

12

데이터 크롤링

데이터 크롤링 방법

▪ BeautifulSoup 라이브러리

```
from bs4 import BeautifulSoup
```

```
page = open('data/test.html', 'r', encoding='utf-8').read()
soup = BeautifulSoup(page, 'html.parser')
print(soup.prettify())#html 페이지의 내용을 전체 다 보고 싶으면 prettify() 옵션 사용
```

Happy PinkWink. [경남대학교](#)

Happy Data Science. [Python](#)

Data Science is funny.

All I need is Love.

<test.html>

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Very Simple HTML Code by PinkWink
    </title>
  </head>
  <body>
    <div>
      <p class="inner-text first-item" id="first">
        Happy PinkWink.
        <a href="http://www.kyungnam.ac.kr" id="pw-link">
          경남대학교
        </a>
      </p>
      <p class="inner-text second-item">
        Happy Data Science.
        <a href="https://www.python.org" id="py-link">
          Python
        </a>
      </p>
    </div>
    <p class="outer-text first-item" id="second">
      <b>
        Data Science is funny.
      </b>
    </p>
    <p class="outer-text">
      <b>
        All I need is Love.
      </b>
    </p>
  </body>
</html>
```

<실행 결과>

데이터 크롤링 방법

■ body 찾기

`list(soup.children)#태그 확인`

```
['html',
 '\n',
 <html>
 <head>
 <title>Very Simple HTML Code by PinkWink</title>
 </head>
 <body>
 <div>
 <p class="inner-text first-item" id="first">
     Happy PinkWink.
     <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>
 </p>
 <p class="inner-text second-item">
     Happy Data Science.
     <a href="https://www.python.org" id="py-link">Python</a>
 </p>
 </div>
 <p class="outer-text first-item" id="second">
 <b>
     Data Science is funny.
 </b>
 </p>
 <p class="outer-text">
 <b>
     All I need is Love.
 </b>
 </p>
 </body>
 </html>]
```

<실행 결과>

`html = list(soup.children)[2]`
`html`

```
<html>
<head>
<title>Very Simple HTML Code by PinkWink</title>
</head>
<body>
<div>
<p class="inner-text first-item" id="first">
    Happy PinkWink.
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>
</p>
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.python.org" id="py-link">Python</a>
</p>
</div>
<p class="outer-text first-item" id="second">
<b>
    Data Science is funny.
</b>
</p>
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>
</body>
</html>
```

<실행 결과>

데이터 크롤링 방법

■ body 찾기

```
list(html.children)
```

```
['\n',  
<head>  
<title>Very Simple HTML Code by PinkWink</title>  
</head>,  
'\n',  
<body>  
<div>  
<p class="inner-text first-item" id="first">  
    Happy PinkWink.  
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>  
</p>  
<p class="inner-text second-item">  
    Happy Data Science.  
    <a href="https://www.python.org" id="py-link">Python</a>  
</p>  
</div>  
<p class="outer-text first-item" id="second">  
<b>  
    Data Science is funny.  
</b>  
</p>  
<p class="outer-text">  
<b>  
    All I need is Love.  
</b>  
</p>  
</body>,  
'\n']
```

<실행 결과>
<실행 결과>

```
body = list(html.children)[3]  
body
```

```
<body>  
<div>  
<p class="inner-text first-item" id="first">  
    Happy PinkWink.  
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>  
</p>  
<p class="inner-text second-item">  
    Happy Data Science.  
    <a href="https://www.python.org" id="py-link">Python</a>  
</p>  
</div>  
<p class="outer-text first-item" id="second">  
<b>  
    Data Science is funny.  
</b>  
</p>  
<p class="outer-text">  
<b>  
    All I need is Love.  
</b>  
</p>  
</body>
```

<실행 결과>

데이터 크롤링 방법

■ body 찾기-2

`soup.body`

```
<body>
<div>
<p class="inner-text first-item" id="first">
    Happy PinkWink.
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>
</p>
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.python.org" id="py-link">Python</a>
</p>
</div>
<p class="outer-text first-item" id="second">
<b>
    Data Science is funny.
</b>
</p>
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>
</body>
```

<실행 결과>

■ body 찾기-3

`list(body.children)`

```
['\n',
<div>
<p class="inner-text first-item" id="first">
    Happy PinkWink.
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>
</p>
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.python.org" id="py-link">Python</a>
</p>
</div>,
'\n',
<p class="outer-text first-item" id="second">
<b>
    Data Science is funny.
</b>
</p>,
'\n',
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>,
'\n']
```

<실행 결과>

데이터 크롤링 방법

■ 태그 찾기(find, find_all)

`soup.find_all('p')`

```
[<p class="inner-text first-item" id="first">
    Happy PinkWink.
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>
</p>,
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.python.org" id="py-link">Python</a>
</p>,
<p class="outer-text first-item" id="second">
<b>
    Data Science is funny.
</b>
</p>,
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>]
```

<실행 결과>

`soup.find('p')#제일 처음 하나만 찾을`

```
<p class="inner-text first-item" id="first">
    Happy PinkWink.
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>
</p>
```

#p 태그의 class가 outer-text인 것을 찾는 것도 가능
`soup.find_all('p', class_='outer-text')`

```
[<p class="outer-text first-item" id="second">
<b>
    Data Science is funny.
</b>
</p>,
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>]
```

#class 이름으로 outer-text인 것을 찾는 것도 가능
`soup.find_all(class_='outer-text')`

#id가 first인 태그들을 찾을 수 있음
`soup.find_all(id='first')`

```
[<p class="inner-text first-item" id="first">
    Happy PinkWink.
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>
</p>]
```

데이터 크롤링 방법

- 원하는 태그 찾고 싶을 때

```
soup.head()
```

```
[<title>Very Simple HTML Code by PinkWink</title>]
```

```
# next_sibling을 통해 soup의 head 다음 확인 가능  
soup.head.next_sibling
```

```
'\n'
```

```
# head와 같은 위치에 있던 body 태그로 접근하는 방법  
soup.head.next_sibling.next_sibling
```

```
<body>  
<div>  
<p class="inner-text first-item" id="first">  
    Happy PinkWink.  
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>  
  
</p>  
<p class="inner-text second-item">  
    Happy Data Science.  
    <a href="https://www.python.org" id="py-link">Python</a>  
  
</p>  
</div>  
<p class="outer-text first-item" id="second">  
<b>  
    Data Science is funny.  
</b>  
  
</p>  
<p class="outer-text">  
<b>  
    All I need is Love.  
</b>  
  
</p>  
</body>
```

데이터 크롤링 방법

- 원하는 태그 찾고 싶을 때

body.p

```
<p class="inner-text first-item" id="first">  
    Happy PinkWink.  
    <a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>  
</p>
```

link를 두 번 걸어 다음 p 태그로 이동할 수 있다.

body.p.next_sibling.next_sibling

```
<p class="inner-text second-item">  
    Happy Data Science.  
    <a href="https://www.python.org" id="py-link">Python</a>  
</p>
```


데이터 크롤링 방법

- `get_text()`를 이용하여 텍스트 가져오기

```
# get_text() 태그 안에 있는 텍스트 가지고 옴.  
for each_tag in soup.find_all('p'):  
    print(each_tag.get_text())
```

Happy PinkWink.
경남대학교

Happy Data Science.
Python

Data Science is funny.

All I need is Love.

```
# body 전체에서 get_text를 하면 줄바꿈(\n)  
body.get_text()
```

'\n\n\n\n'

Happy PinkWink.\nData Science is funny.\n

경남대학교\n\n\n\n

Happy Data Science.\nAll I need is Love.\n

Python\n\n\n'

데이터 크롤링 방법

- href 링크 주소 찾기

```
links = soup.find_all('a')  
links
```

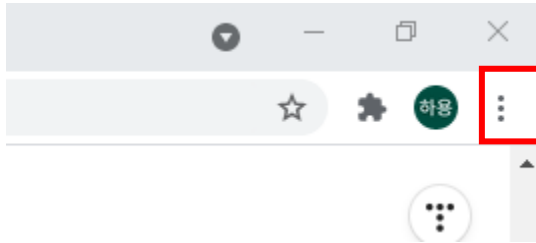
```
[<a href="http://www.kyungnam.ac.kr" id="pw-link">경남대학교</a>,  
  <a href="https://www.python.org" id="py-link">Python</a>]
```

```
# links에서 href 속성으로 링크 주소 획득  
for each in links:  
    href = each['href']  
    text = each.string  
    print(text + '->' + href)
```

```
경남대학교->http://www.kyungnam.ac.kr  
Python->https://www.python.org
```

데이터 크롤링 방법

■ 크롬 개발자 도구를 이용해서 원하는 태그 찾기



<https://finance.naver.com/marketindex/>

- 도구 더보기/개발 도구 클릭
- select element/ 1,117.50클릭
- span 태그의 value class 획득

데이터 크롤링 방법

- 텍스트 획득

```
from urllib.request import urlopen

url = 'https://finance.naver.com/marketindex/'
page = urlopen(url)

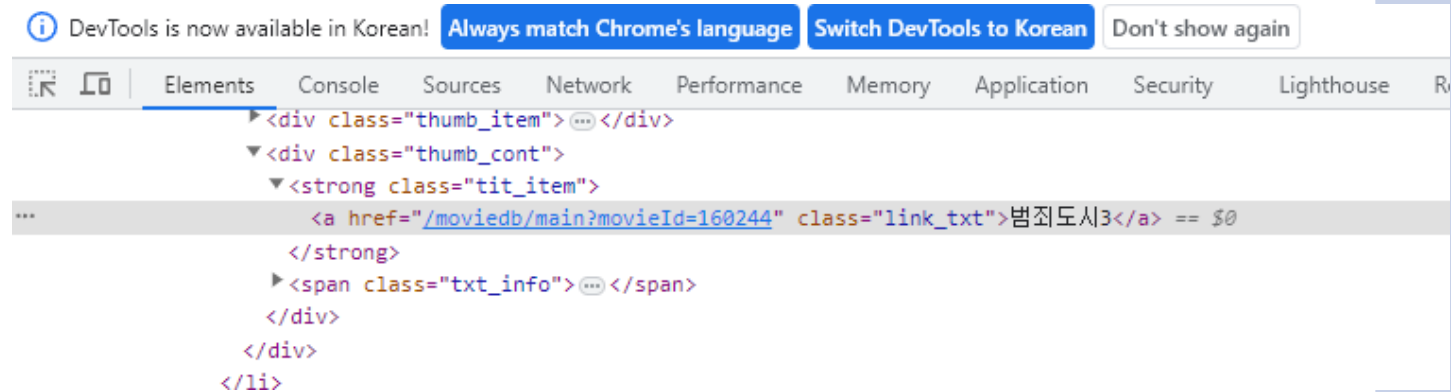
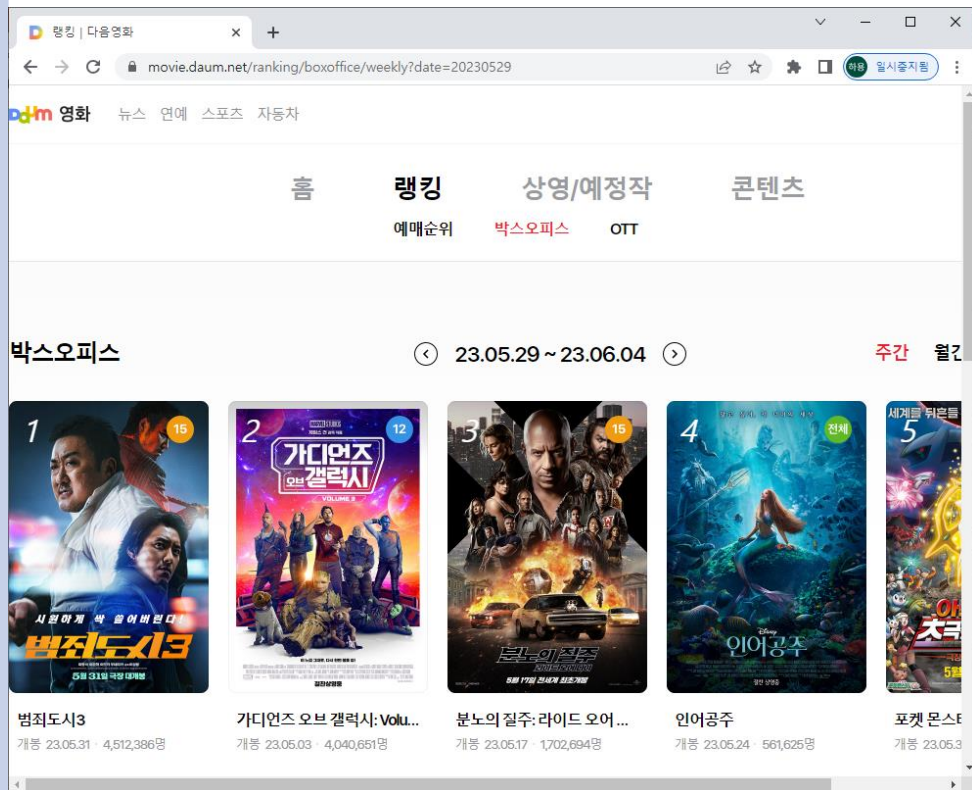
soup = BeautifulSoup(page, 'html.parser')

soup.find_all('span', 'value')[0].string
```

'1,117.50'

데이터 크롤링 실습

- 다음 영화 평점 기준 영화의 평점 변화 확인하기
(<https://movie.daum.net/ranking/boxoffice/weekly?date=20230529>)



데이터 크롤링 실습

- 다음 영화 평점 기준 영화의 평점 변화 확인하기
(<https://movie.daum.net/ranking/boxoffice/weekly?date=20230529>)

```
from bs4 import BeautifulSoup
import pandas as pd
```

```
from urllib.request import urlopen
```

```
url = 'https://movie.daum.net/ranking/boxoffice/weekly?date=20230529'
```

```
page = urlopen(url)
```

```
soup = BeautifulSoup(page, "html.parser")
```

```
soup.find_all('strong', 'tit_item')[:10]
```

```
[<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=196843" title="극장판 바이올렛 에버가든">극장판 바이올렛 에버가든</a>
</div>,
<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=17170" title="레옹">레옹</a>
</div>,
<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=154573" title="다시 태어나도 우리">다시 태어나도 우리</a>
</div>,
<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=35187" title="피아니스트">피아니스트</a>
</div>,
<div class="tit5">
```

데이터 크롤링 실습

```
# 첫번째 제목  
soup.find_all('strong', 'tit_item')[0].a.string
```

'범죄도시3'

```
soup.find_all('span', 'txt_info')[0].get_text()[15:].split('명')[0]
```

'4,512,386'

데이터 크롤링 실습

■ 네이버 영화 평점 기준 영화의 평점 변화 확인하기 (<https://movie.naver.com/movie/sdb/rank/rmovie.nhn>)

```
from bs4 import BeautifulSoup
import pandas as pd
```

```
from urllib.request import urlopen
```

```
url = 'https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20210501'
```

```
page = urlopen(url)
```

```
soup = BeautifulSoup(page, "html.parser")
```

```
soup.find_all('div', 'tit5')[1:10]
```

```
[<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=196843" title="극장판 바이올렛 에버가든">극장판 바이올렛 에버가든</a>
</div>,
<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=17170" title="레옹">레옹</a>
</div>,
<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=154573" title="다시 태어나도 우리">다시 태어나도 우리</a>
</div>,
<div class="tit5">
  <a href="/movie/bi/mi/basic.nhn?code=35187" title="피아니스트">피아니스트</a>
</div>,
<div class="tit5">
```

NAVER 영화

영화홈
상영작 · 예정작
영화랭킹
랭킹
디렉토리
예매
평점 · 리뷰
다운로드
인디극장

로그인
영화검색
검색

랭킹

영화 | 영화인

영화 랭킹

조회순 | **평점순 (현재상영중)** | 평점순 (모든영화) | 2021.05.01

순위	영화명	평점	평점주	변동폭
1	극장판 바이올렛 에버가든	★★★★★ 9.48	평점주	- 0
2	레옹	★★★★★ 9.37	평점주	- 0
3	다시 태어나도 우리	★★★★★ 9.35	평점주	- 0
4	피아니스트	★★★★★ 9.33	평점주	↑ 1
5	미안해요, 리키	★★★★★ 9.32	평점주	↑ 1
6	패왕별희 디 오리저널	★★★★★ 9.32	평점주	↑ 1
7	부활: 그 증거	★★★★★ 9.31	평점주	↓ 3
8	극장판 귀멸의 칼날: 무한열차편	★★★★★ 9.30	평점주	- 0
9	소울	★★★★★ 9.30	평점주	- 0
10	자산어보	★★★★★ 9.30	평점주	- 0

영화 인기검색어 더보기

- 1 비와 당신의 이야기 ↑ 1
- 2 미나리 ↑ 1
- 3 서복 ↓ 2
- 4 노매드랜드 ↑ 1
- 5 내일의 기억 ↓ 1

2021.05.01

영화인 인기검색어 더보기

- 1 윤여정 - 0
- 2 전여빈 - 0
- 3 글렌 클로즈 - 0
- 4 클로이 자오 - 0
- 5 프란시스 맥도먼드 - 0

2021.05.01

데이터 크롤링 실습

첫번째 제목

```
soup.find_all('div', 'tit5')[0].a.string
```

'극장판 바이올렛 에버가든'

첫번째 평점

```
soup.find_all('td', 'point')[0].string
```

'9.48'

4월 1일부터 30일간의 날짜를 정의하자

```
date = pd.date_range('2021-4-1', periods=30, freq='D')  
date
```

```
DatetimeIndex(['2021-04-01', '2021-04-02', '2021-04-03', '2021-04-04',  
               '2021-04-05', '2021-04-06', '2021-04-07', '2021-04-08',  
               '2021-04-09', '2021-04-10', '2021-04-11', '2021-04-12',  
               '2021-04-13', '2021-04-14', '2021-04-15', '2021-04-16',  
               '2021-04-17', '2021-04-18', '2021-04-19', '2021-04-20',  
               '2021-04-21', '2021-04-22', '2021-04-23', '2021-04-24',  
               '2021-04-25', '2021-04-26', '2021-04-27', '2021-04-28',  
               '2021-04-29', '2021-04-30'],  
              dtype='datetime64[ns]', freq='D')
```

데이터 크롤링 실습

- 날짜별로 크롤링

```
import urllib
from tqdm import tqdm_notebook

movie_date = []
movie_name = []
movie_point = []

for today in tqdm_notebook(date):
    html = 'http://movie.naver.com/' + 'movie/sdb/rank/rmovie.nhn?sel=cur&date={date}'
    response = urlopen(html.format(date= urllib.parse.quote(today.strftime('%Y%m%d'))))
    soup = BeautifulSoup(response, 'html.parser')

    end = len(soup.find_all('td', 'point'))

    movie_date.extend([today for n in range(0, end)])
    movie_name.extend([soup.find_all('div', 'tit5')[n].a.string for n in range(0, end)])
    movie_point.extend([soup.find_all('td', 'point')[n].string for n in range(0, end)])
```

데이터 크롤링 실습

```
# 읽은 내용을 pandas로 저장. 날짜별로 영화와 포인트가 저장.  
movie = pd.DataFrame({'date':movie_date, 'name':movie_name, 'point':movie_point})  
movie.head()
```

	date	name	point
0	2021-04-01	극장판 바이올렛 에버그든	9.48
1	2021-04-01	가나의 혼인잔치: 언약	9.42
2	2021-04-01	반지의 제왕: 왕의 귀환	9.38
3	2021-04-01	죽은 시인의 사회	9.38
4	2021-04-01	반지의 제왕: 두 개의 탑	9.36

```
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1500 entries, 0 to 1499  
Data columns (total 3 columns):  
#   Column  Non-Null Count  Dtype  
---  ---  
0    date    1500 non-null    datetime64[ns]  
1    name    1500 non-null    object  
2    point   1500 non-null    object  
dtypes: datetime64[ns](1), object(2)  
memory usage: 35.3+ KB
```

데이터 크롤링 실습

```
# point object 형을 float형으로 변환
movie['point'] = movie['point'].astype(float)
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0   date    1500 non-null    datetime64[ns]
1   name    1500 non-null    object
2   point   1500 non-null    float64
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 35.3+ KB
```

```
# pivot_table을 영화별로 점수 합산.
# aggfunc으로 np.sum을 이용해서 합산하여 영화별 점수의 합계로 정렬
import numpy as np
```

```
movie_unique = pd.pivot_table(movie, index=['name'],aggfunc=np.sum)
movie_best = movie_unique.sort_values(by='point', ascending=False)
movie_best.head()
```

	point
name	
패왕별희 디 오리지널	279.55
극장판 귀멸의 칼날: 무한열차편	279.30
소울	279.00
자산어보	277.44
라야와 마지막 드래곤	274.59

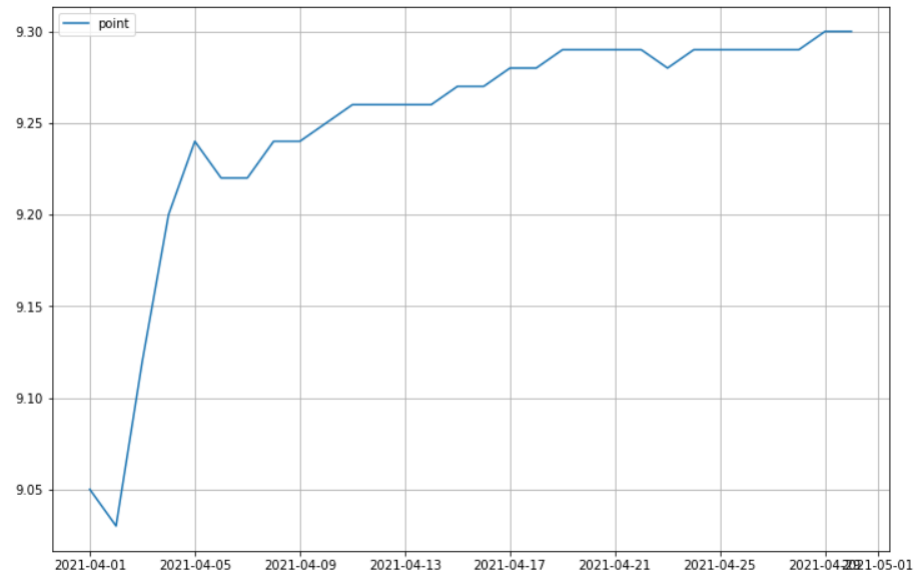
데이터 크롤링 실습

```
# 한 가지 영화만 날짜별 평점 변화 확인
tmp = movie.query('name=="자산어보"')
tmp.head()
```

```
# 날짜별로 그래프를 그려보자
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(12,8))
plt.plot(tmp['date'], tmp['point'])
plt.legend(['point'])
plt.grid()
plt.show
```

	date	name	point
17	2021-04-01	자산어보	9.05
69	2021-04-02	자산어보	9.03
116	2021-04-03	자산어보	9.12
163	2021-04-04	자산어보	9.20
210	2021-04-05	자산어보	9.24



데이터 크롤링 실습

■ Selenium 사용하기

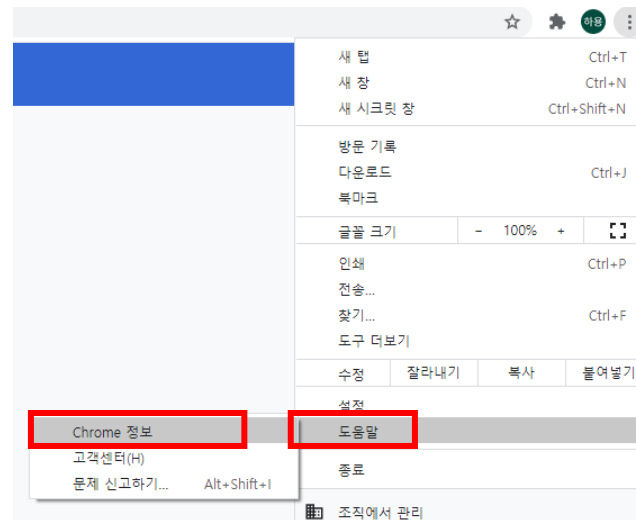
```
from selenium import webdriver
```

selenium 사용하기 위해서는 chromedriver 필요

<https://chromedriver.chromium.org/downloads>

방문하여 사용하고 있는 크롬 브라우저 버전과 일치하는 크롬 드라이버 다운로드

크롬 드라이버 버전 확인 방법
도움말/Chrome 정보



Chrome 정보



Chrome



Chrome이 최신 버전입니다.

버전 90.0.4430.93(공식 빌드) (64비트)

Chrome 도움말 보기

문제 신고

데이터 크롤링 실습

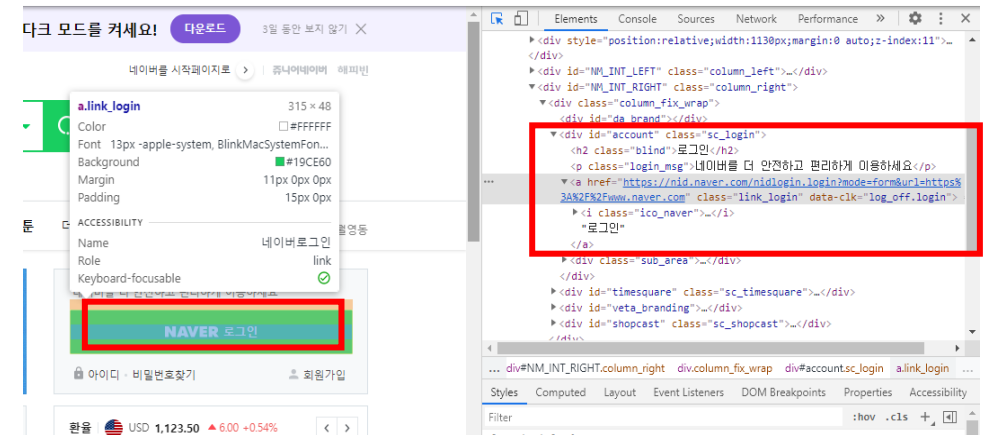
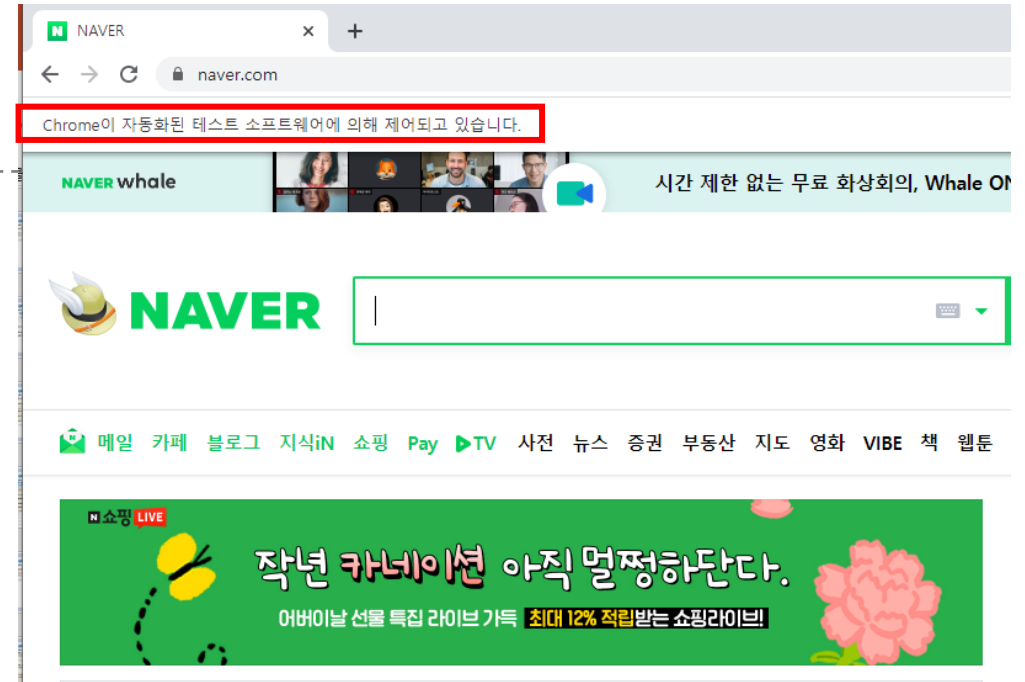
■ Selenium 사용하기

```
from selenium import webdriver
```

```
#chromedriver 저장 경로 지정  
driver = webdriver.Chrome('C:\data\chromedriver')  
driver.get('http://naver.com')
```

■ 개발자 도구 확인

```
xpath='//*[@id="account"]'  
driver.find_element_by_xpath(xpath).click()
```



데이터 크롤링 실습

- 로그인 과정 확인

```
elem_login = driver.find_element_by_id('id')
elem_login.clear()
elem_login.send_keys('*****')
```

```
elem_login = driver.find_element_by_id('pw')
elem_login.clear()
elem_login.send_keys('*****')
```

```
# 태그 오른쪽 클릭->Copy->xpath 밑에 붙여넣기
# find_element_by_xpath(xpath) : xpath 위치 찾기
# click() : 로그인 버튼 클릭
xpath='//*[@id="log.login"]'
driver.find_element_by_xpath(xpath).click()
```


데이터 크롤링 실습

■ 서울 주유소 가격 정보 비교

```
from selenium import webdriver
from selenium.webdriver.common.by import By
driver = webdriver.Chrome('C:\data\chromedriver')
driver.get("http://www.opinet.co.kr")
```

```
driver.get("http://www.opinet.co.kr/searRgSelect.do")
```

서울 클릭을 위한 xpath 확인

```
area = driver.find_element(By.XPATH, '//*[@id="SIDO_NM0"]')
```

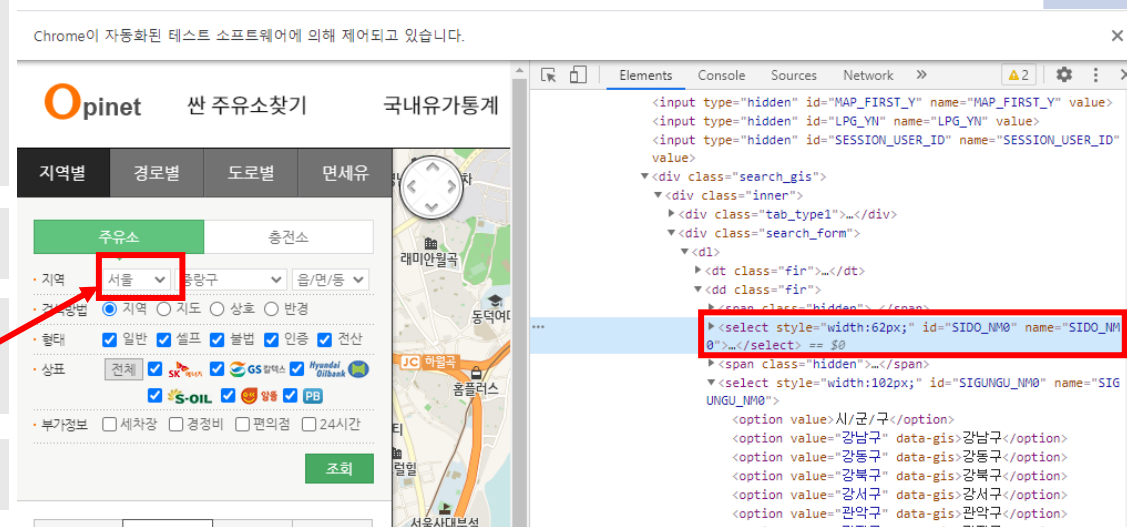
```
area.send_keys('서울')
```

구/데이터 입력을 위한 xpath 확인

```
gu_list_raw = driver.find_element(By.XPATH, '//*[@id="SIGUNGU_NM0"]')
```

구 리스트 확인 위해 find_elements_by_tag_name으로 option 태그 검색

```
gu_list = gu_list_raw.find_elements(By.TAG_NAME, 'option')
```



데이터 크롤링 실습

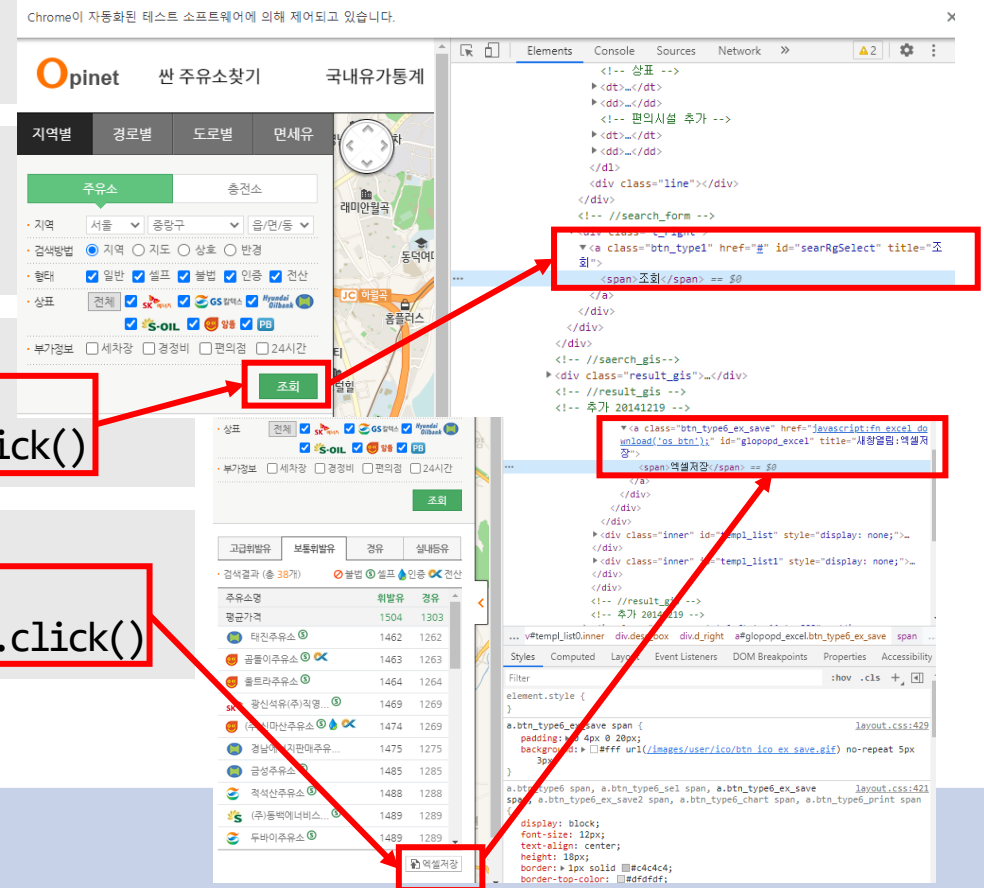
서울 주유소 가격 정보 비교

```
#value 속성을 이용하여 구 리스트 획득
gu_names = [option.get_attribute('value') for option in gu_list]
gu_names.remove('')
gu_names
```

```
#gu_names에서 리스트 첫번째 값 입력하여 테스트 진행
element = driver.find_element(By.ID, 'SIGUNGU_NM0')
element.send_keys(gu_names[0])
```

```
#조회버튼의 Xpath를 찾아서 클릭
xpath = ''//*[@id="searRgSelect"]/span''
element_sel_gu = driver.find_element(By.XPATH, xpath).click()
```

```
#엑셀 저장 버튼 클릭하여 엑셀 내용 저장 테스트
xpath = ''//*[@id="glopdp_excel"]/span''
element_get_excel = driver.find_element(By.XPATH, xpath).click()
```



데이터 크롤링 실습

■ 구별 주유 가격 정리

```
import time
from tqdm import tqdm_notebook

# 반복문을 이용하여 모든 구 엑셀파일 다운로드 진행
for gu in tqdm_notebook(gu_names):
    element = driver.find_element(By.ID, 'SIGUNGU_NM0')
    element.send_keys(gu)

    time.sleep(2) # 데이터 획득 위한 지연 시간












    xpath = ''//*[@id="searRgSelect"]/span''
    element_sel_gu = driver.find_element(By.XPATH, xpath).click()

    time.sleep(1)

    xpath = ''//*[@id="glopopd_excel"]/span''
    element_get_excel = driver.find_element(By.XPATH, xpath).click()

    time.sleep(1)
```

이름

 지역_위치별(주유소)
 지역_위치별(주유소) (24)
 지역_위치별(주유소) (23)
 지역_위치별(주유소) (22)
 지역_위치별(주유소) (21)
 지역_위치별(주유소) (20)
 지역_위치별(주유소) (19)
 지역_위치별(주유소) (18)
 지역_위치별(주유소) (17)
 지역_위치별(주유소) (16)
 지역_위치별(주유소) (15)
 지역_위치별(주유소) (14)
 지역_위치별(주유소) (13)
 지역_위치별(주유소) (12)
 지역_위치별(주유소) (11)
 지역_위치별(주유소) (10)
 지역_위치별(주유소) (9)
 지역_위치별(주유소) (8)
 지역_위치별(주유소) (7)
 지역_위치별(주유소) (6)
 지역_위치별(주유소) (5)
 지역_위치별(주유소) (4)
 지역_위치별(주유소) (3)
 지역_위치별(주유소) (2)
 지역_위치별(주유소) (1)

데이터 크롤링 실습

구별 주유 가격 정리

```
import pandas as pd
from glob import glob
```

```
# station_files 변수에 각 엑셀 파일의 경로와 이름을 리스트로 저장
stations_files = glob('data/지역*.xls')
stations_files
```

```
# concat 명령으로 합쳐본다.
tmp_raw = []
```

```
for file_name in stations_files:
    tmp = pd.read_excel(file_name, header=2)
    tmp_raw.append(tmp)
```

```
station_raw = pd.concat(tmp_raw)
```

```
station_raw.info()
```

```
['data\\\\\\\\지역_위치별(주유소) (1).xls',
 'data\\\\\\\\지역_위치별(주유소) (10).xls',
 'data\\\\\\\\지역_위치별(주유소) (11).xls',
 'data\\\\\\\\지역_위치별(주유소) (12).xls',
 'data\\\\\\\\지역_위치별(주유소) (13).xls',
 'data\\\\\\\\지역_위치별(주유소) (14).xls',
 'data\\\\\\\\지역_위치별(주유소) (15).xls',
 'data\\\\\\\\지역_위치별(주유소) (16).xls',
 'data\\\\\\\\지역_위치별(주유소) (17).xls',
 'data\\\\\\\\지역_위치별(주유소) (18).xls',
 'data\\\\\\\\지역_위치별(주유소) (19).xls',
 'data\\\\\\\\지역_위치별(주유소) (2).xls',
 'data\\\\\\\\지역_위치별(주유소) (20).xls',
 'data\\\\\\\\지역_위치별(주유소) (21).xls',
 'data\\\\\\\\지역_위치별(주유소) (22).xls',
 'data\\\\\\\\지역_위치별(주유소) (23).xls',
 'data\\\\\\\\지역_위치별(주유소) (24).xls',
 'data\\\\\\\\지역_위치별(주유소) (25).xls',
 'data\\\\\\\\지역_위치별(주유소) (26).xls',
 'data\\\\\\\\지역_위치별(주유소) (27).xls',
 'data\\\\\\\\지역_위치별(주유소) (28).xls',
 'data\\\\\\\\지역_위치별(주유소) (29).xls',
 'data\\\\\\\\지역_위치별(주유소) (30).xls',
 'data\\\\\\\\지역_위치별(주유소) (31).xls',
 'data\\\\\\\\지역_위치별(주유소) (32).xls',
 'data\\\\\\\\지역_위치별(주유소) (33).xls',
 'data\\\\\\\\지역_위치별(주유소) (34).xls',
 'data\\\\\\\\지역_위치별(주유소) (35).xls',
 'data\\\\\\\\지역_위치별(주유소) (36).xls',
 'data\\\\\\\\지역_위치별(주유소) (37).xls',
 'data\\\\\\\\지역_위치별(주유소) (38).xls',
 'data\\\\\\\\지역_위치별(주유소) (39).xls',
 'data\\\\\\\\지역_위치별(주유소) (40).xls',
 'data\\\\\\\\지역_위치별(주유소) (41).xls',
 'data\\\\\\\\지역_위치별(주유소) (42).xls',
 'data\\\\\\\\지역_위치별(주유소) (43).xls',
 'data\\\\\\\\지역_위치별(주유소) (44).xls',
 'data\\\\\\\\지역_위치별(주유소) (45).xls',
 'data\\\\\\\\지역_위치별(주유소) (46).xls',
 'data\\\\\\\\지역_위치별(주유소) (47).xls',
 'data\\\\\\\\지역_위치별(주유소) (48).xls',
 'data\\\\\\\\지역_위치별(주유소) (49).xls',
 'data\\\\\\\\지역_위치별(주유소) (50).xls',
 'data\\\\\\\\지역_위치별(주유소) (51).xls',
 'data\\\\\\\\지역_위치별(주유소) (52).xls',
 'data\\\\\\\\지역_위치별(주유소) (53).xls',
 'data\\\\\\\\지역_위치별(주유소) (54).xls',
 'data\\\\\\\\지역_위치별(주유소) (55).xls',
 'data\\\\\\\\지역_위치별(주유소) (56).xls',
 'data\\\\\\\\지역_위치별(주유소) (57).xls',
 'data\\\\\\\\지역_위치별(주유소) (58).xls',
 'data\\\\\\\\지역_위치별(주유소) (59).xls',
 'data\\\\\\\\지역_위치별(주유소) (60).xls',
 'data\\\\\\\\지역_위치별(주유소) (61).xls',
 'data\\\\\\\\지역_위치별(주유소) (62).xls',
 'data\\\\\\\\지역_위치별(주유소) (63).xls',
 'data\\\\\\\\지역_위치별(주유소) (64).xls',
 'data\\\\\\\\지역_위치별(주유소) (65).xls',
 'data\\\\\\\\지역_위치별(주유소) (66).xls',
 'data\\\\\\\\지역_위치별(주유소) (67).xls',
 'data\\\\\\\\지역_위치별(주유소) (68).xls',
 'data\\\\\\\\지역_위치별(주유소) (69).xls',
 'data\\\\\\\\지역_위치별(주유소) (70).xls',
 'data\\\\\\\\지역_위치별(주유소) (71).xls',
 'data\\\\\\\\지역_위치별(주유소) (72).xls',
 'data\\\\\\\\지역_위치별(주유소) (73).xls',
 'data\\\\\\\\지역_위치별(주유소) (74).xls',
 'data\\\\\\\\지역_위치별(주유소) (75).xls',
 'data\\\\\\\\지역_위치별(주유소) (76).xls',
 'data\\\\\\\\지역_위치별(주유소) (77).xls',
 'data\\\\\\\\지역_위치별(주유소) (78).xls',
 'data\\\\\\\\지역_위치별(주유소) (79).xls',
 'data\\\\\\\\지역_위치별(주유소) (80).xls',
 'data\\\\\\\\지역_위치별(주유소) (81).xls',
 'data\\\\\\\\지역_위치별(주유소) (82).xls',
 'data\\\\\\\\지역_위치별(주유소) (83).xls',
 'data\\\\\\\\지역_위치별(주유소) (84).xls',
 'data\\\\\\\\지역_위치별(주유소) (85).xls',
 'data\\\\\\\\지역_위치별(주유소) (86).xls',
 'data\\\\\\\\지역_위치별(주유소) (87).xls',
 'data\\\\\\\\지역_위치별(주유소) (88).xls',
 'data\\\\\\\\지역_위치별(주유소) (89).xls',
 'data\\\\\\\\지역_위치별(주유소) (90).xls',
 'data\\\\\\\\지역_위치별(주유소) (91).xls',
 'data\\\\\\\\지역_위치별(주유소) (92).xls',
 'data\\\\\\\\지역_위치별(주유소) (93).xls',
 'data\\\\\\\\지역_위치별(주유소) (94).xls',
 'data\\\\\\\\지역_위치별(주유소) (95).xls',
 'data\\\\\\\\지역_위치별(주유소) (96).xls',
 'data\\\\\\\\지역_위치별(주유소) (97).xls',
 'data\\\\\\\\지역_위치별(주유소) (98).xls',
 'data\\\\\\\\지역_위치별(주유소) (99).xls',
 'data\\\\\\\\지역_위치별(주유소) (100).xls']
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 485 entries, 0 to 39
Data columns (total 10 columns):
#   Column   Non-Null Count  Dtype
---  ---
0   지역     485 non-null    object
1   상호     485 non-null    object
2   주소     485 non-null    object
3   상표     485 non-null    object
4   전화번호  485 non-null    object
5   셀프여부 485 non-null    object
6   고급취발유 485 non-null    object
7   취발유   485 non-null    object
8   경유     485 non-null    object
9   실내등유 485 non-null    object
dtypes: object(10)
memory usage: 41.7+ KB
```

데이터 크롤링 실습

구별 주유 가격 정리

station_raw.head()

	지역	상호	주소	상표	전화번호	셀프여부	고급휘발유	휘발유	경유	실내등유
0	서울특별시	재건에너지 재정제2주유소 고속셀프지점	서울특별시 강동구 천호대로 1246 (둔촌제2동)	현대오일뱅크	02-487-2030	Y	-	1499	1299	-
1	서울특별시	구천면주유소	서울 강동구 구천면로 357 (암사동)	현대오일뱅크	02-441-0536	N	-	1593	1397	-
2	서울특별시	지에스칼텍스(주) 동서울주유소	서울 강동구 천호대로 1456 (상일동)	GS칼텍스	02-426-5372	Y	-	1615	1415	-
3	서울특별시	(주)퍼스트오일 코알라주유소	서울특별시 강동구 올림픽로 556 (성내동)	S-OIL	02-484-1162	Y	-	1618	1418	-
4	서울특별시	주)지유에너지직영 오렌지주유소	서울 강동구 성안로 102 (성내동)	SK에너지	02-484-6165	N	-	1624	1423	1100

휘발유 데이터 저장.

```
stations = pd.DataFrame({'Oil_store': station_raw['상호'],  
                        '주소': station_raw['주소'],  
                        '가격': station_raw['휘발유'],  
                        '셀프': station_raw['셀프여부'],  
                        '상표': station_raw['상표']  
                        })
```

stations.head()

	Oil_store	주소	가격	셀프	상표
0	재건에너지 재정제2주유소 고속셀프지점	서울특별시 강동구 천호대로 1246 (둔촌제2동)	1499	Y	현대오일뱅크
1	구천면주유소	서울 강동구 구천면로 357 (암사동)	1593	N	현대오일뱅크
2	지에스칼텍스(주) 동서울주유소	서울 강동구 천호대로 1456 (상일동)	1615	Y	GS칼텍스
3	(주)퍼스트오일 코알라주유소	서울특별시 강동구 올림픽로 556 (성내동)	1618	Y	S-OIL
4	주)지유에너지직영 오렌지주유소	서울 강동구 성안로 102 (성내동)	1624	N	SK에너지

데이터 크롤링 실습

구별 주유 가격 정리

```
# 구 이름만 추출
```

```
stations['구'] = [eachAddress.split()[1] for eachAddress in stations['주소']]
stations.head()
```

	Oil_store	주소	가격	셀프	상표	구
0	재건에너지 재정제2주유소 고속셀프지점	서울특별시 강동구 천호대로 1246 (둔촌제2동)	1499	Y	현대오일뱅크	강동구
1	구천면주유소	서울 강동구 구천면로 357 (암사동)	1593	N	현대오일뱅크	강동구
2	지에스칼텍스(주) 동서울주유소	서울 강동구 천호대로 1456 (상일동)	1615	Y	GS칼텍스	강동구
3	(주)퍼스트오일 코알라주유소	서울특별시 강동구 올림픽로 556 (성내동)	1618	Y	S-OIL	강동구
4	주)지유에너지직영 오렌지주유소	서울 강동구 성안로 102 (성내동)	1624	N	SK에너지	강동구

```
# unique() 이용해서 데이터 검사 수행
```

```
stations['구'].unique()
```

```
stations['가격'].unique() #가격에 '-'가 있음
```

```
array([1499, 1593, 1615, 1618, 1624, 1628, 1629, 1635, 1638, 1649, 1677,
       1698, 1708, 2048, 1495, 1514, 1517, 1519, 1525, 1529, 1545, 1549,
       1568, 1577, 1588, 1608, 1650, 1795, 1539, 1578, 1579, 1599, 1619,
       1675, '1528', '1557', '1567', '1598', '1608', '1616', '1628',
       '1658', '1713', '1799', '1847', '-', 1497, 1509, 1534, 1535, 1548,
       1558, '1484', '1573', '1575', '1577', '1585', '1586', '1589',
```

데이터 크롤링 실습

■ 서울 주유소 가격 정보 비교

```
#가격에 '-' 값만 추출
stations[stations['가격']=='-']
```

	Oil_store	주소	가격	셀프	상표	구
12	서강주유소	서울 마포구 독막로 134 (창전동)	-	N	SK에너지	마포구
34	전당앞주유소	서울 서초구 남부순환로 2391 (서초동)	-	N	SK에너지	서초구
28	현대오일뱅크㈜직영 대일셀프주유소	서울 영등포구 영등포로 168	-	Y	현대오일뱅크	영등포구
16	삼육주유소	서울 은평구 수색로 299 (수색동)	-	N	SK에너지	은평구
10	에스씨(주) 역전주유소	서울 중구 퇴계로 15	-	N	GS칼텍스	중구
12	지에스칼텍스㈜ 소망주유소	서울 중랑구 망우로 475	-	Y	GS칼텍스	중랑구
13	재원에너지㈜ 범아주유소	서울 중랑구 동일로 881 (목동)	-	N	S-OIL	중랑구
14	현대오일뱅크㈜직영 대명셀프주유소	서울 광진구 광나루로 460 (화양동)	-	Y	현대오일뱅크	광진구

```
# '-' 문자가 포함된 데이터 제외
stations = stations[stations['가격'] != '-']
stations.head()
```

	Oil_store	주소	가격	셀프	상표	구
0	재건에너지 재정제2주유소 고속셀프지점	서울특별시 강동구 천호대로 1246 (둔촌제2동)	1499	Y	현대오일뱅크	강동구
1	구천면주유소	서울 강동구 구천면로 357 (암사동)	1593	N	현대오일뱅크	강동구
2	지에스칼텍스㈜ 동서울주유소	서울 강동구 천호대로 1456 (상일동)	1615	Y	GS칼텍스	강동구
3	(주)퍼스트오일 코알라주유소	서울특별시 강동구 올림픽로 556 (성내동)	1618	Y	S-OIL	강동구
4	주)지유에너지직영 오렌지주유소	서울 강동구 성안로 102 (성내동)	1624	N	SK에너지	강동구

데이터 크롤링 실습

▪ 서울 주유소 가격 정보 비교

```
# 가격 float 형 변환.  
stations['가격'] = [float(value) for value in  
stations['가격']]
```

```
# reset_index 이용하여 index 재정의  
stations.reset_index(inplace=True)  
del stations['index']# 기존 인덱스 삭제
```

```
stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 473 entries, 0 to 472  
Data columns (total 6 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   Oil_store   473 non-null    object  
1   주소        473 non-null    object  
2   가격        473 non-null    float64  
3   셀프        473 non-null    object  
4   상표        473 non-null    object  
5   구          473 non-null    object  
dtypes: float64(1), object(5)  
memory usage: 22.3+ KB
```


데이터 크롤링 실습

■ 시각화

```
# 한글문제 해결
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

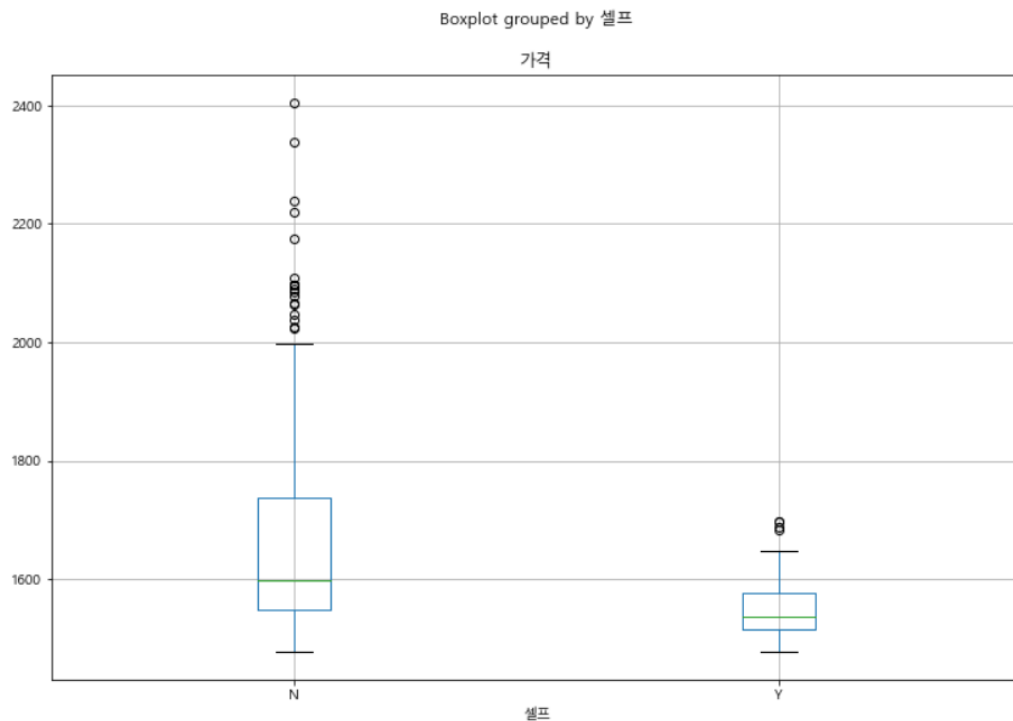
import platform

path = 'c:/Windows/Fonts/malgun.ttf'
from matplotlib import font_manager, rc
if platform.system() == 'Darwin':
    rc('font', family = 'AppleGothic')
elif platform.system() == 'Windows':
    font_name =
font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
else:
    print('Unknown system... sorry~~~')
```

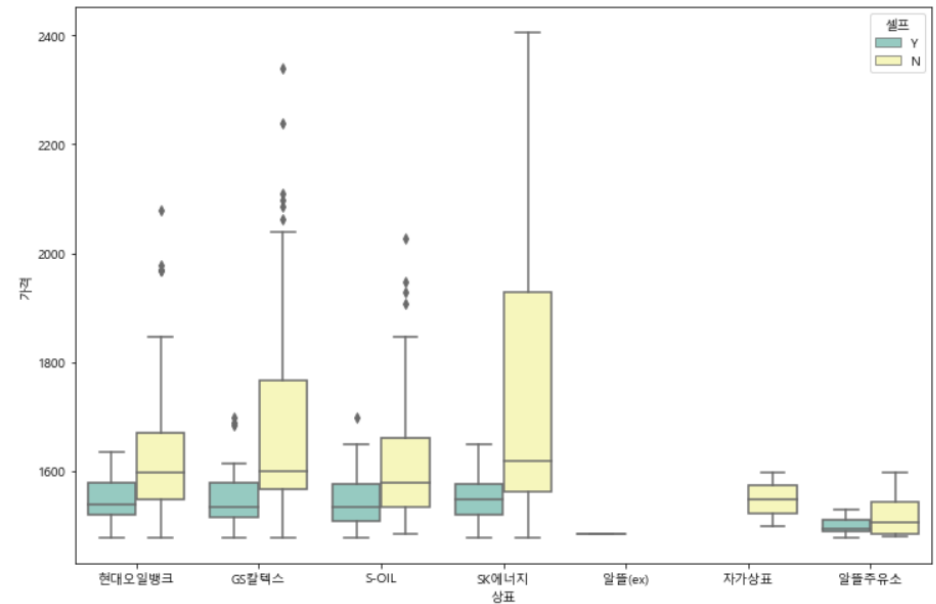
데이터 크롤링 실습

■ 시각화

```
stations.boxplot(column='가격', by='셀프', figsize=(12,8))
```



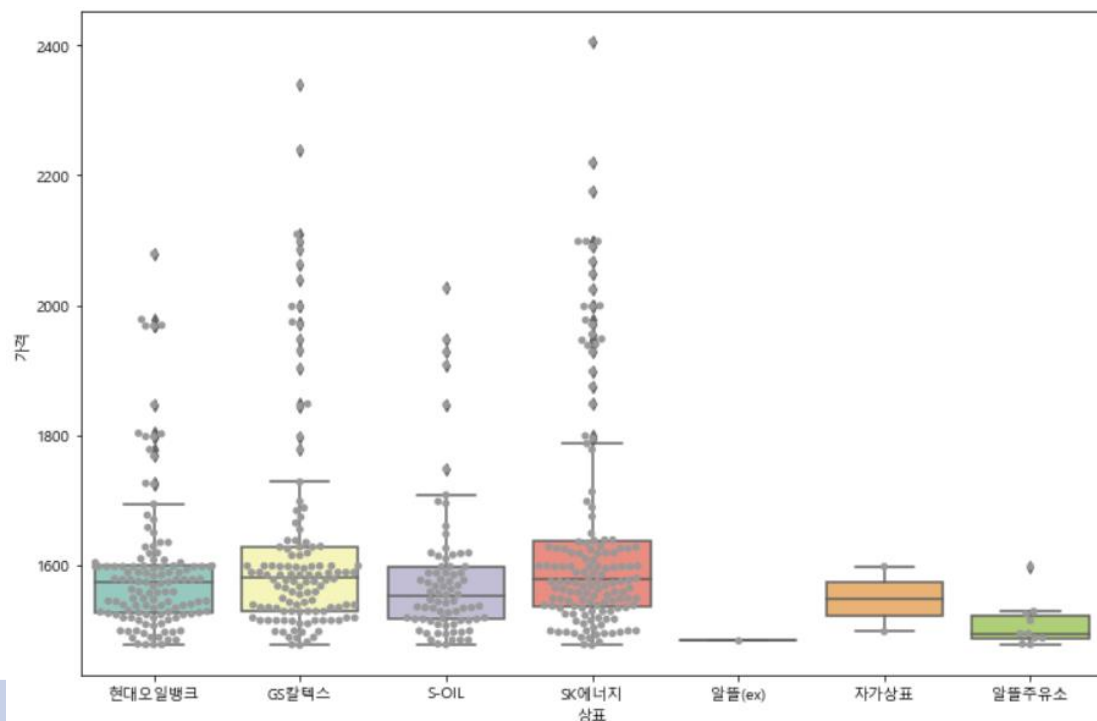
```
plt.figure(figsize=(12,8))  
sns.boxplot(x='상표', y='가격', hue='셀프', data=stations, palette='Set3')  
plt.show()
```



데이터 크롤링 실습

■ 시각화

```
plt.figure(figsize=(12,8))
sns.boxplot(x='상표', y='가격', data=stations, palette='Set3')
sns.swarmplot(x='상표', y='가격', data=stations, color='.6')
plt.show()
```



```
import json
import folium
import googlemaps
# 이제 서울시에서 가장 주유 가격이 비싼 주유소
stations.sort_values(by='가격', ascending=False).head(10)
```

	Oil_store	주소	가격	셀프	상표	구
293	서남주유소	서울 중구 통일로 30	2405.0	N	SK에너지	중구
258	서계주유소	서울 용산구 청파로 367 (서계동)	2339.0	N	GS칼텍스	용산구
292	필동주유소	서울 중구 퇴계로 196 (필동2가)	2239.0	N	GS칼텍스	중구
291	SK에너지(주) 퇴계로주유소	서울 중구 퇴계로 228 (필동2가)	2219.0	N	SK에너지	중구
472	뉴서울(강남)	서울 강남구 언주로 716	2175.0	N	SK에너지	강남구

```
# 서울시에서 가장 주유 가격이 싼 주유소
stations.sort_values(by='가격', ascending=True).head(10)
```

	Oil_store	주소	가격	셀프	상표	구
369	(주)한미석유구로그린주유소	서울 구로구 구로중앙로 76 (구로동)	1477.0	Y	GS칼텍스	구로구
370	구인주유소	서울 구로구 경인로 558 (구로동)	1477.0	N	SK에너지	구로구
210	영등포제일셀프주유소	서울 영등포구 가마산로 379	1478.0	Y	현대오일뱅크	영등포구
205	남서울고속주유소	서울 영등포구 가마산로 367 (대림동)	1478.0	Y	SK에너지	영등포구
206	(주)강서오일	서울 영등포구 도신로 151	1478.0	N	현대오일뱅크	영등포구

데이터 크롤링 실습

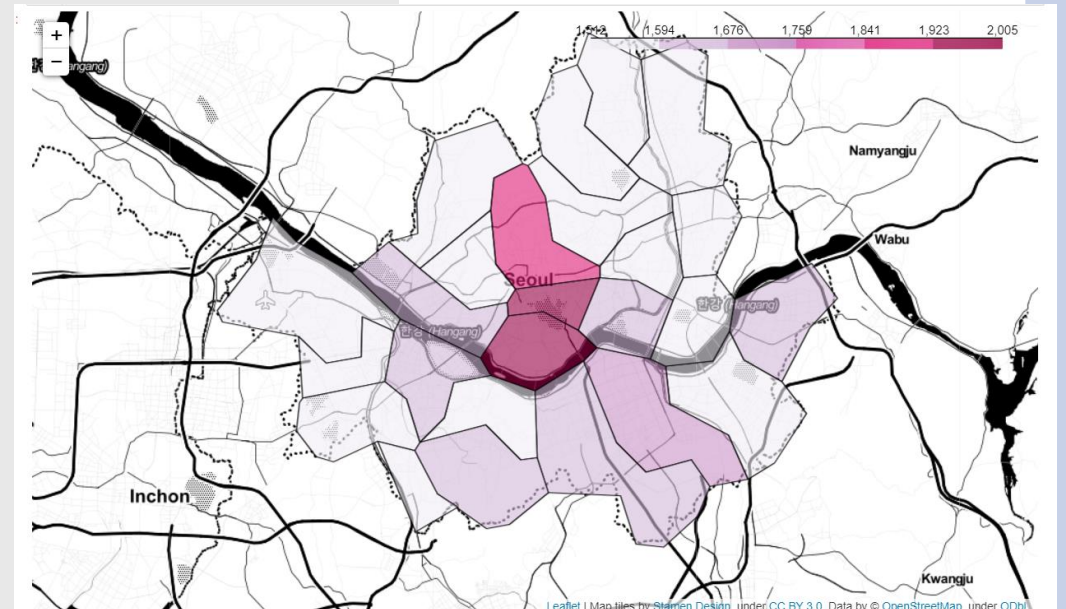
■ 시각화

```
# pivot_table을 이용해서 구별 가격 정보로 변경하고 가격 평균값 정리.  
import numpy as np  
gu_data = pd.pivot_table(stations, index=['구'], values=['가격'], aggfunc=np.mean)  
gu_data.head()
```

가격	
구	
강남구	1745.236842
강동구	1658.250000
강북구	1512.230769
강서구	1569.727273
관악구	1596.312500

```
# 서울시 구별 정보에 대해 지도로 표현  
geo_path = 'data/skorea_municipalities_geo_simple.json'  
geo_str = json.load(open(geo_path, encoding='utf-8'))  
  
map = folium.Map(location=[37.5502, 126.982], zoom_start=10.5,  
                  tiles='Stamen Toner')  
  
map.choropleth(geo_data = geo_str,  
               data = gu_data,  
               columns=[gu_data.index, '가격'],  
               fill_color='PuRd', #PuRd, YlGnBu  
               key_on='feature.id')
```

map



데이터 크롤링 실습

■ 시각화

```
# 주유 가격 상위 10개 주소 oil_price_top10 저장.  
oil_price_top10 = stations.sort_values(by='가격', ascending=False).head(10)  
oil_price_top10
```

	Oil_store	주소	가격	셀프	상표	구
293	서남주유소	서울 중구 통일로 30	2405.0	N	SK에너지	중구
258	서계주유소	서울 용산구 청파로 367 (서계동)	2339.0	N	GS칼텍스	용산구
292	필동주유소	서울 중구 퇴계로 196 (필동2가)	2239.0	N	GS칼텍스	중구
291	SK에너지(주) 퇴계로주유소	서울 중구 퇴계로 228 (필동2가)	2219.0	N	SK에너지	중구
472	뉴서울(강남)	서울 강남구 언주로 716	2175.0	N	SK에너지	강남구

```
# 하위 10개 oil_price_bottom10 저장  
oil_price_bottom10 = stations.sort_values(by='가격',  
ascending=True).head(10)  
oil_price_bottom10
```

	Oil_store	주소	가격	셀프	상표	구
369	(주)한미석유구로그린주유소	서울 구로구 구로중앙로 76 (구로동)	1477.0	Y	GS칼텍스	구로구
370	구인주유소	서울 구로구 경인로 558 (구로동)	1477.0	N	SK에너지	구로구
210	영등포제일셀프주유소	서울 영등포구 가마산로 379	1478.0	Y	현대오일뱅크	영등포구
205	남서울고속주유소	서울 영등포구 가마산로 367 (대림동)	1478.0	Y	SK에너지	영등포구
206	(주)강서오일	서울 영등포구 도신로 151	1478.0	N	현대오일뱅크	영등포구

데이터 크롤링 실습

```
# google maps API용 개인 key 입력
gmap_key = '*****'
gmaps = googlemaps.Client(key=gmap_key)

from tqdm import tqdm_notebook

lat = []
lng = []

for n in tqdm_notebook(oil_price_top10.index):
    try:
        tmp_add = str(oil_price_top10['주소'][n]).split('(')[0]
        tmp_map = gmaps.geocode(tmp_add)

        tmp_loc = tmp_map[0].get('geometry')
        lat.append(tmp_loc['location']['lat'])
        lng.append(tmp_loc['location']['lng'])

    except:
        lat.append(np.nan)
        lng.append(np.nan)
        print('Here is nan !')

oil_price_top10['lat'] = lat
oil_price_top10['lng'] = lng
oil_price_top10
```

	Oil_store	주소	가격	셀프	상표	구	lat	lng
293	서남주유소	서울 중구 통일로 30	2405.0	N	SK에너지	중구	37.558348	126.972090
258	서계주유소	서울 용산구 청파로 367 (서계동)	2339.0	N	GS칼텍스	용산구	37.552290	126.968935
292	필동주유소	서울 중구 퇴계로 196 (필동2가)	2239.0	N	GS칼텍스	중구	37.560850	126.993653
291	SK에너지(주) 퇴계로주유소	서울 중구 퇴계로 228 (필동2가)	2219.0	N	SK에너지	중구	37.561648	126.997142
472	뉴서울(강남)	서울 강남구 연주로 716	2175.0	N	SK에너지	강남구	37.517636	127.035756
290	약수주유소	서울 중구 다산로 173	2109.0	N	GS칼텍스	중구	37.559009	127.012663
256	한남지점	서울 용산구 한남대로21길 4 (한남동)	2098.0	N	SK에너지	용산구	37.534657	127.006063
257	에너지비스	서울 용산구 한남대로 82 (한남동)	2098.0	N	SK에너지	용산구	37.535952	127.006130
282	(주)중앙에너지비스 혜화주유소	서울 종로구 창경궁로35길 1	2098.0	N	SK에너지	종로구	37.586068	127.001058
283	(주)대양씨앤씨 사직주유소	서울 종로구 사직로 65 (사직동)	2098.0	N	GS칼텍스	종로구	37.574464	126.966618

데이터 크롤링 실습

```
lat = []
lng = []

for n in tqdm_notebook(oil_price_bottom10.index):
    try:
        tmp_add = str(oil_price_bottom10['주소'][n]).split('(')[0]
        tmp_map = gmaps.geocode(tmp_add)

        tmp_loc = tmp_map[0].get('geometry')
        lat.append(tmp_loc['location']['lat'])
        lng.append(tmp_loc['location']['lng'])

    except:
        lat.append(np.nan)
        lng.append(np.nan)
        print('Here is nan !')

oil_price_bottom10['lat']=lat
oil_price_bottom10['lng']=lng
oil_price_bottom10
```

	Oil_store	주소	가격	셀프	상표	구	lat	lng
369	(주)한미석유구로그린주유소	서울 구로구 구로중앙로 76 (구로동)	1477.0	Y	GS칼텍스	구로구	37.496223	126.888553
370	구인주유소	서울 구로구 경인로 558 (구로동)	1477.0	N	SK에너지	구로구	37.502491	126.879767
210	영등포제일셀프주유소	서울 영등포구 가마산로 379	1478.0	Y	현대오일뱅크	영등포구	37.502362	126.899452
205	남서울고속주유소	서울 영등포구 가마산로 367 (대림동)	1478.0	Y	SK에너지	영등포구	37.501567	126.898791
206	(주)강서오일	서울 영등포구 도신로 151	1478.0	N	현대오일뱅크	영등포구	37.509969	126.908231
211	(주)대청에너지 대청주유소	서울 영등포구 가마산로 328 (대림동)	1478.0	N	GS칼텍스	영등포구	37.498556	126.895791
207	도림주유소	서울 영등포구 도림로 343 (도림동)	1478.0	Y	알뜰주유소	영등포구	37.507656	126.900191
209	(주)MS주유소	서울 영등포구 대림로 230	1478.0	N	현대오일뱅크	영등포구	37.501330	126.897403
208	성락주유소	서울 영등포구 가마산로 414 (신길동)	1478.0	Y	S-OIL	영등포구	37.503750	126.902823
124	원천주유소	서울 성북구 돌곶이로 142 (장위동)	1479.0	N	알뜰주유소	성북구	37.614921	127.052752

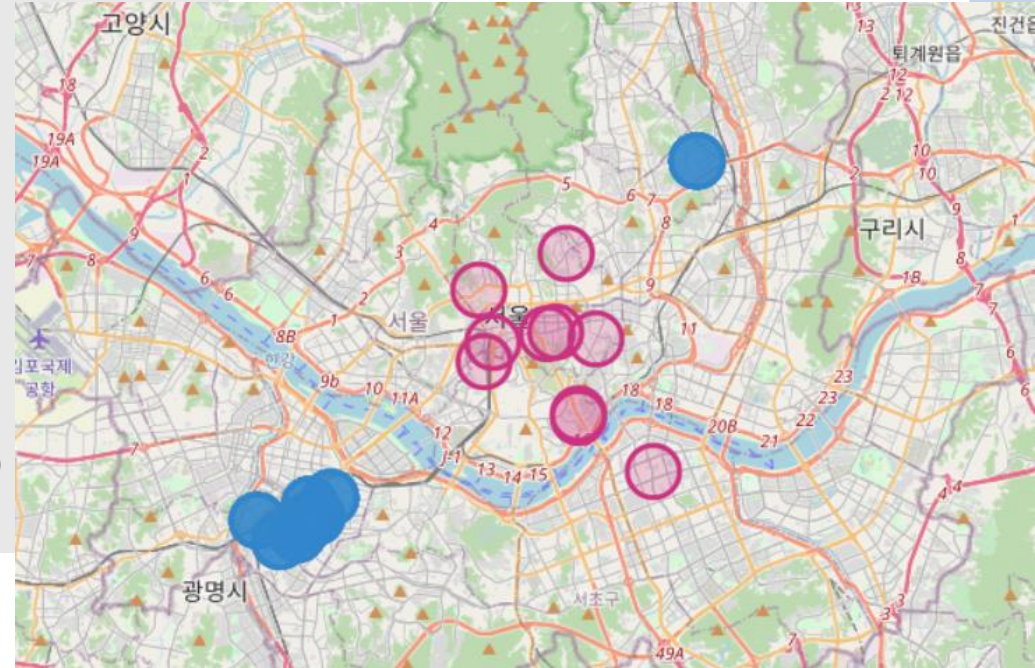
데이터 크롤링 실습

```
map = folium.Map(location=[37.5202, 126.975], zoom_start=10.5)

for n in oil_price_top10.index:
    if pd.notnull(oil_price_top10['lat'][n]):
        folium.CircleMarker([oil_price_top10['lat'][n],
                              oil_price_top10['lng'][n]],
                              radius=15, color='#CD3181',
                              fill_color='#CD3181').add_to(map)

for n in oil_price_bottom10.index:
    if pd.notnull(oil_price_bottom10['lat'][n]):
        folium.CircleMarker([oil_price_bottom10['lat'][n],
                              oil_price_bottom10['lng'][n]],
                              radius=15, color='#3186cc',
                              fill_color='#3186cc').add_to(map)
```

map



감사합니다
