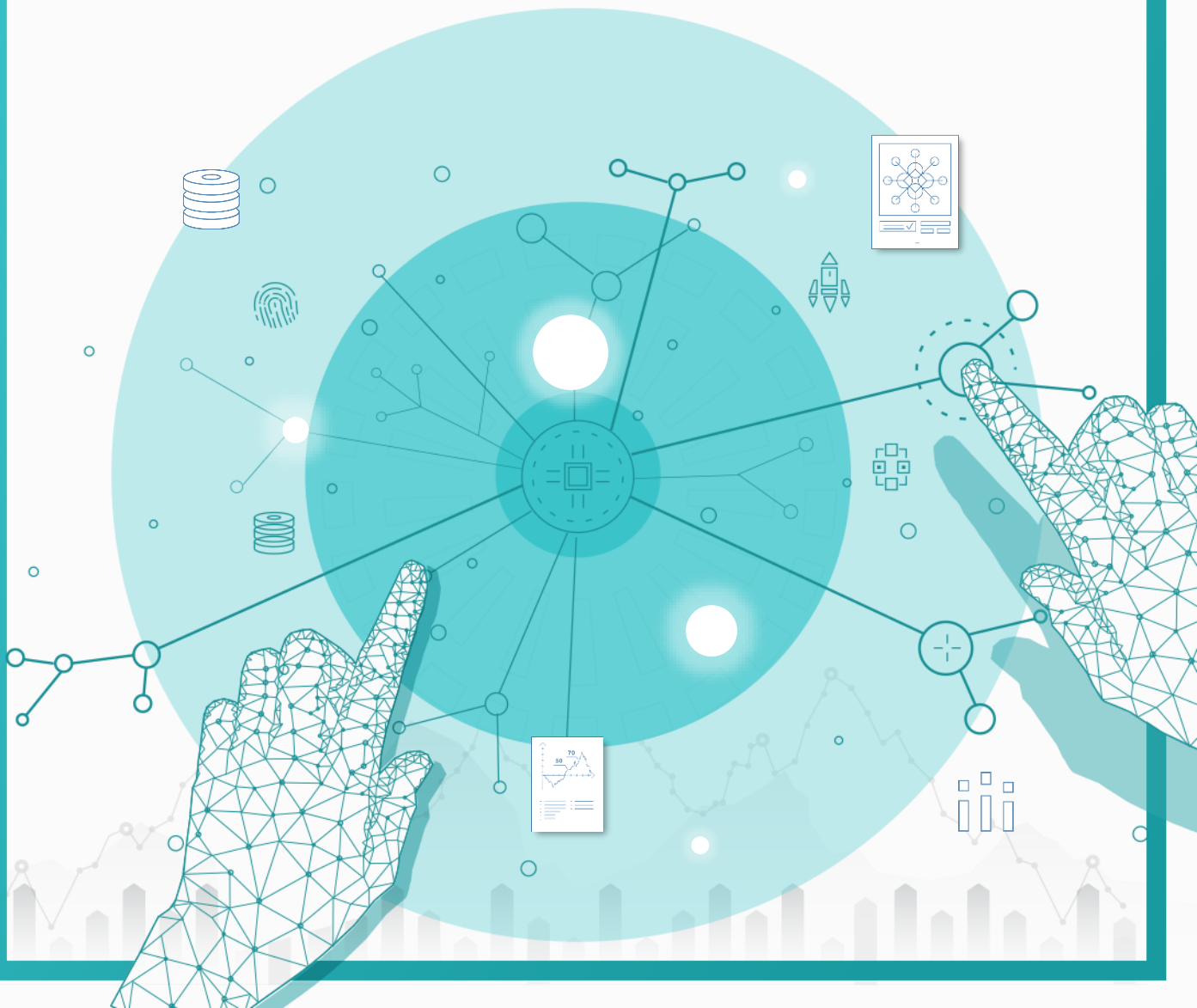
한국기술교육대학교
온라인평생교육원

파이썬 라이브러리로 하는 데이터 분석과 시각화

외부 데이터 분석을 위한 방법



외부 데이터 분석을 위한 방법

학습 목표

1. 파이썬에서 외부 데이터를 가져오거나 저장할 수 있다.
2. 웹 크롤링을 통해 데이터를 수집할 수 있다.
3. API를 활용하여 데이터를 수집할 수 있다.

학습 내용

1. 파이썬을 활용한 외부 데이터 다루기
2. 웹 크롤링을 활용한 데이터 수집
3. API를 활용한 데이터 수집

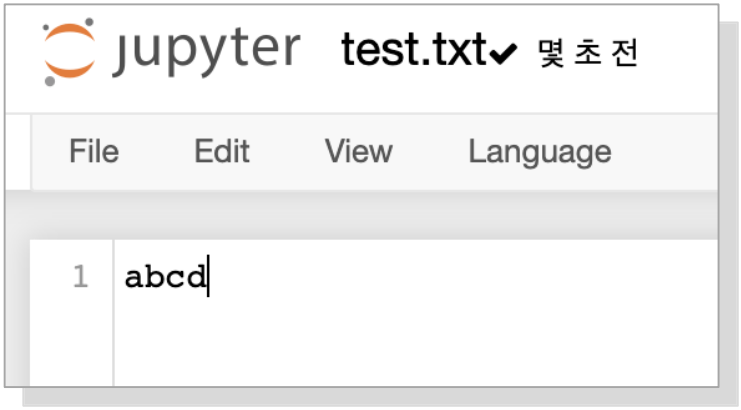
1. 파이썬을 활용한 외부 데이터 다루기

1) 텍스트 파일

(1) open() 함수 활용

- 텍스트 파일 쓰기
  - w모드, write 활용

```
# 쓰기
f = open('test.txt','w')
f.write('abcd')
f.close()
```



  - r모드, read 활용

```
# 읽기
f = open('test.txt','r')
data = f.read()
f.close()
print(data)
```

```
abcd
```

1. 파이썬을 활용한 외부 데이터 다루기

1) 텍스트 파일

(1) open() 함수 활용

- 파일 열기 모드
  - r(읽기 모드): 파일을 읽기만 할 때 사용
  - w(쓰기 모드): 파일에 내용을 쓸 때 사용
  - a(추가 모드): 파일의 마지막에 새로운 내용을 추가할 때 사용
  - x(쓰기 모드): 만약 해당 파일이 있다면 오류 발생
  - b(이진 모드): Binary 형식으로 파일을 열 수 있음
  - +(읽기 및 쓰기 모드): 파일을 읽고 쓸 수 있음

- 다양한 읽기 쓰기 함수
  - write(): 파일에 텍스트 쓰기
  - writelines(): 리스트 등 여러 텍스트를 쓸 때 사용
  - read(): 파일 전체 읽기
  - readline(): 한 번에 한 라인씩 읽기(반복문 등 함께 활용)
  - readlines(): 여러 라인을 리스트에 저장
  - tell(): 파일에서 현재 작업 위치 확인
  - seek(): 파일에서 해당 위치로 이동

2) CSV 파일

(1) 데이터가 콤마로 구분된 형태

- 파이썬의 open 함수 활용
  - 데이터를 문자열 자료형으로 읽어와 CSV 파일 형식의 특징을 살리지 못함
  - 문자열 함수를 활용해 콤마로 구분하여 활용

```
f = open('test.csv','r')
data = f.read()
f.close()
print(data)
print(type(data))
```

```
a,b,c,d
1,2,3,4
<class 'str'>
```

2) CSV 파일

(1) 데이터가 콤마로 구분된 형태

- 파이썬의 CSV 모듈 활용
  - CSV 모듈의 reader 함수로 읽어온 파일을 변환하면 콤마로 구분된 파일을 파이썬의 리스트로 각각 읽어올 수 있음

```
import csv
f = open('test.csv','r')
data = csv.reader(f)
for i in data:
    print(i)
f.close()
```

```
['a', 'b', 'c', 'd']
['1', '2', '3', '4']
```

2) CSV 파일

(1) 데이터가 콤마로 구분된 형태

- Pandas 모듈 활용
  - read_csv 함수: CSV 파일을 Pandas 모듈의 DataFrame 자료형으로 읽어올 수 있음

```
import pandas as pd
data = pd.read_csv('test.csv')
print(data)
print(type(data))
```

```
  a b c d
0 1 2 3 4
<class 'pandas.core.frame.DataFrame'>
```

3) Excel 파일

(1) 별도 모듈을 활용한 엑셀 데이터 로드

- openpyxl 모듈 활용
  - load_workbook 함수를 통해 엑셀을 읽어올 수 있음
  - 엑셀 파일의 Sheet별, Cell별 데이터를 가져올 수 있음

```
import openpyxl as xl

data = xl.load_workbook('test.xlsx')
sheet = data['Sheet1']
print(sheet)
for cell in sheet:
    for i in cell:
        print(i.value)
```

```
<Worksheet "Sheet1">
a
b
c
d
1
2
3
4
```

3) Excel 파일

(1) 별도 모듈을 활용한 엑셀 데이터 로드

- Pandas 모듈 활용
  - read_excel 함수를 활용하여 엑셀을 읽어올 수 있음
  - 데이터는 DataFrame 자료형으로 저장됨
  - 옵션을 활용하여 특정 시트, 특정 컬럼 등 데이터를 세분화하여 가져올 수 있음

```
import pandas as pd

data = pd.read_excel('test.xlsx')
print(data)
print(type(data))
```

```
  a  b  c  d
0  1  2  3  4
<class 'pandas.core.frame.DataFrame'>
```

4) JSON 형식 파일

(1) 키-값으로 이루어진 데이터를 표현하기 위한 표준 포맷

- 구성요소
  - 숫자
  - 배열
  - 문자열
  - 참/거짓
  - 객체
  - null

4) JSON 형식 파일

(2) 파이썬의 JSON 모듈 활용

- 파이썬 객체와 JSON 간의 객체 변환
  - 파이썬 → JSON 변환: dumps 함수
  - JSON → 파이썬 변환: loads 함수

```
import json
data = [1,2,3,{'a':1,'b':2}]
json_data = json.dumps(python_data)
print(json_data)
print(type(json_data))

python_data = json.loads(json_data)
print(python_data)
print(type(python_data))
```

```
[1, 2, 3, {"a": 1, "b": 2}]
<class 'str'>
[1, 2, 3, {'a': 1, 'b': 2}]
<class 'list'>
```

4) JSON 형식 파일

(2) 파이썬의 JSON 모듈 활용

- 파이썬 객체와 JSON 간의 객체 변환

파이썬	JSON
dict	오브젝트(object)
list, tuple	배열(array)
str	문자열(string)
int, float	숫자(number)
True	true
False	false
None	null

JSON	파이썬
오브젝트(object)	dict
배열(array)	list
문자열(string)	str
숫자(정수)	int
숫자(실수)	float
true	True
false	False
null	None

5) Pickle 모듈을 활용한 파일 저장 및 활용

(1) open 함수 한계점

- 데이터는 문자열 형태로 저장해야 함

➡ 추후 다시 활용할 경우, 전처리 과정이 필요함

(데이터에 따라 CSV, Excel, JSON 등 파일 형식을 다르게 하여 활용)

(2) Pickle 모듈 활용

- 데이터를 파이썬 객체 형태 그대로 저장할 수있음

➡ 나중에 다시 불러와도 별도 전처리 과정 없이 바로 활용할 수 있음

```
data = ['a', 'b', 'c']
f = open('test.txt', 'w')
f.write(data)
f.close()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-62-ce16531843c3> in <module>
      1 data = ['a', 'b', 'c']
      2 f = open('test.txt', 'w')
----> 3 f.write(data)
      4 f.close()

TypeError: write() argument must be str, not list
```

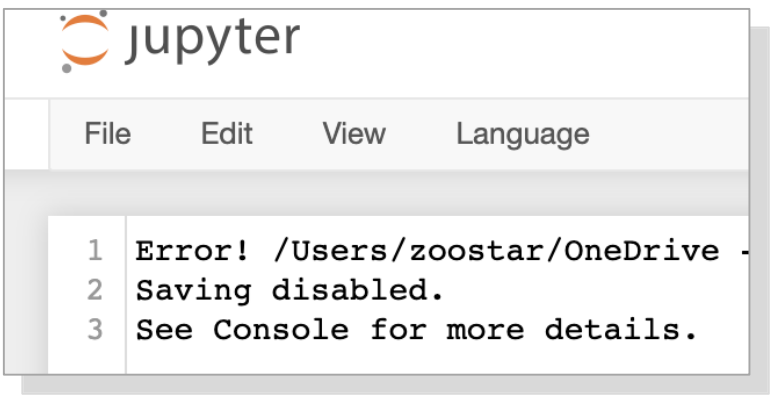
- write 함수: 문자열만 저장할 수 있음

5) Pickle 모듈을 활용한 파일 저장 및 활용

(3) 파일 쓰기

- Pickle 모듈을 활용
- 데이터를 dump 함수로 저장할 수 있음(Binary 형태)

```
import pickle
data = ['a', 'b', 'c']
f = open('pickle.txt', 'wb')
pickle.dump(data, f)
f.close()
```



5) Pickle 모듈을 활용한 파일 저장 및 활용

(4) 파일 읽기

- load 함수를 활용
- 데이터를 파이썬 객체 그대로 읽어올 수 있음

```
import pickle
f = open("pickle.txt","rb")
data = pickle.load(f)
print(data)
print(type(data))
```

```
['a', 'b', 'c']
<class 'list'>
```

2. 웹 크롤링을 활용한 데이터 수집

"파이썬에는 웹 크롤링을 위한 다양한 모듈을 제공하고 있으며
이 모듈들을 활용해 홈페이지 내 데이터를 수집할 수 있음"

1) Requests 모듈

(1) HTTP 요청과 관련된 여러 가지 함수 제공

- get 함수
  - 홈페이지에 요청하여 텍스트 데이터를 가져올 수 있음

```
import requests

url = 'https://www.naver.com'
req = requests.get(url)
print(req)
```

```
<Response [200]>
```

```
print(req.text)
```

```
<!doctype html>                    <html lang="ko" data-da
="viewport" content="width=1190"> <meta name="app
이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요"/>
ontent="https://s.pstatic.net/static/www/mobile/edit/20
요"/> <meta name="twitter:card" content="summary"> <
```

1) Requests 모듈

(1) HTTP 요청과 관련된 여러 가지 함수 제공

- get 함수
  - 문자열 자료형으로 이루어진 홈페이지 소스 데이터를 가져옴

```
print(type(req.text))
print(req.text.find('네이버'))
```

```
<class 'str'>
365
```

➡ 문자열 함수 등을 활용하여 텍스트 내 원하는 정보를 추출, 분석하여 활용할 수 있음

2) BS4 모듈

(1) 태그 기반의 다양한 함수 제공

- 홈페이지 태그 기반의 파싱
  - 태그를 통해 텍스트 데이터를 쉽게 검색, 분석할 수 있음

```
import bs4
data = "<div>안녕하세요</a>"
soup = bs4.BeautifulSoup(data)
print(soup)
```

```
<html><body><div>안녕하세요</div></body></html>
```

```
print(soup.find('div'))
```

```
<div>안녕하세요</div>
```

2) BS4 모듈

(2) 데이터에 맞는 태그 사용

- 데이터의 패턴 분석
  - 홈페이지 내에서 원하는 정보를 가져오기 위해 패턴에 맞는 태그 사용

(예) 네이버 메인 페이지에서 issue 클래스를 가진 a 태그의 첫 번째 데이터의 텍스트 데이터 추출

```
import bs4
import requests
url = 'https://www.naver.com'
req = requests.get(url)
soup = bs4.BeautifulSoup(req.text)
print(soup.find('a',attrs={'class':'issue'}).text)
```

3) Selenium 모듈

(1) Selenium 모듈

- 홈페이지 테스트를 위한 프레임워크
- 브라우저를 직접 프로그래밍하여 조작할 수 있음
- 브라우저에 맞는 웹드라이버 설치 필요

(2) Selenium 모듈 활용 예시

(예) 크롬(Chrome) 브라우저를 활용한 Selenium 모듈

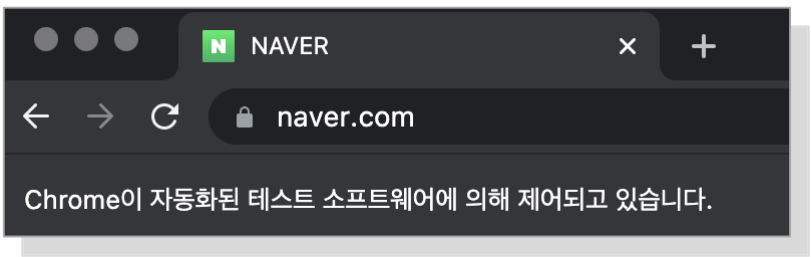
- Chrome 웹 브라우저 기준 환경설정 방법
  ① 브라우저 버전 확인
  ② 사이트에서 본인 PC, 브라우저에 맞는 드라이버 다운로드
  (https://chromedriver.chromium.org/downloads)

3) Selenium 모듈

(2) Selenium 모듈 활용 예시

```
import selenium
from selenium import webdriver
url='https://www.naver.com'
driver = webdriver.Chrome('/Users/zoostar/Downloads/chromedriver')
driver.get(url)
result = driver.page_source
print(result)
```

```
<html lang="ko" data-dark="false" data-useragent="Mozilla/5.0 (Maci
me/92.0.4515.159 Safari/537.36"><head> <meta charset="utf-8"> <tit
eta name="viewport" content="width=1190"> <meta name="apple-m
ofollow"> <meta name="description" content="네이버 메인에서 다양한
```



3) Selenium 모듈

(3) Selenium 모듈의 함수 기능

- 브라우저 조작 및 데이터 선택
  - 키보드 입력, 클릭, 페이지 이동 등
  - find_elements_by_(속성): id, class, name 등 속성을 기반으로 특정 요소에 접근
  - send_keys: 선택한 요소에 키보드로 입력할 수 있음
  - click: 선택한 요소를 마우스로 클릭할 수 있음
  - forward, back: 브라우저를 앞으로, 뒤로 가기 할 수 있음
  - minimize, maximize_window: 브라우저를 최소화, 최대화할 수 있음
  - 기타 등등

3. API를 활용한 데이터 수집

1) API의 개념

(1) API(Application Programming Interface)란?

- 응용 프로그램에서 사용할 수 있도록 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻함

(예) 전국 맛집 지도를 보여주는 프로그램을 개발할 때 필요한 정보는?

① 맛집 정보를 알려줘야 하므로 맛집 데이터를 수집해야 함

② 맛집별 현재 날씨, 위치 등을 알려주면 좋은데 일반 사람이 날씨 정보나 지도 정보를 얻기는 어려움

➡ 기상청 등에서 제공해 주는 API를 활용하여 원하는 데이터를 가져와 사용할 수 있음

(예) 카카오톡 API, 네이버 지도 API, 기상청 날씨 API 등

3. API를 활용한 데이터 수집

2) API의 종류 및 사용법

(1) 공공API 제공(공공데이터포털)

- 전국의 공공 시설들의 정보
- 버스 시간 데이터 등

➡ 국가에서도 다양한 데이터를 활용할 수 있도록 API화하여 제공함

(2) API 사용법

- 홈페이지 기반의 JSON 형식으로 제공되는 경우가 많음
  - API를 제공해 주는 곳마다 다름

(3) API 활용 허가

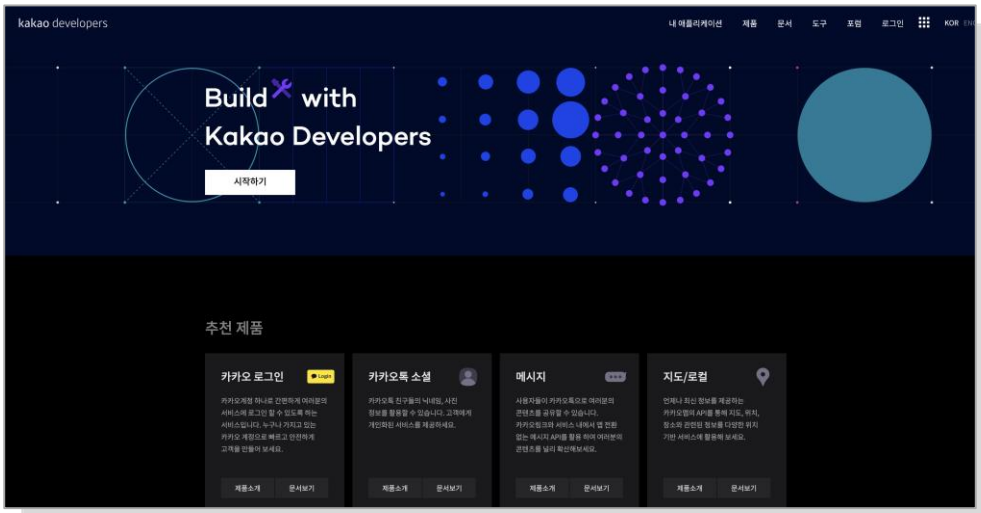
- 인증 단계를 거친 사용자에게만 허용
  - 사이트 가입
  - 인증 키 발급

➡ 아무나 데이터를 가져가거나 오·남용하는 경우 방지

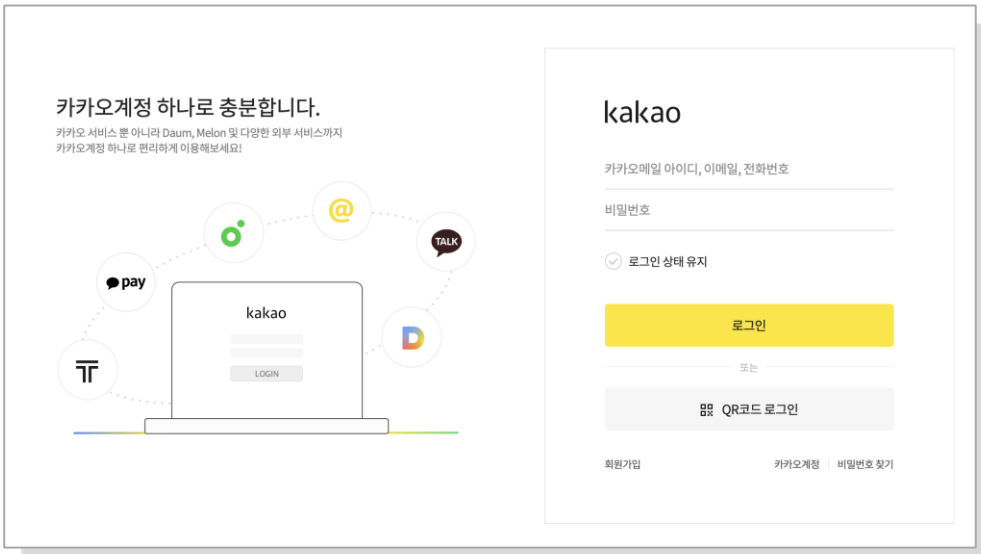
3. API를 활용한 데이터 수집

3) 카카오 API 활용 방법

(1) 카카오 개발자 페이지 접속(https://developers.kakao.com)



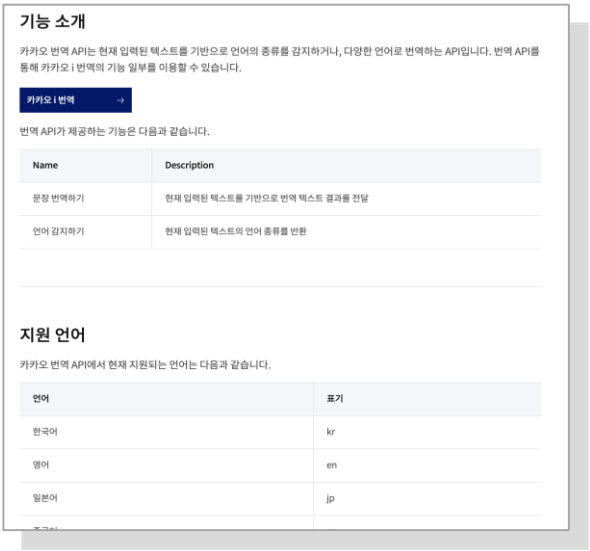
(2) 카카오 계정을 활용한 로그인



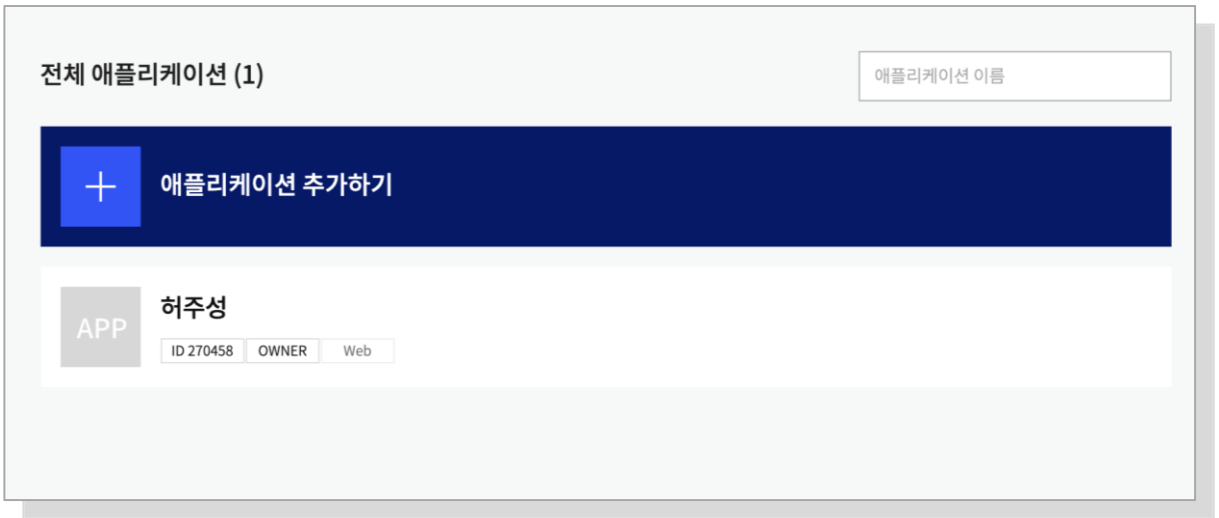
3) 카카오 API 활용 방법

(3) 활용할 API 선정 및 개발 가이드 확인

(예) 번역 API 활용하기
(https://developers.kakao.com/docs/latest/ko/translate/common)



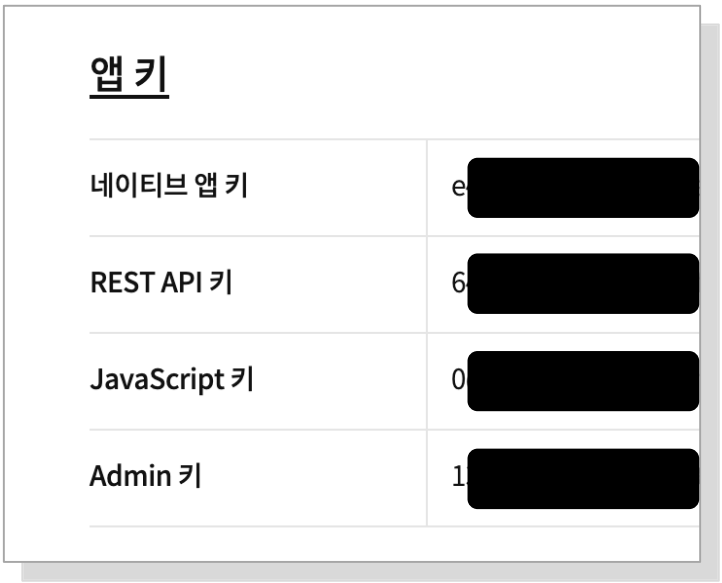
(4) 애플리케이션 생성



3) 카카오 API 활용 방법

(5) API 키 발급 및 확인

- 발급받은 KEY 값은 주기적으로 변경해 주는 것이 좋음
- KEY 값은 절대로 외부로 노출하면 안 됨



3) 카카오 API 활용 방법

(6) 파이썬의 Requests 모듈을 활용한 API 활용

- 카카오에서 필요로 하는 데이터에 맞게 Requests 모듈을 활용하여 요청(header, query 정보 등)

Request

URL

```
GET /v2/translation/translate HTTP/1.1
Host: dapi.kakao.com
Authorization: KakaoAK {REST_API_KEY}
```

```
POST /v2/translation/translate HTTP/1.1
Host: dapi.kakao.com
Authorization: KakaoAK {REST_API_KEY}
Content-type: application/x-www-form-urlencoded
```

Parameter

Name	Type	Description	Required
query	String	번역 대상 문장 최대 5,000자	O
src_lang	String	번역 대상 언어	O
target_lang	String	번역 결과 언어	O

Response

Key

Name	Type	Description
translated_text	List of String[]	번역결과 리스트, 문단 단위의 리스트가 문단 순서대로 정렬

Sample

Request: GET 방식

```
curl -v --get "https://dapi.kakao.com/v2/translation/translate" \
    -H "Authorization: KakaoAK {REST_API_KEY}" \
    -d "src_lang=kr" \
    -d "target_lang=en" \
    --data-urlencode "query=지난해 3월 오픈한 카카오톡 주문하기는 현재까지 약 250만명의 회원을 확보
```

3) 카카오 API 활용 방법

(6) 파이썬의 Requests 모듈을 활용한 API 활용

- API 결과 값은 JSON 형식의 데이터로 제공

```
import requests

url = 'https://dapi.kakao.com/v2/translation/translate'
header = {'Authorization':'KakaoAK 발급받은 API KEY'}
query = {'query' : '안녕하세요. 파이썬을 활용한 데이터 시각화 과정입니다.',
        'src_lang':'kr',
        'target_lang':'en'}
req = requests.get(url,headers=header,params=query)
```

```
print(req.text)
```

```
{"translated_text":[["Hello, this is a data visualization process using Python."]]}
```