

2021년도

ICT 이노베이션스퀘어 확산 사업

[인공지능 : 기초부터 실전까지]

▶ 17차수 ~ 20차수 강의 교안

※ 본 교안은 강의 수강 용도로만 사용 가능합니다.
상업적 이용을 일절 금함.

15

웹 API

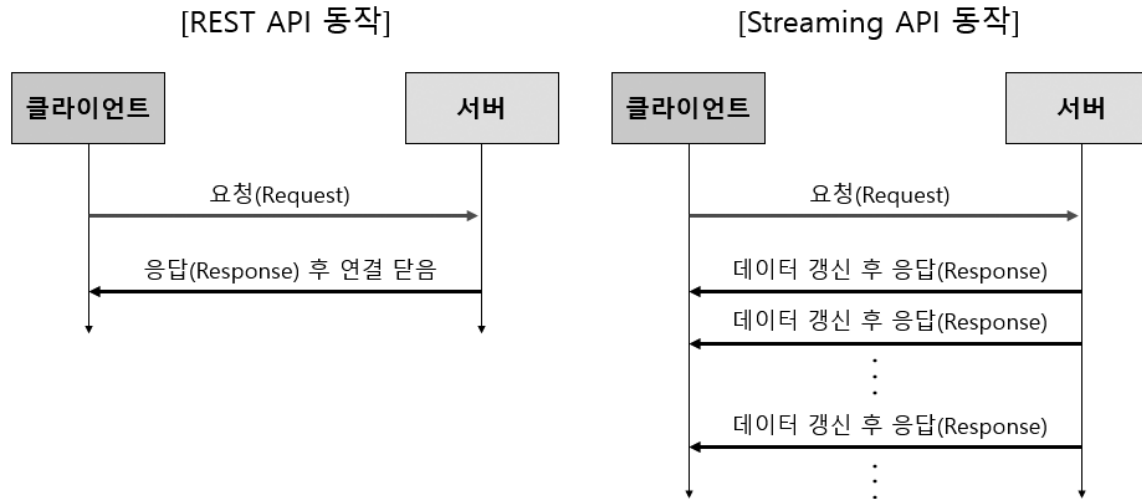


웹 API의 이해

- API(Application Programming Interface, 응용 프로그램 프로그래밍 인터페이스)
 - 응용 프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스
- 웹 API: 웹 애플리케이션 개발에서 다른 서비스에 요청을 보내고 응답을 받기 위해 정의된 명세

웹 API의 이해

- 웹 API의 데이터 획득 과정



웹 API의 이해

- 웹 API의 인증 방식
 - 초기에는 아이디와 비밀번호를 통해 인증하거나 웹 API별로 제각기 다른 인증 방식을 사용
 - 보안과 호환성의 문제로 OAuth가 탄생
 - OAuth는 외부에서 해당 서비스에 접속하는 모바일, 데스크톱, 웹 애플리케이션(응용 프로그램)의 보안 인증을 허용하는 개방형 인증 규약
 - OAuth 인증 방식에서는 API 키(Key)와 접속 토큰(Access Token), 비밀번호를 이용해 애플리케이션별로 인증을 수행하고 서비스를 이용
 - 트위터 API의 예
 - Consumer Key (API Key): L9WwxSUZXGt7vIXEiYmtljcHQ
 - Consumer Secret (API Secret): bjQrLuottDdpg94VWcG1I3LWXUYZNznzCRE7KvFXR6hzuQcjOh
 - Access Token: 503869961253584201-zsbIDKpTNdFCp5Q9JWYhi0MqtoTFqg2
 - Access Token Secret: XuJ8EYgtim6wye7fYuTrQuY80x1TS2hhPHXAtMzeuPRsb

웹 API의 이해

- 응답 데이터의 형식 및 처리
 - JSON 형식의 데이터 처리
 - JSON(JavaScript Object Notation)

```
{  
  "이름": "홍길동",  
  "나이": 25,  
  "거주지": "서울",  
  "신체정보": {  
    "키": 175.4,  
    "몸무게": 71.2  
  },  
  "취미": [  
    "등산",  
    "자전거타기",  
    "독서"  
  ]  
}
```

```
json.dumps(python_data [, indent=n, sort_keys=True or False, ensure_ascii=True or False])
```

```
json.loads(json_data)
```

웹 API의 이해

– JSON 형식의 데이터 처리

```
In: import json
python_dict = {
    "이름": "홍길동",
    "나이": 25,
    "거주지": "서울",
    "신체정보": {
        "키": 175.4,
        "몸무게": 71.2
    },
    "취미": [
        "등산",
        "자전거타기",
        "독서"
    ]
}
type(python_dict)
```

Out: dict

```
In: json_data = json.dumps(python_dict)
type(json_data)
```

Out: str

웹 API의 이해

– JSON 형식의 데이터 처리

```
In: print(json_data)
Out: { "Wuc774Wub984": "Wud64dWuae38Wub3d9", "Wub098Wuc774": 25, "Wuac70Wuc8fcWuc9c0":
      "Wuc11cWuc6b8", "Wuc2e0Wuccb4Wuc815Wubcf4": {"Wud0a4": 175.4,
      "Wubab8Wubb34Wuac8c": 71.2},
      "Wucde8Wubbf8": ["Wub4f1Wuc0b0", "Wuc790Wuc804Wuac70Wud0c0Wuae30".
In: json_data = json.dumps(python_dict, indent=3, sort_keys=True, ensure_ascii=False)
    print(json_data)
Out: {
  "거주지": "서울",
  "나이": 25,
  "신체정보": {
    "몸무게": 71.2,
    "키": 175.4
  },
  "이름": "홍길동",
  "취미": [
    "등산",
    "자전거타기",
    "독서"
  ]
}
```


웹 API의 이해

– JSON 형식의 데이터 처리

```
In: json_dict = json.loads(json_data)
    type(json_dict)
Out: dict
```

```
In: json_dict['신체정보']['몸무게']
Out: 71.2
```

```
In: json_dict['취미']
Out: ['등산', '자전거타기', '독서']
```

```
In: json_dict['취미'][0]
Out: '등산'
```

웹 API의 이해

– XML 형식의 데이터 처리

- XML(eXtensible Markup Language): 데이터 저장 및 전달을 위해 만든 다목적 마크업 언어

```
<?xml version="1.0" encoding="UTF-8" ?>
<사용자정보>
  <이름>홍길동</이름>
  <나이>25</나이>
  <거주지>서울</거주지>
  <신체정보>
    <키 unit="cm">175.4</키>
    <몸무게 unit="kg">71.2</몸무게>
  </신체정보>
  <취미>등산</취미>
  <취미>자전거타기</취미>
  <취미>독서</취미>
</사용자정보>
```

```
xmltodict.parse(xml_input [, xml_attribs=True 혹은 False])
```

웹 API의 이해

– XML 형식의 데이터 처리

```
In: xml_data = """<?xml version="1.0" encoding="UTF-8" ?>
```

```
<사용자정보>
```

```
<이름>홍길동</이름>
```

```
<나이>25</나이>
```

```
<거주지>서울</거주지>
```

```
<신체정보>
```

```
<키 unit="cm">175.4</키>
```

```
<몸무게 unit="kg">71.2</몸무게>
```

```
</신체정보>
```

```
<취미>등산</취미>
```

```
<취미>자전거타기</취미>
```

```
<취미>독서</취미>
```

```
</사용자정보>
```

```
"""
```

```
print(xml_data)
```

```
Out: <?xml version="1.0" encoding="UTF-8" ?>
```

```
<사용자정보>
```

```
<이름>홍길동</이름>
```

```
<나이>25</나이>
```

```
<거주지>서울</거주지>
```

```
<신체정보>
```

```
<키 unit="cm">175.4</키>
```

```
<몸무게 unit="kg">71.2</몸무게>
```

```
</신체정보>
```

```
<취미>등산</취미>
```

```
<취미>자전거타기</취미>
```

```
<취미>독서</취미>
```

```
</사용자정보>
```

웹 API의 이해

– XML 형식의 데이터 처리

```
In: import xmltodict
    dict_data = xmltodict.parse(xml_data, xml_attribs=True)
    dict_data
Out: OrderedDict([('사용자정보',
    OrderedDict([('이름', '홍길동'),
    ('나이', '25'),
    ('거주지', '서울'),
    ('신체정보',
    OrderedDict([('키',
    OrderedDict([('@unit', 'cm'),
    ('#text', '175.4')])),
    ('몸무게',
    OrderedDict([('@unit', 'kg'),
    ('#text', '71.2')])))])),
    ('취미', ['등산', '자전거타기', '독서'])]))])
```

```
In: dict_data['사용자정보']['이름']
Out: '홍길동'
```

```
In: dict_data['사용자정보']['신체정보']
Out: OrderedDict([('키', OrderedDict([('@unit', 'cm'), ('#text', '175.4')])),
    ('몸무게', OrderedDict([('@unit', 'kg'), ('#text', '71.2')]))])
```

웹 API의 이해

– XML 형식의 데이터 처리

```
In: dict_data['사용자정보']['신체정보']['키']['@unit']
```

```
Out: 'cm'
```

```
In: dict_data['사용자정보']['신체정보']['키']['#text']
```

```
Out: '175.4'
```

```
In: import xmltodict
```

```
dict_data = xmltodict.parse(xml_data)
```

```
user_name = dict_data['사용자정보']['이름']
```

```
body_data = dict_data['사용자정보']['신체정보']
```

```
height = body_data['키']['#text']
```

```
height_unit = body_data['키']['@unit']
```

```
weight = body_data['몸무게']['#text']
```

```
weight_unit = body_data['몸무게']['@unit']
```

```
print("[사용자 {0}의 신체정보"].format(user_name))
```

```
print("*키: {0}{1}".format(height, height_unit))
```

```
print("*몸무게: {0}{1}".format(weight, weight_unit))
```

```
Out: [사용자 홍길동의 신체정보]
```

```
*키: 175.4cm
```

```
*몸무게: 71.2kg
```

웹 API의 이해

– XML 형식의 데이터 처리

```
In: dict_data2 = xmltodict.parse(xml_data, xml_attribs=False)
    dict_data2
```

```
Out: OrderedDict([('사용자정보',
    OrderedDict([('이름', '홍길동'),
        ('나이', '25'),
        ('거주지', '서울'),
        ('신체정보',
            OrderedDict([('키', '175.4'), ('몸무게', '71.2')])),
        ('취미', ['등산', '자전거타기', '독서'])]))])
```

웹 API의 이해

- 웹 사이트 주소에 부가 정보 추가하기
 - 웹 사이트 주소에 경로 추가하기

```
url = "https://api.github.com/"  
r = requests.get(url)
```

```
In: base_url = "https://api.github.com/"  
    sub_dir = "events"  
    url = base_url + sub_dir  
    print(url)
```

Out: <https://api.github.com/events>

```
In: import requests  
    base_url = "https://api.github.com/"  
    sub_dirs = ["events", "user", "emails" ]  
    for sub_dir in sub_dirs:  
        url_dir = base_url + sub_dir  
        r = requests.get(url_dir)  
        print(r.url)
```

Out: <https://api.github.com/events>
<https://api.github.com/user>
<https://api.github.com/emails>

웹 API의 이해

– 웹 사이트 주소에 매개변수 추가하기

```
In: import requests
```

```
LAT = '37.57' # 위도
```

```
LON = '126.98' # 경도
```

```
API_KEY = 'b235c57pc357fb68acr1e81' # API 키(임의의 API 키)
```

```
UNIT = 'metric' # 단위
```

```
site_url = "http://api.openweathermap.org/data/2.5/weather"
```

```
parameter = "?lat=%s&lon=%s&appid=%s&units=%s"%(LAT, LON, API_KEY, UNIT)
```

```
url_para = site_url + parameter
```

```
r = requests.get(url_para)
```

```
print(r.url)
```

Out:

```
http://api.openweathermap.org/data/2.5/weather?lat=37.57&lon=126.98&appid=b235c57pc357fb68acr  
1e81&units=metric
```

```
url = 'http://abc.org/get'
```

```
req_parameter= {'key1': 'value1', 'key2': 'value2'}
```

```
r = requests.get(url, params=req_parameter)
```


웹 API의 이해

– 웹 사이트 주소에 매개변수 추가하기

```
In: import requests
```

```
LAT = '37.57' # 위도
```

```
LON = '126.98' # 경도
```

```
API_KEY = 'b235c57pc357fb68acr1e81' # API 키(임의의 API 키)
```

```
UNIT = 'metric' # 단위
```

```
req_url = "http://api.openweathermap.org/data/2.5/weather"
```

```
req_parameter = {"lat":LAT, "lon":LON , "appid": API_KEY, "units":UNIT}
```

```
r = requests.get(req_url,params=req_parameter)
```

```
print(r.url)
```

Out:

```
http://api.openweathermap.org/data/2.5/weather?lat=37.57&lon=126.98&appid=b235c57pc357fb68acr1e81&units=metric
```

웹 API의 이해

– 웹 사이트 주소의 인코딩과 디코딩

```
In: import requests
```

```
API_KEY = "et5piq3pfpqLEWPpCbvtSQ%2Bertert%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoewRV%2Fovmrk3dq%3D%3D"
API_KEY_decode = requests.utils.unquote(API_KEY)
print("Encoded url:", API_KEY)
print("Decoded url:", API_KEY_decode)
```

```
Out: Encoded url: et5piq3pfpqLEWPpCbvtSQ%2Bertert%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoewRV%2Fovmrk3dq%3D%3D
Decoded url: et5piq3pfpqLEWPpCbvtSQ+ertertg+x3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw+9rkvoewRV/ovmrk3dq==
```

```
In: req_url = "http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfoInquireSvc/getNearbyMsrstnList"
tm_x = 244148.546388
tm_y = 412423.75772
req_parameter = {"ServiceKey":API_KEY_decode, "tmX":tm_x, "tmY":tm_y}
r = requests.get(req_url, params = req_parameter)
print(r.url)
```

```
Out: http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfoInquireSvc/getNearbyMsrstnList?
```

```
ServiceKey=et5piq3pfpqLEWPpCbvtSQ%2Bertert%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoew
```

```
RV%2Fovmrk3dq%3D%3D&tmX=244148.546388&tmY=412423.75772
```

웹 API의 이해

– 웹 사이트 주소의 인코딩과 디코딩

```
In: req_parameter = {"ServiceKey":API_KEY, "tmX":tm_x, "tmY":tm_y}
    r = requests.get(req_url, params = req_parameter)
```

```
    print(r.url)
```

```
Out: http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfolnquireSvc/getNearbyMsrstnList?
```

```
ServiceKey=et5piq3pfpqLEWPpCbvtSQ%252Bertertg%252Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw
%252B9rkvoewRV%252Fovmrk3dq%253D%253D&tmX=244148.546388&tmY=412423.75772
```

API 키를 사용하지 않고 데이터 가져오기

- 국제 우주 정거장의 정보 가져오기

```
In: import requests  
import json
```

```
url = "http://api.open-notify.org/iss-now.json"
```

```
r = requests.get(url)  
print(r.text)
```

```
Out: '{"message": "success", "iss_position": {"latitude": "35.9958", "longitude": "-111.8448"},  
      "timestamp": 1529196236}'
```

```
In: json_to_dict = json.loads(r.text)  
type(json_to_dict)
```

```
Out: dict
```

```
In: import requests
```

```
url = "http://api.open-notify.org/iss-now.json"
```

```
r = requests.get(url)  
json_to_dict = r.json()  
type(json_to_dict)
```

```
Out: dict
```

API 키를 사용하지 않고 데이터 가져오기

- 국제 우주 정거장의 정보 가져오기

```
In: import requests
```

```
url = "http://api.open-notify.org/iss-now.json"
```

```
json_to_dict = requests.get(url).json()  
type(json_to_dict)
```

```
Out: dict
```

```
In: json_to_dict
```

```
Out: {'iss_position': {'latitude': '35.6611', 'longitude': '-111.3872'},  
      'message': 'success',  
      'timestamp': 1529196244}
```

```
In: print(json_to_dict["iss_position"])  
    print(json_to_dict["iss_position"]["latitude"])  
    print(json_to_dict["iss_position"]["longitude"])  
    print(json_to_dict["message"])  
    print(json_to_dict["timestamp"])
```

```
Out: {'latitude': '35.6611', 'longitude': '-111.3872'}  
35.6611  
-111.3872  
success  
1529196244
```

API 키를 사용하지 않고 데이터 가져오기

- 국제 우주 정거장의 정보 가져오기

```
In: import requests
import time

url = "http://api.open-notify.org/iss-now.json"

def ISS_Position(iss_position_api_url):
    json_to_dict = requests.get(iss_position_api_url).json()
    return json_to_dict["iss_position"]

for k in range(5):
    print(ISS_Position(url))
    time.sleep(10)    # 10초 동안 코드 실행을 일시적으로 중지한다.
```

```
Out: {'latitude': '35.3034', 'longitude': '-110.9056'}
{'latitude': '34.8585', 'longitude': '-110.3170'}
{'latitude': '34.4318', 'longitude': '-109.7628'}
{'latitude': '33.9807', 'longitude': '-109.1874'}
{'latitude': '33.5265', 'longitude': '-108.6186'}
```

API 키를 사용하지 않고 데이터 가져오기

- 국가 정보 가져오기

```
In: import requests
```

```
url_temp = "https://restcountries.eu/rest/v1/name/"  
country = "South Korea"  
url = url_temp + country
```

```
r = requests.get(url)  
print(r.text)
```

```
Out: '[{"name":"South Korea", "topLevelDomain":[".kr"], "alpha2Code":"KR", "alpha3Code":"KOR",  
      "callingCodes":["82"], "capital":"Seoul", "altSpellings":["KR", "Republic of Korea"],  
      "region":"Asia", "subregion":"Eastern Asia", "population":51448183, "latIng" :[37.0,127.5],  
      "demonym":"South Korean", "area":100210.0, "gini":31.3, "timezones":["UTC+09:00"],  
      "borders":["PRK"], "nativeName":"대한민국", "numericCode":"410", "currencies":["KRW"],  
      "languages":["ko"], "translations":{"de":"S dkorea", "es":"Corea del Sur",  
      "fr":"Cor e du Sud", "ja":"", "it":"Corea del Sud"}, "relevance":"1.5"}]'
```

```
In: json_to_list = requests.get(url).json()  
    json_to_list
```

```
Out: [{'alpha2Code': 'KR',  
      'alpha3Code': 'KOR',  
      'altSpellings': ['KR', 'Republic of Korea'],  
      'area': 100210.0,  
      'borders': ['PRK'],
```

API 키를 사용하지 않고 데이터 가져오기

- 국가 정보 가져오기

```
'callingCodes': ['82'],  
'capital': 'Seoul',  
'currencies': ['KRW'],  
'demonym': 'South Korean',  
'gini': 31.3,  
'languages': ['ko'],  
'latlng': [37.0, 127.5],  
'name': 'South Korea',  
'nativeName': '대한민국',  
'numericCode': '410',  
'population': 51448183,  
'region': 'Asia',  
'relevance': '1.5',  
'subregion': 'Eastern Asia',  
'timezones': ['UTC+09:00'],  
'topLevelDomain': ['.kr'],  
'translations': {'de': 'S dkorea',  
'es': 'Corea del Sur',  
'fr': 'Cor e du Sud',  
'it': 'Corea del Sud',  
'ja': ' ' }}}
```

In: json_to_list[0]["capital"]

Out: 'Seoul'

API 키를 사용하지 않고 데이터 가져오기

- 국가 정보 가져오기

```
In: import requests
```

```
import json
```

```
countries = ["South Korea", "United States of America", "United Kingdom", "France", "Germany"]
```

```
def country_to_capital(country):
```

```
    url_temp = "https://restcountries.eu/rest/v1/name/"
```

```
    url = url_temp + country
```

```
    json_to_list = requests.get(url).json()
```

```
    return json_to_list[0]["capital"]
```

```
for country in countries:
```

```
    capital = country_to_capital(country)
```

```
    print("{}: {}".format(country, capital))
```

```
Out: *South Korea: Seoul
```

```
*United States of America: Washington, D.C.
```

```
*United Kingdom: London
```

```
*France: Paris
```

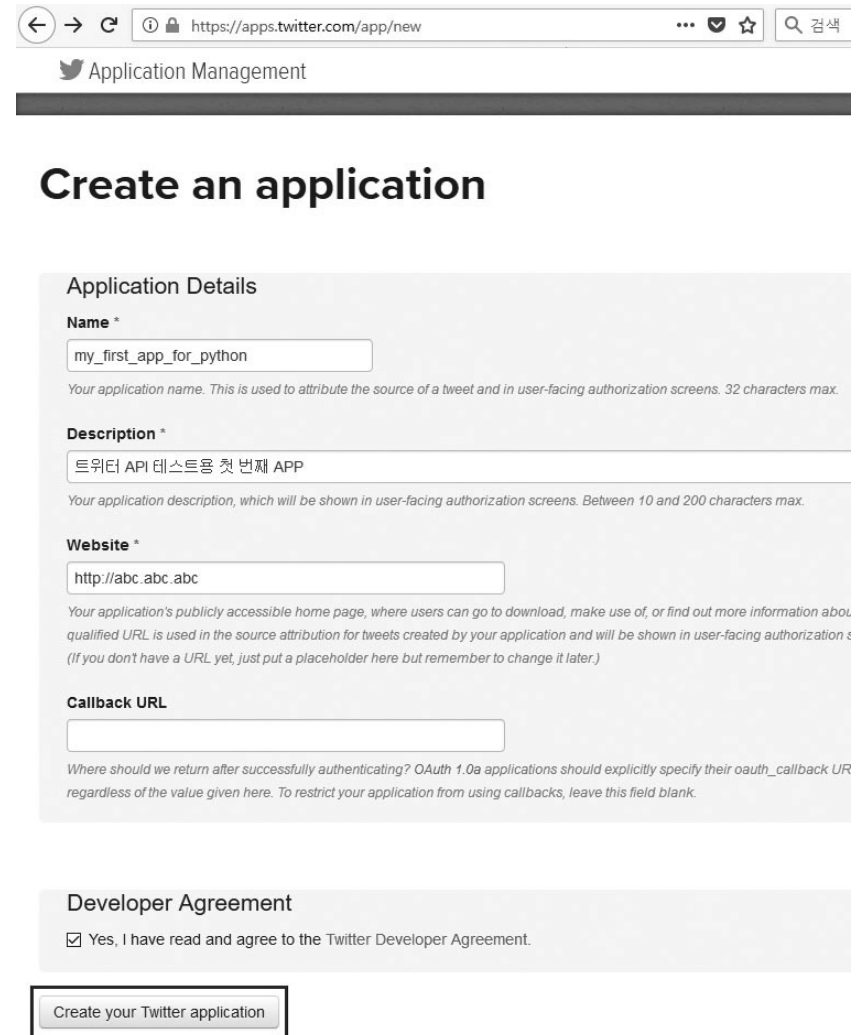
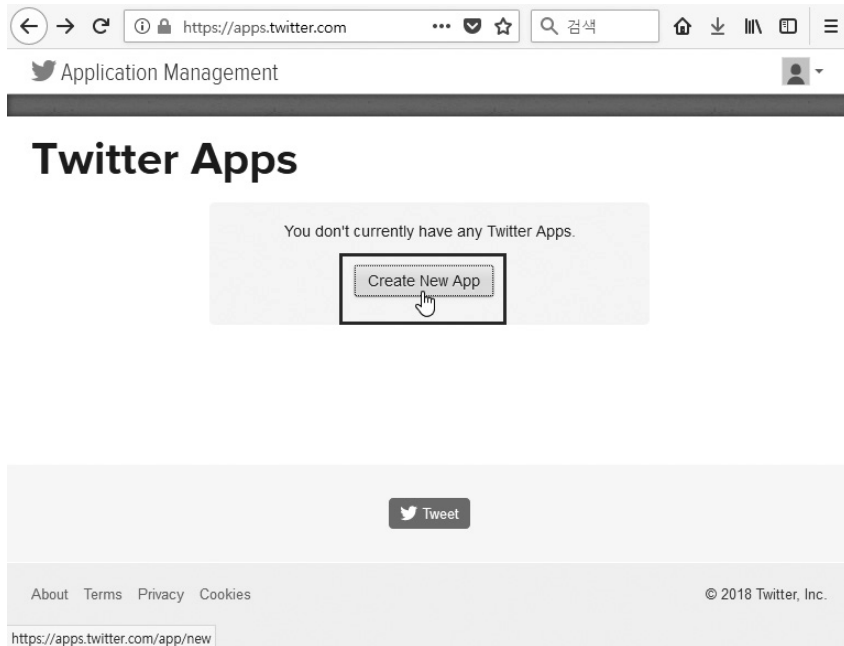
```
*Germany: Berlin
```

트위터에 메시지 작성하고 가져오기

- 트위터(Twitter)
 - 대표적인 소셜 네트워킹 서비스(Social Networking Service, SNS) 중 하나
 - 다른 사람을 팔로우(Follow)
 - 다른 사람을 팔로우하는 사람을 팔로워(Follower)
 - 팔로워가 있다면 자신이 쓴 글은 즉시 팔로워들에게 공개

트위터에 메시지 작성하고 가져오기

- API 키 및 접속 토큰 생성



트위터에 메시지 작성하고 가져오기

- API 키 및 접속 토큰 생성


Application Management

Your application has been created. Please take a moment to review and adjust your application's settings.

my_first_app_for_python

Test OAuth

Details Settings Keys and Access Tokens Permissions

 트위터 API 테스트용 첫 번째 APP
http://abc.abc.abc

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	<input type="text" value="idH5avillmC6u4d8Pnq4tCq3anagp1eys and more"/>

Application Management

my_first_app_for_python

Test OAuth

Details Settings Keys and Access Tokens Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	<input type="text" value="idH5avillmC6u4d8Pnq4tCq3anagp1eys and more"/>
Consumer Secret (API Secret)	<input type="text" value="idH5avillmC6u4d8Pnq4tCq3anagp1eys and more"/>
Access Level	Read and write (modify app permissions)
Owner	<input type="text" value="twitter_123"/>
Owner ID	<input type="text" value="1234567890123456789"/>

Application Actions

Regenerate Consumer Key and Secret Change App Permissions

Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

Create my access token

트위터에 메시지 작성하고 가져오기

- API 키 및 접속 토큰 생성

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[Redacted]
Access Token Secret	[Redacted]
Access Level	Read and write
Owner	[Redacted]
Owner ID	[Redacted]

Token Actions

Regenerate My Access Token and Token Secret Revoke Token Access

트위터에 메시지 작성하고 가져오기

- Tweepy 설치 및 인증
 - <http://docs.tweepy.org/en/v3.6.0/index.html>

```
In: import tweepy
```

```
# 아래는 예이며 본인이 신청해서 생성한 문자열을 각각 복사해 넣습니다.
```

```
consumer_key = 'L9WwxSUZXGt7vIXEiYmtljcHQ'
```

```
consumer_secret = 'bjQrLuottDdpg94VWcG1I3LWXUYZNznzCRE7KvFXR6hzuQcjOh'
```

```
access_token = '503869961253584201-zsblDKpTNdFCp5Q9JWYhi0MqtoTFqg2'
```

```
access_secret = 'XuJ8EYgtim6wye7fYuTrQuY80x1TS2hhPHXAtMzeuPRsb'
```

```
In: auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)
```

```
In: api = tweepy.API(auth)
```

```
API.me()
```

```
API.me().name
```

```
In: print("name:",api.me().name)
```

```
Out: name: my_python_API_test
```

트위터에 메시지 작성하고 가져오기

- 트윗 작성하기

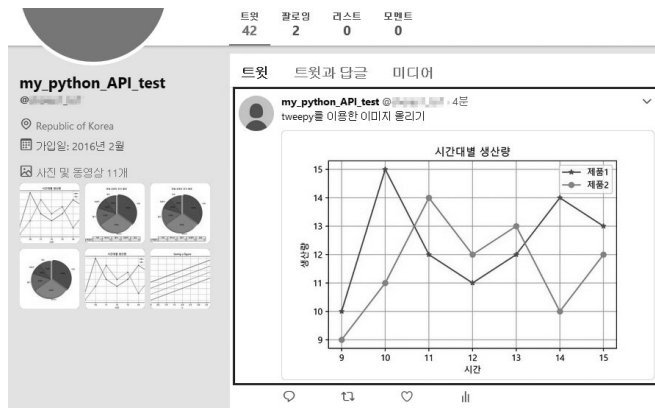
```
API.update_status(status)
```

```
In: tweet_update_status = api.update_status('파이썬에서 Tweepy 라이브러리를 이용한 첫 번째 트윗')
```



```
API.update_with_media(filename [, status])
```

```
In: tweet_media_update_status = api.update_with_media("C:/myPyCode/figures/fig_for_excel1.png",  
'tweepy를 이용한 이미지 올리기')
```



트위터에 메시지 작성하고 가져오기

- 타임라인에서 메시지 가져오기

```
Cursor(api.home_timeline).items([n])
```

```
In: for status in tweepy.Cursor(api.home_timeline).items(2):  
    print("*", status.text)
```

```
Out: * tweepy를 이용한 이미지 올리기 https://t.co/CBcDEU19U8  
    * 파이썬에서 Tweepy 라이브러리를 이용한 첫 번째 트윗
```

```
In: for status in tweepy.Cursor(api.home_timeline).items(2):  
    print("*", status._json['text'])  
    print(" ==> Created at", status._json['created_at'])
```

```
Out: * tweepy를 이용한 이미지 올리기 https://t.co/CBcDEU19U8  
    ==> Created at Sun Jun 17 05:09:03 +0000 2018  
    * 파이썬에서 Tweepy 라이브러리를 이용한 첫 번째 트윗  
    ==> Created at Sun Jun 17 05:08:55 +0000 2018
```


트위터에 메시지 작성하고 가져오기

- 키워드를 지정해 데이터 가져오기
 1. Tweepy의 StreamListener를 상속받아 클래스를 정의
 2. 정의한 클래스를 이용해 객체를 생성
 3. 생성한 객체를 Tweepy의 Stream을 이용해 트위터 Stream API와 연결
 4. Stream의 Filter를 이용해 단어를 지정하고 Stream을 시작
 - Tweepy의 StreamListener를 상속받아 클래스를 정의

```
In: import tweepy
    class MyStreamListener(tweepy.StreamListener):

        def on_status(self, status):
            print(status.text) # 140자까지 출력
```

- 정의한 클래스를 이용해 객체를 생성

```
In: myStreamListener = MyStreamListener()
```

- 생성한 객체를 Tweepy의 Stream을 이용해 트위터 Stream API와 연결

```
In: myStream = tweepy.Stream(auth, myStreamListener)
```

트위터에 메시지 작성하고 가져오기

- Stream의 Filter를 이용해 단어를 지정하고 Stream을 시작

```
In: #myStream.filter(track = ['파이썬', 'python'])
```

```
In: class MyStreamListener(tweepy.StreamListener):  
    def __init__(self):  
        super().__init__()  
        self.tweet_num = 0  
    def on_status(self, status):  
        self.tweet_num = self.tweet_num + 1  
        if(self.tweet_num <= 5):  
            print("***", status.text) # 140자까지 출력  
            return True  
        else:  
            return False
```

트위터에 메시지 작성하고 가져오기

- Stream의 Filter를 이용해 단어를 지정하고 Stream을 시작

```
In: myStreamListener = MyStreamListener()
```

```
    myStream = tweepy.Stream(auth, myStreamListener)
```

```
    myStream.filter(track = ['머신 러닝', 'Machine Learning'])
```

```
Out: *** RT @BenedictEvans: Talk about machine learning now is a bit like talking about SQL in  
the late 70s. "Now we can do pattern recognition/arbi...
```

```
*** RT @TessFerrandez: This is an awesome opportunity to learn and work on real ML problems  
with me and my colleagues - it's super awesome fun...
```

```
*** RT @machinelearnflx: Clustering & Classification With Machine Learning in Python  
https://t.co/9y4cawex5D #machinelearning #ad
```

```
*** Ngene iki yo keno https://t.co/nazLEZKta3
```

```
*** @hospitalvespers machine learning is the WIP i'm reading right now: https://t.co/cnPXLISM5A
```

```
my pal wrote one last n... https://t.co/jE06JEgm7f
```

트위터에 메시지 작성하고 가져오기

- Stream의 Filter를 이용해 단어를 지정하고 Stream을 시작

```
In: import tweepy
```

```
# 키, 토큰, 비밀번호 지정
# consumer_key = 'YOUR-CONSUMER-KEY'
# consumer_secret = 'YOUR-CONSUMER-SECRET'
# access_token = 'YOUR-ACCESS-TOKEN'
# access_secret = 'YOUR-ACCESS-SECRET'
# OAuth 인증 진행
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

# 인증된 auth 변수를 이용해 트위터 API 클래스의 정의
class MyStreamListener2(tweepy.StreamListener):
    def __init__(self, max_num):
        super().__init__()
        self.tweet_num = 0
        self.max_num = max_num

    def on_status(self, status):
        self.tweet_num = self.tweet_num + 1
        file_name = 'C:/myPyCode/data/twitter_stream_test.txt'
        if(self.tweet_num <= self.max_num ):
            with open(file_name, 'a', encoding="utf-8") as f:
```

트위터에 메시지 작성하고 가져오기

- Stream의 Filter를 이용해 단어를 지정하고 Stream을 시작

```
        write_text = "*** " + status.text + "\n"
        f.write(write_text)
    return True
else:
    return False

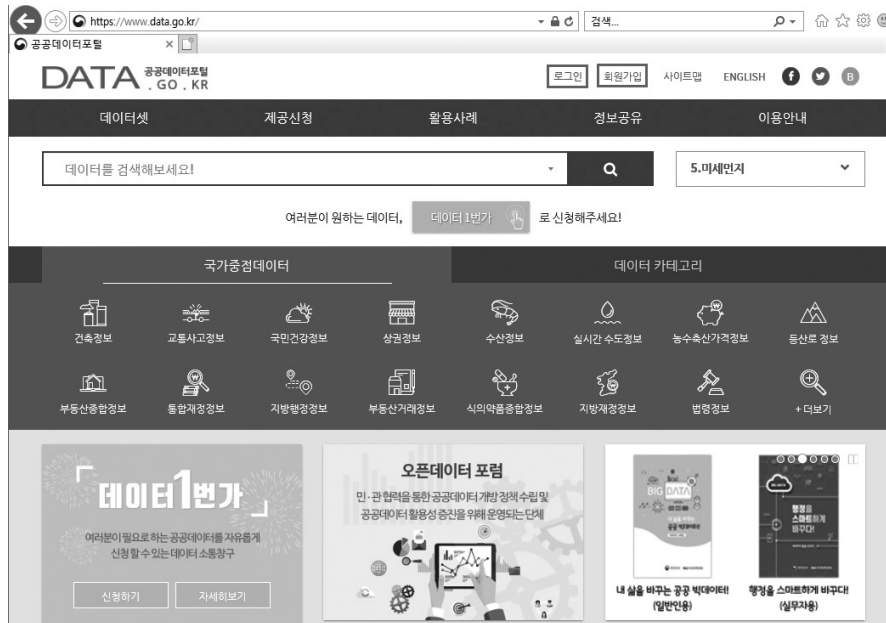
def on_error(self, status):
    print(status) # 오류 메시지 출력
    return False

if __name__ == '__main__':
    myStreamListener = MyStreamListener2(5)
    myStream = tweepy.Stream(auth, myStreamListener)
    myStream.filter(track = ['머신 러닝', 'Machine Learning'])
    print("End of streaming!")
```

Out: End of streaming!

정부의 공공 데이터 가져오기

- 회원 가입 및 서비스 신청



정부의 공공 데이터 가져오기

• 회원 가입 및 서비스 신청

DATA공공데이터포털
.GO . KR

마이페이지로그아웃사이트맵ENGLISH

데이터셋제공신청활용사례정보공유이용안내

도로명주소조회

상세검색

전체(2)파일데이터(0)오픈API(2)표준데이터(0)

오픈API 2건을 찾았습니다.

Q 기관별 검색

인기검색어

국가중점데이터

서비스유형필터(OPENAPI)

제공기관필터

분류체계필터

이용허락범위필터

도로명주소조회

조회수 : 29,883 활용신청건수 : 2,227

수정일 : 2016.06.29 기관 : 과학기술정보통신부 우정사업본부 서비스유형 : REST

우정사업본부에서는 도로명주소 체계로 변경되는 새우편번호(2015.8.1 시행) 및 기존 우편번호 정보를 조회하는 기능의 오픈A...

서울특별시 도로명주소 정보

조회수 : 1,804 바로가기 횟수 : 104

수정일 : 2016.10.27 기관 : 서울특별시 서비스유형 : LINK

서울시 내의 지번주소와 도로명주소, 좌표정보를 제공합니다

DATA공공데이터포털
.GO . KR

마이페이지로그아웃사이트맵ENGLISH

데이터셋제공신청활용사례정보공유이용안내

도로명주소조회서비스

ENGLISH

우정사업본부에서는 도로명주소 체계로 변경되는 새우편번호(2015.8.1 시행) 및 기존 우편번호 정보를 조회하는 기능의 오픈API 서비스를 제공합니다.

활용신청 (바로그기) 건수 : 2,225

서비스 오류가 많을시 오류신고 버튼을 이용해주세요.

XML도로명주소 우편번호 조회서비스

활용신청닫기오류신고

도로명주소 우편번호 조회서비스

End Point	http://openapi.epost.go.kr:80/postal/retrieveNewAddressAreaCdService?_wadt&type=xml 스크럼		
데이터포맷	XML	API 유형	REST
비용부과유무	무료	활용신청 건수	2225
심의유형	개발과정: 자동승인 / 운영과정: 자동승인		
등록일	2015-03-09	수정일	2016-06-29
이용허락범위	이용허락범위 제한 없음		
참고문서	새주소5자리우편번호조회서비스명세서.docx 미리보기		
관리부서명	우정사업정보센터 인터넷우체국팀	관리부서 전화번호	061-0338-2227

정부의 공공 데이터 가져오기

• 회원 가입 및 서비스 신청

인터넷 옵션

일반 보안 개인 정보 내용 연결 프로그램 고급

보안 설정을 보거나 변경할 영역을 선택하십시오.

인터넷 로컬 인터넷 신뢰할 수 있는 사이트 제한된 사이트

신뢰할 수 있는 사이트

이 영역에는 사용자 컴퓨터나 파일을 손상시키지 않을 것으로 신뢰되는 웹 사이트가 포함됩니다.

신뢰할 수 있는 사이트

이 영역에 웹 사이트를 추가하거나 제거할 수 있습니다. 추가한 모든 웹 사이트에는 이 영역의 보안 설정이 적용됩니다.

영역에 웹 사이트 추가(D):

https://*.data.go.kr

추가(A)

웹 사이트(W):

제거(R)

이 영역에 있는 모든 사이트에 대해 서버 검증(https) 필요(S)

닫기(C)

DATA 공공데이터포털 .GO .KR

마이페이지 로그아웃 사이트맵 ENGLISH

데이터셋 개발계정 신청

마이페이지

오픈API

개발계정 신청

기본정보

서비스명	도로명주소 우편번호 조회서비스	서비스 유형	REST
심의여부	자동승인	신청유형	개발계정 활용신청
처리상태	신청	활용기간	승인일로부터 24개월 간 활용가능

시스템유형 선택

일반 서버 구축

시스템 유형

* 일반 : OpenAPI 서비스를 호출하여 응답받은 결과값을 서버에 저장하지 않고 사용할 경우 (서버 미구축)
* 서버 구축 : OpenAPI 서비스를 호출하여 응답받은 결과값을 서버에 저장하거나 DB화 하여 사용할 경우

활용정보

* 활용목적

웹 사이트 개발 앱개발 (모바일,솔루션등) 기타 참고자료 연구(논문 등)

첨부파일

*파일 첨부시 팝업창단 기능이 해제되어야 합니다.

첨가 삭제 한 개의 파일만 첨부 할 수 있습니다.

상세기능정보 필수 입력 정보입니다.

*자동승인 상세기능은 신청과 동시에 활용 가능합니다.

상세기능	설명	일일 트래픽
<input checked="" type="checkbox"/>	새우편번호 도로명주소 조 지번주소를 페이지당 출력개수와 출력될 페이지번호 기준으로 도로명 주소 값으로 반환해주는 도로명주소 조회서비스	10000

라이선스표시

이용허락범위

이용허락범위 제한 없음

(사유:) 동의합니다

신청 취소

정부의 공공 데이터 가져오기

• 회원 가입 및 서비스 신청

활용정보

활용목적

☐ 웹 사이트 개발
☒ 앱개발 (모바일, 솔루션 등)
☐ 기타
☐ 참고자료
☐ 연구(논문 등)

첨부파일

첨부파일

첨부파일

상세기능정보

상세기능정보

상세기능정보

라이선스표시

이용허락범위

이용허락범위 제한 없음

System Message

개발계정 신청이 완료되었습니다.

'마이페이지>OPEN API>개발계정'에서 신청내역을 확인하실 수 있습니다.

확인

신뢰

취소

마이페이지

개발계정 상세보기

기본정보

서비스명

도로명주소 우편번호 조회서비스

서비스 유형

REST

알림트래픽

0

평균응답속도(초)

0

실의여부

자동승인

신청유형

개발계정 | 활용신청

처리상태

승인

유효기간

2018-03-31 ~ 2020-03-31

서비스정보

일반 인증키 (UTF-8)

http://openapi.epost.go.kr:80/postal/retrieveNewAddressAreaCdService?_wadi&type=xml

데이터포맷

XML

참고문서

새주소5자리우편번호조회서비스명세서.docx

상세기능정보

NO

상세기능

설명

활용제한 여부

일일 트래픽

심의결과

개발기타정보

1

새우편번호 도로명주소 조회

지번주소를 폐지된 주소로 출력하고, 출력된 폐지된 주소로 도로명주소로 변환해주는 도로명주소 조회서비스

-

10000

승인

개발기타정보

마이페이지

오픈API

- 개발계정
- 활용현황
- 운영계정
- 인증키 발급현황

DATA

나의 문의

나의 관심

회원정보

> 마이페이지 > OPEN API > 개발계정

개발계정

활용자가 개발계정을 신청중인 단계	활용자의 개발계정신청이 승인되어 활용중인 단계	활용자가 중지신청하여 운영이 중지된 단계
<div>신청</div> <div>0건</div>	<div>활용</div> <div>6건</div>	<div>중지</div> <div>2건</div>

상세검색

총 8건

[승인] 도로명주소 우편번호 조회서비스	신청일:2018-03-31 [활용신청] 만료예정 :2020-03-31
서비스유형 :REST	분류 :교통및물류 > 물류등기타
제공기관 :과학기술정보통신부	우정사업본부

정부의 공공 데이터 가져오기

- 주소 및 우편번호 가져오기

- 요청 주소

- <http://openapi.epost.go.kr/postal/retrieveNewAdressAreaCdService/retrieveNewAdressAreaCdService/getNewAddressListAreaCd>

- 도로명 주소 조회 서비스의 요청 변수

항목명(영문)	항목명(국문)	항목 크기	항목 구분	샘플 데이터	항목 설명
ServiceKey	서비스 키	255	필수	SERVICE_KEY	서비스 인증
searchSe	검색구분	1	필수	dong	dong:동(읍/면)명, road :도로명 [default], post:우편번호
srchwrд	검색어	200	필수	주월동 408-1	검색어
countPerPage	페이지당 출력 개수	10	옵션	10	페이지당 출력될 개수를 지정
currentPage	출력될 페이지 번호	10	옵션	1	출력될 페이지 번호

- 사용 예

- <http://openapi.epost.go.kr/postal/retrieveNewAdressAreaCdService/retrieveNewAdressAreaCdService/getNewAddressListAreaCd?ServiceKey=서비스키&searchSe=road&srchwrд=세종로 17>

- 출력 결과

- XML 형식으로 우편번호(5자리), 도로명 주소, 지번 주소 등이 출력

정부의 공공 데이터 가져오기

- 주소 및 우편번호 가져오기

```
In: import requests
```

```
# API_KEY = 'YOUR-API-KEY' # 자신의 인증키를 복사해서 입력합니다.
```

```
API_KEY = "et5piq3pfpqLEWPpCbvtSQ%2Bertert%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoewRV %2Fovmrk3dq%3D%3D"
```

```
API_KEY_decode = requests.utils.unquote(API_KEY)
```

```
API_KEY_decode
```

Out:

```
'et5piq3pfpqLEWPpCbvtSQ+ertertg+x3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw+9rkvoewRV/ovmrk3dq=='
```

정부의 공공 데이터 가져오기

- 주소 및 우편번호 가져오기

```
In: req_url = "http://openapi.epost.go.kr/postal/retrieveNewAdressAreaCdService/  
retrieveNewAdressAreaCdService/getNewAddressListAreaCd"
```

```
search_Se = "road"  
srch_wrd = "반포대로 201"
```

```
req_parameter = {"ServiceKey":API_KEY_decode, "searchSe":search_Se, "srchwr":srch_wrd}
```

```
r = requests.get(req_url, params = req_parameter)  
xml_data = r.text  
print(xml_data)
```

```
Out: <?xml version="1.0" encoding="UTF-8" standalone="yes"?><NewAddressListResponse><  
cmmMsgHeader><requestMsgId></requestMsgId><responseMsgId></responseMsgId><respo  
nseTime>20180416:23010001</responseTime><successYN>Y</successYN><returnCode>00</  
returnCode><errMsg></errMsg><totalCount>1</totalCount><countPerPage>10</  
countPerPage><totalPage>1</totalPage><currentPage></currentPage></cmmMsgHeader><newA  
ddressListAreaCd><zipNo>06579</zipNo><InmAdres>서울특별시 서초구 반포대로 201 (반포동,  
국립중앙도서관)</InmAdres><rnAdres>서울특별시 서초구 반포동 산60-1 국립중앙도서관  
</rnAdres></newAddressListAreaCd></NewAddressListResponse>
```

정부의 공공 데이터 가져오기

- 주소 및 우편번호 가져오기

```
In: import xmltodict
    dict_data = xmltodict.parse(xml_data)
    dict_data
Out: OrderedDict([('NewAddressListResponse',
    OrderedDict([('cmmMsgHeader',
        OrderedDict([('requestMsgId', None),
            ('responseMsgId', None),
            ('responseTime',
                '20180408:191742825'),
            ('successYN', 'Y'),
            ('returnCode', '00'),
            ('errMsg', None),
            ('totalCount', '1'),
            ('countPerPage', '10'),
            ('totalPage', '1'),
            ('currentPage', None)]))),
    ('newAddressListAreaCd',
    OrderedDict([('zipNo', '06579'),
        ('InmAdres',
            '서울특별시 서초구 반포대로 201 (반포동,
            국립중앙도서관)'),
        ('rnAdres',
            '서울특별시 서초구 반포동 산60-1
            국립중앙도서관')]))]))])
```

정부의 공공 데이터 가져오기

- 주소 및 우편번호 가져오기

```
In: adress_list = dict_data['NewAddressListResponse']['newAddressListAreaCd']
```

```
print("[입력한 도로명 주소]", srch_wrd)
print("[응답 데이터에서 추출한 결과]")
print("- 우편번호:", adress_list['zipNo'])
print("- 도로명 주소:", adress_list['lnmAdres'])
print("- 지번 주소:", adress_list['rnAdres'])
```

Out: [입력한 도로명 주소] 반포대로 201

[응답 데이터에서 추출한 결과]

- 우편번호: 06579

- 도로명 주소: 서울특별시 서초구 반포대로 201 (반포동, 국립중앙도서관)

- 지번 주소: 서울특별시 서초구 반포동 산60-1 국립중앙도서관

정부의 공공 데이터 가져오기

- 날씨 정보 가져오기
 - 날씨 정보를 위한 서비스 신청

DATA 공공데이터포털 .GO. KR 마이페이지 로그아웃 사이트맵 ENGLISH

데이터셋 제공신청 활용사례 정보공유 이용안내

☛ / 데이터셋 / 통합검색

동네예보정보조회

전체(1) 파일데이터(0) **오픈API(1)** 표준데이터(0)

오픈API 1건을 찾았습니다.

Q 기관별 검색

인기검색어

- 1 미세먼지
- 2 지하철
- 3 도서관
- 4 범죄
- 5 물가 조사 정보

국가중점데이터

서비스유형별(OPENAPI)

🔍 오픈API [1건]

지역도 ▾ 날 짜 계 목 조회수 활용신청

동네예보정보조회서비스 조회수 : 17,862 활용신청건수 : 7,049

수정일 : 2018.01.25 기관 : 기상청 서비스유형 : REST

설명, 초단기예보, 동네예보 정보를 조회하는 서비스입니다.

XML

DATA 공공데이터포털 .GO. KR 마이페이지 로그아웃 사이트맵 ENGLISH

데이터셋 제공신청 활용사례 정보공유 이용안내

☛ / 데이터셋 / 오픈API

동네예보정보조회서비스 ENGLISH

설명, 초단기예보, 동네예보 정보를 조회하는 서비스입니다.

활용신청 (비로그인) 건수 : 7,049

※ 서비스 오류가 있을시 오류신고 버튼을 이용해주세요.

XML (신)동네예보정보조회서비스

활용신청 닫기 오류신고

(신)동네예보정보조회서비스			
End Point	http://newsky2.kma.go.kr/service/SecndSrtpdFrstInfoService2 스크랩		
데이터포맷	XML	API 유형	REST
비용부과유무	무료	활용신청 건수	7049
심의유형	개발계정: 자동승인 / 운영계정: 자동승인		
등록일	2015-12-01	수정일	2018-01-25
이용허락범위	이용허락범위 제한 없음		
참고문서	(신)동네예보정보조회서비스_1.5.zip 마리보기		
관리부서명	정보통신기술과	관리부서 전화번호	

정부의 공공 데이터 가져오기

- 날씨 정보를 위한 서비스 신청

▶ 상세기능정보 필수 입력 정보입니다.		*자동승인 상세기능은 신청과 동시에 활용 가능합니다.	
<input type="checkbox"/>	상세기능	설명	일일 트래픽
<input checked="" type="checkbox"/>	예보버전조회	수정된 예보 버전을 파악하기 위해 예보버전을 조회하는 기능	1000
<input checked="" type="checkbox"/>	동네예보조회	동네예보 정보를 조회하기 위해 예보일자, 예보시간, 예보지점X좌표, 예보지점Y좌표의 조회 조건으로 예측일자, 예측시간, 자료구분분야, 예보값, 예보일자, 예보시간, 예보지점X좌표, 예보지점Y좌표의 정보를 조회하는 기능	1000
<input checked="" type="checkbox"/>	초단기예보조회	초단기예보 정보를 조회하기 위해 예보일자, 예보시간, 예보지점X좌표, 예보지점Y좌표의 조회 조건으로 자료구분코드, 예보값, 예보일자, 예보시간, 예보지점X좌표, 예보지점Y좌표의 정보를 조회하는 기능	1000
<input checked="" type="checkbox"/>	초단기실황조회	초단기실황정보를 조회하기 위해 예보일자, 예보시간, 예보지점 X좌표, 예보지점 Y좌표의 조회 조건으로 자료구분코드, 실황 값, 예보일자, 예보시간, 예보지점X좌표, 예보지점Y좌표의 정보를 조회하는 기능	1000

마이페이지
▶ 마이페이지 > OPEN API > 개발계정 상세보기


개발계정 상세보기

▶ 운영계정 신청
▶ 연장 신청
▶ 변경 신청
▶ 중지 신청
▶ 일반 인증키 재발급
▶ 목록

> 기본정보

서비스명	(신)동네예보정보조회서비스 상세설명				
서비스 유형	REST	알트랙	0	평균응답속도(초)	0
접근여부	자동승인	신청유형	개발계정 변경신청	처리상태	승인
활용기간	2018-04-01 ~ 2020-04-01				

> 서비스정보

일반 인증키 (UTF-8)	<div style="background-color: #f0f0f0; padding: 5px;">[Redacted Key]</div> 복사				
End Point	http://newsky2.kma.go.kr/service/SecndSrtpdFrcstInfoService2				
데이터포맷	XML				
참고문서	(신)동네예보정보조회서비스_1.5.zip				

정부의 공공 데이터 가져오기

– 날씨 실황 조회

- 요청 주소(Request URL)

- <http://newsky2.kma.go.kr/service/SecndSrtpdFrcstInfoService2/ForecastGrib>

- 요청 변수(Request Parameters)

항목명(영문)	항목명(국문)	항목 크기	항목 구분	샘플 데이터	항목 설명
ServiceKey	서비스 키	255	필수	SERVICE_KEY	서비스 인증
base_date	발표 일자	8	필수	20180402	2018년 4월 2일 발표
base_time	발표 시각	4	필수	1500	15시 발표(정시 단위)
nx	예보지점 X 좌표	2	필수	60	예보지점의 X 좌표값
ny	예보지점 Y 좌표	2	필수	127	예보지점의 Y 좌표값
numOfRows	한 페이지에 포함된 결과 수	2	옵션	10	한 페이지에 포함된 결과 수
pageNo	페이지 번호	5	옵션	1	페이지 번호
_type	타입		옵션	xml 혹은 json	xml(기본값), json

- 사용 예제

- http://newsky2.kma.go.kr/service/SecndSrtpdFrcstInfoService2/ForecastGrib?ServiceKey=서비스키&base_date=20180402&base_time=1500&nx=60&ny=127&pageNo=1&numOfRows=1&_type=json

정부의 공공 데이터 가져오기

– 날씨 상황 조회

- 응답 결과

- XML 혹은 JSON 형식의 예보 지점의 날씨 데이터

```
In: import requests
```

```
# API_KEY = 'YOUR-API-KEY' # 자신의 인증키를 복사해서 입력합니다.
```

```
API_KEY = "et5piq3pfpqLEWPpCbvtSQ%2Bertertg%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoewRV %2Fovmrk3dq%3D%3D"
```

```
API_KEY_decode = requests.utils.unquote(API_KEY)
```

```
API_KEY_decode
```

Out:

```
'et5piq3pfpqLEWPpCbvtSQ+ertertg+x3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw+9rkvoewRV/ovmrk3dq=='
```

정부의 공공 데이터 가져오기

– 날씨 실황 조회

```
In: import json
import datetime

# [날짜 및 시간 설정]
now = datetime.datetime.now() # 현재 날짜 및 시간 반환

# baseDate에 날짜를 입력하기 위해 날짜를 출력 형식을 지정해 변수에 할당
date = "{:%Y%m%d}".format(now)

# baseTime에 시간(정시)을 입력하기 위해 출력 형식을 지정해 시간만 변수에 할당
time = "{:%H00}".format(now)

# 현재 분이 30분 이전이면 이전 시간(정시)을 설정
if (now.minute >= 30):
    time = "{0}00".format(now.hour)
else:
    time = "{0}00".format(now.hour-1)

# [요청 주소 및 요청 변수 지정]
req_url = "http://newsky2.kma.go.kr/service/SecndSrtpdFrcstInfoService2/ForecastGrib"

baseDate = date # 발표 일자 지정
baseTime = time # 발표 시간 지정(정시로 지정)
```

정부의 공공 데이터 가져오기

– 날씨 상황 조회

```
nx_val = 60 # 예보지점 X 좌표(서울시 종로구 사직동)
ny_val = 127 # 예보지점 Y 좌표(서울시 종로구 사직동)

num_of_rows = 6 # 한 페이지에 포함된 결과 수
page_no = 1 # 페이지 번호

output_type = "json" # 응답 데이터 형식 지정
req_parameter = {"ServiceKey":API_KEY_decode,
                 "nx":nx_val, "ny": ny_val,
                 "base_date":baseDate, "base_time":baseTime,
                 "pageNo":page_no, "numOfRows":num_of_rows,
                 "_type":output_type}

# [데이터 요청]
r = requests.get(req_url, params = req_parameter)

# [JSON 형태로 응답받은 데이터를 딕셔너리 데이터로 변환]
dict_data = r.json()
dict_data
```

정부의 공공 데이터 가져오기

– 날씨 상황 조회

```
Out: {'response': {'body': {'items': {'item': [{'baseDate': 20180618,
    'baseTime': 2100,
    'category': 'LGT',
    'nx': 60,
    'ny': 127,
    'obsrValue': 0},
    {'baseDate': 20180618,
    'baseTime': 2100,
    'category': 'PTY',
    'nx': 60,
    'ny': 127,
    'obsrValue': 0},
    {'baseDate': 20180618,
    'baseTime': 2100,
    'category': 'REH',
    'nx': 60,
    'ny': 127,
    'obsrValue': 59},
    {'baseDate': 20180618,
    'baseTime': 2100,
    'category': 'RN1',
    'nx': 60,
    'ny': 127,
    'obsrValue': 0},
```

정부의 공공 데이터 가져오기

– 날씨 상황 조회

```
{'baseDate': 20180618,  
  'baseTime': 2100,  
  'category': 'SKY',  
  'nx': 60,  
  'ny': 127,  
  'obsrValue': 1},  
{'baseDate': 20180618,  
  'baseTime': 2100,  
  'category': 'T1H',  
  'nx': 60,  
  'ny': 127,  
  'obsrValue': 23.9}}},  
'numOfRows': 6,  
'pageNo': 1,  
'totalCount': 10},  
'header': {'resultCode': '0000', 'resultMsg': 'OK'}}}
```

정부의 공공 데이터 가져오기

– 날씨 상황 조회

In: # [딕셔너리 데이터를 분석해서 원하는 값 추출]

```
weather_items = dict_data['response']['body']['items']['item']
```

```
sky_cond = ["맑음", "구름 조금", "구름 많음", "흐림"]
```

```
rain_type = ["없음", "비", "진눈개비", "눈"]
```

```
print("[ 발표 날짜: {} ]".format(weather_items[0]['baseDate']))
```

```
print("[ 발표 시간: {} ]".format(weather_items[0]['baseTime']))
```

```
for k in range(len(weather_items)):
```

```
    weather_item = weather_items[k]
```

```
    obsrValue = weather_item['obsrValue']
```

```
    if(weather_item['category'] == 'T1H'):
```

```
        print("* 기온: {} 도".format(obsrValue))
```

```
    elif(weather_item['category'] == 'REH'):
```

```
        print("* 습도: {} 퍼센트".format(obsrValue))
```

```
    elif(weather_item['category'] == 'SKY'):
```

```
        print("* 하늘: {}".format(sky_cond[obsrValue-1]))
```

```
    elif(weather_item['category'] == 'PTY'):
```

```
        print("* 강수: {}".format(rain_type[obsrValue]))
```

정부의 공공 데이터 가져오기

– 날씨 실황 조회

Out: [발표 날짜: 20180618]

[발표 시간: 2100]

* 강수: 없음

* 습도: 59 퍼센트

* 하늘: 맑음

* 기온: 23.9 도

정부의 공공 데이터 가져오기

– 일기 예보 조회

```
In: import json
import datetime

# [날짜 및 시간 설정]
now = datetime.datetime.now() # 현재 날짜 및 시간 반환

# baseDate에 날짜를 입력하기 위해 날짜를 출력 형식을 지정해 변수에 할당
date = "{:%Y%m%d}".format(now)

# baseTime에 시간(정시)을 입력하기 위해 출력 형식을 지정해 시간만 변수에 할당
time = "{:%H00}".format(now)

# 현재 분이 30분 이전이면 이전 시간(정시)을 설정
if (now.minute >= 30):
    time = "{0}00".format(now.hour)
else:
    time = "{0}00".format(now.hour-1)

# [요청 주소 및 요청 변수 지정]
req_url = "http://newsky2.kma.go.kr/service/SecndSrtpdFrcstInfoService2/ForecastTimeData"

baseDate = date # 발표 일자 지정
baseTime = time # time # 발표 시간 지정(정시로 지정)
```

정부의 공공 데이터 가져오기

– 일기 예보 조회

```
nx_val = 60 # 예보지점 X 좌표(서울시 종로구 사직동)
ny_val = 127 # 예보지점 Y 좌표(서울시 종로구 사직동)

num_of_rows = 30 # 한 페이지에 포함된 결과 수
page_no = 1 # 페이지 번호

output_type = "json" # 응답 데이터 형식 지정

req_parameter = {"ServiceKey":API_KEY_decode,
                 "nx":nx_val, "ny": ny_val,
                 "base_date":baseDate, "base_time":baseTime,
                 "pageNo":page_no, "numOfRows":num_of_rows,
                 "_type":output_type}

# [데이터 요청]
r = requests.get(req_url, params = req_parameter)

# [JSON 형태로 응답받은 데이터를 딕셔너리 데이터로 변환]
dict_data = r.json()

# [딕셔너리 데이터를 분석해서 원하는 값 추출]
weather_items = dict_data['response']['body']['items']['item']
```

정부의 공공 데이터 가져오기

– 일기 예보 조회

```
sky_cond = ["맑음", "구름 조금", "구름 많음", "흐림"]
rain_type = ["없음", "비", "진눈개비", "눈"]

print("[ 발표 날짜: {} ]".format(weather_items[0]['baseDate']))
print("[ 발표 시간: {} ]".format(weather_items[0]['baseTime']))

print("[ 초단기 일기 예보 ]")

for k in range(len(weather_items)):
    weather_item = weather_items[k]

    fcstTime = weather_item['fcstTime']
    fcstValue = weather_item['fcstValue']

    if(weather_item['category'] == 'T1H'):
        print("* 시간: {0}, 기온: {1} 도".format(fcstTime, fcstValue))
    elif(weather_item['category'] == 'REH'):
        print("* 시간: {0}, 습도: {1} 퍼센트".format(fcstTime, fcstValue))
    elif(weather_item['category'] == 'SKY'):
        print("* 시간: {0}, 하늘: {1}".format(fcstTime, sky_cond[fcstValue-1]))
    elif(weather_item['category'] == 'PTY'):
        print("* 시간: {0}, 강수: {1}".format(fcstTime, rain_type[fcstValue]))
```

정부의 공공 데이터 가져오기

– 일기 예보 조회

Out: [발표 날짜: 20180618]

[발표 시간: 2130]

[초단기 일기 예보]

- * 시간: 2200, 강수: 없음
- * 시간: 2300, 강수: 없음
- * 시간: 0000, 강수: 없음
- * 시간: 2200, 하늘: 맑음
- * 시간: 2300, 하늘: 맑음
- * 시간: 0000, 하늘: 맑음
- * 시간: 2200, 기온: 24.5 도
- * 시간: 2300, 기온: 24.3 도
- * 시간: 0000, 기온: 24 도
- * 시간: 2200, 습도: 58 퍼센트
- * 시간: 2300, 습도: 59 퍼센트
- * 시간: 0000, 습도: 60 퍼센트

정부의 공공 데이터 가져오기

- 대기 오염 정보 가져오기
 - 대기 오염 정보를 위한 서비스 신청

한국환경공단

상세검색

전체(25)

파일데이터(18)

오픈API(7)

표준데이터(0)

오픈API 7건을 찾았습니다.

Q 관련별 검색

오픈API [7건]

검색도

날 짜

제 목

조회수

활용신청

한국환경공단 대기오염정보 조회 서비스

조회수 : 5,126 활용신청건수 : 6,343

수정일 : 2018.01.16 기관 : 한국환경공단 서비스유형 : REST

각 측정소별 대기오염정보를 조회하기 위한 서비스로 시간별, 시도별 대기오염 정보와 통합대기환경지수 나쁨 이상 측정소 내...

XML

한국환경공단 천기차충전소 운영현황

조회수 : 56 활용신청건수 : 303

수정일 : 2017.09.15 기관 : 한국환경공단 서비스유형 : REST

전국 급속충전소 위치 및 상태정보 등 제공을 통해 전기자동차 충전인프라 소개

XML

한국환경공단 측정소정보 조회 서비스

조회수 : 836 활용신청건수 : 1,459

수정일 : 2018.01.16 기관 : 한국환경공단 서비스유형 : REST

대기질 측정소 정보를 조회하기 위한 서비스로 TM 좌표기반의 가까운 측정소 및 측정소 목록과 측정소의 정보를 조회할 수 있다.

XML

환경기상

환경기상

환경기상

인기검색어

1 미세먼지

2 지하철

3 범죄

4 날씨

5 대구

국가중점데이터

서비스유형필터(OPENAPI)

제공기관필터

분류체계필터

이용허락범위필터

태그

확장자

상세기능정보 필수 입력 정보입니다. *자동승인 상세기능은 신청과 동시에 활용 가능합니다.

<input type="checkbox"/>	상세기능	설명	일일 트래픽
<input checked="" type="checkbox"/>	시군구별 실시간 평균정보 조회	시도의 각 시군구별 측정소목록의 일반 항목에 대한 시간대별 평균농도를 제공하는 시군구별 실시간 평균정보 조회	500
<input checked="" type="checkbox"/>	시도별 실시간 평균정보 조회	시도별 측정소목록에 대한 일반 항목의 시간 및 일평균 자료 및 지역 평균 정보를 제공하는 시도별 실시간 평균정보 조회	500
<input checked="" type="checkbox"/>	대기질 예보통보 조회	통보코드와 통보시간으로 예보정보와 발생 원인 정보를 조회하는 대기질 (미세먼지/오존) 예보통보 조회	500
<input checked="" type="checkbox"/>	시도별 실시간 측정정보 조회	시도명을 검색조건으로 하여 시도별 측정소목록에 대한 일반 항목과 CAI 최종 실시간 측정값과 지수 정보 조회 기능을 제공하는 시도별 실시간 측정정보 조회	500
<input checked="" type="checkbox"/>	통합대기환경지수 나쁨 이상 측정소 목록조회	통합대기환경지수가 나쁨 등급 이상인 측정소명과 주소 목록 정보를 제공하는 통합대기환경지수 나쁨 이상 측정소 목록조회	500
<input checked="" type="checkbox"/>	측정소별 실시간 측정정보 조회	측정소명과 측정데이터 기간(일, 한달, 3개월)으로 해당 측정소의 일반항목 측정정보를 제공하는 측정소별 실시간 측정정보조회	500

상세기능정보 필수 입력 정보입니다. *자동승인 상세기능은 신청과 동시에 활용 가능합니다.

<input type="checkbox"/>	상세기능	설명	일일 트래픽
<input checked="" type="checkbox"/>	TM 기준좌표 조회	TM 좌표를 알 수 없는 사용자를 위해 읍면동 이름으로 검색하여 TM기준 좌표 내역을 조회하는 기능 제공	500
<input checked="" type="checkbox"/>	측정소 목록 조회	측정소 주소 또는 측정소 명칭으로 검색하여 측정소 목록 또는 단 건의 측정소 상세 정보 조회 가능 제공	500
<input checked="" type="checkbox"/>	근접측정소 목록 조회	TM 좌표를 입력하여 입력된 좌표 주변 측정소 정보와 입력 좌표와의 거리 조회 가능 제공	500

정부의 공공 데이터 가져오기

- 대기 오염 정보 가져오기

- 근접 측정소 목록 조회하기: airkorea_openapi_guide-v1_6_1.docx 참조

- 요청 주소(Request URL)

- <http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfoInquireSvc/getTMStdCrCnt>

- 요청 변수(Request Parameters)

항목명(영문)	항목명(국문)	항목 크기	항목 구분	샘플 데이터	항목 설명
ServiceKey	서비스 키	255	필수	SERVICE_KEY	서비스 인증
umdName	읍면동명	60	필수	혜화동	읍면동명
numOfRows	한 페이지에 포함된 결과 수	2	옵션	10	한 페이지에 포함된 결과 수
pageNo	페이지 번호	5	옵션	1	페이지 번호
_returnType	리턴 타입		옵션	json	지정하지 않으면 xml

- 사용 예제

- http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfoInquireSvc/getTMStdCrCnt?umdName=논현동&pageNo=1&numOfRows=10&ServiceKey=서비스키&_returnType=json

- 응답 결과

- XML 혹은 JSON 형식

정부의 공공 데이터 가져오기

– 근접 측정소 목록 조회하기

In: import requests

```
# API_KEY = 'YOUR-API-KEY' # 자신의 인증키를 복사해서 입력합니다.
```

```
API_KEY = "et5piq3pfpqLEWPpCbvtSQ%2Bertert%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoewRV%2Fovmrk3dq%3D%3D"
```

```
API_KEY_decode = requests.utils.unquote(API_KEY)
```

```
API_KEY_decode
```

Out:

```
'et5piq3pfpqLEWPpCbvtSQ+ertertg+x3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw+9rkvoewRV/ovmrk3dq%3D%3D'
```

```
In: req_url = "http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfolnquireSvc/getTMStdCrCnt"
    umd_name = "논현동" #읍, 면, 동 지정
```

```
num_of_rows = 10 # 한 페이지에 포함된 결과 수
```

```
page_no = 1 # 페이지 번호
```

```
output_type = "json"
```

```
req_parameter = {"ServiceKey":API_KEY_decode, "umdName":umd_name,
                  "pageNo":page_no, "numOfRows":num_of_rows,
                  "_returnType":output_type}
```

```
dict_data = requests.get(req_url, params = req_parameter).json()
```

```
dict_data['totalCount'] # 전체 결과의 개수
```

Out: 2

정부의 공공 데이터 가져오기

– 근접 측정소 목록 조회하기

```
In: print("[입력한 읍/면/동명]", umd_name)
    print("[TM 기준 좌표 조회 결과]")

    for k in range(dict_data['totalCount']):
        sido = dict_data['list'][k]['sidoName']
        sgg = dict_data['list'][k]['sggName']
        umd = dict_data['list'][k]['umdName']
        tmX = dict_data['list'][k]['tmX']
        tmY = dict_data['list'][k]['tmY']

    print("- 위치: {0} {1} {2}".format(sido, sgg, umd))
    print("- k = {0}, TM 좌표(X, Y): {1}, {2}Wn".format(k, tmX, tmY))
Out: [입력한 읍/면/동명] 논현동
     [TM 기준 좌표 조회 결과]
     - 위치: 서울특별시 강남구 논현동
     - k = 0, TM 좌표(X, Y): 202733.974301, 445717.50469
     - 위치: 인천광역시 남동구 논현동
     - k = 1, TM 좌표(X, Y): 175850.136025, 434153.586394
```

```
In: k = 0 # 원하는 위치 선택 (여기서는 첫 번째 위치)
    TM_X = dict_data['list'][k]['tmX'] # TM X 좌표
    TM_Y = dict_data['list'][k]['tmY'] # TM Y 좌표
    print("TM 좌표(X, Y): {0}, {1}".format(TM_X, TM_Y))
Out: TM 좌표(X, Y): 202733.974301, 445717.50469
```


정부의 공공 데이터 가져오기

– 근접 측정소 목록 조회하기

- 요청 주소(Request URL)

- <http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfolnquireSvc/getNearbyMsrstnList>

- 요청 변수(Request Parameters)

항목명(영문)	항목명(국문)	항목 크기	항목 구분	샘플 데이터	항목 설명
ServiceKey	서비스 키	255	필수	SERVICE_KEY	서비스 인증
tmX	TM_X 좌표	16.6	필수	244148.546388	TM 측정방식 X좌표
tmY	TM_Y 좌표	16.6	필수	412423.75772	TM 측정방식 Y좌표
numOfRows	한 페이지에 포함된 결과 수	2	옵션	10	한 페이지에 포함된 결과 수
pageNo	페이지 번호	5	옵션	1	페이지 번호
_returnType	리턴 타입		옵션	json	지정하지 않으면 xml

- 사용 예제

- http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfolnquireSvc/getNearbyMsrstnList?ServiceKey={apikey}&tmX=x&tmY=y&pageNo=1&numOfRows=10&_returnType=json

- 응답 결과

- XML 혹은 JSON 형식

정부의 공공 데이터 가져오기

– 근접 측정소 목록 조회하기

```
In: req_url = "http://openapi.airkorea.or.kr/openapi/services/rest/MsrstnInfoInquireSvc/
    getNearbyMsrstnList"
```

```
x_value = TM_X # TM 측정방식 X좌표
y_value = TM_Y # TM 측정방식 Y좌표
```

```
num_of_rows = 10 # 한 페이지에 포함된 결과 수
page_no = 1 # 페이지 번호
```

```
output_type = "json"
req_parameter = {"ServiceKey":API_KEY_decode,
                 "tmX":x_value, "tmY":y_value,
                 "pageNo":page_no, "numOfRows":num_of_rows,
                 "_returnType":output_type}
dict_data = requests.get(req_url, params = req_parameter).json()
```

```
print("해당 지역 근처에 있는 측정소의 개수:", dict_data['totalCount'])
```

```
Out: 해당 지역 근처에 있는 측정소의 개수: 3
```

정부의 공공 데이터 가져오기

– 근접 측정소 목록 조회하기

```
In: print("[측정소 정보]")
    for k in range(dict_data['totalCount']):

        stationName = dict_data['list'][k]['stationName']
        distance = dict_data['list'][k]['tm']
        addr = dict_data['list'][k]['addr']

        print("- 측정소 이름:{0}, 거리:{1}[km]".format(stationName, distance))
        print("- 측정소 주소:{0} \n".format(addr))
```

Out: [측정소 정보]

- 측정소 이름:도산대로, 거리:1.1[km]
- 측정소 주소:서울 강남구 도산대로 104신사역2번출구 앞
- 측정소 이름:강남구, 거리:1.7[km]
- 측정소 주소:서울 강남구 학동로 426강남구청 별관 1동
- 측정소 이름:강변북로, 거리:2.9[km]
- 측정소 주소:서울 성동구 강변북로 257한강사업본부 옆

정부의 공공 데이터 가져오기

- 측정 정보 가져오기

- 요청 주소(Request URL)
 - <http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInquireSvc/getMsrstnAcctoRltmMesureDnsty>
- 요청 변수(Request Parameters)

항목명(영문)	항목명(국문)	항목 크기	항목 구분	샘플 데이터	항목 설명
ServiceKey	서비스 키	255	필수	SERVICE_KEY	서비스 인증
stationName	측정소명	30	필수	종로구	측정소 이름
dataTerm	데이터 기간	10	필수	DAILY	요청 데이터 기간(1일: DAILY, 1개월: MONTH, 3개월: 3MONTH)
numOfRows	한 페이지에 포함된 결과 수	2	옵션	10	한 페이지에 포함된 결과 수
pageNo	페이지 번호	5	옵션	1	페이지 번호
ver	오퍼레이션 버전	4	옵션	1.0	1.0, 1.1, 1.2, 1.3 중 선택
_returnType	리턴 타입		옵션	json	지정하지 않으면 xml

- 사용 예제
 - http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInquireSvc/getMsrstnAcctoRltmMesureDnsty?stationName=종로구&dataTerm=month&pageNo=1&numOfRows=10&ServiceKey=서비스키&ver=1.3&_returnType=json
- 응답 결과: XML 혹은 JSON 형식

정부의 공공 데이터 가져오기

- 측정 정보 가져오기

```
In: req_url = "http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInquireSvc/
    getMsrstnAcctoRltmMesureDnsty"
    station_name = "도산대로"
    data_term = "DAILY"
    num_of_rows = 10
    page_no = 1
    version = 1.3
    output_type = "json"
    req_parameter = {"ServiceKey": API_KEY_decode,
                    "stationName": station_name,
                    "dataTerm": data_term, "ver": version,
                    "pageNo": page_no, "numOfRows" : num_of_rows,
                    "_returnType": output_type}
    dict_data = requests.get(req_url, params = req_parameter).json()
    dict_data['list'][0]
    # dict_data
Out: {'_returnType': 'json',
     'coGrade': '1',
     'coValue': '0.7',
     'dataTerm': '',
     'dateTime': '2018-06-18 21:00',
     'khaiGrade': '3',
     'khaiValue': '115',
     'mangName': '도로변대기',
```

정부의 공공 데이터 가져오기

```
'no2Grade': '3',  
'no2Value': '0.074',  
'numOfRows': '10',  
'o3Grade': '1',  
'o3Value': '0.007',  
'pageNo': '1',  
'pm10Grade': '2',  
'pm10Grade1h': '2',  
'pm10Value': '44',  
'pm10Value24': '48',  
'pm25Grade': '2',  
'pm25Grade1h': '2',  
'pm25Value': '26',  
'pm25Value24': '30',  
'resultCode': '',  
'resultMsg': '',  
'rnum': 0,  
'serviceKey': '',  
'sidoName': '',  
'so2Grade': '1',  
'so2Value': '0.005',  
'stationCode': '',  
'stationName': '',  
'totalCount': '',  
'ver': ''}
```

정부의 공공 데이터 가져오기

– 측정 정보 가져오기

```
In: dateTime = dict_data['list'][0]['dateTime']

so2Grade = dict_data['list'][0]['so2Grade']
coGrade = dict_data['list'][0]['coGrade']
o3Grade = dict_data['list'][0]['o3Grade']
no2Grade = dict_data['list'][0]['no2Grade']

pm10Grade1h = dict_data['list'][0]['pm10Grade1h']
pm25Grade1h = dict_data['list'][0]['pm25Grade1h']
khaiGrade = dict_data['list'][0]['khaiGrade']

print("[측정소({0})에서 측정된 대기 오염 상태"].format(station_name))
print("- 측정 시간:{0}".format(dateTime))

print("- [지수] ", end='')
print("아황산가스:{0}, 일산화탄소:{1}, 오존:{2}, 이산화질소:{3}".
      format(so2Grade, coGrade, o3Grade, no2Grade))

print("- [등급] ", end='')
print("미세 먼지:{0}, 초미세 먼지:{1}, 통합대기환경:{2}".
      format(pm10Grade1h, pm25Grade1h, khaiGrade))
```

정부의 공공 데이터 가져오기

– 측정 정보 가져오기

Out: [측정소(도산대로)에서 측정된 대기 오염 상태]

- 측정 시간:2018-06-18 21:00
- [지수] 아황산가스:1, 일산화탄소:1, 오존:1, 이산화질소:3
- [등급] 미세먼지:2, 초미세 먼지:2, 통합대기환경:3

```
In: gradeNum2Str = {"1":"좋음", "2":"보통", "3":"나쁨", "4":"매우나쁨" }
```

```
print("[측정소({0})에서 측정된 대기 오염 상태]".format(station_name))
print("- 측정 시간:{0}".format(dateTime))
print("- 아황산가스:{0}, 일산화탄소:{1}, 오존:{2}, 이산화질소:{3}".
      format(gradeNum2Str[so2Grade], gradeNum2Str[coGrade],
            gradeNum2Str[o3Grade], gradeNum2Str[no2Grade]))

print("- 미세먼지:{0}, 초미세 먼지:{1}, 통합대기환경:{2}".
      format(gradeNum2Str[pm10Grade1h], gradeNum2Str[pm25Grade1h],
            gradeNum2Str[khaiGrade]))
```

Out: [측정소(도산대로)에서 측정된 대기 오염 상태]

- 측정 시간:2018-06-18 21:00
- 아황산가스:좋음, 일산화탄소:좋음, 오존:좋음, 이산화질소:나쁨
- 미세먼지:보통, 초미세 먼지:보통, 통합대기환경:나쁨

감사합니다.

※ 본 교안은 강의 수강 용도로만 사용 가능합니다.
상업적 이용을 일절 금함.