

공공 데이터 API 활용

컴퓨터정보공학과 강환수 교수

프로젝트 주제 선정 이유

- 서울 지하철 역 관련 데이터는 빅데이터 처리 활용과 학습 측면에서 적절한 데이터
 - 서울 지하철 역 중심의 상권 정보를 분석해 유동인구와의 관계성 분석
 - 이를 위해 다음 공공데이터를 활용
- 서울시 열린데이터광장
 - <https://data.seoul.go.kr>
- 공공데이터포털
 - <https://www.data.go.kr/index.do>

인증키 신청 1

• 서울 열린 데이터 광장

– 서울시 지하철 호선별 역별 시간대별 승하차 인원 정보

- <http://data.seoul.go.kr/dataList/OA-12252/S/1/datasetView.do#AXexec>



교통

공공데이터

활용갤러리 등록

URL 복사

목록 이동

서울시 지하철 호선별 역별 시간대별 승하차 인원 정보

교통카드(신하플교통카드 및 1회용 교통카드)를 이용한 지하철 호선별 역별(1~9호선, 서울시 관할 운송기관에 한함) 시간대별 승하차인원을 나타내는 정보입니다.
(* 데이터 직재는 매월 5일 전월 데이터를 갱신합니다.)

데이터 정보

공개일자	2015.02.17	최신수정일자	2021.07.05
갱신주기	매월 5일	분류	교통
원본시스템	교통카드 정산시스템	저작권자	서울특별시
제공기관	서울특별시	제공부서	도시교통실 교통기획관 교통정책과
담당자	이운정 (02-2133-2236)		
원본형태	DB	제3저작권자	없음
라이선스	 저작권자표시(BY): 이용이나 변경 및 2차적 저작물의 작성을 포함한 자유이용을 허락합니다.		
관련 태그	교통, 통계, 교통카드, 지하철역, 승차, 하차, 인원		

미리보기

Sheet Open API

Open API 이용안내 인증키 신청 영세서다운로드

▶ 샘플 URL

샘플 URL	2015년 1월 지하철 호선별 역별 시간대별 승객 현황 조회 http://openapi.seoul.go.kr:8088/(인증키)/xml/CardSubwayTime/1/5/201501/
	<?xml version="1.0" encoding="UTF-8"?>

인증키 신청 2

• 서울 열린 데이터 광장

– 서울시 지하철호선별 역별 승하차 인원 정보

- <http://data.seoul.go.kr/dataList/OA-12914/S/1/datasetView.do>

The screenshot shows the '미리보기' (Preview) tab of the Open API page. It displays a sample URL and a preview of the XML data. Below the preview, there is a table of required parameters (요청인자).

요청인자

변수명	타입	변수설명	값설명
KEY	String(필수)	인증키	OpenAPI에서 발급된 인증키
TYPE	String(필수)	요청파일타입	xml: xml, xml파일: xml, 엑셀파일: xls, json파일: json
SERVICE	String(필수)	서비스명	CardSubwayStatsNew
START_INDEX	INTEGER(필수)	요청시작위치	정수 입력 (페이지 시작번호입니다: 데이터 행 시작번호)
END_INDEX	INTEGER(필수)	요청종료위치	정수 입력 (페이지 끝번호입니다: 데이터 행 끝번호)
USE_DT	STRING(필수)	사용일자	YYYYMMDD 형식의 문자열

활용 신청 1

• 공공데이터포털

– 소상공인시장진흥공단_상가(상권)정보

- <https://www.data.go.kr/data/15012005/openapi.do>

오픈API 상세

소상공인시장진흥공단_상가(상권)정보

영업 중인 전국 상가업소 데이터를 제공합니다.
(상호명, 업종코드, 업종명, 지번주소, 도로명주소, 시도, 시도 등)

1 0 0 관심

활용신청

오류신고 및
답장자 문의

OpenAPI 정보

메타데이터 다운로드

분류체계	산업·통상 중소기업 - 산업·통상기업일반	제공기관	소상공인시장진흥공단
관리부서명	상권분석실	관리부서 전화번호	042-363-7881
API 유형	REST	데이터포맷	JSON+XML
활용신청	4118	키워드	상가업소, 소상공인, 상권정보
등록	2015-12-22	수정	2020-11-23
심의유형	개발단계 : 허용 / 운영단계 : 허용		
비용부과유무	무료		
이용허락범위	이용허락범위 제한 없음		
참고문서	20200828_SEMAS_OpenAPI활용기이드.hwp		

상세기능

활용사례

상세기능

목록

반경내 상권조회

조회

행정동코드에 대한 업소정보를 조회

· 활용승인 절차 개발단계 : 허용 / 운영단계 : 허용
· 신청가능 트래픽 10,000 / 운영계정은 활용사례 등록시 신청하면 트래픽 증가 가능
· 요청주소 <http://apis.data.go.kr/8553077/api/open/xdsc/storeListInDong>
· 서비스URL <http://apis.data.go.kr/8553077/api/open/xdsc>

활용신청

활용 신청 2

• 공공데이터포털

– 전국도시철도역사정보표준데이터

• <https://www.data.go.kr/data/15013205/standard.do>

– 직접 파일로 다운로드

The screenshot shows the '표준데이터 상세' (Standard Data Detail) page for '전국도시철도역사정보표준데이터' (National Subway Station Information Standard Data). The page includes a description of the data, a table of file information, a list of data items, and a download button.

표준데이터 상세

CSW 전국도시철도역사정보표준데이터

도시철도역사정보(영원역사명, 원자역사명, 환승역역명 등)를 제공합니다. 공공데이터 개방 표준데이터 속성정보(표현형식/단위 등)는 (공공데이터 개방 표준)고시를 참고하시기 바랍니다.(장부공유>자문답>법령(고시/지침) 각 기관에서 등록한 표준데이터를 유효하여 제공합니다. 표준데이터의 갱신주기는 개별 국마다 다릅니다. (기관에서 등록한 데이터를 확인할 것으로 개별 국별 갱신시점이 다름)

오류신고 및 담당자 문의

1 0 0

파일데이터 정보 **메타데이터 다운로드**

분류체계	고종및종류 - 철도	데이터셋 유형	종류
키워드	장가장정보, 역사노선정보, 지하철역사명	수정일	2021-06-30

데이터 목록 (총 13 건)

구분	제목	제공기관	수정일
전체	대구도시철도공사_도시철도역사정보	제공기관 : 대구도시철도공사	수정일 : 2021-06-30
전체	경기도_의정부시_도시철도역사정보	제공기관 : 경기도 의정부시	수정일 : 2021-06-22
전체	광주광역시도시철도공사_도시철도역사정보	제공기관 : 광주광역시도시철도공사	수정일 : 2021-06-21
전체	대전광역시도시철도공사_도시철도역사정보	제공기관 : 대전광역시도시철도공사	수정일 : 2021-05-21
전체	부산광역시_도시철도역사정보	제공기관 : 부산광역시	수정일 : 2021-04-29

1 2 3

그라드 **Open API**

전국 도시철도역사 정보(영원역사명, 원자역사명, 환승역역명 등)를 제공합니다. 공공데이터 개방 표준데이터 속성정보(표현형식/단위 등)는 법령자치부에서 고시한 (공공데이터 개방 표준)고시(2015-51호)를 참고하시기 바랍니다.

· 활용승인 절차 개발단계 : 허용 / 운영단계 : 허용
· 신청가능 트래픽 1,000
· 요청주소 http://api.data.go.kr/openapi/ttn_public_citizenroad_sttn_api

활용신청

파일 `my-open-API-Term-Project.ipynb`

인터넷을 통한 데이터 획득

- `서울 열린데이터 광장`에서 공공 API로 다음 데이터를 획득
 - <https://data.seoul.go.kr/index.do>
 - 서울시 지하철호선별 역별 시간대별 승하차 인원 정보: 시간대별 월 통계
 - <http://data.seoul.go.kr/dataList/OA-12252/S/1/datasetView.do#AXexec>
 - 서울시 지하철호선별 역별 승하차 인원 정보: 일별 통계
 - <http://data.seoul.go.kr/dataList/OA-12914/S/1/datasetView.do>
- `공공데이터포털`에서 공공API로 다음 데이터를 획득한다.
 - <https://www.data.go.kr/index.do>
 - 위도, 경도에 따른 저장한 반경내 상권정보
 - <https://www.data.go.kr/data/15012005/openapi.do>

한글 엑셀 파일 열기

- 내려 받은 파일 열기

- '서울시 지하철 호선별 역별 시간대별 승하차 인원 정보.csv'

```
import pandas as pd
```

```
# 한글 처리를 위해 인코딩 utf-8로 읽기
```

```
# df = pd.read_csv('서울시 지하철 호선별 역별 시간대별 승하차 인원 정보.csv', encoding='utf-8')
```

```
df = pd.read_csv('서울시 지하철 호선별 역별 시간대별 승하차 인원 정보.csv', encoding='cp949')
```

```
df.head()
```

공공 API로 다음 데이터 내려받기

- `서울시 지하철호선별 역별 월별 시간대별 승하차 인원 정보`
 - API_2021년3월_지하철역별_시간대별_승하차인원.csv
 - 오픈 API로 데이터를 얻어오기 위해 다음 사이트에서 인증키를 신청 후 사용
 - <https://data.seoul.go.kr/together/mypage/actkeyMain.do>
- 서울열린데이터광장 '서울시 지하철호선별 역별 시간대별 승하차 인원 정보' 공공API 호출 방식 참조
 - <http://data.seoul.go.kr/dataList/OA-12252/S/1/datasetView.do#AXexec>
- 다음으로 url 주소 확인, 인증키를 실제 키 값으로 입력하면 조회 가능
 - 2015년 1월 지하철 호선별 역별 시간대별 승객 현황 조회(1에서 1000천 조회. 천개 이상은 조회 불가)
 - [http://openapi.seoul.go.kr:8088/\(인증키\)/xml/CardSubwayTime/1/1000/201501/](http://openapi.seoul.go.kr:8088/(인증키)/xml/CardSubwayTime/1/1000/201501/)

주소 구성

```
In [10]: # 받은 인증키 저장
valid_key = '7a47797456676865313133704270746e'
# 가져올 시작 번호, 끝 번호 지정, 1000번 이상은 오류 발생
start, end = 1, 1000
# 년월 지정
month = '202103'

#URL 주소 완성
url = 'http://openapi.seoul.kr:8088/'
url += valid_key + '/json/CardSubwayTime/'
url += str(start) + '/' + str(end) + '/' + month
url
```

Df 생성

```
In [11]: import urllib
import json
import pandas as pd

# url을 불러오고 이것을 인코딩을 utf-8로 전환하여 결과를 받자.
response = urllib.request.urlopen(url)
json_str = response.read().decode("utf-8")

# 받은 데이터가 문자열이라서 이를 json으로 변환한다.
json_object = json.loads(json_str)

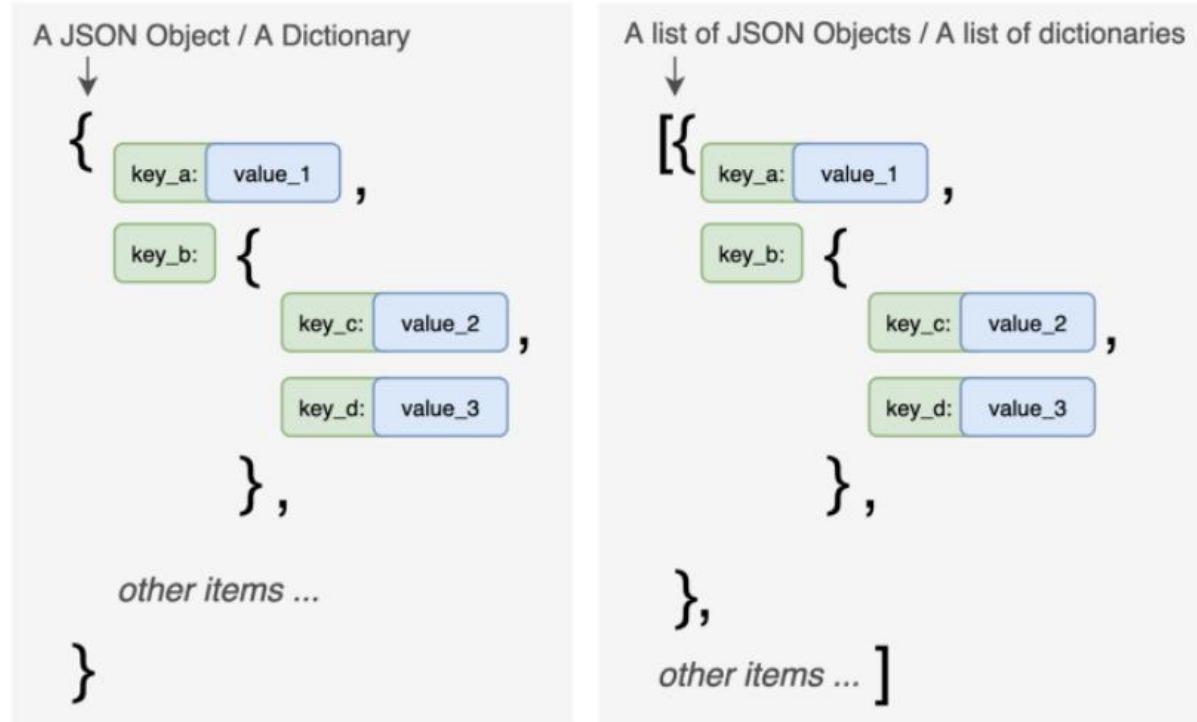
# ['CardSubwayStatsNew']['row']를 데이터프레임 df로 불러오기
df = pd.json_normalize(json_object['CardSubwayTime']['row'])
df.shape
```

Out[11]: (606, 52)

JSON

• JSON (JavaScript Object Notation) 데이터

- 데이터 저장 및 교환에 널리 사용되는 형식
- 예를 들어 MongoDB와 같은 NoSQL 데이터베이스는 데이터를 JSON 형식으로 저장하고 REST API의 응답은 대부분 JSON으로 제공



JSON 다루기

- 사전리스트 생성

```
In [9]: dlist = [ {c:i for c in list('ABCD')} for i in range(4) ]
         dlist
```

```
Out[9]: [{ 'A': 0, 'B': 0, 'C': 0, 'D': 0},
         { 'A': 1, 'B': 1, 'C': 1, 'D': 1},
         { 'A': 2, 'B': 2, 'C': 2, 'D': 2},
         { 'A': 3, 'B': 3, 'C': 3, 'D': 3}]
```

```
In [11]: import pandas as pd
         mydf = pd.json_normalize(dlist)
         mydf
```

Out[11]:

	A	B	C	D
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

```
In [13]: mydf.to_json('mydata.json')
         pd.read_json('mydata.json')
```

Out[13]:

	A	B	C	D
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

```
In [14]: !type mydata.json
```

```
{ "A": { "0": 0, "1": 1, "2": 2, "3": 3 }, "B": { "0": 0, "1": 1, "2": 2, "3": 3 }, "C": { "0": 0, "1": 1, "2": 2, "3": 3 }, "D": { "0": 0, "1": 1, "2": 2, "3": 3 } }
```

JSON 정규화

• pandas의 pd.json_normalize

- 딕셔너리 비슷한 JSON 포맷의 평탄화 작업에 위해 데이터프레임으로 생성

In [14]: `import pandas as pd`

```
a_dict = {
    'school': 'ABC primary school',
    'location': 'London',
    'ranking': 2,
}

df = pd.json_normalize(a_dict)
df
```

Out [14]:

	school	location	ranking
0	ABC primary school	London	2

```
In [15]: json_list = [
    { 'class': 'Year 1', 'student number': 20, 'room': 'Yellow' },
    { 'class': 'Year 2', 'student number': 25, 'room': 'Blue' },
]
pd.json_normalize(json_list)
```

Out [15]:

	class	student number	room
0	Year 1	20	Yellow
1	Year 2	25	Blue

```
In [16]: json_list = [
    { 'class': 'Year 1', 'num_of_students': 20, 'room': 'Yellow' },
    { 'class': 'Year 2', 'room': 'Blue' }, # no num_of_students
]
pd.json_normalize(json_list)
```

Out [16]:

	class	num_of_students	room
0	Year 1	20.0	Yellow
1	Year 2	NaN	Blue

중첩된 사전

- 모든 중첩 된 값은 평면화되고 별도의 열로 변환

```
In [17]: json_obj = {
    'school': 'ABC primary school',
    'location': 'London',
    'ranking': 2,
    'info': {
        'president': 'John Kasich',
        'contacts': {
            'email': {
                'admission': 'admission@abc.com',
                'general': 'info@abc.com'
            },
            'tel': '123456789',
        },
    },
}
pd.json_normalize(json_obj)
```

```
Out [17]:
```

	school	location	ranking	info.president	info.contacts.email.admission	info.contacts.email.general	info.contacts.tel
0	ABC primary school	London	2	John Kasich	admission@abc.com	info@abc.com	123456789

날짜 정보 생성

• 특정 년도의 1년 날짜 생성

- API를 위한 날짜와 URL 생성
 - 2019년 모든 날짜를 리스트 `date_range()`로 생성

```
In [187]: import pandas as pd

# date2019 = pd.date_range(start = '2019-1-1', periods=365)
date2019 = pd.date_range(start = '2019-1-1', end = '2019-12-31')
date2019
```

```
Out[187]: DatetimeIndex(['2019-01-01', '2019-01-02', '2019-01-03', '2019-01-04',
                        '2019-01-05', '2019-01-06', '2019-01-07', '2019-01-08',
                        '2019-01-09', '2019-01-10',
                        ...,
                        '2019-12-22', '2019-12-23', '2019-12-24', '2019-12-25',
                        '2019-12-26', '2019-12-27', '2019-12-28', '2019-12-29',
                        '2019-12-30', '2019-12-31'],
                        dtype='datetime64[ns]', length=365, freq='D')
```

날짜 정보

• 문자열 변환

```
In [188]: lst2019 = []
           for date in date2019:
               lst2019.append(f'{str(date)[:4]}{str(date)[5:7]}{str(date)[8:10]}')
           lst2019[-10:]
```

```
Out [188]: ['20191222',
            '20191223',
            '20191224',
            '20191225',
            '20191226',
            '20191227',
            '20191228',
            '20191229',
            '20191230',
            '20191231']
```



- 문자열에서 숫자만 남기는 다른 방법 3가지

```
In [200]: string = '2021-03-01'

           new_string = ''.join(char for char in string if char.isdigit())
           print(new_string)

           20210301
```

```
In [203]: lst2019 = []
           for date in date2019:
               s = str(date)[:10]
               lst2019.append(''.join(c for c in s if c.isdigit()))
           lst2019[-10:]
```

```
Out [203]: ['20191222',
            '20191223',
            '20191224',
            '20191225',
            '20191226',
```

```
In [204]: string = '2021-03-01'
```

```
new_string = ''.join(filter(str.isdigit, string))
print(new_string)
```

```
20210301
```

```
In [205]: lst2019 = []
for date in date2019:
    s = str(date)[:10]
    lst2019.append(''.join(filter(str.isdigit, s)))
lst2019[-10:]
```

```
Out [205]: ['20191222',
            '20191223',
            '20191224',
            '20191225',
            '20191226',
            '20191227',
            '20191228',
            '20191229',
            '20191230',
            '20191231']
```



```
In [206]: import re
```

```
string = '2021-03-01'
```

```
new_string = re.sub(r"^[0-9]", "", string)
print(new_string)
```

```
20210301
```

```
In [208]: lst2019 = []
for date in date2019:
    lst2019.append(re.sub(r"^[0-9]", '', str(date)[:10]))
lst2019[-10:]
```

```
Out [208]: ['20191222',
            '20191223',
            '20191224',
            '20191225',
            '20191226',
```

공공 데이터 API호출 및 pandas 로 변환하기

- <https://m.blog.naver.com/varkiry05/221696609835>

공공 데이터 API 활용

folium


파일 `my-folium.ipynb`

패키지 folium

- 지도를 그리는 패키지
 - 사용법
 - <https://continuous-development.tistory.com/152>

지도 그리기

```
In [ ]: import folium  
        folium.__version__
```

```
In [ ]: folium.Map? 
```

```
In [ ]: folium.Map((45.523, -122.675))
```

```
In [ ]: folium.Map([45.523, -122.675])
```

```
In [ ]: folium.Map(location=[45.523, -122.675])
```


```
In [ ]: folium.Map(location=[37.5, 127.2])
```

```
In [ ]: folium.Map(location=[45.523, -122.675], width=750, height=500)
```


줌과 지도 종류

```
In [ ]: folium.Map(location=[45.523, -122.675], tiles='cartodb positron')
```

```
In [ ]: m = folium.Map(
    location=[45.523, -122.675],
    zoom_start=8
)
m
```



```
In [ ]:
```

```
In [ ]: # 그릴 위치의 평균 값의 위도, 경도를 지정해서 그리기
map = folium.Map(location=[37.42, 127.02])
map
```

```
In [ ]: m = folium.Map(
    location = [36.5053542, 127.7043419],
    zoom_start = 8,
    tiles = 'Cartodb Positron'
)
m
```

원 그리기

```
In [ ]: latlon = [[37.31355679999999, 127.08034150000003],
[37.359593000000016, 127.105316],
[37.388204699999996, 126.66208460000007],
[37.19821445962207, 127.07333060688757],
[37.3862876275833, 126.96253325015414],
[37.31864776315991, 127.08885641049494],
[37.56661020000001, 126.97838810000007]
]
m = folium.Map(
    location = [36.5053542, 127.7043419],
    zoom_start = 8,
    tiles = 'Cartodb Positron'
)
for i in range(len(latlon)):
    folium.Circle(
        location = latlon[i],
        radius = 200,
        color = '#000000',
        fill = 'crimson',
    ).add_to(m)
m
```

```
In [ ]: m.save('map.html')
```

히트맵

```

In [ ]: from folium.plugins import HeatMap

latlon = [[37.31355679999999, 127.08034150000003],
           [37.35959300000016, 127.105316],
           [37.388204699999996, 126.66208460000007],
           [37.19821445962207, 127.07333060688757],
           [37.3862876275833, 126.96253325015414],
           [37.31864776315991, 127.08885641049494],
           [37.56661020000001, 126.97838810000007]
          ]

m = folium.Map(
    location = [36.5053542, 127.7043419],
    zoom_start = 8,
    tiles = 'Cartodb Positron'
)

HeatMap(latlon).add_to(m)
m

```