PDF 문서 기반 챗봇 만들기

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

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/67.3 kB ? eta -:--:--
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.3/67.3 kB 2.7 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 1.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 292.8/292.8 kB 13.4 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 33.1 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 405.1/405.1 kB 30.4 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.5/51.5 kB 4.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 68.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 599.2/599.2 kB 38.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 78.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.6/94.6 kB 8.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 290.0/290.0 kB 21.0 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 375.0/375.0 kB 28.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 39.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 273.8/273.8 kB 21.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.4/76.4 kB 6.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 4.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 318.9/318.9 kB 22.1 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 52.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.3/49.3 kB 2.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 88.9/88.9 kB 4.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.2/13.2 MB 51.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.0/64.0 kB 5.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.5/52.5 kB 3.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 149.7/149.7 kB 9.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 5.0 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 141.9/141.9 kB 10.3 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.3/54.3 kB 4.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.4/71.4 kB 5.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 kB 4.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 3.6 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 341.4/341.4 kB 19.4 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.4/3.4 MB 21.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 425.7/425.7 kB 19.7 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 157.3/157.3 kB 11.4 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 3.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 6.2 MB/s eta 0:00:00
Building wheel for pypika (pyproject.toml) ... done
```

In [2]:
```python
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 [10]:
```
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()
)
```

In [11]:
```
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain_openai import OpenAIEmbeddings
from langchain_chroma import Chroma

# PDF 파일이 저장된 폴더 경로를 설정
folder_path = './pdf_data/'

# 추출된 텍스트를 저장할 리스트
all_texts = []

# 텍스트를 분할할 때 사용할 CharacterTextSplitter 객체 생성
# separator = "\n", # 텍스트를 분할할 때 사용할 구분자
# chunk_size = 1000, # 각 분할된 텍스트의 최대 길이
# chunk_overlap = 50 # 분할된 텍스트 간의 중첩 길이

text_splitter = CharacterTextSplitter(
    separator = "\n",
    chunk_size = 1000,
    chunk_overlap = 50)


# 지정된 폴더 내 모든 파일을 확인
for filename in os.listdir(folder_path):

    # 파일이 PDF 형식인 경우에만 처리
    if filename.endswith(".pdf"):

        # PDF 파일을 로드하여 raw_documents에 저장
        raw_documents = PyPDFLoader(folder_path + '/' + filename).load()

        # 로드된 문서를 분할하여 documents에 저장
        documents = text_splitter.split_documents(raw_documents)
```

```
        # 분할된 텍스트를 리스트에 추가
        all_texts.extend(documents)

# 분할된 텍스트를 임베딩으로 변환하고 Chroma 데이터베이스에 저장
db = Chroma.from_documents(all_texts, OpenAIEmbeddings())

# 데이터베이스에서 검색을 수행할 수 있는 retriever 객체 생성
# search_kwargs: 검색 시 반환할 결과 수를 설정
retriever = db.as_retriever(search_kwargs={"k": 10})
```

In [12]:
```
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)  CHATGPT API는 뭐니?
 AI : ChatGPT API는 OpenAI의 GPT-4 모델을 활용하여 챗봇을 구현하는 데 사용됩니다. 사용자에게 여행 정보를 제공하거나 대화를 생성하는 데 활용됩니다. '모르겠습니다'.
USER :(quit or q : 종료)  q
```

챗봇 만들기

- [도전 1] streamlit을 활용한 앱을 구현해 보자.
- [도전 2] 나의 분야의 챗봇을 만들어보자.