LangChain를 활용한 간단한 챗봇 만들어보기

라이브러리 설치

```
pip install -Uq langchain_core langchain_openai langchain
```

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

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 409.3/409.3 kB 5.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.4/50.4 kB 1.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 28.6 MB/s eta 0:00:00
```

API_Key 인증

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

def init_api():
    with open("chatgpt.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 [3]:
```
from langchain_core.runnables import RunnablePassthrough
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_openai import ChatOpenAI
```

In [4]:
```
# 프롬프트 템플릿 생성
# 템플릿은 system과 human으로 구성되어 있음.
prompt_template = ChatPromptTemplate.from_messages([
    ("system", "안녕하세요. 친구처럼 대화하고 이모티콘을 사용해 주세요 😊"),
    ("human", "{input_text}")
])

# LangChain 체인 생성
# RunnablePassthrough()를 쓰지 않으면 human의 변수명은 user_input으로 고정 사용해
# RunnablePassthrough()를 쓸 경우,  human의 사용자 입력 변수의 이름을 "input_text
# | prompt_template : 생성한 것을 prompt_template 체인에 연결
# | ChatOpenAI()    : OpenAI의 채팅 모델을 체인에 연결
# | StrOutputParser()  : AI의 응답을 문자열로 파싱 - 모델의 출력을 사람이 읽을 수
chain = {"input_text": RunnablePassthrough()} | prompt_template | ChatOpenAI() |


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

```
    print(" AI : ", end="", flush=True)

    for response in chain.stream(message):
        print(response, end="", flush=True)
    print()
```

AI : 안녕! 어떻게 지내? 😊✨
AI : 안녕하세요! 어떻게 지내셨어요? 😊✨
AI : 안녕하세요! 저는 대화를 나누고 질문에 답변하며 정보를 제공할 수 있어요. 무엇이든 물어보세요! 💬✨

실습 및 도전 과제

- [실습 1] Prompt 메시지를 변경해 보자.여행용 챗봇 만들어보기
- [실습 2] Prompt 메시지로 할루시네이션을 없애는 프롬프트를 작성해 보자.
- [도전 1] 이전의 대화를 기능을 추가하는 것을 만들어보자.

Prompt 엔지니어링

In [5]:
```
from langchain_core.runnables import RunnablePassthrough
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_openai import ChatOpenAI

# 할루시네이션 방지 시스템 메시지
prompt_template = ChatPromptTemplate.from_messages([
    ("system",
     "안녕하세요. 대화할 때 정확하고 신뢰할 수 있는 정보만 제공해 주세요. "
     "잘 모르는 내용은 추측하지 말고 '모르겠습니다'라고 답변해 주세요. 😊"),
    ("human", "{input_text}")
])

# LangChain 체인 생성
chain = {"input_text": RunnablePassthrough()} | prompt_template | ChatOpenAI() |

# 대화 루프 시작
while True:
    message = input("YOU :(quit : 종료)  ")
    if message.lower() == "quit":
        break

    print(" AI : ", end="", flush=True)

    for response in chain.stream(message):
        print(response, end="", flush=True)
    print()
```

AI : 안녕하세요! 무엇을 도와드릴까요? 😊
AI : 안녕, 나는 챗봇이야. 대화나 정보 제공에 도움을 줄 수 있어. 함께 어떤 이야기를 나눌까? 😊
AI : 안녕하세요! 챗봇은 다양한 영역에서 도움을 줄 수 있습니다. 주로 FAQ 답변, 예약 서비스, 제품 정보 제공, 일정 확인, 날씨 정보 제공, 번역, 음악 추천, 간단한 질문에 대한 답변 등 다양한 기능을 수행할 수 있습니다. 하지만 몇몇 복잡하고 전문적인 주제에 대해서는 정확한 정보를 제공하기 어려울 수 있습니다.
AI : 별 말씀을요. 궁금한 것이 있으시면 언제든지 물어보세요. 😊

In [ ]: