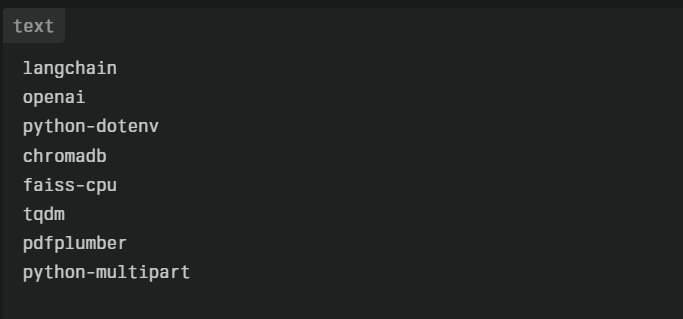
Started a project in VS Code title is RAG-ChatBot   
  
**Creating environment**

* python -m venv Chatbot

**activating environment**

* Chatbot\Scripts\activate

**Now created requirements.txt file**   


Executed command

* pip install -r requirements.txt

steps to form dynamic upload:

1.imports

2.key load

3.llm , embedding

4.prompt

5.INPUT : upload , query

6.If upload 🡪 Rag

7.button 🡪 ret.invoke🡪 print

Code:  
  
import os

from dotenv import load\_dotenv

from langchain\_openai import ChatOpenAI, OpenAIEmbeddings

from langchain\_community.document\_loaders import PyPDFLoader

from langchain\_community.vectorstores import Chroma

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.prompts import ChatPromptTemplate

from langchain.chains.combine\_documents import create\_stuff\_documents\_chain

from langchain.chains.retrieval import create\_retrieval\_chain

# 1. Keys & models

# -----------------------------------------------------------

load\_dotenv()

OPENAI\_API\_KEY = os.getenv("OPENAI\_API\_KEY")

llm = ChatOpenAI(model="gpt-3.5-turbo", api\_key=OPENAI\_API\_KEY)

embeddings = OpenAIEmbeddings(model="text-embedding-3-small",

                              api\_key=OPENAI\_API\_KEY)

# 2. Prompt MUST expose both input vars:  input  and  context

# -----------------------------------------------------------

prompt = ChatPromptTemplate.from\_messages([

    ("system",

     "You are an assistant. Use the following context to answer the user.\n"

     "Context:\n{context}"),

    ("user", "{input}")

])

# 3. Ingest document → split → embed

# -----------------------------------------------------------

docs = PyPDFLoader("EswarReddy\_FullStack(Resume).pdf").load()

splitter = RecursiveCharacterTextSplitter(chunk\_size=300, chunk\_overlap=50)

chunks = splitter.split\_documents(docs)

vector\_db = Chroma.from\_documents(chunks, embeddings)

retriever = vector\_db.as\_retriever()

# 4. Build chains

# -----------------------------------------------------------

combine\_docs\_chain = create\_stuff\_documents\_chain(llm, prompt)   # needs context

rag\_chain          = create\_retrieval\_chain(retriever,           # retriever FIRST

                                           combine\_docs\_chain)   # combine-docs SECOND

# 5. Ask a question

query   = "What is the name and mobile number of the person in the resume? and 5+5?"

result  = rag\_chain.invoke({"input": query})

print(result["answer"])      # <= only the answer