ChatGPT를 이용해서 챗봇 개발하기

|  |
| --- |
| openai에서 제공하는 api을 기능한 대화형 챗봇을 streamlit 라이브러리를 이용해서 코드를 만들어줘. |
| import streamlit as st  import openai  # OpenAI API 키 설정  openai.api\_key = "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"  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:")  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine="text-davinci-003",  prompt=user\_input,  max\_tokens=500,  temperature=0.7  )  # 챗봇의 응답을 출력합니다.  bot\_response = response['choices'][0]['text'].strip()  st.text("챗봇 응답: " + bot\_response) |
|  |

|  |
| --- |
| 이전에 주고 받은 대화 메시지를 계속 보여주는 기능을 추가해줘 |
| import streamlit as st  import openai  # OpenAI API 키 설정  openai.api\_key = "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"  # 이전 대화 메시지를 저장할 리스트를 초기화합니다.  conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:")  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine="text-davinci-003",  prompt='\n'.join(conversation\_history),  max\_tokens=500,  temperature=0.7  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  conversation\_history.append(f'챗봇: {bot\_response}')  # 대화 기록을 출력합니다.  st.text('\n'.join(conversation\_history)) |
|  |

|  |
| --- |
| 이전 대화 메시지를 상단에서 출력하고, 사용자 메시지를 하단에서 입력 받는 형식으로 화면 레이아웃을 변경해줘 |
| import streamlit as st  import openai  # OpenAI API 키 설정  openai.api\_key = "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"  # 이전 대화 메시지를 저장할 리스트를 초기화합니다.  conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:")  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine="text-davinci-003",  prompt='\n'.join(conversation\_history),  max\_tokens=100,  temperature=0.7  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  conversation\_history.append(f'챗봇: {bot\_response}')  # 대화 기록을 역순으로 정렬하여 상단에 이전 대화 메시지를 표시합니다.  conversation\_history.reverse()  # 이전 대화 메시지를 출력합니다.  st.text('\n'.join(conversation\_history))  # 대화 기록을 다시 정렬하여 리스트의 순서를 복원합니다.  conversation\_history.reverse()  # 사용자 입력 부분은 하단에 배치합니다.  st.text\_input("새로운 질문을 입력하세요:") |
|  |

|  |
| --- |
| 화면 구성을 sidebar와 컨텐츠 영역 이렇게 두개의 영역으로 분리해주고,  sidebar에서 api 키를 입력받아서 변수에 저장하고 코드에서 이를 사용, 화면상에서는 입력한 키 정보가 안보이게 처리해주고,  컨텐츠 영역의 화면은 상단은 이전 대화 내용이 히스토리 형태로 보여주고, 화면 하단에서 사용자 입력받는 형식으로 코드를 다시 작성해줘. |
| import streamlit as st  import openai  # Streamlit 사이드바에 API 키를 입력받습니다.  api\_key = st.sidebar.text\_input("OpenAI API 키 입력", type="password")  # OpenAI API 키를 설정합니다.  openai.api\_key = api\_key  # 이전 대화 메시지를 저장할 리스트를 초기화합니다.  conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:")  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine="text-davinci-003",  prompt='\n'.join(conversation\_history),  max\_tokens=100,  temperature=0.7  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  conversation\_history.append(f'챗봇: {bot\_response}')  # 대화 기록을 역순으로 정렬하여 상단에 이전 대화 메시지를 표시합니다.  conversation\_history.reverse()  # 이전 대화 메시지를 출력합니다.  st.text('\n'.join(conversation\_history))  # 대화 기록을 다시 정렬하여 리스트의 순서를 복원합니다.  conversation\_history.reverse()  # 사용자 입력 부분은 하단에 배치합니다.  st.text\_input("새로운 질문을 입력하세요:") |
|  |

|  |
| --- |
| 이전 대화 내용을 모두 세션 변수에 저장해서 전체 대화 목록을 보여주는 코드로 변경해줘 |
| import streamlit as st  import openai  # Streamlit 사이드바에 API 키를 입력받습니다.  api\_key = st.sidebar.text\_input("OpenAI API 키 입력", type="password")  # OpenAI API 키를 설정합니다.  openai.api\_key = api\_key  # 세션 변수를 사용하여 이전 대화 메시지를 저장합니다.  if 'conversation\_history' not in st.session\_state:  st.session\_state.conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:")  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  st.session\_state.conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine="text-davinci-003",  prompt='\n'.join(st.session\_state.conversation\_history),  max\_tokens=100,  temperature=0.7  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  st.session\_state.conversation\_history.append(f'챗봇: {bot\_response}')  # 전체 대화 목록을 출력합니다.  st.text('\n'.join(st.session\_state.conversation\_history)) |

|  |
| --- |
| 대화 토큰의 숫자를 500으로 설정하도록 코드를 변경해줘 |
| import streamlit as st  import openai  # Streamlit 사이드바에 API 키를 입력받습니다.  api\_key = st.sidebar.text\_input("OpenAI API 키 입력", type="password")  # OpenAI API 키를 설정합니다.  openai.api\_key = api\_key  # 세션 변수를 사용하여 이전 대화 메시지를 저장합니다.  if 'conversation\_history' not in st.session\_state:  st.session\_state.conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:")  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  st.session\_state.conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine="text-davinci-003",  prompt='\n'.join(st.session\_state.conversation\_history),  max\_tokens=500, # 대화 토큰 수를 500으로 설정합니다.  temperature=0.7  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  st.session\_state.conversation\_history.append(f'챗봇: {bot\_response}')  # 전체 대화 목록을 출력합니다.  st.text('\n'.join(st.session\_state.conversation\_history)) |
|  |

|  |
| --- |
| openai api 연동시 모델 등 각종 파라미터를 사용자가 선택할 수 있도록 sidebar 에 보여줘 |
| import streamlit as st  import openai  # Streamlit 사이드바에 API 키를 입력받습니다.  api\_key = st.sidebar.text\_input("OpenAI API 키 입력", type="password")  # OpenAI API 키를 설정합니다.  openai.api\_key = api\_key  # 세션 변수를 사용하여 이전 대화 메시지를 저장합니다.  if 'conversation\_history' not in st.session\_state:  st.session\_state.conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:")  # OpenAI API 매개변수를 설정합니다.  engine = st.sidebar.selectbox("엔진 선택", ["text-davinci-003", "text-curie-001"])  temperature = st.sidebar.slider("온도", min\_value=0.1, max\_value=1.0, value=0.7)  max\_tokens = st.sidebar.slider("최대 토큰 수", min\_value=10, max\_value=1000, value=500)  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  st.session\_state.conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine=engine,  prompt='\n'.join(st.session\_state.conversation\_history),  max\_tokens=max\_tokens,  temperature=temperature  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  st.session\_state.conversation\_history.append(f'챗봇: {bot\_response}')  # 전체 대화 목록을 출력합니다.  st.text('\n'.join(st.session\_state.conversation\_history)) |
|  |

|  |
| --- |
| 사용자가 메시지를 입력하면 사용자 입력 창을 clear 해주고, 대화 내용은 길어지면 자동을 아래로 스크롤 되도록 기능을 추가해줘 |
| import streamlit as st  import openai  # Streamlit 사이드바에 API 키를 입력받습니다.  api\_key = st.sidebar.text\_input("OpenAI API 키 입력", type="password")  # OpenAI API 키를 설정합니다.  openai.api\_key = api\_key  # 세션 변수를 사용하여 이전 대화 메시지를 저장합니다.  if 'conversation\_history' not in st.session\_state:  st.session\_state.conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:", key="user\_input", value="")  # OpenAI API 매개변수를 설정합니다.  engine = st.sidebar.selectbox("엔진 선택", ["text-davinci-003", "text-curie-001"])  temperature = st.sidebar.slider("온도", min\_value=0.1, max\_value=1.0, value=0.7)  max\_tokens = st.sidebar.slider("최대 토큰 수", min\_value=10, max\_value=1000, value=500)  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  st.session\_state.conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine=engine,  prompt='\n'.join(st.session\_state.conversation\_history),  max\_tokens=max\_tokens,  temperature=temperature  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  st.session\_state.conversation\_history.append(f'챗봇: {bot\_response}')  # 전체 대화 목록을 출력합니다.  conversation\_text = '\n'.join(st.session\_state.conversation\_history)  # 대화가 길어질 경우 아래로 자동 스크롤합니다.  st.text\_area("대화 내용", value=conversation\_text, height=400, max\_chars=5000, key="conversation\_area")  # 스크롤을 아래로 이동합니다.  st.markdown("최근 대화로 스크롤 다운", unsafe\_allow\_html=True) |
|  |

|  |
| --- |
| with st.chat\_message(messages.data[0].role):  st.write(messages.data[0].content[0].text.value)  코드를 추가해줘 |
| import streamlit as st  import openai  # Streamlit 사이드바에 API 키를 입력받습니다.  api\_key = st.sidebar.text\_input("OpenAI API 키 입력", type="password")  # OpenAI API 키를 설정합니다.  openai.api\_key = api\_key  # 세션 변수를 사용하여 이전 대화 메시지를 저장합니다.  if 'conversation\_history' not in st.session\_state:  st.session\_state.conversation\_history = []  # Streamlit 앱의 제목을 설정합니다.  st.title("대화형 챗봇")  # 사용자에게 질문을 입력받습니다.  user\_input = st.text\_input("질문을 입력하세요:", key="user\_input", value="")  # OpenAI API 매개변수를 설정합니다.  engine = st.sidebar.selectbox("엔진 선택", ["text-davinci-003", "text-curie-001"])  temperature = st.sidebar.slider("온도", min\_value=0.1, max\_value=1.0, value=0.7)  max\_tokens = st.sidebar.slider("최대 토큰 수", min\_value=10, max\_value=1000, value=500)  # 사용자가 질문을 입력한 경우에만 챗봇의 응답을 생성합니다.  if user\_input:  # 사용자가 입력한 질문을 이전 대화 메시지에 추가합니다.  st.session\_state.conversation\_history.append(f'사용자: {user\_input}')  # OpenAI API를 사용하여 챗봇의 응답을 생성합니다.  response = openai.Completion.create(  engine=engine,  prompt='\n'.join(st.session\_state.conversation\_history),  max\_tokens=max\_tokens,  temperature=temperature  )  # 챗봇의 응답을 이전 대화 메시지에 추가합니다.  bot\_response = response['choices'][0]['text'].strip()  st.session\_state.conversation\_history.append(f'챗봇: {bot\_response}')  # Chat Message로 전체 대화 목록을 출력합니다.  for message in st.session\_state.conversation\_history:  role = "user" if message.startswith("사용자:") else "bot"  content = message.split(": ")[1]    with st.chat\_message(role):  st.write(content) |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |