

01

## 데이터 크롤링

01-1

## 데이터 크롤링

## 크롤링

- 자신의 원하는 정보를 웹페이지에서 가져오는 행위 또는 작업

# 크롤링 예시

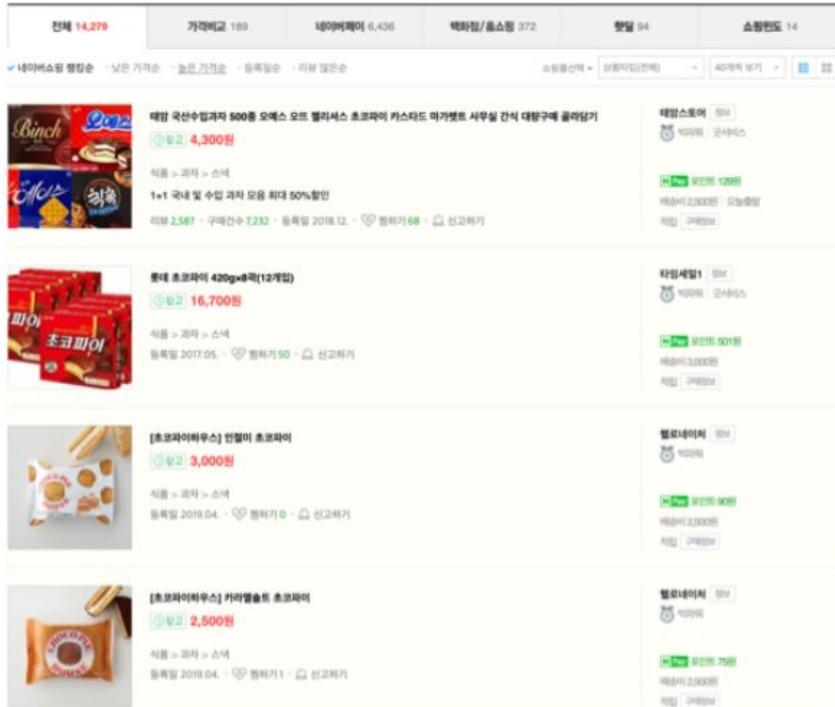
- 구글의 크롤링 봇이 전세계의 웹사이트를 돌며 주기적으로 크롤링을 하여 검색엔진에 데이터를 집어 넣는 것

1. abc.com  
2. bcd.or.kr  
3. abcde.com  
4. test.com



# 크롤링 예시

- 다른 소호 마켓의 판매 크롤링해서 출력



대한 국산수입과자 500종 오레스 오드 블리서스 초코파이 카스터드 마가렛 시루실 간식 대량구매 물리답기

최고 4,300원

식품 > 과자 > 스낵

1+1 국내 및 수입 과자 모음 최대 50%할인

리뷰 2,587 · 구매건수 2,332 · 등록일 2018.12 · 팔하기 68 · 신고하기

롯데 초코파이 420g\*8팩(12개입)

최고 16,700원

식품 > 과자 > 스낵

등록일 2017.05 · 팔하기 50 · 신고하기

[초코파이하우스] 한정미 초코파이

최고 3,000원

식품 > 과자 > 스낵

등록일 2018.04 · 팔하기 0 · 신고하기

[초코파이하우스] 카라멜솔트 초코파이

최고 2,500원

식품 > 과자 > 스낵

등록일 2018.04 · 팔하기 1 · 신고하기

대형스토어

최고 4,300원

식품 > 과자 > 스낵

등록일 2018.04 · 팔하기 1208 · 신고하기

롯데

최고 16,700원

식품 > 과자 > 스낵

등록일 2017.05 · 팔하기 50 · 신고하기

롯데

최고 3,000원

식품 > 과자 > 스낵

등록일 2018.04 · 팔하기 0 · 신고하기

롯데

최고 2,500원

식품 > 과자 > 스낵

등록일 2018.04 · 팔하기 1 · 신고하기

## 크롤링 흐름

1. 크롤링 도구를 가져옵니다.
2. 필요한 정보가 있는 웹사이트에 접속합니다.
3. 필요한 정보가 있는 페이지나 구역을 선택합니다.
4. 선택한 구역이나 페이지의 데이터를 가져옵니다.
5. 가져온 데이터에서 필요한 정보만을 저장 또는 사용합니다.

# BeautifulSoup로 스크레이핑하기

## ■ BeautifulSoup로 스크레이핑하기

- HTML 페이지에서 원하는 정보를 추출

### □ BeautifulSoup 설치

`pip install beautifulSoup4`

## ■ HTML

□ Hyper Text MarkUp Langauge의 약자로 웹문서를 기술하는 언어

- 웹 문서를 표현하기 위한 태그들로 구성

□ HTML Tag

- <tag name> □ 여는 태그(시작)
- </tag name> □ 닫는 태그(종료)

□ HTML 문서 구조

```
<HTML>
<HEAD>

</HEAD>

<BODY>

</BODY>

</HTML>
```

## ■ 웹에서 정보를 추출하는 방법

- 파이썬은 웹 사이트로부터 데이터를 추출하기 위해 "urllib" 패키지 활용.
  - HTTP나 FTP를 사용해서 데이터를 다운로드할 수 있음.
  - urllib.request 모듈은 웹사이트에 있는 데이터에 접근하는 가능 제공

## ■ 네이버 메인 웹페이지 가져오기

```
import urllib.request
url = "https://www.naver.com/"
html = urllib.request.urlopen(url)

print( html.read() )
```

## ■ urllib.request를 이용한 이미지 다운로드

```
import urllib.request
url = "http://uta.pw/shodou/img/28/214.png"
filename = "test.png"
urllib.request.urlretrieve(url, filename)
print("saved!")
```

# urllib.request를 이용한 이미지 다운로드

## urllib.request를 이용한 이미지 다운로드

```
import urllib.request
url = "https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png"
filename = "google.png"
urllib.request.urlretrieve(url, filename)
print("saved!")
```

- ▶ 다운로드 받은 이미지를 메모리에 저장하지 않고 바로 하드디스크에 저장

## urllib.request.urlopen()으로 이미지를 파일에 저장

```
import urllib.request
url = "https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png"
filename = "google.png"
mem = urllib.request.urlopen(url).read()
with open(filename, mode="wb") as f:
    f.write(mem)
    print("saved!")
```

- ▶ 다운로드 받아 "mem" 변수에 저장 후 하드디스크에 저장

## ■ BeautifulSoup로 스크레이핑하기

- HTML 페이지에서 원하는 부분을 추출

### ▶ 예제

```
# 라이브러리 읽어 들이기 --- (※1)
from bs4 import BeautifulSoup
# 분석하고 싶은 HTML --- (※2)
html = """
<html><body>
  <h1>스크레이핑이란?</h1>
  <p>웹 페이지를 분석하는 것</p>
  <p>원하는 부분을 추출하는 것</p>
</body></html>
"""

```

```
# HTML 분석하기 --- (※3)
soup = BeautifulSoup(html, 'html.parser')
# 원하는 부분 추출하기 --- (※4)
h1 = soup.html.body.h1
p1 = soup.html.body.p
p2 = p1.next_sibling.next_sibling
# 요소의 글자 출력하기 --- (※5)
print("h1 = " + h1.string)
print("p  = " + p1.string)
print("p  = " + p2.string)
```

# BeautifulSoup로 스크레이핑하기

## ■ BeautifulSoup로 스크레이핑하기

- BeautifulSoup를 import
- 'html'이라는 객체에 문자열(html tag) 할당.

```
In [3]: from bs4 import BeautifulSoup
html = """
<html><body>
<h1>스크레이핑이란?</h1>
<p>웹 페이지를 분석하는 것</p>
<p>원하는 부분을 추출하는 것</p>
</body></html>
"""

```

- Html 객체를 BeautifulSoup( ) 함수에 전달 □ html 문서 정보를 가지고 있는 객체를 반환

```
In [5]: soup = BeautifulSoup(html, 'html.parser')
soup
Out [5]:
<html><body>
<h1>스크레이핑이란?</h1>
<p>웹 페이지를 분석하는 것</p>
<p>원하는 부분을 추출하는 것</p>
</body></html>
```

# BeautifulSoup로 스크레이핑하기

## ▶ soup.html.body

```
In [9]: soup.html.body
```

```
Out [9]: <body>
<h1>스크레이핑이란?</h1>
<p>웹 페이지를 분석하는 것</p>
<p>원하는 부분을 추출하는 것</p>
</body>
```

```
In [7]: h1 = soup.html.body.h1
h1
```

```
Out [7]: <h1>스크레이핑이란?</h1>
```

```
In [12]: soup.html.body.p
```

```
Out [12]: <p>웹 페이지를 분석하는 것</p>
```

```
# 라이브러리 읽어 들이기 --- (※1)
from bs4 import BeautifulSoup
# 분석하고 싶은 HTML --- (※2)
html = """
<html><body>
<h1>스크레이핑이란?</h1>
<p>웹 페이지를 분석하는 것</p>
<p>원하는 부분을 추출하는 것</p>
</body></html>
"""

```

```
# HTML 분석하기 --- (※3)
soup = BeautifulSoup(html, 'html.parser')
# 원하는 부분 추출하기 --- (※4)
h1 = soup.html.body.h1
p1 = soup.html.body.p
p2 = p1.next_sibling.next_sibling
# 요소의 글자 출력하기 --- (※5)
print("h1 = " + h1.string)
print("p = " + p1.string)
print("p = " + p2.string)
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기

### soup.html.body

```
In [7]: h1 = soup.html.body.h1  
h1
```

```
Out [7]: <h1>스크레이핑이란?</h1>
```

```
In [12]: soup.html.body.p
```

```
Out [12]: <p>웹 페이지를 분석하는 것</p>
```

```
In [16]: p1 = soup.html.body.p  
p1
```

```
Out [16]: <p>웹 페이지를 분석하는 것</p>
```

```
In [18]: p1.next_sibling
```

```
Out [18]: '#n'
```

```
In [21]: p1.next_sibling.next_sibling
```

```
Out [21]: <p>원하는 부분을 추출하는 것</p>
```

```
# 라이브러리 읽어 들이기 --- (※1)
```

```
from bs4 import BeautifulSoup
```

```
# 분석하고 싶은 HTML --- (※2)
```

```
html = """
```

```
<html><body>
```

```
<h1>스크레이핑이란?</h1>
```

```
<p>웹 페이지를 분석하는 것</p>
```

```
<p>원하는 부분을 추출하는 것</p>
```

```
</body></html>
```

```
"""
```

```
# HTML 분석하기 --- (※3)
```

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

```
# 원하는 부분 추출하기 --- (※4)
```

```
h1 = soup.html.body.h1
```

```
p1 = soup.html.body.p
```

```
p2 = p1.next_sibling.next_sibling
```

```
# 요소의 글자 출력하기 --- (※5)
```

```
print("h1 = " + h1.string)
```

```
print("p = " + p1.string)
```

```
print("p = " + p2.string)
```

# BeautifulSoup로 스크레이핑하기

## Code 분석

### ▶ (※1)

- BeautifulSoup 모듈 임포트

### ▶ (※3)

- Soup에는 HTML 전체가 저장됨.

### ▶ (※4)

- `h1 = soup.html.body.h1`

- » `h1`에는 `<h1>`과 `</h1>`이 감싸고 있는 내용 전체가 저장됨.

- » `print(h1) → "<h1>스크레이핑이란?</h1>"`

```
# 라이브러리 읽어 들이기 --- (※1)
from bs4 import BeautifulSoup
# 분석하고 싶은 HTML --- (※2)
html = """
<html><body>
<h1>스크레이핑이란?</h1>
<p>웹 페이지를 분석하는 것</p>
<p>원하는 부분을 추출하는 것</p>
</body></html>
"""

# HTML 분석하기 --- (※3)
# soup에는 html 코드 전체가 저장됨.
soup = BeautifulSoup(html, 'html.parser')
# 원하는 부분 추출하기 --- (※4)
h1 = soup.html.body.h1
p1 = soup.html.body.p
p2 = p1.next_sibling.next_sibling
# 요소의 글자 출력하기 --- (※5)
print("h1 = " + h1.string)
print("p = " + p1.string)
print("p = " + p2.string)
```

# BeautifulSoup로 스크레이핑하기

## ▶ id로 요소를 찾는 방법

```
from bs4 import BeautifulSoup

html = """
<html><body>
  <h1 id="title1">스크레이핑이란?</h1>
  <p id = "body1">웹 페이지를 분석하는 것</p>
  <p>원하는 부분을 추출하는 것</p>
  <h1 id="title2">스크레이핑이란?</h1>
  <p id = "body2">웹 페이지를 분석하는 것</p>
  <p>원하는 부분을 추출하는 것</p>
</body></html>

"""


```

```
# HTML 분석하기 --- (※1)
soup = BeautifulSoup(html, 'html.parser')
# find() 메서드로 원하는 부분 추출하기 --- (※2)
title = soup.find(id="title2")
body = soup.find(id="body2")
# 텍스트 부분 출력하기
print("#title=" + title.string)
print("#body=" + body.string)
```

# BeautifulSoup로 스크레이핑하기

## ■ 코드 설명

### ▶ print(soup)

- Html 문서 전체

### ▶ print(title)

- "<h1 id='title2'>우리가 추가한 문서</h1>"

```
from bs4 import BeautifulSoup

html = """
<html><body>
<h1 id="title1">스크레이핑이란?</h1>
<p id = "body1">웹 페이지를 분석하는 것</p>
<p>원하는 부분을 추출하는 것</p>
<h1 id="title2">우리가 추가한 문서</h1>
<p id = "body2">잘 해보자구요</p>
<p>끝까지 가는 거야~ ^~</p>
</body></html>
"""

# HTML 분석하기 --- (※1)
soup = BeautifulSoup(html, 'html.parser')
# find() 메서드로 원하는 부분 추출하기 --- (※2)
title = soup.find(id="title2")
body = soup.find(id="body2")
# 텍스트 부분 출력하기
print("#title=" + title.string)
print("#body=" + body.string)
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기 – `find( )`

```
from bs4 import BeautifulSoup
html = """
<html>
  <body>
    <ul class="greet">
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
    <ul class="reply">
      <li>ok</li>
      <li>no</li>
      <li>sure</li>
    </ul>
    <div>
      <ul>
        <li>open</li>
        <li>close</li>
      </ul>
    </div>
  </body>
</html>
"""
```

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

# BeautifulSoup을 이용한 데이터 추출
# print(type(soup))
soup

soup.find('div') #<div> ~ </div>의 내용 프린트
# <ul>~</ul> 패턴 하나만 찾는다.

soup.find('ul') #<ul> ~ </ul>의 내용 프린트
ls=soup.find('ul')
ls.text
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기 – `find()`

```
In [2]: from bs4 import BeautifulSoup
html = """
<html>
  <body>
    <ul class="greet">
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
    <ul class="reply">
      <li>ok</li>
      <li>no</li>
      <li>sure</li>
    </ul>
    <div>
      <ul>
        <li>open</li>
        <li>close</li>
      </ul>
    </div>
  </body>
</html>
"""
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기 – `find()`

```
In [5]: soup = BeautifulSoup(html, "html.parser")
soup
```

```
Out [5]:
<html>
<body>
<ul class="greet">
<li>hello</li>
<li>bye</li>
<li>welcome</li>
</ul>
<ul class="reply">
<li>ok</li>
<li>no</li>
<li>sure</li>
</ul>
<div>
<ul>
<li>open</li>
<li>close</li>
</ul>
</div>
</body>
</html>
```

```
In [8]: soup.find('div')
```

```
Out [8]:
<div>
<ul>
<li>open</li>
<li>close</li>
</ul>
</div>
```

```
In [11]: ls=soup.find('ul')
```

```
Out [11]:
ls
<ul class="greet">
<li>hello</li>
<li>bye</li>
<li>welcome</li>
</ul>
```

```
In [15]: ls.text
```

```
Out [15]:
'hello\nbye\nwelcome\n'
```

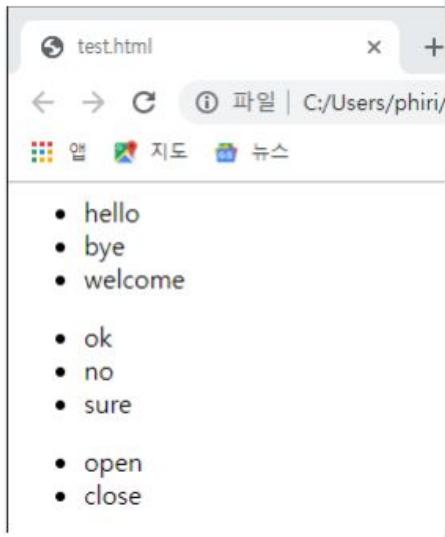
```
In [16]: print(ls.text)
```

```
hello
bye
welcome
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기 – **findAll( )**

```
# file name : "BeautifulSoupEx2.py"
from bs4 import BeautifulSoup
html = """
<html>
  <body>
    <ul class="greet">
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
    <ul class="reply">
      <li>ok</li>
      <li>no</li>
      <li>sure</li>
    </ul>
    <div>
      <ul>
        <li>open</li>
        <li>close</li>
      </ul>
    </div>
  </body>
</html>
"""
```



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

# findAll 함수로 텍스트 뽑아내기
# findAll() : 조건에 해당하는 모든 요소를 리스트[ ] 형태로 추출

lis = soup.findAll("li") # <li> ~ </li> 형태의 모두 찾는다.
#soup.findAll() 함수의 반환값은 list 형이다.

print( type(lis) )
print( lis )

for item in lis:
    print(item)

# 인덱스 값을 이용한 출력
print(lis[0].text)
print(lis[1].text)

for item in lis:
    print(item.text)
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기 – `findAll( )`

```
In [1]: from bs4 import BeautifulSoup
html = """<html><body><ul class="greet"><li>hello</li><li>bye</li><li>welcome</li></ul><ul class="reply"><li>ok</li><li>no</li><li>sure</li></ul><div><ul><li>open</li><li>close</li></ul></div></body></html>"""
```

```
In [7]: soup = BeautifulSoup(html, "html.parser")
type(soup)
```

```
Out [7]: bs4.BeautifulSoup
```

```
In [10]: lis = soup.findAll('li')
lis
```

```
Out [10]: [<li>hello</li>,
<li>bye</li>,
<li>welcome</li>,
<li>ok</li>,
<li>no</li>,
<li>sure</li>,
<li>open</li>,
<li>close</li>]
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기 – `findAll( )`

```
In [12]: for item in lis:  
    print(item)  
  
<li>hello</li>  
<li>bye</li>  
<li>welcome</li>  
<li>ok</li>  
<li>no</li>  
<li>sure</li>  
<li>open</li>  
<li>close</li>
```

```
In [16]: lis[0]  
Out [16]: <li>hello</li>
```

```
In [15]: lis[0].text  
Out [15]: 'hello'
```

```
In [17]: lis[1].text  
Out [17]: 'bye'
```

```
In [19]: for item in lis:  
    print(item.text)  
  
hello  
bye  
welcome  
ok  
no  
sure  
open  
close
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 스크레이핑하기 – `find('tag', {'class':'name'})`

▶ `ul_reply = soup.find("ul", {"class":"reply"})`

```
# file name : BeautifulSoupEx3.py
from bs4 import BeautifulSoup
html = """
<html>
  <body>
    <ul class="greet">
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
    <ul class="reply"> <-- green arrow
      <li>ok</li>
      <li>no</li>
      <li>sure</li>
    </ul> <-- orange arrow
  </body>
</html>
"""
```

```
soup = BeautifulSoup(html, "html.parser")
# ok no sure
ul_reply = soup.find( "ul", {"class":"reply"} )
lis = ul_reply.findAll("li")

for item in lis:
    print(item)

for item in lis:
    print(item.text)
```

# BeautifulSoup로 스크레이핑하기

## ■ 크롤링하려고 하는 웹 문서(html) 가져 오기

- ▶ 가져온 웹 문서를 BeautifulSoup로 스크레이핑하기

```
# BeautifulSoup 사용
import urllib.request
import bs4

url = "https://www.naver.com/"
html = urllib.request.urlopen(url)

soup = bs4.BeautifulSoup(html, "html.parser")
print(soup)
```

## BeautifulSoup로 스크레이핑하기

## 크롤링하려고 하는 웹 문서(html) 가져 오기

## ▶ 가져온 웹 문서를 BeautifulSoup로 스크레이핑하기

# BeautifulSoup로 스크레이핑하기

## ■ 크롤링하려고 하는 웹 문서(html) 가져 오기

### ▶ 가져온 웹 문서를 BeautifulSoup로 스크레이핑하기

```
In [15]: soup = bs4.BeautifulSoup(html, "html.parser")
print(soup)

<!DOCTYPE doctype html>

<html lang="ko">
<head>
<meta charset="utf-8"/>
<meta content="origin" name="Referrer"/>
<meta content="text/javascript" http-equiv="Content-Script-Type"/>
<meta content="text/css" http-equiv="Content-Style-Type"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta content="width=1100" name="viewport"/>
<meta content="NAVER" name="apple-mobile-web-app-title">
<meta content="index,nofollow" name="robots">
<meta content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요" name="description">
<meta content="네이버" property="og:title"/>
<meta content="https://www.naver.com/" property="og:url"/>
<meta content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png" property="og:image"/>
<meta content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요" property="og:description">
<meta content="summary" name="twitter:card"/>
<meta content="" name="twitter:title"/>
<meta content="https://www.naver.com/" name="twitter:url"/>
<meta content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png" name="twitter:image"/>
<meta content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요" name="twitter:description">
<meta content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요" name="twitter:label">
```

# BeautifulSoup로 스크레이핑하기

## ■ BeautifulSoup로 Naver 메인 웹페이지 추출하기

### □ 크롬으로 naver 접속

- 마우스 우클릭 후 '검사' 선택
- 검사 창 선택 후 Ctrl+F로 list\_nav type\_fix을 검색함.

# file name : BeautifulSoupEx4.py

```
import urllib.request import bs4

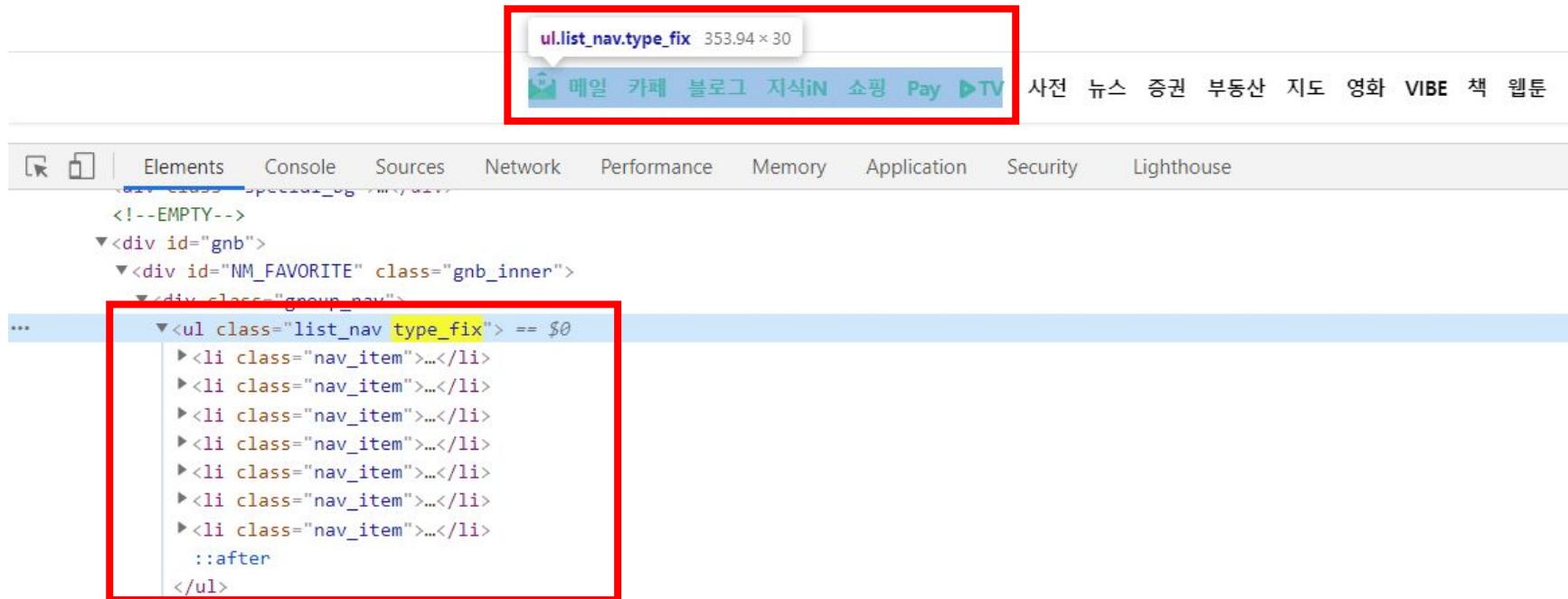
url = "https://www.naver.com/"
html = urllib.request.urlopen(url)

soup = bs4.BeautifulSoup(html, "html.parser")

# type_fix 클래스를 찾아 ul 값 뽑아내기
ul = soup.find("ul", {"class": "type_fix"})
print(ul)
```

```
▼<ul class="list_nav type_fix">
  ▶<li class="nav_item">...</li>
  ▶<li class="nav_item">...</li>
  ▶<li class="nav_item">...</li> (선택된 li 태그)
  ▶<li class="nav_item">...</li>
  ▶<li class="nav_item">...</li>
  ▶<li class="nav_item">...</li>
  ▶<li class="nav_item">...</li>
  ▶<li class="nav_item">...</li> == $0
    ::after
</ul>
```

# BeautifulSoup로 스크레이핑하기



The screenshot shows the browser developer tools with the 'Elements' tab selected. The page header is highlighted with a red box, showing the class 'ul.list\_nav.type\_fix' and a size of 353.94 x 30. Below the header, the 'Elements' tab shows the HTML structure of the navigation bar. A red box highlights the 'ul' element with the class 'list\_nav type\_fix' and its child 'li' elements, which are listed as 'nav\_item'.

```
<!--EMPTY-->
<div id="gnb">
  <div id="NM_FAVORITE" class="gnb_inner">
    <div class="group_nav">
      <ul class="list_nav type_fix"> == $0
        <li class="nav_item">...</li>
        ::after
      </ul>
    </div>
  </div>
</div>
```

# BeautifulSoup로 스크레이핑하기

```
▼<ul class="list_nav type_fix"> == $0
  ▼<li class="nav_item">
    ▼<a href="https://mail.naver.com/" class="nav" data-clk="svc.mail">
      <i class="ico_mail"></i>
      "메일"
    </a>
  </li>
  ▼<li class="nav_item">
    <a href="https://section.cafe.naver.com/" class="nav" data-clk="svc.cafe">카페</a>
  </li>
  ▼<li class="nav_item">
    <a href="https://section.blog.naver.com/" class="nav" data-clk="svc.blog">블로그</a>
  </li>
  ▼<li class="nav_item">
    <a href="https://kin.naver.com/" class="nav" data-clk="svc.kin">지식IN</a>
  </li>
  ▼<li class="nav_item">
    <a href="https://shopping.naver.com/" class="nav" data-clk="svc.shopping">쇼핑</a>
  </li>
  ▼<li class="nav_item">
    <a href="https://order.pay.naver.com/home" class="nav" data-clk="svc.pay">Pay</a>
  </li>
  ▼<li class="nav_item">
    ▶<a href="https://tv.naver.com/" class="nav" data-clk="svc.tvcast">...</a>
  </li>
  ::after
</ul>
```

# BeautifulSoup로 스크레이핑하기

## BeautifulSoup로 Naver 메인 웹페이지 추출하기

```
import urllib.request  
import bs4
```

```
url = "https://www.naver.com/"  
html = urllib.request.urlopen(url)  
soup = bs4.BeautifulSoup(html, "html.parser")
```

```
# type_fix 클래스를 찾아 ul 값 뽑아내기  
ul = soup.find("ul", {"class": "type_fix"})  
# print(ul)
```

```
# ul 값에서 li값 뽑아내기  
list = ul.findAll("li")  
print(list)
```

```
for li in list:  
    print(li.te  
        xt)
```

```
<ul class="list_nav type_fix"> == $0  
  <li class="nav_item">  
    <a href="https://mail.naver.com/" class="nav" data-clk="svc.mail">  
      <i class="ico_mail"></i>  
      "메일"  
    </a>  
  </li>  
  <li class="nav_item">  
    <a href="https://section.cafe.naver.com/" class="nav" data-clk="svc.cafe">카페</a>  
  </li>  
  <li class="nav_item">  
    <a href="https://section.blog.naver.com/" class="nav" data-clk="svc.blog">블로그</a>  
  </li>  
  <li class="nav_item">  
    <a href="https://kin.naver.com/" class="nav" data-clk="svc.kin">지식iN</a>  
  </li>  
  <li class="nav_item">  
    <a href="https://shopping.naver.com/" class="nav" data-clk="svc.shopping">쇼핑</a>  
  </li>  
  <li class="nav_item">  
    <a href="https://order.pay.naver.com/home" class="nav" data-clk="svc.pay">Pay</a>  
  </li>  
  <li class="nav_item">  
    <a href="https://tv.naver.com/" class="nav" data-clk="svc.tvcast">...</a>  
  </li>  
  ::after  
</ul>
```

# BeautifulSoup로 스크레이핑하기

```
In [1]: import urllib.request  
import bs4
```

```
In [2]: url = "https://www.naver.com/"  
html = urllib.request.urlopen(url)  
soup = bs4.BeautifulSoup(html, "html.parser")
```

```
In [6]: # type_fix 클래스를 찾아 ul 태그를 가져오기  
ul = soup.find("ul", {"class": "type_fix"})  
print (ul)  
  
<ul class="list_nav type_fix">  
  <li class="nav_item">  
    <a class="nav" data-clk="svc.mail" href="https://mail.naver.com/"><i class="ico_mail"></i>메일</a>  
  </li>  
  <li class="nav_item"><a class="nav" data-clk="svc.cafe" href="https://section.cafe.naver.com/">카페</a></li>  
  <li class="nav_item"><a class="nav" data-clk="svc.blog" href="https://section.blog.naver.com/">블로그</a></li>  
  <li class="nav_item"><a class="nav" data-clk="svc.kin" href="https://kin.naver.com/">지식IN</a></li>  
  <li class="nav_item"><a class="nav" data-clk="svc.shopping" href="https://shopping.naver.com/">쇼핑</a></li>  
  <li class="nav_item"><a class="nav" data-clk="svc.pay" href="https://order.pay.naver.com/home">Pay</a></li>  
  <li class="nav_item">  
    <a class="nav" data-clk="svc.tvcast" href="https://tv.naver.com/"><i class="ico_tv"></i>TV</a>  
  </li>  
</ul>
```

# BeautifulSoup로 스크레이핑하기

```
In [7]: # ul 밑에서 li를 뽑아내기
list = ul.findAll("li")
print(list)
```

```
[<li class="nav_item">
<a class="nav" data-clk="svc.mail" href="https://mail.naver.com/"><i class="ico_mail"></i>메일</a>
</li>, <li class="nav_item"><a class="nav" data-clk="svc.cafe" href="https://section.cafe.naver.com/">카페</a></li>, <li class="nav_item">
<a class="nav" data-clk="svc.blog" href="https://section.blog.naver.com/">블로그</a></li>, <li class="nav_item"><a class="nav" data-clk="svc.kin" href="https://kin.naver.com/">지식iN</a></li>, <li class="nav_item"><a class="nav" data-clk="svc.shopping" href="https://shopping.naver.com/">쇼핑</a></li>, <li class="nav_item"><a class="nav" data-clk="svc.pay" href="https://order.pay.naver.com/home">Pay</a></li>, <i class="nav_item">
<a class="nav" data-clk="svc.tvcast" href="https://tv.naver.com/"><i class="ico_tv"></i>TV</a>
</li>]
```

```
In [8]: for li in list:
    print(li.text)
```

메일

카페

블로그

지식iN

쇼핑

Pay

TV

# BeautifulSoup로 스크레이핑하기

## ■ 영화 평점 분석

□ 웹 페이지 :

<https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20210427>

```
In [48]: import urllib.request
from bs4 import BeautifulSoup

url ="http://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20190321"

wpage =urllib.request.urlopen( url )
soup =BeautifulSoup(wpage, 'html.parser')
```

# BeautifulSoup로 스크레이핑하기

## 영화 평점 분석

□ 웹 페이지 : <https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20210421>

```
import urllib.request as req
from bs4 import BeautifulSoup

url = "https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20190321"

html = req.urlopen(url)
soup = BeautifulSoup(html, 'html.parser')

titleOfMovie = soup.findAll('td', {"class": "title"})
titleOfMovie

titleOfMovie = soup.findAll('div', {"class": "tit5"})

for item in titleOfMovie:
    print(item.find('a').text)

points = soup.findAll('td', {"class": "point"})

for point in points:
    print(point.text)
```

# BeautifulSoup로 스크레이핑하기

## ■ 영화 평점 분석

### □ 포인트 추출

```
title = []
for item in titleOfMovie:
    title.append(item.find('a').text)
title

pointOfMovie = soup.findAll('td', {"class": "point"})

points = []
for p in pointOfMovie:
    points.append(p.text)
points

for i in range(0, len(title)):
    print(str(i+1) + ":" + title[i] + "(" + points[i] + ")")
```

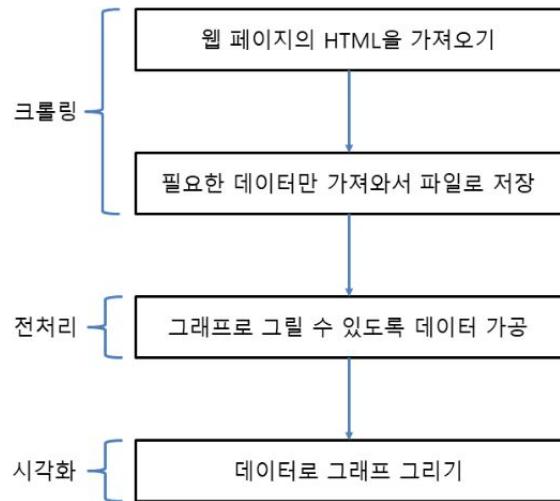
02

## 날씨 데이터 크롤링

# 날씨데이터 크롤링

- 클롤링한 날씨데이터로 그래프 그리기
  - 파이썬은 다양한 분야에서 사용하지만 요즘은 데이터 처리와 분석에 유용하게 사용하고 있음
  - 웹의 데이터를 가져온 뒤 그래프로 그리는 방법을 알아보자

## ▼ 크롤링, 전처리, 시각화 과정



## 클롤링한 날씨데이터로 그래프 그리기

- 클롤링한 날씨데이터로 그래프 그리기
  - 웹 페이지를 가져오고 파일로 저장하는 크롤링(crawling) 과정, 데이터를 가공하는 전처리 과정, 데이터로 그래프를 그리는 시각화 과정으로 구분할 수 있음

## 웹 페이지의 HTML을 가져와서 파일로 저장하기

- 웹 페이지의 HTML을 가져와서 파일로 저장하기
  - 기상청 웹 사이트에서 도시별 현재날씨 페이지의 HTML을 가져와보자

<http://www.weather.go.kr/weather/observation/currentweather.jsp>

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

## ▼ 기상청 도시별 현재날씨

The screenshot shows the 'Current Weather' section of the KMA website. The URL is [www.kma.go.kr/weather/observation/currentweather.jsp](http://www.kma.go.kr/weather/observation/currentweather.jsp). The page displays current weather data for 12 cities in South Korea, including Seoul, Busan, Gwangju, Daegu, Incheon, Suwon, Daejeon, Pusan, Gwangju, Daegu, Incheon, Suwon, and Gwangju. The data includes temperature (°C), humidity (%), wind speed (m/s), and atmospheric pressure (hPa). A sidebar on the right provides links to various KMA services and centers.

지점	현재일기	날씨			기온(°C)			강수			바람		기압(hPa)
		시설 km	운량 1/10	증하운량	현재 기온 온도	불래 지수	일강수 mm	습도 %	풍향	풍속 m/s	해면 기압		
서울	맑음	18.9	1	1	25.6	6.7	70		30	서남서	2.1	1010.1	
부산	구름조금	19.8	3	0	18.4	10.9	64		62	남서	5.2	1011.2	
인천	맑음	20 이상	0	0	20.8	11.1	67		54	서남서	2.9	1011.0	
수원	구름조금	12.6	3	3	25.0	10.8	71		41	북서	2.4	1010.6	
동두천		20 이상			24.9	7.9	70		34	서북서	1.7	1009.9	
파주		17.5			25.1	10.1	71		39	서남서	2.2	1010.6	
강화		20 이상			20.0	10.9	66		56	서남서	5.0	1011.3	

## 웹 페이지의 HTML을 가져와서 파일로 저장하기

- 가져올 HTML 확인하기

- 그럼 웹 브라우저에서 다음 주소로 이동하고, F12를 눌러서 개발자 도구를 표시함(여기서는 크롬을 사용함)
- 왼쪽 아래 커서 버튼을 클릭(Ctrl+Shift+C)한 뒤 현재날씨 표에서 지점을 클릭함

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

## ▼ 개발자 도구에서 지점 선택

The screenshot shows a web browser window displaying a weather observation table. The table includes columns for location, current date/time, and various weather parameters like temperature, humidity, and pressure. A specific cell in the 'Location' column of the first row is highlighted with a red box and labeled '2'. The developer tools are open at the bottom, with the 'Elements' tab selected. A red box highlights the 'Elements' tab and the '1' label. The 'Elements' panel shows the HTML structure of the table, with the first 'th' element (containing '지점') highlighted. The 'Styles' panel on the right shows the CSS for the 'th.top\_line' class, which includes a border and padding.

지점	날씨	기온(°C)			강수		바람		기압(hPa)		
현재일기	시정 km	운량 t/10	중하운량	현재 기온	이슬점 온도	몰래 지수	일강수 mm	습도 %	풍향	풍속 m/s	해면 기압
서울	맑음	18.9	1	1	25.6	6.7	70	30	서남서	2.1	1010.1
부산	구름조금	19.8	3	0	18.4	10.9	64	62	남서	5.2	1011.2
인천	맑음	20 이상	0	0	20.8	11.1	67	54	서남서	2.9	1011.0
수원	구름조금	12.6	3	3	25.0	10.8	71	41	북서	2.4	1010.6
동두천	20 이상				24.9	7.9	70	34	서북서	1.7	1009.9
파주		17.5			25.1	10.1	71	39	서남서	2.2	1010.6
강화		20 이상			20.0	10.9	66	56	서남서	5.0	1011.3
양평		13.9			25.5	7.5	70	32	서북서	2.9	1009.9
이천		19.6			25.6	5.7	70	28	북서	2.9	1010.0

1

2

Elements Console Sources Network Performance Memory Application Security Audits

1

1 <caption>기상실황표</caption>  
1 > <colgroup>...</colgroup>  
1 > <thead>...  
1 > <tr id="table\_header1" class="table\_header">  
1 > > <th scope="col" rowspan="2" class="top\_line">  
1 > > > 지점  
1 > > </th> = 50  
1 > > <th scope="col" colspan="4" class="top\_line">날씨</th>  
1 > > <th scope="col" colspan="3" class="top\_line">기온(°C)</th>  
1 > > <th scope="col" colspan="2" class="top\_line">강수</th>  
1 > > <th scope="col" colspan="2" class="top\_line">바람</th>  
1 > > <th scope="col" class="top\_line">기압(hPa)</th>  
1 > > </tr>  
1 > > <tr class="table\_header1.table\_header">  
1 > > > th.top\_line

Styles Computed Event Listeners >

Filter :hover .cls

element.style {  
}

.table\_develop3 thead th {  
border-top: 2px solid #006ad0;  
}

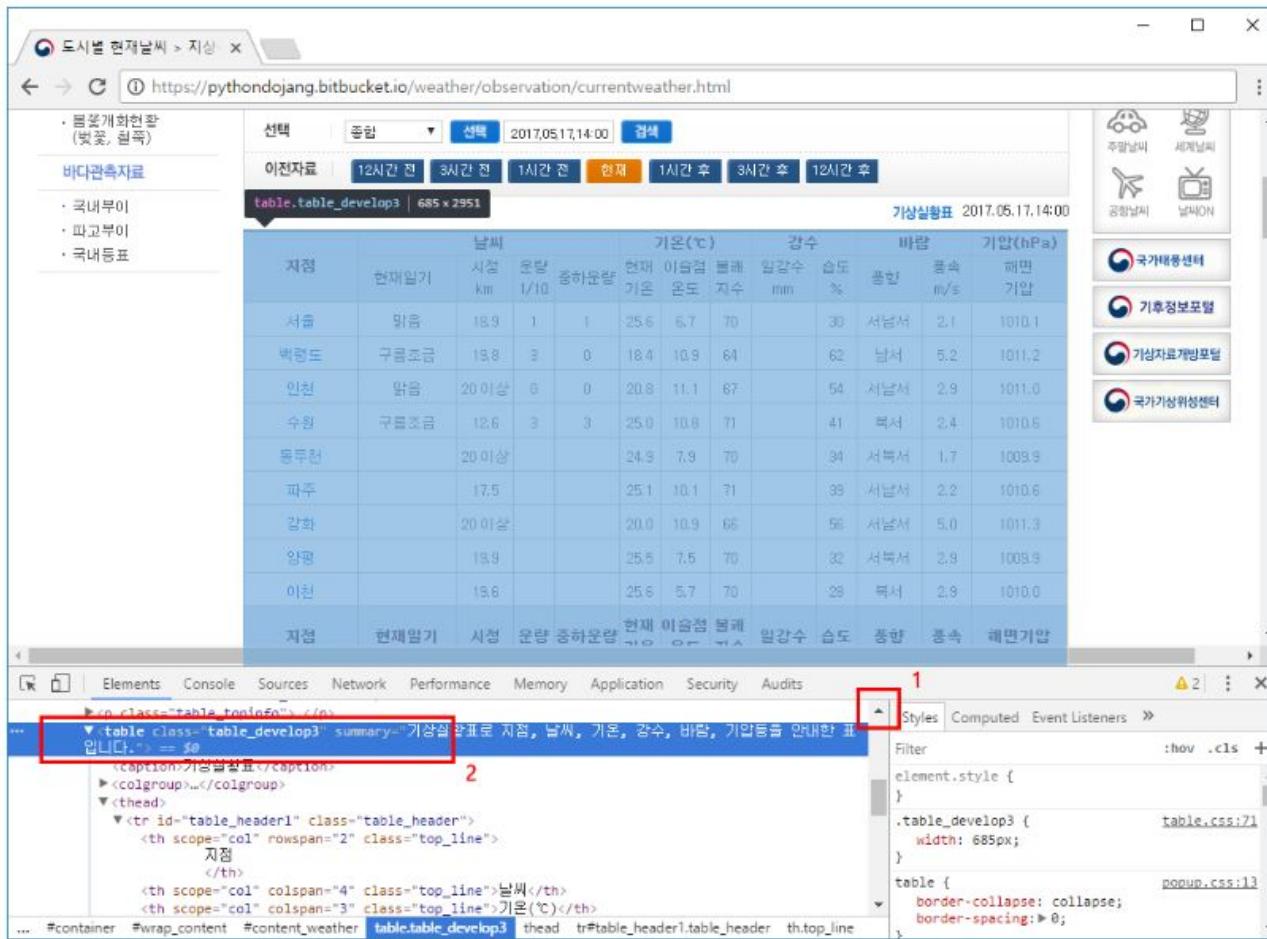
.table\_develop3 thead th {  
padding: 1px 0 0 0;  
border-top: none;

## 웹 페이지의 HTML을 가져와서 파일로 저장하기

- 가져올 HTML 확인하기
  - 이제 지점에 해당하는 HTML 코드가 표시됨
  - 스크롤을 위쪽으로 조금 올린 뒤 `<table class="table_develop3">` 부분을 클릭함
  - 표 전체가 선택되는데 이 표가 우리가 가져올 부분임

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

## ▼ 개발자 도구에서 지점 선택



The screenshot shows a web browser window for '도시별 현재날씨 > 지점' (Current Weather by City > Location). The URL is <https://pythondojang.bitbucket.io/weather/observation/currentweather.html>. The main content is a table titled '기상실황표 2017.05.17.14:00' showing current weather data for various locations in South Korea. The table has columns for location, current time, temperature, humidity, wind, and pressure. On the left, there's a sidebar with navigation links like '도시별 현재날씨' and '바다관측자료'. On the right, there's a sidebar with icons for various weather services. At the bottom, the developer tools' Elements tab is open, showing the HTML structure of the table. A red box highlights the table element, and a red number '1' is placed above the table. A red number '2' is placed next to the table element in the DOM tree. The developer tools also show the CSS styles applied to the table.

지점	현재일기	날씨			기온(°C)			강수		바람		기압(hPa)	
		시청 km	온도 1/10	증하운량	현재	이슬점	濡れ	일강수 mm	습도 %	풍향	풍속 m/s	해면 기압	
서울	맑음	18.9	1	1	25.6	6.7	70	30	서남서	2.1	1010.1		
부평도	구름조금	19.8	3	0	18.4	10.9	64	62	남서	5.2	1011.2		
인천	맑음	20.0이상	6	0	20.8	11.1	67	54	서남서	2.9	1011.0		
수원	구름조금	12.6	3	3	25.0	10.8	71	41	북서	2.4	1010.6		
동두천	20.0이상				24.9	7.9	70	34	서북서	1.7	1009.9		
파주		17.5			25.1	10.1	71	39	서남서	2.2	1010.6		
강화		20.0이상			20.0	10.9	66	56	서남서	5.0	1011.3		
임평		19.9			25.5	7.5	70	32	서북서	2.9	1009.9		
아천		19.8			25.6	5.7	70	29	북서	2.9	1010.0		

Developer Tools Elements Tab:

- Selected element: `<table class="table_tableInfo" ...>`
- Element summary: "기상실황표로 지점, 날씨, 기온, 강수, 바람, 기압 등을 안내합니다." => 50
- Element ID: 2
- Styles panel:

  - Filter: :hover .cls
  - element.style {
  - .table\_tableInfo { table.css:71 width: 685px; }
  - table { popup.css:13 border-collapse: collapse; border-spacing: 0; }

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

- 주피터 노트북 만들기

- 주피터 노트북에서 새 노트북을 만든 뒤 코드 셀에 다음 내용을 입력함

weather.ipynb

```
import requests          # 웹 페이지의 HTML을 가져오는 모듈
from bs4 import BeautifulSoup # HTML을 파싱하는 모듈

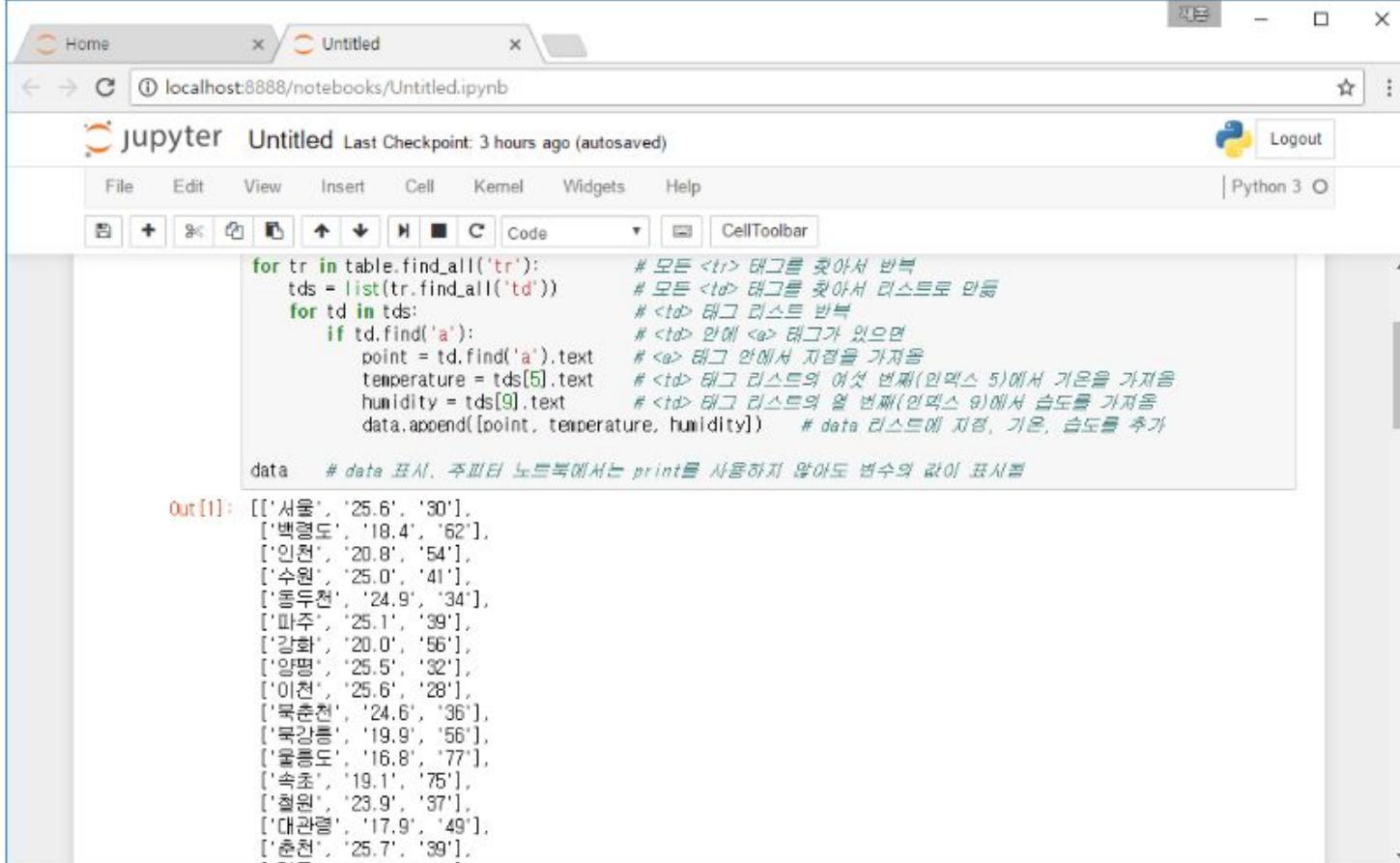
# 웹 페이지를 가져온 뒤 BeautifulSoup 객체로 만들
response = requests.get('https://pythondojang.bitbucket.io/weather/observation/currentweather.html')
soup = BeautifulSoup(response.content, 'html.parser')

table = soup.find('table', { 'class': 'table_develop3' })      # <table class="table_develop3">을 찾음
data = []                                         # 데이터를 저장할 리스트 생성
for tr in table.find_all('tr'):                  # 모든 <tr> 태그를 찾아서 반복(각 지점의 데이터를 가져옴)
    tds = list(tr.find_all('td'))               # 모든 <td> 태그를 찾아서 리스트로 만들
                                                # (각 날씨 값을 리스트로 만들)
    for td in tds:                            # <td> 태그 리스트 반복(각 날씨 값을 가져옴)
        if td.find('a'):                      # <td> 안에 <a> 태그가 있으면(지점인지 확인)
            point = td.find('a').text        # <a> 태그 안에서 지점을 가져옴
            temperature = tds[5].text      # <td> 태그 리스트의 여섯 번째(인덱스 5)에서 기온을 가져옴
            humidity = tds[9].text        # <td> 태그 리스트의 열 번째(인덱스 9)에서 습도를 가져옴
            data.append([point, temperature, humidity])  # data 리스트에 지점, 기온, 습도를 추가

data    # data 표시. 주피터 노트북에서는 print를 사용하지 않아도 변수의 값이 표시됨
```

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

## ▼ 웹 페이지에서 가져온 데이터 표시



The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** Home, Untitled, localhost:8888/notebooks/Untitled.ipynb, Jupyter Untitled, Last Checkpoint: 3 hours ago (autosaved), Python 3, Logout.
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, CellToolbar buttons.
- Code Cell:** Contains Python code to extract data from a table and print it. The code is annotated with comments explaining the steps:

```
for tr in table.find_all('tr'):
    tds = list(tr.find_all('td'))
    for td in tds:
        if td.find('a'):
            point = td.find('a').text
            temperature = tds[5].text
            humidity = tds[9].text
            data.append([point, temperature, humidity])
            # data 리스트에 지점, 기온, 습도를 추가

data # data 표시. 주피터 노트북에서는 print를 사용하지 않아도 변수의 값이 표시됨
```

- Output:** Shows the list of data extracted from the table, with each row containing three values: point, temperature, and humidity.

Point	Temperature	Humidity
서울	25.6	30
밴경도	18.4	62
인천	20.8	54
수원	25.0	41
동두천	24.9	34
파주	25.1	39
강회	20.0	56
양평	25.5	32
이천	25.6	28
목춘천	24.6	36
목강릉	19.9	56
율동도	16.8	77
속초	19.1	75
청원	23.9	37
대관절	17.9	49
춘천	25.7	39

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

- HTML의 데이터를 가져오는 방식 알아보기
  - 그럼 지점, 기온(현재기온), 습도 값을 어떻게 가져오는지 알아보자

## ▼ 웹 페이지에서 원하는 값 찾기

도시별 현재날씨 > 지상

https://pythondjango.bitbucket.io/weather/observation/currentweather.html

선택 종합 선택 2017.05.17.14:00 결제

비다관측지역

1시간 전 3시간 전 1시간 전 현재 1시간 후 3시간 후 12시간 후

기상실황표 2017.05.17.14:00

지점	날씨	기온(°C)	강수	바람	기압(hPa)
서울	맑음	18.9	1	70	1010.1
제주도	구름조금	19.8	3	62	1011.2
인천	맑음	20.0	0	54	1011.0
수원	구름조금	12.6	3	41	1010.6

기상 실황 표시

국내부이  
· 피고부이  
· 국내동표

국가 대응센터  
기후 정보포털  
기상자료개발포털  
국가 기상 위성센터

Elements Console Sources Network Performance Memory Application Security Audits

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

6610

6611

6612

6613

6614

6615

6616

6617

6618

6619

6620

6621

6622

6623

6624

6625

6626

6627

6628

6629

6630

6631

6632

6633

6634

6635

6636

6637

6638

6639

6640

6641

6642

6643

6644

6645

6646

6647

6648

6649

6650

6651

6652

6653

6654

6655

6656

6657

6658

6659

6660

6661

6662

6663

6664

6665

6666

6667

6668

6669

66610

66611

66612

66613

66614

66615

66616

66617

66618

66619

66620

66621

66622

66623

66624

66625

66626

66627

66628

66629

66630

66631

66632

66633

66634

66635

66636

66637

66638

66639

66640

66641

66642

66643

66644

66645

66646

66647

66648

66649

66650

66651

66652

66653

66654

66655

66656

66657

66658

66659

66660

66661

66662

66663

66664

66665

66666

66667

66668

66669

666610

666611

666612

666613

666614

666615

666616

666617

666618

666619

666620

666621

666622

666623

666624

666625

666626

666627

666628

666629

666630

666631

666632

666633

666634

666635

666636

666637

666638

666639

666640

666641

666642

666643

666644

666645

666646

666647

666648

666649

666650

666651

666652

666653

666654

666655

666656

666657

666658

666659

666660

666661

666662

666663

666664

666665

666666

666667

666668

666669

6666610

6666611

6666612

6666613

6666614

6666615

6666616

6666617

6666618

6666619

6666620

6666621

6666622

6666623

6666624

6666625

6666626

6666627

6666628

6666629

6666630

6666631

6666632

6666633

6666634

6666635

6666636

6666637

6666638

6666639

6666640

6666641

6666642

6666643

6666644

6666645

6666646

6666647

6666648

6666649

6666650

6666651

6666652

6666653

6666654

6666655

6666656

6666657

6666658

6666659

6666660

6666661

6666662

6666663

6666664

6666665

6666666

6666667

6666668

6666669

66666610

66666611

66666612

66666613

66666614

66666615

66666616

66666617

66666618

66666619

66666620

66666621

66666622

66666623

66666624

66666625

66666626

66666627

66666628

66666629

66666630

66666631

66666632

66666633

66666634

66666635

66666636

66666637

66666638

66666639

66666640

66666641

66666642

66666643

66666644

66666645

66666646

66666647

66666648

66666649

66666650

66666651

66666652

66666653

66666654

66666655

66666656

66666657

66666658

66666659

66666660

66666661

66666662

66666663

66666664

66666665

66666666

66666667

66666668

66666669

666666610

666666611

666666612

666666613

666666614

666666615

666666616

666666617

666666618

666666619

666666620

666666621

666666622

666666623

666666624

666666625

666666626

666666627

666666628

666666629

666666630

666666631

666666632

666666633

666666634

666666635

666666636

666666637

666666638

666666639

666666640

666666641

666666642

666666643

666666644

666666645

666666646

666666647

666666648

666666649

666666650

666666651

666666652

666666653

666666654

666666655

666666656

666666657

666666658

666666659

666666660

666666661

666666662

666666663

666666664

666666665

666666666

666666667

666666668

666666669

6666666610

6666666611

6666666612

6666666613

6666666614

6666666615

6666666616

6666666617

6666666618

6666666619

6666666620

6666666621

6666666622

6666666623

6666666624

6666666625

6666666626

6666666627

6666666628

6666666629

6666666630

6666666631

6666666632

6666666633

6666666634

6666666635

6666666636

6666666637

6666666638

6666666639

6666666640

6666666641

6666666642

6666666643

6666666644

6666666645

6666666646

6666666647

6666666648

6666666649

6666666650

6666666651

6666666652

6666666653

6666666654

6666666655

6666666656

6666666657

6666666658

6666666659

6666666660

6666666661

6666666662

6666666663

6666666664

6666666665

6666666666

6666666667

6666666668

6666666669

66666666610

66666666611

66666666612

66666666613

66666666614

66666666615

66666666616

66666666617

66666666618

66666666619

66666666620

66666666621

66666666622

66666666623

66666666624

66666666625

66666666626

66666666627

66666666628

66666666629

66666666630

66666666631

66666666632

66666666633

66666666634

66666666635

66666666636

66666666637

66666666638

66666666639

66666666640

66666666641

66666666642

66666666643

66666666644

66666666645

66666666646

66666666647

66666666648

66666666649

66666666650

66666666651

66666666652

66666666653

66666666654

66666666655

66666666656

66666666657

66666666658

66666666659

66666666660

66666666661

66666666662

66666666663

66666666664

66666666665

66666666666

66666666667

66666666668

66666666669

666666666610

666666666611

666666666612

666666666613

666666666614

666666666615

666666666616

666666666617

666666666618

666666666619

666666666620

666666666621

66666

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

- HTML의 데이터를 가져오는 방식 알아보기
  - 이제 HTML 코드를 살펴보자

```
<table class="table_develop3" summary="기상실황표로 지점, 날씨, 기온, 강수, 바람, 기압등을 안내한 표입니다.">
...생략...
    <tr>
        <td><a href="/weather/observation/currentweather.jsp?tm=2017.5.17.14:00&type=t99&mode=0&auto_man=m&stn=108" >서울</a></td>
        <td>맑음</td>

        <td>18.9</td>
        <td>1</td>
        <td>1</td>
        <td>25.6</td>
        <td>6.7</td>
        <td>70</td>
        <td>&ampnbsp</td>

        <td>30</td>
        <td>서남서</td>
        <td>2.1</td>
        <td>1010.1</td>
    </tr>
...생략...
```

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

- HTML의 데이터를 가져오는 방식 알아보기
  - 다시 파이썬에서 HTML을 가져오는 부분임
  - HTML 파싱은 텍스트 형태의 HTML 코드를 분석해서 객체로 만든 뒤 검색하거나 편집할 수 있도록 만드는 작업임
  - bs4는 BeautifulSoup 라이브러리이고 HTML 코드를 파싱하는데 사용함

```
import requests           # 웹 페이지의 HTML을 가져오는 모듈
from bs4 import BeautifulSoup # HTML을 파싱하는 모듈

# 웹 페이지를 가져온 뒤 BeautifulSoup 객체로 만듦
response = requests.get('https://pythondojang.bitbucket.io/weather/observation/currentweather.html')
soup = BeautifulSoup(response.content, 'html.parser')
```

- content 속성에는 텍스트 형태의 HTML이 들어있으며, 파이썬의 html.parser 모듈을 사용해서 파싱하도록 설정함

# 웹 페이지의 HTML을 가져와서 파일로 저장하기

- HTML의 데이터를 가져오는 방식 알아보기
  - 이제 BeautifulSoup 클래스로 만든 soup 객체로 태그를 찾음

```
table = soup.find('table', { 'class': 'table_develop3' })      # <table class="table_develop3">을 찾음
```

```
data = []                      # 데이터를 저장할 리스트 생성
for tr in table.find_all('tr'):  # 모든 <tr> 태그를 찾아서 반복(각 지점의 데이터를 가져옴)
```

```
tds = list(tr.find_all('td'))    # 모든 <td> 태그를 찾아서 리스트로 만듦
# (각 날씨 값을 리스트로 만듦)
```

```
for td in tds:                  # <td> 태그 리스트 반복(각 날씨 값을 가져옴)
    if td.find('a'):            # <td> 안에 <a> 태그가 있으면(지점인지 확인)
        point = td.find('a').text # <a> 태그 안에서 지점을 가져옴
        temperature = tds[5].text # <td> 태그 리스트의 여섯 번째(인덱스 5)에서 기온을 가져옴
        humidity = tds[9].text   # <td> 태그 리스트의 열 번째(인덱스 9)에서 습도를 가져옴
        data.append([point, temperature, humidity])  # data 리스트에 지점, 기온, 습도를 추가
```

## 웹 페이지의 HTML을 가져와서 파일로 저장하기

- 데이터를 csv 파일에 저장하기
  - 이 코드 셀에서 다음 코드를 실행함
  - (셀이 생기지 않았다면 메뉴의 Insert > Insert Cell Below 실행)

weather.ipynb

```
with open('weather.csv', 'w') as file:    # weather.csv 파일을 쓰기 모드로 열기
    file.write('point,temperature,humidity\n')          # 컬럼 이름 추가
    for i in data:                                     # data를 반복하면서
        file.write('{0},{1},{2}\n'.format(i[0], i[1], i[2]))  # 지점,온도,습도를 줄 단위로 저장
```

- csv 파일은 Comma-separated values의 약자인데 각 컬럼을 ,(콤마)로 구분해서 표현한다고 해서 csv라고 부름
- 콤마와 값 사이에는 공백을 넣지 않고 반드시 붙여줌

## 웹 페이지의 HTML을 가져와서 파일로 저장하기

- 데이터를 csv 파일에 저장하기
  - weather.csv 파일을 메모장이나 기타 텍스트 편집기로 열어보면 다음과 같은 모양으로 지점, 기온, 습도 값이 저장된 것을 볼 수 있음

weather.csv

```
point,temperature,humidity
서울,25.6,30
백령도,18.4,62
인천,20.8,54
수원,25.0,41
...생략...
```

- csv 파일을 저장할 때 컬럼 이름은 영어로 지정해줌
- 영어로 지정하면 나중에 각 컬럼에 접근할 때 df.temperature처럼 속성으로 깔끔하게 사용할 수 있음

# 데이터로 그래프 그리기

- 데이터로 그래프 그리기

- 주피터 노트북은 각 코드 셀의 실행 상태가 이어지므로 노트북을 새로 만들 필요 없이 코드 셀을 계속 추가하면서 작성하면 됨
- 다음은 `weather.csv` 파일을 읽어서 `pandas`의 `DataFrame` 객체로 만듬 (`pandas`는 데이터를 처리할 때 사용하는 패키지)

## weather.ipynb

```
# %matplotlib inline을 설정하면 matplotlib.pyplot의 show 함수를 호출하지 않아도
# 주피터 노트북 안에서 그래프가 표시됨
%matplotlib inline
import pandas as pd          # 데이터를 저장하고 처리하는 패키지
import matplotlib as mpl     # 그래프를 그리는 패키지
import matplotlib.pyplot as plt # 그래프를 그리는 패키지

# csv 파일을 읽어서 DataFrame 객체로 만듦. 인덱스 헤더는 point로 설정
df = pd.read_csv('weather.csv', index_col='point')
df    # df 표시
```

## 실행 결과

```
-----
UnicodeDecodeError                         Traceback (most recent call last)
<ipython-input-4-81f61a36d931> in <module>()
    7
    8 # csv 파일을 읽어서 DataFrame 객체로 만듦. 인덱스 항목(컬럼)은 point로 설정
----> 9 df = pd.read_csv('weather.csv', index_col='point')
   10 df    # df 표시
```

# 데이터로 그래프 그리기

- 데이터로 그래프 그리기

- 실행을 해보면 `pd.read_csv`에서 에러가 발생함
- 우리가 가져온 도시별 현재날씨 페이지는 인코딩이 `euc-kr`로 만들어져 있기 때문임

```
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />
```

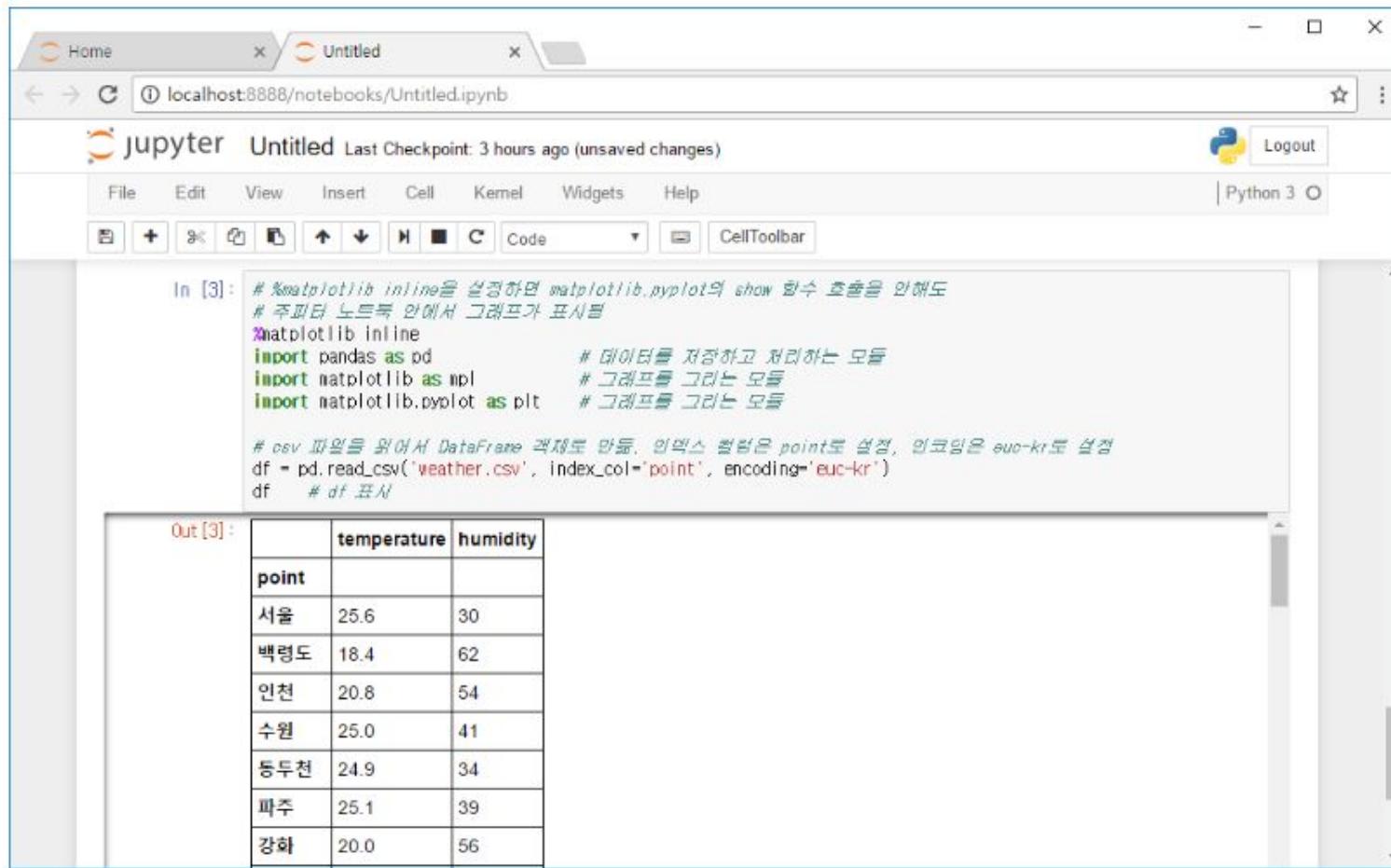
- `weather.csv` 파일도 인코딩이 `euc-kr`로 저장됨
- `pd.read_csv` 함수에 `EUC-KR` 인코딩을 설정해서 파일을 읽어야 함

```
# csv 파일을 읽어서 DataFrame 객체로 만들. 인덱스 컬럼은 point로 설정, 인코딩은 euc-kr로 설정
df = pd.read_csv('weather.csv', index_col='point', encoding='euc-kr')
```

- 주피터 노트북은 `pandas` 패키지의 `DataFrame`을 표 형태로 보여주므로 매우 편리함

# 데이터로 그래프 그리기

▼ 그림 46-20 csv 파일을 읽은 뒤 DataFrame 표시



The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** Home, Untitled, localhost:8888/notebooks/Untitled.ipynb, Logout, Python 3.
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, CellToolbar.
- In [3]:** Python code for importing pandas and matplotlib, and reading a CSV file named 'weather.csv' into a DataFrame 'df'. The code includes comments explaining the use of %matplotlib inline and the reading of the CSV file.
- Out [3]:** A table showing the data from the DataFrame 'df'. The table has columns: point, temperature, and humidity. The data rows are: 서울 (25.6, 30), 백령도 (18.4, 62), 인천 (20.8, 54), 수원 (25.0, 41), 등두천 (24.9, 34), 파주 (25.1, 39), and 강화 (20.0, 56).

point	temperature	humidity
서울	25.6	30
백령도	18.4	62
인천	20.8	54
수원	25.0	41
등두천	24.9	34
파주	25.1	39
강화	20.0	56

# 데이터로 그래프 그리기

- 특별시 광역시만 모으기
  - 이번에는 df에서 특별시, 광역시만 모아서 따로 DataFrame 객체를 만들어보자

weather.ipynb

```
# 특별시, 광역시만 모아서 DataFrame 객체로 만듦
city_df = df.loc[['서울', '인천', '대전', '대구', '광주', '부산', '울산']]
city_df # city_df 표시
```

- 이렇게 DataFrame의 loc 속성을 이용하면 특정 인덱스의 데이터만 가져올 수 있음(loc 속성은 label-location의 약자로 레이블을 지정해 특정 인덱스의 데이터만 가져오는데 쓰임)

# 데이터로 그래프 그리기

## ▼ 그림 46-21 특별시, 광역시만 뽑아서 DataFrame 객체로 만듬

```
In [4]: # 특별시, 광역시만 뽑아서 DataFrame 객체로 만듬
city_df = df.loc[['서울', '인천', '대전', '대구', '광주', '부산', '울산']]
city_df # city_df 표시
```

Out [4]:

point	temperature	humidity
서울	25.6	30
인천	20.8	54
대전	25.1	37
대구	25.7	31
광주	24.0	35
부산	20.2	66
울산	19.7	70

# 데이터로 그래프 그리기

- 특별히 광역시만 모으기

- loc에 인덱스를 하나만 지정하면 해당 인덱스의 데이터만 가져오고, 인덱스 여러 개를 리스트 형태로 지정하면 DataFrame 형태로 가져옴
- 이때 인덱스는 문자열로 지정해줌

- DataFrame 객체.loc['인덱스']
- DataFrame 객체.loc[['인덱스1', '인덱스2']]

```
>>> df.loc['서울']
temperature    25.6
humidity      30.0
Name: 서울, dtype: float64
>>> df.loc[['서울', '부산']]
      temperature  humidity
point
서울            25.6        30
부산            20.2        66
```

# 데이터로 그래프 그리기

- 특별시, 광역시만 그래프 그리기
  - 코드 셀에서 다음 코드를 실행해보자
  - (셀이 생기지 않았다면 메뉴의 Insert > Insert Cell Below 실행)

weather.ipynb

```
# Windows 한글 폰트 설정
font_name = mpl.font_manager.FontProperties(fname='C:/Windows/Fonts/malgun.ttf').get_name()
mpl.rc('font', family=font_name)

# 차트 종류, 제목, 차트 크기, 범례, 폰트 크기 설정
ax = city_df.plot(kind='bar', title='날씨', figsize=(12, 4), legend=True, fontsize=12)
ax.set_xlabel('도시', fontsize=12)          # x축 정보 표시
ax.set_ylabel('기온/습도', fontsize=12)      # y축 정보 표시
ax.legend(['기온', '습도'], fontsize=12)    # 범례 지정
```

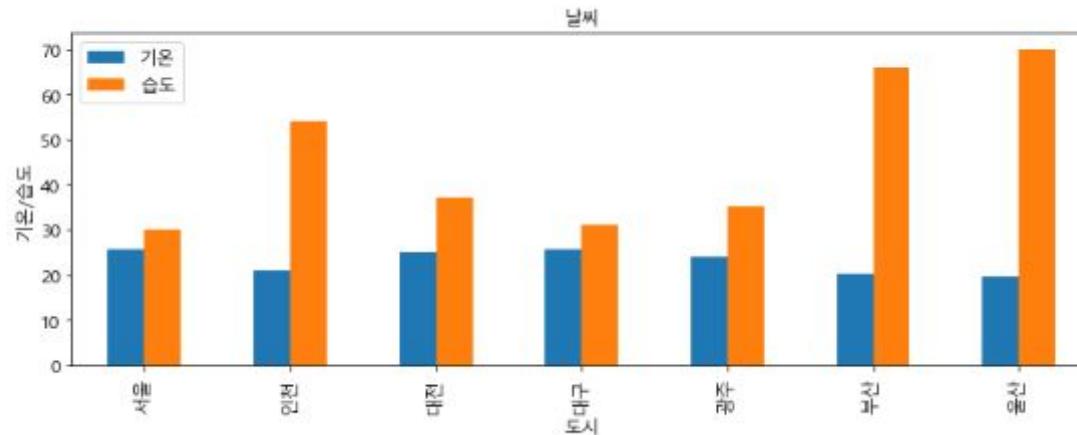
# 데이터로 그래프 그리기

## ▼ 그림 46-22 DataFrame으로 그래프 그리기

```
In [5]: # 한글 폰트 설정
font_name = mpl.font_manager.FontProperties(fname='C:/Windows/Fonts/nalgun.ttf').get_name()
mpl.rcParams['font'].family=font_name

# 차트 종류, 제목, 차트 크기, 폰트, 폰트 크기 설정
ax = city_df.plot(kind='bar', title='날씨', figsize=(12, 4), legend=True, fontsize=12)
ax.set_xlabel('도시', fontsize=12)          # x축 정보 표시
ax.set_ylabel('기온/습도', fontsize=12)      # y축 정보 표시
ax.legend(['기온', '습도'], fontsize=12)    # 범례 지정
```

```
Out [5]: <matplotlib.legend.Legend at 0x23da02b0e80>
```



# 데이터로 그래프 그리기

- 소스 코드 살펴보기
  - matplotlib은 기본적으로 그래프에서 한글 표시가 안 됨

```
# Windows 한글 폰트 설정
font_name = mpl.font_manager.FontProperties(fname='C:/Windows/Fonts/malgun.ttf').get_name()
mpl.rc('font', family=font_name)
```

# 데이터로 그래프 그리기

- `plot` 메서드로 그래프 그리기
  - `DataFrame` 객체로 그래프를 그릴 때는 `plot` 메서드를 사용함

```
# 차트 종류, 제목, 차트 크기, 범례, 폰트 크기 설정
ax = city_df.plot(kind='bar', title='날씨', figsize=(12, 4), legend=True, fontsize=12)
```

- `city_df`에서 `plot` 메서드를 사용하면 축 서브플롯(AxesSubplot) 객체가 나옴(`plot`은 그래프를 그리다라는 뜻)

```
ax.set_xlabel('도시', fontsize=12)          # x축 정보 표시
ax.set_ylabel('기온/습도', fontsize=12)      # y축 정보 표시
```

- `legend` 메서드에 범례를 리스트 형태로 넣어줌(범례는 차트의 각 막대가 무슨 값인지 표시)

03

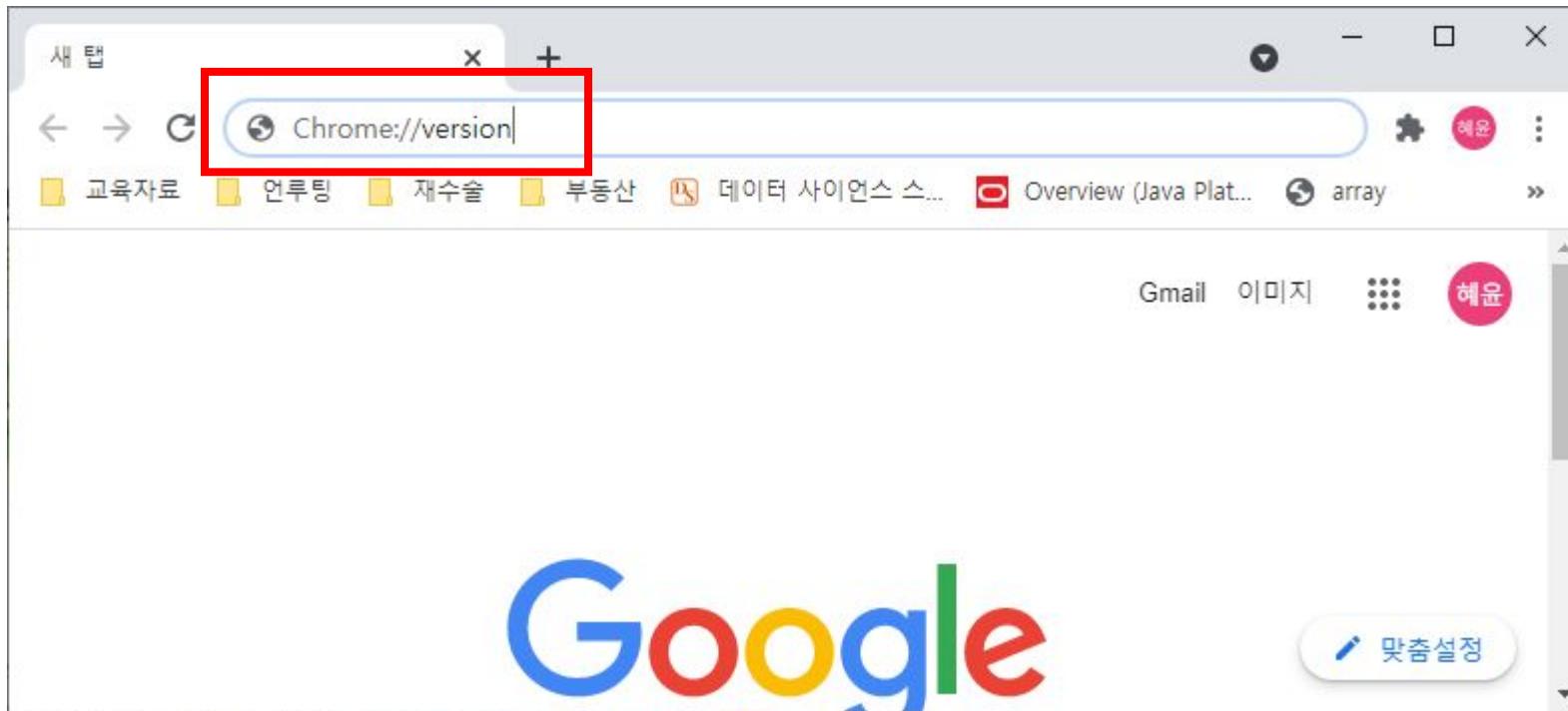
## Selenium을 이용한 크롤링

## Selenium

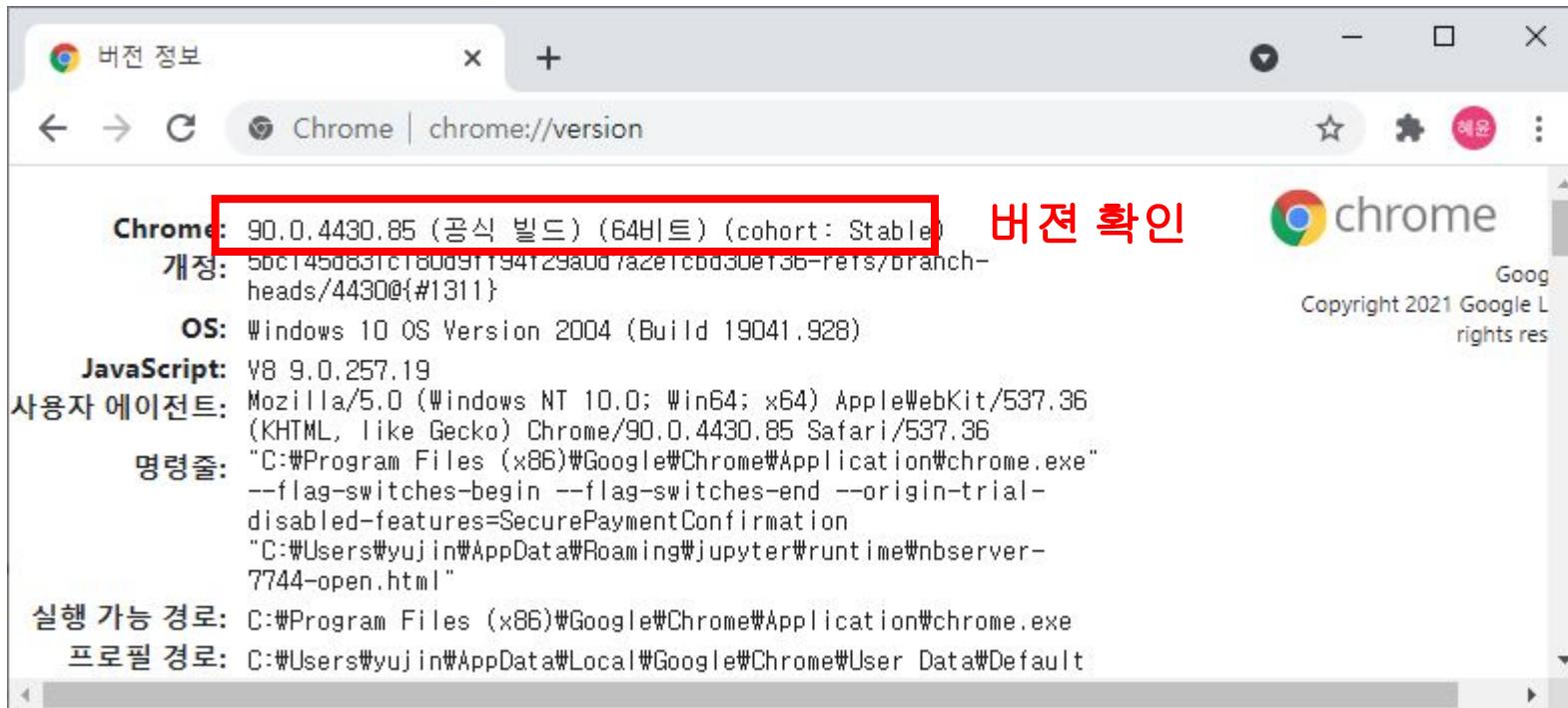
- 주로 웹앱을 테스트하는데 이용하는 프레임워크
- webdriver라는 API를 통해 운영체제에 설치된 Chrome등의 브라우저를 제어해서 버튼 클릭, 검색어 입력 등의 작업을 실행

# 크롬 버전 확인

- 크롬을 실행시키고 주소 표시줄에 입력



# 크롬 버전 확인



<https://sites.google.com/a/chromium.org/chromedriver/downloads>

## Downloads

### Current Releases

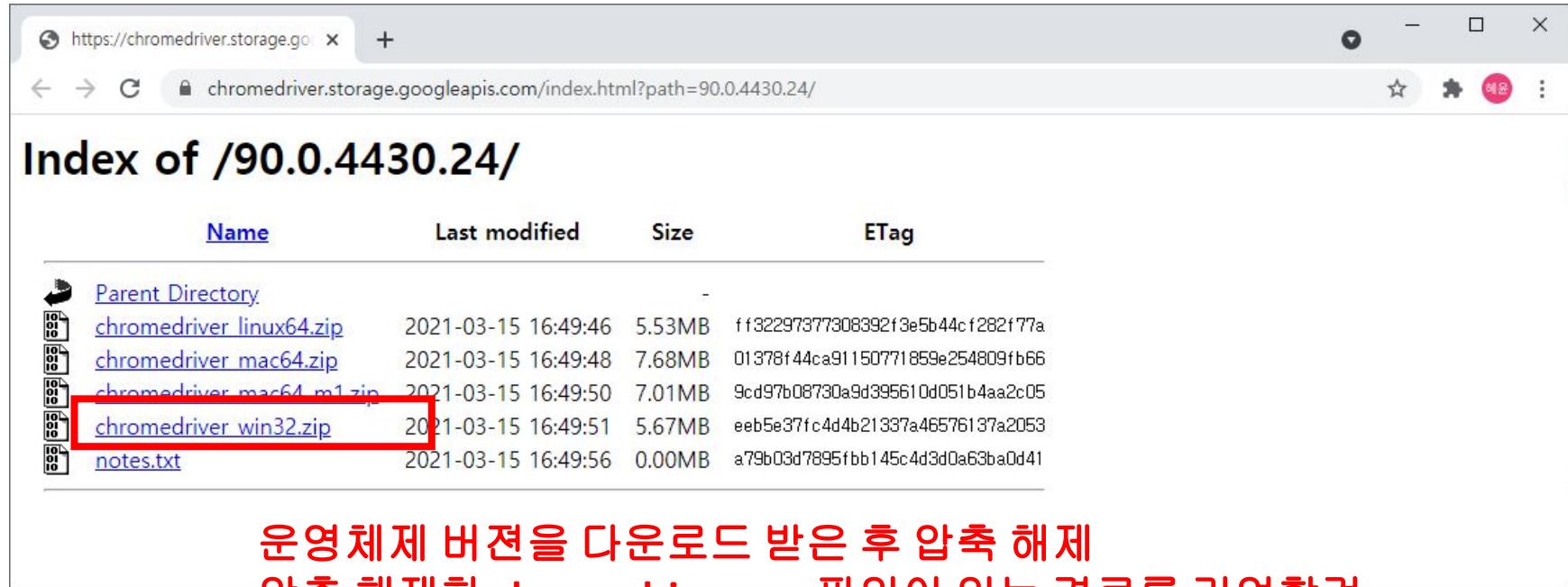
앞페이지에서 확인한 크롬버전과  
같은 프로그램 클릭

- If you are using Chrome version 90, please download [ChromeDriver 91.0.4472.19](#)
- If you are using Chrome version 90, please download [ChromeDriver 90.0.4430.24](#)
- If you are using Chrome version 89, please download [ChromeDriver 89.0.4389.23](#)
- If you are using Chrome version 88, please download [ChromeDriver 88.0.4324.96](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

If you are using Chrome from Dev or Canary channel, please following instructions on the [ChromeDriver Canary](#) page.

For more information on selecting the right version of ChromeDriver, please see the [Version Selection](#) page.

# 크롬 드라이버 다운로드



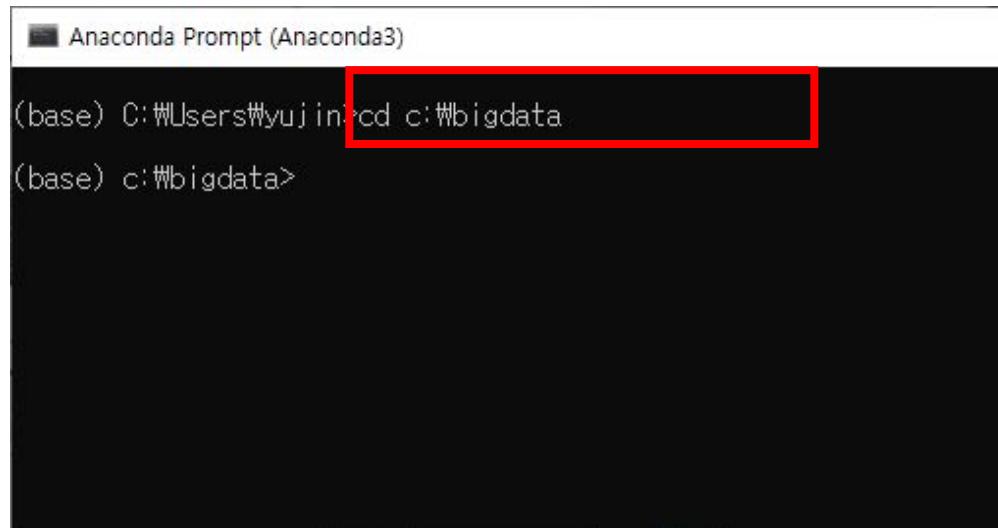
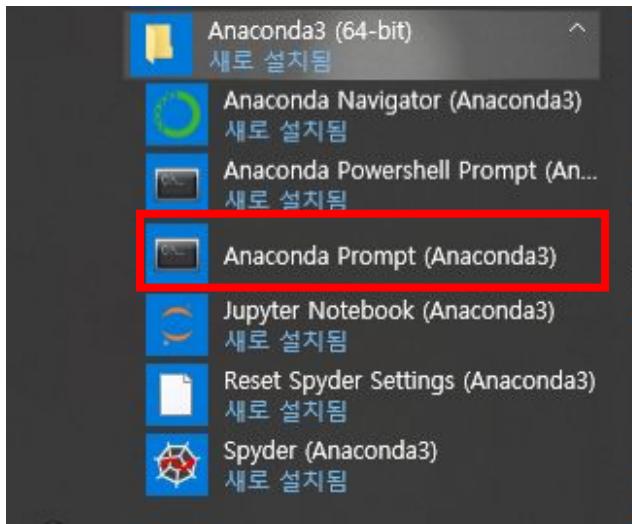
Index of /90.0.4430.24/

<u>Name</u>	Last modified	Size	ETag
<a href="#">Parent Directory</a>		-	
<a href="#">chromedriver_linux64.zip</a>	2021-03-15 16:49:46	5.53MB	ff32297377308892f3e5b44cf282f77a
<a href="#">chromedriver_mac64.zip</a>	2021-03-15 16:49:48	7.68MB	01378f44ca91150771859e254809fb66
<a href="#">chromedriver_mac64_m1.zip</a>	2021-03-15 16:49:50	7.01MB	9cd97b08730a9d395610d051b4aa2c05
<a href="#">chromedriver_win32.zip</a>	2021-03-15 16:49:51	5.67MB	eeb5e37fc4d4b21337a46576137a2053
<a href="#">notes.txt</a>	2021-03-15 16:49:56	0.00MB	a79b03d7895fbb145c4d3d0a63ba0d41

운영체제 버전을 다운로드 받은 후 압축 해제  
압축 해제한 chromedriver.exe 파일이 있는 경로를 기억할것

# 크롬 드라이버 다운로드

- 앞페이지에서 압축을 해제한 chromedriver.exe 파일이 있는 경로로 이동
- 예제에서는 c:\bigdata 폴더에 chromedriver.exe 파일이 존재하기 때문에 이동



The image shows a terminal window titled 'Anaconda Prompt (Anaconda3)'. The command '(base) C:\Users\yujin>cd c:\bigdata' is being typed and is highlighted with a red box. The prompt then changes to '(base) c:\bigdata>'. The background of the terminal window is black, and the text is white.

```
Anaconda Prompt (Anaconda3)
(base) C:\Users\yujin>cd c:\bigdata
(base) c:\bigdata>
```

# 크롬 드라이버 다운로드

- Jupyter notebook 실행

```
Anaconda Prompt (Anaconda3) - jupyter notebook

(base) C:\Users\yujin>cd c:\bigdata
(base) c:\bigdata>jupyter notebook
[1 06:33:22.843 NotebookApp] JupyterLab extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[1 06:33:22.843 NotebookApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[1 06:33:22.845 NotebookApp] Serving notebooks from local directory: c:\bigdata
[1 06:33:22.845 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[1 06:33:22.846 NotebookApp] http://localhost:8888/?token=6788b09c785439a41790b87d57e1d6dbc1481a2448bc5a5c
[1 06:33:22.846 NotebookApp] or http://127.0.0.1:8888/?token=6788b09c785439a41790b87d57e1d6dbc1481a2448bc5a5c
[1 06:33:22.846 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 06:33:22.919 NotebookApp]

To access the notebook, open this file in a browser:
  file:///C:/Users/yujin/AppData/Roaming/jupyter/runtime/nbserver-29108-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=6788b09c785439a41790b87d57e1d6dbc1481a2448bc5a5c
  or http://127.0.0.1:8888/?token=6788b09c785439a41790b87d57e1d6dbc1481a2448bc5a5c
```

# Selenium 설치

- Jupyter notebook에서 새로운 노트북을 만들고 Selenium 설치

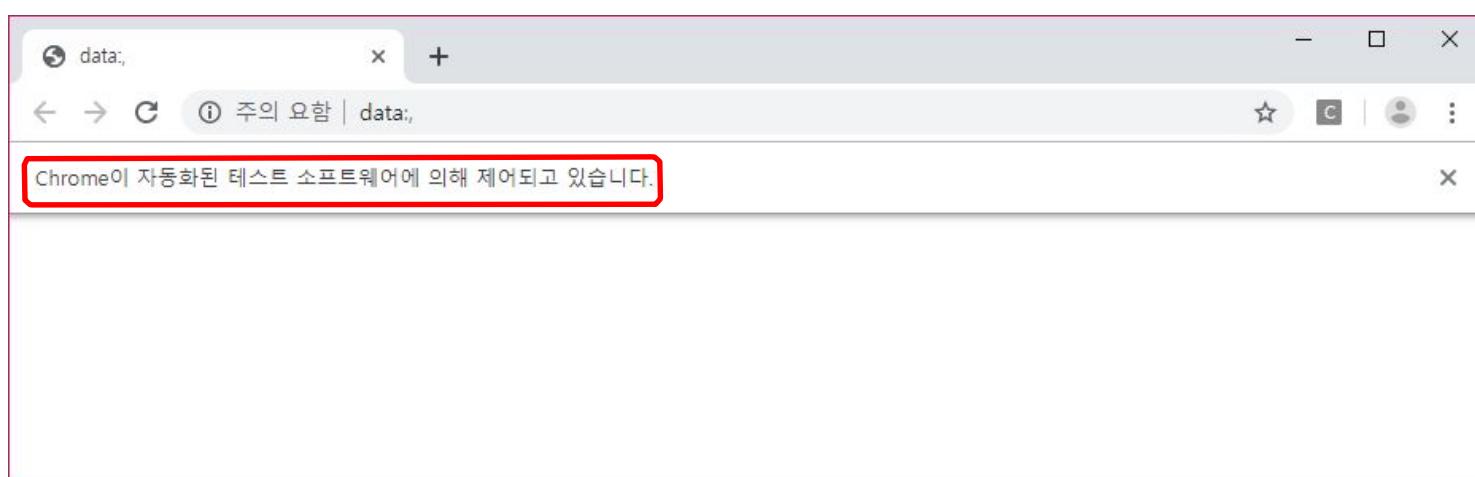
```
In [1]: ! pip install selenium
```

```
Requirement already satisfied: selenium in c:\programdata\anaconda3\lib\site-packages (3.141.0)
Requirement already satisfied: urllib3 in c:\programdata\anaconda3\lib\site-packages (from selenium) (1.25.11)
```

# 크롬 드라이버 로드

## 크롬 드라이버 로드

```
n [1]: from selenium import webdriver as wd  
driver = wd.Chrome(executable_path='chromedriver.exe')
```



# 크롬 드라이버 로드

```
In [2]: from selenium import webdriver as wd
url = 'http://www.naver.com'

#chrome driver load
driver = wd.Chrome(executable_path='chromedriver.exe')

#page load
driver.get(url)
```

## 실행 결과



# Naver 메인 페이지 검색

## ■ Naver 메인 페이지 검색

```
#naverSearch0.py
from selenium import webdriver as wd

url = 'http://www.naver.com'

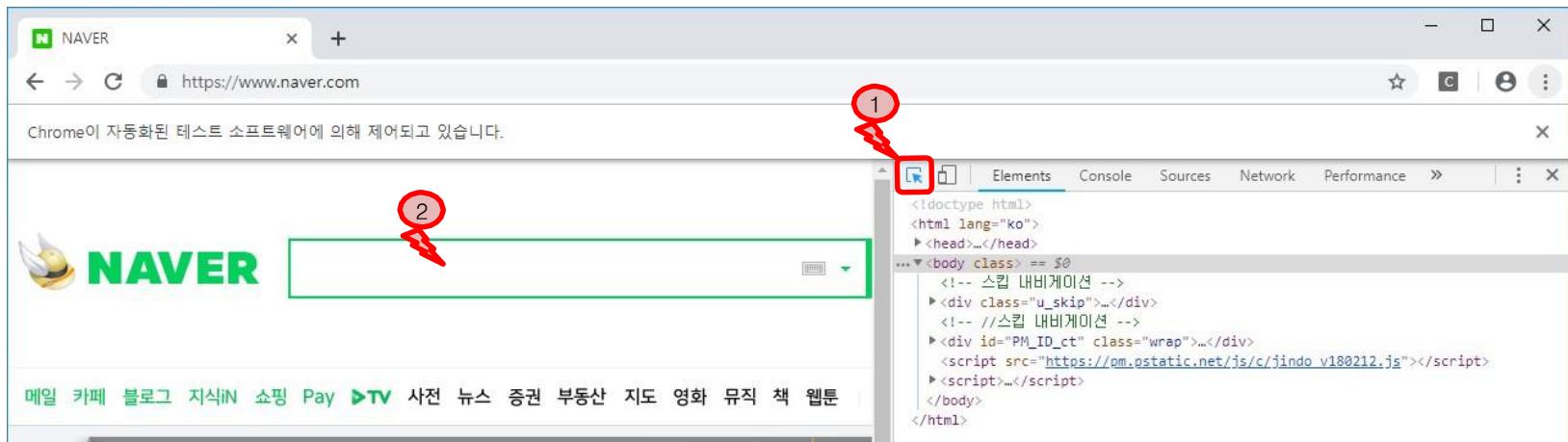
#chrome driver load
driver = wd.Chrome(executable_path='chromedriver.exe')

#page load
driver.get(url)
```

# Naver 메인 페이지 검색

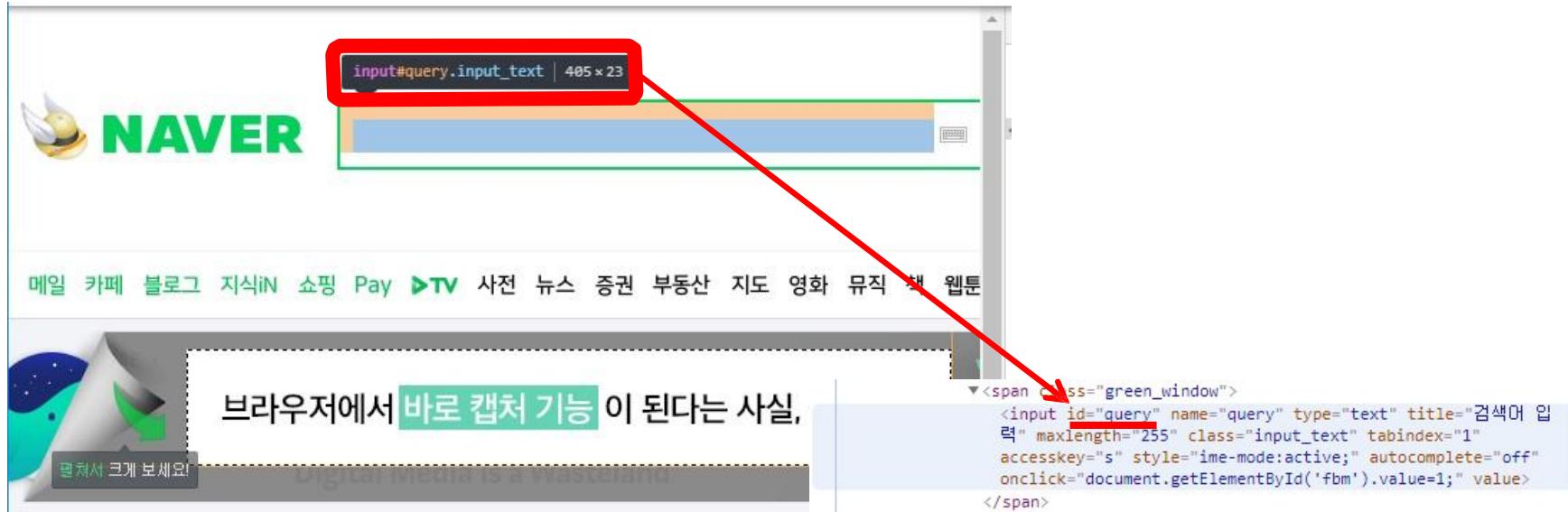
## Naver 메인 페이지 검색

### □ 마우스 우클릭 □ 검사



# Naver 메인 페이지 검색

## Naver 메인 페이지 검색



# Naver 메인 페이지 검색

## Naver 메인 페이지 검색

```
#naverSearch1.py
from selenium import webdriver as
wd url = 'http://www.naver.com'
#chrome driver load
driver =
wd.Chrome(executable_path='chromedriver.exe')
#page load
driver.get(url)
element =
driver.find_element_by_id('query') keyword
= '도루코 면도기'
element.send_keys(keyword)
elementBtn =
driver.find_element_by_id('search_btn')
elementBtn.click()
```

```
▼<span class="green_window">
  <input id="query" name="query" type="text" title="검색어 입력" maxlength="255" class="input_text" tabindex="1" accesskey="s" style="ime-mode:active;" autocomplete="off" onclick="document.getElementById('fbm').value=1;" value>
</span>
```

```
...
  <button id="search_btn" type="submit" title="검색" class="sch_smit" onmouseover="this.className='sch_smit over'" onmousedown="this.className='sch_smit down'" onclick="clickcr(this,'sch.action','','','event');" > == :>
    <span class="blind">검색</span>
    <span class="ico_search_submit"></span>
  </button>
```

03

## Selenium을 이용한 여행정보 크롤링

## ■ 인터파크 사이트 검색

```
#interpartSearch0.py
from selenium import webdriver as wd

url = 'http://tour.interpark.com'

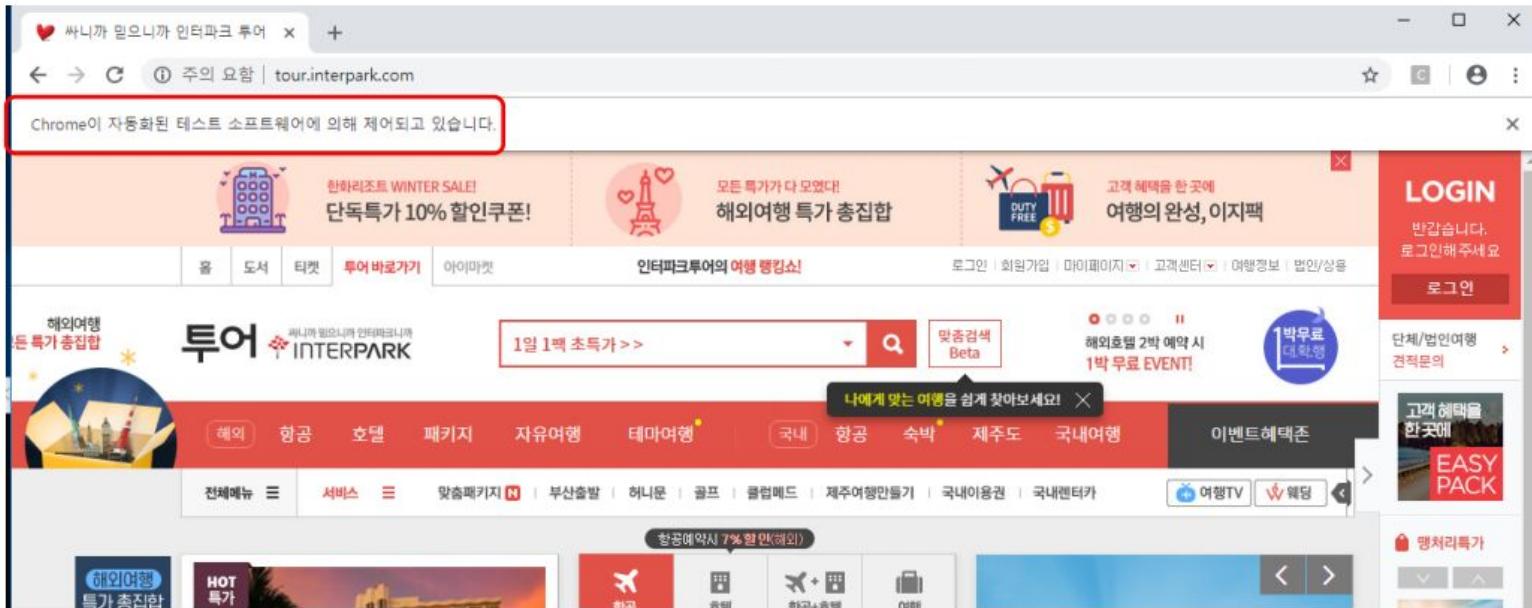
#chrome driver load
driver = wd.Chrome(executable_path='chromedriver.exe')

#page load
driver.get(url)
```

# 파이썬 웹 크롤링

## ■ 사이트 주소 수집

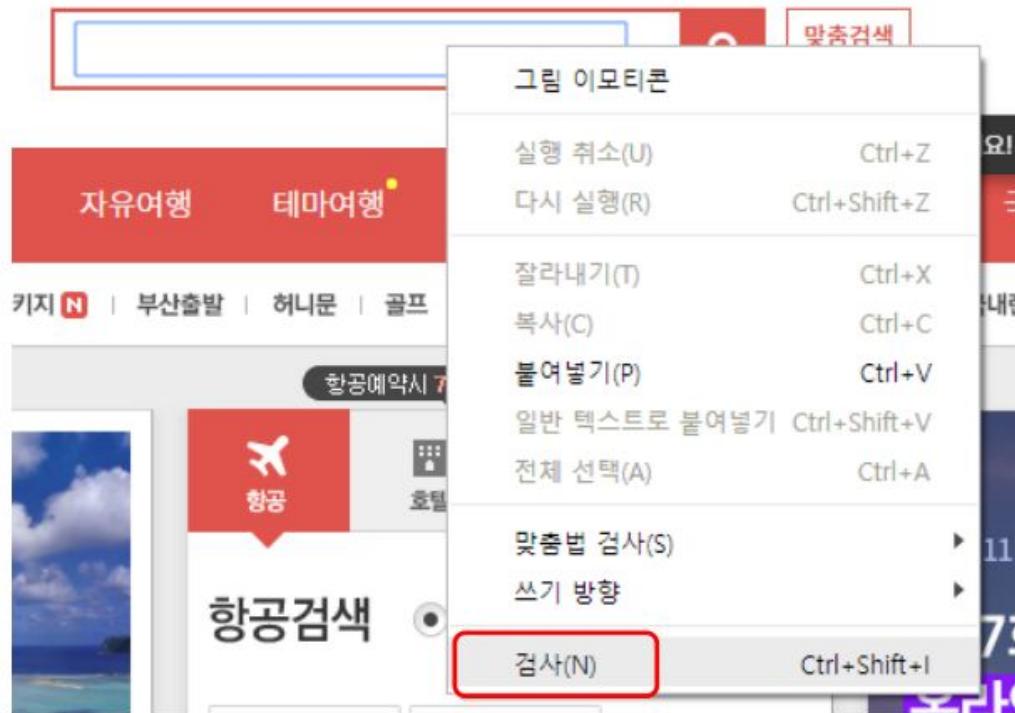
### ▶ 인터파크 투어 url



# 파이썬 웹 크롤링

## ■ 검색 메뉴의 페이지 소스 분석

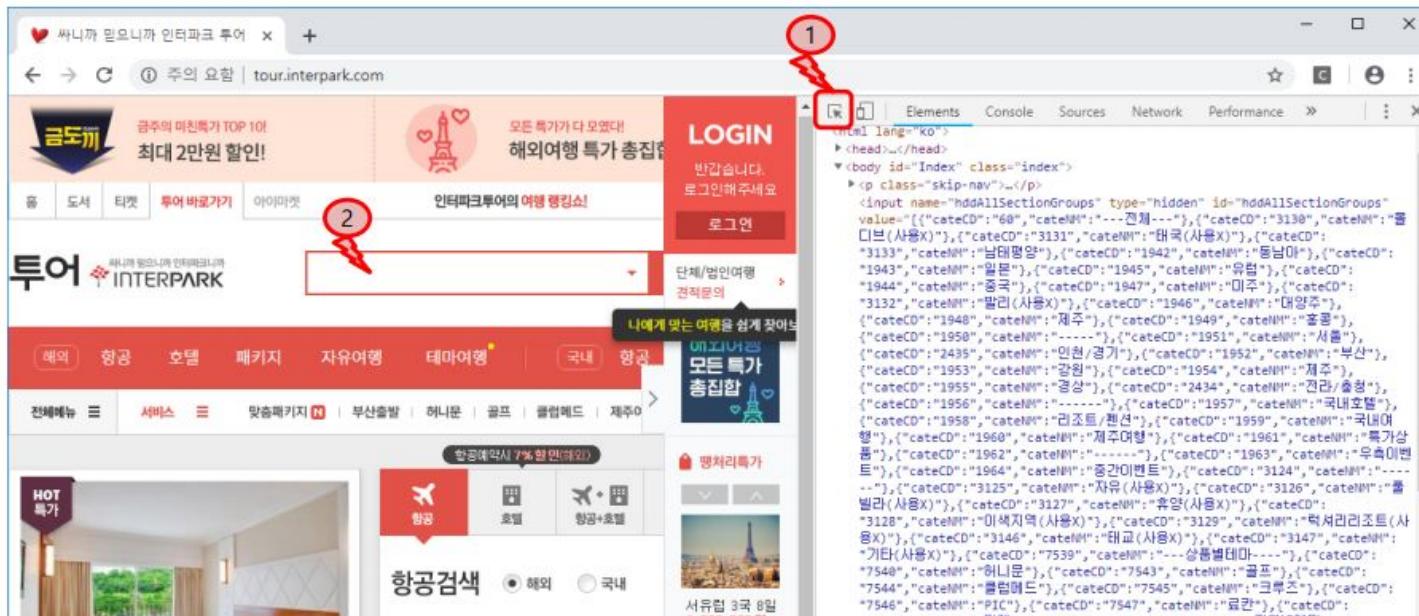
▶ 검색창 우클릭 → 검사



# 파이썬 웹 크롤링

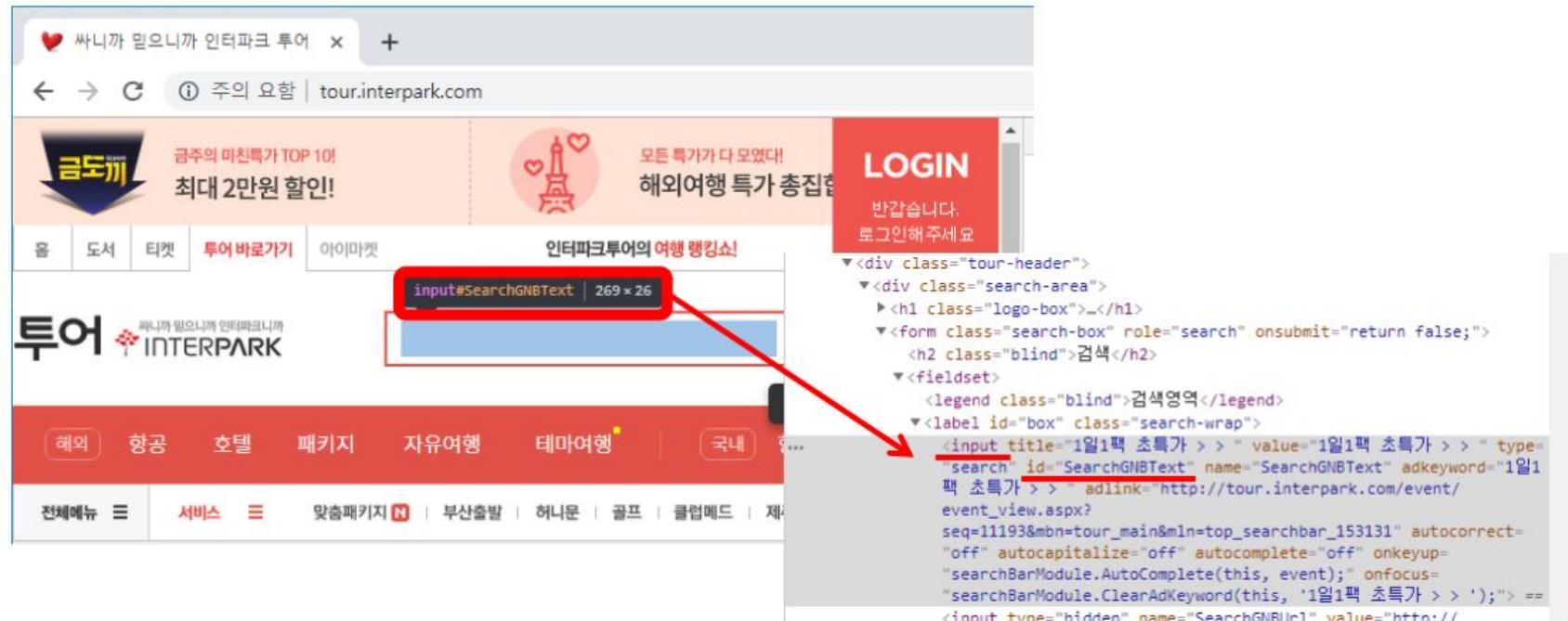
## 검색 메뉴의 페이지 속스 분석

### ▶ 검색창 우클릭 → 검사



# 파이썬 웹 크롤링

## 검색 메뉴의 페이지 소스 분석



The screenshot shows the homepage of tour.interpark.com. At the top, there is a search bar with the placeholder text "검색어를 입력하세요". Below the search bar, the page features various promotional banners and a navigation menu. The search input field is highlighted with a red box and a red arrow points to its corresponding HTML code in the bottom right corner.

```
<input id="SearchGNBText" name="SearchGNBText" adkeyword="1일1  
팩 초특가 > > " type="search" value="1일1팩 초특가 > > " />  
<input id="SearchGNBText" name="SearchGNBText" adkeyword="1일1  
팩 초특가 > > " type="search" value="1일1팩 초특가 > > " />  
<input id="SearchGNBText" name="SearchGNBText" adkeyword="1일1  
팩 초특가 > > " type="search" value="1일1팩 초특가 > > " />
```

## ■ 검색 메뉴의 페이지 소스 분석

### ▶ id → class → 종속관계 순서로 분석

- 검색 메뉴의 id는 페이지 내부에서 유일한 것 확인 → id 사용
  - 마우스로 검색 창을 클릭 후 Ctrl + F

```
▼<label id="box" class="search-wrap">
  <input title="제 7회 온라인 박람회 OPEN" value="제 7회 온라인 박람회 OPEN" type="search" id="SearchGNBText" name="SearchGNBText" adkeyword="제 7회 온라인 박람회 OPEN" adlink="http://tourexpo.interpark.com/?expoid1=pc&expoid2=2018-7_expo_main&mbn=tour_main&mln=top_searchbar_153131" autocorrect="off" autocapitalize="off" autocomplete="off" onkeyup="searchBarModule.AutoComplete(this, event); onfocus="searchBarModule.ClearAdKeyword(this, '제 7회 온라인 박람회 OPEN');">
  <input type="hidden" name="SearchGNBUrl" value="http://tourexpo.interpark.com/?expoid1=pc&expoid2=2018-7_expo_main&mbn=tour_main&mln=top_searchbar_153131" id="SearchGNBUrl" style="display:none;">
  ▶<iframe style="width: 0; height: 0; border:0; border:none;" id="searchFrame" src="//search-tour.interpark.com/SearchTour/recentlyKeyword.html">...</iframe>
</label>
<!-- 자동완성 및 최근인기검색어 -->
<!-- 자동완성영역 -->
<div class="autoComBox" role="combobox" aria-label="자동완성검색어" style="display: none;"></div>
```

html body#index.index div#dHead div.tour-header div.search-area h1.logo-box

SearchGNBText

1 of 1

Cancel

## ■ 검색 키워드 입력

```
#interpartSearch1.py
from selenium import webdriver as wd

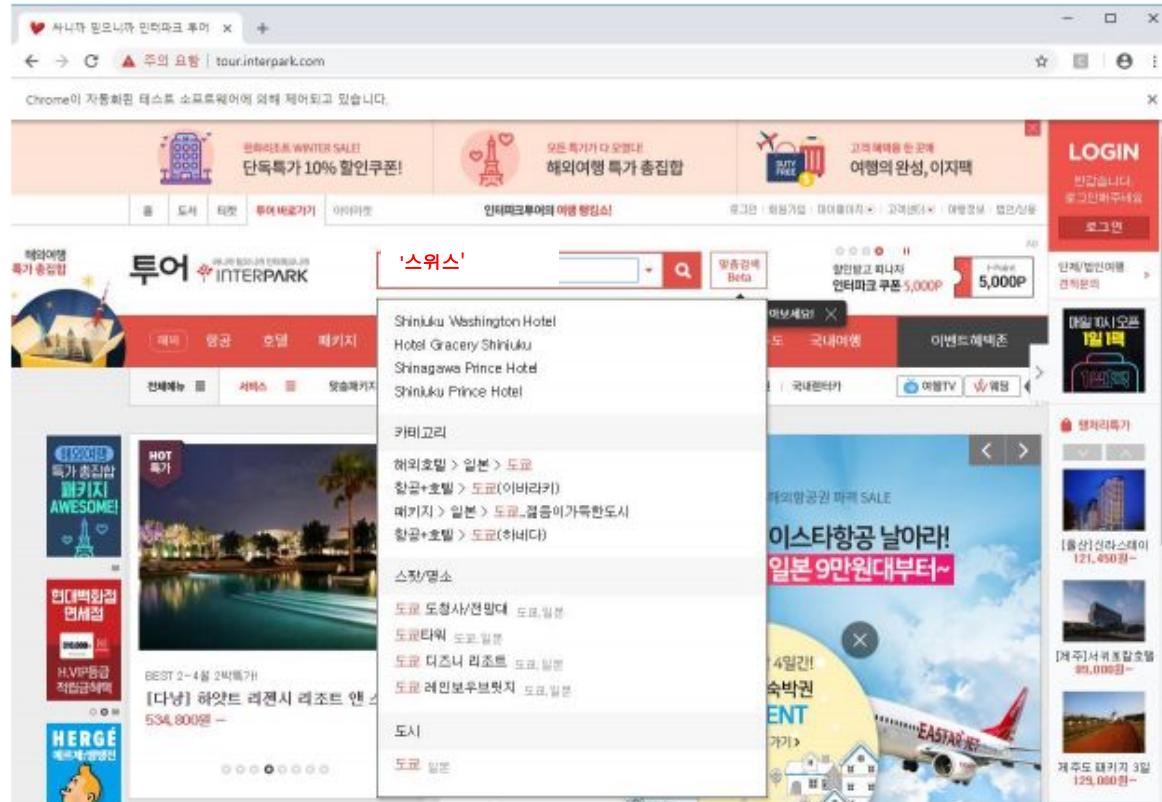
url = 'http://tour.interpark.com'
keyword = '스위스'

driver = wd.Chrome(executable_path='chromedriver.exe') #chrome driver load
driver.get(url) #page load

#검색 창 선택 및 키워드 입력
element = driver.find_element_by_id('SearchGNBText')
element.send_keys(keyword)                                #검색 창 선택
# 검색 창에 keyword 입력
```

# 파이썬 웹 크롤링

## 검색 키워드 입력 ▶ 실행 화면



## ■ 검색 버튼 페이지 소스 분석

- ▶ 클래스 명 : search-btn, 중복 없음

```
...      <button class="search-btn" type="button" onclick="searchBarModule.ClickForSearch();" title="검색">검색</button> == $0
    ▶ <button class="recent-btn" type="button" onclick="searchBarModule.HotKeyUpdate();" title="최근검색어">...</button>
  </fieldset>
</form>
▶ <div class="spot-banner-box">...</div>
▶ <div id="GNBRollingBanner" class="banner-box doing">...</div>
▶ <div id="GNBFloatingBanner" class="left-side-banner">...</div>
</div>
▶ <div class="menu-area">...</div>
</div>
<script src="//www.interpark.com/static/js/egs/egs_common.js"></script>
</div>
<hr class="access">
▶ <div id="body">...</div>
html body#index.index div#dHead div.tour-header div.search-area form.search-box fieldset button.search-btn
search-btn
```

1 of 1

## ■ 검색 버튼 클릭

```
#interpartSearch2.py
from selenium import webdriver as wd

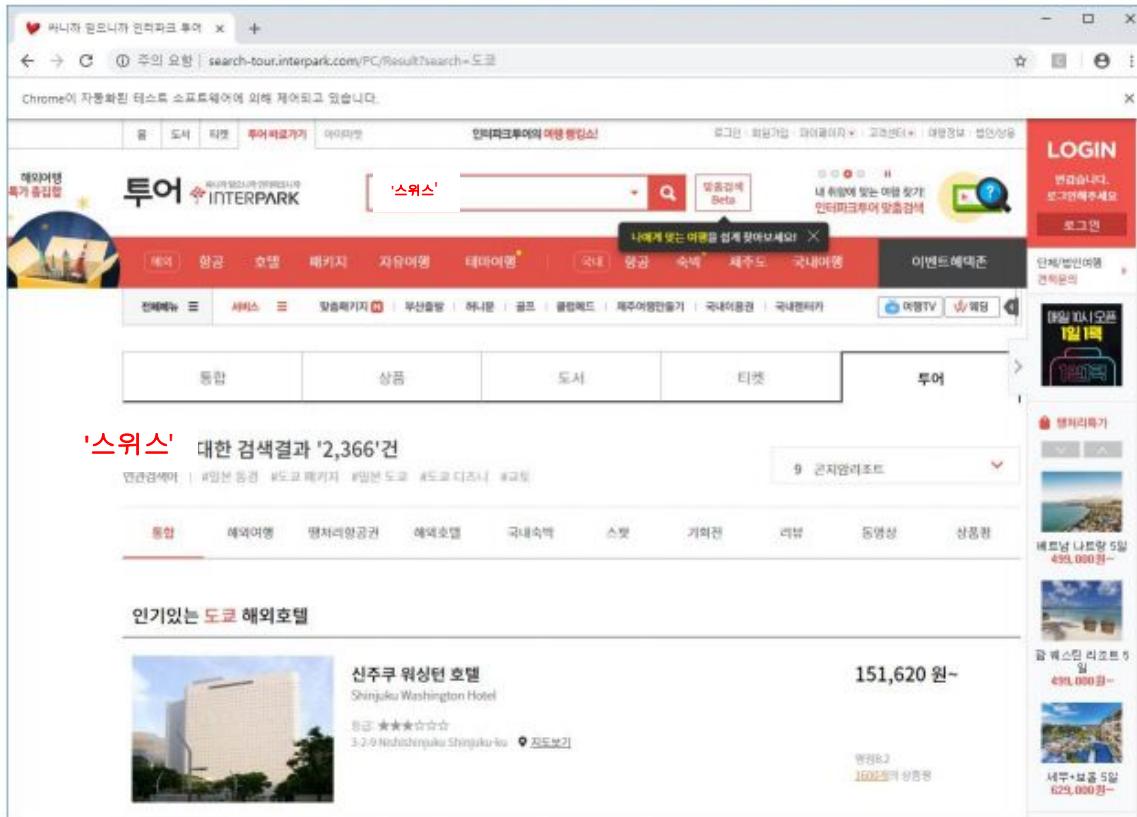
url = 'http://tour.interpark.com'
serachBox_id = 'SearchGNBText'
keyword = '스위스'
searchBtn_class = 'search-btn'

driver = wd.Chrome(executable_path='chromedriver.exe')      #크롬 실행
driver.get(url)                                              #인터파크 접속
#검색창 선택 및 키워드 입력
driver.find_element_by_id(serachBox_id).send_keys(keyword)
driver.find_element_by_class_name(searchBtn_class).click()    #검색 버튼 클릭
```

# 파이썬 웹 크롤링

## 검색 버튼 클릭

### ▶ 실행 결과



## ■ 페이지 로드 및 대기 시간

- ▶ 페이지가 모두 불려올 때 까지 기다려줘야 함
  - <https://selenium-python.readthedocs.io/waits.html>

## ■ Selenium Waits

- ▶ Explicit Waits : 명시적 대기
  - 특정 값(id, class 등)이 불려올 때까지 대기
- ▶ Implicit Waits : 암시적 대기
  - 시간을 정해놓고 대기
  - 시간이 다 지나기 전에 로드 완료 시 생략 함

## ■ 암시적 대기 적용

### ▶ 소스코드

```
from selenium import webdriver as wd

url = 'http://tour.interpark.com'
serachBox_id = 'SearchGNBText'
keyword = '스위스'
searchBtn_class = 'search-btn'

#크롬 실행
driver = wd.Chrome(executable_path='chromedriver.exe')

#페이지 로드 대기시간 10초로 설정(10초 이전에 페이지 로드 완료시 무시)
driver.implicitly_wait(10)

#인터파크 투어 접속
driver.get(url)

#검색창 선택 및 키워드 입력
driver.find_element_by_id(serachBox_id).send_keys(keyword)

#검색 버튼 클릭
driver.find_element_by_class_name(searchBtn_class).click()
```

## ■ 해외여행 정보 보기 소스코드 분석

The screenshot shows the homepage of the Interpark Travel (투어) website. At the top, there is a navigation bar with links for Home, Book, Ticket, Travel Direct, and My Market. The main search bar contains the text '스위스' (Switzerland). To the right of the search bar is a 'Matching Search Beta' button. Above the search bar, there is a banner for the '제7회 온라인 여행박람회' (7th Online Travel Expo) and a promotional message for NH Nonghyup Card users. Below the search bar, there is a 'Travel Recommendation' box with the text '나에게 맞는 여행을 쉽게 찾아보세요!' (Find travel that suits you easily!). The main menu includes categories like Domestic, International, Hotel, Package, Free Travel, and Theme Travel. The search results for '스위스' show 2,215 results, with the top result being '5 파라다이스시티' (Paradise City). The results are categorized into sections such as Travel, Hotel, Package, and Events.

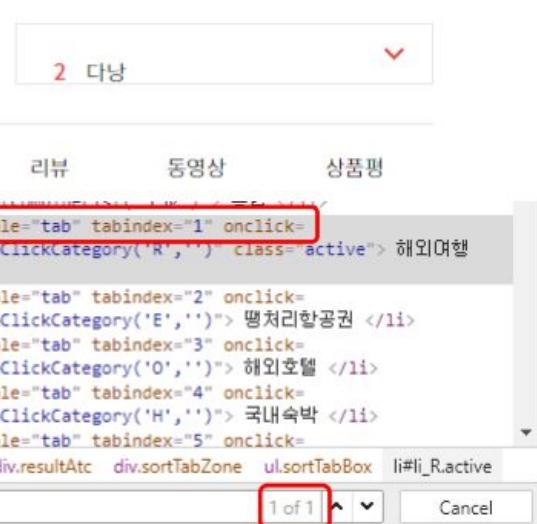
# 파이썬 웹 크롤링

## ■ 해외여행 정보 보기

- ▶ 검사 창에서 번개표시 클릭 후
- ▶ 웹페이지에서 검사하기 원하는 곳을 선택

### '스위스'에 대한 검색결과 '157'건

연관검색어 | #일본 동경 #도쿄 패키지 #일본 도쿄 #도쿄 디즈니 #교토



# 파이썬 웹 크롤링

## 해외여행 정보만 나타내기

### ▶ 소스코드

```
#interpartSearch3.py
from selenium import webdriver as wd
import time

url = 'http://tour.interpark.com'
serachBox_id = 'SearchGNBText'
keyword = '스위스'
searchBtn_class = 'searchn-btn'
infoBtn_id = 'li_R'

#크롬 실행
driver = wd.Chrome(executable_path='chromedriver.exe')
#페이지 로드 대기시간 10초로 설정(10초 이전에 페이지 로드 완료 시 무시)
driver.implicitly_wait(10)
#인터파크 투어 접속
driver.get(url)
#검색 창 선택 및 키워드 입력
driver.find_element_by_id(serachBox_id).send_keys(keyword)
#검색 버튼 클릭
driver.find_element_by_class_name(searchBtn_class).click()
#해외 여행 정보 클릭
driver.find_element_by_id(infoBtn_id).click()
```



```
<div class="sortTabZone">
  <ul class="sortTabBox" role="tablist">
    <li role="tab" tabindex="0" class="active" onclick="searchModule.SetCombinelist('도쿄')"><a>통합</a></li>
    <li id="li_R" role="tab" tabindex="1" onclick="searchModule.OnClickCategory('R','')"><a>해외여행</a> == $0
    ...
    <li id="li_E" role="tab" tabindex="2" onclick="searchModule.OnClickCategory('E','')"><a>필자리항공권</a>
    <li id="li_O" role="tab" tabindex="3" onclick="searchModule.OnClickCategory('O','')"><a>해외호텔</a>
    <li id="li_H" role="tab" tabindex="4" onclick="searchModule.OnClickCategory('H','')"><a>국내숙박</a>
    <li id="li_S" role="tab" tabindex="5" onclick="searchModule.OnClickCategory('S','')"><a>스팟</a>
    <li id="li_CB" role="tab" tabindex="6" onclick="searchModule.OnClickCategory('C','B')"><a>리뷰</a>
    <li id="li_CR" role="tab" tabindex="7" onclick="searchModule.OnClickCategory('C','R')"><a>상품평</a>
    <li id="li_V" role="tab" tabindex="8" onclick="searchModule.OnClickCategory('V','')"><a>기획전</a>
    <li id="li_CM" role="tab" tabindex="9" onclick="searchModule.OnClickCategory('C','M')"><a>동영상</a>
  </ul>
</div>
```

# 파이썬 웹 크롤링

## 해외여행 정보만 나타내기

### ▶ 실행 결과

The screenshot shows the search results for '스위스' (Switzerland) on the Interpark Travel website. The results are filtered to show only travel information, as indicated by the highlighted '해외여행' (Overseas Travel) tab in the navigation bar. The search results include various travel packages and deals, such as '1.19-12.16 티파크투어 여행 박람회' (1.19-12.16 TIPAKC Travel Expo) and '슈퍼세일' (Super Sale). The results are presented in a grid format with columns for '동합' (Donghae), '상품' (Product), '도서' (Book), and '티켓' (Ticket). The results are limited to 50 items.

## ▶ 실행 결과 확인

```
print("x = {}, y = {}".format(22.2, 44))
```

```
stringValue = "x = {}, y = {}".format(23.4, 44.1)
print(stringValue)
```

```
for page in range(1, 5):
    print("page is {}".format(page))
```

## ■ 해외여행 정보만 나타내기

### ▶ 실행 결과

The screenshot shows the homepage of the Interpark Travel website (search-tour.interpark.com/PC/Result?search=도쿄&code1=R&code2=). The search term '스위스' (Switzerland) is entered in the search bar. The results are displayed in a grid format, with the first few items being travel-related. The interface includes various filters and categories such as '국내', '항공', '숙박', '제주도', '국내여행', and '이벤트 혜택존'. The overall layout is clean and modern, with a red and white color scheme.

## ■ 해외여행 정보 페이지 목록

해외여행 (122)



페이지

[홈쇼핑따라잡기]동경/하코네 온천 3일(전일관광/준특급/3대특식)

여행 기간: 2박3일 부산도쿄하코네사이타마요코하마부산

출발 가능 기간: 2019.07.21~2019.07.21



페이지

[판매 no.1]동경/요코하마/하코네 OR 디즈니 3일(전일관광/트윈룸)

여행 기간: 2박3일 부산도쿄요코하마부산

출발 가능 기간: 2019.07.14~2019.12.31



# 파이썬 웹 크롤링

## 해외여행 정보 페이지 목록 소스 분석

- ▶ 검색 창에서 번개표시 클릭 후
- ▶ 페이지 변경하는 곳을 선택



- ▶ searchModule.SetCategoryList(페이지번호,) 함수
  - 위 함수 실행으로 페이지 변경

```
Elements Console Sources Network Performance > 0 2 : X
Select an element in the page to inspect it. Ctrl + Shift + C
<head>...
<body>
  <div id="dHead">...
    <script type="text/javascript" id="wcs_naver.net/wcslog.js"></script>
    <script type="text/javascript" id="tour.interpark.com/scripts/views/common/egs_common.js"></script>
  </div>
  <div>
    <ul>
      <li role="button" tabindex="1" onclick="searchModule.SetCategoryList(1, '')>1</li>
      <li class="active" role="button" tabindex="2">2</li> == $0
      <li role="button" tabindex="3" onclick="searchModule.SetCategoryList(3, '')>3</li>
      <li role="button" tabindex="4" onclick="searchModule.SetCategoryList(4, '')>4</li>
      <li role="button" tabindex="5" onclick="searchModule.SetCategoryList(5, '')>5</li>
      <li role="button" tabindex="6" onclick="searchModule.SetCategoryList(6, '')>6</li>
      <li role="button" tabindex="7" onclick="searchModule.SetCategoryList(7, '')>7</li>
      <li role="button" tabindex="8" onclick="searchModule.SetCategoryList(8, '')>8</li>
      <li role="button" tabindex="9" onclick="searchModule.SetCategoryList(9, '')>9</li>
      <li role="button" tabindex="10" onclick="searchModule.SetCategoryList(10, '')>10</li>
    </ul>
    <button type="button" class="nextBtn" onclick="searchModule.SetCategoryList">next</button>
  </div>
</body>
```

7 of 14 Cancel

## ■ 해외여행 정보 모든 페이지에 접근

### ▶ 소스코드

- 이전의 코드에 아래의 코드를 마지막 부분에 추가

```
#각각의 페이지에 접근
for page in range(1,17):
    driver.execute_script(
        "searchModule.SetCategoryList({}, '')".format(page)
        # {}에 page값을 넣은 문자열
        #searchModule.SetCategoryList(1, '') ~ searchModule.SetCategoryList(13, '')
    )
    time.sleep(3)#페이지 이동 스크립트 실행 대기 시간
    print("{}번째 페이지로 이동".format(page))
```

## ■ 해외여행 정보 모든 페이지에 접근

### ▶ 실행 결과

```
import time
for page in range(1,14): # 1~13 페이지
    driver.execute_script( "searchModule.SetCategoryList({}, '')".format(page) )
    time.sleep(3)
    print("{} 페이지로 이동".format(page))
```

1 페이지로 이동  
2 페이지로 이동  
3 페이지로 이동  
4 페이지로 이동  
5 페이지로 이동  
6 페이지로 이동  
7 페이지로 이동  
8 페이지로 이동  
9 페이지로 이동  
10 페이지로 이동  
11 페이지로 이동  
12 페이지로 이동

# 파이썬 웹 크롤링

## 각각의 정보에 접근

### ▶ panelZone > oTravelBox > boxList > li

```
▼<div class="panelZone" style="display: block;">
  ▶<div class="recommService">...</div>
  ▶<div id="tempdiv" style="text-align: center; display: none;">...</div>
  ▶<div class="oTravelBox">...</div>
  ▶<div class="pageNumBox">...</div>
</div>
</div>

▼<div class="panelZone" style="display: block;">
  ▶<div class="recommService">...</div>
  ▶<div id="tempdiv" style="text-align: center; display: none;">...</div>
  ▶<div class="oTravelBox">
    ▶<h4 class="boxTitle">...</h4>
    ▶<div class="boxFilter">...</div>
    ▶<ul class="boxList">
      ▶<li class="boxItem">...</li>
      ▶<li class="boxItem">...</li>
    </ul>
  </div>
  ▶<div class="pageNumBox">...</div>
</div>
</div>
```

```
▼<li class="boxItem">
  ▶<a href="javascript:;" onclick="searchModule.OnClickDetail('http://tour.interpark.com/goods/detail/?BaseGoodsCd=B6010973','')" class="detailBtn" data-click="target">...</a>
  ▶<div class="boxTables">
    ▶<div class="icon-row" data-row>...</div>
    ▶<div class="title-row" data-row>
      ▶<div data-cell>
        <h5 class="proTit" aria-label=" [0]스타항공 ] 도쿄 자유여행 3박 4일 " onclick="searchModule.OnClickDetail('http://tour.interpark.com/goods/detail/?BaseGoodsCd=B6010973','')" data-click="link">[0]스타항공 ] 도쿄 자유여행 3박 4일 </h5> == $0
        <p class="proSub">여유롭게 즐기는 도쿄</p>
      </div>
    </div>
    ▶<div data-cell>...</div>
  </div>
  ▶<div class="info-row" data-row>...</div>
</li>
▶<li class="boxItem">...</li>
```

# 파이썬 웹 크롤링

## 각각의 정보에 접근

### ▶ panelZone > oTravelBox > boxList > li

```
▼<div class="panelZone" style="display: block;">
  ▶<div class="recommService">...</div>
  ▶<div id="tempdiv" style="text-align: center; display: none;">...</div>
  ▶<div class="oTravelBox">...</div>
  ▶<div class="pageNumBox">...</div>
</div>
</div>

▼<div class="panelZone" style="display: block;">
  ▶<div class="recommService">...</div>
  ▶<div id="tempdiv" style="text-align: center; display: none;">...</div>
  ▶<div class="oTravelBox">
    ▶<h4 class="boxTitle">...</h4>
    ▶<div class="boxFilter">...</div>
    ▶<ul class="boxList">
      ▶<li class="boxItem">...</li>
      ▶<li class="boxItem">...</li>
    </ul>
  </div>
  ▶<div class="pageNumBox">...</div>
</div>
</div>
```

```
▼<div class="panelZone" style="display: block;">
  ▶<div class="recommService">...</div>
  ▶<div id="tempdiv" style="text-align: center; display: none;">...</div>
  ▶<div class="oTravelBox">
    ▶<h4 class="boxTitle">...</h4>
    ▶<div class="boxFilter">...</div>
    ▶<ul class="boxList">
      ▶<li class="boxItem">...</li>
      ▶<li class="boxItem">...</li>
    </ul>
  </div>
  ▶<div class="pageNumBox">...</div>
</div>
</div>
```

## ■ 각각의 정보 가져오기

▶ 소스코드(code1.py) body > div.container > div > div > div.panelZone > div.oTravelBox > ul > li:nth-child(1) > div > div.title-row > div:nth-child(1) > h5

#각각의 페이지에 접근

```
for page in range(1,17):
    driver.execute_script(
        "searchModule.SetCategoryList({}, '')".format(page)
        # {}에 page값을 넣은 문자열
        #searchModule.SetCategoryList(1, '') ~ searchModule.SetCategoryList(16, '')
    )
    time.sleep(2)#페이지 이동 스크립트 실행 대기 시간
    print("{}번째 페이지로 이동".format(page))
    boxItems = driver.find_elements_by_css_selector('.panelZone>.oTravelBox>.boxList>li')
    for li in boxItems:
        print('상품명', li.find_element_by_css_selector('h5.proTit').text )
        print('가격',li.find_element_by_css_selector('.proPrice').text.split('원')[0])
```

추가된 내용

# 파이썬 웹 크롤링

## 각각의 정보 가져오기

### ▶ 실행 화면

상품명 ★ 마사지 1+1 ★ [자유+관광] [렌트카+오마후 섬일주포함] ★ 코나(빅아일랜드) 2박+오마후 3박 ★ 5박 7일  
가격 2,319,000  
상품명 [패키지+] [오마후 섬일주포함] [준특급] 힐튼 가든 인 스탠다드룸 4박 6일  
가격 2,018,600  
상품명 ★ 하코네 ★ 마우라타치바나 자유여행 3일  
가격 647,000  
15번째 페이지로 이동  
상품명 [판매 no. 1] 동경/요코하마/하코네 OR 디즈니 3일 (전일관광/트윈룸)  
가격 949,000  
상품명 [3.6.9 할인~유후♪] [▶ 부산출발] [후지산▲X온천욕] 동경/하코네/후지산/마사쿠사/카와구치 3일  
가격 1,039,800  
상품명 ★ 디즈니 할인+1일 자유+옵션★동경/오다이바/오오에도온천 3일 [세미더블]  
가격 759,000  
상품명 [ONE CITY] 파리 자유여행 5일  
가격 1,437,100  
상품명 [패키지+]/스페셜 동경특가] 동경/요코하마 3일 (준특급/정더블/1일 자유/나이트투어 포함)  
가격 569,000  
상품명 [가성비UP 후지산/하코네] 동경/후지산/하코네 온천 4일 (하코네로프웨이/유람선/오오에도온천 포함)  
가격 1,119,000  
상품명 [이스타항공 오후출발] 동경/이바라키/닛코 국립공원 4일 (트윈/펜타이코(명란젓) 풍물 견학 포함)  
가격 1,041,800  
상품명 ★ 렌트카포함★▶ MY HAWAII STORY◀ 폴리레이 인 악스프레스 와이키키 시티뷰 4박 6일  
가격 1,814,000  
상품명 ★ 렌트카포함★ [특급] [하와이 자유여행]▶▶ 코트야드 메리엇 주니어 스위트룸 4박 6일  
가격 2,014,000  
상품명 ★ 렌트카포함★ [초특급] ★ LUXURY HAWAII ★ 쇼라톤 와이키키 오션프론트룸 4박 6일  
가격 2,514,000  
16번째 페이지로 이동  
상품명 ★ 렌트카포함★ [초특급] ★ LUXURY HAWAII ★ 하얏트 리젠시 와이키키 오션뷰 4박 6일  
가격 2,414,000  
상품명 ★ 리턴연장 가능★ 관광과 휴양을 동시에 뉴욕+마이애미 자유여행 8일  
가격 2,690,000  
상품명 [베이직 하와이/일급] 와이키키 리조트 스탠다드룸 4박 6일  
가격 1,718,600  
상품명 □ 제주출발 □ [실속여행] 동경/하코네/요코하마 4일  
가격 890,600  
상품명 [대구출발] ★ 모두단독★ 동경 하코네 오다이바+모리노유 온천욕  
가격 495,800  
상품명 [도쿄 시티투어] 도쿄 일일 완전정복 (편도/왕복 선택가능)  
가격 50,900  
상품명 [도쿄] 신주쿠 출발~ 에노시마&가마쿠라 일일투어  
가격 70,200

04

## 유튜브 크롤링

# Youtube 페이지 접속

- Selenium을 이용해서 Youtube 백종원 요리비책 채널 (<https://www.youtube.com/channel/UCyn-K7rZLXjGI7VXGwellcA/videos>)에 접속

```
In [1]: from selenium import webdriver
from selenium.webdriver.common.keys import Keys
#크롬 드라이버를 실행
driver = webdriver.Chrome("chromedriver.exe")
#백종원 요리비책 페이지로 이동
driver.get('https://www.youtube.com/channel/UCyn-K7rZLXjGI7VXGwellcA/videos')
```

백종원의 요리비책 - YouTube

youtube.com/channel/UCyn-K7rZLXjGI7VXGwellcA/videos

Chrome이 자동화된 테스트 소프트웨어에 의해 제어되고 있습니다.

YouTube KR

검색

로그인

백종원의 요리비책  
Paik's Cuisine

구독자 282만명

구독

홈 동영상 재생목록 커뮤니티 채널 정보

업로드한 동영상 모두 재생 정렬 기준

최근 본 동영상

집에 있는 그 치즈로 치즈토스트 10:30

백종원의 참사미야기 35-3

프렌차이즈 '레시피'의 비밀!! 6:10

고들어케밥 아기고들어케밥 13:32

디키아스탄을 고들어케밥의 정체 6:52

# 화면 끝까지 스크롤

- Youtube 페이지는 페이지가 없고 화면 끝까지 스크롤 후 페이지의 내용을 스크롤 해야함
- `body.send_keys(Keys.END)`
  - 화면에 END 키를 입력(화면에 스크롤됨)
- `driver.execute_script("return document.documentElement.scrollHeight")`
  - 현재 화면의 크기
  - 스크롤키를 누렸을때의 화면크기와 비교해서 화면크기의 변화가 없으면 스크롤을 종료함

In [4]:

```
import time
#한번 스크롤 하고 잠깐 멈출 시간을 저장한 변수 0.5초 멈출것임
SCROLL_PAUSE_TIME=0.5
#화면의 본문인 body태그를 선택해서 body에 대입
body = driver.find_element_by_tag_name("body")

#break 가 실행될때(스크롤 전의 화면크기와 스크롤 후의 화면 크기가 같음) 까지 반복
while True:
    #현재 화면의 길이를 리턴 받아서 last_height에 대입
    last_height=driver.execute_script("return document.documentElement.scrollHeight")
    print("=====")
    #현재 화면의 크기 출력
    print("last_height:", last_height)

    #0~9까지 10번 반복해서 실행
    for i in range(10):
        #body본문에 END키를 입력(화면이 끝까지 스크롤 되고 화면이 끝나짐)
        body.send_keys(Keys.END)
        #SCROLL_PAUSE_TIME에 저장된시간 0.5초 멈춤
        time.sleep(SCROLL_PAUSE_TIME)
        #스크롤 후의 화면의 길이를 리턴 받아서 new_height에 대입
        new_height = driver.execute_script("return document.documentElement.scrollHeight")
        #스크롤 후의 화면의 길이를 출력
        print("new_height:", new_height)
        print("=====")

        #스크롤 후의 화면의 길이 (new_height) 스크롤 전의 화면의 길이 (last_height)
        #가 같다면
        if new_height == last_height:
            break #반복 종료
```

화면 끝까지 스크롤 되는지 확인



# 페이지 소스 리턴

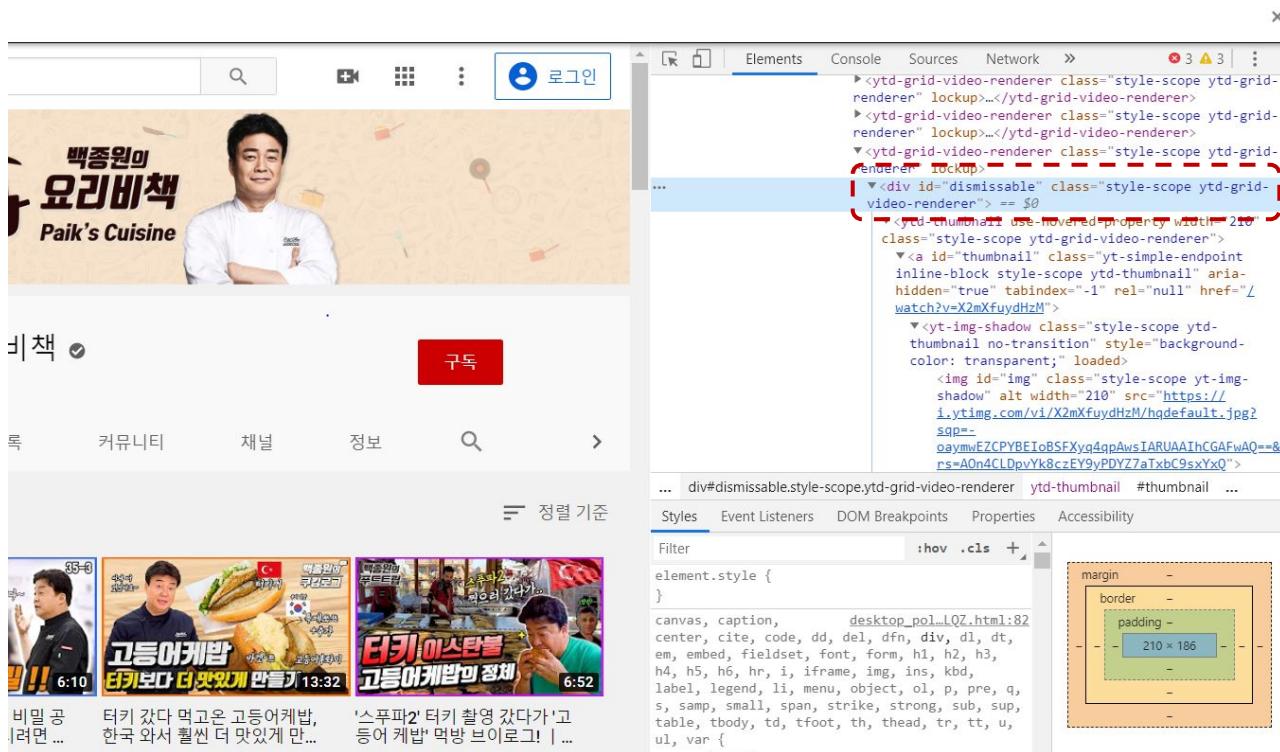
```
In [5]: #Youtube 페이지 소스를 page에 대입
page = driver.page_source
page
```

```
w.performance ? w.performance.timing.responseStart : null);var isPrerender = (d.visibilityState || d.webkitVisibilityState) == 'pre
render';var vName = (!d.visibilityState && d.webkitVisibilityState)? '#webkitvisibilitychange' : '#visibilitychange';if
(isPrerender) {ytcsi.info('#prerender', 1);var startTick = function() {ytcsi.setStart('#dhs');d.removeEventListener(vName, startTi
ck);};d.addEventListener(vName, startTick, false);}if (d.addEventListener) {d.addEventListener(vName, function() {ytcsi.tick('#vc
#');}, false);}var slt = function(el, t) {setTimeout(function() {var n = ytcси.now();el.loadTime = n;if (el.slt) {el.slt();}}, t);};w
._ytRIL = function(el) {if (!el.getAttribute('data-thumb')) {if (w.requestAnimationFrame) {w.requestAnimationFrame(function() {s
lt(el, 0)});} else {slt(el, 16)};}};(window, document);</script><link rel="preload" href="https://i.ytimg.com/generate_204" as="f
etch" />      <script>if (window.ytcси) {window.ytcси.tick("rsbe_dph", null, '#');}</script>#n#n<!--#n@license#nCopyright (c) 201
5 The Polymer Project Authors. All rights reserved.#nThis code may only be used under the BSD style license found at http://polymer.github.io/LICENSE.txt#nThe complete set of authors may be found at http://polymer.github.io/AUTHORS.txt#nThe complete set of contrib
utors may be found at http://polymer.github.io/CONTRIBUTORS.txt#nCode distributed by Google as part of the polymer project is also#n
subject to an additional IP rights grant found at http://polymer.github.io/PATENTS.txt#n--><!--#n@license#nCopyright (c) 2017 The Po
lymer Project Authors. All rights reserved.#nThis code may only be used under the BSD style license found at http://polymer.github.io/LICENSE.txt#nThe complete set of authors may be found at http://polymer.github.io/AUTHORS.txt#nThe complete set of contributors ma
y be found at http://polymer.github.io/CONTRIBUTORS.txt#nCode distributed by Google as part of the polymer project is also#nsubject
to an additional IP rights grant found at http://polymer.github.io/PATENTS.txt#n--><meta charset="UTF-8" /><style css-build-single
="">/*start:custom-style///*end:custom-style///*start:custom-style*/<html>:not(.style-scope){--yt-std-body-300:hsla(0, 0%, 0%, .54);--yt
-std-surface-200:hsl(0, 0%, 98%);--yt-std-surface-300:hsl(0, 0%, 96%);--yt-std-surface-400:hsl(0, 0%, 93%);--yt-primary-color:hsl
(0, 0%, 6.7%);--yt-primary-text-color:hsl(0, 0%, 6.7%);--yt-hovered-text-color:hsla(0, 0%, 6.7%, .8);--yt-secondary-text-color:hsla
(0, 0%, 6.7%, .8);--yt-tertiary-text-color:hsla(0, 0%, 6.7%, .6);--yt-placeholder-text-color:hsla(0, 0%, 6.7%, .6);--yt-border-colo
r:hsl(0, 0%, 62.9%);--yt-commentbox-border-color:hsl(0, 0%, 62.9%);--yt-commentbox-border-color:hsl(0, 0%, 6.7%, .6);--yt-prime
```

```
In [7]: from bs4 import BeautifulSoup
#page에 저장된 태그를 BeautifulSoup을 이용해서 정리해서 리턴
soup = BeautifulSoup(page, 'lxml')
soup
```

```
Out[7]: <!DOCTYPE html>
<html lang="ko-KR" style="font-size: 10px; font-family: Roboto, Arial, sans-serif; " xmlns="http://www.w3.org/1999/xhtml"><head><script data-original-src="/yts/jsbin/player_ias_es6-vflIbPtm6/ko_KR/miniplayer.js" src="/yts/jsbin/player_ias_es6-vflIbPtm6/ko_KR/miniplayer.js"></script><script data-original-src="/yts/jsbin/player_ias_es6-vflIbPtm6/ko_KR/remote.js" src="/yts/jsbin/player_ias_es6-vflIbPtm6/ko_KR/remote.js"></script><meta content="AgJc7xYOCYsCYj0w3o6XqSYYRSBYxaX3lhxUyz+piton3LBVj3pW03DhcWh75fza50ybeMuuGUxvm/2tmDAJsAKAAAABneyJvcmlnaW4i0iJodHRwczovL31vdXR1YmUuY29t0jQ0MyIsImZlYXR1cmUi0iJXZWJDb21wb251bnRzYjA1LCJ1eHBpcnki0jE1NzMW0DQ20TQsImizU3ViZG9tYWluIjpoenVlfQ==" data-exppires="2019-08-15" data-feature="Web Components V0" http-equiv="origin-trial"/><link class="css-httpsgrckisv2scrkasperskylabscomE3E8934C235A4B0E825A35A08381A191abnmaincss" crossorigin="anonymous" href="https://gc.kis.v2.scr.kaspersky-labs.com/E3E8934C-235A-4B0E-825A-35A08381A191/abn/main.css" rel="stylesheet"/><script>var ytcfg = {d: function() {return (window.yt &amp; yt.config_) || ytcfg.data_ || (ytcfg.data_ = {})}},get: function(k, o) {return (k in ytcfg.d()) ? ytcfg.d()[k] : o},set: function() {var a = arguments;if (a.length > 1) {ytcfg.d()[a[0]] = a[1];} else {for (var k in a[0]) {ytcfg.d()[k] = a[0][k];}}};window.ytcfg.set('EMERGENCY_BASE_URL', "##/error_204?t=jerror##u0026level=ERROR##u0026client.name=1##u0026client.version=2.20190926.06.01");
</script><script>window.ShadyDOM = {force: true};window.ShadyCSS = {disableRuntime: true};window.ShadyDOM.preferPerformance = true;window.ShadyDOM.noPatch = true;window.Polymer = (window.Polymer || {});window.Polymer.legacyOptimizations = true;</script><link href="https://s.ytimg.com/yts/img/favicon-vfl18qSY2F.ico" rel="shortcut icon" type="image/x-icon"/><link href="https://s.ytimg.com/yts/img/favicon_32-vfl0ogEID.png" rel="icon" sizes="32x32"/><link href="https://s.ytimg.com/yts/img/favicon_48-vflYjB_Qk.png" rel="icon" sizes="48x48"/><link href="https://s.ytimg.com/yts/img/favicon_96-vflW9Ec0w.png" rel="icon" sizes="96x96"/><link href="https://s.ytimg.com/yts/img/favicon_144-vflILAfaB.png" rel="icon" sizes="144x144"/><title>슈카월드 - YouTube</title><script>var ytcsi = {gt: function(n) {n = (n || '') + 'data_';return ytcsi[n] || (ytcsi[n] = {tick: {}, info: {}})};now: (window.performance &amp; window.performance.timing &amp; window.performance.now &amp; window.performance.timing.navigationStart) ?function() {return window.n
```

# Youtube 동영상 조회



백종원의  
요리비책  
Paik's Cuisine

비책

구독

목록 커뮤니티 채널 정보

정렬 기준

비밀 공... 터키 갔다 먹고온 고등어케밥, 6:10  
터키보다 더 맛있게 만들기 13:32

터키 갔다 먹고온 고등어케밥, 한국 와서 훨씬 더 맛있게 만... 6:52

터키 이스탄불 고등어케밥의 정체 '스포파2 터키 촬영 갔다가 '고등어 케밥' 먹방 브이로그! | ...

Elements

Console Sources Network

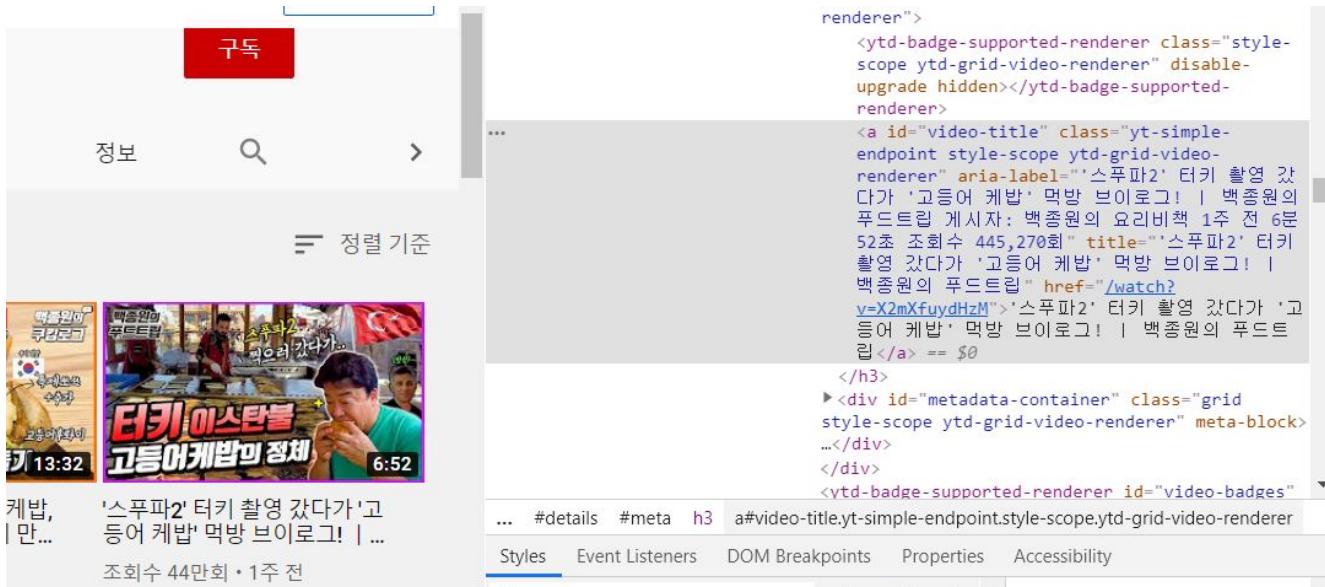
div#dismissable.ytd-grid-video-renderer yt-thumbnail #thumbnail

margin - border - padding - 210 × 186 - - -

```
In [24]: #div태그중 id속성이 dismissable 인 객체 조회
all_videos = soup.find_all(id="dismissable")
all_videos
```

```
Out[24]: [<div class="style-scope ytd-grid-video-renderer" id="dismissable"><yt-thumbnail class="style-scope ytd-grid-video-renderer" use-ho
vered-property="" width="210">
  <a aria-hidden="true" class="yt-simple-endpoint inline-block style-scope yt-thumbnail" href="/watch?v=S9MQ1zjgf7k" id="thumbnail"
  rel="null" tabindex="-1">
    <yt-img-shadow class="style-scope yt-thumbnail no-transition" loaded="" style="background-color: transparent;"></yt-img-shadow>
    <div class="style-scope yt-thumbnail" id="overlays"><yt-thumbnail-overlay-time-status-renderer class="style-scope yt-thumbnail"
    overlay-style="DEFAULT"><yt-icon class="style-scope yt-thumbnail-overlay-time-status-renderer" disable-upgrade="" hidden=""></yt-i
    con><span aria-label="10분" class="style-scope yt-thumbnail-overlay-time-status-renderer">
      10:29
    </span></yt-thumbnail-overlay-time-status-renderer><yt-thumbnail-overlay-now-playing-renderer class="style-scope yt-thumbnail
    "|>
    <span class="style-scope yt-thumbnail-overlay-now-playing-renderer">지금 재생 중</span>
  </yt-thumbnail-overlay-now-playing-renderer></div>
  <div class="style-scope yt-thumbnail" id="mouseover-overlay"></div>
  <div class="style-scope yt-thumbnail" id="hover-overlays"></div>
</a>
</yt-thumbnail><div class="style-scope ytd-grid-video-renderer" id="details"><div class="style-scope ytd-grid-video-renderer" id
="meta"><h3 class="style-scope ytd-grid-video-renderer"><yt-badge-supported-renderer class="style-scope ytd-grid-video-renderer" di
```

# Youtube 제목 조회



구독

정보

정렬 기준

13:32 6:52

케밥, '스푸파2' 터키 촬영 갔다가 '고등어 케밥' 먹방 브이로그! | ...

조회수 44만회 • 1주 전

```
renderer">
<ytd-badge-supported-renderer class="style-scope ytd-grid-video-renderer" disable-upgrade hidden></ytd-badge-supported-renderer>
<a id="video-title" class="yt-simple-endpoint style-scope ytd-grid-video-renderer" aria-label="스푸파2' 터키 촬영 갔다가 '고등어 케밥' 먹방 브이로그! | 백종원의 푸드트립 게시자: 백종원의 요리비책 1주 전 6분 52초 조회수 445,270회" title="스푸파2' 터키 촬영 갔다가 '고등어 케밥' 먹방 브이로그! | 백종원의 푸드트립</a> == $0
</h3>
▶ <div id="metadata-container" class="grid style-scope ytd-grid-video-renderer" meta-block>
...</div>
</div>
<ytd-badge-supported-renderer id="video-badges" ...>
... #details #meta h3 a#video-title.yt-simple-endpoint.style-scope.ytd-grid-video-renderer
Styles Event Listeners DOM Breakpoints Properties Accessibility
```

# 유튜브 제목 크롤링

```
In [25]: title_list = []
#all_videos에서 video 하나를 리턴
for video in all_videos:
    #video에서 id가 video-title 속성을 갖는 객체 조회 (제목)
    title = video.find(id="video-title")
    #title.text.strip() 공백을 제거하고 len() 글자수가 0보다 큰 제목이면
    if len(title.text.strip())>0:
        #리스트에 추가
        title_list.append(title.text)

print(title_list)
print(len(title_list))
```

['돼지고기 사려 마트 다녀왔습니다! | 백종원의 쿠킹로그 번외편', "'골목식당 때문에 매장 매출이 줄었어요!?' | 백종원의 장사이야기", '참 쉬운 김밥만들기, 김밥 A~Z까지~! | 백종원의 쿠킹로그', '내 지출을 머릿속으로 감안할 수 있는 식당이 잘돼요. | 백종원의 장사이야기', '햄, 치즈, 달걀 환상조화 국민 간식 토스트, 신박하게 쉬운 방법~! | 백종원의 쿠킹로그', '초코초코초코 브라우니 & 건더기 생생 사과 흥차! 특집 사과대잔치 | 백종원의 쿠킹로그', '프랜차이즈 본사와 가맹점의 역할! | 백종원의 장사이야기', '불맛 가득 볶음밥을 더 맛있게! 우리집이 볶음밥 맛집! | 업그레이드 볶음밥 | 백종원의 쿠킹로그', '불맛 가득한 달걀볶음밥! 맛없으면 이상한 거죠~ | 백종원의 쿠킹로그', '가을 햇무가 좋은 시기입니다. 가을 무로 만드는 간단한 무생채와 맛있는 무생채 비빔밥입니다! | 백종원의 쿠킹로그', '한 지붕 두 가게, 족발집과 세계맥주 전문점! | 백종원의 장사이야기', '에어프라이어로 쉽게 촉촉한 사과파이와 초간단 사과 고르곤졸라 피자! [특집 사과 대잔치] | 백종원의 쿠킹로그', "비 오는 날, 막걸리 땡기시죠? 안주로 잘 어울리는 '애호박전' 알려드릴게요. | 백종원의 백종원 레시피", '사과 농가를 응원합니다! 사과조림으로 만들 수 있는 무궁무진한 활용메뉴들을 확인해 보세요! | 백종원의 쿠킹로그', "'돼지갈비 포으기, 유튜브 보고 배웠어요~' 돼지갈빗집 사장님의 고민을 한번 들어보세요 | 백종원의 장사이야기", "먹다 남긴 족발로 만드는 '족발덮밥', 스푸파 방콕 '카오카무' 그 맛 납니다~! | 백종원의 쿠킹로그", "천국의 맛 '치즈토스트', 그리고 사연 있는 초특급 신메뉴 최초 공개! | 백종원의 쿠킹로그", "프랜차이즈 '레시피'의 비밀 공개!! 프랜차이즈화 하시려면 일반 식당과 시작부터 다릅니다.", '터키 갔다 먹고 온 고등어케밥, 한국 와서 훨씬 더 맛있게 만들어 봤습니다. | 백종원의 쿠킹로그', "'스푸파2' 터키 촬영 갔다가 '고등어 케밥' 먹방 브이로그! | 백종원의 푸드트립", '초:초:초 간단 만능양념장으로 만드는 춘천식 닭갈비! 1:1:1:1:1:1 | 백종원의 쿠킹로그', "'백종원 브랜드를 모아 '백종원 탄문'을 만든다면 장사가 잘 될까요?' | 백종원의 장사이야기", "'크리스피'하게 구워내고 '특급소스'와 함께 하는 '치킨 스테이크'. 사랑하는 가족과 함께하세요. | 백종원의 쿠킹로그", "'대학가 와플 매장, 확장으로 매출이 오를 수 있을까요?' 백종원의 장사이야기 35회 첫 번째 질문입니다.", "'잡채 초간단 버전', 그리고 '잡채밥'입니다. 어제꺼 보다 훨씬 쉬워요! 그리고 명절 즐겁게 보내세요 ^\_^ | 백종원의 쿠킹로그", "명절에 빼질 수 없는 '잡채' 이번 추석에는 온 가족이 다함께 만들어 보는 건 어떠세요? | 백종원의 쿠킹로그", "'갈비찜' 의외로 무지하게 쉽습니다. 이번 추석엔

# Youtube 비디오 시간 조회

```
shadow" alt width="210" src="https://  
i.ytimg.com/vi/X2mXfuydHzM/hqdefault.jpg?  
sqp=-  
oaymwEZCPYBEIoBSFXyq4qpAwsIARUAAThCGAFwAQ==&  
rs=AOn4CLDpvYk8czEY9yPDYZ7aTxbC9sxYxQ">  
</yt-img-shadow>  
▼<div id="overlays" class="style-scope ytd-  
thumbnail">  
  ▼<ytd-thumbnail-overlay-time-status-renderer  
  class="style-scope ytd-thumbnail" overlay-  
  style="DEFAULT">  
    ...  
    <span class="style-scope ytd-thumbnail-  
    overlay-time-status-renderer" aria-label=  
    "6분 52초">  
      6:52  
      </span> == $0  
    </ytd-thumbnail-overlay-time-status-  
    renderer>  
    ▶<ytd-thumbnail-overlay-now-playing-renderer  
    class="style-scope ytd-thumbnail">...</ytd-  
    thumbnail-overlay-now-playing-renderer>  
  </div>  
  ...
```

# 유튜브 재생 시간 조회

```
In [26]: video_time_list=[]
#all_videos에 저장된 video정보를 하나를 video에 대입하고 반복
for video in all_videos:
    #비디오 타임을 찾음
    #span태그 이면서 class속성이 'style-scope ytd-thumbnail-overlay-time-status-renderer'인 객체 조회
    video_time = video.find('span', {"class": 'style-scope ytd-thumbnail-overlay-time-status-renderer'})
    #비디오 타임의 공백을 제거 video_time.text.strip()
    #하고 video_time_list에 추가 video_time_list.append(video_time.text.strip())
    video_time_list.append(video_time.text.strip())

print(video_time_list)
print(len(video_time_list))

['10:29', '14:56', '19:36', '11:34', '12:16', '7:30', '10:12', '12:05', '8:57', '13:46', '18:28', '14:21', '5:41', '12:22', '6:06', '11:59', '10:30', '6:10', '13:32', '6:52', '13:44', '5:33', '16:31', '11:09', '11:47', '14:15', '9:04', '3:26', '15:23', '21:34', '5:20', '7:39', '6:04', '6:32', '17:49', '10:03', '6:15', '7:57', '10:18', '2:22', '8:58', '10:07', '7:00', '6:01', '7:47', '4:14', '12:35', '7:02', '6:27', '8:19', '14:56', '5:58', '6:49', '7:01', '10:07', '5:29', '16:15', '8:48', '7:44', '2:49', '7:16', '4:57', '4:15', '4:52', '2:12', '6:44', '6:18', '7:18', '5:42', '16:19', '17:26', '15:19', '5:29', '5:15', '1:46', '18:57', '6:47', '5:21', '18:40', '5:44', '8:07', '2:16', '4:39', '1:31', '1:35', '1:17']
```

86

```
In [8]: video_time_seperate_list=[]
#비디오 타임 흐리나를 time에 대입
for time in video_time_list:
    #time을 :를 기준으로 분리
    time_list=time.split(":")
    #길이가 3이면
    if len(time_list)<3:
        #분 초를 계산해서 video_time_seperate_list에 추가
        video_time_seperate_list.append(int(time_list[0])*60+int(time_list[1]))
    else:
        #시 분 초를 계산해서 video_time_seperate_list에 추가
        video_time_seperate_list.append(int(time_list[0])*3600+int(time_list[1])*60+int(time_list[2]))
```

```
In [28]: video_time_seperate_list
```

```
Out [28]: [629,
 896,
 1176,
 694,
 736,
 450,
 612,
 725,
 537,
 826,
 1108,
 861,
 341,
 742,
 366,
 659,
 630,
 370,
 812,
 412,
 824,
 333,
 991,
 669,
 707,
 855,
```

# Youtube 조회수 조회

```
</div>
▼<div id="metadata-line" class="style-scope
ytd-grid-video-renderer">
  ...
  <span class="style-scope ytd-grid-video-
  renderer"> == $0
    "조회수 44만회"
    ::after
  </span>
  <span class="style-scope ytd-grid-video-
  renderer">1주 전</span>
```

# 조회수 크롤링

```
In [29]: view_num_list=[]
import re
#조회수를 포함하는 문자열을 검색하기 위한 객체 view_num_regex 생성
view_num_regex = re.compile(r'조회수')
#all_videos에 저장된 video정보를 하나를 video에 대입하고 반복
for video in all_videos:
    #비디오에 포함된 조회수를 리턴
    #span 태그 이면서 class속성이 style-scope ytd-grid-video-renderer 인 객체 조회
    view_num = video.find('span', {"class": 'style-scope ytd-grid-video-renderer'})
    #view_num의 문자열(view_num.text)에 '조회수' 문자열이 포함되어 있으면 true
    #if view_num_regex.search(view_num.text):
    if view_num_regex.search(view_num.text):
        #view_num_list에 view_num.text(조회수 문자열)을 추가 :view_num_list.append(view_num.text)
        view_num_list.append(view_num.text)

view_num_list
```

```
Out [29]: ['조회수 71만회',
'조회수 33만회',
'조회수 126만회',
'조회수 12만회',
'조회수 122만회',
'조회수 26만회',
'조회수 15만회',
'조회수 59만회',
'조회수 150만회',
'조회수 132만회',
'조회수 14만회',
'조회수 49만회',
'조회수 58만회']
```

```
In [11]: #조회수 를 숫자로 변환해서 저장할 리스트
view_number_type_list=[]
#view_num_list에서 조회수 하나를 view에 대입
for view in view_num_list:
    # "조회수" 문자열 삭제
    view=view.replace("조회수 ","")
    #view의 마지막 2번째 문자열까지(숫자가 저장) view[:-2] 을 실수로 변환 float()
    #해서 num에 대입
    num=float(view[:-2])
    #조회수의 마지막 2번째 부터 마지막 문자열 까지 (만화, 천회) danwee에 대입
    danwee=view[-2:]
    #단위가 "만화"이면
    if danwee=="만화":
        #조회수 숫자에 10000 곱함
        view_number_type_list.append(int(num*10000))
    else:
        #아니면 조회수 숫자에 1000 곱함
        view_number_type_list.append(int(num*1000))
```

```
In [12]: view_number_type_list
```

```
Out[12]: [130000,
1150000,
130000,
410000,
440000,
530000,
```

# 데이터 프레임 생성

```
In [43]: dict_youtube={"title":title_list, "video_time":video_time_seperate_list, "view_num": view_number_type_list}
```

```
In [44]: import pandas as pd
youtube=pd.DataFrame(dict_youtube)
youtube.head()
```

Out [44] :

	title	video_time	view_num
0	돼지고기 사례 마트 다녀왔습니다!   백종원의 쿠킹로그 번외편	629	710000
1	'골목식당 때문에 매장 매출이 줄었어요!?'   백종원의 장사이야기	896	330000
2	참 쉬운 김밥만들기, 김밥 A~Z까지~!   백종원의 쿠킹로그	1176	1260000
3	내 지출을 머릿속으로 감안할 수 있는 식당이 잘돼요.   백종원의 장사이야기	694	120000
4	햄,치즈,달걀 환상조화 국민 간식 토스트, 신박하게 쉬운 방법~!   백종원의 쿠킹로그	736	1220000

# 텍스트 분석

## ■ 텍스트 분석

- ◆ 텍스트 분석, 텍스트 마이닝 (Text Mining)
- ◆ 자연어 처리(NLP, Natural Language Processing)에 기반하고 있음
- ◆ 텍스트 즉, 비정형 데이터로부터 정보를 추출해 내는 작업
- 텍스트 정규화 (Text Normalization)
  - ◆ 텍스트의 형태를 일관되게 변형하는 작업
  - ◆ 토큰화 (Tokenization)
    - 텍스트를 의미 단위(토큰)로 분학하는 작업
  - ◆ 어간 추출 (Stemming)
    - 형태가 변형된 단어로부터 어간을 분리하는 작업
- 형태소 분석 (POS-Tagging)
  - ◆ 토큰의 형태소를 파악하는 작업

## ■ 텍스트 분석의 종류

- ◆ 정보 추출 (Information Retrieval)
  - 문서내의 정형 데이터를 추출하는 작업
- ◆ 문서 분류 (Text Classification)
  - 문서들을 특정 분류 체계에 따라 분류하는 작업
- ◆ 감성 분석 (Sentiment Analysis)
  - 문서에 내포되어 있는 감정과 의견을 추출하는 작업

## 2. 토큰화

### ■ 토큰화

- ◆ 텍스트를 의미 단위로 분석하는 작업으로, 단위를 토큰(token)이라고 함
- Type과 Token
  - ◆ Type: 단어 요소, 어간
  - ◆ Token: 텍스트 내부의 원소

## 2. 토큰화

- ◆ 예: Text normalization is the process of transforming text into a single canonical form that it might not have had before ([https://en.wikipedia.org/wiki/Text\\_normalization](https://en.wikipedia.org/wiki/Text_normalization))
  - Token은 공백을 기준으로 단순 분활
  - 20 tokens
    - text, normalization, is, the, process, of, transforming, text, into, a, single, canonical, form, that, it, might, not, have, had, before
  - Type은 중복을 제외하고 (text) 어간이 같은 단어의 중복을 제외 (had)
  - 18 types
    - text, normalization, is, the, process, of, transforming, **text**, into, a, single, canonical, form, that, it, might, not, have, **had**, before

## 2. 토큰화

### ■ 토큰화 작업은 언어마다 방법이 다름

- ◆ 영어: 단순 공백 기준 분활
- ◆ 독일어: 복합명사에 대한 분리 방법이 요구됨
- ◆ 중국어: 공백이 없으며 여러 문자로 한 단어가 이루어짐, 평균 2.4개의 문자들로 한 개의 단어가 구성
- ◆ 한국어: 단순 공백 기준이 아닌 품사 기준으로 토큰화

- 예: 한국어를 토큰화하는 예시 문장입니다.
- 한국어, 를, 토큰화, 하다, 예시, 문장, 입, 니다

# 어간 추출

## ■ Lemmatization

- ◆ 문장 속에서 다양한 형태로 변형된 단어의 표제어(lemma)를 찾는 작업
- ◆ 표제어: 단어의 의미를 사전에서 찾을 때 사용하는 기본형
  - 예: stemming → stem
- ◆ 어간 추출 (stemming)과의 차이점
  - 어간 추출 작업은 단어의 단순한 어근을 추출하는 작업
  - Lemmatization은 문장 구조상 단어의 의미를 이해한 후 기본형 추출
  - Stemming 예: flies → fly
  - Lemmatization 예: flies가 "날다" (동사)인지 "파리" (명사)인지 파악

# 어간 추출

## ■ 어간 추출 (Stemming)

- ◆ 형태가 변형된 단어에 대해 접사 등을 제거하고 어간을 분리해 내는 작업
- ◆ 예시
  - fly, flying, flies → fly
  - stemming, stemmed, stemmer → stem

# 형태소 분석

## ■ 형태소 분석

- ◆ POS-Tagging (Part-Of-Speech Tagging)
  - ◆ 토큰의 품사 정보를 추출하는 작업
    - 예: Jane plays well with her friends  
Jane(NNP), play(VB), well(RB), her(PRP),  
friends(NNS)
- Open class vs. Closed class
  - ◆ POS-Tagging은 open class와 closed class로 구분
  - ◆ Closed class: 상대적으로 고정되어 있는 셋
    - 전치사: of, in, at, ...
    - 조동사: may, have, can, will, must, ...
    - 대명사: I, you, she, mine, their, ...
    - 주로 문장 내에서 문법적인 역할을 하는 단어

## ◆ Open class

- 영어의 open class: Noun, Verbs, Adjectives, Adverbs
- 명사: Sejong University, Samsung, cat, table, ...
- 동사: listen, see, enter, ...
- 형용사: good, better, worse, ...
- 부사: slowly, hardly, ...

## ■ 영어 형태소 분석기

- ◆ Stanford POS-Tagger, NLTK POS-Tagger

## ■ 한국어 형태소 분석기

- ◆ Hannanum, Kkma, Komoran, Twitter

## ■ 정보 추출 (Information Retrieval, IR)

- ◆ 구조화 되지 않은 문서 군집 내에서 필요한 정보를 포함하고 있는 문서를 찾아내는 작업
- ◆ 예: web search

## ■ 6개의 동화 데이터를 예시로 정보 추출 진행

- ◆ 주어진 6개의 동화 중 "Cricket", "Princess" 두 단어를 포함하면서 "Pinocchio"를 포함하고 있지 않은 희극을 찾는 작업

## ■ Bag of Words 모델

- ◆ 문서 내에 등장하는 단어들의 순서를 무시함
- ◆ 예:

John is quicker than Mary.  
Mary is quicker than John.

- 위의 두 문장은 같은 벡터를 가지고 있음
  - (John, Mary, is, quicker, than)
- 등장하는 단어의 순서는 고려하지 않음

# 정보 추출

## ■ Term Frequency (TF)

- ◆ 단어 빈도  $tf_{t,d}$ 는 특정 단어  $t$ 가 특정 문장  $d$  내에 등장하는 빈도
- ◆ 특정 단어가 10번 등장하는( $tf=10$ ) 문서가 1번 등장하는( $tf=1$ ) 문서보다 더 많이 타당하다고 볼 수 없음
- ◆ 문서의 관련도, 타당성은 term frequency와 비례하게 증가하지 않음
- ◆  $tf$  값이 클 수록 문서에 많이 등장하는 단어이지만, 정보가 부족한 단어일 가능성 높음  
예) good, increase, high
- ◆ 예의 단어들을 포함하는 문서들은 문서를 구분하는 척도로 사용하기에 적절하지 못함
- ◆ 드물게 등장하는 단어가 더 중요한 정보를 가지고 있음  
예) back-propagation
- ◆ back-propagation과 같이 드문 단어들에 대해 가중치를 부여할 필요 있음

```
word_list=[  
    "남자 남자 남자 밥을 먹는다",  
    "남자 남자 걷는다",  
    "남자 남자 남자",  
    "남자 남자 남자 남자",  
    "남자 남자 남자 여자 여자",  
    "남자 남자 남자 밥 밥 먹는다"  
]
```

tf	남자	여자	밥
[ 3, 0, 1 ]			
[ 2, 0, 0 ]			
[ 3, 0, 0 ]			
[ 4, 0, 0 ]			
[ 3, 2, 0 ]			
[ 3, 0, 2 ]			

- Inverse Document Frequency (IDF)
  - ◆  $df_t$ 는 document frequency로 단어  $t$ 가 등장한 문장의 수
  - ◆  $df_t$ 는 단어  $t$ 의 중요도의 역수와 같음
  - ◆ 단어  $t$ 의 idf 수치  $idf_t$ 는  $t$ 의 단어의 중요도를 나타내는 척도
    - $N$ : 문서 군집에 속하는 전체 문자의 개수 개수 (6)
  - ◆ 예 (첫번째 문장에서 남자의 IDF)

$$n = 6$$

$$idf(t) = \log \frac{1+n}{1+df(t)} + 1,$$

## ■ TF-IDF

- ◆ TF-IDF (Term Frequency – Inverse Document Frequency)는 TF와 IDF의 곱
- ◆ 문서군 내에서 특정 단어의 중요도를 수치화한 값
- ◆ tf-idf 값은 특정 문서 내 특정 단어의 빈도(tf)가 높아질 수록 커짐
- ◆ tf-idf 값은 문서군 내에서 단어가 희박하게 등장할 수록(idf) 커짐

남자 tf-idf<sub>term1</sub> = tf × idf = 3 × 1 = 3

밥 tf-idf<sub>term3</sub> = 1 × log(7/3) + 1 ≈ 1.8473

$$v_{norm} = \frac{v}{\|v\|_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \dots + v_n^2}}$$

$$\frac{[3, 0, 1.8473]}{\sqrt{(3^2 + 0^2 + 1.8473^2)}} = [0.8515, 0, 0.5243]:$$

# 정보 추출

- ◆ 문서 내 단어의 등장 유무를 나타내는 행렬
  - 문서 내 단어의 등장 빈도를 나타내는 행렬
  - 단어의 tf-idf 값으로 이루어진 행렬

	남자	밥	여자
0	0.851513	0.524333	0.000000
1	1.000000	0.000000	0.000000
2	1.000000	0.000000	0.000000
3	1.000000	0.000000	0.000000
4	0.554229	0.000000	0.832364
5	0.630357	0.776305	0.000000

- ◆ 각 문서를 tf-idf 값으로 이루어진 벡터로 표현 할 수 있음
- ◆ 문서는 벡터 공간에서 점 또는 벡터로 표현됨

## ■ 데이터

- ◆ 유튜브 제목 데이터를 이용하여 조회수를 예측

- `pip install konlpy`  한글 형태소/단어 분리를 위한 라이브러리 설치 실패시 `visualcppbuildtools_full.exe` 설치
- `pip install Twitter`  형태소/단어 분리에 도움을 주는 라이브러리

# 품사 추출

```
In [16]: from konlpy.tag import Twitter
# 문장을 단어 단위로 나누고 단어, 품사를 리턴하는 객체
twitter = Twitter()
```

```
In [52]: tagged = twitter.pos(youtube['title'][0])
tagged
```

```
Out [52]: [(['돼지고기', 'Noun'),
           ('사리', 'Verb'),
           ('마트', 'Noun'),
           ('다녀왔습니다', 'Verb'),
           ('!', 'Punctuation'),
           ('!', 'KoreanParticle'),
           ('백종원', 'Noun'),
           ('의', 'Josa'),
           ('쿠킹', 'Noun'),
           ('로그', 'Noun'),
           ('번외편', 'Noun')]
```

```
In [49]: tagged = twitter.pos(youtube['title'][0], stem=True)
tagged
```

```
Out [49]: [(['돼지고기', 'Noun'),
           ('사르다', 'Verb'),
           ('마트', 'Noun'),
           ('다녀오다', 'Verb'),
           ('!', 'Punctuation'),
           ('!', 'KoreanParticle'),
           ('백종원', 'Noun'),
           ('의', 'Josa'),
           ('쿠킹', 'Noun'),
           ('로그', 'Noun'),
           ('번외편', 'Noun')]
```

# 명사 추출

```
In [50]: for i in range (0, len(tagged)):
    if (tagged[i][1]=='Noun') :
        print(tagged[i])
```

```
('돼지고기', 'Noun')
('마트', 'Noun')
('백종원', 'Noun')
('쿠킹', 'Noun')
('로그', 'Noun')
('번외편', 'Noun')
```

```
In [53]: for i in range (0, len(tagged)):
    if (tagged[i][1]=='Noun') :
        print(tagged[i][0])
```

```
돼지고기
마트
백종원
쿠킹
로그
번외편
```

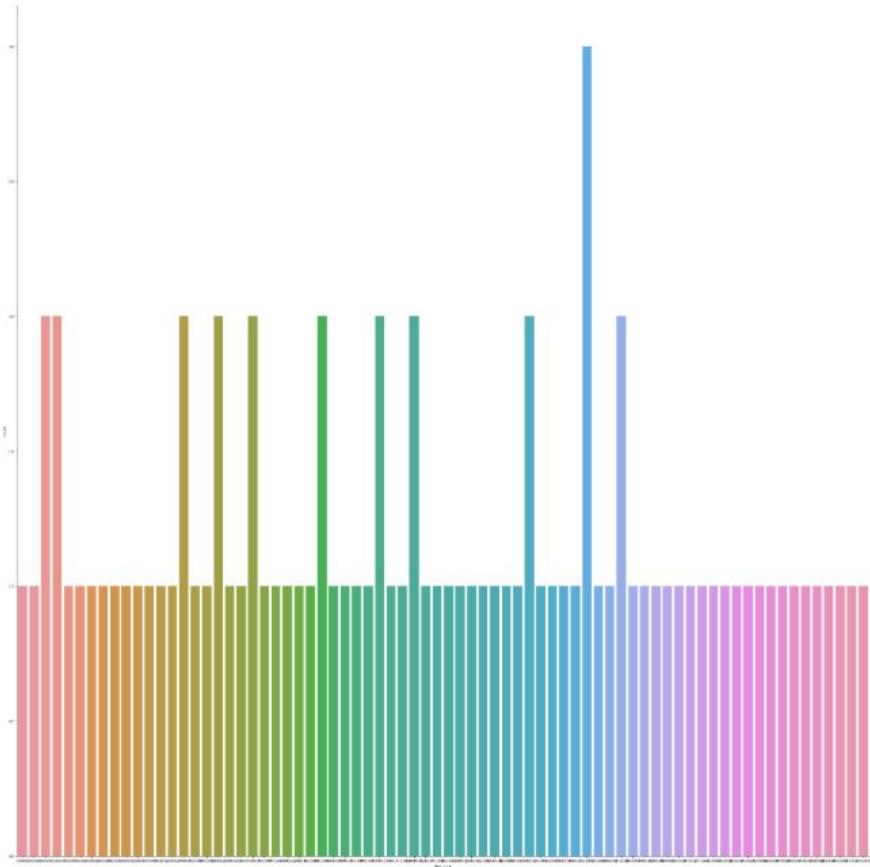
# 조회수 시각화

```
In [54]: import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [55]: from matplotlib import font_manager, rc
font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

```
In [56]: g = sns.factorplot('view_num', data=youtube, kind='count', size=30)
g.set_xlabels()
```

# 조회수 시각화



# 조회수 시각화

```
In [57]: #조회수가 null인 줄 삭제
youtube = youtube.dropna(subset=['view_num'])
#view_num2컬럼을 생성하고 0으로 초기화
youtube['view_num2'] = 0
```

```
In [58]: youtube.head()
```

```
Out [58]:
```

		title	video_time	view_num	view_num2
0		돼지고기 사러 마트 다녀왔습니다!   백종원의 쿠킹로그 번외편	629	720000	0
1		'골목식당 때문에 매장 매출이 줄었어요!?'   백종원의 장사이야기	896	330000	0
2		참 쉬운 김밥만들기, 김밥 A~Z까지~!   백종원의 쿠킹로그	1176	1290000	0
3		내 지출을 머릿속으로 감안할 수 있는 식당이 잘돼요.   백종원의 장사이야기	694	120000	0
4		햄, 치즈, 달걀 활용조화 국민 간식 토스트, 신박하게 쉬운 방법~!   백종원의 쿠킹로그	736	1230000	0

# 조회수 시각화

```
In [61]: #youtube의 줄의 수만큼 반복
for index in range(len(youtube)) :
    #youtube "view_num" 컬럼의 index번째 데이터를 view_num0에 대입
    view_num=youtube["view_num"][index]
    #view_num이 50만 미만이면
    if(view_num< 500000 ) :
        #view_num2컬럼에 0 대입
        youtube['view_num2'][index] = 0
    #view_num이 100만 미만이면
    elif (view_num< 1000000) :
        #view_num2컬럼에 1 대입
        youtube['view_num2'][index] = 1
    #view_num이 100만 이상이면
    else :
        #view_num2컬럼에 2 대입
        youtube['view_num2'][index] = 2

youtube.head()
```

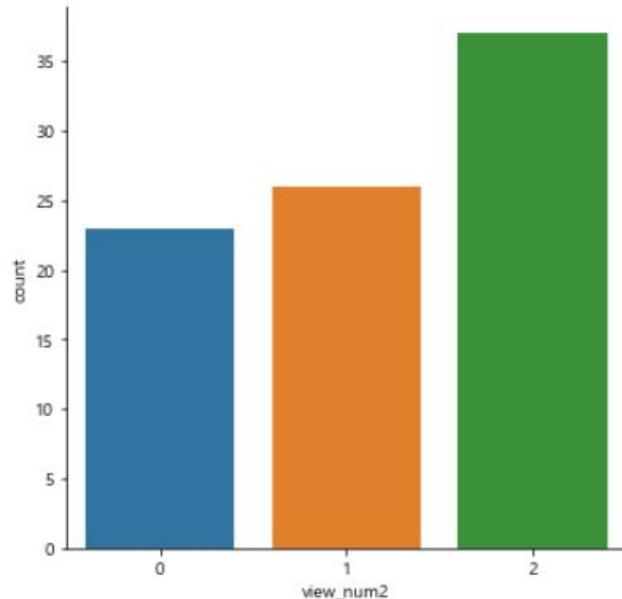
Out [61]:

		title	video_time	view_num	view_num2
0	돼지고기 사려 마트 다녀왔습니다!   백종원의 쿠킹로그 번외편		629	720000	1
1	'골목식당 때문에 매장 매출이 줄었어요!?'   백종원의 장사이야기		896	330000	0
2	참 쉬운 김밥만들기, 김밥 A~Z까지~!   백종원의 쿠킹로그		1176	1290000	2
3	내 지출을 머릿속으로 감안할 수 있는 식당이 잘돼요.   백종원의 장사이야기		694	120000	0
4	햄, 치즈, 달걀 환상조화 국민 간식 토스트, 신박하게 쉬운 방법~!   백종원의 쿠킹로그		736	1230000	2

# 조회수 시각화

```
In [62]: g = sns.factorplot('view_num2', data=youtube, kind='count', size=5)  
g.set_xlabels()
```

```
Out [62]: <seaborn.axisgrid.FacetGrid at 0x18385189cc0>
```



# 학습데이터와 테스트 데이터 분리

```
In [74]: import numpy as np
from sklearn.model_selection import train_test_split

In [75]: X_train, X_test, y_train, y_test = train_test_split(youtube["title"], youtube["view_num2"])

In [76]: X_train
Out[76]: 68  [백종원의 잘사이야기] 32회 두 번째 '볶음밥 없어요?' 물는 건 단골이 별로 없...
35  순한맛 라면의 반전 매력!! 햄, 소시지, 달걀프라이와의 환상조화 순정이라면! | ...
76  초간단 김치찌개 | 백종원의 백종원 레시피
84  [안녕하세요 백종원입니다-2] 요리에 자신감을 드리고 싶었습니다
74  [감사영상] 백만 명이 되었습니다!!

32  '외식업이 좋아서 뛰어든 사람들이 상처를 많이 입어요..' 백종원의 잘사이야기 34...
57  강식당2 화제의 메뉴! 김치밥이 피오씁니다 | 백종원의 백종원 레시피
20  초:초:초간단 만능양념장으로 만드는 춘천식 닭갈비! 1:1:1:1:1:1 | 백종원...
70  양파 농가를 응원합니다! 만능양파볶음 대작전 1편: 양파 손질과 보관법 | 백종원의...
29  달걀 허라이 만큼 정~밀 쉬운 '전' 보여드릴게요. 육전/동태전/호박전/배추전/새우...
Name: title, Length: 64, dtype: object

In [77]: y_train
Out[77]: 68  2
35  2
76  2
84  1
74  2
...
32  0
57  2
20  1
70  2
29  2
Name: view_num2, Length: 64, dtype: int64
```

# 명사와 형용사를 리턴하는 함수 선언

```
In [70]: #명사와 형용사를 리턴하는 함수
def getNounAndAdjective(text):
    stems = []
    tagged = twitter.pos(text, stem=True)
    for i in range (0, len(tagged)):
        if (tagged[i][1]=='Noun' or tagged[i][1]=='Adjective') :
            stems.append(tagged[i][0])
    return stems
```

```
In [78]: getNounAndAdjective("아버지가 방에 들어 가신다 그래서 슬프다")
```

```
Out [78]: ['아버지', '방', '슬프다']
```

# Tfidf계산

```
In [71]: from sklearn.feature_extraction.text import TfidfVectorizer
```

#문자열의 tfidf를 계산하는 객체

```
vectorizer = TfidfVectorizer(min_df=2, tokenizer=getNounAndAdject)
```

```
In [72]: #학습데이터의 tfidf를 계산
```

```
X_train_vector = vectorizer.fit_transform(X_train)
```

#테스트 데이터의 tfidf를 계산

```
X_test_vector = vectorizer.transform(X_test)
```

```
In [73]: #X_train_vector를 데이터 프레임으로 변환
```

```
df_tfidf1 = pd.DataFrame(X_train_vector.A, columns=vectorizer.get_feature_names())
```

```
df_tfidf1
```

Out [73]:

	가족	간단	강	거	건	고등어	국수	김치	냉	네	...	터키	특집	편	프랜차이즈	한국	한식	햄	화제	활용	회복하다
0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
1	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
2	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
3	0.317248	0.0	0.0	0.0	0.317248	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
4	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.351713	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.278596	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
60	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.745267	0.0	0.0	0.0	0.0	0.000000	0.0
61	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
62	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
63	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.283412	0.0

64 rows × 83 columns

# Tfidf계산

```
In [38]: df_tfidf2 = pd.DataFrame(X_test_vector, columns=vectorizer.get_feature_names())
df_tfidf2
```

Out [38]:

	가게	가족	간단	강	거	건	고등어	공개	국수	김치	...	케밥	쿠킹	터키	토스트	특급	특집	편	햄	화제	활용
0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
1	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
2	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
3	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.432459	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
4	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.392110	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
5	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.662900	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
6	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
7	0.0	0.0	0.000000	0.0	0.486683	0.0	0.0	0.0	0.0	0.0	...	0.0	0.251493	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
8	0.0	0.0	0.583048	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
9	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
10	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
11	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
12	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
13	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
14	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.258247	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
15	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
16	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.373439	0.0	0.0	0.0	0.0	0.0	0.826224	0.0	0.0
17	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
18	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
19	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
20	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
21	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0

22 rows x 85 columns

# Decision Tree 개념

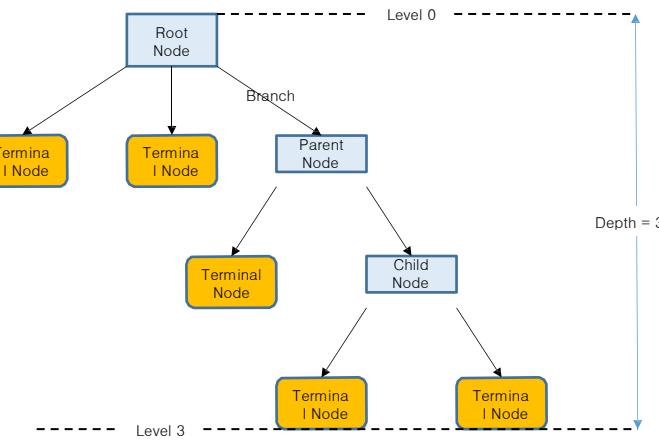
## Decision Tree란?

- 의사결정 트리 또는 의사결정 나무(Decision Tree)는 기계학습(Machine Learning)에서 지도학습(Supervised Learning)의 알고리즘으로 분류(Classification) 또는 회귀(Regression)분석 목적으로 사용
- 의사결정 규칙(Decision Rule)을 나무구조 표현을 통해 분류와 예측을 수행하는 분석 방법
- 분류 또는 예측 과정이 나무구조로 표현되어 비교적 쉽게 이해
- 목표변수 유형에 따른 의사결정 트리
  - 범주형 목표변수 : 분류 트리(Classification Tree)
    - 목표변수가 이산형인 경우, 각각의 범주에 속하는 빈도에 기초해 분리 발생 → 분류 트리 구성
  - 연속형 목표변수 : 회귀 트리(Regression Tree)
    - 목표변수가 연속형인 경우, 평균과 표준편차에 기초해 분리 발생 → 회귀 트리 구성

# Decision Tree 개념

## Decision Tree 구성요소

- 뿌리 마디(Root Node)
  - 트리 구조가 시작되는 마디, 전체 자료로 구성
- 부모 마디(Parent Node)
  - 자식 마디의 상위 마디
- 자식 마디(Child Node)
  - 하나의 마디로부터 분리되어 나간 2개 이상의 마디들을 의미
- 끝 마디(Terminal Node) 또는 잎(Leaf Node)
  - 트리 줄기의 끝에 위치하고 있고 자식 마디가 없는 마디
- 가지(Branch)
  - 뿌리 마디로부터 끝 마디까지 연결된 마디들
- 깊이(Depth)
  - 뿌리 마디로부터 끝 마디를 이루는 총의 수



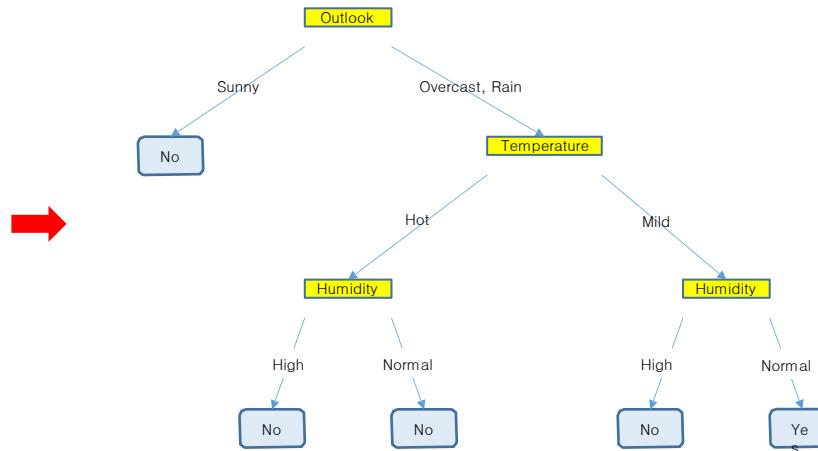
# Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정

Training Data

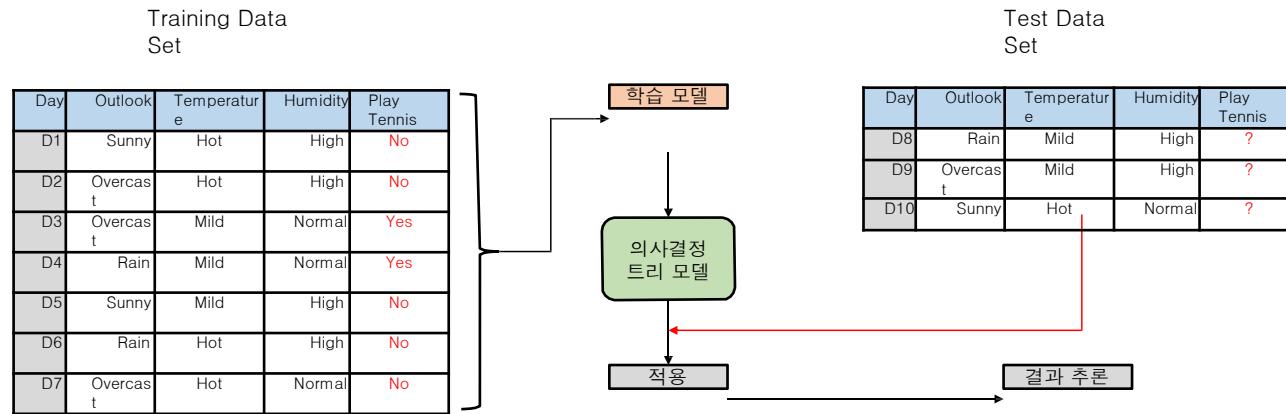
Day	Outlook	Temperature	Humidity	Play Tennis
D1	Sunny	Hot	High	No
D2	Overcast	Hot	High	No
D3	Overcast	Mild	Normal	Yes
D4	Rain	Mild	Normal	Yes
D5	Sunny	Mild	High	No
D6	Rain	Hot	High	No
D7	Overcast	Hot	Normal	No

Model : Decision Tree



# Decision Tree 개념

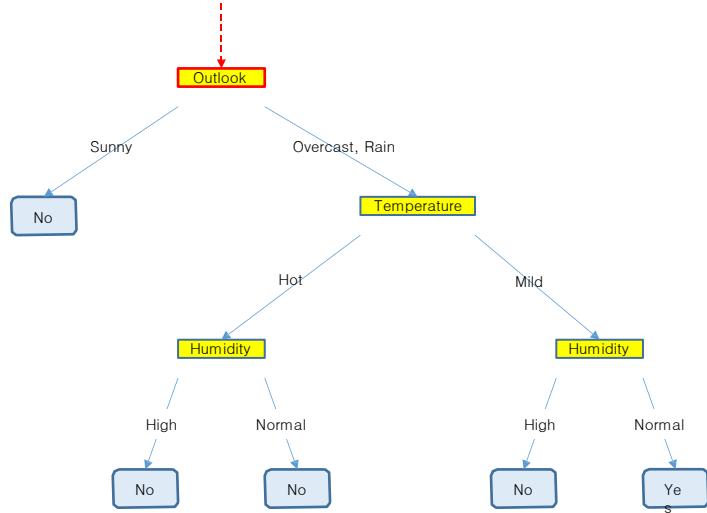
## 예제를 이용한 Decision Tree 동작 과정



# Decision Tree 개념

## 예제를 이용한 Decision Tree 동작 과정

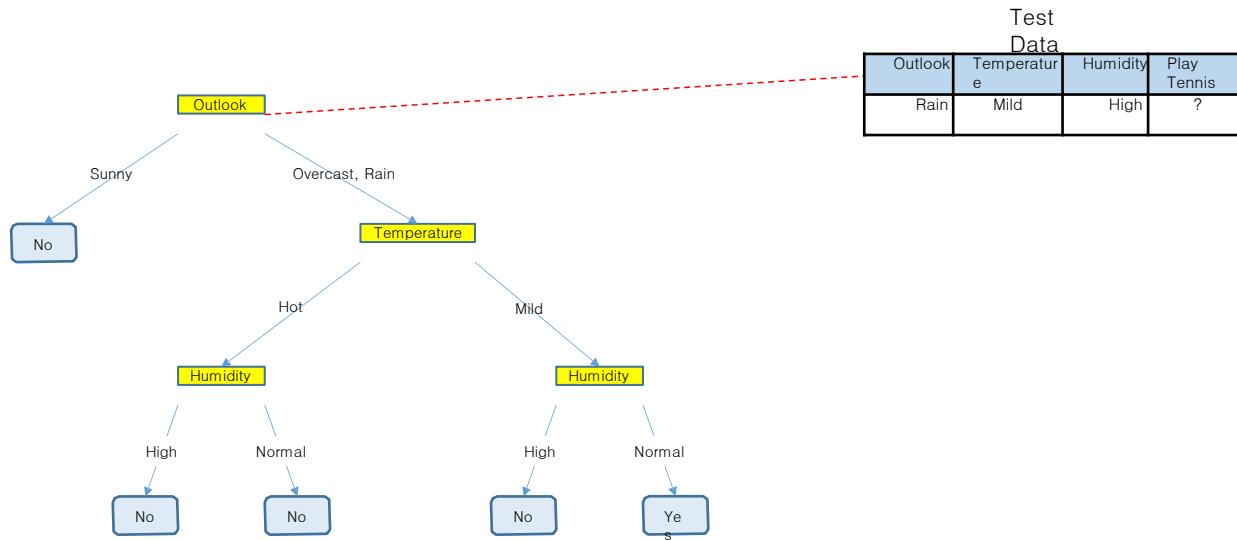
트리 구조의 뿌리 노드로 부터 시작



Test Data			
Outlook	Temperature	Humidity	Play Tennis
Rain	Mild	High	?

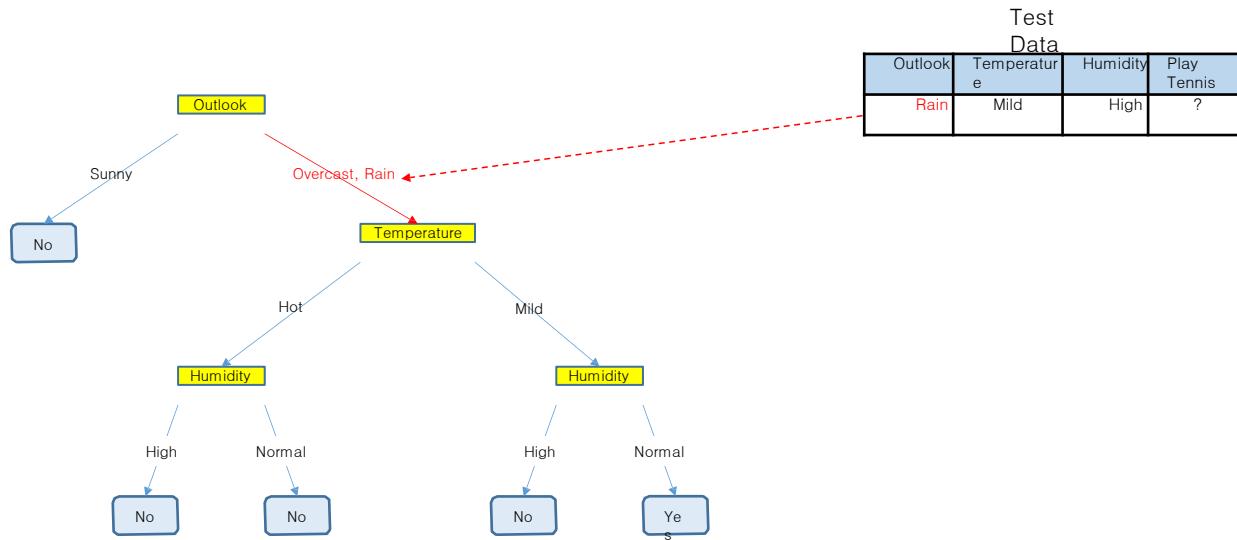
# Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



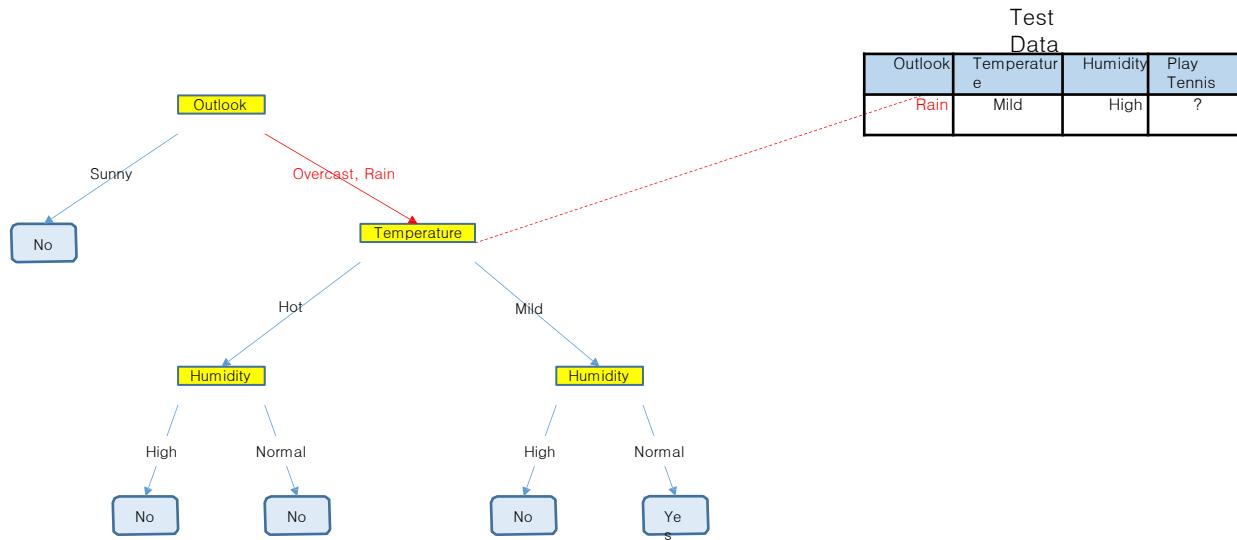
# Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



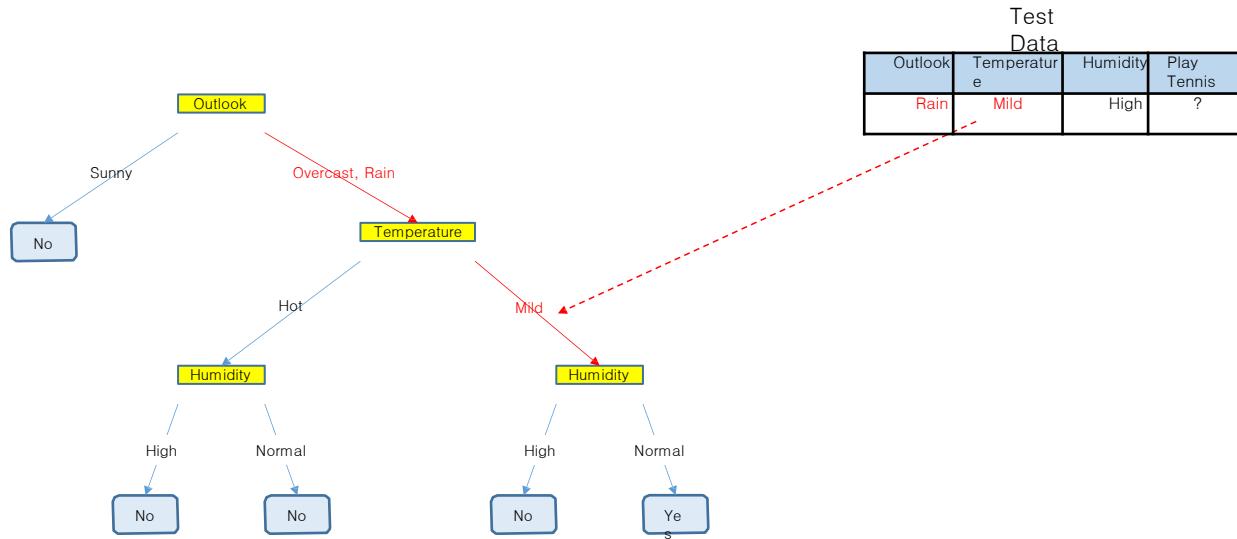
# Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



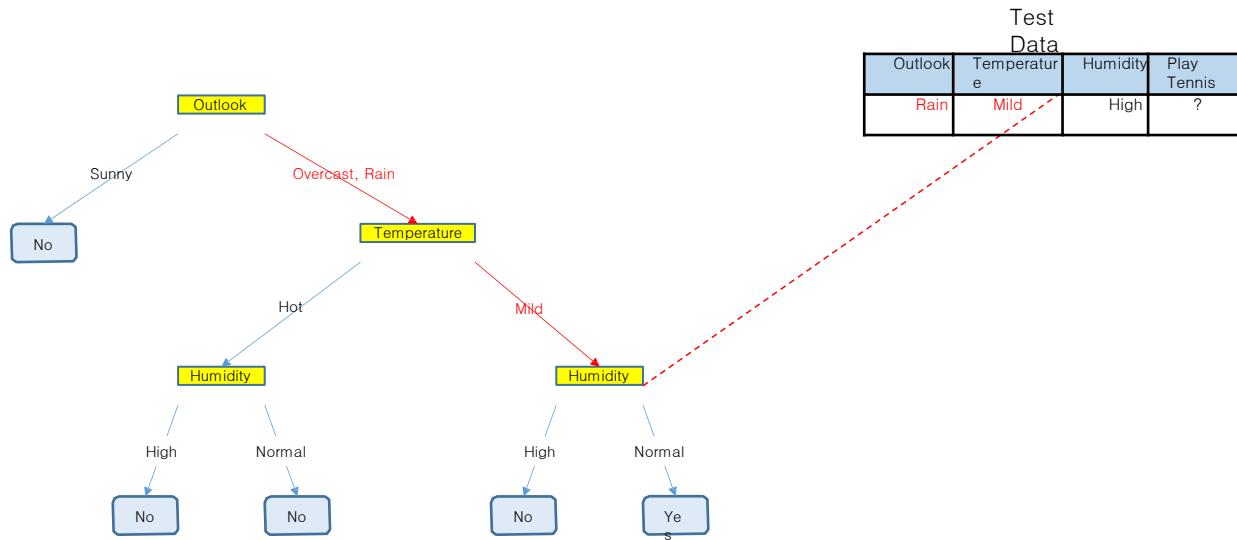
# Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



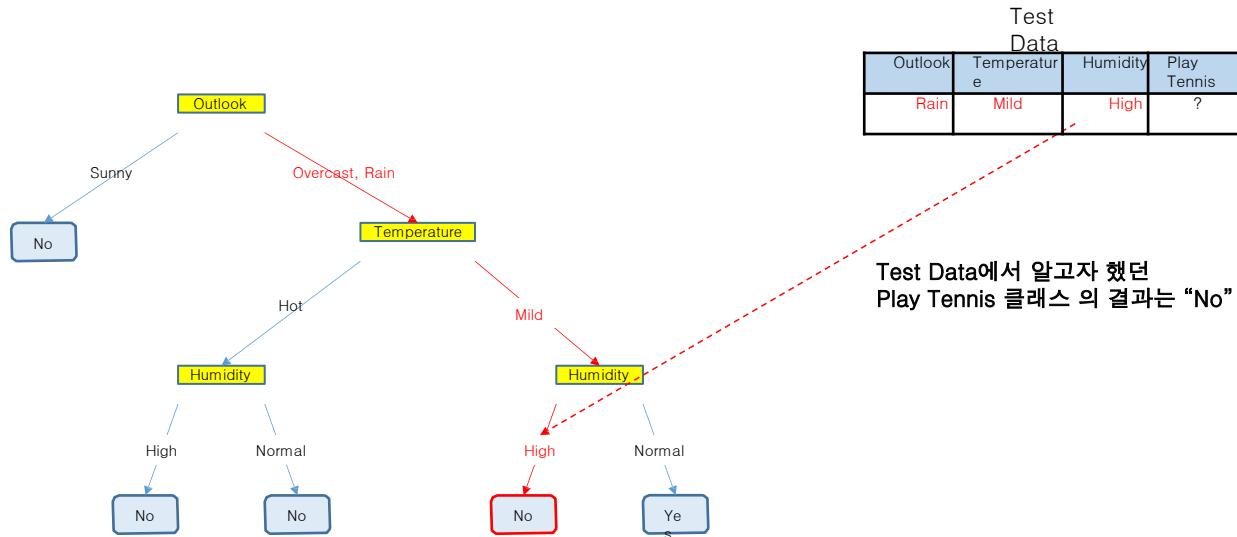
# Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



# Decision Tree 개념

## 예제를 이용한 Decision Tree 동작 과정



# Decision Tree 개념

## Decision Tree 분리기준(Split Criterion)

- 부모 마디(노드)로부터 자식 마디(노드)들이 형성될 때, 생성된 자식 노드에 속하는 자료의 순수도(Purity)가 가장 크게 증가하도록 트리를 형성하며 진행
- 입력 변수를 이용해 목표 변수의 분포를 얼마나 잘 구별하는 정도를 파악해 자식 마디가 형성되는데, 목표 변수의 구별 정도를 불순도(Impurity, 다양한 범주들의 개체들이 포함되어 있는 정도)에 의해 측정

# Decision Tree 개념

## Decision Tree 분리기준(Split Criterion)

- 지니 지수(Gini Index)
  - 데이터 집합의 불순도를 측정
  - 지니 지수는 0~1 사이의 값을 가지며, 어떤 데이터 집합에 속한 개체(레코드)들이 같은 범주(클래스)로 구성되어 있으면 지니 지수는 최솟값이 0을 갖고 해당 데이터 집합은 순수하다고 볼 수 있음
  - 즉, 지니 지수가 작을수록 잘 분류된 것으로 볼 수 있음

# 의사 결정트리를 이용한 유튜브 조회수 예측

```
In [39]: # 의사결정 트리 알고리즘 서브 패키지를 불러들임
from sklearn.tree import DecisionTreeClassifier

dt_clf = DecisionTreeClassifier()

In [40]: dt_clf = dt_clf.fit(X_train_vector, y_train)

In [41]: dt_prediction = dt_clf.predict(X_test_vector)

In [42]: dt_prediction

Out[42]: array([1, 2, 0, 2, 2, 2, 0, 2, 2, 0, 1, 0, 0, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2])

In [43]: y_test|
```

Out[43]: [1, 2, 0, 1, 0, 1, 0, 2, 2, 0, 1, 0, 0, 2, 1, 2, 2, 2, 0, 2, 1, 2]

# 모델의 성능 평가

```
In [44]: # 모델의 성능을 평가하기 위한 패키지 불러들임
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, dt_prediction)
```

```
Out [44]: array([[4, 1, 2],
                 [0, 2, 4],
                 [0, 1, 8]], dtype=int64)
```

```
In [45]: from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, dt_prediction)
print("정확도:", accuracy)
```

정확도: 0.6363636363636364

# 의사 결정트리 시각화

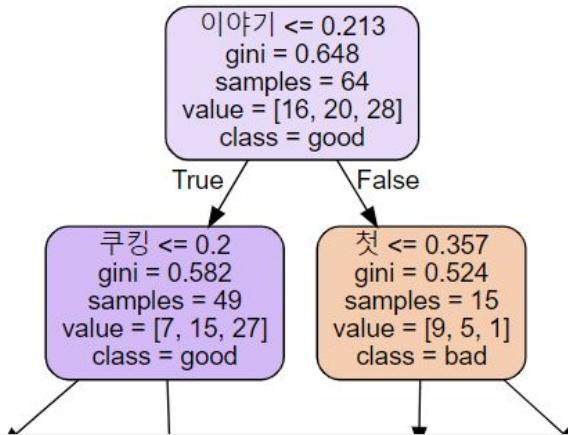
```
In [46]: from sklearn.tree import export_graphviz

export_graphviz(
    dt_clf,
    out_file="youtube_tree.dot",
    feature_names=df_tfidf1.columns,
    class_names=["bad", "normal", "good"],
    rounded=True,
    filled=True
)
```

```
In [47]: import graphviz
with open("youtube_tree.dot", encoding="UTF-8") as f:
    dot_graph = f.read()

dot = graphviz.Source(dot_graph, encoding="UTF-8")

dot
```

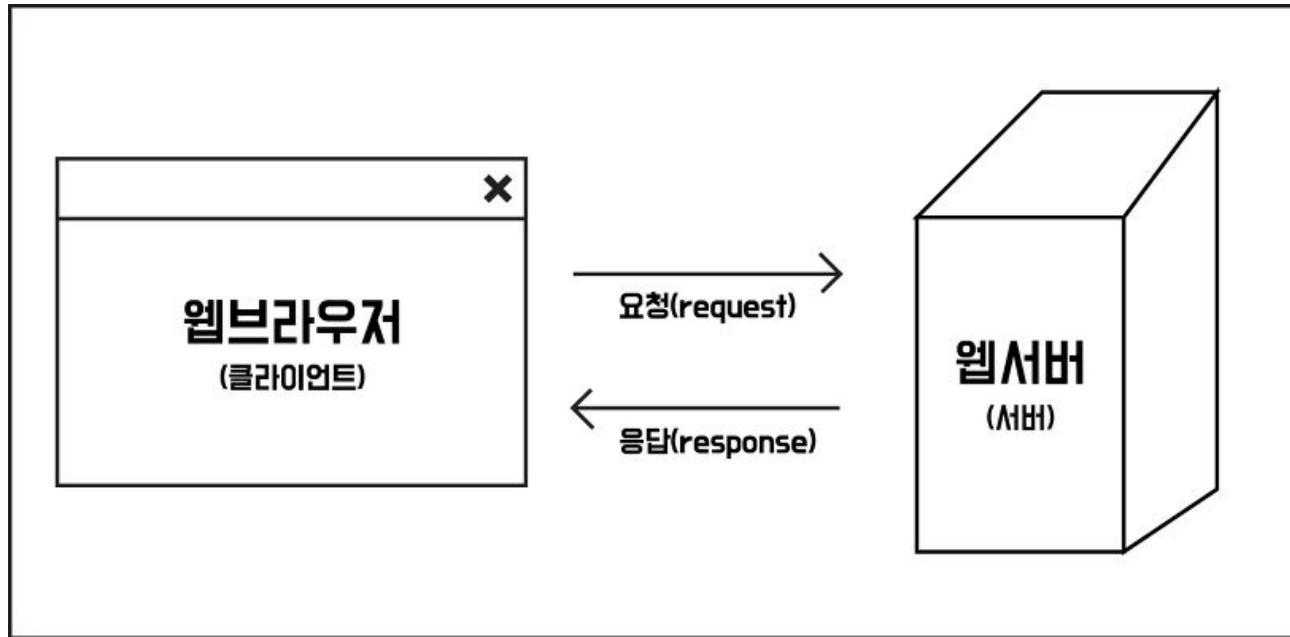




## 05. 네이버 Open API

## 네이버 오픈 API

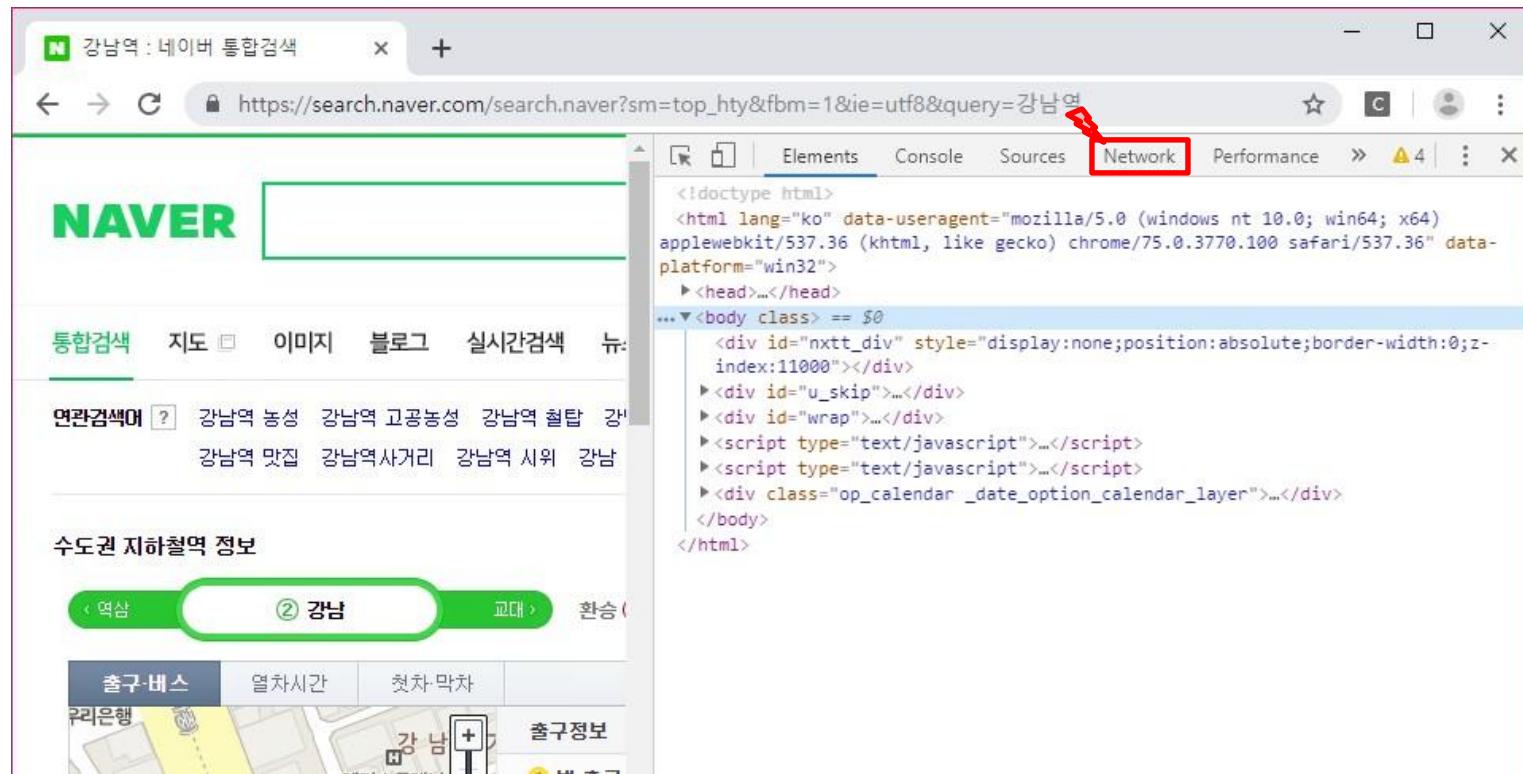
- 네이버 오픈 API를 사용하여 네이버 서버에 정보를 요청하고 응답 받으려고 함



# 네이버 오픈 API

## 일반적인 요청 내용 확인

- 크롬에서 네이버 접속  F12 Key  network 메뉴 선택  '강남역' 검색



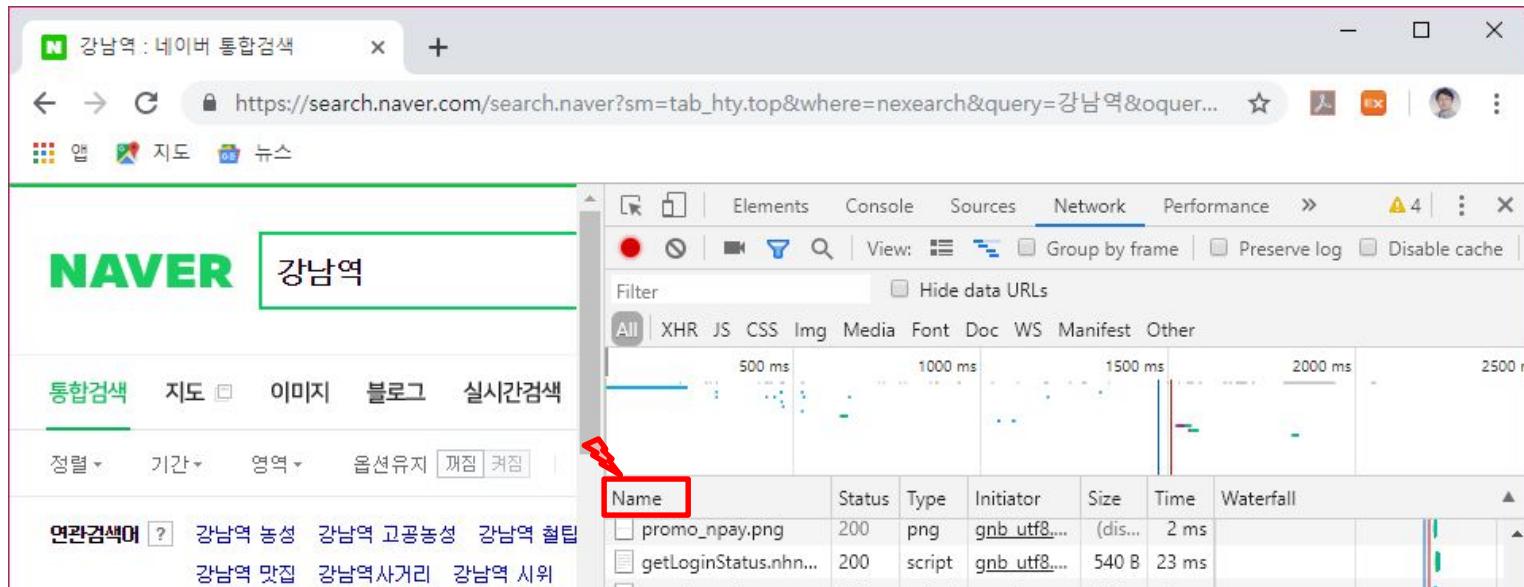
# 네이버 오픈 API

## 일반적인 요청 내용 확인

크롬에서 네이버 접속  $\square$  F12 Key  $\square$  network 메뉴 선택  $\square$  '강남역' 검색

- Name 클릭 후 아래 문장을 찾습니다.

- search.naver?sm=

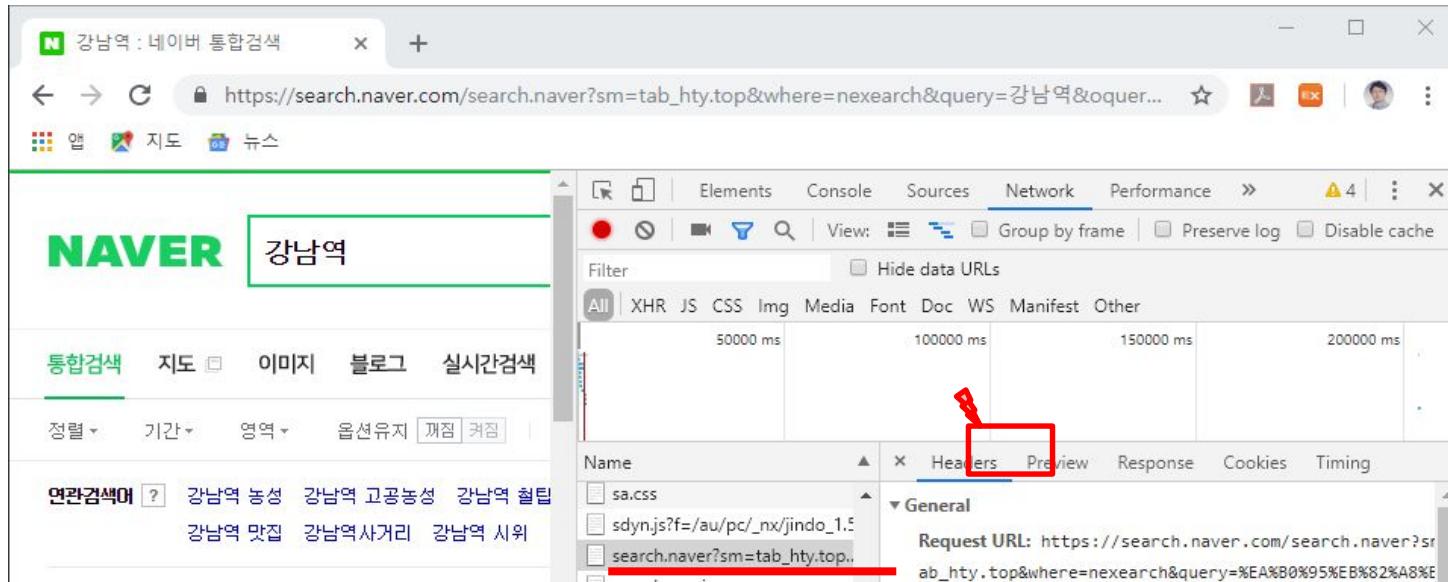


Name	Status	Type	Initiator	Size	Time	Waterfall
promo_npay.png	200	png	gnb_utf8...	(dis...	2 ms	
getLoginStatus.nhn...	200	script	gnb_utf8...	540 B	23 ms	

## 일반적인 요청 내용 확인

크롬에서 네이버 접속 □ F12 Key □ network 메뉴 선택 □ '강남역' 검색

- Name 클릭 후 아래 문장을 찾습니다.
  - search.naver?sm=
- Headers 클릭



# 네이버 오픈 API

## 일반적인 요청 내용 확인

- Request URL과 Request Header를 작성하여 네이버 서버에 전송하려고 함
- 네이버 API는 위의 과정

Request URL: https://search.naver.com/search.naver?sm=tab\_hty.top&query=%EA%B0%95%EB%82%A8%EC%97%AD&oquery=%EA%B0%95%EB%82%A8%EC%97%AD&tqi=fVqc1prvmZssPfNgFNssssssGN-487277

Request Method: GET

Status Code: 200

Remote Address: 210.89.172.84:443

Referrer Policy: unsafe-url

cache-control: no-cache, no-store, must-revalidate, max-age=0

content-encoding: gzip

content-type: text/html; charset=UTF-8

date: Sun, 14 Jul 2019 13:12:52 GMT

pragma: no-cache

server: npxg

set-cookie: page\_uid=UfVqSsprvTVssNC2vQsssssth0-060333; path=/; domain=.naver.com

set-cookie: \_naver\_usersession\_=E1r+HDXph0LtiX3HbixGnQ==; path=/; expires=Sun 14-Jul-19 13:17:52 GMT; domain=.naver.com

status: 200

vary: Accept-Encoding

x-frame-options: SAMEORIGIN

x-xss-protection: 1; report=/p/er/post/xss

## 네이버 오픈 API

### ■ 네이버 오픈 API 사용을 위하여

- 네이버 로그인 한 후 이 주소(<https://developers.naver.com/apps/#/register?defaultScope=search>)로 이동
- API 이용 신청

애플리케이션 등록 (API 이용신청)

애플리케이션의 기본 정보를 등록하면, 좌측 내 [애플리케이션](#) 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어집니다.



애플리케이션 이름

애플리케이션 이름

- 네이버 아이디로 로그인할 때 사용자에게 표시되는 이름이므로 가급적 10자 이내의 간결한 이름을 사용해주세요.
- 40자 이내의 영문, 한글, 숫자, 공백문자, "-", "\_" 만 입력 가능합니다.

선택하세요.

검색

비로그인 오픈 API  
서비스 환경

환경 추가

- HTTP Header에 애플리케이션 등록 시 발급받은 Client ID와 Client Secret 값을 같이 전송
  - API Client ID와 Client Secret은 코드에 첨부된 사용자의 ID와 Secret을 사용함
  - 예) Client-Id : ecMW562VrPlu\_LqFMsKs, Client-Secret : QtltGH23cF

# 네이버 오픈 API

## 네이버 오픈 API

### ■ 네이버 오픈 API 사용을 위하여

- API 이용 신청

#### 애플리케이션 등록 (API 이용 신청)

애플리케이션의 기본 정보를 등록하면, 좌측 [내 애플리케이션](#) 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어집니다.

애플리케이션 이름

edu

선택하세요.

검색

환경 추가

WEB 설정

웹 서비스 URL (최대 10개)

http://naver.com

등록하기

# 네이버 오픈 API

## 네이버 오픈 API

### 네이버 오픈 API 사용을 위하여

- Client ID와 Client Secret 기억할 것.



### 애플리케이션 정보



## ■ 네이버 오픈 API

### □ GET method

- HTTP 프로토콜에서 웹 서버가 요청과 응답 데이터를 전송할 때 사전에 약속된 동작 방식
- 데이터를 요청할 때 사용됨
- 쿼리 문자열 URL로 전송

🔒 [https://search.naver.com/search.naver?sm=tab\\_hty.top&where=nexearch&query=강남역&oquery=강남역&tqi=Ua5wUspVuEVssvmWyzZsssssj8-398855](https://search.naver.com/search.naver?sm=tab_hty.top&where=nexearch&query=강남역&oquery=강남역&tqi=Ua5wUspVuEVssvmWyzZsssssj8-398855)

- 네이버에서 강남역을 검색
  - naver.com/ 이후의 부분을 파이썬을 통해 작성하여 보내야함

## ■ 네이버 오픈 API

### □ 요청 URL 정의

- 참고 <https://developers.naver.com/docs/search/blog/>
- 요청 URL 예)

**https://openapi.naver.com/v1/search/blog?query=강남역  
\$display=20&start=50&sort=sim**

### □ 요청 변수

요청 변수	타입	필수 여부	기본값	설명
query	string	Y	-	검색을 원하는 문자열로서 UTF-8로 인코딩한다.
display	integer	N	10(기본값), 100(최대)	검색 결과 출력 건수 지정
start	integer	N	1(기본값), 1000(최대)	검색 시작 위치로 최대 1000까지 가능
sort	string	N	sim(기본값), date	정렬 옵션: sim (유사도순), date (날짜 순)

## ■ 네이버 오픈 API

### □ 소스코드(1/2)

```
import requests

#API 인증 정보
client_id =
"ecMW562VrPlu_LqFMsK"
client_secret = "QtltGH23c"
url1 = "https://openapi.naver.com/v1/search/blog?query=강남역"
url2 = "https://openapi.naver.com/v1/search/blog?query=강남역$display=20&start=50&sort=sim"
url3 = "https://openapi.naver.com/v1/search/blog?query=강남역$display=20&start=50&sort=date"

#header에 추가될 내용
headres = {'X-Naver-Client-Id':client_id, 'X-Naver-Client-Secret':client_secret}
```

## ■ 네이버 오픈 API – 블로그 검색

### □ 요청 URL 예

- 블로그에서 강남역을 검색한 결과(10개의 결과가 첫번째 결과부터 출력)
  - 검색결과 출력 개수는 10개를 첫번째 결과부터 출력하는게 기본값으로 설정되어 있음
- <https://openapi.naver.com/v1/search/blog?query=강남역>
- 블로그에서 강남역을 검색한 결과 20개를 50번째 결과부터 출력(유사도순)
- <https://openapi.naver.com/v1/search/blog?query=강남역&display=20&start=50&sort=sim>
- 블로그에서 강남역을 검색한 결과 20개를 50번째 결과부터 출력(날짜순)
- <https://openapi.naver.com/v1/search/blog?query=강남역&display=20&start=50&sort=date>

## ■ 네이버 오픈 API – 블로그 검색

### □ 소스코드(2/2)

```
import requests
#API 인증 정보
client_id = "ecMW562VrPlu_LqFMsK"
client_secret = "QtltGH23c"
url1 = "https://openapi.naver.com/v1/search/blog?query=강남역"
url2 = "https://openapi.naver.com/v1/search/blog?query=강남역&display=20&start=50&sort=sim"
url3 = "https://openapi.naver.com/v1/search/blog?query=강남역&display=20&start=50&sort=date"
#header에 추가될 내용
headers = {'X-Naver-Client-Id':client_id, 'X-Naver-Client-Secret':client_secret}

#requests.get(url, headers=headers)
# url 주소에 headers를 포함한 GET 요청을 보냄
result = requests.get(url1, headers=headers)
print(result.json())

#print("url2 결과")
#result=requests.get(url2, headers=headres)
#print(result.json())
#print("url3 결과")
#result=requests.get(url3, headers=headres)
#print(result.json())
```

## ■ 네이버 오픈 API – 블로그 검색

### □ 소스코드

```
In [1]: import requests
# API 인증 정보
client_id = 'cA1bX85p4JJ6yu1FQpV3'
client_secret = 'V8kLK0vC0L'

In [3]: url1 = "https://openapi.naver.com/v1/search/blog?query=강남역"
[REDACTED]

In [9]: #header에 추가할 내용
headers = {'X-Naver-Client-Id':client_id, 'X-Naver-Client-Secret':client_secret}
result = requests.get(url1, headers=headers)
result.json()
```

# 네이버 오픈 API

## 네이버 오픈 API – 블로그 검색

### 실행결과 json 포맷의 결과가 출력됨

```
Out[13]: {'lastBuildDate': 'Sun, 14 Jul 2019 22:43:54 +0900',
 'total': 1985824,
 'start': 1,
 'display': 10,
 'items': [{"title": '<b>강남역</b> 맛집 특별한 치즈 폭포',
 'link': 'https://blog.naver.com/miminimkkk?Redirect=Log&logNo=2215790846',
 'description': '<b>강남역</b> 맛집 특별한 치즈 폭포 안녕하세요 미미찡입니다. 최근에 퇴근 후에 <b>강남역</b>으로 가서 퍼포먼스도 구경하고 고퀄리티를 했기....',
 'bloggername': '♥ 미미찡의 글 까는 놀이터 ♥',
 'loggerlink': 'https://blog.naver.com/miminimkkk',
 'postdate': '2019.07.06'},
 {"title": '<b>강남역</b> 맛집 :: 미친 갓성비 스테이터',
 'link': 'https://blog.naver.com/dbswj16955?Redirect=Log&logNo=2215845007',
 'description': '요즘 강남에서 이 가격대에 맨찮은 스테이크를 썰 수 있다면 <b>강남역</b> 맛집이고 인기가 많은 곳인만큼 종종 웨이팅이 있을 수 있다고...',
 'bloggername': '꽃보다 청춘, 젤라홀릭',
 'loggerlink': 'https://blog.naver.com/dbswj16955',
 'postdate': '2019.07.13'},
 {"title": '<b>강남역</b> 맛집 :: 잊지못해 다시옴!',
 'link': 'https://blog.naver.com/llzzin11?Redirect=Log&logNo=221567844263',
 'description': '있어요 강남에 볼일 보러 갔다가 밥을 먹고 가기로 했는데요 어디있나 검색했는데 이탈리안 레스토랑 BOBIRED가 딱 보이더라고요 예전에 흥대',
 'bloggername': '미미찡의 글 까는 놀이터...',
 'loggerlink': 'blog.naver.com/mimin...',
 'postdate': '2019.07.06'},
 {"title": '강남역 맛집 : 마초쉐프 터프한 매력!',
 'link': 'https://blog.naver.com/dodol...?Redirect=Log&logNo=221567844263',
 'description': '마초쉐프 강남역점 주소 서울 강남구 강남대로98길 16 6.7층 (지번 : 서울 강남구 역삼동 817-35번지 파빌리온빌딩 6, 7층) 전화번호 02-566-8886 영업시간 오전 11시 30분 ~ 오후...',
 'bloggername': '도도리의 다이어리',
 'loggerlink': 'blog.naver.com/dodol...',
 'postdate': '2019.04.26'}]}
```

블로그 1-10 / 1,985,809건



강남역 맛집 :: 미친 갓성비 스테이터

요즘 강남에서 미 가격대에 맨찮은 스테이크를 썰 수 있다면 가성비로는 들어가기도 전에 만족입니다. 강남역 맛집이고 인기가 많은 곳인만큼 종종 웨이팅이 있을 수... 꽃보다 청춘, 젤라홀릭 [blog.naver.com/dbswj16955](https://blog.naver.com/dbswj16955) | 블로그 내 검색 | 약도



강남역 맛집 :: 잊지못해 다시옴!

있어요 강남에 볼일 보러 갔다가 밥을 먹고 가기로 했는데요 강남역 맛집으로 유명한 곳에 어디있나 검색했는데 이탈리안 레스토랑 BOBIRED가 딱 보이더라고요 예전에... 일상기록 [blog.naver.com/llzzin11](https://blog.naver.com/llzzin11) | 블로그 내 검색 | 약도



강남역 맛집 특별한 치즈 폭포

강남역 맛집 특별한 치즈 폭포 안녕하세요 미미찡입니다. 제가 얼마 전 마초쉐프를... 최근 후에 강남역으로 가서 퍼포먼스도 구경하고 고퀄리티의 음식을 즐기면서... ♥ 미미찡의 글 까는 놀이터... [blog.naver.com/mimin...](https://blog.naver.com/mimin...) | 블로그 내 검색 | 약도



강남역 맛집 : 마초쉐프 터프한 매력!

마초쉐프 강남역점 주소 서울 강남구 강남대로98길 16 6.7층 (지번 : 서울 강남구 역삼동 817-35번지 파빌리온빌딩 6, 7층) 전화번호 02-566-8886 영업시간 오전 11시 30분 ~ 오후... 도도리의 다이어리 [blog.naver.com/dodol...](https://blog.naver.com/dodol...) | 블로그 내 검색 | 약도

## ■ 네이버 오픈 API – 블로그 검색

### □ 데이터 추출(파싱)

```
url 결과  
{'lastBuildDate': 'Thu, 31 Jan 2019 13:28:45 +0900', 'total': 1926320, 'start': 1, 'display': 10, 'items': [ {'title': '<b>비기입니다:</b> 오늘은 오개복모임에서, 다녀왔던 <b>강남역</b> 맛집에 대  
?Redirect=Log&logNo=221439805503' }, {'description': '비기입니다:</b> 오늘은 오개복모임에서, 다녀왔던 <b>강남역</b> 맛집에 대'}
```



검색 일: Thu, 31 Jan 2019 14:14:40 +0900  
총 검색 결과: 1926411  
시작 위치: 1  
출력 결과: 10

## 네이버 오픈 API – 블로그 검색

### 실행결과

- json 포맷의 결과가 출력됨

필드	타입	설명
- lastBuildDate	datetime	검색 결과를 생성한 시간이다.
- postdate	datetime	블로그 포스트를 작성한 날짜이다.
- total	integer	검색 결과 문서의 총 개수를 의미한다.
- start	integer	검색 결과 문서 중, 문서의 시작점을 의미한다.
- display	integer	검색된 검색 결과의 개수이다.
- item/items	-	XML 포맷에서는 item 태그로, JSON 포맷에서는 items 속성으로 표현된다. 개별 검색 결과이며 title, link, description, bloggername, bloggerlink을 포함한다.
- title	string	검색 결과 문서의 제목을 나타낸다. 제목에서 검색어와 일치하는 부분은 태그로 감싸져 있다.
- link	string	검색 결과 문서의 하이퍼텍스트 link를 나타낸다.
- description	string	검색 결과 문서의 내용을 요약한 페이지 정보이다. 문서 전체의 내용은 link를 따라가면 읽을 수 있다. 페이지에서 검색어와 일치하는 부분은 태그로 감싸져 있다.
- bloggername	string	검색 결과 블로그 포스트를 작성한 블로거의 이름이다.
- bloggerlink	string	검색 결과 블로그 포스트를 작성한 블로거의 하이퍼텍스트 link이다.

## ■ 네이버 오픈 API – 블로그 검색

### □ 소스코드

```
import requests

#API 인증 정보
client_id =
"CiymXRcfPNt7vfsHrdh"
client_secret = "et1QdF_Oz"

url1 = "https://openapi.naver.com/v1/search/blog?query=강남역"

headres = {'X-Naver-Client-Id':client_id,
'X-Naver-Client-Secret':client_secret}

result = requests.get(url1, headers=headres)
infomation = result.json()
print("검색 일:",
infomation['lastBuildDate']) print("총 검색
결과:", infomation['total'])
print("시작 위치:", infomation['start'])
print("출력 결과:", infomation['display'])
```

## ■ 네이버 오픈 API – 블로그 검색

### □ 소스코드

```
import requests

#API 인증 정보
client_id =
"CiymXRcfPNt7vfsHrdh"
client_secret = "et1QdF_Oz"

url1 = "https://openapi.naver.com/v1/search/blog?query=강남역"

headres = {'X-Naver-Client-Id':client_id,
'X-Naver-Client-Secret':client_secret}

result = requests.get(url1, headers=headres)
information = result.json()
print("items:",
infomation['items'])
```

## 네이버 오픈 API – 블로그 검색

### 실행결과 json 포맷의 결과가 출력됨

```
In [32]: infomation['items']
```

```
Out[32]: [ {'title': '<b>강남역</b> 맛집 특별한 치즈 폭포',
  'link': 'https://blog.naver.com/miminimkkk?Redirect=Log&logNo=221579084618',
  'description': '<b>강남역</b> 맛집 특별한 치즈 폭포 만녕하세요 미미짱입니다. 제가 얼마 전 마초쉐프를 다녀오
  게 되었어요.... 최근 후에 <b>강남역</b>으로 가서 폴포먼스도 구경하고 고퀄리티의 음식을 즐기면서 만족스러운 식
  사를 했기....',
  'bloggername': '♥ 미미짱의 꿀 까는 놀이터 ♥',
  'bloggerlink': 'https://blog.naver.com/miminimkkk',
  'postdate': '20190706'},
  {'title': '<b>강남역</b> 맛집 :: 미친 갓성비 스테이터',
  'link': 'https://blog.naver.com/dbswjd6955?Redirect=Log&logNo=221584500795',
  'description': '요즘 강남에서 이 가격대에 괜찮은 스테이크를 썸 수 있다면 가성비로는 들어가기도 전에 만족입니
  다. <b>강남역</b> 맛집이고 인기가 많은 곳인만큼 종종 웨이팅이 있을 수 있어요. 밖에 요렇게 의자도 준비되어 있
  고....',
  'bloggername': '꽃보다 청춘, 젤라홀릭',
  'bloggerlink': 'https://blog.naver.com/dbswjd6955',
  'postdate': '20190713'},
  {'title': '<b>강남역</b> 맛집 :: 및지못해 다시옴!',
  'link': 'https://blog.naver.com/lizzin11?Redirect=Log&logNo=221567844263',
  'description': '있어요 강남에 볼일 보러 갔다가 밥을 먹고 가기로 했는데요 <b>강남역</b> 맛집으로 유명한 곳에
  어디있나 검색했는데 미탈리안 레스토랑 BOBIPED가 딱 보이더라고요 예전에 홍대 매장에서 정말 맛나게 먹었는데...
  ',
  'bloggername': '일상기록',
  'bloggerlink': 'https://blog.naver.com/lizzin11',
  'postdate': '20190622'}]
```

## ■ 네이버 오픈 API – 블로그 검색

### □ 소스코드

```
import requests

#API 인증 정보
client_id =
"CiymXRcfPNt7vfsHrdh"
client_secret = "et1QdF_Oz"

url1 = "https://openapi.naver.com/v1/search/blog?query=강남역"

headres = {'X-Naver-Client-Id':client_id,
'X-Naver-Client-Secret':client_secret}

result = requests.get(url1, headers=headres)
information = result.json()
print("item:",
infomation['items'][0])
#print("item:", infomation['items'][1])
```

# 네이버 오픈 API

## 네이버 오픈 API

### 실행결과 json 포맷의 결과가 출력됨

```
In [34]: infomation['items'][0]
```

```
Out[34]: {'title': '<b>강남역</b> 맛집 특별한 치즈 폭포',
           'link': 'https://blog.naver.com/miminimkkk?Redirect=Log&logNo=221579084618',
           'description': '<b>강남역</b> 맛집 특별한 치즈 폭포 안녕하세요 미미찡입니다. 제가 얼마 전 마초쉐프를 다녀오게 되었어요.... 퇴근 후에 <b>강남역</b>으로 가서 퍼포먼스도 구경하고 고퀄리티의 음식을 즐기면서 만족스러운 식사를 했기....',
           'bloggername': '♥ 미미찡의 글 까는 놀이터 ♥',
           'bloggerlink': 'https://blog.naver.com/miminimkkk',
           'postdate': '20190706'}
```



강남역 맛집 특별한 치즈 폭포 2019.07.06.

강남역 맛집 특별한 치즈 폭포 안녕하세요 미미찡입니다. 제가 얼마 전 마초쉐프를... 퇴근 후에 강남역으로 가서 퍼포먼스도 구경하고 고퀄리티의 음식을 즐기면서...

♥ 미미찡의 글 까는 놀이터... [blog.naver.com/mimin...](https://blog.naver.com/miminimkkk) | 블로그 내 검색 | 약도 ▾

```
In [36]: infomation['items'][1]
```

```
Out[36]: {'title': '<b>강남역</b> 맛집 :: 미친 간성비 스테이터',
           'link': 'https://blog.naver.com/dbswjdg6955?Redirect=Log&logNo=221584500795',
           'description': '요즘 강남에서 이 가격대에 맨찮은 스테이크를 썰 수 있다면 가성비로는 들어가기도 전에 만족입니다. <b>강남역</b> 맛집이고 인기가 많은 곳인만큼 종종 웨이팅이 있을 수 있어요. 밖에 요렇게 의자도 준비되어 있고...',
           'bloggername': '꽃보다 청춘, 젤라홀릭',
           'bloggerlink': 'https://blog.naver.com/dbswjdg6955',
           'postdate': '20190713'}
```



강남역 맛집 :: 미친 간성비 스테이터 머제

요즘 강남에서 이 가격대에 맨찮은 스테이크를 썰 수 있다면 가성비로는 들어가기도 전에 만족입니다. 강남역 맛집이고 인기가 많은 곳인만큼 종종 웨이팅이 있을 수...

꽃보다 청춘, 젤라홀릭 [blog.naver.com/dbswj...](https://blog.naver.com/dbswjdg6955) | 블로그 내 검색 | 약도 ▾

# 스스로 해보기 – 네이버 오픈 API

## ■ 네이버 오픈 API

□ 데이터 추출 : “글 제목 @ 블로그 이름 @ 글 주소 링크”의 형식으로 출력

[강남역 맛집] 청류벽 – 별미 들기름막국수를 찾아주세요 @여의도 직장인 문선의 밥상 (舊  
을지로) @<http://blog.naver.com/sielle83?Redirect=Log&logNo=221453985380> 강남역 데이트 : 이색적인 매력@Youriful day

'.'@<http://blog.naver.com/youriful?Redirect=Log&logNo=221454784237>

강남역 맛집 맛으로 승부하네요@이프리의 프리카메라@ <http://blog.naver.com/cherrytue?Redirect=Log&logNo=221451755057>

:: 강남역 맛집 리얼리 맛있는 곳!@내 맘속의 풍경(風景) 하나 바람 속의 풍경(風聲)

들@<http://blog.naver.com/1004coucou?Redirect=Log&logNo=221448927833> 강남역 맛집 :: 뜬땀(feat.곱창쌀국수)@찌주의 태일라블로그

♡@<http://blog.naver.com/0122ki?Redirect=Log&logNo=221454935497>

강남역 한정식 가족모임에 언제나 좋은 곳@인생 뭐 있나…@ <http://blog.naver.com/rickydoll?Redirect=Log&logNo=221453758512>

# 스스로 해보기 – 네이버 오픈 API

## ■ 네이버 오픈 API – 블로그 검색

### ▣ 소스코드

```
import requests

#API 인증 정보
client_id = "CiymXRcfPNt7vfsHrdh1" client_secret = "et1QdF_OzG"

#header에 추가될 내용
headres = {'X-Naver-Client-Id':client_id, 'X-Naver-Client-Secret':client_secret}

def get_api_result(keyword, display, start):
    url = "https://openapi.naver.com/v1/search/blog?query=" + keyword \
    + "&display=" + str(display) \
    + "&start=" + str(start) result=requests.get(url, headers=headres)
    return result.json()

def call_and_print(keyword, page):
    json_obj = get_api_result(keyword, 20, page) for item in json_obj['items']:
        title = item['title'].replace("<b>", "").replace("</b>", "") print(title + "@"+ item['bloggername'] \
        + "@" + item['link'])

keyword = "강남역"
call_and_print(keyword, 1)
```

# Naver 지식인 검색

## Naver 지식인 검색

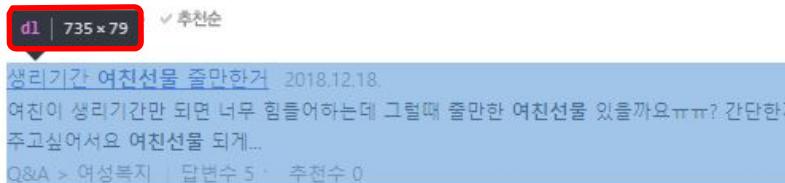
### □ Naver 지식인에 접속

- <https://kin.naver.com/index.nhn>

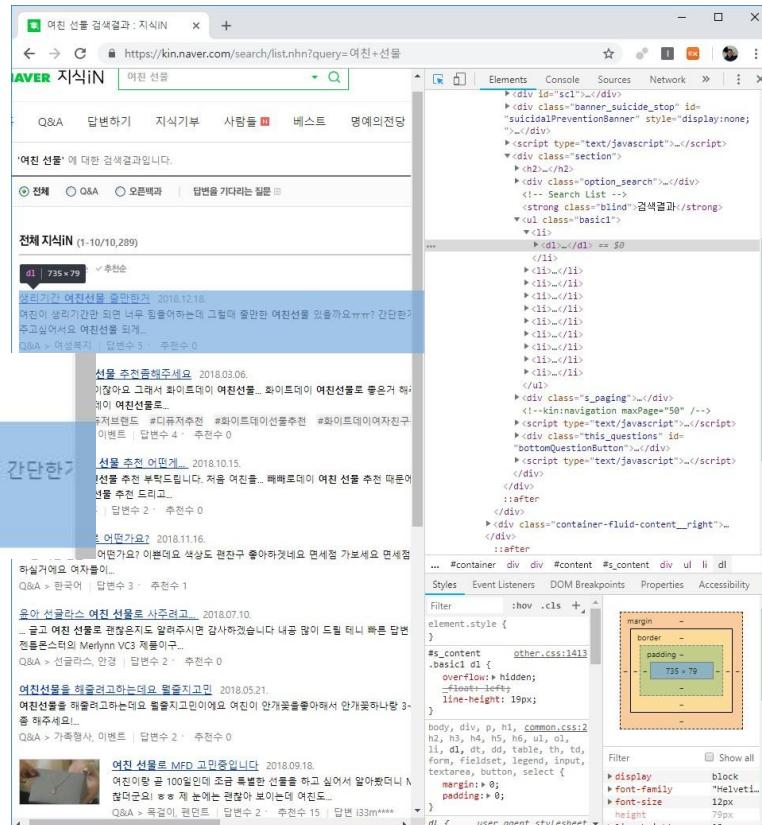
### □ 검색 키워드 '여친 선물'로 검색

### □ 마우스 우클릭 후 '검사(N)'을 클릭

### □ 웹 페이지 구조 파악



- <d1> ... </d1>



# Naver 지식인 검색

## Naver 지식인 검색

### □ Naver 지식인에 접속

- <https://kin.naver.com/index.nhn>

### □ 검색 키워드 '여친 선물'로 검색



여친 선물 검색결과 : 지식iN

NAVER 지식iN

여친 선물

총 Q&A 답변하기 지식기부 사람들 베스트 명예의전당 프로필 파트너센터

질문하기

여친 선물'에 대한 검색결과입니다.

전체 Q&A 오픈백과 답변을 기다리는 질문

전체 지식iN (1-10/10,434)

정확도 최신순 추천순

설리기간 여친선물 출만한거 2018.12.18

여친이 생리기간만 되면 너무 힘들어하는데 그럴때 출만한 여친선물을 있음까요ㅠㅠ? 간단한거라도 좋아용. 생리기간에 힘내라고 여친선물 주고싶어서요 여친선물 되게...

Q&A > 여성복지 | 답변수 5 · 추천수 0

연관검색어

여자 물걸이 여친 생일선물 여친선물추천 여름인 세일 여친 100일 선물 도움말

### □ 2페이지로 이동



여친 선물 검색결과 : 지식iN

NAVER 지식iN

여친 선물

총 Q&A 답변하기 지식기부 사람들 베스트 명예의전당 프로필 파트너센터

질문하기

여친 선물'에 대한 검색결과입니다.

전체 Q&A 오픈백과 답변을 기다리는 질문

전체 지식iN (11-20/10,434)

정확도 최신순 추천순

여친선물 가방이나 지갑 2018.06.01

가방이나 지갑입니다 저울 선물하는거라서 비싼거 주고 싶네요 미국브랜드 보가타 을더썩은 어떠세요~? 제가 선물받아서 잘 쓰고 있는

연관검색어

여자 물걸이 여친 생일선물 여친선물추천 여름인 세일 여친 100일 선물 도움말

# 한글 폰트 설정

```
In [1]: import pandas as pd
import numpy as np

import platform
import matplotlib.pyplot as plt

%matplotlib inline

from matplotlib import font_manager, rc
font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

# 네이버 지식인 검색

```
In [24]: from bs4 import BeautifulSoup
from urllib.request import urlopen
import urllib
import time
```

```
In [32]: tmp1 = 'https://search.naver.com/search.naver?where=kin'
html = tmp1 + '&sm=tab_jum&ie=utf8&query={key_word}&start={num}'

response = urlopen(html.format(num=1, key_word=urllib.parse.quote('여친 선물')))
```

```
In [33]: soup = BeautifulSoup(response, "html.parser")
soup
```

```
Out[33]: <!DOCTYPE doctype html>
<html lang="ko"> <head> <meta charset="utf-8"/> <meta content="always" name="referrer"/> <meta content="telephone=no, address=no, email=no" name="format-detection"/> <meta content="width=device-width, initial-scale=1.0, maximum-scale=2.0" name="viewport"/> <meta content="여친 선물 : 네이버 지식인검색" property="og:title"> <meta content="https://ssl.pstatic.net/sstatic/search/common/og_v3.png" property="og:image"/> <meta content="여친 선물'의 네이버 지식인검색 결과입니다." property="og:description"/> <meta content="여친 선물'의 네이버 지식인검색 결과입니다." lang="ko" name="description"/> <title>여친 선물 : 네이버 지식인검색</title> <link href="https://ssl.pstatic.net/sstatic/search/favicon/favicon_140327.ico" rel="shortcut icon"/> <link href="https://ssl.pstatic.net/sstatic/search/opensearch-description.xml" rel="search" title="Naver" type="application/opensearchdescription+xml"><link href="https://ssl.pstatic.net/sstatic/search/pc/css/search1_191010.css" rel="stylesheet" type="text/css"/> <link href="https://ssl.pstatic.net/sstatic/search/pc/css/search2_191010.css" rel="stylesheet" type="text/css"/> <link href="https://ssl.pstatic.net/sstatic/search/pc/css/api_atcmp_190612.css" rel="stylesheet" type="text/css"/><script type="text/javascript"> if (!String.prototype.trim) { String.prototype.trim = function () { return this.replace(/^[\u200b\uFEFF\uAO]+|[^\u200b\uFEFF\uAO]+$/g, ''); } } if (!Array.prototype.indexOf) { Array.prototype.indexOf = function(searchElement, fromIndex) { var k; if (this == null) { throw new TypeError("this is null or not defined"); } var o = Object(this); var len = o.length >> 0; if (len === 0) { return -1; } var n = fromIndex > 0; if (n >= len) { return -1; } k = Math.max(n >= 0 ? n : len - Math.abs(n), 0); while (k < len) { if (k in o && o[k] === searchElement) { return k; } k++; } return -1; } } if (typeof(encodeURIComponent) != "function") { encodeURIComponent = function (s) { function toHex (n) { var hexchars = "0123456789ABCDEF"; return "%" + hexchars.charAt(n>4) + hexchars.charAt(n&0xF); } var es = ""; for (var i = 0; i < s.length;) { var c = s.charCodeAt(i++); if ((c&0xF800) == 0xD800) { var sc = s.charCodeAt(i++); c = ((c-0xD800)<<10) + (sc-0xDC00) + 0x10000; } if (!(c&0x7F)) { if ((c>=65&&c<=90) || (c>=97&&c<=122) || (c>=48&&c<=57) || (c>=45&&c<=46) || c==95 || c==33 || c==126 || (c>=39&&c<=42)) es += String.fromCharCode(c); else es += toHex(c); } else es += toHex(c); } } } }>
```

# 네이버 지식인 검색

```
In [34]: tmp = soup.find_all('dl')
```

```
tmp
```

```
Out [34]: [<dl class="sch_option"> <dt><label for="inpop0">기본검색</label></dt> <dd> <div class="inp_op"><input class="_base" id="inpop0" title="기본검색어 입력" type="text"/></div> <p class="rule_check"><input class="incr_or" id="rch" onclick="tCR('a=fno.dtors');" type="checkbox"/> <label for="rch">입력한 단어가 하나이상 포함된 문서 검색</label></p> </dd> </dl>, <dl class="sch_option more_sch"> <dt><label for="inpop1">상세검색</label></dt> <dd> <div class="inp_op"><input class="_exact" id="inpop1" type="text"/><label class="_placeholder_exact" for="inpop1">정확히 일치하는 단어/문장(" ")</label></div> <div class="inp_op"><input class="_include" id="inpop2" type="text"/><label class="_placeholder_include" for="inpop2">반드시 포함하는 단어(+)</label></div> <div class="inp_op"><input class="_exclude" id="inpop3" type="text"/><label class="_placeholder_exclude" for="inpop3">제외하는 단어(-)</label></div> <div class="tx_area"> <p class="tx_noti">기본검색의 결과 범위를 줄이고자 할 때 사용합니다.<br/>여러개의 단어를 입력하실 때는<span class="tx">쉼표(,)</span>로 구분해서 입력하세요.</p> </div> </dd> </dl>, <dl class="relate_area _related_keyword_area"> <dt><span class="tit_relate _related_keyword_lis">연관검색어</span><a class="link_help" href="https://help.naver.com/support/alias/search/#word/#word_1.naver" onclick="return goOtherCR(this, 'a=rsk.guide&r=&i=' + encodeURIComponent(urlExpand(this.href)))" target="_blank"><i class="spnnew_api_ico_help">도움말</i></a></dt> <dd class="lst_relate _related_keyword_list" style="display: block;"> <ul class="_related_keyword_ul"> <li> <a data-area="*q" data-idx="1" href="?where=kin&query=%EB%80%98%EC%A7%80%EA%80%91&ie=ut f8&sm=tab_she&qdt=0">반지갑</a> </li> <li> <a data-area="*q" data-idx="2" href="?where=kin&query=%E9%97%AC%EC%B9%9C+%EC%83%9D%EC%9D%BC%EC%84%AD%EB%AC%BC&ie=ut f8&sm=tab_she&qdt=0">여친 샐일선물</a> </li> <li> <a data-area="*q" data-idx="3" href="?where=kin&query=sk2&ie=ut f8&sm=tab_she&qdt=0">sk2</a> </li> <li> <a data-area="*q" data-idx="4" href="?where=kin&query=%EC%9D%BC%EB%AB%AC%EC%95%A4&ie=ut f8&sm=tab_she&qdt=0">릴리앤</a> </li> <li> <a data-area="*q" data-idx="5" href="?where=kin&query=%EC%88%98%EC%97%90%EB%A5%BA%EC%96%BC+%EC%84%9D%EB%A5%98&ie=ut f8&sm=tab_she&qdt=0">수메르에 석류</a> </li> <li> <a data-area="*q" data-idx="6" href="?where=kin&query=sk2+%ED%94%BC%ED%85%8C%EB%9D%BC+%EC%97%90%EC%84%BC%EC%8A%4&ie=ut f8&sm=tab_she&qdt=0">sk2 피테라 에센스</a> </li> <li> <a data-area="*q" data-idx="7" href="?where=kin&query=%ED%8C%8C%EB%8B%88%EB%8B%88%EB%80%1&ie=ut f8&sm=tab_she&qdt=0">파니니백</a> </li> <li> <a data-area="*q" data-idx="8" href="?where=kin&query=%EC%97%AC%EC%9E%90%EC%89%9C%EA%B5%AC+%EC%84%AD%EB%AC%BC&ie=ut f8&sm=tab_she&qdt=0">여자친구 선물</a> </li> </ul> </dd> <dd class="closed _related_keyword_closed" style="display:none"> 닫기 후 1주일간 유지됩니다. 연관검색어를 다시 보시겠습니까? <a class="open _related_keyword_toggle" href="#" onclick="return false;" role="button">열기</a></dd> </dl>, <dl> <dt class="question"> <strong class="spkn">질문</strong> <a href="https://kin.naver.com/qna/detail.nhn?d1_id=9&dir_id=90107&do_id=334667036&qb=7Jes7Lmc10yEo0usvA==&enc=ut f8&ion=kin&rank=1&search_sort=0&spq=1" onclick="return goOtherCR(this, 'a=kin+q.tit&r=1&i=10000009_000013f29d1c&u=' + encodeURIComponent(urlExpand(this.href)))" target="_blank"> <strong class="h1">여친선물</strong>로 생로랑 틴트 펄이 좋아보이길래 틴트랑 이것저것 필요한걸로 좀 사줄려구 하는데요. 사는지역이 구미인데 구매 할수있는곳이 있나요? 아니면 대구 신세계... </dt> <dd class="answer"> <strong class="spkn">답변</strong> 백화점에서 구매하는게 좋습니다 일단 다른 오프매장은 마땅히 구매할 만한 곳이 없구요! 또 유통채널들 중에서 가장 다양하게 틴트 라인업 보유하고 있고 재고가 충분하... </dd> <dd class="txt_block"> <span class="inline"> </span> </dd> </dl>,
```

# 네이버 지식인 검색

```
In [3]: tmp_list = []
#tmp에서 1개의 d1을 리턴받아서 line에 대입
for line in tmp:
    #line의 텍스트를 tmp_list에 추가
    tmp_list.append(line.text)

tmp_list
```

```
Out [3]: [' 기본검색 입력한 단어가 하나이상 포함된 문서 검색 ',
  ' 상세검색 정확히 일치하는 단어/문장(" ") 반드시 포함하는 단어(+) 제외하는 단어(-) 기본검색의 결과 범위를 줄이고자 할 때 사용합니다.
  여러개의 단어를 입력하실 때는逗표(,)로 구분해서 입력하세요. ',
  ' 연관검색어도움말 반지갑 여친 생일선물 일리맨 sk2 sk2 피테라 메센스 파니니백 수에르예 석류 달기 후 1주일간 유지됩니다. 연관검색어를 다시 보시겠습니까? 열기 ',
  ' 질문 여친선물로 생로랑 틴트 평이 좋아보이 2019.08.30. 여친선물로 생로랑 틴트 평이 좋아보이길래 틴트랑 이것저것 필요한걸로 좀
  사줄려구 하는데요. 사는지역이 구미인데 구미에 구매 할수있는곳이 있나요? 아니면 대구 신세계... 답변 백화점에서 구매하는게 좋습니다 일단 다른 오프매장은 마땅히 구매할 만한 곳이 없구요! 또 유통채널들 중에서 가장 다양하게 틴트 라인업 보유하고 있고 재고가 충분하... ',
  ' 질문 생리기간 여친선물 출만한거 2018.12.18. 여친이 생리기간만 되면 너무 힘들어하는데 그럴때 출만한 여친선물 있을까요ㅠㅠ? 간단한거라도 좋아용..생리기간에 힘내라고 여친선물 주고싶어서요 답변 생리기간에는 예민해서 여친선물로 여자친구가 조아하는걸로 주는게 좋을 거 같아요 생리기간 잘 이겨내도록 좋은 여친선물 고르세요! ',
  ' 질문 10대 여친 선물 2019.10.18. 이제 기념일이 다가와서 그런데요 여자친구한테 무슨 선물하는게 좋을까요?? 할수 말고요..ㅎ 자세히 알려주세요^^ 답변 토닥토닥워터 너의이름은장미를 추천해요 장미와 건조레몬, 딸기 등이 블랜딩 되어 있어 상큼하고 맛있게 먹을수 있어요 간편하게 보틀에 우려 마시면 되고... ',
  ' 질문 여친 선물 2019.10.11. 여자들이 좋아할만한 선물이 무엇이 있나요?? 곧 200일이라... 답변 안녕하세요 수제가죽 브랜드 톰크롬입니다. 여자친구분에게 선물로 수제가죽으로 만든 카드지갑은 어떨까요? 수제가죽으로 만든 지갑에 관심이 있으시면 저희 브랜드 한번... ',
  ' 질문 여친 선물 2019.10.21. 여자친구랑 100일 됐는데 선물은 뭘로 줘야 좋을까요??중학생입니다 추천 부탁드려요 답변 백일선물로는
```

## 페이지를 이동하면서 여친 선물 검색

```
In [*]: #프로그래스바를 출력하는 모듈
from tqdm import tqdm_notebook
#여자친구 선물을 검색해서 저장할 리스트
present_candi_text = []
#1~1000 까지 10씩 증가하고 프로그래스바를 출력
for n in tqdm_notebook(range(1, 1000, 10)):
    #네이버 검색
    response = urlopen(html.format(num=n, key_word=urllib.parse.quote('여자 친구 선물')))

    #검색정보 soup에 대입
    soup = BeautifulSoup(response, "html.parser")
    ##1태그를 검색해서 tmp에 대입
    tmp = soup.find_all('div')
    #tmp에서 1개의 div 라인
    for line in tmp:
        #present_candi_text에 추가
        present_candi_text.append(line.text)
    #0.5초 대기
    time.sleep(0.5)
```

26%  26/100 [00:19<00:56, 1.32it/s]

```
In [5]: present_candi_text
```

Out [5]: ['기본검색 입력한 단어가 하나이상 포함된 문서 검색', '상세검색 정확히 일치하는 단어/문장(“”) 반드시 포함하는 단어(+) 제외하는 단어(-) 기본검색의 결과 범위를 줄이고자 할 때 사용합니다. 여러개의 단어를 입력하실 때는 쉼표(,)로 구분해서 입력하세요.', '연관검색어도움말 여자친구목록이 스크린케어 여자스킨로션추천 여자30대선물 에센스추천 20대여자친구선물 여자친구선물뷰티다 바이스 선물 여자친구생일선물 여자근력운동 달기 후 1주일간 유지됩니다. 연관검색어를 다시 보시겠습니까? 열기', '질문 20대 여자친구 선물로 어떤 걸 해줘야 할까요?? 2019.06.10. 이번에는 좀 색다른 걸 해주고 싶은데요. 웬만한 20대 여자친구 선물로 해줄건 다... 이미 출 건 다 쳐봤습니다. 20대 여자친구 선물로 실용적인 거 있음 추천해주세요. 답변 20대 여자친구 선물에 대한 답변 드립니다. 20대 여자친구 선물로 보편적인 것은 반지, 팔찌, 목걸이 등등의... 이상으로 20대 여자친구 선물에 대한 답변이었습니다.', '질문 여자친구 선물(20-30)대 랜찮은거 있나요? 2019.08.26. 안녕하세요 여자친구 선물로 고민하고 있습니다 늘 기념일마다... 싶은 여자친구 선물이었을 좋겠는데 가격은 큰 상관없지만 특별하고 간직할 만한 그런 여자친구 선물이면... 답변 여자친구 선물 이건 누구나 한번쯤 고민해 봤을 것... 노미네이션을여자친구 선물로 추천하는 이유는 여전과의... 좀 캐주얼하고 부담없는 가격대 여자친구 선물 중에서는...', '질문 30대여자친구선물 여려개 추천좀 해주세요 2019.08.22. 30대여자친구선물로 좋은 거 여려개 추천해주세요. 그 중에서 골라서 선물 하던지... 그동안 30대여자친구선물이라고 할은 할수, 시계, 목걸이, 등등 딱히 업그레이드 좋아했던... 답변 30대여자친구선물 같은 경우는 실용적인 걸 원하는 경우가 많습니다. 그간 사줬던... 답변처럼 좀 더 건강에 길숙히 관리된 30대여자친구선물이면 더 좋아할거 같습니다.', '질문 20대여자친구선물 뭘 죄여 좋아할까요? 2019.08.13. 20대여자친구선물로 엄마에게 좋을까요? 답변 안녕하세요 20대여자친구선물 어떤걸로 준비할지 고민하셔네요 제가 좋은것 알려드리겠습니다 20대여자친구선물로 준비했는데 너무 좋아하네요 좋은것 알려드릴게요.', '질문 여자친구 선물 출건데요... 인형에 세 2019.10.30. 여자친구 선물 출건데요... 인형에 세밀 문구좀 골라주세요. 유자니~ 유자니 나루토~ 유자친구 선물로 출건데요... 인형에 세밀 문구좀 골라주세요. 유자니~ 유자니~ 답변 여자친구 선물, 디자인, 세밀 문구, 인형, 출건데요, 여자친구 선물, 유자니~ 유자니~ 나루토~']

# 단어 분리

```
In [38]: #저장된 데이터의 수를 조회  
len(present_candi_text)
```

```
Out [38]: 1300
```

```
In [42]: import nltk  
from konlpy.tag import Twitter  
twitter = Twitter()
```

```
In [43]: present_candi_text[0]
```

```
Out [43]: '기본검색 입력한 단어가 하나이상 포함된 문서 검색 '
```

```
In [46]: tagged = twitter.pos(present_candi_text[0])  
tagged
```

```
Out [46]: [('기본', 'Noun'),  
           ('검색', 'Noun'),  
           ('입력', 'Noun'),  
           ('한', 'Josa'),  
           ('단어', 'Noun'),  
           ('가', 'Josa'),  
           ('하나', 'Noun'),  
           ('이상', 'Noun'),  
           ('포함', 'Noun'),  
           ('되다', 'Verb'),  
           ('문서', 'Noun'),  
           ('검색', 'Noun')]
```

```
In [47]: tagged = twitter.pos(present_candi_text[0], stem=True)  
tagged
```

```
Out [47]: [('기본', 'Noun'),  
           ('검색', 'Noun'),  
           ('입력', 'Noun'),  
           ('한', 'Josa'),  
           ('단어', 'Noun'),  
           ('가', 'Josa'),  
           ('하나', 'Noun'),  
           ('이상', 'Noun'),  
           ('포함', 'Noun'),  
           ('되다', 'Verb'),  
           ('문서', 'Noun'),  
           ('검색', 'Noun')]
```

# 단어 분리

```
In [89]: for i in range (len(tagged)):
          print(tagged[i][0])
```

기검입한단가하미포되문검

```
In [72]: tag_list = []
for text in present_candi_text:
    tagged = twitter.pos(text,stem=True)
    for i in range (0, len(tagged)):
        tag_list.append(tagged[i][0])

tag_list
```

Out [72]: [기본적인  
제작법에  
대한  
가장  
하나  
이상  
포함된다.  
제작  
세계  
상의  
제작  
활  
용처  
이다.  
단문장  
반복

# 가장 많이 사용된 단어 100개 조회

```
In [106]: ko = nltk.Text(tokens_ko)
ko.vocab().most_common(100)
```

```
Out [106]: [(['선물', 6200),
 ('여자친구', 5800),
 ('추천', 800),
 ('30', 800),
 ('좋은', 800),
 ('♡', 600),
 ('화장품', 600),
 ('검색', 500),
 ('걸', 500),
 ('고민', 500),
 ('너무', 500),
 ('여러', 400),
 ('여자', 400),
 ('적', 400),
 ('인', 400),
 ('해주세요', 400),
 ('!', 400),
 ('지갑', 400),
 ('입력', 300),
 ('할인', 200)]
```

# 특수 문자 제거

```
In [74]: import re
pattern="[^#\uac00-\ud7a5s]"
present_candi_text = [re.sub(pattern=pattern, repl='',string=text) for text in present_candi_text]
present_candi_text
```

```
Out [74]: ['기본검색 입력한 단어가 하나이상 포함된 문서 검색',
'상세검색 정확히 일치하는 단어문장 반드시 포함하는 단어 제외하는 단어 기본검색의 결과 범위를 줄이고자 할 때 사용합니다여러개의 단어를 입력하실 때는쉼표로 구분해서 입력하세요',
'연관검색어도움말 여자친구목걸이 스킨케어 여자스킨로션추천 여자30대선물 에센스추천 20대여자친구선물 여자친구선물뷰티디바이스 선물 여자친구생일선물 여자근력운동 달기 후 1주일간 유지됩니다연관검색어를 다시 보시겠습니까 열기',
'질문 20대 여자친구 선물로 어떤 걸 해줘야 할까요 20190610 이번에는 좀 색다른 걸 해주고 싶은데요 웬만한 20대 여자친구 선물로 해줄건 다 이미 줄 건 다 줘봤습니다20대 여자친구 선물로 실용적인 거 있음 추천해주세요 답변 20대 여자친구 선물에 대한 답변 드립니다 20대 여자친구 선물로 보편적인 것은 반지 팔찌 목걸이 등등의 이상으로 20대 여자친구 선물에 대한 답변이었습니다',
'질문 여자친구 선물2030대 괜찮은거 뭐 있나요 20190826 안녕하세요 여자친구 선물로 고민하고 있습니다 늘 기념일마다 싫은 여자친구 선물이였음 좋겠는데 가격은 큰 상관없지만 특별하고 간직할 만한 그런 여자친구 선물이면 답변 여자친구 선물 이건 누구나 한번쯤 고민해 봤을 것 노미네이션을여자친구 선물로 추천하는 이유는 여친과의 좀 캐쥬얼하고 부담없는 가격대 여자친구 선물 중에서는',
'질문 30대여자친구선물 여러개 추천좀 해주세요 20190822 30대여자친구선물로 좋은 거 여러개 추천해주세요 그 중에서 골라서 선물하던 지 그동안 30대여자친구선물이라고 함은 향수 시계 목걸이 등등 딱히 엄청나게 좋아했던 답변 30대여자친구선물 같은 경우는 실용적인 걸 원하는 경우가 많습니다 그간 사줬던 답변처럼 좀 더 건강에 길숙히 관련된 30대여자친구선물이면 더 좋아할거 같습니다',
'질문 20대여자친구선물 뭘 줘야 좋아할까요 20190813 20대여자친구선물 검색해도 다 거기서 거기인 선물들만 뜨니깐 여기다 질문 남겨 봅니다 새로운걸 선물해주고 싶은데 20대여자친구선물로 어떤게 좋을까요 답변 안녕하세요 20대여자친구선물 어떤걸로 준비할지 고민하시네요 제가 좋은것 알려드리겠습니다 20대여자친구선물로 준비했는데 너무 좋아하네요 좋은것 알려드릴게요',
'질문 여자친구 선물 줄건데요 인형에 세 20191030 여자친구 선물 줄건데요 인형에 세길 문구좀 골라주세요 유지니 유지나 사랑해 김유진 사랑해 난 항상 나면 김유진 내꾸 자기야 힘내 답변 난 항상 나면',
'질문 20대여자친구선물 어떤걸로 준비해주세요 20190810 고 그 퀸그 세이어 디자이너 2021여자친구 선물카드 미切尔 깨끗하고 깔끔합니다']
```

# 단어길이 2미만인 단어 제거

```
In [93]: tag_list = []
for text in present_candi_text:
    tagged = twitter.pos(text, stem=True)
    for i in range(0, len(tagged)):
        tag_list.append(tagged[i][0])
```

```
tag_list
```

```
Out [93]: ['기본',
'검색',
'일렉',
'한',
'단어',
'가',
'하나',
'이상',
'포함',
'되다',
'문서',
'검색',
'상세',
'검색',
'정확하다',
'일치',
'하다',
'단어',
'문장',
'나는']
```

```
In [94]: tag_list=[tag for tag in tag_list if len(tag)>=2]
tag_list
```

```
Out [94]: ['기본',
'검색',
'일렉',
'단어',
'하나',
'이상',
'포함',
'되다',
'문서',
'검색',
'상세',
'검색',
'정확하다',
'일치',
'하다']
```

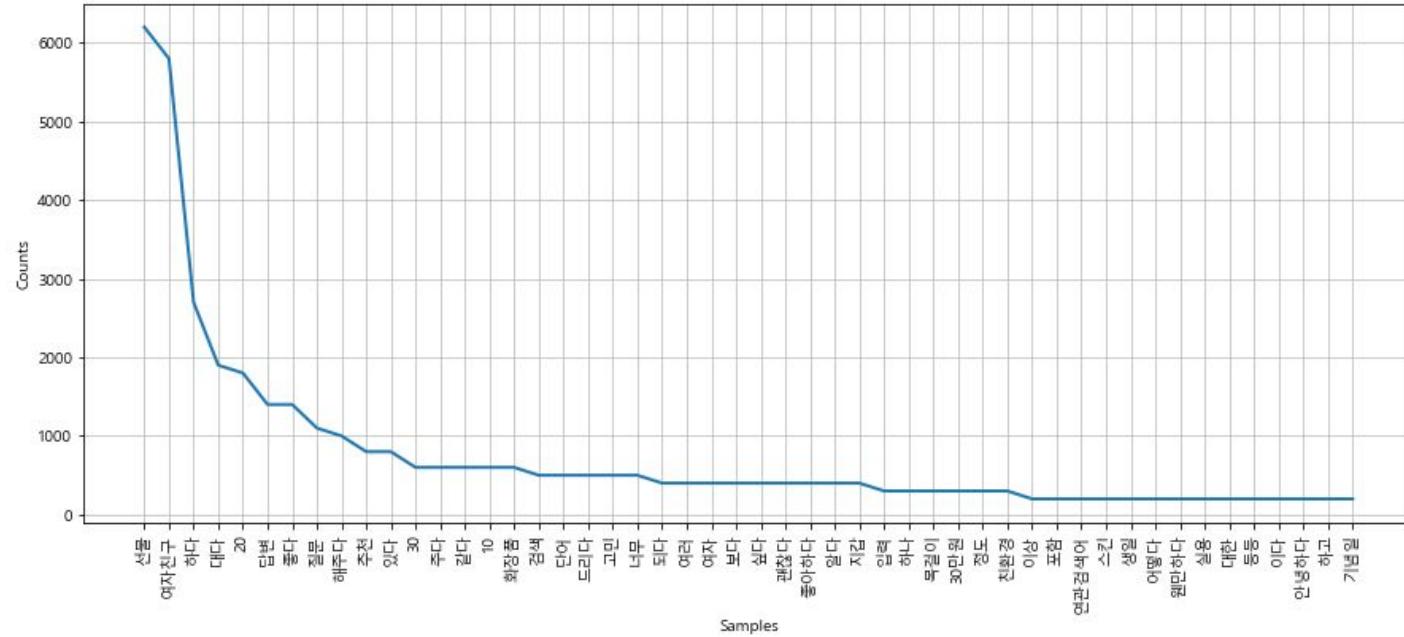
# 가장 많이 사용된 단어 200개 조회

```
In [110]: ko = nltk.Text(tag_list)
ko.vocab().most_common(200)
```

```
Out[110]: [('선물', 6200),
 ('여자친구', 5800),
 ('하다', 2700),
 ('대다', 1900),
 ('20', 1800),
 ('답변', 1400),
 ('좋다', 1400),
 ('질문', 1100),
 ('해주다', 1000),
 ('추천', 800),
 ('있다', 800),
 ('30', 600),
 ('주다', 600),
 ('같다', 600),
 ('10', 600),
 ('화장품', 600),
 ('검색', 500),
 ('단어', 500),
 ('드리다', 500),
 ('... ', 500)]
```

# 단어 빈도수 시각화

```
In [112]: plt.figure(figsize=(15,6))
ko.plot(50)
plt.show()
```



# 워드 클라우드 시각화

```
In [113]: from wordcloud import WordCloud, STOPWORDS  
         from PIL import Image
```

```
In [114]: data = ko.vocab().most_common(300)

wordcloud = WordCloud(font_path='c:/Windows/Fonts/malgun.ttf',
                      relative_scaling = 0.2,
                      stopwords=STOPWORDS,
                      background_color='white',
                      ).generate_from_frequencies(dict(data))

plt.figure(figsize=(16,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

