

# PJT 01

# INDEX

- Python을 활용한 데이터 수집 1
  - 개요
  - 준비사항
  - 요구사항
  - 도전과제

# INDEX

- Python을 활용한 데이터 수집 2
  - 개요
  - 사전 준비
  - 사전 학습 - API
  - API 활용 - dog API
  - API 활용 - Spotify API
  - 요구사항

# Python을 활용한 데이터 수집 1

# 개요

## 프로젝트 개요

- 커뮤니티 서비스 개발을 위한 데이터 구성 단계로,  
필요한 데이터를 직접 추출하고 재구성하는 과정

## 프로젝트 목표

- Python 기본 문법 습득
- 파일 입출력에 대한 이해
- 데이터 구조에 대한 분석과 이해
- 데이터를 가공하고 JSON 형태로 구성하기

# 준비사항



## | 개발도구 및 라이브러리

- 개발도구
  - Visual Studio Code
  - Python 3.9+
- 필수 라이브러리
  - <https://docs.python.org/3.9/library/json.html>

## 제공사항

- examples 폴더
  - 이번 프로젝트 해결을 위해 알아야 하는 혹은 직접적인 도움이 될 수 있는 코드
- data 폴더
  - 실제 문제 풀이에 사용되는 데이터

# 요구사항

## | 필수 요구사항

- A. 제공되는 아티스트 데이터의 주요내용 수집
- B. 제공되는 아티스트 데이터의 주요내용 수정
- C. 다중 데이터 분석 및 수정
- D. 가장 인기도가 높은 아티스트 데이터 수집
- E. 특정 팔로워 수 이상의 아티스트 데이터 수집

## A. 제공되는 아티스트 데이터의 주요내용 수집 - 요구사항

- `problem_a.py` 풀이
- 필요한 정보
  - `id`, `name`, `genres_ids`, `images`, `type`
- `artist.json`에서 필요한 정보에 해당하는 값을 추출
- 필요한 정보를 새로운 dictionary로 반환하는 함수 `artist_info`를 작성
  - 완성된 함수는 다음 문제의 기본 기능으로 사용됨

## A. 제공되는 아티스트 데이터의 주요내용 수집 - 결과

- problem\_a.py 실행 예시

```
{'genres_ids': [651, 816],  
  'id': 178,  
  'images': [{'height': 640,  
              'url': 'https://i.scdn.co/image/ab6761610000e5eb59f8cfc8e71dcaf8c6ec4bde',  
              'width': 640},  
             {'height': 320,  
              'url': 'https://i.scdn.co/image/ab6761610000517459f8cfc8e71dcaf8c6ec4bde',  
              'width': 320},  
             {'height': 160,  
              'url': 'https://i.scdn.co/image/ab6761610000f17859f8cfc8e71dcaf8c6ec4bde',  
              'width': 160}],  
  'name': 'Jimin',  
  'type': 'artist'}
```

❖ 주의) [pprint](#) 함수로 인해 dictionary의 key 순서가 정렬되어서 출력

## B. 제공되는 아티스트 데이터의 주요내용 수정 - 요구사항

- `problem_b.py` 풀이
- 필요한 정보
  - `id`, `name`, `images`, `type`, `genres_names`
- `artist.json`에서 필요한 정보에 해당하는 값을 추출
  - 이때, `genres_names` 대신 `genres_ids`를 추출
- `genres.json`을 이용하여 이전 단계에서 만들었던 데이터 중 `genres_ids`를 장르 번호가 아닌 장르 이름 리스트 `genres_names`로 바꿔 반환하는 함수를 `artist_info`를 작성
  - 완성된 함수는 다음 문제의 기본 기능으로 사용됨

## B. 제공되는 아티스트 데이터의 주요내용 수정 - 결과

- problem\_b.py 실행 예시

```
{'genres_names': ['punk-rock', 'anime'],  
 'id': 178,  
 'images': [{'height': 640,  
             'url': 'https://i.scdn.co/image/ab6761610000e5eb59f8cfc8e71dcaf8c6ec4bde',  
             'width': 640},  
            {'height': 320,  
             'url': 'https://i.scdn.co/image/ab6761610000517459f8cfc8e71dcaf8c6ec4bde',  
             'width': 320},  
            {'height': 160,  
             'url': 'https://i.scdn.co/image/ab6761610000f17859f8cfc8e71dcaf8c6ec4bde',  
             'width': 160}],  
 'name': 'Jimin',  
 'type': 'artist'}
```

❖ 주의) [pprint](#) 함수로 인해 dictionary의 key 순서가 정렬되어서 출력



## C. 다중 데이터 분석 및 수정 - 요구사항

- `problem_c.py` 풀이
- 필요한 정보
  - `id`, `name`, `images`, `type`, `genres_names`
- `artists.json`, `genres.json` 활용
  - `artists.json`은 전체 아티스트 데이터를 제공
- 이전 단계의 함수 구조를 재사용
  - `genres_names` 대신 `genres_ids`를 추출
- 위 요구사항을 반영한 새로운 `list`를 반환하는 함수 `artist_info`를 작성

## C. 다중 데이터 분석 및 수정 - 결과

- problem\_c.py 실행 예시

```
[{'genres_names': ['acoustic', 'electro', 'j-rock'],  
  'id': 451,  
  'images': [{'height': 640,  
              'url': 'https://i.scdn.co/image/ab6761610000e5ebd642648235ebf3460d2d1f6a',  
              'width': 640},  
             {'height': 320,  
              'url': 'https://i.scdn.co/image/ab67616100005174d642648235ebf3460d2d1f6a',  
              'width': 320},  
             {'height': 160,  
              'url': 'https://i.scdn.co/image/ab6761610000f178d642648235ebf3460d2d1f6a',  
              'width': 160}],  
  'name': 'BTS',  
  'type': 'artist'},  
 {'genres_names': ['edm'],  
  'id': 116,  
  'images': [{'height': 640,  
              'url': 'https://i.scdn.co/image/ab6761610000e5eb6199c3c2f414880e2b9077a9',  
              'width': 640},  
             {'height': 320,  
              'url': 'https://i.scdn.co/image/ab676161000051746199c3c2f414880e2b9077a9',  
              'width': 320},  
             {'height': 160,  
              'url': 'https://i.scdn.co/image/ab6761610000f1786199c3c2f414880e2b9077a9',  
              'width': 160}],  
  'name': 'NewJeans',  
  'type': 'artist'},  
  # 이하 생략  
]
```

## D. 가장 인기도가 높은 아티스트 데이터 수집 - 요구사항

- `problem_d.py` 풀이
- `artists.json`, `artists` 폴더 파일 활용
- 아티스트 세부 정보 중 인기도(`popularity`) 활용
- 반복문을 통해 `artists` 폴더 내부의 파일들을 오픈
  - 제공된 `03_example.py` 코드 참고
- 인기도가 같은 아티스트 없음
- 가장 인기도가 높은 아티스트의 이름을 출력하는 함수 `max_popularity`를 작성

## D. 가장 인기도가 높은 아티스트 데이터 수집 - 결과

- problem\_d.py 실행 예시

The image shows the logo for the K-pop group NewJeans. The text "NewJeans" is displayed in a white, sans-serif font with a slight shadow effect, set against a solid black rectangular background.

## E. 특정 팔로워 수 이상의 아티스트 데이터 수집 - 요구사항

- `problem_e.py` 풀이
- `artists.json`, `artists` 폴더 파일 활용
- 아티스트 세부 정보 중 팔로워 수(followers) 활용
- 반복문을 통해 `artists` 폴더 내부의 파일들을 오픈
  - 제공된 `03_example.py` 코드 참고
- 팔로워 수의 총 합이 10,000,000이상인 아티스트들의 `name`과 `uri-id`를 출력하는 함수 `dec_artists`를 작성
  - `uri-id`는 아티스트 세부정보에서 `uri` 값 중 두번째 콜론(:) 뒤의 값  
예시: `spotify:artist:6HvZYSbFfjnjFrWF950C9d`

## E. 특정 팔로워 수 이상의 아티스트 데이터 수집 - 결과

- problem\_e.py 실행 예시

```
[{'name': 'BTS', 'uri-id': '3Nrfpe0tUJi4K4DXYWgMUX'},  
 {'name': 'BLACKPINK', 'uri-id': '41MozSoPIsD1dJM0CLPjZF'},  
 {'name': 'Stray Kids', 'uri-id': '2dIgFja1Vxs4ThymZ67YCE'},  
 {'name': 'j-hope', 'uri-id': '0b1sIQumIAsNbqAoIClSpy'}]
```

# 도전과제

## 도전 과제 안내

- 생성형 AI 도구를 활용하여 도전과제 요구사항 해결하기
- 생성형 AI를 통해 코드 생성, 아이디어 구상, 문제 해결 방법 탐색 등 다양한 방식으로 활용 가능
- 선택한 생성형 AI 서비스는 자유롭게 결정
- 최종 결과물은 AI 생성 내용을 바탕으로 직접 수정 및 개선하여 적용하기



## | AI 활용 주의사항

- AI 도구는 보조 수단으로 활용하되, 능동적인 자세로 학습에 임할 것
- 최종적인 이해와 적용은 자기 주도적 학습을 통해 이루어지며,  
배운 내용을 스스로 기록하고 정리하며 학습 효과를 높일 것

## F. 도전 과제 - 요구사항

- problem\_f\_1.py
  - 팔로워가 5,000,000이상, 10,000,000미만인 아티스트들의 이름과 팔로워 수 수집
- problem\_f\_2.py
  - 장르에 acoustic이 포함된 아티스트 이름 수집

## F. 도전 과제 - 결과

- problem\_f\_1.py 실행 예시

```
[{'followers': 5901644, 'name': 'Jimin'},  
 {'followers': 9571638, 'name': 'SEVENTEEN'},  
 {'followers': 8041766, 'name': 'IU'},  
 {'followers': 8000368, 'name': 'Jung Kook'},  
 {'followers': 6278033, 'name': '(G)I-DLE'},  
 {'followers': 6785638, 'name': 'NCT DREAM'}]
```

- problem\_f\_2.py 실행 예시

```
[ 'BTS' ]
```

# Python을 활용한 데이터 수집 2

# 개요

## 프로젝트 개요

- 커뮤니티 서비스 개발을 위한 데이터 구성 단계로,  
필요한 데이터를 직접 추출하고 재구성하는 과정
- API를 활용한 요청과 이에 따른 응답 데이터를 활용

## 프로젝트 목표

- API에 대한 이해
- 서버와의 요청과 응답에 대한 이해
- 데이터 타입 List, Dictionary 조작하기
- JSON 데이터를 재구성하기

# 사전 준비



## 개발도구 및 라이브러리

- 개발도구
  - Visual Studio Code
  - Python 3.9+
- 필수 라이브러리
  - <https://requests.readthedocs.io/en/latest/>
- API
  - [Spotify API](#) - 음악 정보 및 API 서비스

## 제공사항

- `examples` 폴더 내 예시 파일
  - 이번 프로젝트 해결을 위해 알아야 하는 혹은 직접적인 도움이 될 수 있는 코드
- `config` 폴더 내 `spotify_config.py`
  - Spotify 서버에 요청을 보내기 위해 작성된 코드

## | 사전 준비 (1/2)

- 공용 문서 (<https://abit.ly/pb-document>)
  - “Spotify API Key 발급 가이드” 진행

## 사전 준비 (2/2)

- 발급 받은 Client ID와 Client Secret ID를 각각 복사해서 config 폴더 내 `spotify_config.py`에 입력

```
'''
Spotify에 요청을 보내기 위한 Header파일
'''

import requests

# 여기에서 발급받은 API_CLIENT_ID와 API_CLIENT_SECRET
API_URL = 'https://api.spotify.com/v1'
API_CLIENT_ID = ''
API_CLIENT_SECRET = ''
```

spotify\_config.py

Dashboard > My Test App > Settings > Basic Information

### M Basic Information

Basic Information User Management Extension Requests

Client ID

① API\_CLIENT\_ID

[View client secret](#) ② API\_CLIENT\_SECRET

❖ 발급받은 API Key는 외부에 유출하지 않도록 주의

# 사전 학습 - API

# API

## (Application Programming Interface)

두 소프트웨어(또는 시스템)가 서로 통신할 수 있게 하는 메커니즘

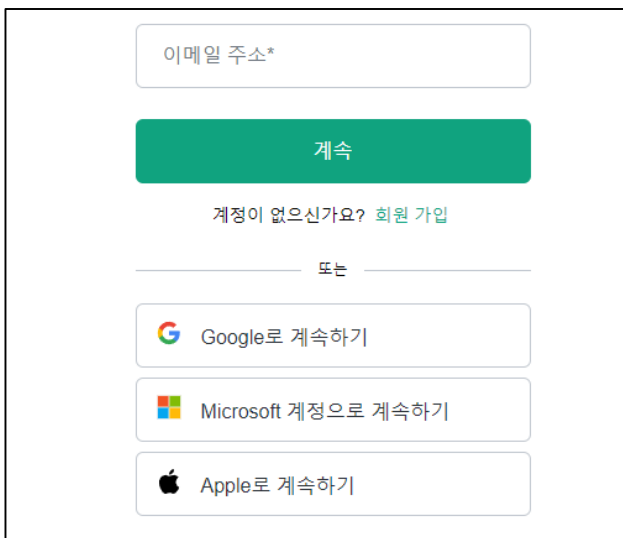
- ‘약속된 방식의 인터페이스’로, 특정 규칙에 따라 데이터를 요청하고 응답하는 규칙을 제공

# Application

특정 기능을 수행하는 모든 소프트웨어

- 웹·모바일·데스크톱 앱 등, 우리가 만든 서비스나 프로그램도 모두 앱의 일종

## API 예시 1 - 소셜 로그인 (1/4)



이메일 주소\*

계속

계정이 없으신가요? 회원 가입

또는

Google로 계속하기

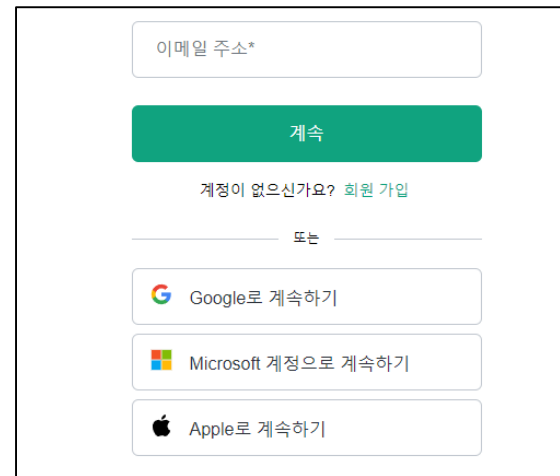
Microsoft 계정으로 계속하기

Apple로 계속하기

- 최근에는 특정 사이트에 직접 회원가입을 하지 않고 다른 소셜 미디어 계정으로 회원가입 및 로그인 하는 경우가 많음
- 어떻게 가능할까?



## API 예시 1 - 소셜 로그인 (2/4)

A screenshot of the ChatGPT login interface. It features a text input field for an email address, a green '계속' (Continue) button, a link for '계정이 없으신가요? 회원 가입' (Don't have an account? Sign up), a separator line with '또는' (or), and three social login buttons: 'Google로 계속하기' (Continue with Google), 'Microsoft 계정으로 계속하기' (Continue with Microsoft account), and 'Apple로 계속하기' (Continue with Apple).

- ChatGPT에서 Google로 회원가입 및 로그인을 진행하는 과정 알아보기
- 사용자는 ChatGPT 로그인 화면에서 “Google로 계속하기”를 클릭

## API 예시 1 - 소셜 로그인 (3/4)



- ChatGPT가 제공한 Google 로그인 화면에서 Google 계정으로 로그인 진행

## API 예시 1 - 소셜 로그인 (4/4)



ChatGPT

1. 어떤 사람이 Google로 내 사이트에 회원가입 하려고 하는데 어떤 사용자인지 정보를 줘



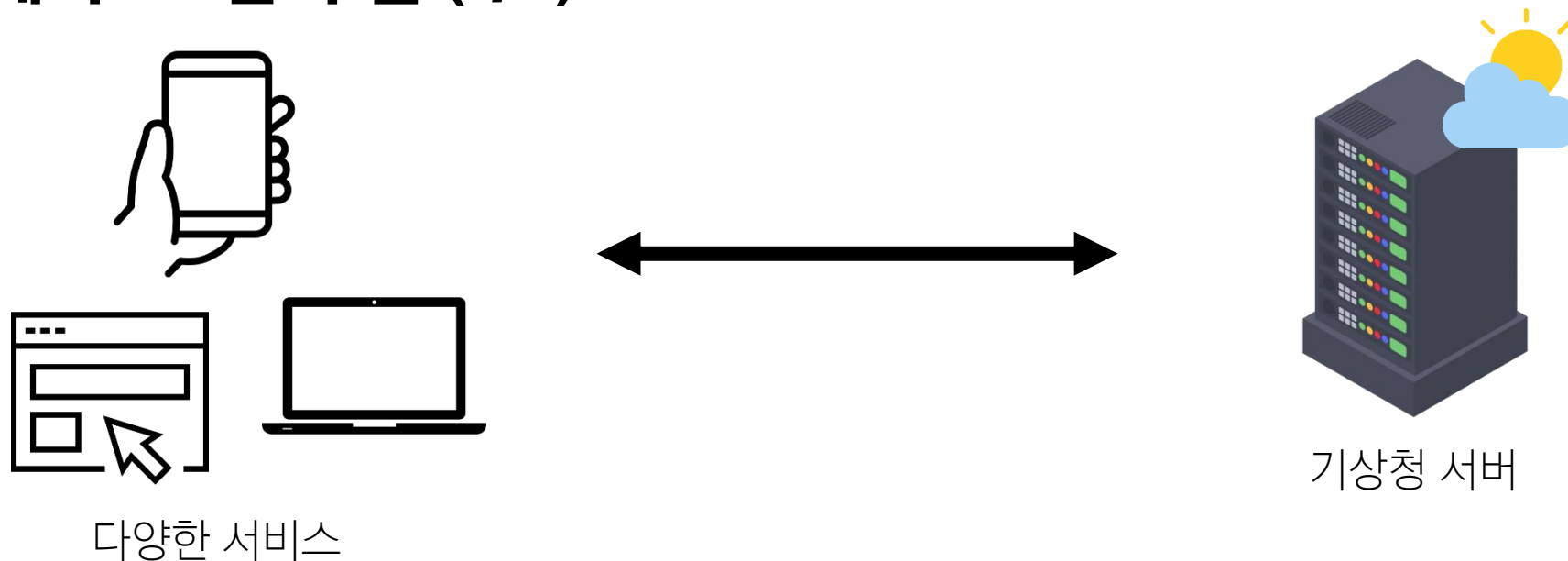
2. 알겠어. 이 사용자는 내 사이트에 올바르게 로그인 했고 확인해보니 ~~ 한 정보를 가진 사용자야



Google API

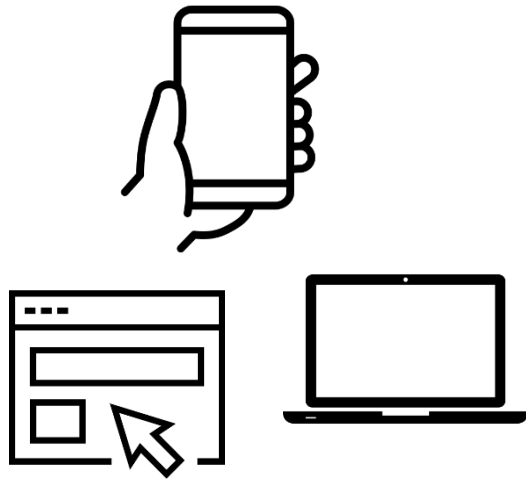
- Google 로그인 계정으로 로그인을 성공했을 경우 Google API는 ChatGPT에게 로그인에 성공한 인증된 사용자 정보를 넘겨줌
- 사용자 정보를 넘겨받은 ChatGPT는 해당 정보를 활용해 회원가입 및 로그인을 진행

## API 예시 2 - 날씨 앱 (1/3)

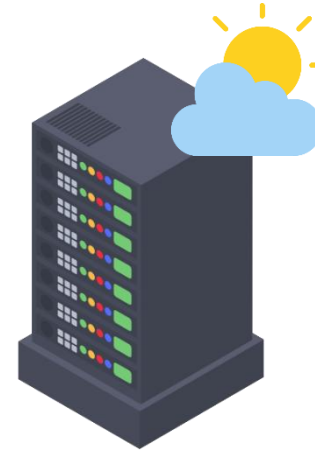


- 기상 데이터가 들어있는 기상청의 서버
- 스마트폰의 날씨 앱, 웹 사이트의 날씨 정보 등 다양한 서비스들이 이 기상청 서버로부터 데이터를 요청해서 응답을 받음

## API 예시 2 - 날씨 앱 (2/3)



다양한 서비스



기상청 서버

- 날씨 데이터를 얻으려면?
  - 기상청 서버에는 날씨 정보를 얻고 싶으면 이런 식으로 요청해야 한다는 지정된 형식들이 작성되어 있음
  - 지역, 날짜, 조회할 내용들(온도, 바람 등)을 제공하는 매뉴얼

## API 예시 2 - 날씨 앱 (3/3)



- “이렇게 요청을 보내면, 이렇게 정보를 제공 해줄 것이다”라는 매뉴얼
  - 소프트웨어와 소프트웨어 간 지정된 정의(형식)으로 소통하는 수단 → API
- 스마트폰의 날씨 앱은 기상청에서 제공하는 API를 통해 기상청 시스템과 대화하여 매일 최신 날씨 정보를 표시 할 수 있음

## API Key

API에게 요청을 보내는 애플리케이션을  
구별하기 위한 고유한 식별 문자열

- 예) `abc123xyz456..` 처럼 랜덤하게 생성된 키를 서버가 발급

## API Key 활용



Client

1. 사용자 회원가입



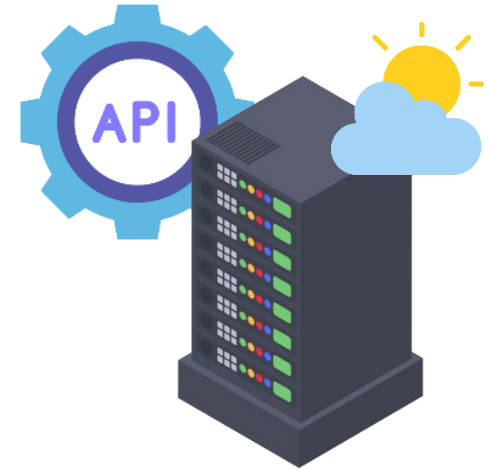
2. 응답 (+ API Key)



3. 데이터 요청(+ API key)



5. 응답(+ 요청 데이터)



Server

4. API key 검증

1. Client가 날씨 API 사이트에서 회원 가입
2. 날씨 사이트서 Client에게 API Key를 발급
3. Client가 요청 시 발급 받은 API Key를 요청마다 함께 보냄
4. 요청 받은 Server는 Key를 검사해 인증 혹은 인증 실패 여부를 결정하여 응답함



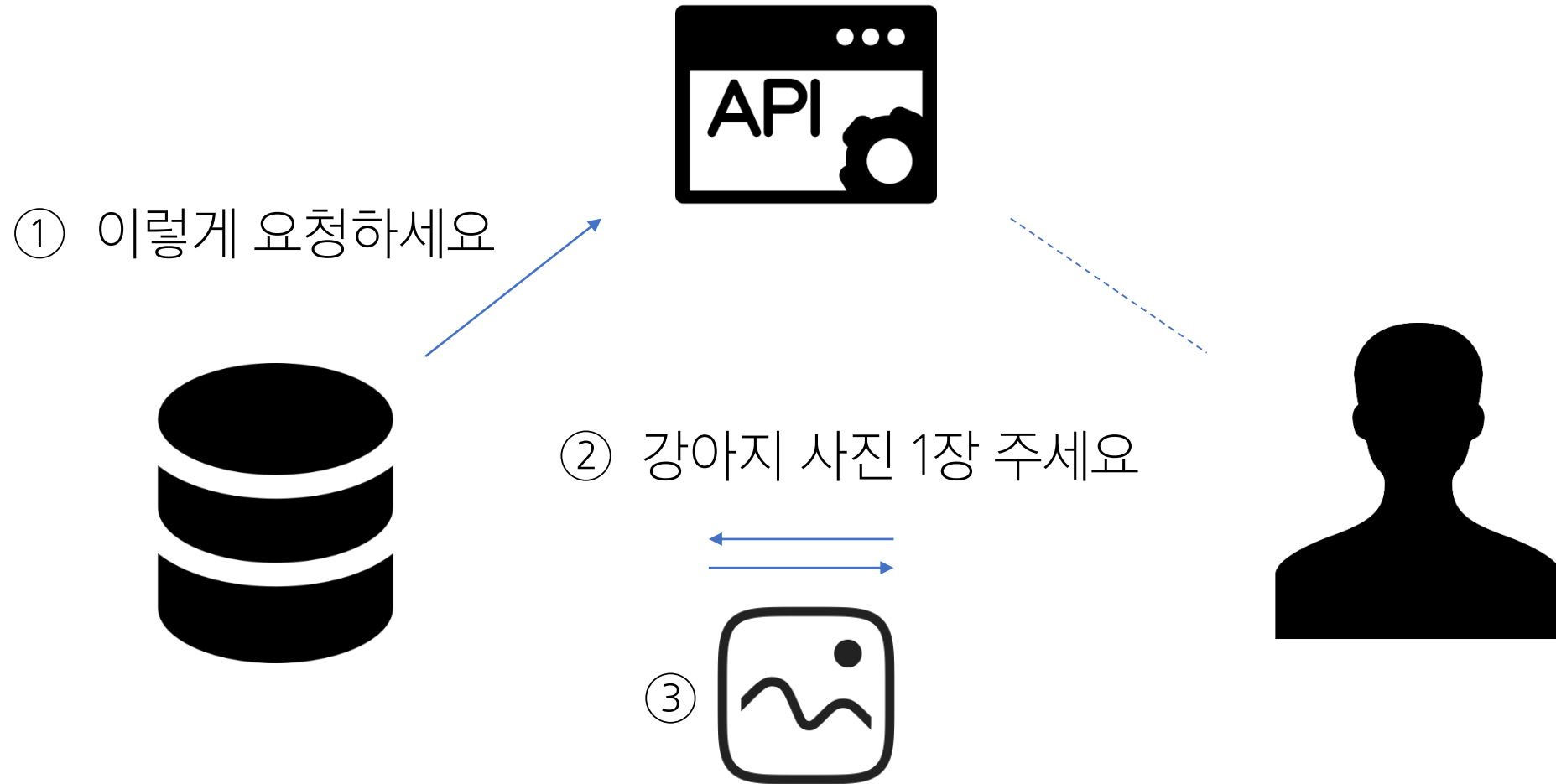
## API Key가 필요한 이유


- 보안 강화
  - 무단 접근을 막고, 승인된 사용자(또는 앱)만 요청할 수 있도록
- 데이터 관리
  - API 호출 횟수, 사용량 모니터링
  - 일정량 이상의 사용 시 제한 또는 과금 정책 적용 가능

## | API Key 사용 시 주의사항

- 공개된 곳에 노출하지 말 것
- 키가 유출될 경우 무단 사용 위험 → 정기 갱신 필요
- 서버-클라이언트 구조에서 키를 안전하게 저장하는 방법들 고려

# API 활용 - dog API





### Dog API

- Documentation
- Breeds list
- About
- Submit your dog

---

- Dog CEO Zine
- Buy me a dog treat
- View on GitHub
- Follow on Twitter

The internet's biggest collection of **open source dog pictures**.

Read our [documentation](#) to find out more or try it out for yourself below.

Want to add your dog to the collection? Submit your photos as a [pull request here](#).


Need more dog in your life? Get issue 1 of [Dog CEO Zine](#) - a quarterly business and lifestyle magazine for the modern dog. Featuring an exclusive interview with Scottie the Monopoly dog. [Order your copy](#) from Side Orders Publishing. Ships worldwide.

Fetch!

JSON

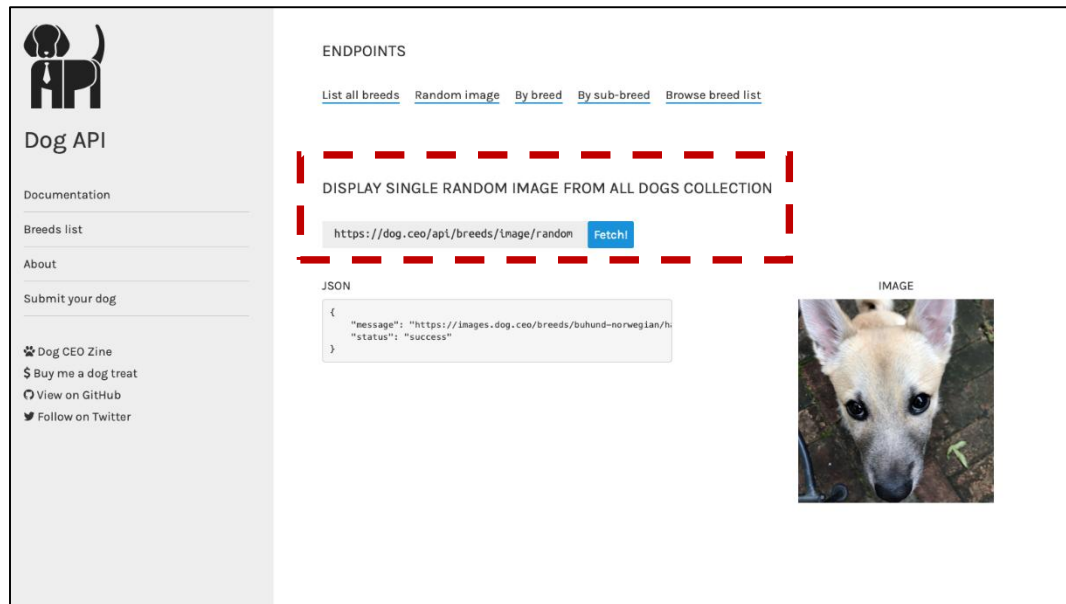
```
{  "message": "https://images.dog.ceo/breeds/boxer/n02108089_35",  "status": "success"}
```

IMAGE



강아지 사진을 요청하면 응답해주는 Dog API  
<https://dog.ceo/dog-api/>

무작위 강아지 사진 한 장 필요할 땐 이렇게 요청

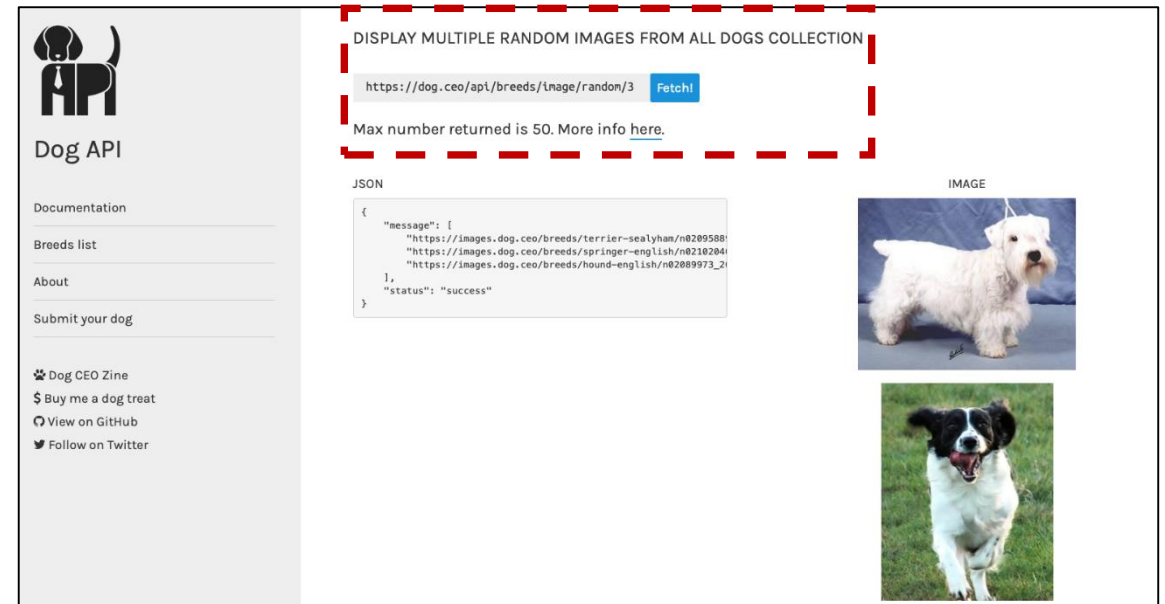


The screenshot shows the Dog API website interface. On the left is a sidebar with navigation links: Documentation, Breeds list, About, and Submit your dog. The main content area is titled 'ENDPOINTS' and lists several options: List all breeds, Random image, By breed, By sub-breed, and Browse breed list. The 'Random image' endpoint is highlighted with a red dashed box. Below this box, the URL `https://dog.ceo/api/breeds/image/random` is shown next to a 'Fetch!' button. To the right of the URL, a JSON response is displayed: 

```
{  "message": "https://images.dog.ceo/breeds/buhund-norwegian/h",  "status": "success"}
```

. Below the JSON, a small image of a dog's face is shown, labeled 'IMAGE'.

무작위 강아지 사진 여러 장 필요할 땐 이렇게 요청



The screenshot shows the Dog API website interface. On the left is a sidebar with navigation links: Documentation, Breeds list, About, and Submit your dog. The main content area is titled 'DISPLAY MULTIPLE RANDOM IMAGES FROM ALL DOGS COLLECTION'. Below this title, the URL `https://dog.ceo/api/breeds/image/random/3` is shown next to a 'Fetch!' button. Below the URL, a note states: 'Max number returned is 50. More info [here](#).' To the right of the URL, a JSON response is displayed: 

```
{  "message": {    "https://images.dog.ceo/breeds/terrier-sealyham/n0209598",    "https://images.dog.ceo/breeds/springer-english/n0210204",    "https://images.dog.ceo/breeds/hound-english/n02089973_2",  },  "status": "success"}
```

. To the right of the JSON, two small images of dogs are shown, labeled 'IMAGE'.

적절한 요청 방법과 응답 결과를 알려주는 가이드 문서

```
# 01_example.py

import requests

URL = 'https://dog.ceo/api/breeds/image/random'

response = requests.get(URL).json()
print(response)

results = response.get('message')
print(results)
```

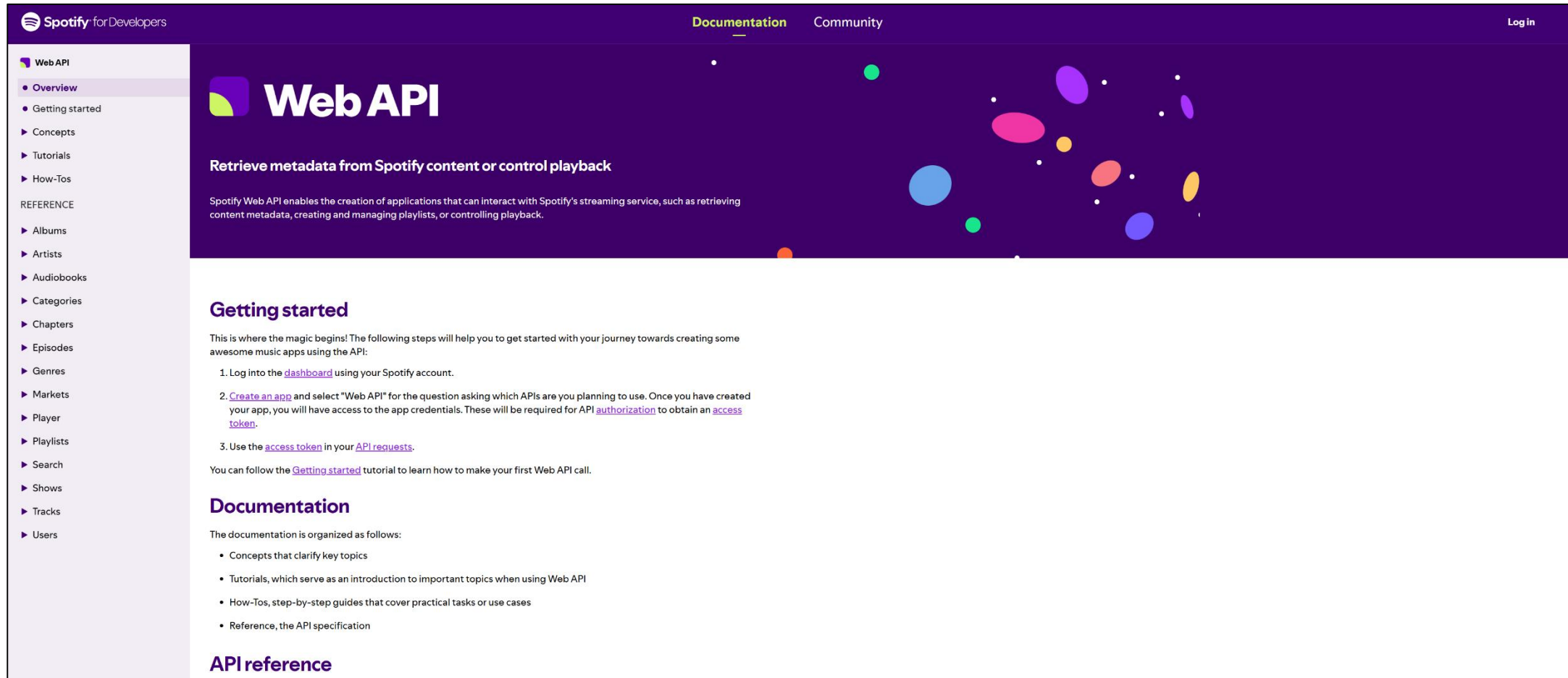
무작위 강아지 사진 한 장 요청

```
{'message': 'https://images.dog.ceo/breeds/spaniel-brittany/n02101388_10373.jpg',  
 'status': 'success'}  
'https://images.dog.ceo/breeds/spaniel-brittany/n02101388_10373.jpg'
```

API는 message에 사진 경로 데이터를  
담아 응답

# API 활용 - Spotify API





The screenshot shows the Spotify for Developers Web API documentation page. The page has a dark purple header with the Spotify logo and 'for Developers' text. Navigation links for 'Documentation' and 'Community' are visible. A sidebar on the left lists various API categories under 'Web API', with 'Overview' selected. The main content area features a large 'Web API' title and a subtitle 'Retrieve metadata from Spotify content or control playback'. Below this, a paragraph explains the API's capabilities. The 'Getting started' section provides a three-step guide: logging into the dashboard, creating an app, and using the access token. The 'Documentation' section lists the types of content available: Concepts, Tutorials, How-Tos, and Reference. The 'API reference' section is also visible at the bottom.

Spotify for Developers

Documentation Community Login

## Web API

Retrieve metadata from Spotify content or control playback

Spotify Web API enables the creation of applications that can interact with Spotify's streaming service, such as retrieving content metadata, creating and managing playlists, or controlling playback.

### Getting started

This is where the magic begins! The following steps will help you to get started with your journey towards creating some awesome music apps using the API:

1. Log into the [dashboard](#) using your Spotify account.
2. [Create an app](#) and select "Web API" for the question asking which APIs are you planning to use. Once you have created your app, you will have access to the app credentials. These will be required for API [authorization](#) to obtain an [access token](#).
3. Use the [access token](#) in your [API requests](#).

You can follow the [Getting started](#) tutorial to learn how to make your first Web API call.

### Documentation

The documentation is organized as follows:

- Concepts that clarify key topics
- Tutorials, which serve as an introduction to important topics when using Web API
- How-Tos, step-by-step guides that cover practical tasks or use cases
- Reference, the API specification

### API reference

Spotify API와 가이드 문서  
<https://developer.spotify.com/documentation/web-api>

## | 요청 URL 예시 (1/2)

- 무슨 자료를 요청할지 (endpoint)
- 검색 기능 요청 예시

`https://api.spotify.com/v1/search  
?q=genre:k-pop&type=artist&market=KR`

## | 요청 URL 예시 (2/2)

- 자료 요청에 필요한 정보 (query)
  - endpoint와 query는 ‘?’ 로 연결 및 구분됨
  - 각각의 query는 ‘&’ 으로 연결

`https://api.spotify.com/v1/search`  
`?q=genre:k-pop&type=artist&market=KR`

## URL 파라미터 (쿼리 파라미터)

1. URL에 포함된 파라미터로 '?' 이후의 문자열
2. 여기서 파라미터는 [파라미터 Key]=[파라미터 Value]가 한 세트이다.
  - 파라미터 Key는 대소문자를 구분한다.
  - API 사용 시, 공식문서에 명시된 파라미터 Key를 사용해야 한다.
3. 파라미터가 여러 개면 '&'로 구분한다.

예시) `https://example.com?fRuiT=orange&gAmE=tic-tac-toe`

```
URL = 'https://example.com'
params = {
    'fRuiT': 'orange',
    'gAmE': 'tic-tac-toe'
}
response = requests.get(URL, params=params).json()
print(response)
```

요청에 대한 응답 예시

```
{'fruit_param': 'I love orange!', 'game': 'tic-tac-toe is my favorite!'}
```

## 요청 코드 예시

```
# 02_example.py
```

```
def get_music():
```

```
    URL = 'https://api.spotify.com/v1'
```

```
    headers = getHeaders()
```

```
    params = {
```

```
        'q': 'artist:BTS', # 필수 파라미터
```

```
        'type': 'track', # 필수 파라미터
```

```
        'market': 'KR',
```

```
        'limit': 1,
```

```
    }
```

```
# 요청을 보내 받아온 결과는 requests 타입의 데이터이고, 파이썬에서 바로 쓸 수 없으며
```

```
response = requests.get(f'{URL}/search', headers=headers, params=params)
```

```
# 파이썬에서 쓸 수 있도록 하기 위해 json() 메서드를 사용해 json 타입의 데이터를 파이썬의 자료형으로 변환한다.
```

```
response = response.json()
```

```
# 응답 데이터 확인 및 필요 정보를 추출한다.
```

```
result = response.get('tracks').get('items')
```

```
return result
```

단, Spotify API의 경우 인증을 위한 토큰을 먼저 받은 후

이를 함께 요청 header에 담아 진행해야 사용 가능

토큰 요청 및 받는 코드는 제공된 `spotify_config.py` 확인

(<https://developer.spotify.com/documentation/web-api/tutorials/getting-started> 참고)

## Spotify API 사용 시 숙지사항

1. Spotify API를 통해 실습하기 위해서는 `config` 폴더 하위에 있는 `spotify_config.py`를 먼저 완성해야만 실습이 가능
2. 또한, API를 요청할 때 모든 Spotify API의 Headers에 `spotify_config.py`의 `getHeaders()`가 요청 headers에 필수로 작성되어야 함

```
def get_music():
    URL = 'https://api.spotify.com/v1'
    headers = getHeaders()
    params = {
        'q': 'artist:BTS', # 필수 파라미터
        'type': 'track', # 필수 파라미터
        'market': 'KR',
        'limit': 1,
    }

    response = requests.get(f'{URL}/search', headers=headers, params=params)
```

## Spotify API 사용 시 참고 링크

- 메인 참고 문서
  - <https://developer.spotify.com/documentation/web-api/reference/search>
- 추가 참고 문서
  - <https://developer.spotify.com/documentation/web-api/reference/get-track>
  - <https://developer.spotify.com/documentation/web-api/reference/get-an-artist>
  - <https://developer.spotify.com/documentation/web-api/reference/get-an-artists-top-tracks>

# 요구사항



## | 공통 요구사항

- API 요청 시 시장(market)은 “대한민국”의 국가코드(‘KR’)를 활용하여 진행

## | 필수 요구사항

- A. 아티스트 정보 조회
- B. 아티스트 정보 조회 & 조건
- C. 트랙 정보 조회 & 조건
- D. 아티스트의 인기 트랙 정보 조회

## A. 아티스트 정보 조회 - 요구사항

- `problem_a.py` 풀이
- k-pop 장르의 아티스트에 대한 카탈로그 정보 조회
- 최대 20명의 아티스트 이름 조회
- 요구사항을 만족하는 함수 `get_artists` 작성

## A. 아티스트 정보 조회 - 결과

- problem\_a.py 실행 예시

```
['Jimin',  
'Jung Kook',  
'Jin',  
'ROSÉ',  
'BIGBANG',  
'NewJeans',  
'aespa',  
'DAY6',  
'G-DRAGON',  
'Park Hyo Shin',  
'BTS',  
'PLAVE',  
'LE SSERAFIM',  
'BOYNEXTDOOR',  
'Stray Kids',  
'ILLIT',  
'IU',  
'SEVENTEEN',  
'BIG Naughty',  
'Jack Harlow']
```

❖ Spotify “Search for Item” 응답 데이터는 시기에 따라 예시 출력과 다를 수 있음

## B. 아티스트 정보 조회 & 조건 - 요구사항

- `problem_b.py` 풀이
- k-pop 장르에서 인기도가 높은 아티스트에 대한 카탈로그 정보 조회
- 최대 20명의 아티스트 이름을 조회한 결과에서 인기도가 **80**이상인 아티스트 조회
- 요구사항을 만족하는 함수 `get_popular_artists` 작성

## B. 아티스트 정보 조회 & 조건 - 결과

- problem\_b.py 실행 예시

```
['Jimin',  
'Jung Kook',  
'Jin',  
'ROSÉ',  
'NewJeans',  
'aespa',  
'Park Hyo Shin',  
'BTS',  
'LE SSERAFIM',  
'Stray Kids',  
'SEVENTEEN',  
'Jack Harlow']
```

❖ Spotify “Search for Item” 응답 데이터는 시기에 따라 예시 출력과 다를 수 있음

## C. 트랙 정보 조회 & 조건 - 요구사항

- `problem_c.py` 풀이
- 클래식 장르에서 가장 최근에 발매된 트랙에 대한 카탈로그 정보 조회
- 최대 10개의 트랙을 조회한 결과에서 발매일이 가장 빠른 순서대로 5개 트랙을 조회
  - 각 트랙의 앨범 이름, 아티스트 이름 그리고 발매일을 함께 출력
- 요구사항을 만족하는 함수 `get_recently_tracks` 작성
- 참고)
  - [sort 메서드](#)
  - [sorted 함수](#)

## C. 트랙 정보 조회 & 조건 - 결과

- problem\_c.py 실행 예시

```
[{'발매일': '2024-09-27',  
  '아티스트': 'Erik Satie',  
  '앨범명': 'Satie: 6 Gnossiennes: No. 2, Avec étonnement'},  
{ '발매일': '2021-08-27',  
  '아티스트': 'Frédéric Chopin',  
  '앨범명': 'Chopin: Piano Concerto No. 2; Scherzi'},  
{ '발매일': '2017-11-17', '아티스트': 'Claude Debussy', '앨범명': 'Debussy'},  
{ '발매일': '2016-04-01',  
  '아티스트': 'Various Artists',  
  '앨범명': 'Erik Satie & Friends'},  
{ '발매일': '2015-09-18', '아티스트': 'Yo-Yo Ma', '앨범명': 'Songs from the Arc of Life'}]
```

❖ Spotify “Search for Item” 응답 데이터는 시기에 따라 예시 출력과 다를 수 있음



## D. 아티스트의 인기 트랙 정보 조회 - 요구사항

- `problem_d.py` 풀이
- 특정 아티스트의 인기 트랙 정보 조회
- 제공된 아티스트 `BTS`와 `OldShirts` 활용
- 유사한 아티스트가 없을 경우 `None`을 반환
- 요구사항을 만족하는 함수 `get_related_artists` 작성

## D. 아티스트의 인기 트랙 정보 조회 - 결과

- problem\_d.py 실행 예시

```
['나의 낙하,(My falling,)',  
'돛단배,(a sailboat,)',  
'나쁜세상,(a bad world)',  
'Collapse Again',  
'숲속,(in the forest)',  
'엔딩,(The ending)',  
'이별,(Parting,)',  
'색종이,(Colored paper,)',  
'내마음,(My heart,)',  
'무아,無我(selfless)']  
None
```

❖ Spotify “Get Artist’s Top Tracks” 응답 데이터는 시기에 따라 예시 출력과 다를 수 있음

# 제출

## 제출 시 주의사항

- 제출기한은 금일 18시까지 입니다. 제출기한을 지켜 주시기 바랍니다.
- 반드시 **README.md** 파일에 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점을 등을 상세히 기록하여 제출합니다.
  - 단순히 완성된 코드만을 나열하지 않습니다.
- 위에 명시된 요구사항은 최소 조건이며, 추가 개발을 자유롭게 진행할 수 있습니다.
- <https://lab.ssafy.com/>에 프로젝트를 생성하고 제출합니다.
  - 프로젝트 이름은 '프로젝트 번호 + pjt'로 지정합니다. (ex. **01-pjt**)
- 반드시 각 반 담당 강사님을 Maintainer로 설정해야 합니다.