

챗봇 웹서버 구현

1. Flask 웹 서버 구현

1. Flask 웹서버 구현

Flask 웹 프레임워크

Flask는 파이썬으로 웹사이트나 웹 애플리케이션을 쉽게 만들 수 있게 도와주는 도구이다. 아주 가볍고 단순해서, 작은 프로젝트부터 큰 프로젝트까지 유연하게 사용할 수 있다

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, World!"

if __name__ == '__main__':
    app.run(debug=True)
```

1. Flask 웹서버 구현

1. Flask 설치

먼저 Flask를 설치해야 합니다. Python의 패키지 관리자인 `pip`을 사용하여 Flask를 설치할 수 있습니다:

```
bash
```

[코드 복사](#)

```
pip install Flask
```

1. Flask 웹서버 구현

Flask 웹서버 구현 코드

```
from flask import Flask, request, jsonify
```

```
app = Flask(__name__)
```

```
@app.route('/')  
def index():
```

```
    return "Flask Chatbot Server"
```

```
@app.route('/chat', methods=['POST'])  
def chat():
```

```
    user_message = request.json.get("message")
```

1. Flask 웹서버 구현

```
# 간단한 응답 로직
if user_message:
    response_message = generate_response(user_message)
else:
    response_message = "Please send a message."

return jsonify({"response": response_message})
```

1. Flask 웹서버 구현

```
def generate_response(user_message):  
    # 여기에서 더 복잡한 로직이나 모델을 사용할 수 있습니다.  
    if "hello" in user_message.lower():  
        return "Hi there! How can I help you today?"  
    elif "bye" in user_message.lower():  
        return "Goodbye! Have a nice day!"  
    else:  
        return "I'm sorry, I don't understand that."  
  
if __name__ == '__main__':  
    app.run(debug=True)
```

1. Flask 웹서버 구현

Flask 애플리케이션 구현

```
from flask import Flask, request, jsonify
```

```
app = Flask(__name__)
```

```
@app.route('/')  
def index():
```

```
    return "Flask Chatbot Server"
```

```
@app.route('/chat', methods=['POST'])  
def chat():
```

```
    user_message = request.json.get("message")
```


1. Flask 웹서버 구현

클라이언트 테스트 구현 코드

```
import requests
```

```
url = "http://127.0.0.1:5000/chat"
```

```
data = {"message": "Hello"}
```

```
response = requests.post(url, json=data)
```

```
print(response.json()) # {"response": "Hi there! How can I help you today?"}
```

1. Flask 웹서버 구현

서버 실행 후 클라이언트 테스트 프로그램 실행 화면

```
[1]: import requests

url = "http://127.0.0.1:5000/chat"
data = {"message": "hello"}

response = requests.post(url, json=data)
print(response.json()) # {"response": "Hi there! How can I help you today?"}

{'response': 'Hi there! How can I help you today?'}
```

1. Flask 웹서버 구현

Flask, request, jsonify 모듈

1. 모듈 импорт

python

코드 복사

```
from flask import Flask, request, jsonify
```

- **Flask:** Flask는 파이썬으로 작성된 경량 웹 애플리케이션 프레임워크입니다.
- **request:** 클라이언트로부터 전송된 요청 데이터를 다루기 위한 객체입니다. 주로 요청의 데이터(body, headers 등)를 가져올 때 사용합니다.
- **jsonify:** 파이썬의 딕셔너리나 리스트를 JSON 형식으로 변환하여 클라이언트에 응답할 때 사용합니다.

1. Flask 웹서버 구현

2. Flask 애플리케이션 생성

python

 코드 복사

```
app = Flask(__name__)
```

- ``app = Flask(__name__)``: Flask 애플리케이션 인스턴스를 생성합니다.
``__name__``은 현재 모듈의 이름을 Flask에 전달하는데, 이로 인해 Flask는 현재 실행 중인 파일을 기준으로 경로 등을 설정하게 됩니다.


1. Flask 웹서버 구현

3. 기본 루트 엔드포인트 정의

python

코드 복사

```
@app.route('/')  
def index():  
    return "Flask Chatbot Server"
```

- `@app.route('/')`: Flask에게 URL 경로가 `'/'` 일 때 이 함수가 호출되도록 지정합니다. 이는 웹 애플리케이션의 기본 페이지로, 브라우저에서 서버의 기본 URL(`http://127.0.0.1:5000/`)에 접속하면 이 함수가 실행됩니다.
- `def index()`: `'/'` 경로에 대한 요청이 있을 때 호출되는 함수로, 단순히 텍스트 `"Flask Chatbot Server"`  반환합니다. 이 텍스트가 웹 페이지에 표시됩니다.

1. Flask 웹서버 구현

4. 챗봇 엔드포인트 정의

python

📄 코드 복사

```
@app.route('/chat', methods=['POST'])
def chat():
    user_message = request.json.get("message")

    if user_message:
        response_message = generate_response(user_message)
    else:
        response_message = "Please send a message."

    return jsonify({"response": response_message})
```



1. Flask 웹서버 구현

- `@app.route('/chat', methods=['POST'])`: 이 부분은 `/chat` 경로로 POST 요청이 들어올 때 이 함수(`chat`)가 호출되도록 Flask에 알립니다.
- `request.json.get("message")`: 클라이언트가 보낸 JSON 데이터에서 `"message"`라는 키에 해당하는 값을 가져옵니다. 사용자가 보낸 메시지를 얻는 데 사용됩니다.
- `if user_message:`: 사용자가 메시지를 보냈는지 확인합니다. 메시지가 존재하면 `generate_response()` 함수를 호출하여 사용자의 메시지에 대한 응답을 생성합니다.
- `jsonify({"response": response_message})`: 응답 메시지를 JSON 형식으로 변환하여 클라이언트에 반환합니다.

5. 응답 생성 함수 정의

python

 코드 복사

```
def generate_response(user_message):  
    if "hello" in user_message.lower():  
        return "Hi there! How can I help you today?"  
    elif "bye" in user_message.lower():  
        return "Goodbye! Have a nice day!"  
    else:  
        return "I'm sorry, I don't understand that."
```


1. Flask 웹서버 구현

- ``def generate_response(user_message)``: 사용자의 메시지를 받아서 적절한 응답을 생성하는 함수입니다.
- ``user_message.lower()``: 사용자가 보낸 메시지를 소문자로 변환하여 비교를 용이하게 합니다.
- 여러 조건을 사용하여 사용자의 메시지에 따라 적절한 응답을 반환합니다. 예를 들어, 메시지에 "hello"가 포함되어 있으면 인사말을, "bye"가 포함되어 있으면 작별 인사를 반환합니다. 그 외의 경우에는 이해하지 못한다는 메시지를 반환합니다.

1. Flask 웹서버 구현

6. 애플리케이션 실행

python

 코드 복사

```
if __name__ == '__main__':  
    app.run(debug=True)
```

- ``if __name__ == '__main__':``: 이 코드는 Python에서 모듈이 직접 실행될 때만 실행되도록 보장합니다. 즉, 이 파일이 다른 모듈에 의해 import되지 않고 직접 실행될 때 Flask 애플리케이션이 실행됩니다.
- ``app.run(debug=True)``: Flask 개발 서버를 실행합니다. ``debug=True``는 디버그 모드로 실행하겠다는 의미로, 코드를 수정하면 자동으로 서버가 다시 시작되고, 오류가 발생했을 때 상세한 오류 메시지를 웹 페이지에서 확인할 수 있습니다.

1. Flask 웹서버 구현

Flask HTML 템플릿 사용 웹서버 구현 (배포 소스 참조)

```
from flask import Flask, request, jsonify, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')
```

1. Flask 웹서버 구현

templates/index.html (HTML 파일) (배포 소스 참조)

html

 코드 복사

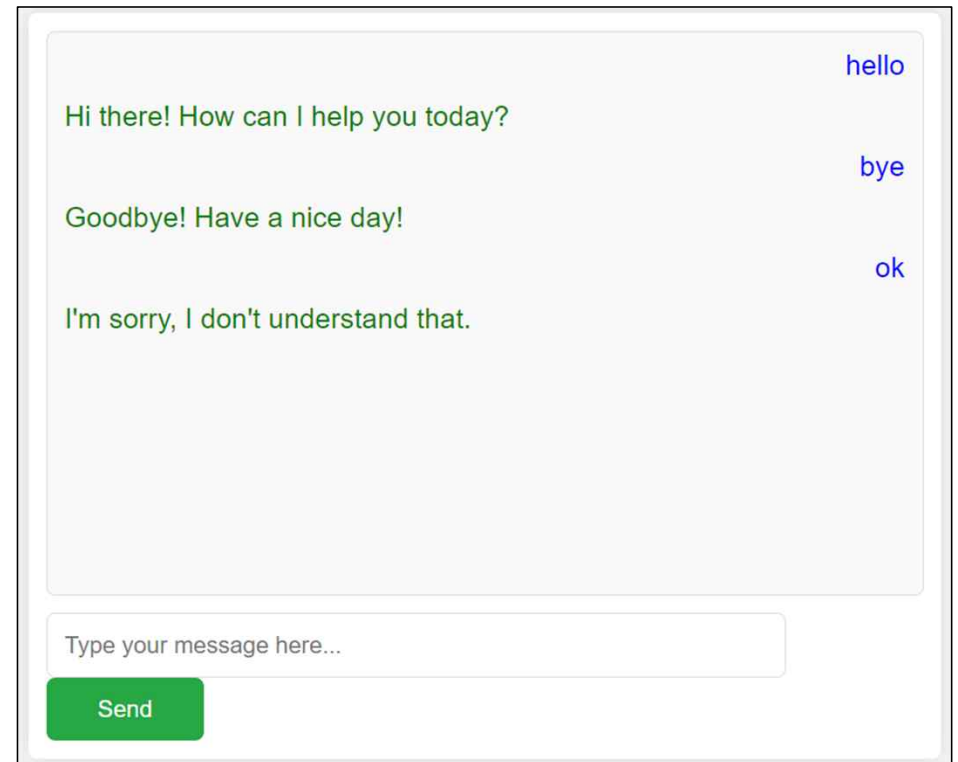
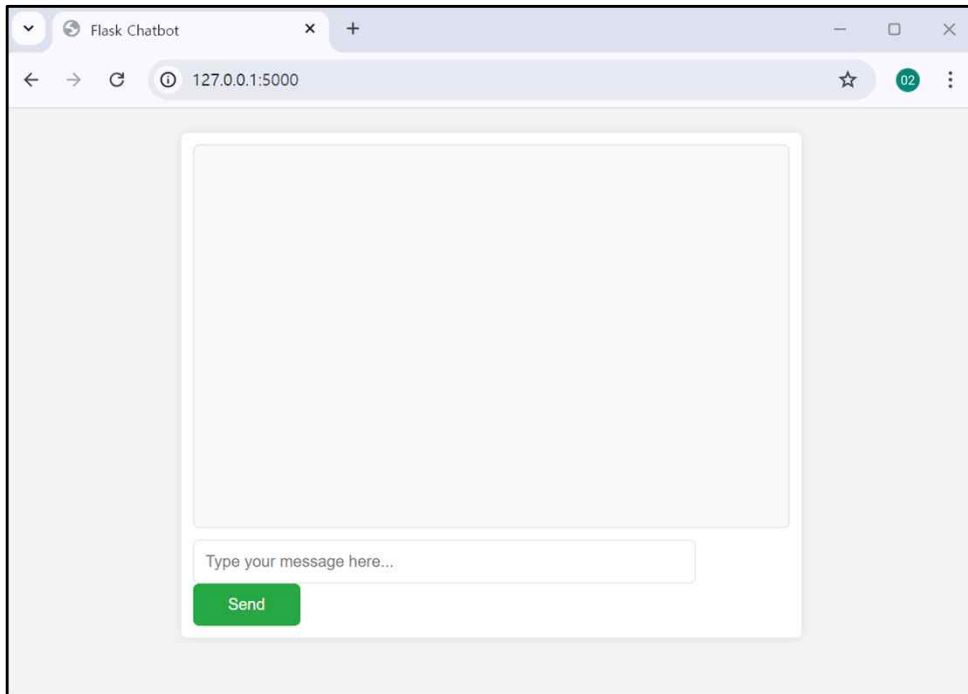
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flask Chatbot</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 20px;
      background-color: #f4f4f4;
    }
  </style>
</head>
<body>
  <div class="chat-container">
    <div class="chat-header">
      <h2>Flask Chatbot</h2>
    </div>
    <div class="chat-messages">
      <div class="message">
        <div class="avatar"></div>
        <div class="text">Hello! I am a Flask Chatbot.</div>
      </div>
    </div>
    <div class="chat-input">
      <input type="text" value="Enter your message here" />
      <button type="button" value="Send" />
    </div>
  </div>
</body>
</html>
```

1. Flask 웹서버 구현

- ``index.html``: 이 HTML 파일은 간단한 챗봇 UI를 제공합니다. 사용자가 입력한 메시지를 서버에 전송하고, 서버로부터 받은 응답을 웹 페이지에 표시합니다.
- ``#user-input``: 사용자가 메시지를 입력하는 텍스트 필드입니다.
- ``#send-button``: 사용자가 입력한 메시지를 서버로 전송하는 버튼입니다.
- ``#messages``: 사용자의 메시지와 서버의 응답을 표시하는 영역입니다.
- **JavaScript 코드:**
 - 사용자가 메시지를 입력하고 "Send" 버튼을 클릭하면, JavaScript 코드가 메시지를 서버에 POST 요청으로 보냅니다.
 - 서버의 응답을 받아서 ``#messages`` 영역에 새로운 메시지로 추가합니다.

1. Flask 웹서버 구현

Flask HTML 템플릿 사용 웹페이지 화면



2. Flask와 OpenAI 연동하기

2. Flask와 OpenAI 연동하기

OpenAI API 사용 Flask 웹 서버 소스 코드

```
import os
from flask import Flask, render_template, request, jsonify
import openai
from openai import OpenAI

app = Flask(__name__)

# OpenAI API 키 설정
API_KEY = os.getenv("OPENAI_API_KEY")
client = OpenAI(api_key=API_KEY)

@app.route("/")
def home():
    return render_template("index.html")
```


2. Flask와 OpenAI 연동하기

OpenAI API 사용 Flask 웹 서버 소스 코드

```
@app.route("/get_response", methods=["POST"])
def get_response():
    user_message = request.json.get("message")

    # OpenAI 모델에 요청
    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[{"role": "user", "content": user_message}],
        max_tokens=300,
        n=1,
        stop=None,
        temperature=0.7,
    )

    bot_response = response.choices[0].message.content
    return jsonify({"response": bot_response})

if __name__ == "__main__":
    app.run(debug=False)
```

2. Flask와 OpenAI 연동하기

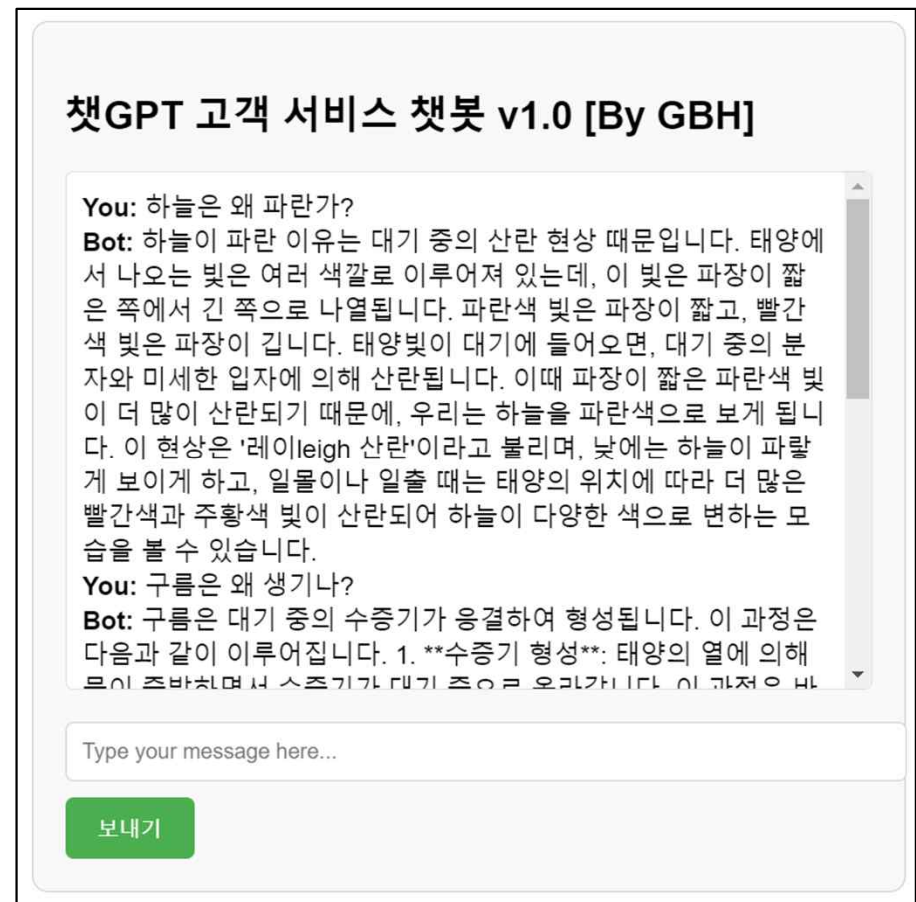
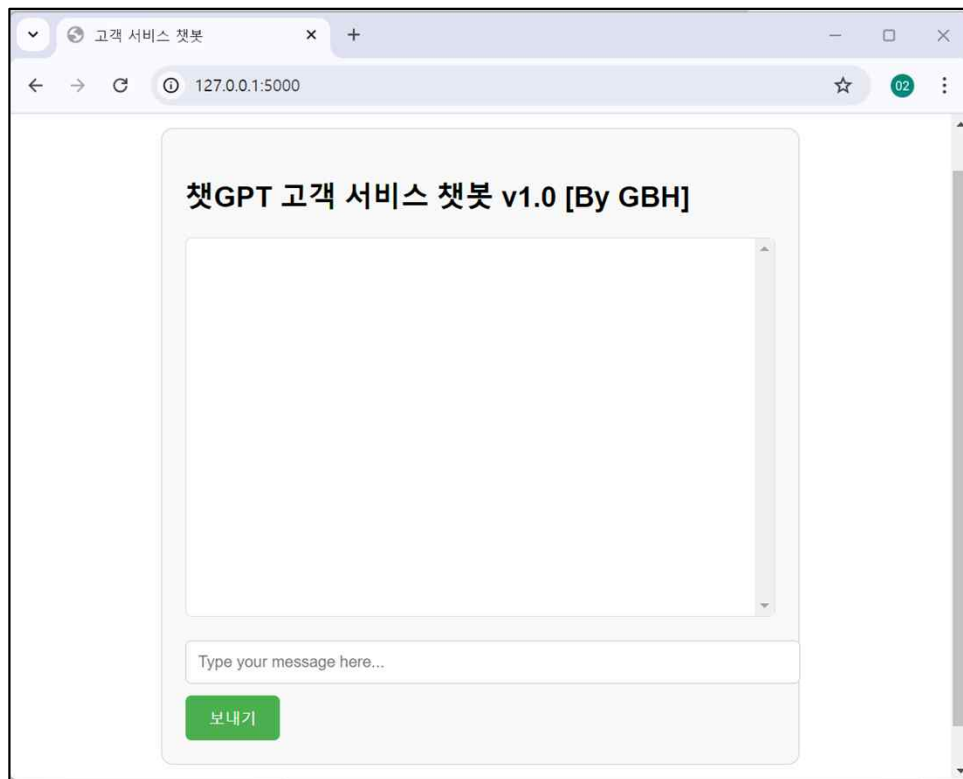
OpenAI API 사용 Flask 웹 서버 실행

서버 소스 코드 실행 후 아래 “http://127.0.0.1:5000” 를 클릭한다

```
if __name__ == "__main__":  
    app.run(debug=False)  
  
* Serving Flask app '__main__'  
* Debug mode: off  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
127.0.0.1 - - [23/Aug/2024 16:29:08] "GET / HTTP/1.1" 200 -
```

2. Flask와 OpenAI 연동하기

OpenAI API 사용 Flask 웹 페이지 화면



2. Flask와 OpenAI 연동하기

질문한 내용을 Context로 계속 유지하면서 질문하기

```
# 대화 히스토리를 저장할 변수
conversation_history = []

@app.route("/get_response", methods=["POST"])
def get_response():
    user_message = request.json.get("message")
    conversation_history.append({"role": "user", "content": user_message})
    print(conversation_history)
    # OpenAI 모델에 요청
    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=conversation_history,
        # max_tokens=300,
        n=1,
        stop=None,
        temperature=0.2
    )
```

2. Flask와 OpenAI 연동하기

질문한 내용을 Context로 계속 유지하면서 질문하기

```
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [23/Aug/2024 16:37:23] "GET / HTTP/1.1" 200 -
[{'role': 'user', 'content': '하늘은 왜 파란가?'}]
127.0.0.1 - - [23/Aug/2024 16:37:35] "POST /get_response HTTP/1.1" 200 -
[{'role': 'user', 'content': '하늘은 왜 파란가?'}, {'role': 'user', 'content': '다시 쉽고 간단하게 설명해줘.'}]
127.0.0.1 - - [23/Aug/2024 16:38:03] "POST /get_response HTTP/1.1" 200 -
[{'role': 'user', 'content': '하늘은 왜 파란가?'}, {'role': 'user', 'content': '다시 쉽고 간단하게 설명해줘.'}, {'role': 'user', 'content': '더 간단하게?'}]
127.0.0.1 - - [23/Aug/2024 16:43:02] "POST /get_response HTTP/1.1" 200 -
[{'role': 'user', 'content': '하늘은 왜 파란가?'}, {'role': 'user', 'content': '다시 쉽고 간단하게 설명해줘.'}, {'role': 'user', 'content': '더 간단하게?'}, {'role': 'user', 'content': '한단어로'}]
127.0.0.1 - - [23/Aug/2024 16:43:39] "POST /get_response HTTP/1.1" 200 -
```

2. Flask와 OpenAI 연동하기

OpenAI API 사용 Flask 웹 페이지 화면

: 질문한 내용을 Context로 계속 유지하면서 질문하기

챗GPT 고객 서비스 챗봇 v1.0 [By GBH]

You: 하늘은 왜 파란가?

Bot: 하늘이 파란 이유는 대기 중의 산란 현상 때문입니다. 태양빛은 여러 색의 빛으로 구성되어 있으며, 이 중 파란색 빛은 다른 색에 비해 파장이 짧습니다. 대기 중의 분자와 미세한 입자들은 짧은 파장의 빛을 더 많이 산란시키는데, 이 현상을 '레이리 산란'이라고 합니다. 태양이 하늘 높이 있을 때, 태양빛이 대기를 통과하면서 파란색 빛이 다른 색보다 더 많이 산란되어 하늘이 파랗게 보이게 됩니다. 반면, 해가 지거나 떠 있을 때, 태양빛이 대기를 통과하는 거리가 길어지면, 파란색 빛은 많이 산란되고, 주황색이나 빨간색 빛이 더 많이 남아 하늘이 붉게 보이기도 합니다. 이러한 이유로 우리는 낮에는 파란 하늘을, 일몰이나 일출 때는 붉은 하늘을 보게 됩니다.

You: 다시 쉽고 간단하게 설명해줘.

Bot: 하늘이 파란 이유는 태양빛이 대기를 통과할 때, 빛의 색깔 중 파란색이 다른 색보다 더 많이 산란되기 때문이에요. 태양빛은 여러 색의 빛으로 구성되어 있으며, 이 중 파란색 빛은 다른 색에 비해 파장이 짧습니다. 대기 중의 분자와 미세한 입자들은 짧은 파장의 빛을 더 많이 산란시키는데, 이 현상을 '레이리 산란'이라고 합니다. 태양이 하늘 높이 있을 때, 태양빛이 대기를 통과하면서 파란색 빛이 다른 색보다 더 많이 산란되어 하늘이 파랗게 보이게 됩니다. 반면, 해가 지거나 떠 있을 때, 태양빛이 대기를 통과하는 거리가 길어지면, 파란색 빛은 많이 산란되고, 주황색이나 빨간색 빛이 더 많이 남아 하늘이 붉게 보이기도 합니다. 이러한 이유로 우리는 낮에는 파란 하늘을, 일몰이나 일출 때는 붉은 하늘을 보게 됩니다.

보내기

2. Flask와 OpenAI 연동하기

질문한 내용을 Context로 계속 유지하면서 질문하기

You: 하늘은 왜 파란가?

Bot: 하늘이 파란 이유는 대기 중의 산란 현상 때문입니다. 태양빛은 여러 색의 빛으로 구성되어 있으며, 이 중 파란색 빛은 다른 색에 비해 파장이 짧습니다. 대기 중의 분자와 미세한 입자들은 짧은 파장의 빛을 더 많이 산란시키는데, 이 현상을 '레이리 산란'이라고 합니다. 태양이 하늘 높이 있을 때, 태양빛이 대기를 통과하면서 파란색 빛이 다른 색보다 더 많이 산란되어 하늘이 파랗게 보이게 됩니다. 반면, 해가 지거나 떠 있을 때, 태양빛이 대기를 통과하는 거리가 길어지면, 파란색 빛은 많이 산란되고, 주황색이나 빨간색 빛이 더 많이 남아 하늘이 붉게 보이기도 합니다. 이러한 이유로 우리는 낮에는 파란 하늘을, 일몰이나 일출 때는 붉은 하늘을 보게 됩니다.

You: 다시 쉽고 간단하게 설명해줘.

Bot: 하늘이 파란 이유는 태양빛이 대기를 통과할 때, 빛의 색깔 중 파란색이 다른 색보다 더 많이 산란되기 때문이에요. 태양빛은 여러 색깔의 빛으로 이루어져 있는데, 파란색 빛은 짧은 파장을 가지고 있어서 공기 중의 작은 입자들과 더 많이 부딪혀서 퍼지게 됩니다. 그래서 우리는 하늘을 파랗게 보게 되는 거예요.

You: 더 간단하게?

Bot: 하늘이 파란 이유는 태양빛이 대기를 통과할 때, 파란색 빛이 다른 색보다 더 많이 퍼지기 때문이에요. 그래서 우리는 하늘을 파랗게 보게 되는 거죠.

You: 한단어로

Bot: 산란.

감사합니다