API

API란?

- **API(Application Programming Interface)**: 프로그램 간의 데이터 교환을 위한 인터페이스.
- API를 사용하는 이유:
  - 직접 데이터를 처리하지 않고, **회사 서버에서 코드를 실행하여 결과만 받을 수 있음**.
  - 데이터 및 기능을 쉽게 활용할 수 있도록 제공됨.

API 예시

- **카카오 지도 API**: 특정 위치의 지도 데이터를 제공.
- **구글 캘린더 API**: 캘린더 이벤트를 추가, 수정, 삭제할 수 있음.
- **OpenAI API**: AI 모델을 활용한 텍스트 생성 및 분석 기능 제공.
- **LoremPicsum API**: 랜덤 이미지를 제공하는 API.

URL

- **URL(Uniform Resource Locator)**: 인터넷에서 리소스를 찾기 위한 주소.
- URL 문법 및 쿼리 파라미터:
  - URL 구조: `https://example.com/path?param1=value1&param2=value2`
  - `?` 뒤의 **쿼리 파라미터(Query Parameter)** 를 사용하여 API 요청을 커스터마이징 가능.
- 예시: 구글 검색에서 `q` 파라미터 사용

```
https://www.google.com/search?q=streamlit
```

  - `q=streamlit`을 변경하면 검색어가 변경됨.

Request

- **Request(요청)**: API에 특정 데이터를 요청하는 과정.
- **Request Code Status(응답 코드 상태)**:
  - `200 OK`: 요청이 성공적으로 수행됨.
  - `400 Bad Request`: 잘못된 요청.
  - `401 Unauthorized`: 인증되지 않은 요청.
  - `403 Forbidden`: 권한이 없는 요청.
  - `404 Not Found`: 요청한 리소스를 찾을 수 없음.
  - `500 Internal Server Error`: 서버 내부 오류.

Lorem Picsum API 사용법

> Lorem Picsum은 이미지를 가져올 수 있는 API이다. 아래의 Lorem Picsum API 사용법을 익혀보자.

1. 기본 이미지 요청

- 특정 크기의 랜덤 이미지 가져오기:

```
https://picsum.photos/200/300
```

위 URL은 가로 200픽셀, 세로 300픽셀 크기의 랜덤 이미지를 반환합니다.

- 정사각형 이미지 가져오기:

```
https://picsum.photos/200
```

위 URL은 200픽셀 x 200픽셀 크기의 정사각형 이미지를 반환합니다.

2. 특정 이미지 요청

- 특정 이미지 ID로 요청:

```
https://picsum.photos/id/237/200/300
```

위 URL은 ID가 237인 이미지를 가로 200픽셀, 세로 300픽셀 크기로 반환합니다.

- 모든 이미지의 ID 목록 조회:

```
https://picsum.photos/v2/list
```

3. 고정된 랜덤 이미지 요청

- 동일한 랜덤 이미지를 항상 가져오기 (seed 값 사용):

```
https://picsum.photos/seed/picsum/200/300
```

4. 이미지 효과 적용

- **그레이스케일 이미지**:

```
https://picsum.photos/200/300?grayscale
```

- **블러 효과 적용**:

```
https://picsum.photos/200/300?blur=2
```

> **퀴즈** : 그레이스케일과 블러를 동시에 적용하려면 어떻게 해야할까?

Lorem Picsum API 사용법

```
import streamlit as st

import requests
from PIL import Image
from io import BytesIO

st.title("Lorem Picsum Test")

# URL session state
if 'url' not in st.session_state:
    st.session_state['url'] = 'https://picsum.photos/1280/720'

# Fetching and displaying the image
if st.button("Fetch Image"):
    response = requests.get(st.session_state['url'])
    with st.expander("Result"):
      st.write("Response status code: ", response.status_code)
      st.write("Response content: ", response.content)
    if response.status_code == 200:
        image = Image.open(BytesIO(response.content))
        st.image(image)
    else:
        st.error("Failed to fetch image. Please check the settings and try
again.")
```

Response Code Status(응답 코드 상태):

- `response.status_code`: 응답의 상태 코드를 확인하는 속성.
- 예제:

```
if response.status_code == 200:
    print("요청이 성공적으로 수행되었습니다.")
```

Response Content:

- `response.content`: 응답 데이터를 바이트(byte) 형식으로 가져옴.
- 예제:

```
image_data = response.content
```

BytesIO 사용:

- `BytesIO`는 바이트 데이터를 파일처럼 다룰 수 있도록 도와주는 객체.
- 이미지 데이터를 `BytesIO`를 활용하여 처리 가능.
- 예제:

```
from io import BytesIO
from PIL import Image

image = Image.open(BytesIO(response.content))
image.show()
```