

해당 코드를 app.py라는 파일을 만들어 입력한 뒤 저장해주시면 됩니다

```
In [ ]: #!pip install Langchain streamlit PyPDF2 langchain-openai

import os
from dotenv import load_dotenv
load_dotenv()
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")

from PyPDF2 import PdfReader
import streamlit as st
from langchain_classic.text_splitter import CharacterTextSplitter
from langchain_openai import OpenAIEMBEDDINGS
from langchain_classic import FAISS
from langchain_classic.chains.question_answering import load_qa_chain
from langchain_classic.chat_models import ChatOpenAI
from langchain_classic.callbacks import get_openai_callback

def process_text(text):
    #CharacterTextSplitter를 사용하여 텍스트를 청크로 분할
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)

    #임베딩 처리(벡터 변환), 임베딩은 OpenAI 모델을 사용합니다.
    embeddings = OpenAIEMBEDDINGS(model="text-embedding-ada-002")
    documents = FAISS.from_texts(chunks, embeddings)
    return documents

def main(): #streamlit을 이용한 웹사이트 생성
    st.title("📄 PDF 요약하기")
    st.divider()
```

```

pdf = st.file_uploader('PDF파일을 업로드해주세요', type='pdf')

if pdf is not None:
    pdf_reader = PdfReader(pdf)
    text = "" # 텍스트 변수에 PDF 내용을 저장
    for page in pdf_reader.pages:
        text += page.extract_text()

    documents = process_text(text)
    query = "업로드된 PDF 파일의 내용을 약 3~5문장으로 요약해주세요." # LLM에게 PDF파일 요약 요청

    if query:
        docs = documents.similarity_search(query)
        llm = ChatOpenAI(model="gpt-4.1-mini", temperature=0.1)
        chain = load_qa_chain(llm, chain_type='stuff')

        with get_openai_callback() as cost:
            response = chain.run(input_documents=docs, question=query)
            print(cost)

        st.subheader('--요약 결과--')
        st.write(response)

if __name__ == '__main__':
    main()

```

만든 뒤 터미널에서 해당 폴더의 위치까지 이동한 뒤 streamlit run app.py 명령어로 streamlit을 실행합니다.