat PDF System Workflow

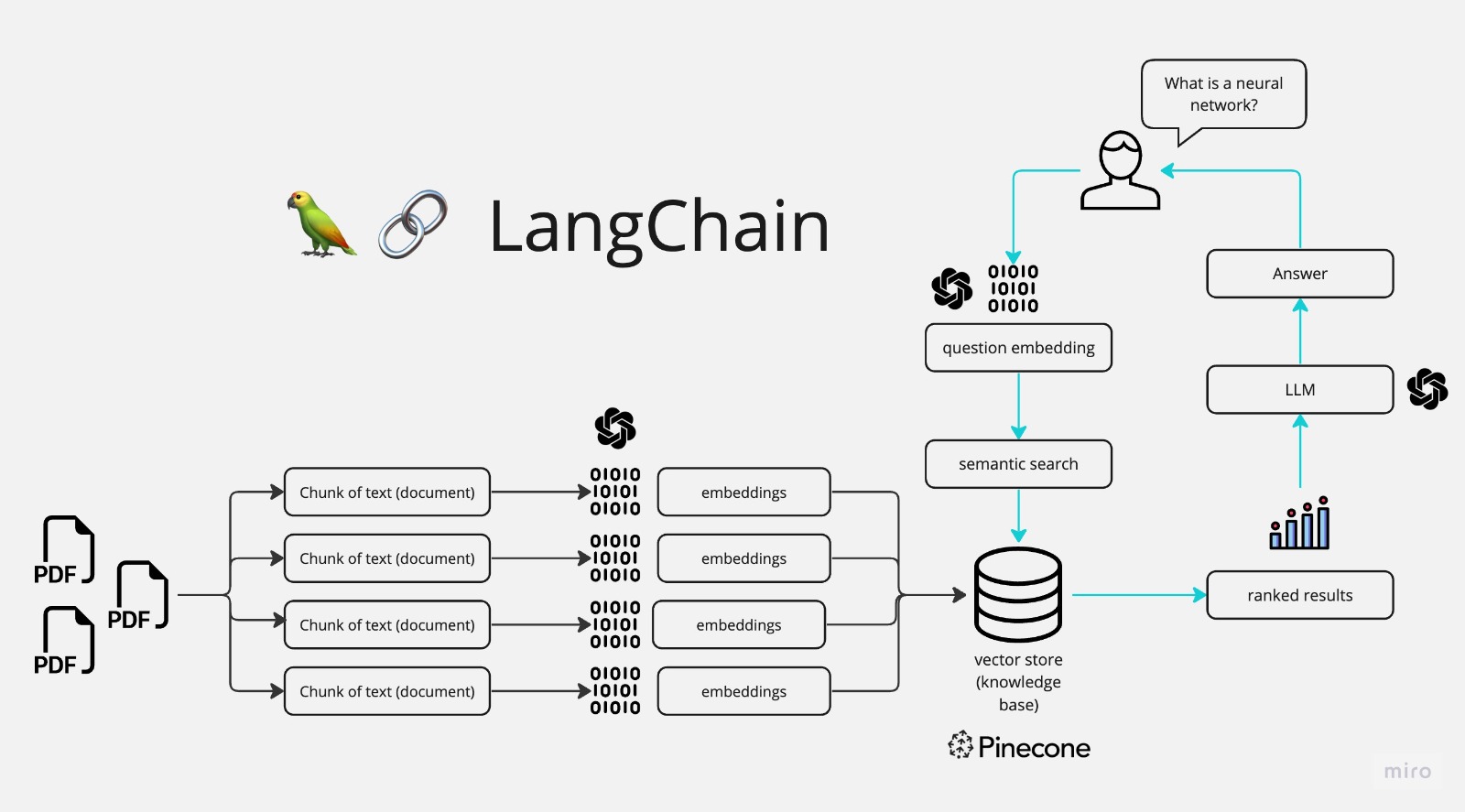




# 4. Architecture

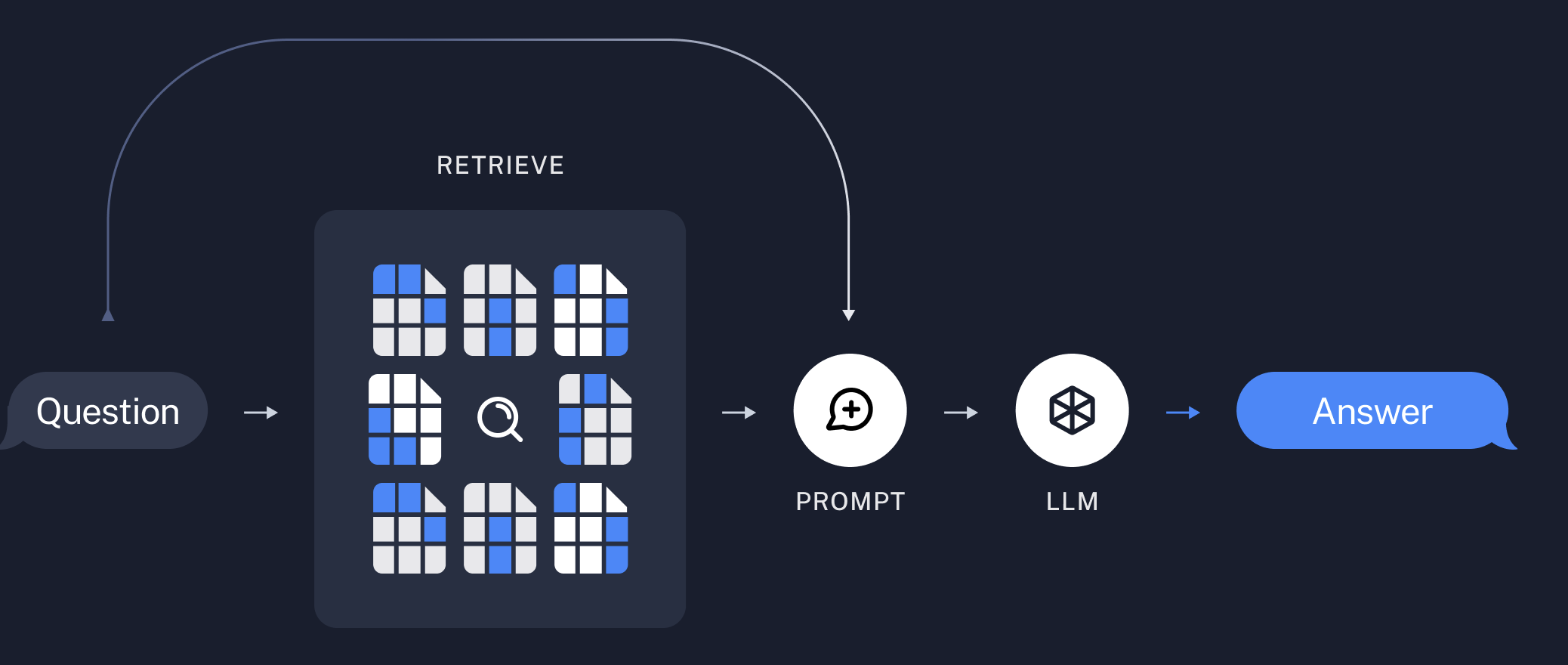
LangChain is a framework designed to simplify the creation of applications using large language models (LLMs). Langchain overlap with language models in general, including document analysis and summarization, chatbots, and code analysis.

RAG is a technique for augmenting LLM knowledge with additional data.



A typical RAG application has two main components:

* Indexing: a pipeline for ingesting data from a source and indexing it. This usually happens offline
* Retrieval and generation: the actual RAG chain, which takes the user query at run time and retrieves the relevant data from the index, then passes that to the model.



# 3. High Level Process

* Upload and extract text from PDF documents using PyPDF.
* Split extracted text into manageable chunks and prepare it for processing.
* Identify PDF attributes and build metadata for vector database storage.
* Create word embeddings using OpenAI’s langchain models.
* Set up and interact with Pinecone's vector database to store and retrieve embeddings.
* Perform similarity searches in the vector database to find relevant text passages.

# 4. Detailed Level Process

## 4.1 Text Extraction and Chunking

The process begins with the extraction of text from PDF documents using Document loaders. Extracted text is then split into chunks of 1000 words with an overlap of 200 words.

## 4.2 Metadata Creation and Storage

Metadata is created for each text chunk to facilitate efficient storage and retrieval. This includes characteristics like character length, page number, and source of a file

## 4.3 Word Embeddings Generation (Openai , Langchain)

Word embeddings are generated using OpenAI's text-embedding models, which are then used to represent the text chunks in a vector space.

## 4.4 Vector Database Interaction(PineCone)

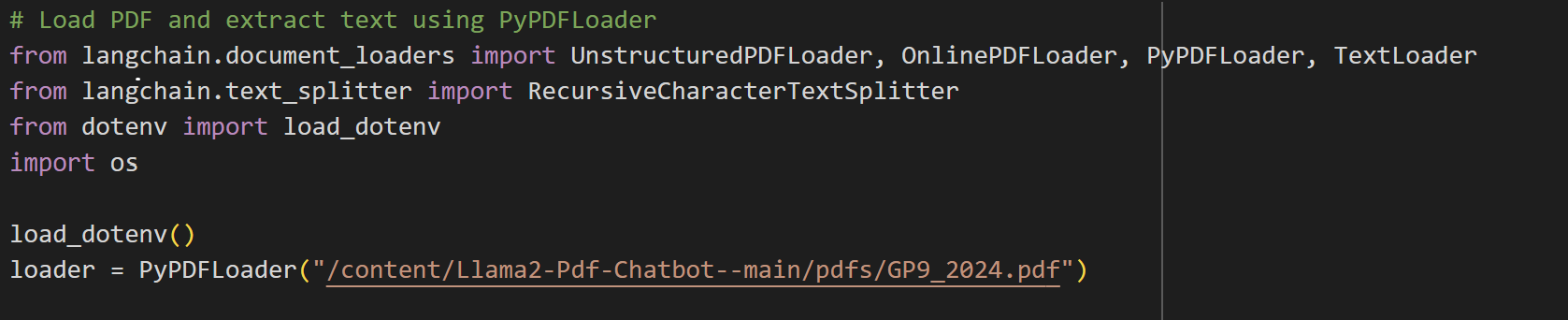
Embeddings are stored in Pinecone's vector database.Perform a search on created index and store the sentences /text along with their corresponding similarity matched scores using models.

## 4.5 Retrieve relevant items from vector DB

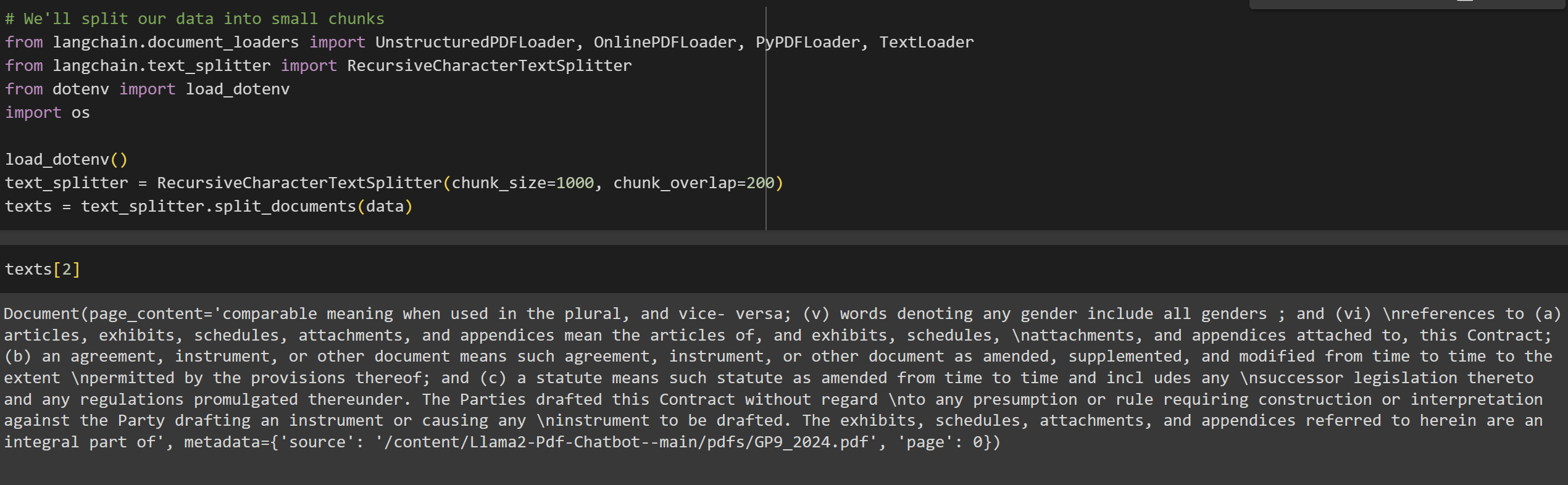
Using prompts using a query to bot for relevant text and get the context of same using semantic search as a summarized result.

# 5. Code Snippets

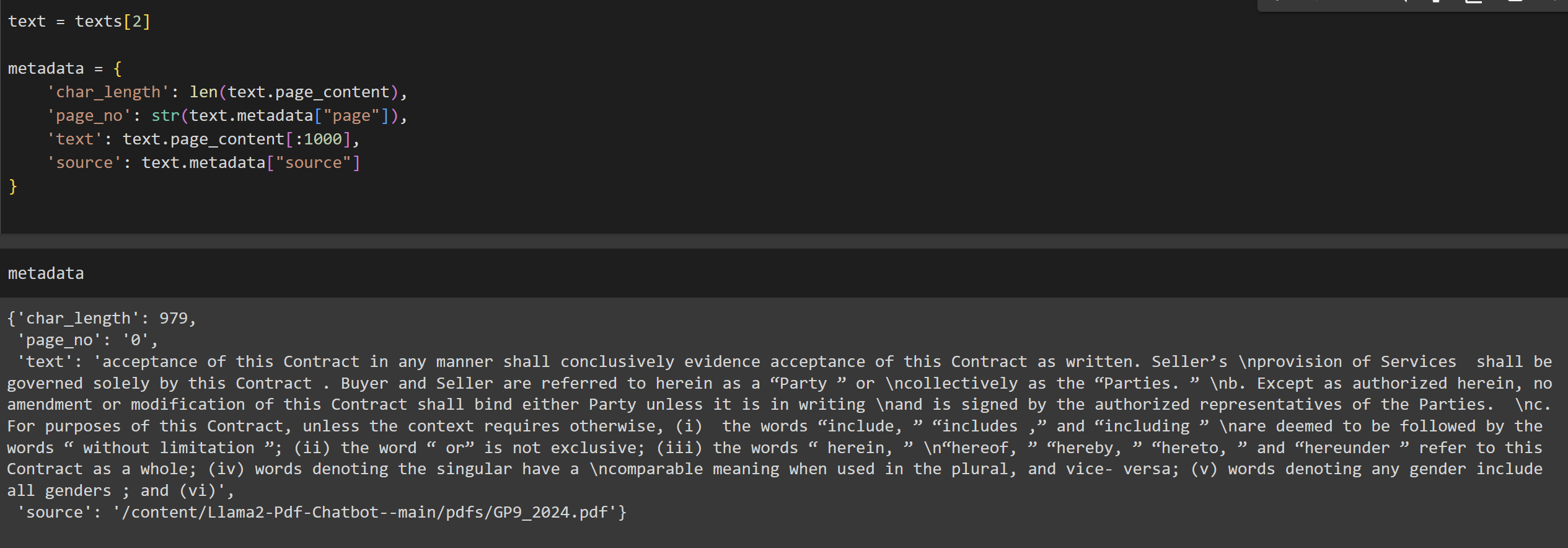
* Load document into the system



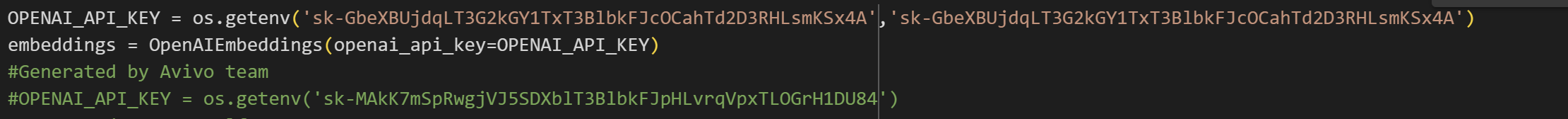
* Split the document in chunks

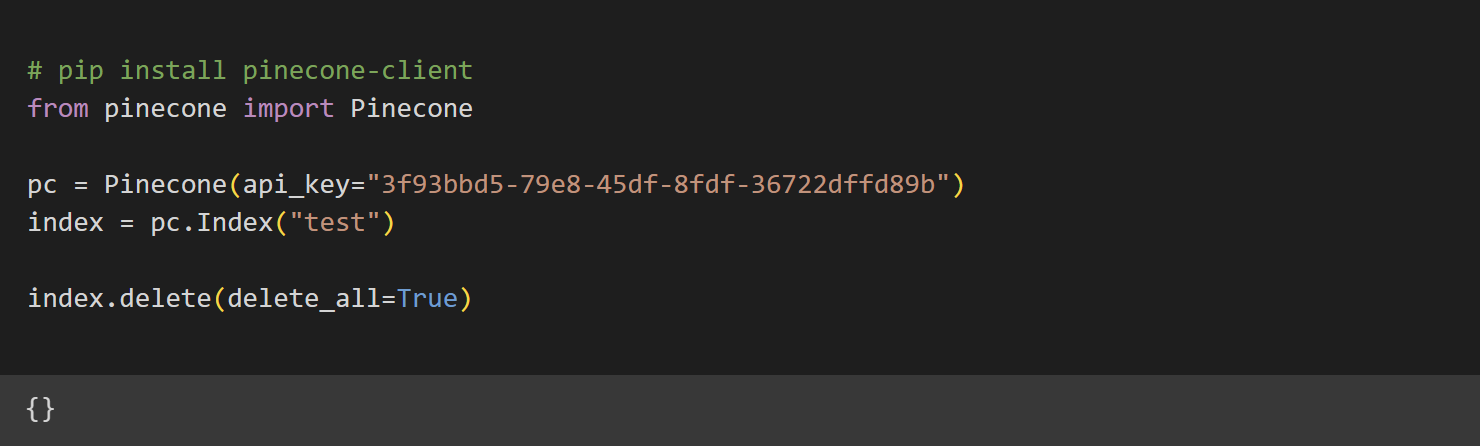


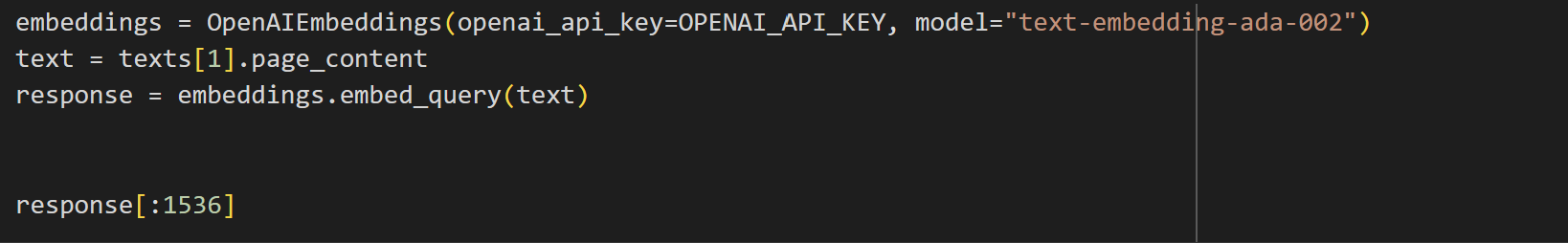
* Define the metadata attributes of pdf file

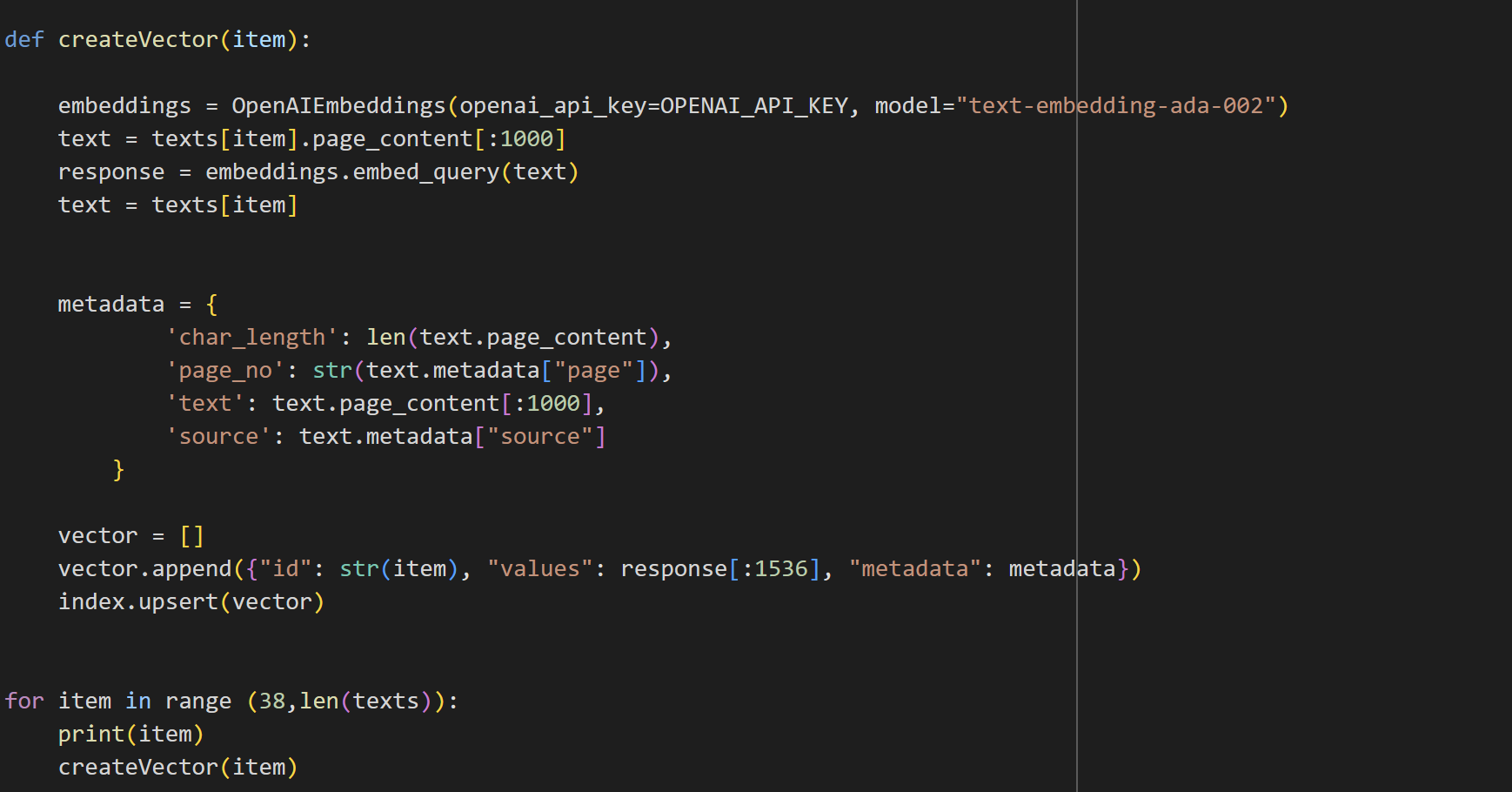


* Create Openai embeddings and create vector db index for storage

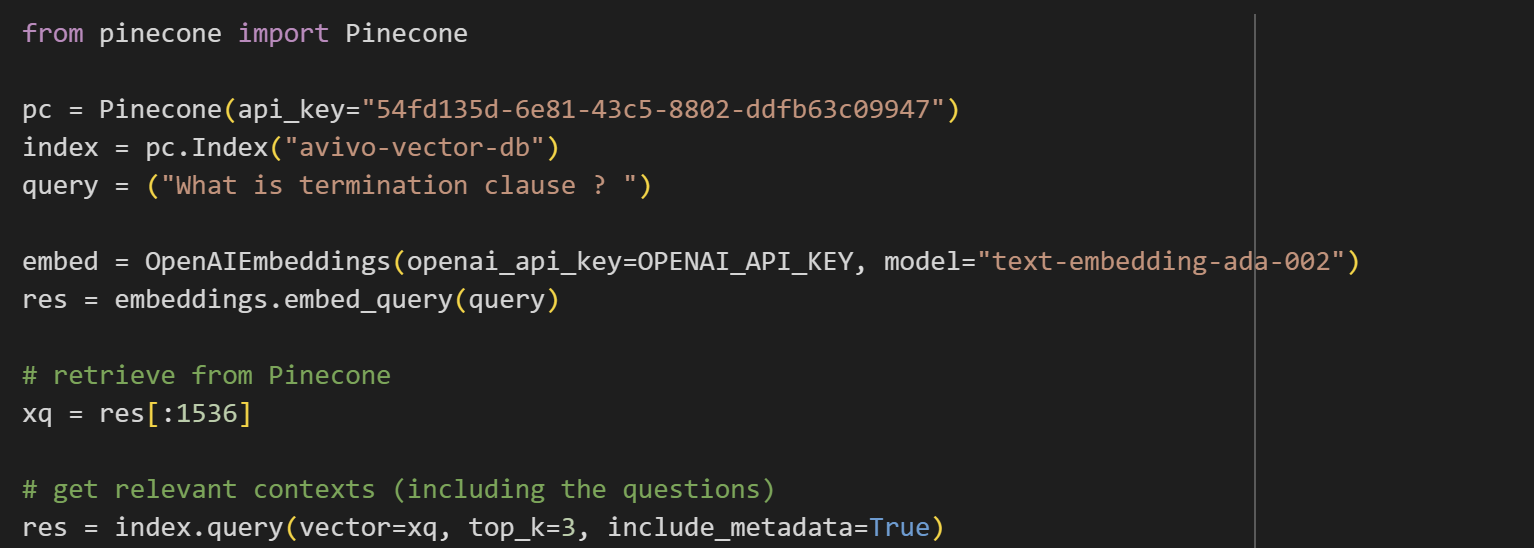








* Retrieve answers querying the bot



# 5. Conclusion

This proof-of-concept study demonstrates the potential of using OpenAI's text-embedded models for summarizing PDF documents. By leveraging advanced NLP techniques and vector database technology, the system can efficiently process and summarize large volumes of text, offering valuable insights from complex documents.

# 6. References

1. [ChatGPT (openai.com)](https://openai.com/chatgpt)
2. [Get started | 🦜️🔗 Langchain](https://python.langchain.com/docs/get_started)
3. [Quickstart - Pinecone](https://docs.pinecone.io/guides/getting-started/quickstart)