PDF 문서 기반 챗봇 만들기

- 디스크에 데이터를 저장하여 프로그램 재 시작시에도 데이터가 유지
- 변경된 파일에 대해서만 추가.
- 파일 해시를 사용하여 변경된 파일만 처리

In [1]:
```
!pip install -Uq openai pypdf langchain langchain_core langchain_openai langchai
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.3/67.3 kB 2.6 MB/s eta 0:00:00
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.4/50.4 kB 469.3 kB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 383.5/383.5 kB 9.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.5/294.5 kB 11.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 19.6 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 401.8/401.8 kB 17.0 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.7/49.7 kB 2.6 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 41.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 604.0/604.0 kB 25.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 43.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.6/94.6 kB 5.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.4/76.4 kB 4.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.0/78.0 kB 4.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 326.6/326.6 kB 16.3 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.6/294.6 kB 15.0 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 35.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 273.8/273.8 kB 14.6 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 42.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.3/49.3 kB 2.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 93.2/93.2 kB 4.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.2/13.2 MB 54.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.5/52.5 kB 3.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 141.9/141.9 kB 9.4 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.4/54.4 kB 2.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.5/54.5 kB 3.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.5/71.5 kB 4.3 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.7/63.7 kB 4.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 3.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 341.4/341.4 kB 18.6 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.4/3.4 MB 78.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 425.7/425.7 kB 24.3 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 164.1/164.1 kB 10.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 2.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 5.8 MB/s eta 0:00:00
Building wheel for pypika (pyproject.toml) ... done
```

In [2]:
```
import os
from openai import OpenAI

def init_api():
    with open("chatgpt_kict2409.env") as env:
       for line in env:
           key, value = line.strip().split("=")
           os.environ[key] = value

init_api()
```

```
# client = OpenAI(api_key  = os.environ.get("API_KEY"))
os.environ["OPENAI_API_KEY"] = os.environ.get("API_KEY")
```

In [17]:

```
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain_openai import OpenAIEmbeddings
from langchain_chroma import Chroma
import os
import hashlib
from datetime import datetime
import json

# Chroma DB를 영구 저장소로 설정
persist_directory = './chroma_db'

# 파일 변경 추적을 위한 파일 경로
tracker_file = 'file_tracker.json'

def get_file_hash(filepath):
    with open(filepath, "rb") as f:
        file_hash = hashlib.md5()
        chunk = f.read(8192)
        while chunk:
            file_hash.update(chunk)
            chunk = f.read(8192)
    return file_hash.hexdigest()

def load_file_tracker():
    if os.path.exists(tracker_file):
        with open(tracker_file, 'r') as f:
            return json.load(f)
    return {}

def save_file_tracker(tracker):
    with open(tracker_file, 'w') as f:
        json.dump(tracker, f)

def process_pdf_files(folder_path):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=50
    )

    embedding_fun = OpenAIEmbeddings()
    db = Chroma(persist_directory=persist_directory, embedding_function=embeddin

    file_tracker = load_file_tracker()

    for filename in os.listdir(folder_path):
        if filename.endswith(".pdf"):
            filepath = os.path.join(folder_path, filename)
            file_hash = get_file_hash(filepath)

            if filepath in file_tracker and file_tracker[filepath]['hash'] == fi
                print(f"Skipping {filename} (no changes)")
                continue

            print(f"Processing {filename}...")
```

```
            raw_documents = PyPDFLoader(filepath).load()
            documents = text_splitter.split_documents(raw_documents)

            existing_docs = db.get(where={"source": filepath})
            if existing_docs['ids']:
                db.delete(ids=existing_docs['ids'])

            db.add_documents(documents)

            file_tracker[filepath] = {
                'hash': file_hash,
                'last_processed': datetime.now().isoformat()
            }

    save_file_tracker(file_tracker)
    return db

# PDF 파일이 저장된 폴더 경로를 설정
folder_path = './pdf_data/'
db = process_pdf_files(folder_path)

# 데이터베이스에서 검색을 수행할 수 있는 retriever 객체 생성
retriever = db.as_retriever(search_kwargs={"k": 10})

print("처리 완료. DB 검색을 위한 준비 완료")
```

```
Skipping 01_데이터의이해및데이터수집.pdf (no changes)
Skipping CHATGPT기본소개_V10_2406.pdf (no changes)
Processing CL01_02_Pandas알아보기_v12_class_230218.pdf...
Skipping CL01_01_python_dataVis_m_ver10_2408.pdf (no changes)
Skipping 01_predict_future_sales_v10_2409.pdf (no changes)
Skipping ChatGPT를활용한프로그래밍(1).pdf (no changes)
처리 완료. DB 검색을 위한 준비 완료
```

In [9]:

```
from langchain_core.runnables import RunnablePassthrough
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import ChatOpenAI
from langchain_core.output_parsers import StrOutputParser

prompt_template = ChatPromptTemplate.from_template(
    "당신은 질문 답변 작업의 영리하고 창의적인 어시스턴트입니다. "
    "다음 문맥을 사용하여 질문에 답하세요. "
    "정확하고 신뢰성 있는 정보를 제공하고, 모르는 내용은 '모르겠습니다'라고 답변하
    "답변은 명확하고 간결하게, 최대 세 문장 이내로 작성하세요. 메타데이터나 추가적
    "한국어로 작성합니다.\n\n"
    "질문: {question}\n"
    "문맥: {context}\n"
    "답변:"
)

chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt_template
    | ChatOpenAI()
    | StrOutputParser()
)

def chat_with_user(user_message):
    ai_message = chain.invoke(user_message)
    return ai_message
```

```
# 대화 루프 시작
while True:
    message = input("USER :(quit or q : 종료)  ")
    if message.lower() == "quit" or message.lower() == "q":
        break

    ai_message = chat_with_user(message)
    print(f" AI : {ai_message}")
```

```
USER :(quit or q : 종료)  AI 기술 구분에 대해 알려주렴.
 AI : AI 기술은 머신러닝, 자연어처리, 자동화 및 로보틱스, Computer Vision 등으로 구분됩니다. 이러한 기술들은 자율주행, 농업, 교육 등 다양한 분야에 활용되고 있습니다.더 자세한 정보는 해당 링크를 참고하세요: https://www.youtube.com/watch?v=z4hGF_eKCxw
USER :(quit or q : 종료)  q
```

In [ ]:

In [ ]: