

GEN AI 인텐시브 과정

강사장철원

Section 0

코스소개

DAY1

DAY2

DAY3

DAY4

DAY5

DAY6

DAY7

DAY8

LLM
Basic
Concept

Transformers
paper
review

Transformers
LangChain
LangGraph

LLM
service
develop

Final Project

□ streamlit 라이브러리 기초

GEN AI 인텐시브 과정

Section 1. streamlit

Section 1-1. streamlit 라이브러리 설치

streamlit 라이브러리 설치

```
(base) C:\Users\stoic>conda activate py3_11_9
```

```
(py3_11_9) C:\Users\stoic>pip install streamlit
```

GEN AI 인텐시브 과정

Section 1. streamlit

Section 1-2. streamlit 라이브러리 Hello world

Section

streamlit 라이브러리 Hello World

Hello World

```
streamlit_test01.py •
streamlit_test01.py > ...
1  import streamlit as st
2
3  def main():
4      st.title('안녕하세요')
5      st.title('Hello World')
6
7  if __name__ == '__main__':
8      main()
```



실행 후 웹브라우저 확인

안녕하세요
Hello World

```
stoic@WIN-7702CJ32RBN MINGW64 /c/Users/stoic/Documents/
$ streamlit run streamlit_test01.py
```

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501

Network URL: http://10.36.204.31:8501

→ 터미널에서 소스코드 파일 경로로 이동한 후

\$ streamlit run [파일명]

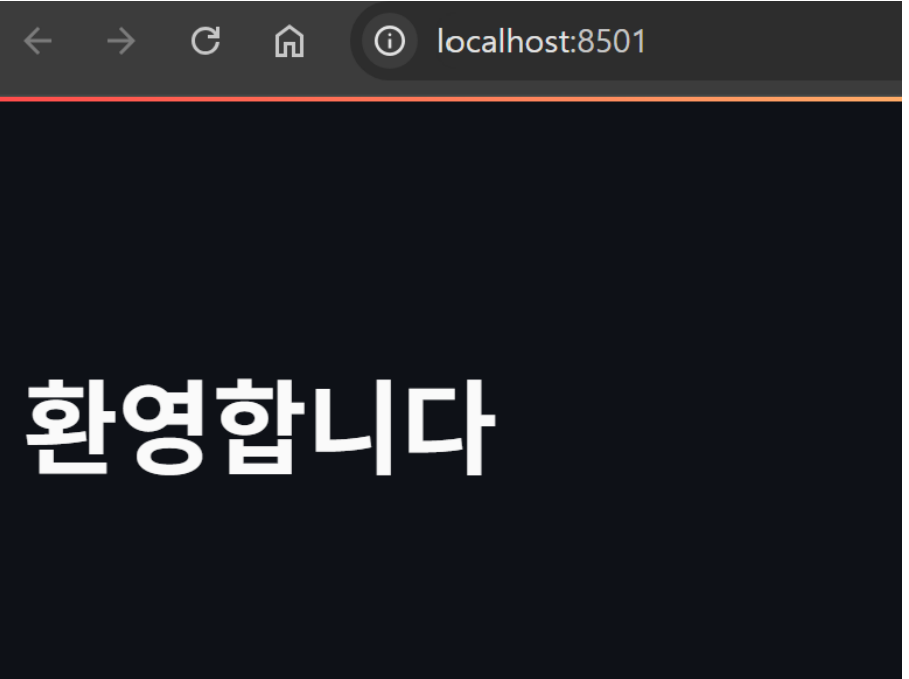
→ streamlit은 기본적으로 8501 포트 사용

main 함수를 사용하지 않아도 가능

```
streamlit_test02.py
1 import streamlit as st
2
3 st.title('환영합니다')
4
```



실행 후 웹브라우저 확인



```
stoic@WIN-7702CJ32RBN MINGW64 /c/Users/stoic/Documents/work/
$ streamlit run streamlit_test02.py
```

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501

Network URL: http://10.36.204.31:8501

→ 터미널에서 소스코드 파일 경로로 이동한 후

\$ streamlit run [파일명]

→ streamlit은 기본적으로 8501 포트 사용

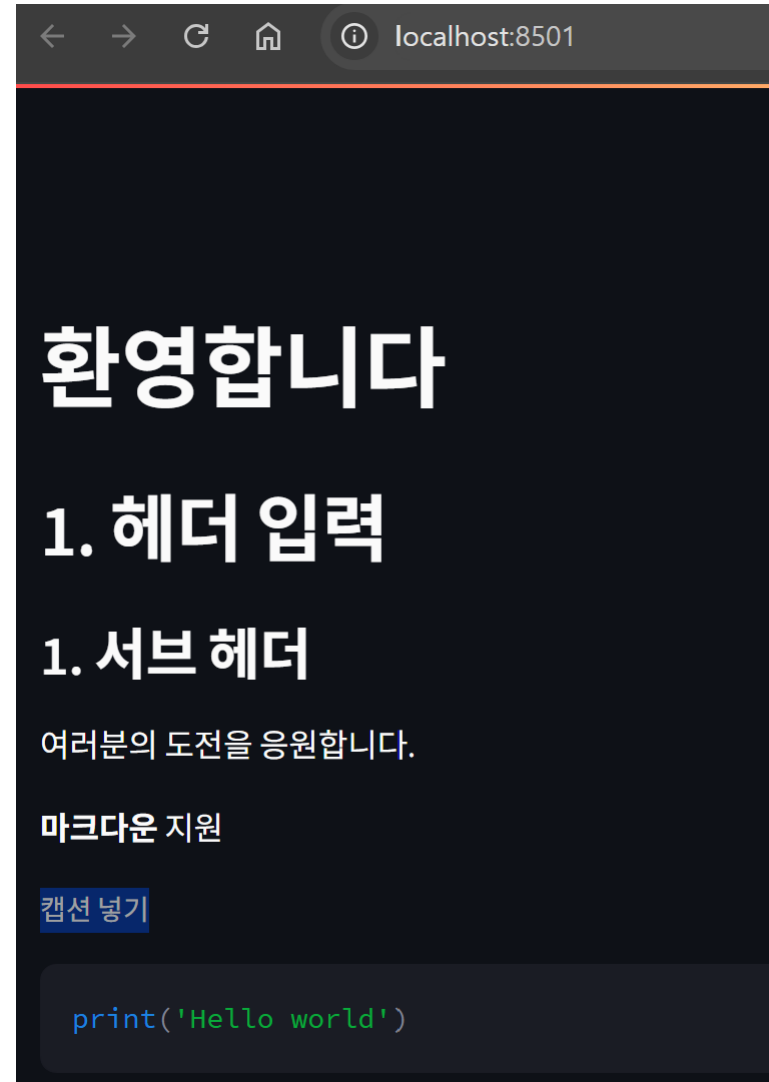
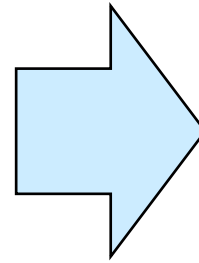
GEN AI 인텐시브 과정

Section 1. streamlit

Section 1-3. streamlit 라이브러리 기초

1. 텍스트 입력 하기

```
streamlit_test03.py
1  import streamlit as st
2
3  st.title('환영합니다')
4  st.header('1. 헤더 입력')
5  st.subheader('1. 서브 헤더')
6
7  st.write('여러분의 도전을 응원합니다.')
8  st.markdown('**마크다운** 지원')
9  st.caption('캡션 넣기')
10 st.code("print('Hello world')", language="python")
11
```

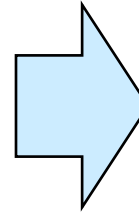


Section

streamlit 라이브러리 기초

streamlit_test04.py > ...

```
1  import streamlit as st
2
3  st.header('2. 사용자 입력 받기')
4
5  name = st.text_input("이름을 입력하세요:")
6  st.write(f"입력한 이름: {name}")
7
8  sop = st.text_area("자기소개를 입력하세요:")
9  st.write(f"자기소개: {sop}")
10
11 age = st.number_input("나이를 입력하세요.:", min_value=0, max_value=100, step=1)
12 st.write(f"나이: {age}")
13
14 birth = st.date_input("생년월일을 선택하세요:")
15 st.write(f"선택한 날짜: {birth}")
16
17 my_color = st.selectbox("좋아하는 색상을 선택하세요:", ["빨강", "파랑", "초록", "노랑"])
18 st.write(f"선택한 색상: {my_color}")
19
20 hobby = st.multiselect("취미를 선택하세요:", ["독서", "운동", "게임", "요리"])
21 st.write(f"선택한 취미: {hobby}")
22
23 gender = st.radio("성별을 선택하세요:", ["남성", "여성", "비공개"])
24 st.write(f"선택한 성별: {gender}")
25
26 agree = st.checkbox("이용 약관에 동의합니다.")
27 if agree:
28     st.write("약관에 동의하셨습니다!")
29
30 score = st.slider("점수를 선택하세요:", min_value=0, max_value=100)
31 st.write(f"선택한 점수: {score}")
32
```



2. 사용자 입력 받기

이름을 입력하세요:

입력한 이름:

자기소개를 입력하세요:

자기소개:

나이를 입력하세요.:

0

나이: 0

생년월일을 선택하세요:

2025/02/15

선택한 날짜: 2025-02-15

좋아하는 색상을 선택하세요:

빨강

선택한 색상: 빨강

취미를 선택하세요:

Choose an option

선택한 취미: []

성별을 선택하세요:

☒ 남성

☐ 여성

☐ 비공개

선택한 성별: 남성

☐ 이용 약관에 동의합니다.

점수를 선택하세요:

0

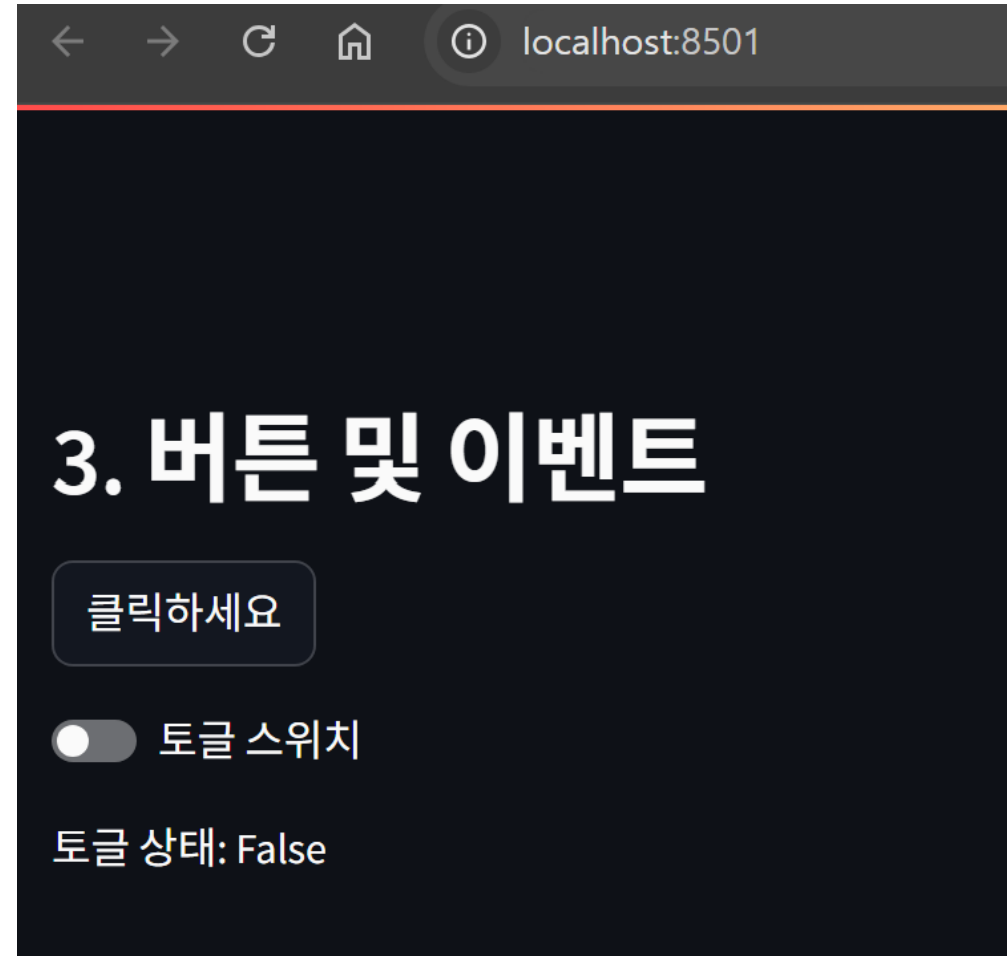
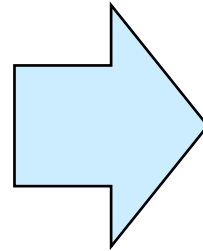
0

선택한 점수: 0

100

2. 버튼 & 이벤트

```
streamlit_test05.py > ...  
1  import streamlit as st  
2  
3  
4  st.header("3. 버튼 및 이벤트")  
5  
6  if st.button("클릭하세요"):  
7      st.write("버튼이 클릭되었습니다!")  
8  
9  toggle_state = st.toggle("토글 스위치")  
10 st.write(f"토글 상태: {toggle_state}")  
11
```



Section

streamlit 라이브러리 기초

streamlit_test06.py > ...

```
1 import streamlit as st
2 import pandas as pd
3 import numpy as np
4
5 st.header("4. 데이터 시각화")
6
7 df = pd.DataFrame(np.random.randn(5, 3), columns=["A", "B", "C"])
```

```
9 st.write(df)
10 st.dataframe(df)
```

```
12 col1, col2 = st.columns(2)
```

```
14 with col1:
15     st.write(df)
```

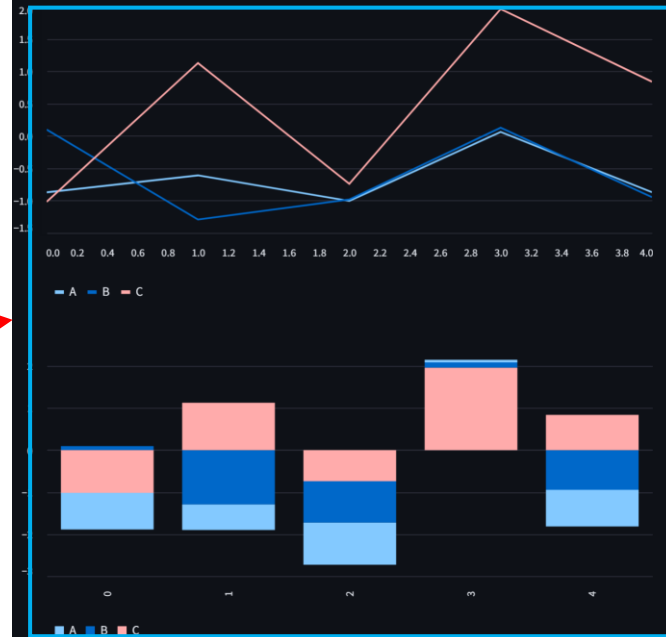
```
17 with col2:
18     st.write(df)
```

```
20 st.line_chart(df)
21 st.bar_chart(df)
```

4. 데이터 시각화

	A	B	C
0	-0.8717	0.0964	-1.0185
1	-0.6101	-1.2923	1.1262
2	-1.0057	-0.9829	-0.741
3	0.0615	0.1266	1.961
4	-0.8697	-0.9467	0.8366

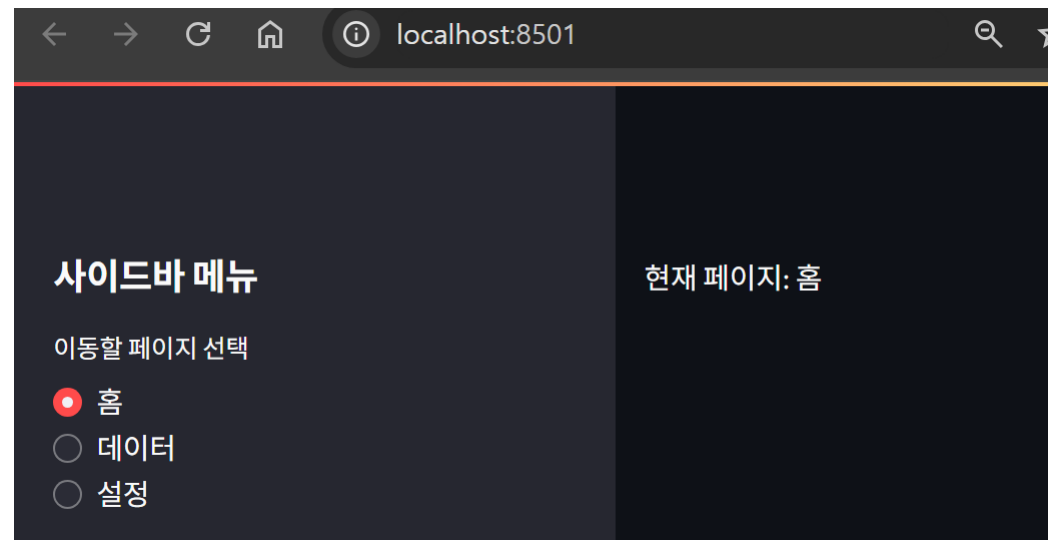
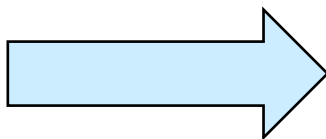
	A	B	C
0	-0.8717	0.0964	-1.0185
1	-0.6101	-1.2923	1.1262
2	-1.0057	-0.9829	-0.741
3	0.0615	0.1266	1.961
4	-0.8697	-0.9467	0.8366



3. 사이드바

streamlit_test07.py > ...

```
1 import streamlit as st
2
3 st.sidebar.header("사이드바 메뉴")
4 page = st.sidebar.radio("이동할 페이지 선택", ["홈", "데이터", "설정"])
5 st.write(f"현재 페이지: {page}")
6
7
```



4. 파일 업로드 & 다운로드

streamlit_test08.py > ...

```
1 import streamlit as st
2
3
4 st.header("6. 파일 업로드 및 다운로드")
5
6 uploaded_file = st.file_uploader("파일을 업로드하세요.", type=["csv", "txt", "png", "jpg"])
7 if uploaded_file is not None:
8     st.write("업로드한 파일 이름:", uploaded_file.name)
9
10 st.download_button(label="샘플 다운로드", data="Hello, Streamlit!", file_name="sample.txt")
```



6. 파일 업로드 및 다운로드

파일을 업로드하세요.



Drag and drop file here

Limit 200MB per file • CSV, TXT, PNG, JPG, JPEG

Browse files

샘플 다운로드

4. 파일 업로드 & 다운로드

6. 파일 업로드 및 다운로드

파일을 업로드하세요.

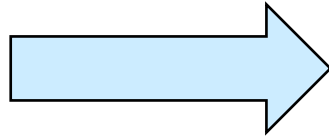


Drag and drop file here

Limit 200MB per file • CSV, TXT, PNG, JPG, JPEG

Browse files

샘플 다운로드



업로드된 파일은 메모리(RAM)상에서 임시적으로 관리
(물리 디스크에는 저장되지 않음)

6. 파일 업로드 및 다운로드

파일을 업로드하세요.



Drag and drop file here

Limit 200MB per file • CSV, TXT, PNG, JPG, JPEG

Browse files



text01.txt 21.0B



업로드한 파일 이름: text01.txt

샘플 다운로드

Section

streamlit 라이브러리 기초

5. 대화형 챗봇

다음 페이지에 자세히

사용자 메시지 추가 & 출력

챗봇 메시지 추가 & 출력

```
streamlit_test09.py > ...
1  import streamlit as st
2
3  # 대화 이력 저장용 세션 상태 초기화
4  if "messages" not in st.session_state:
5      st.session_state.messages = []
6
7  st.title("챗봇 데모")
8  st.write("대화형 챗봇 입니다.")
9
10
11 # 기존 메시지 출력
12 for msg in st.session_state.messages:
13     with st.chat_message(msg["role"]):
14         st.markdown(msg["content"])
15
16 # 사용자 입력 받기
17 user_input = st.chat_input("메시지를 입력하세요")
18
19 if user_input:
20     # 사용자 메시지 추가 및 출력
21     st.session_state.messages.append({"role": "user", "content": user_input})
22     with st.chat_message("user"):
23         st.markdown(user_input)
24
25     # 챗봇 응답
26     bot_response = f"저도 따라할래요 '{user_input}'"
27
28     # 챗봇 메시지 추가 및 출력
29     st.session_state.messages.append({"role": "assistant", "content": bot_response})
30     with st.chat_message("assistant"):
31         st.markdown(bot_response)
```

} 세션 상태 초기화

} 기존 메시지 출력

사용자 입력

5. 대화형 챗봇

다음 페이지에 자세히

```
streamlit_test09.py > ...  
1  import streamlit as st  
2  
3  # 대화 이력 저장용 세션 상태 초기화  
4  if "messages" not in st.session_state:  
5      | st.session_state.messages = []  
6  
7  st.title("챗봇 데모")  
8  st.write("대화형 챗봇 입니다.")
```

딕셔너리처럼 동작하는 세션 전용 공간

`st.session_state`

사용자 세션 별로 상태(state)를 저장하고
관리할 수 있게 해주는 기능



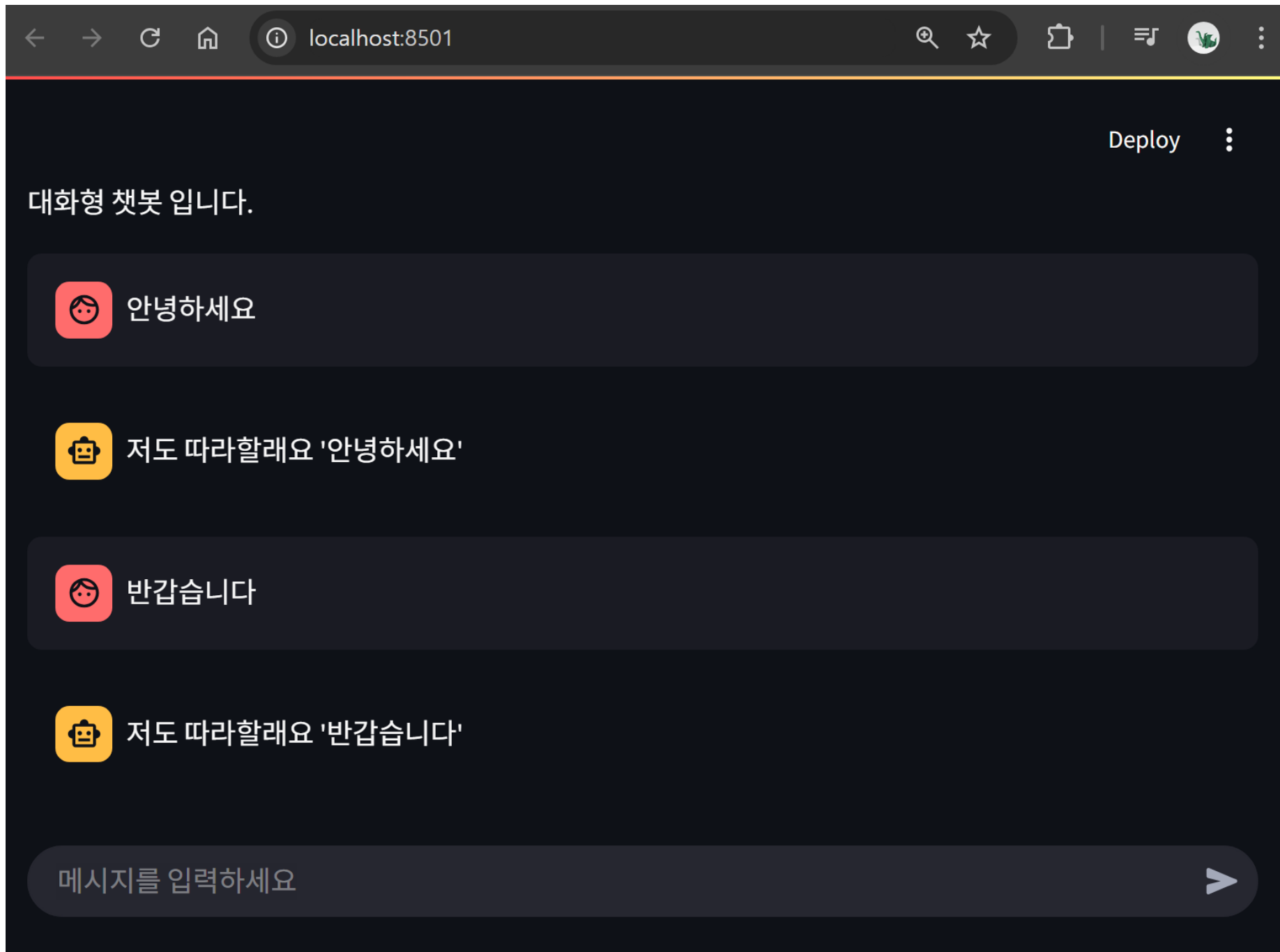
입력값, 대화 이력 등 유지 가능

딕셔너리처럼 키값 쌍으로 접근 OK
속성으로 접근해도 OK

사용 예제

```
st.session_state["name"] = "홍길동"  
st.session_state.name = "홍길동"
```

5. 대화형 챗봇



GEN AI 인텐시브 과정

Section 1. streamlit

Section 1-4. streamlit을 활용한 앱 만들기 기초

main 함수 사용하는 버전(전체 코드)

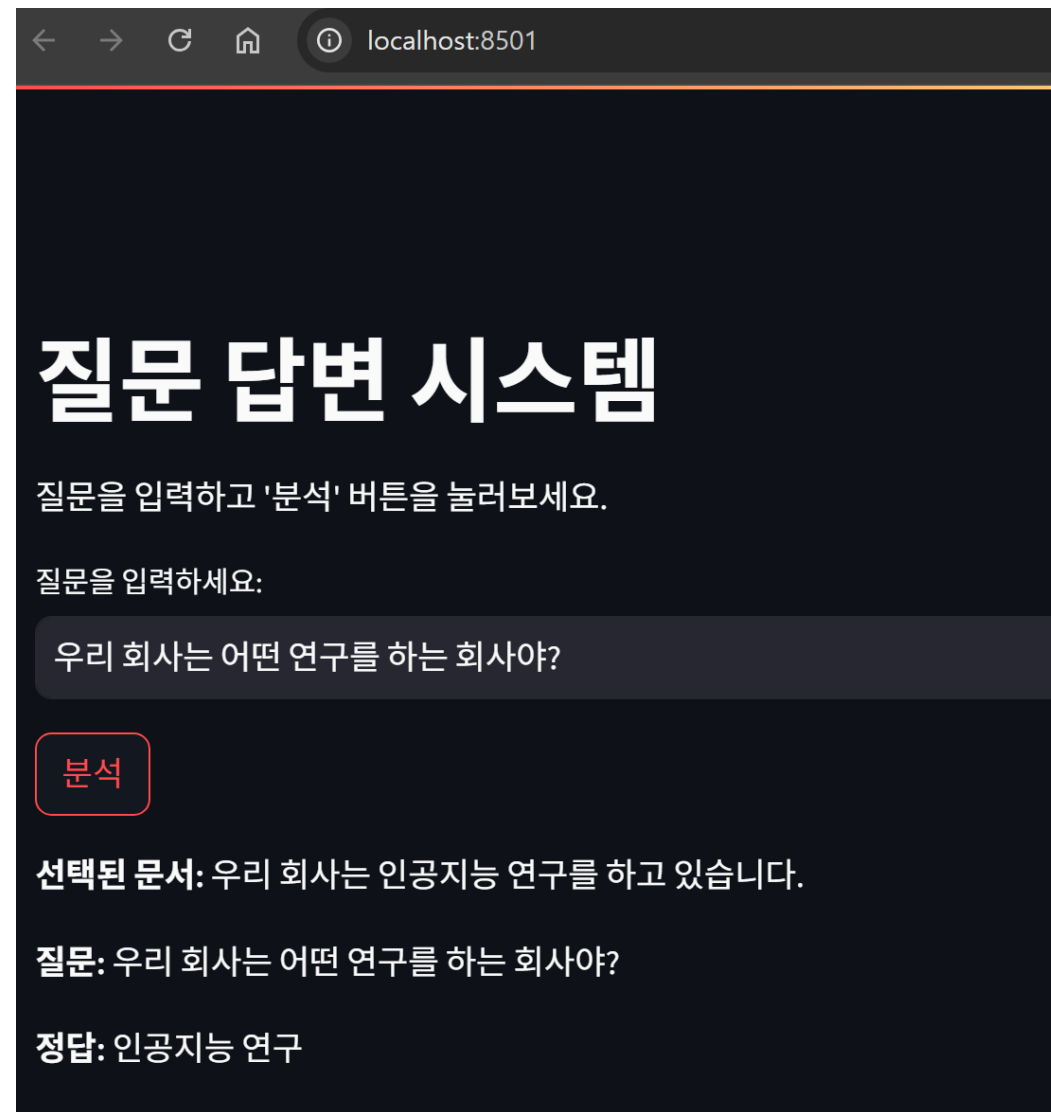
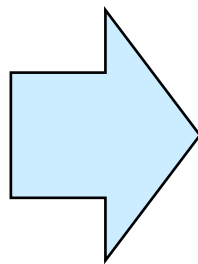
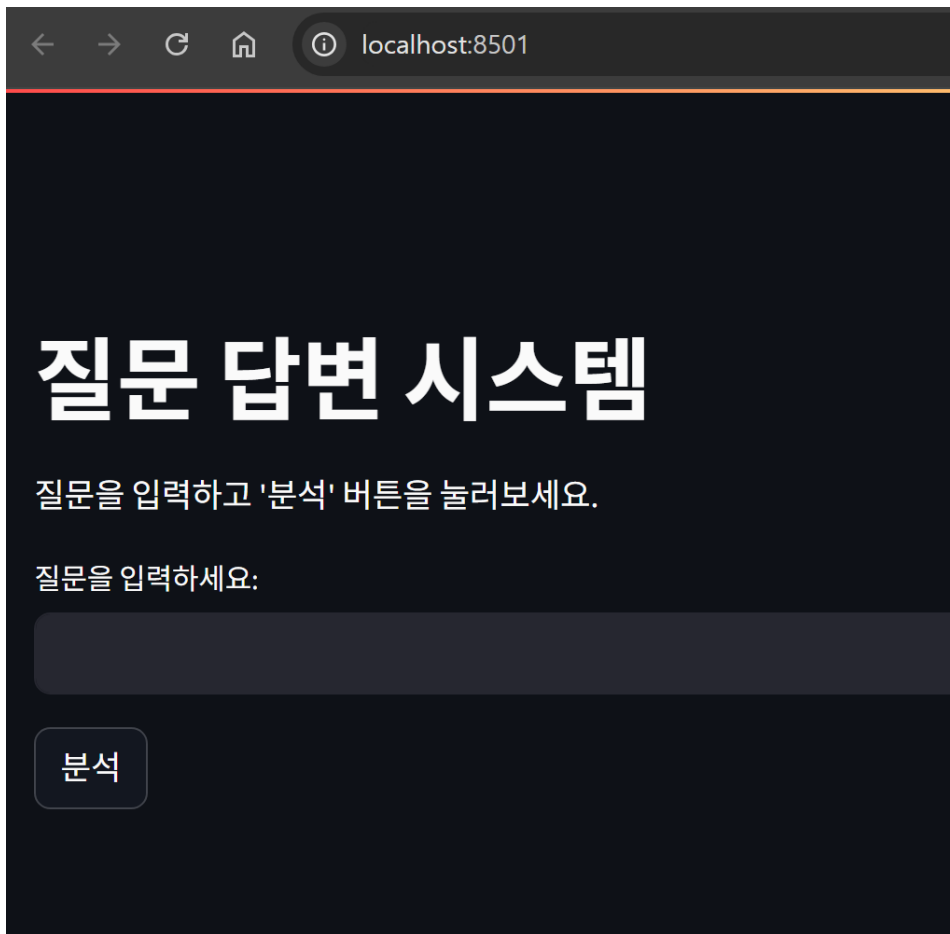
streamlit_test10.py > ...

```
1  import streamlit as st
2  from transformers import AutoTokenizer, AutoModelForQuestionAnswering
3  import torch
4
5  # 모델과 토크나이저 로드
6  model_name = "monologg/koelectra-base-v3-finetuned-korquad"
7  tokenizer = AutoTokenizer.from_pretrained(model_name)
8  model = AutoModelForQuestionAnswering.from_pretrained(model_name)
9
10 # 예제 문서 데이터
11 documents = [
12     "우리 회사는 인공지능 연구를 하고 있습니다.",
13     "우리 제품은 2023년에 출시되었습니다."
14 ]
15
16 def select_best_context(documents, question):
17     result = []
18     for doc in documents:
19         doc_bool = []
20         for word in question.split():
21             doc_bool.append(word in doc)
22         doc_score = sum(doc_bool)
23         result.append(doc_score)
24     best_context_idx = result.index(max(result))
25     best_context = documents[best_context_idx]
26     return best_context
27
```

main 함수 사용하는 버전(전체 코드)

```
29 def get_answer(question):
30     best_context = select_best_context(documents, question)
31     inputs = tokenizer(question, best_context, return_tensors="pt")
32
33     with torch.no_grad():
34         outputs = model(**inputs)
35
36     start_index = torch.argmax(outputs.start_logits)
37     end_index = torch.argmax(outputs.end_logits) + 1
38     answer = tokenizer.decode(inputs["input_ids"][0][start_index:end_index], skip_special_tokens=True)
39
40     return best_context, answer
41
42 def main():
43     st.title("질문 답변 시스템")
44     st.write("질문을 입력하고 '분석' 버튼을 눌러보세요.")
45
46     # 사용자 입력
47     question = st.text_input("질문을 입력하세요:")
48
49     if st.button("분석"):
50         if question:
51             best_context, answer = get_answer(question)
52             st.write(f"**선택된 문서:** {best_context}")
53             st.write(f"**질문:** {question}")
54             st.write(f"**정답:** {answer}")
55         else:
56             st.warning("질문을 입력해주세요!")
57
58 if __name__ == "__main__":
59     main()
```

main 함수 사용하는 버전

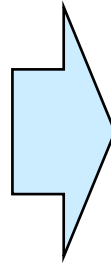


main 함수 사용하지 않는 버전

```

42 def main():
43     st.title("질문 답변 시스템")
44     st.write("질문을 입력하고 '분석' 버튼을 눌러보세요.")
45
46     # 사용자 입력
47     question = st.text_input("질문을 입력하세요:")
48
49     if st.button("분석"):
50         if question:
51             best_context, answer = get_answer(question)
52             st.write(f"**선택된 문서:** {best_context}")
53             st.write(f"**질문:** {question}")
54             st.write(f"**정답:** {answer}")
55         else:
56             st.warning("질문을 입력해주세요!")
57
58 if __name__ == "__main__":
59     main()

```



```

43     st.title("질문 답변 시스템")
44     st.write("질문을 입력하고 '분석' 버튼을 눌러보세요.")
45
46     # 사용자 입력
47     question = st.text_input("질문을 입력하세요:")
48
49     if st.button("분석"):
50         if question:
51             best_context, answer = get_answer(question)
52             st.write(f"**선택된 문서:** {best_context}")
53             st.write(f"**질문:** {question}")
54             st.write(f"**정답:** {answer}")
55         else:
56             st.warning("질문을 입력해주세요!")

```


감사합니다.

Q & A