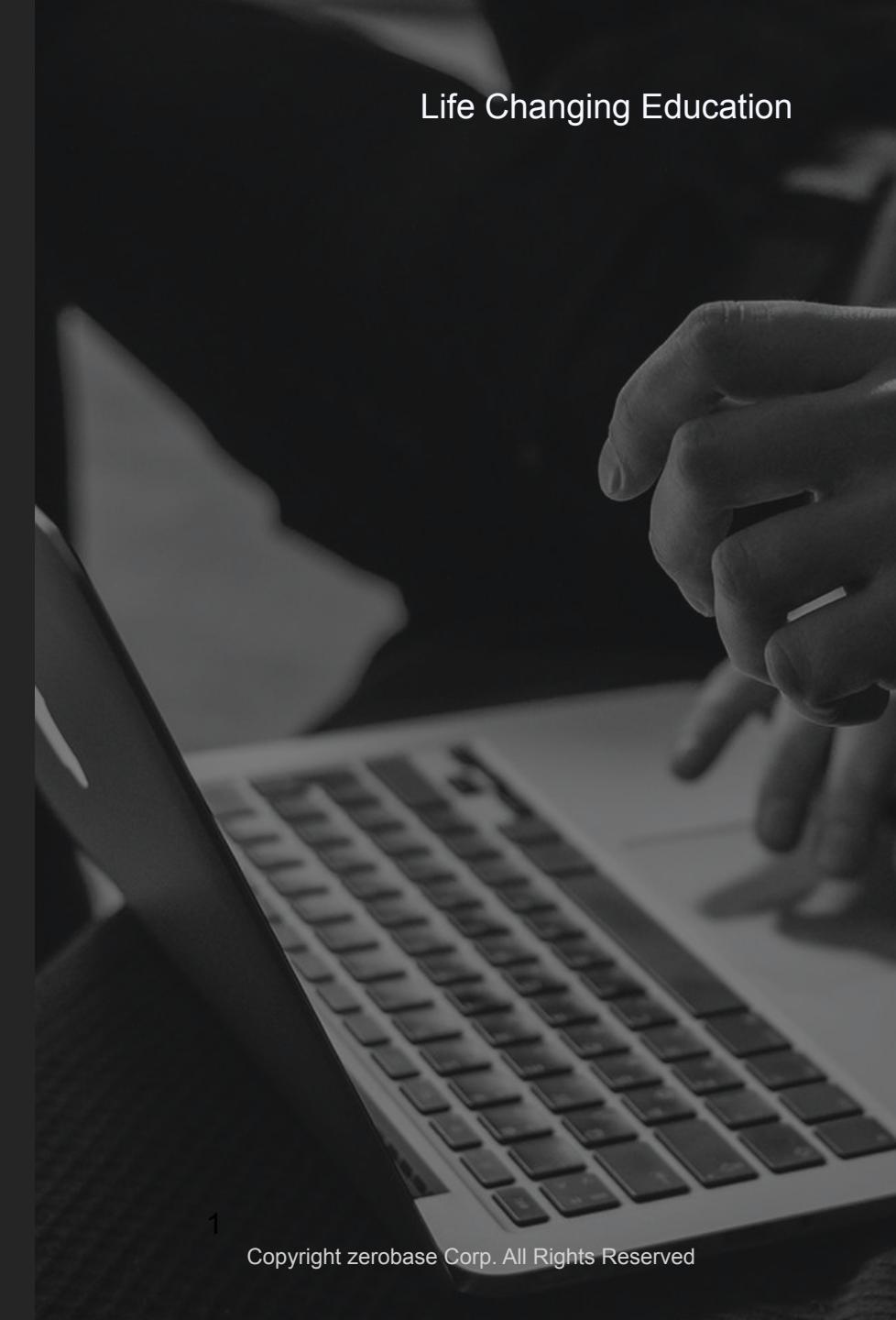


Project 03.

웹 데이터 수집하고 정리하기

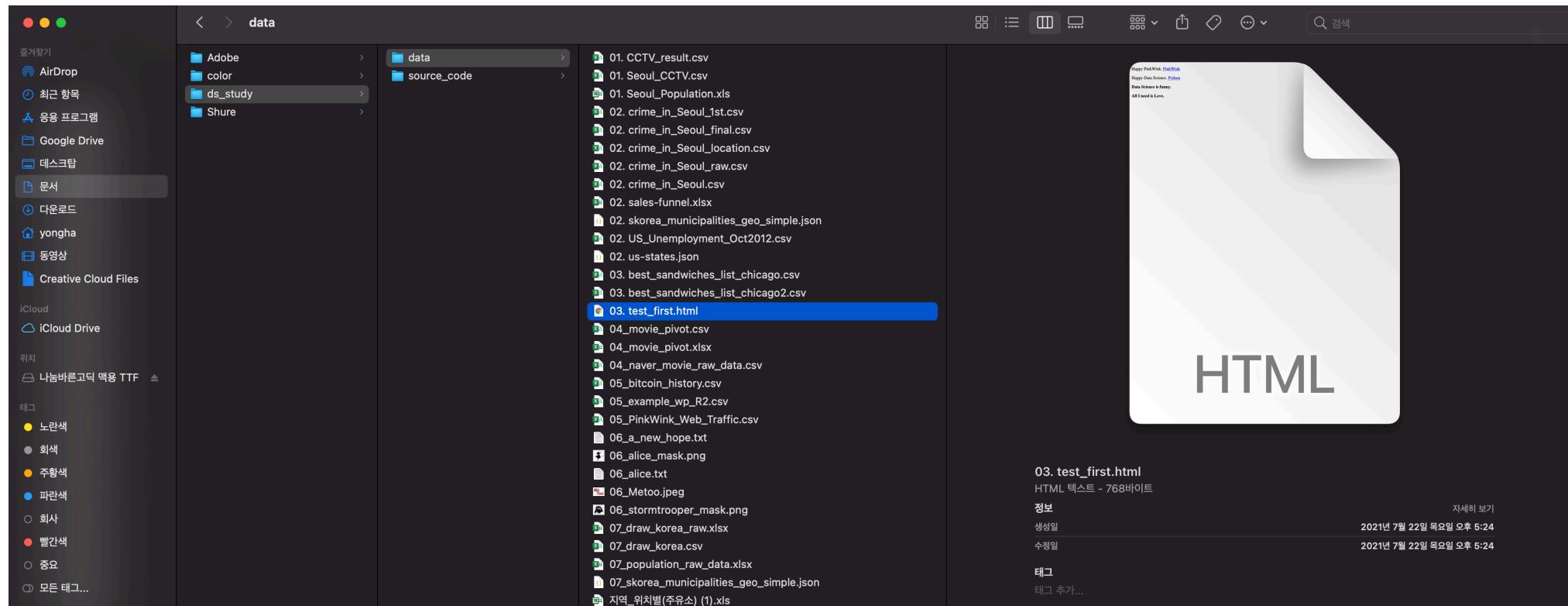


Beautiful Soup for web data

웹에서 데이터를 얻어오기

전에…

Beautiful Soup 사용법 익히기



- 03.test_first.html 파일을 실행

← → ⌂ ⌂ ⓘ 파일 | /Users/yongha/Documents/ds_study/data/03.%20test_first.html

앱 YouTube 퀴즈 교육 취업 업무 퀴즈/과제 제작

Happy PinkWink. [PinkWink](#)

Happy Data Science. [Python](#)

Data Science is funny.

All I need is Love.

- 이렇게 웹브라우저에 글자가 나타난다
- 이렇게 어떻게 나타나지??

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

HTML 언어가 브라우저를 통해
우리에게 이쁜 화면을 제공한다

휴~~~ 이젠 HTML도 배워야 하나요???

넵~~ 그러나 딱 필요한 만큼만 이해하고...

나머진 뭐 그냥 부딪히죠...

HTML 태그는
웹 페이지를
표현

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

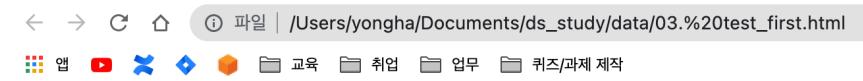
HEAD 태그는 눈에 보이진 않지만
문서에 필요한 헤더 정보를 보관

BODY 태그에는 눈에 보이는 정보를 보관

test_first.html 내용

zero-base /

```
1   <!doctype html>
2   <html>
3   |   <head>
4   |   |   <title>Very Simple HTML Code by PinkWink</title>
5   |   </head>
6   <body>
7   |   <div>
8   |   |   <p class="inner-text first-item" id="first">
9   |   |   |   Happy PinkWink.
10  |   |   |   <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
11  |   |   </p>
12  |   <p class="inner-text second-item">
13  |   |   Happy Data Science.
14  |   |   <a href="https://www.python.org" id="py-link">Python</a>
15  |   |   </p>
16  |   </div>
17  <p class="outer-text first-item" id="second">
18  |   <b>
19  |   |   Data Science is funny.
20  |   </b>
21  </p>
22  <p class="outer-text">
23  |   <b>
24  |   |   All I need is Love.
25  |   </b>
26  </p>
27  </body>
28  </html>
```



```
← → ⌂ ⌄ 파일 | /Users/yongha/Documents/ds_study/data/03.%20test_first.html
 앱  YouTube  X  Diamond  교육  취업  업무  퀴즈/과제 제작
```

Happy PinkWink. [PinkWink](#)

Happy Data Science. [Python](#)

Data Science is funny.

All I need is Love.

```
1  <!doctype html>
2  <html>
3  |   <head>
4  |   |       <title>Very Simple HTML Code by PinkWink</title>
5  |   </head>
6  |   <body>
7  |   |       <div>
8  |   |       |           <p class="inner-text first-item" id="first">
9  |   |       |               Happy PinkWink.
10 |   |       |               <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
11 |   |       |           </p>
12 |   |       <p class="inner-text second-item">
13 |   |       |           Happy Data Science.
14 |   |       |           <a href="https://www.python.org" id="py-link">Python</a>
15 |   |       </p>
16 |   |       </div>
17 |   <p class="outer-text first-item" id="second">
18 |   |       <b>
19 |   |       |           Data Science is funny.
20 |   |       </b>
21 |   |       </p>
22 |   <p class="outer-text">
23 |   |       <b>
24 |   |       |           All I need is Love.
25 |   |       </b>
26 |   |       </p>
27 |   </body>
28 </html>
```

← → ⌂ ⌂ 🔒 crummy.com/software/BeautifulSoup/bs4/doc/

앱 YouTube 드롭박스 교육 취업 업무 퀴즈/과제 제작

[Beautiful Soup 4.9.0 documentation](#) » [Beautiful Soup Documentation](#)

Table of Contents

- [Beautiful Soup Documentation](#)
 - [Getting help](#)
- [Quick Start](#)
- [Installing Beautiful Soup](#)
 - [Problems after installation](#)
 - [Installing a parser](#)
- [Making the soup](#)
- [Kinds of objects](#)
 - [**Tag**](#)
 - [Name](#)
 - [Attributes](#)
 - [Multi-valued attributes](#)
 - [**NavigableString**](#)
 - [**BeautifulSoup**](#)
 - [Comments and other special strings](#)
- [Navigating the tree](#)

Beautiful Soup Documentation

[Beautiful Soup](#) is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers Beautiful Soup version 4.9.3. The examples in this documentation should work the same way in Python 2.7 and Python 3.8.

You might be looking for the documentation for [Beautiful Soup 3](#). If so, you should know that Beautiful Soup 3 is no longer being developed and that support for it will be dropped on or after December 31, 2020. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see [Porting code to BS4](#).



출처: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Beautiful Soup Basic

```
In [1]: from bs4 import BeautifulSoup
```

```
In [2]: page = open("../data/03. test_first.html", "r").read()
soup = BeautifulSoup(page, "html.parser")
print(soup.prettify())
```

```
<!DOCTYPE doctype html>
<html>
  <head>
    <title>
      Very Simple HTML Code by PinkWink
    </title>
```

- 파일로 저장된 html 파일을 읽을 때
- open : 파일명과 함께 읽기(r) / 쓰기(w) 속성을 지정
- html.parser : Beautiful Soup의 html을 읽는 엔진 중 하나(xml도 많이 사용)
- prettify() : html 출력을 이쁘게 만들어 주는 기능

Beautiful Soup 기초

zero-base /

```
1  <!doctype html>
2  <html>
3  |  <head>
4  |  |  <title>Very Simple HTML Code by PinkWink</title>
5  |  </head>
6  <body>
7  |  <div>
8  |  |  <p class="inner-text first-item" id="first">
9  |  |  |  Happy PinkWink.
10 |  |  |  <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
11 |  |  </p>
12 <div>
13 |  <p class="inner-text second-item">
14 |  |  Happy Data Science.
15 |  |  <a href="https://www.python.org" id="py-link">Python</a>
16 |  </p>
17 <div>
18 |  <p class="outer-text first-item" id="second">
19 |  |  <b>
20 |  |  |  Data Science is funny.
21 |  |  </b>
22 |  </p>
23 <div>
24 |  <p class="outer-text">
25 |  |  <b>
26 |  |  |  All I need is Love.
27 |  |  </b>
28 |  </p>
29 </body>
30 </html>
```

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

Beautiful Soup 기초

zero-base /

```
In [3]: soup.body
```

```
Out[3]: <body>
    <div>
        <p class="inner-text first-item" id="first">
            Happy PinkWink.
            <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
        </p>
        <p class="inner-text second-item">
            Happy Data Science.
            <a href="https://www.python.org" id="py-link">Python</a>
        </p>
    </div>
    <p class="outer-text first-item" id="second">
        <b>
            Data Science is funny.
        </b>
    </p>
    <p class="outer-text">
        <b>
            All I need is Love.
        </b>
    </p>
</body>
```

```
In [4]: soup.find("p")
```

```
Out[4]: <p class="inner-text first-item" id="first">
          Happy PinkWink.
          <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
        </p>
```

```
1  <!doctype html>
2  <html>
3  |  <head>
4  |  |  <title>Very Simple HTML Code by PinkWink</title>
5  |  </head>
6  <body>
7  |  <div>
8  |  |  <p class="inner-text first-item" id="first">
9  |  |  |  Happy PinkWink.
10 |  |  |  <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
11 |  |  </p>
```

```
In [5]: soup.find_all("p")
```

```
Out[5]: [<p class="inner-text first-item" id="first">
           Happy PinkWink.
           <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
         </p>,
         <p class="inner-text second-item">
           Happy Data Science.
           <a href="https://www.python.org" id="py-link">Python</a>
         </p>,
         <p class="outer-text first-item" id="second">
           <b>
             Data Science is funny.
           </b>
         </p>,
         <p class="outer-text">
           <b>
             All I need is Love.
           </b>
         </p>]
```

- `find_all()`은 지정된 태그를 모두 찾아준다

Beautiful Soup 기초

```
6  <body>
7    <div>
8      <p class="inner-text first-item" id="first">
9        Happy PinkWink.
10       <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
11     </p>
12    <p class="inner-text second-item">
13      Happy Data Science.
14      <a href="https://www.python.org" id="py-link">Python</a>
15    </p>
16  </div>
17  <p class="outer-text first-item" id="second">
18    <b>
19      Data Science is funny.
20    </b>
21  </p>
22  <p class="outer-text">
23    <b>
24      All I need is Love.
25    </b>
26  </p>
```

물론 p 태그 안에
특정 클래스만
찾을 수 있다

```
1   <!doctype html>
2   <html>
3   |   <head>
4   |       <title>Very Simple HTML Code by PinkWink</title>
5   |   </head>
6   |   <body>
7   |       <div>
8   |           <p class="inner-text first-item" id="first">
9   |               Happy PinkWink.
10  |               <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
11  |           </p>
12  |           <p class="inner-text second-item">
13  |               Happy Data Science.
14  |               <a href="https://www.python.org" id="py-link">Python</a>
15  |           </p>
16  |       </div>
17  |       <p class="outer-text first-item" id="second">
18  |           <b>
19  |               Data Science is funny.
20  |           </b>
21  |       </p>
22  |       <p class="outer-text">
23  |           <b>
24  |               All I need is Love.
25  |           </b>
26  |       </p>
27   |   </body>
28   </html>
```

```
In [8]: soup.find_all(class_ = "outer-text")  
  
Out[8]: [<p class="outer-text first-item" id="second">  
    <b>  
        Data Science is funny.  
    </b>  
    </p>,  
    <p class="outer-text">  
        <b>  
            All I need is Love.  
        </b>  
    </p>]
```

```
In [9]: soup.find_all(id="first")
```

```
Out[9]: [<p class="inner-text first-item" id="first">
           Happy PinkWink.
           <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
         </p>]
```

- 그러나… 그냥 이렇게 사용하는 경우가 더 많다.

```
In [10]: soup.find(id="first")
```

```
Out[10]: <p class="inner-text first-item" id="first">
            Happy PinkWink.
            <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
        </p>
```

- HTML 내에서 속성 id는 딱 한 번만 나타난다
- 그래서 find_all() 함수는 의미가 없다
- 단, 검색결과를 list로 받고 싶다면 id라도 find_all() 함수를 사용한다

Beautiful Soup 기초

zero-base /

```
In [11]: for each_tag in soup.find_all("p"):  
    print(" ----- ")  
    print(each_tag.get_text())
```

Happy PinkWink.
PinkWink

Happy Data Science.
Python

Data Science is funny.

All I need is Love.

```
In [12]: links = soup.find_all("a")
links
```

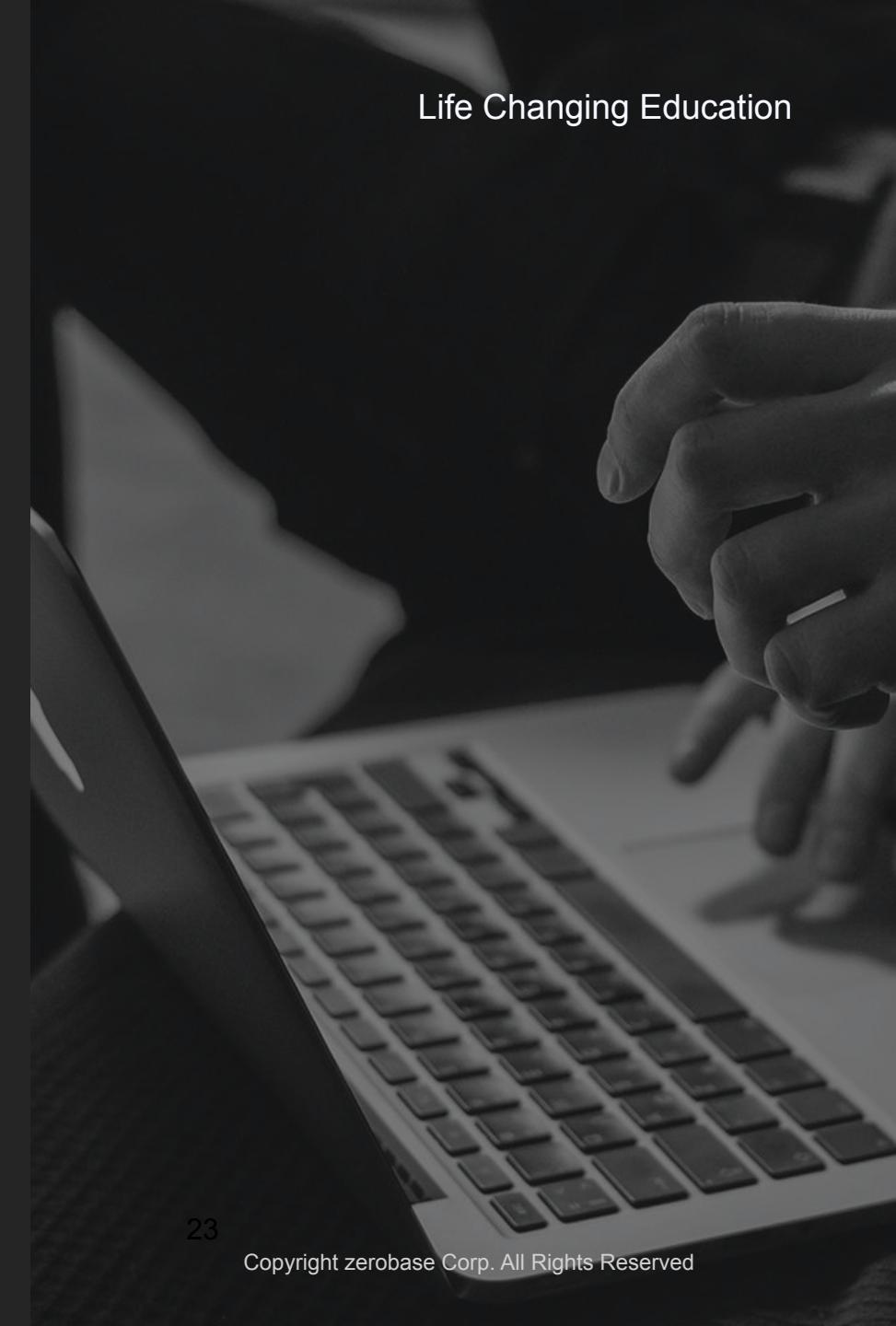
```
Out[12]: [<a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>,
           <a href="https://www.python.org" id="py-link">Python</a>]
```

```
In [13]: for each in links:
            href = each["href"]
            text = each.string
            print(text + " -> " + href)
```

```
PinkWink -> http://www.pinkwink.kr
Python -> https://www.python.org
```

- 외부로 연결되는 링크의 주소를 알아내는 방법

크롬 개발자 도구 - 환율정보 가져오기



간단한 예제로

Beautiful Soup과 친해지기

feat. Chrome 개발자 도구

The screenshot shows the Naver Finance homepage. At the top, there is a navigation bar with a green 'N' logo, the text '네이버 금융', and icons for keyboard, dropdown, and search. Below the navigation bar is a horizontal menu with links: 통합 (Integration), VIEW, 이미지 (Image), 지식iN, 동영상 (Video), 쇼핑 (Shopping), 뉴스 (News), 어학사전 (Dictionary), 지도 (Map), and 책 (Books). A three-dot ellipsis is also present. In the center, there is a box containing the URL 'finance.naver.com', the title '네이버 금융', and links for '시장지표' (Market Index), '뉴스' (News), '펀드' (Fund), '국내증시' (Domestic Stock Market), '리서치' (Research), and '해외증시' (Overseas Stock Market). Below this box, the text '국내 해외 증시 지수, 시장지표, 펀드, 뉴스, 증권사 리서치 등 제공' (Provides domestic and overseas stock market indices, market indices, funds, news, securities company research, etc.) is displayed. At the bottom of the page, there is a section titled '증권정보' (Securities Information) with dropdown menus for '주가지수' (Stock Price Index) and '주요증시' (Major Stock Exchanges).

Beautiful Soup 예제 1 - 네이버 금융

zero-base /

[금융 홈](#) [국내증시](#) [해외증시](#) [시장지표](#) [펀드](#) [리서치](#) [뉴스](#) [MY](#)


눈물영양제 닥터바이 아이즈 2.0

'눈물자국' 냄새뿐 아니라 세균이 있어요!

[더 알아보기 >](#)

AD



- 시장지표 탭으로 이동
- <https://finance.naver.com/marketindex/>

Beautiful Soup 예제 1 - 네이버 금융

zero-base /

금융 흠 국내증시 해외증시 시장지표 펀드 리서치 뉴스 MY



눈물영양제 닥터바이 아이즈 2.0

'눈물자국' 냄새뿐 아니라 세균이 있어요!

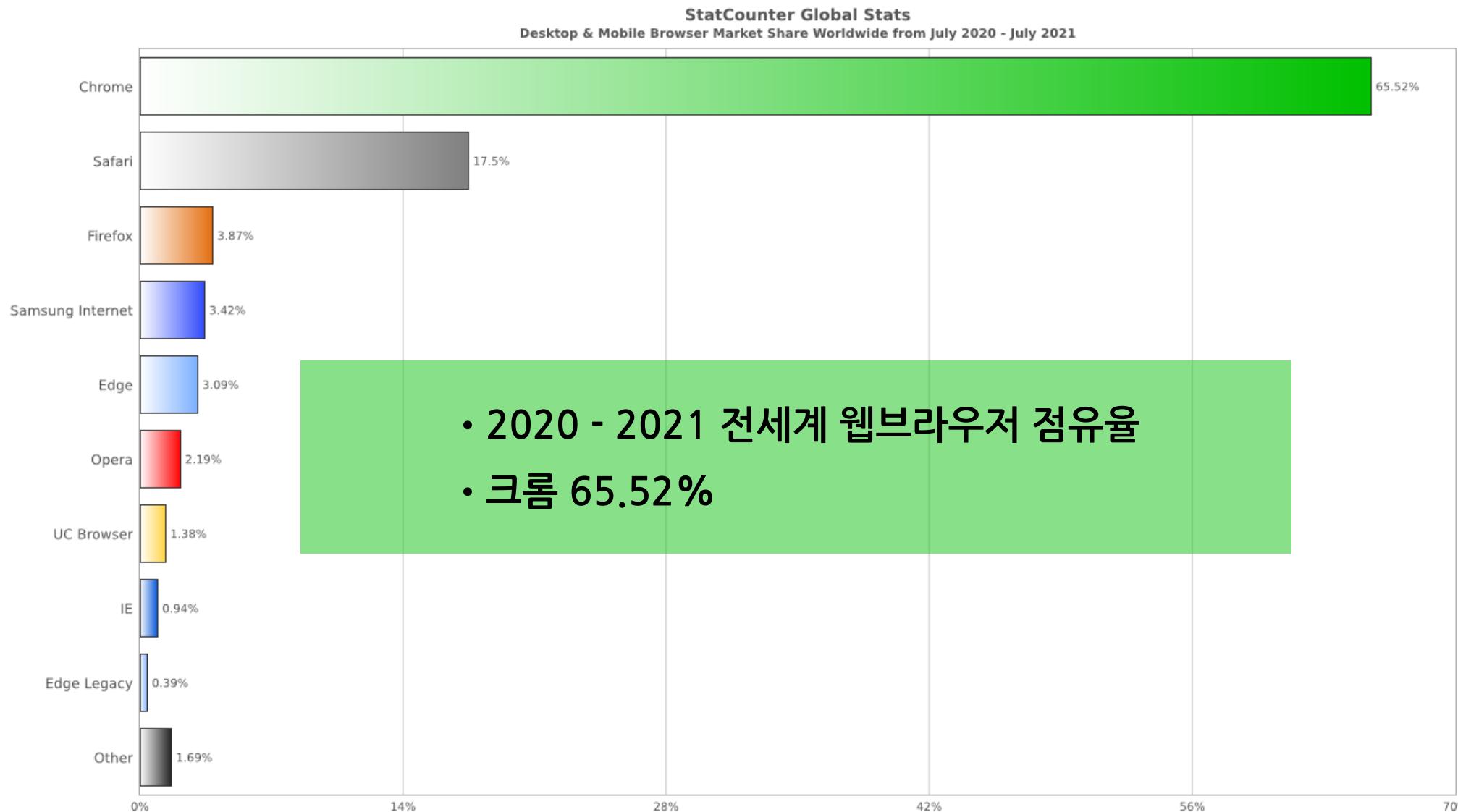
더 알아보기 >

AD



- USD 환율 체크를 파이썬으로 하고 싶은데..
- HTML을 잘 모른다
- 그럴 때 사용하는 것이 크롬 개발자 도구

Beautiful Soup 예제 1 - 네이버 금융



출처: <https://gs.statcounter.com/browser-market-share/desktop-mobile/worldwide/#monthly-202007-202107-bar>

Beautiful Soup 예제 1 - 네이버 금융

zero-base /



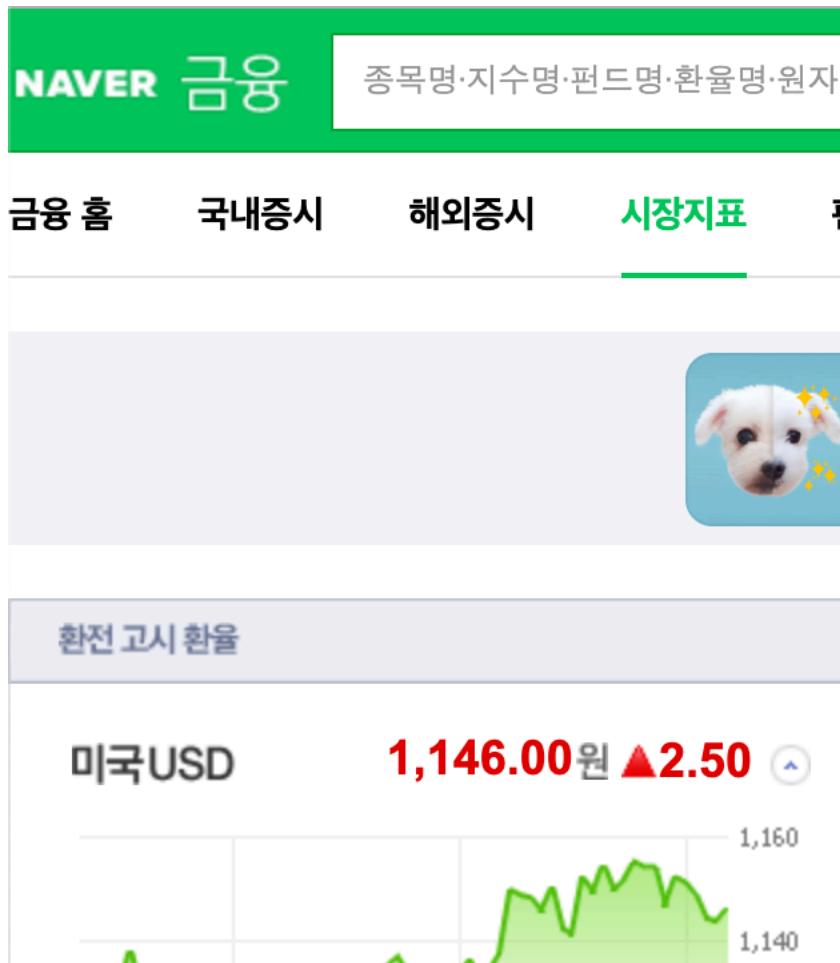
The screenshot shows the NAVER Finance homepage. On the left, there's a sidebar with currency conversion rates (e.g., USD: 1,146.00, JPY: 1,039.31, EUR: 1,347.87, CNY: 176.96) and a chart for CNY showing a recent dip. In the center, there's a chart for USD/EUR showing a slight increase. On the right, there's another chart for GBP/USD showing a slight increase. A context menu is open on the right side of the screen, specifically under the '도구 더보기' (More Tools) section. The menu items include '페이지를 다른 이름으로 저장...', '바로가기 만들기...', '창 이름 지정...', '인터넷 사용 기록 삭제...', '확장 프로그램', '작업 관리자', and '개발자 도구'. The '개발자 도구' option is highlighted with a red arrow. The top right corner of the slide has a red checkmark icon.

- 크롬 개발자 도구 활용
- 크롬 설정 - 도구 더보기 - 개발자 도구(화면 오른쪽부터 선택)

Beautiful Soup 예제 1 - 네이버 금융

zero-base /

이 아이콘 선택

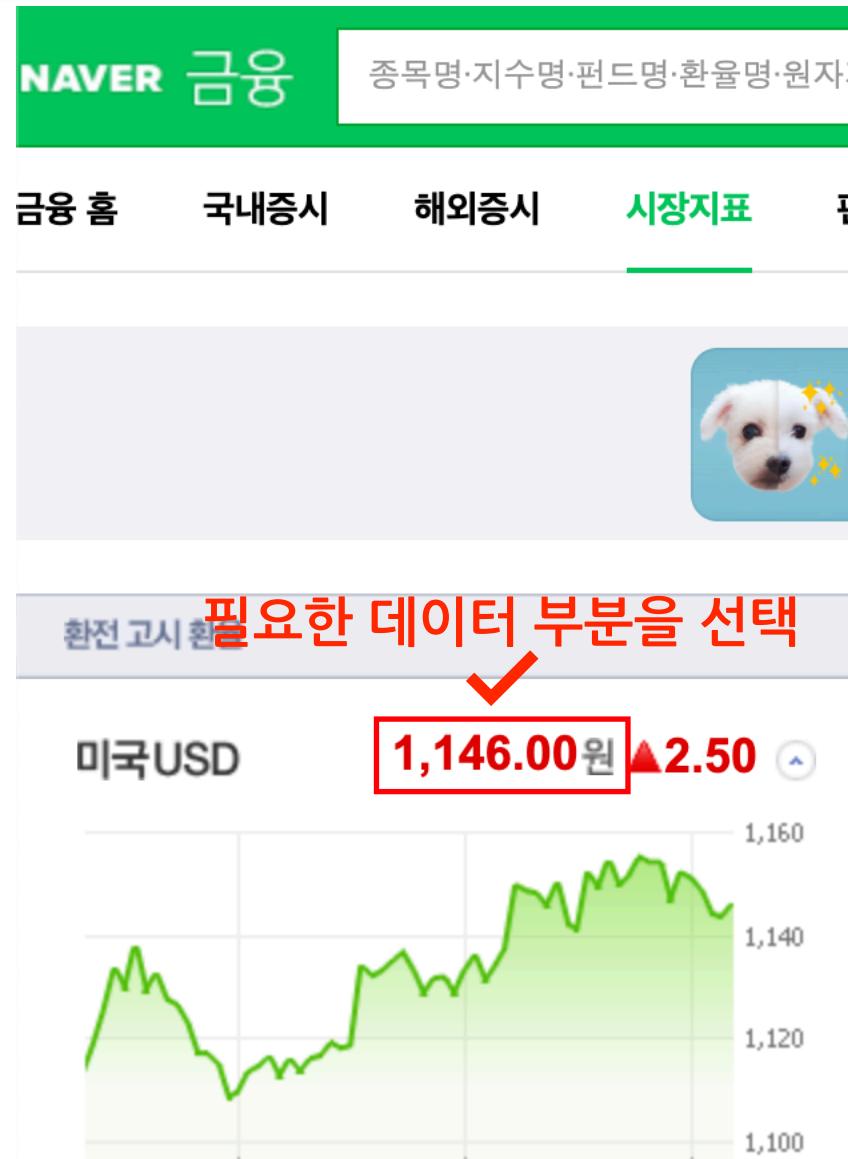


```
Elements Console Sources Network Performance

<html lang="ko">
  <head>...</head>
  ... <body onload="getGNB();"> == $0
    <script type="text/javascript"> document.domain = 'naver.com' </script>
    <script type="text/javascript">...</script>
    <div id="u_skip">...</div>
    <div id="header">...</div>
    <div id="wrap">...</div>
    <div class="selectbox-layer ly_calculator" style="display: none;">...</div>
    <div class="selectbox-layer ly_calculator" style="display: none;">...</div>
  </body>
</html>
```

Beautiful Soup 예제 1 - 네이버 금융

zero-base /



```

Elements Console Sources Network Performance
<html lang="ko">
  <head>...</head>
  <body onload="getGNB();">
    <script type="text/javascript"> document.domain = 'naver.com' </script>
    <script type="text/javascript" src="https://ssl.pstatic.net...</script>
    <div id="u_skip">...</div>
    <div id="header">...</div>
    <div id="wrap">
      <div class="banner_smart">...</div>
      <script type="text/javascript" src="https://ssl.pstatic.net...</script>
      <script type="text/javascript" src="https://ssl.pstatic.net...</script>
      <div id="container" style="padding-bottom:0px;">
        <div class="market_include">
          <div class="market_data">
            <div class="market1">
              <div class="title">...</div>
              <!-- data -->
              <div class="data">
                <ul class="data_lst" id="exchangeList">
                  <li class="on">
                    <a href="/marketindex/exchangeDetail.nhn?marketCd=USD...</a>
                    <h3 class="h_lst">...</h3>
                    <div class="head_info point_up">
                      <span class="value">1,146.00</span> == $0
                      <span class="txt_krw">...</span>
                      <span class="change">2.50</span>
                      <span class="blind">상승</span>
                    </div>
                  </li>
                </ul>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>

```



```
▶ <div class="title">...</div>
  <!-- data -->
  ▼ <div class="data">
    ▼ <ul class="data_lst" id="exchangeList">
      ▼ <li class="on">
        ▼ <a href="/marketindex/exchangeDetail.nhn?marketCd=USD&...>
          ▶ <h3 class="h_lst">...</h3>
          ▼ <div class="head_info point_up">
            ...
            <span class="value">1,146.00</span> == $0
            <span class="txt_krw">...</span>
            <span class="change">2.50</span>
            <span class="blind">상승</span>
```

- 내가 원하는 HTML 태그가 위치한 곳을 찾아 갈 수 있다
- 여기서 기억해야할 것은 **** 이다

Beautiful Soup 예제 1 - 네이버 금융

zero-base /

The screenshot shows the Naver Finance homepage (finance.naver.com/marketindex/). The top navigation bar includes links for App, YouTube, News, Education, Job, Business, and Quiz/Assignment Creation. The main menu features tabs for 금융 홈 (Finance Home), 국내증시 (Domestic Stock Market), 해외증시 (Overseas Stock Market), 시장지표 (Market Indicators) (which is underlined in green), 펀드 (Funds), 리서치 (Research), 뉴스 (News), and M. The market indicators section displays the exchange rate between the US dollar and the Korean won, showing 1,146.00원 (1,146.00 won) with a red upward arrow indicating a price increase of 2.50 won. A promotional banner for '눈물영양제 닥터바이' (Doctor Eye) eye drops is visible on the right.

- URL 주소 복사
- <https://finance.naver.com/marketindex/>

```
In [4]: from urllib.request import urlopen
```

```
In [5]: # url = "http://info.finance.naver.com/marketindex/" ## url error
url = "https://finance.naver.com/marketindex/"
page = urlopen(url)

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

```
print(soup.prettify())
<a class= head_jpy href= /marketindex/exchangedetail.nhn?marketin
1.jspyt', '', '', event);">
    <h3 class="h_lst">
        <span class="blind">
            일본 JPY(100엔)
```

- 웹주소(URL)에 접근할 때는 urllib의 request 모듈이 필요하다

Beautiful Soup 예제 1 - 네이버 금융

zero-base /



```
▼<div class="market_include">
  ▼<div class="market_data">
    ▼<div class="market1">
      ▶<div class="title">...</div>
      <!-- data -->
      ▼<div class="data">
        ▼<ul class="data_lst" id="exchangeList">
          ▼<li class="on">
            ▼<a href="/marketindex/exchangeDetail.nhn?market...">
              ▶<h3 class="h_lst">...</h3>
              ▼<div class="head_info_point_up">
                <span class="value">1,146.00</span> == $0
                <span class="txt_krw">...</span>
                <span class="change">2.50</span>
                <span class="blind">상승</span>
              </div>
            &lt;&gt;
            </a>
```

```
In [6]: soup.find_all("span", "value")[0].string
```

```
Out[6]: '1,146.00'
```

매우 당연한 이야기지만,
HTML을 이용한 웹 페이지의 데이터를 얻어 올려면,
당연히 HTML을 공부해야 합니다.

그것도 요즘 웹 페이지들의
다양하고 깊은 기능을 학습해야 맞는 이야기입니다

위키백과 데이터 가져오기



한 번 더 친해지기 예제…

여명의 눈동자



김종학 연출, 송지나 극본의 드라마의 시작
이후, 모래시계도 연출된다.
드라마에서는

일제시대, 위안부, 강제 동원 등이 그려지고
남과북의 상황이 그려지고
제주 4.3 항쟁의 슬픈 역사도 그려진다.

Beautiful Soup 예제 2 - 여명의 눈동자

zero-base /



여명의 눈동자

위기백과, 우리 모두의 백과사전.

《여명의 눈동자》는 1991년 10월 7일부터 1992년 2월 6일까지 방영된 문화방송 수목 미니시리즈이다.

목차 [술기기]	
1 개요	
2 등장 인물	2.1 주요 인물
	2.2 어육의 주변 인물
	2.3 허립의 주변 인물
	2.4 그 외
3 제작진	
4 시청률	5 본방송 편성 변경
	6 재방송 결방 사유 및 편성 변경
	7 수상 경력
	8 OST
	9 참고 사항
	10 고증 오류
11 주제	11.1 주제
12 각주	12.1 각주
13 외부 링크	

개요 [편집]

1989년 10월부터 청사 30주년 기념 특집극으로 기획에 들어가 그 다음해인 1990년 5월 3일부터 MBC 정동 스튜디오에서 실내 촬영을 하는 것으로 시작되었다. 당시 총 10권 분량의 소설 《여명의 눈동자》를 원작으로 제작됐다.

1990년 6월부터 해외 촬영을 시작하여 필리핀에서 최대치가 소속된 일본군 제15사단의 부로미(부리) 행군 장면과 허립이 근무하는 군 약전 병원의 모습 등을 촬영하였으나 현지 여건상 일본인과 비슷한 사람들을 구하기 쉽지 않아 현지 필리핀 원주민을 엑스트리로 대거 동원하여 극의 여러 장면들을 촬영한 후 성우들의 후사음을으로 장면을 완성하였다.^[1]

특히 1991년 2월, 당시 대한민국과 국교를 아직 맺지 않은 중화인민공화국으로 들어가 하얼빈 교외와 상하이, 난징, 부지우, 구이린에서 731 부대, 팔로군 연안 본부 장면 등을 촬영함으로써 극의 완성도를 높였다. 또한 제주 A·3 항쟁 장면을 찍기 위해 1991년 5월에는 제주도에서 촬영하였다.^[1]

방송 시작 전까지 1년 5개월의 긴 사전 제작 기간을 가졌으나 정작 방송 시작 후에도 촬영을 완전히 끌어내리지 못하고 마지막 회인 제36부 방송일인 1992년 2월 6일 하루 전날인 2월 5일에서야 마지막 장면인 최대치와 윤여숙의 최후 장면을 촬영하고 방송 시작 10분 전에서야 편집 작업이 마무리되어 제작·촬영의 2년 4개월 간의 일정을 끝맺었다.^[2]

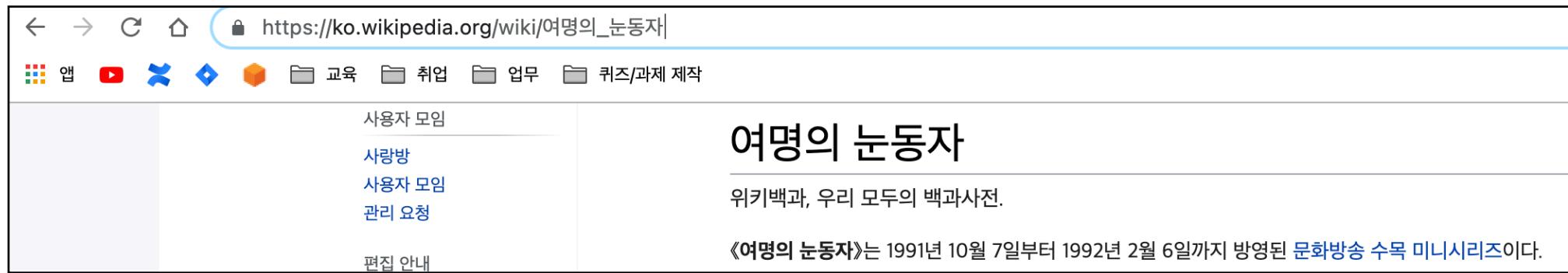
여명의 눈동자 15	
장르	드라마
방송 국가	대한민국
방송 채널	MBC TV
방송 기간	1991년 10월 7일 ~ 1992년 2월 6일 (본방송) 1992년 11월 28일 ~ 1993년 2월 27일 (재방송)
방송 시간	매주 수요일, 목요일 밤 9시 50분 (본방송) 토요일, 일요일 오후 5시 40분 (재방송)
방송 분량	50회
방송 횟수	36부작
주기	제작 MBC ON
원작	김성중의 소설 《여명의 눈동자》
기획	최종수
제작자	이강훈
연출	김종학
출연자	이창순, 이랑호
조연출	이창순, 이랑호
각본	송지나
음악	체사비, 박상원, 최자성 외
에는 곡	Main title Love Theme - 최경식
닫는 곡	End Title Love Theme - 최경식
음성	2체널 스테레오 사운드(아날로그)

The screenshot shows a Google search results page with a dark theme. The search query '여명의 눈동자' is entered in the search bar. Below the search bar, there are several navigation links: All (highlighted), Videos, Images, News, Maps, More, and Tools. The search results indicate 'About 378,000 results (0.42 seconds)'. The top result is a link to the Wikipedia page for '여명의 눈동자', which is titled '여명의 눈동자 - 위키백과, 우리 모두의 백과사전'. Below the title, there is a summary of the show's cast and broadcast information. The summary includes: '그 외[편집] · 박근형 : 최두일(스즈키) 역 · 이정길 : 김기문 역 · 장항선 : 오오에 오장 역 · 박인환 : 구보다 일병 역 · 임현식 : 황성철 역 · 흥승옥 : 성철 처 역 · 김홍기 : ...', '원작: 김성종의: 소설; 《여명의 눈동자》', '방송 시간: 매주 수요일, 목요일 밤 9시 50분 ...', '방송 기간: 1991년 10월 7일 ~ 1992년 2월 6일 ...', '방송 횟수: 36부작', and '개요 · 등장 인물 · 제작진 · 재방송 결방 사유 및 편성 변경'.

- 여명의 눈동자 위키백과 페이지로 이동
- https://ko.wikipedia.org/wiki/여명의_눈동자

Beautiful Soup 예제 2 - 여명의 눈동자

zero-base /



- 웹페이지(URL)를 복사해야 한다
- Ctrl(Cmd) + V 를 눌러서 복사한 뒤에..

```
In [ ]: https://ko.wikipedia.org/wiki/%EC%97%AC%EB%AA%85%EC%9D%98_%EB%88%88%EB%8F%99%EC%9E%90
```

- 메모장이나 Jupyter Notebook 셀에 붙여 넣어 보자
- 그러면 뭔가 이상하게 바뀌어서 나타난다… 뭘까???
- 웹주소는 UTF-8로 인코딩 되어야 한다

Beautiful Soup 예제 2 - 여명의 눈동자

In [7]:

```
import urllib
from urllib.request import Request

html = "https://ko.wikipedia.org/wiki/{search_words}"
req = Request(html.format(search_words=urllib.parse.quote("여명의_눈동자")))

response = urlopen(req)

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

등장 인물 [편집]**주요 인물** [편집]

- 채시라 : 윤여옥 역 (아역: 김민정)
- 박상원 : 장하림(하리모토 나츠오) 역 (아역: 김태진)
- 최재성 : 최대치(사카이) 역 (아역: 장덕수)

여옥의 주변 인물 [편집]

- 최불암 : 윤흥철 역 - 윤여옥의 아버지
- 한차돌 : 최대운 역 - 최대치와 윤여옥의 아들
- 오연수 : 봉순 역

하림의 주변 인물 [편집]

- 김소원 : 장하림의 어머니 역

```

▶ <p>...</p>
▶ <p>...</p>
▶ <p>...</p>
▶ <h2>...</h2>
▶ <h3>...</h3>
▼ <ul>
  ▼ <li>
    ::marker
    ...
    <a href="/wiki/%EC%B1%84%EC%8B%9C%EB%9D%BC" title="김
      (수필가)">채시라</a> == $0
    "&nbsp;: 윤여옥 역 (아역: "
    <a href="/wiki/%EA%B9%80%EB%AF%BC%EC%A0%95_(1982%EB%8
      4)" title="김민정 (1982년)">김민정</a>
    ")"
  
```

Beautiful Soup 예제 2 - 여명의 눈동자

zero-base /

```

n = 0
for each in soup.find_all('ul'):
    print("=>" + str(n) + "====")
    print(each)
    n += 1

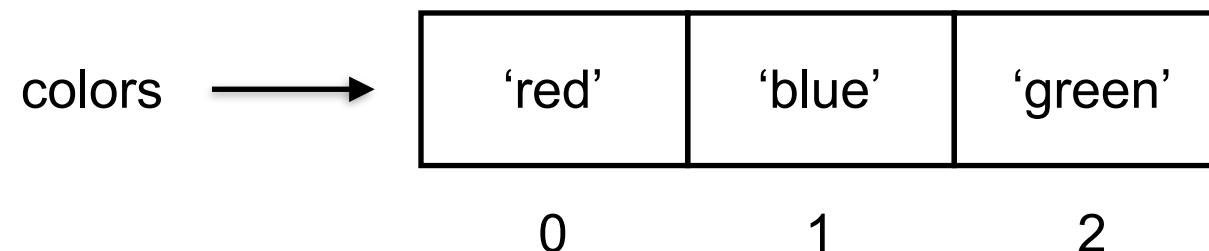
<li class="toclevel-2 tocsection-6"><a href="#그_외"><span class="tocnumber">2.4</span> <span c>
an></a></li>
</ul>
=>15=====
<ul><li><a href="/wiki/%EC%B1%84%EC%8B%9C%EB%9D%BC" title="채시라">채시라</a> : 윤여옥 역 (아역: <a h
B%AF%BC%EC%A0%95_(1982%EB%85%84)" title="김민정 (1982년)">김민정</a>)</li>
<li><a href="/wiki/%EB%B0%95%EC%83%81%EC%9B%90" title="박상원">박상원</a> : 장하림(하리모토 나츠오) 역 (
A%B9%80%ED%83%9C%EC%A7%84_(%EC%88%98%ED%95%84%EA%B0%80)" title="김태진 (수필가)">김태진</a>)</li>
<li><a href="/wiki/%EC%B5%9C%EC%9E%AC%EC%84%B1_(%EB%B0%B0%EC%9A%B0)" title="최재성 (배우)">최재성</
역: <a href="/wiki/%EC%9E%A5%EB%8D%95%EC%88%98_(%EB%B0%B0%EC%9A%B0)" title="장덕수 (배우)">장덕수</a>
=>16=====
<ul><li><a href="/wiki/%EC%B5%9C%EB%B6%88%EC%95%94" title="최불암">최불암</a> : 윤흥철 역 - 윤여옥의 아

```

```
In [19]: soup.find_all("ul")[2]
```

```
Out[19]: <ul class="vector-menu-content-list"><li id="n-projectchat"><a href="/2%AC%EB%9E%91%EB%B0%A9">사랑방</a></li><li id="n-portal"><a href="/wiki.C%EC%9A%A9%EC%9E%90_%EB%AA%A8%EC%9E%84" title="위키백과 참여자를 위한 토론/대t"><a href="/wiki/%EC%9C%84%ED%82%A4%EB%B0%B1%EA%B3%BC:%EC%9A%94%EC%B2
```

Python List 데이터 형

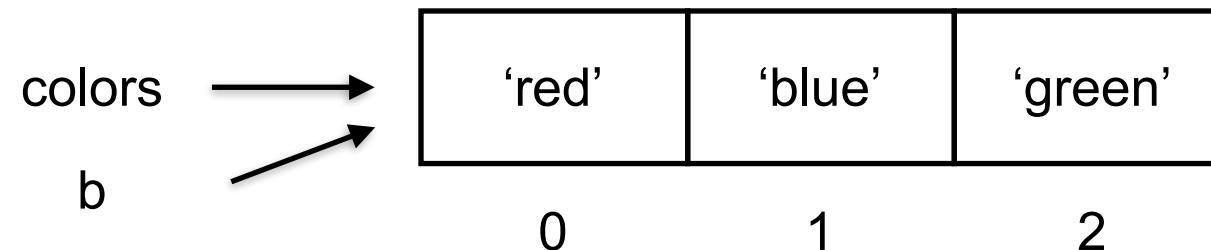


- List 형은 대괄호로 생성한다

```
In [1]: colors = ['red', 'blue', 'green']
```

```
print(colors[0])
print(colors[2])
print(len(colors))
```

```
red
green
3
```



```
In [2]: b = colors  
b
```

```
Out[2]: ['red', 'blue', 'green']
```

```
In [3]: b[1] = 'black'  
b
```

```
Out[3]: ['red', 'black', 'green']
```

```
In [4]: colors
```

```
Out[4]: ['red', 'black', 'green']
```

- List 형을 반복문(for)에서 사용하는 방법이 다른 언어에 비해 편하다

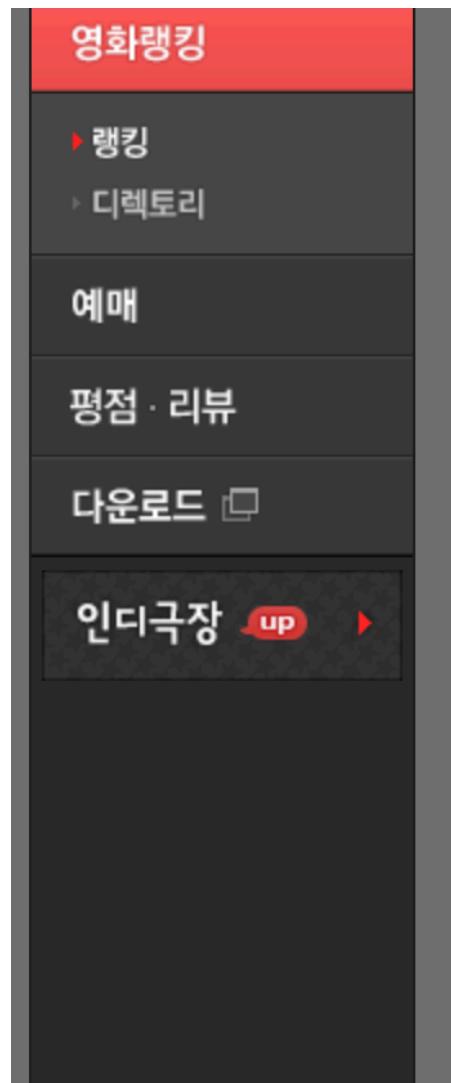
```
In [5]: for color in colors:  
    print(color)
```

```
red  
black  
green
```

- in 명령으로 조건문(if)에 적용하는 것도 역시 다른 명령에 비해 편하다

```
In [6]: if 'black' in colors:  
    print('True')
```

```
True
```



| 영화 랭킹

조회순	평점순 (현재상영영화)	평점순 (모든영화) ▾	2021.08.08
순위	영화명	평점	변동폭
1	그린 북	★★★★★ 9.60	평점주기 - 0
2	가버나움	★★★★★ 9.59	평점주기 - 0
3	디지몬 어드벤쳐 라스트 에볼루션 : 인연	★★★★★ 9.54	평점주기 - 0
4	먼 훗날 우리	★★★★★ 9.53	평점주기 - 0
5	원더	★★★★★ 9.52	평점주기 - 0
6	베일리 어게인	★★★★★ 9.52	평점주기 - 0
7	아일라	★★★★★ 9.51	평점주기 - 0
8	당갈	★★★★★ 9.49	평점주기 - 0
9	포드 V 페라리	★★★★★ 9.48	평점주기 - 0
10	극장판 바이올렛 에버가든	★★★★★ 9.48	평점주기 - 0

출처: <https://movie.naver.com/movie/sdb/rank/rmovie.naver?sel=pnt&date=20210808>

그린북

Green Book, 2018

관람객  **9.55** 기자·평론가  **7.29**

네티즌  **9.59** 내 평점  등록>

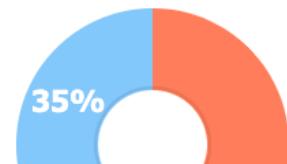
개요 드라마 | 미국 | 130분 | 2019.01.09 개봉

감독 피터 패럴리

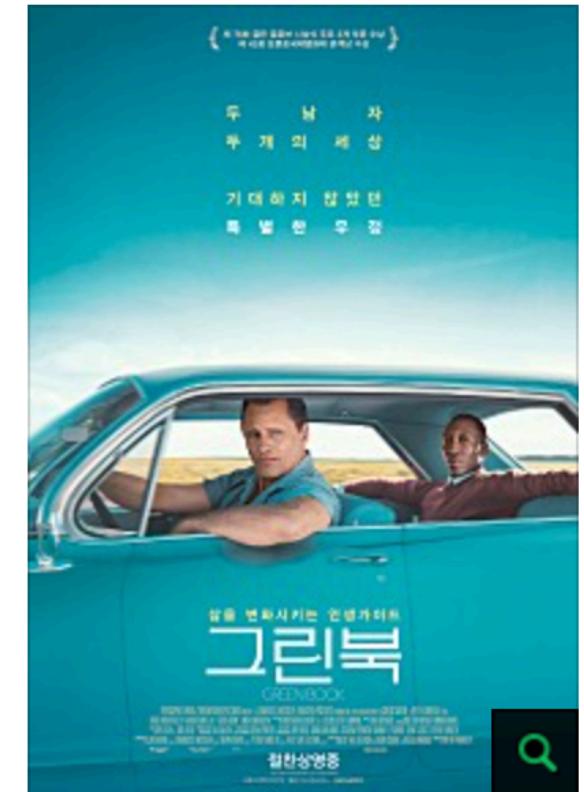
출연 비고 모텐슨(토니 발레롱가), 마허살라 알리(돈 셜리 박사) [더보기](#)▶

등급 [국내] 12세 관람가 [해외] PG-13 

성별·나이별 관람추이



출처: <https://movie.naver.com/movie/bi/mi/basic.naver?code=171539#>



```
In [10]: movies = ['그린 북', '가버나움', '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리']
print(movies)
```

```
[ '그린 북', '가버나움', '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리' ]
```

- **append : list 제일 뒤에 하나 추가**

```
In [11]: movies.append('원더')
print(movies)
```

```
[ '그린 북', '가버나움', '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리', '원더' ]
```

- **pop : 제일 뒤 자료를 지움**

```
In [12]: movies.pop()
print(movies)
```

```
[ '그린 북', '가버나움', '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리' ]
```

- **extend** : 제일 뒤에 다수의 자료를 추가

```
In [13]: movies.extend(['베일리 어겐', '아일라', '당갈'])  
print(movies)
```

```
[ '그린 북', '가버나움', '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리', '베일리 어겐', '아일라', '당갈' ]
```

- **remove** : 같은 이름의 자료를 지움

```
In [14]: movies.remove('가버나움')  
print(movies)
```

```
[ '그린 북', '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리', '베일리 어겐', '아일라', '당갈' ]
```

- 슬라이싱 : [n:m] n 번째 부터 m-1 까지

```
In [16]: print(movies[3:5])
```

```
[ '베일리 어겐', '아일라' ]
```

```
In [18]: favorite_movies = movies[0:3]  
print(favorite_movies)
```

```
[ '그린 북', '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리' ]
```

• insert : 원하는 위치에 자료를 삽입

```
In [21]: favorite_movies.insert(1, 9.60)
print(favorite_movies)
```

```
Out[21]: ['그린 북', 9.6, 0, 0, '디지몬 어드벤처 라스트 에볼루션:인연', '먼 훗날 우리']
```

```
In [29]: favorite_movies.insert(3, 9.54)
print(favorite_movies)
```

```
['그린 북', 9.6, '디지몬 어드벤처 라스트 에볼루션:인연', 9.54, 9.59, '먼 훗날 우리']
```

```
In [31]: favorite_movies.append(9.53)
print(favorite_movies)
```

```
['그린 북', 9.6, '디지몬 어드벤처 라스트 에볼루션:인연', 9.54, '먼 훗날 우리', 9.53]
```

- list 안에 list를 가질 수 있다

```
In [33]: favorite_movies.insert(2, ['비고 모텐슨', '마허샬라 알리', '린다 카델리니'])
print(favorite_movies)

['그린 북', 9.6, ['비고 모텐슨', '마허샬라 알리', '린다 카델리니'], '디지몬 어드벤처 라스트 에볼루션:인연', 9.54, '먼 훗날 우리', 9.53]
```

- **isinstance** : 자료형이 list인지 확인 할 수 있다

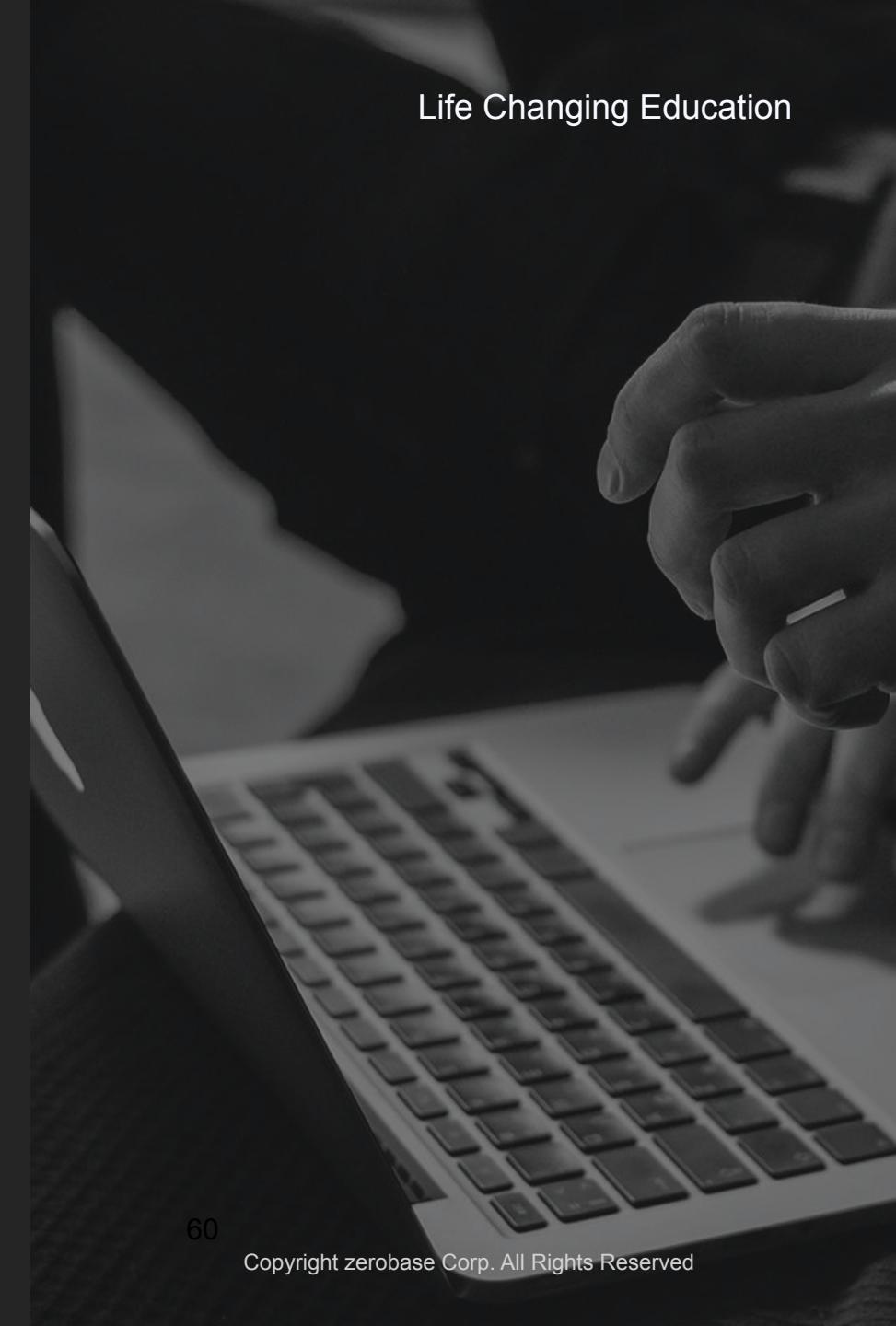
```
In [35]: isinstance(favorite_movies, list)
```

```
Out[35]: True
```

```
In [36]: for each_item in favorite_movies:  
    if isinstance(each_item, list):  
        for nested_item in each_item:  
            print(nested_item)  
    else:  
        print(each_item)
```

그린 북
9.6
비고 모텐슨
마허샬라 알리
린다 카델리니
디지몬 어드벤처 라스트 에볼루션:인연
9.54
먼 훗날 우리
9.53

시카고 맷집 데이터 분석 - 개요



Beautiful

웹 스크래핑의 기초



Soup을 통해 익히는



BLT at Old Oak Tap, the No. 1 sandwich in Chicago.
Photograph: Anna Knott; Food Stylist: Lisa Kuehl

Google chicago magazine the 50 best sandwiches X |  

All Images News Videos Maps More Tools

About 23,900,000 results (0.61 seconds)

Only joy.

- BLT. Old Oak Tap. [Read more](#).
- Fried Bologna. Au Cheval. [Read more](#).
- Woodland Mushroom. Xoco. [Read more](#).
- Roast Beef. Al's Deli. [Read more](#).
- PB&L. Publican Quality Meats. [Read more](#).
- Belgian Chicken Curry Salad. Hendrickx Belgian Bread Crafter. [Read more](#).
- Lobster Roll. Acadia. [Read more](#).
- Smoked Salmon Salad. Birchwood Kitchen. [Read more](#).

[More items...](#) • Oct 9, 2012



<https://www.chicagomag.com> › November-2012 › Best-S...

The 50 Best Sandwiches in Chicago – Chicago Magazine

The screenshot shows a web browser displaying the Chicago Magazine website. The URL in the address bar is [chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/](https://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/). The page features a dark header with the magazine's logo and navigation links for news categories like NEWS & POLITICS, DINING & DRINKING, CITY LIFE, CULTURE, REAL ESTATE, STYLE, and TOP DOCS. The main content is titled "The 50 Best Sandwiches in Chicago" and includes a subtext "Our list of Chicago's 50 best sandwiches, ranked in order of deliciousness". Below the title is a timestamp "OCTOBER 9, 2012, 6:12 PM" and social sharing icons for Facebook, Twitter, Email, and Print. A large image of a sandwich is centered on the page.

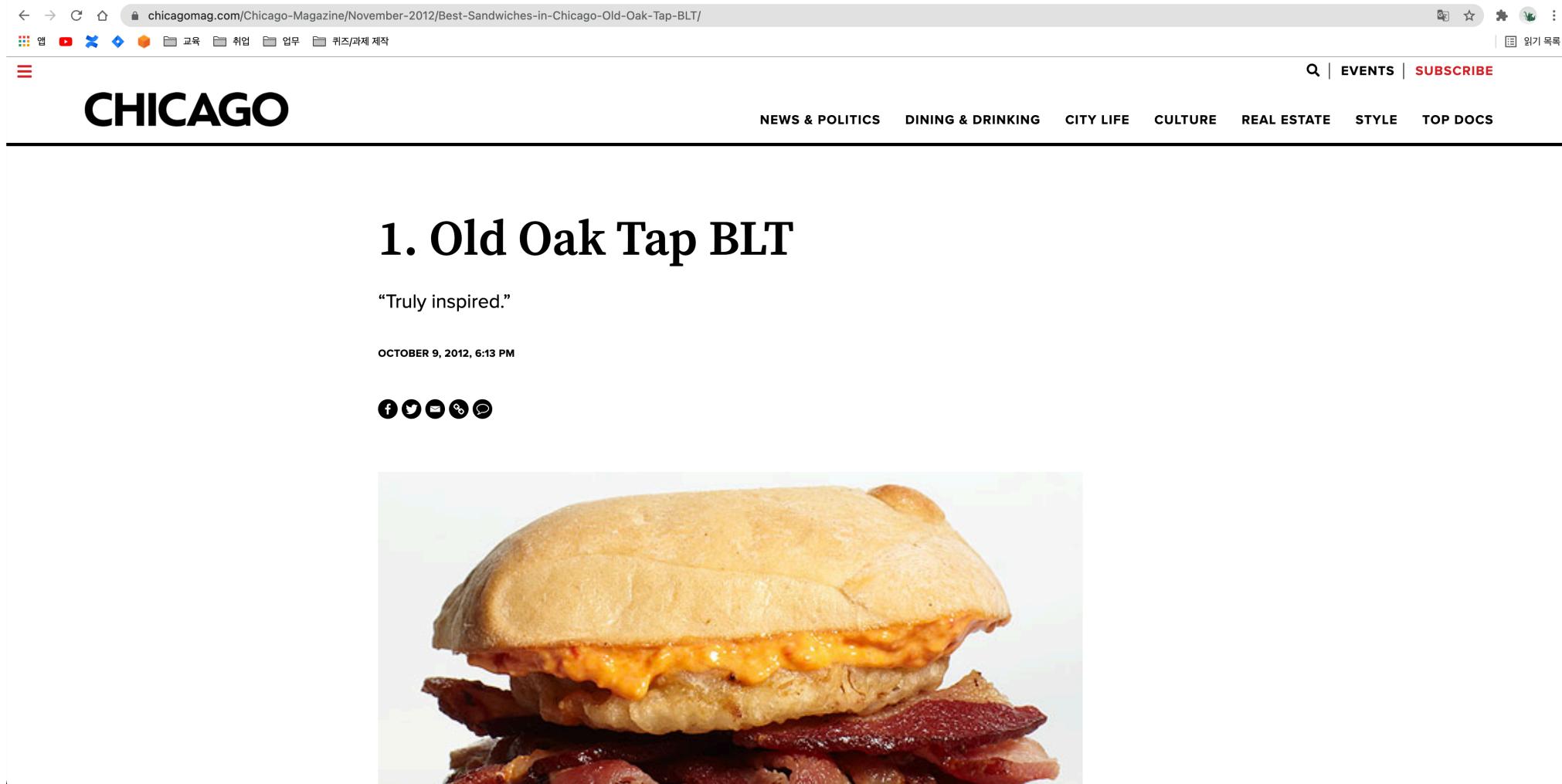
- 메인 페이지
- <https://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/>

chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/

≡ CHICAGO

- 1 **BLT**
Old Oak Tap
Read more
- 2 **Fried Bologna**
Au Cheval
Read more
- 3 **Woodland Mushroom**
Xoco
Read more
- 4 **Roast Beef**
Al's Deli
Read more
- 5 **PB&L**
Publican Quality Meats
Read more
- 6 **Belgian Chicken Curry Salad**
Hendrickx Belgian Bread Crafter
Read more
- 7 **Lobster Roll**
Acadia
Read more
- 8 **Smoked Salmon Salad**
Birchwood Kitchen
Read more

- 시카고의 50개 맛집 샌드위치 가게에 대한 메뉴와 가게 이름이 있다
- 그 중 하나의 페이지에 접속



The screenshot shows a web browser displaying the Chicago Magazine website at chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/. The page features a large image of a sandwich with bacon and cheese. The title '1. Old Oak Tap BLT' is prominently displayed, followed by the quote 'Truly inspired.' and the date 'OCTOBER 9, 2012, 6:13 PM'. Below the image are social sharing icons for Facebook, Twitter, Email, Google+, and Print.

- 순위 세부 페이지
- <https://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/>

The B is applewood smoked—nice and snappy. The L is arugula—fresh and peppery. The T is a fried green slice—jacketed in cornmeal and greaseless. Slathered with pimiento cheese, the grilled ciabatta somehow stays crisp, providing three distinct layers of crunch. Truly inspired.

\$10. 2109 W. Chicago Ave., 773-772-0406, theoldoaktap.com

[Previous](#)

50. The Gatsby
Phoebe's Bakery

[Next](#)

2. Fried Bologna
Au Cheval

- 가게 주소와 대표 메뉴의 가격이 있다

메인페이지

시카고의 50개 맛집 샌드위치에 대해 메뉴와 가게 이름이 정리되어 있다

세부페이지

각각을 소개한 50개의 페이지에 가게 주소와 대표 메뉴의 가격이 있다

최종목표

총 51개 페이지에서 각 가게의 정보를 가져온다

The B is applewood smoked—nice and snappy. The L is arugula—fresh and peppery. The T is a fried green slice—jacketed in cornmeal and greaseless. Slathered with pimiento cheese, the grilled ciabatta somehow stays crisp, providing three distinct layers of crunch. Truly inspired.

\$10. 2109 W. Chicago Ave., 773-772-0406, theoldoaktap.com

- 가게이름
- 대표메뉴
- 대표메뉴의 가격
- 가게주소

Previous
50. The Gatsby
Phoebe's Bakery

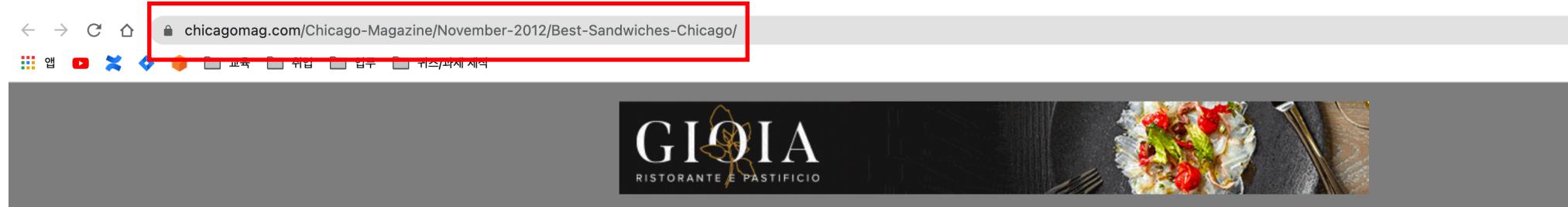
Next
2. Fried Bologna
Au Cheval

시카고 맷집 데이터 분석 - 메인페이지



접근 태그를 확인하자

zero-base /



The 50 Best Sandwiches in Chicago

Our list of Chicago's 50 best sandwiches, ranked in order of deliciousness

OCTOBER 9, 2012, 6:12 PM



시카고 샌드위치 맛집 소개 페이지를 분석해보자

```
In [20]: from bs4 import BeautifulSoup

# from urllib.request import urlopen
from urllib.request import Request, urlopen

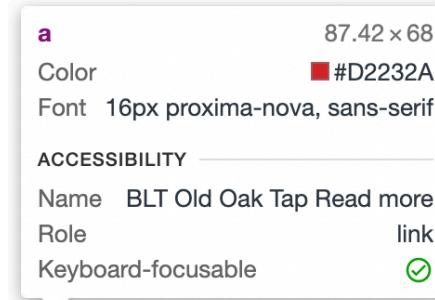
url_base = "http://www.chicagomag.com"
url_sub = "/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/"
url = url_base + url_sub

# html = urlopen(url)

req = Request(url, headers={"User-Agent": "Chrome"})
html = urlopen(req).read()
soup = BeautifulSoup(html, "html.parser")

soup
```

Out[20]: <!DOCTYPE html>



BLT
Old Oak Tap
Read more

- 2 Fried Bologna
- Au Cheval
- Read more*
- 3

```

▶ <p>...</p>
▶ <section class="related-content pull-right">...
</section>
▶ <p>...</p>
▶ <p>...</p>
▶ <p>...</p>
▶ <p>...</p>
▶ <p>...</p>
▼ <div class="sammy" style="position: relative;">
  <div class="sammyRank">1</div>
  ▼ <div class="sammyListing">
    ▶ <a href="/Chicago-Magazine/November-2012/B
      est-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/">
      ...</a> == $0
    ...
  ...

```

- div의 sammy 클래스가 눈에 보인다

```
In [21]: print(soup.find_all("div", "sammy"))
```

```
[<div class="sammy" style="position: relative;">
<div class="sammyRank">1</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-BLT</b><br/>
Old Oak Tap<br/>
<em>Read more</em> </a></div>
</div>, <div class="sammy" style="position: relative;">
<div class="sammyRank">2</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-na/"><b>Fried Bologna</b><br/>
Au Cheval<br/>
<em>Read more</em> </a></div>
</div>, <div class="sammy" style="position: relative;">
<div class="sammyRank">3</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-m/"><b>Woodland Mushroom</b><br/>
Xoco<br/>
```

- div의 sammy 클래스를 읽으면 될 것 같다

```
In [22]: len(soup.find_all("div", "sammy"))
```

```
Out[22]: 50
```

- 50개가 제대로 들어온 것을 확인!

```
In [23]: print(soup.find_all("div", "sammy")[0])
```

```
<div class="sammy" style="position: relative;">
<div class="sammyRank">1</div> ← 랭킹
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/"><b>BL
T</b><br/>
Old Oak Tap<br/> ← 가게 이름
<em>Read more</em> </a></div>
</div>
```

메뉴 ↑

- 한 항목만 살펴보기
- 원하는 정보 중에서 랭킹, 가게이름, 메뉴가 보인다

```
In [24]: tmp_one = soup.find_all("div", "sammy")[0]
type(tmp_one)
```

```
Out[24]: bs4.element.Tag
```

- type이 bs4.element.Tag 라는 것은 find 명령을 사용할 수 있다는 뜻

```
In [25]: tmp_one.find(class_="sammyRank")
```

```
Out[25]: <div class="sammyRank">1</div>
```

```
In [26]: tmp_one.find(class_="sammyRank").get_text()
```

```
Out[26]: '1'
```

- 랭킹 데이터 확보

```
In [27]: tmp_one.find(class_="sammyListing").get_text()
```

```
Out[27]: 'BLT\nOld Oak Tap\nRead more '
```

- 가게 이름과 메뉴 데이터가 한 번에 있다…

```
In [28]: tmp_one.find("a")["href"]
```

```
Out[28]: '/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/'
```

- 연결되는 홈페이지 주소가 “상대경로”이다…

```
In [29]: import re

tmp_string = tmp_one.find(class_="sammyListing").get_text()
re.split("\n|\r\n"), tmp_string)

Out[29]: ['BLT', 'Old Oak Tap', 'Read more ']
```

- 가게 이름과 메뉴는 re 모듈의 split으로 쉽게 구분할 수 있다

```
In [30]: print(re.split("\n|\r\n"), tmp_string)[0])
print(re.split("\n|\r\n"), tmp_string)[1])
```

```
BLT
Old Oak Tap
```

- 이렇게 찾을 수 있다!

50개 베스트 가게를 소개하는 메인페이지에서
랭크, 메뉴이름, 가게 이름,
각각의 가게를 소개하는 웹주소를 알아내는 방법 확인
이제 전체 반복문으로 완성하자...

```
In [31]: from urllib.parse import urljoin

url_base = "http://www.chicagomag.com"

rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all("div", "sammy")

for item in list_soup:
    rank.append(item.find(class_="sammyRank").get_text())
    tmp_string = item.find(class_="sammyListing").get_text()
    main_menu.append(re.split("\n|\r\n", tmp_string)[0])
    cafe_name.append(re.split("\n|\r\n", tmp_string)[1])
    url_add.append(urljoin(url_base, item.find("a")["href"]))
```

```
In [31]: from urllib.parse import urljoin

url_base = "http://www.chicagomag.com"

rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all("div", "sammy")

for item in list_soup:
    rank.append(item.find(class_="sammyRank").get_text())
    tmp_string = item.find(class_="sammyListing").get_text()
    main_menu.append(re.split("\n|\r\n", tmp_string)[0])
    cafe_name.append(re.split("\n|\r\n", tmp_string)[1])
    url_add.append(urljoin(url_base, item.find("a")["href"]))
```

필요한 내용을 담을 빈 리스트를 준비
리스트로 하나씩 컬럼을 만들고
DataFrame으로 합칠 예정

```
In [31]: from urllib.parse import urljoin

url_base = "http://www.chicagomag.com"

rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all("div", "sammy") → div의 sammy 태그 가져오기

for item in list_soup:
    rank.append(item.find(class_="sammyRank").get_text())
    tmp_string = item.find(class_="sammyListing").get_text()
    main_menu.append(re.split("\n|\r\n", tmp_string)[0])
    cafe_name.append(re.split("\n|\r\n", tmp_string)[1])
    url_add.append(urljoin(url_base, item.find("a")["href"]))
```

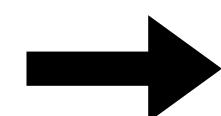
```
In [31]: from urllib.parse import urljoin

url_base = "http://www.chicagomag.com"

rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all("div", "sammy")

for item in list_soup:
    rank.append(item.find(class_="sammyRank").get_text())
    tmp_string = item.find(class_="sammyListing").get_text()
    main_menu.append(re.split("\n|\r\n", tmp_string)[0])
    cafe_name.append(re.split("\n|\r\n", tmp_string)[1])
    url_add.append(urljoin(url_base, item.find("a")["href"]))
```

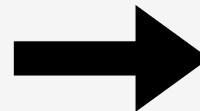


**for 문에서 가져온 태그를 순차적으로 반복하면서
랭크와 가게이름, 메뉴 데이터를 가져온다**

그런데 알 수 없는 이유로... 시카고 매거진이 이 페이지에서 연결하는 하위 50개 페이지의 주소가
상대주소와 절대주소가 혼용되어 있다

In [31]: `from urllib.parse import urljoin`

`url_base = "http://www.chicagomag.com"`



상대주소 절대주소 대응을 위한 명령

```
rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all("div", "sammy")

for item in list_soup:
    rank.append(item.find(class_="sammyRank").get_text())
    tmp_string = item.find(class_="sammyListing").get_text()
    main_menu.append(re.split("\n|\r\n", tmp_string)[0])
    cafe_name.append(re.split("\n|\r\n", tmp_string)[1])
    url_add.append(urljoin(url_base, item.find("a")["href"]))
```

```
In [32]: rank[:5]
```

```
Out[32]: ['1', '2', '3', '4', '5']
```

```
In [33]: main_menu[:5]
```

```
Out[33]: ['BLT', 'Fried Bologna', 'Woodland Mushroom', 'Roast Beef', 'PB&L']
```

```
In [34]: cafe_name[:5]
```

```
Out[34]: ['Old Oak Tap', 'Au Cheval', 'Xoco', 'Al's Deli', 'Publican Quality Meat']
```

```
In [35]: url_add[:5]
```

```
Out[35]: ['http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/',
          'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Au-Cheval-Fried-Bologna/',
          'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Xoco-Woodland-Mushroom/',
          'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Als-Deli-Roast-Beef/',
          'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Publican-Quality-Meats-PB-L/']
```

```
In [36]: len(rank), len(main_menu), len(cafe_name), len(url_add)
```

```
Out[36]: (50, 50, 50, 50)
```

```
In [37]: import pandas as pd
```

```
data = {"Rank": rank, "Menu": main_menu, "Cafe": cafe_name, "URL": url_add}
df = pd.DataFrame(data)
df.head()
```

Out[37]:

	Rank	Menu	Cafe	URL
0	1	BLT	Old Oak Tap	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Fried Bologna	Au Cheval	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Woodland Mushroom	Xoco	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Roast Beef	Al's Deli	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	PB&L	Publican Quality Meats	http://www.chicagomag.com/Chicago-Magazine/Nov...

- 50개 자료에 대해 랭킹, 메뉴, 가게이름, URL까지 모두 정리

```
In [38]: df = pd.DataFrame(data, columns=["Rank", "Cafe", "Menu", "URL"])
df.head(5)
```

Out[38]:

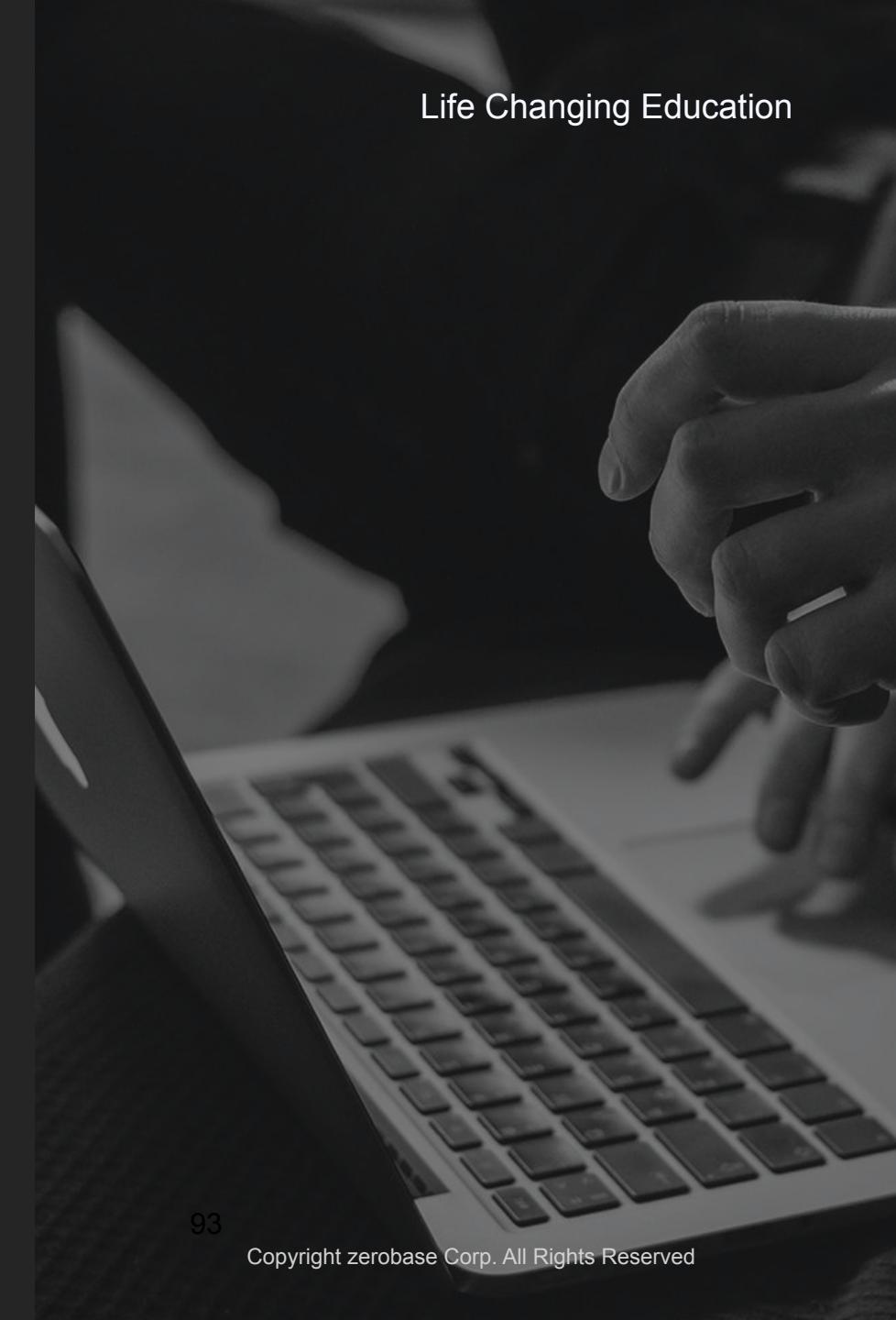
	Rank	Cafe	Menu	URL
0	1	Old Oak Tap	BLT	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Au Cheval	Fried Bologna	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Xoco	Woodland Mushroom	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Al's Deli	Roast Beef	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	Publican Quality Meats	PB&L	http://www.chicagomag.com/Chicago-Magazine/Nov...

- 칼럼 순서 변경
- Rank, Cafe, Menu, URL 순서

```
In [39]: df.to_csv(  
                  "../data/03. best_sandwiches_list_chicago.csv", sep=",", encoding="UTF-8"  
                 )
```

- 데이터 저장!

시카고 맷집 데이터 분석 - 하위페이지



다시 본론으로....

아직... 남았다 ~~ ^^

이제 URL 정보를 따라

50개 페이지 각각의

가격과 주소를 가져와야 한다.

뭐 당연한 이야기지만,
일단, 하나의 케이스만 먼저 도전하자 ~~

```
In [40]: from bs4 import BeautifulSoup
from urllib.request import urlopen

import pandas as pd
```

```
In [41]: df = pd.read_csv("../data/03. best_sandwiches_list_chicago.csv", index_col=0)
df.head()
```

Out[41]:				
	Rank	Cafe	Menu	URL
0	1	Old Oak Tap	BLT	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Au Cheval	Fried Bologna	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Xoco	Woodland Mushroom	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Al's Deli	Roast Beef	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	Publican Quality Meats	PB&L	http://www.chicagomag.com/Chicago-Magazine/Nov...

```
In [42]: df["URL"][0]
```

```
Out[42]: 'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches  
-in-Chicago-Old-Oak-Tap-BLT/'
```

- 50 URL 중 하나를 대상으로 잡는다
- 해당 페이지에 가보면...

The B is applewood smoked—nice and snappy.
 The L is arugula—fresh and peppery. The T is a
em 352.78×55
 Color #000000
 Font 20px utopia-std, serif
 ACCESSIBILITY
 Name
 Role emphasis
 Keyboard-focusable

keted in cornmeal and
 with pimiento cheese, the
 how stays crisp, providing
 of crunch. Truly inspired.

\$10. 2109 W. Chicago Ave., 773-772-0406,
theoldoaktap.com



1133 W RANDOLPH

- 내가 얻고 싶은 정보는 저기 있다…
- p 태그에 addy라는 class

```
<div class="grid-container grid-parent article-wrap">
  ::before
  <div class="grid-4 hide-on-tablet hide-on-mobile">
    <div class="grid-12 prefix-1 tablet-grid-14 tablet-prefix-1 mobile-grid-100 mobile-prefix-0 mobile-suffix-0">
      ::before
      <div class="article-body">
        <p>...</p>
        <p class="addy">
          <em>...</em> == $0
        </p>
        <aside class="story-nav-aside">...</aside>
      </div>
    </div>
  </div>
</div>
```

```
In [43]: # html = urlopen(df['URL'][0])
req = Request(df["URL"][0], headers={"User-Agent": "Chrome"})
html = urlopen(req).read()

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

```
In [44]: print(soup_tmp.find("p", "addy"))
```

```
<p class="addy">
<em>$10. 2109 W. Chicago Ave., 773-772-0406, <a href="http://www.theoldoaktap.co
```

- OK…
- 그런데 가격만 가져오고 싶은데…
- 가격과 주소가 같이 있다… 😱

\$10. 2109 W. Chicago Ave., 773-772-0406, theoldoaktap.com

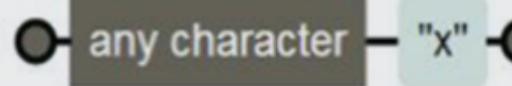
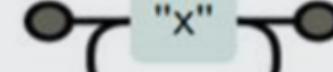
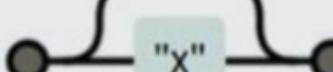
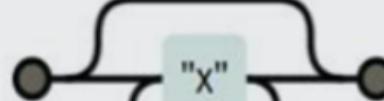
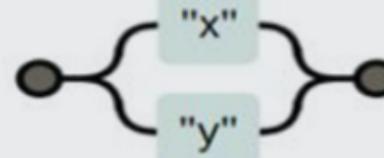
- 보자 ~~~
- 먼저 \$ 달러 기호를 만나서 나타나는 숫자들은 .을 만날 때 까지 가격이다
- 단, 10.5 처럼 . 기호 후에 다시 연달아 숫자가 나올 수도 있다
- 그리고 띄어쓰기 후에 숫자 혹은 문자가 나타나면 주소이다
- 주소는 ., 로 끝난다
- 이걸 조건문으로…??? 🤔

그래서 하나 더 ~~~ 연습하자.

그냥 이번 생은 망했다고 생각하고... 열심히 하자 ^^

Regular Expression

Regular Expression - 기초

.x		임의의 한 문자를 표현합니다. (x가 마지막으로 끝납니다.)
x ⁺		x가 1번이상 반복합니다.
x?		x가 존재하거나 존재하지 않습니다.
x [*]		x가 0번이상 반복합니다.
x y		x 또는 y를 찾습니다. (or연산자를 의미합니다.)

```
In [45]: price_tmp = soup_tmp.find("p", "addy").get_text()  
price_tmp
```

```
Out[45]: '\n$10. 2109 W. Chicago Ave., 773-772-0406, theoldoaktap.com'
```

- 먼저 text로 변환하고 ~~

```
In [46]: import re
```

```
re.split(".", price_tmp)
```

```
Out[46]: ['\n$10. 2109 W. Chicago Ave', ' 773-772-040', ' theoldoaktap.com']
```

```
In [47]: price_tmp = re.split(".", price_tmp)[0]  
price_tmp
```

```
Out[47]: '\n$10. 2109 W. Chicago Ave'
```

- 가격과 주소만 가져오기 위해 ., 로 분리하고 ~~

```
In [48]: re.search("\$\d+\.\d+?", price_tmp).group()
```

```
Out[48]: '$10.'
```

- 숫자로 시작하다가 꼭 .을 만나고 그 뒤 숫자가 있을 수도 있고 아닐 수도 있다

```
In [49]: tmp = re.search("\$\d+\.( \d+)?", price_tmp).group()  
price_tmp[len(tmp) + 2:]
```

```
Out[49]: '2109 W. Chicago Ave'
```

- 가격이 끝나는 지점의 위치를 이용해서 그 뒤는 주소로 생각한다

이제 방금한 내용을
50번 반복하면 되겠다 ~~~

pythonic 한 이야기

zero-base /

```
In [50]: price = []
address = []

for n in df.index[:3]:
    # html = urlopen(df['URL'][n])
    req = Request(df["URL"][n], headers={"User-Agent": "Mozilla/5.0"})
    html = urlopen(req).read()

    soup_tmp = BeautifulSoup(html, "lxml")

    gettings = soup_tmp.find("p", "addy").get_text()

    price_tmp = re.split(", ", gettings)[0]
    tmp = re.search("\$\d+\.\d+?", price_tmp).group()

    price.append(tmp)
    address.append(price_tmp[len(tmp) + 2 :])

print(n)
```

0
1
2

- 코드의 동작 유무를 살짝 확인하기 위해 세 번만 돌리자

그런데 ~~~~~

```
In [50]: price = []
address = []

for n in df.index[:3]:
    # html = urlopen(df['URL'][n])
    req = Request(df["URL"][n], headers={"User-Agent": "Mozilla/5.0"})
    html = urlopen(req).read()

    soup_tmp = BeautifulSoup(html, "lxml")

    gettings = soup_tmp.find("p", "addy").get_text()

    price_tmp = re.split(", ", gettings)[0]
    tmp = re.search("\$\d+\.\d+", price_tmp).group()

    price.append(tmp)
    address.append(price_tmp[len(tmp) + 2 :])

print(n)
```

이건 너무 C 스럽지 않나???

0
1
2

```
In [51]: for item in df[:3]["URL"]:  
    print(item)
```

```
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/  
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Au-Cheval-Fried-Bologna/  
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Xoco-Woodland-Mushroom/
```

- List 형 데이터를 반복시킬 때는 이렇게 하는 것이 정석 ~~
- 그러나 여러 컬럼을 for 문 내에서 사용할 때는 위 방법이 어렵다

```
In [52]: for n in df[:3].index:  
    print(df["URL"][n])
```

```
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/  
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Au-Cheval-Fried-Bologna/  
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Xoco-Woodland-Mushroom/
```

- 이렇게 배열의 순서를 찾는 것이 안될 것은 없지만 ~~

```
In [53]: for idx, row in df[:3].iterrows():
    print(row["URL"])
```

```
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Au-Cheval-Fried-Bologna/
http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Xoco-Woodland-Mushroom/
```

- iterrows() 함수를 사용하는 경우가 좀 더 좋다
- 받는 인자로 인덱스와 나머지 row를 받는다는 것에 주의하자

pythonic 한 이야기

zero-base /

```
In [54]: price = []
address = []

for idx, row in df[:3].iterrows():
    # html = urlopen(row['URL'])
    req = Request(row["URL"], headers={"User-Agent": "Chrome"})
    html = urlopen(req).read()

    soup_tmp = BeautifulSoup(html, "lxml")

    gettings = soup_tmp.find("p", "addy").get_text()

    price_tmp = re.split(", ", gettings)[0]
    tmp = re.search("\$\d+\.\d+", price_tmp).group()

    price.append(tmp)
    address.append(price_tmp[len(tmp) + 2 :])

print(idx)
```

```
0
1
2
```

```
In [55]: price
```

```
Out[55]: ['$10.', '$9.', '$9.50']
```

```
In [56]: address
```

```
Out[56]: ['2109 W. Chicago Ave', '800 W. Randolph St', ' 445 N. Clark St']
```

- OK
- 이제 전체로 다 돌리자

OK 50번 가자 ~~~~~

(상투적인 수법... 이젠 익숙해 지셨죠???)

그런데.... (라고 말할려고 한다는 걸^^)

for 문을 사용할 때

이게 동작중인지...

얼마나 시간이 남은건지..ㅠㅠ.

답답할 때가 많습니다.

이 때, 사용하는 TQDM

```
In [57]: from tqdm import tqdm

price = []
address = []

for idx, row in tqdm(df.iterrows()):
    # html = urlopen(row['URL'])
    req = Request(row["URL"], headers={"User-Agent": "Mozilla/5.0"})
    html = urlopen(req).read()

    soup_tmp = BeautifulSoup(html, "lxml")

    gettings = soup_tmp.find("p", "addy").get_text()

    price_tmp = re.split(", ", gettings)[0]
    tmp = re.search("\$\d+\.\d+", price_tmp).group()

    price.append(tmp)
    address.append(price_tmp[len(tmp) + 2:] )
```

- conda install -c conda-forge tqdm

```
In [58]: price
```

```
Out[58]: ['$10.',  
 '$9.',  
 '$9.50',  
 '$9.40',  
 '$10.',  
 '$7.25',
```

```
In [59]: address
```

```
Out[59]: ['2109 W. Chicago Ave',  
 '800 W. Randolph St',  
 ' 445 N. Clark St',  
 ' 914 Noyes St',  
 '825 W. Fulton Mkt',  
 ' 100 E. Walton St',
```

```
In [60]: df.head(10)
```

	Rank	Cafe	Menu	URL
0	1	Old Oak Tap	BLT	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Au Cheval	Fried Bologna	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Xoco	Woodland Mushroom	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Al's Deli	Roast Beef	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	Publican Quality Meats	PB&L	http://www.chicagomag.com/Chicago-Magazine/Nov...
5	6	Hendrickx Belgian Bread Crafter	Belgian Chicken Curry Salad	https://www.chicagomag.com/Chicago-Magazine/No...
6	7	Acadia	Lobster Roll	http://www.chicagomag.com/Chicago-Magazine/Nov...
7	8	Birchwood Kitchen	Smoked Salmon Salad	http://www.chicagomag.com/Chicago-Magazine/Nov...
8	9	Cemitas Puebla	Atomica Cemitas	http://www.chicagomag.com/Chicago-Magazine/Nov...
9	10	Nana	Grilled Laughing Bird Shrimp and Fried Po' Boy	http://www.chicagomag.com/Chicago-Magazine/Nov...

```
In [61]: len(price), len(address), len(df)
```

```
Out[61]: (50, 50, 50)
```

```
In [62]: df["Price"] = price
df["Address"] = address

df = df.loc[:, ["Rank", "Cafe", "Menu", "Price", "Address"]]
df.set_index("Rank", inplace=True)
df.head()
```

Out[62]:

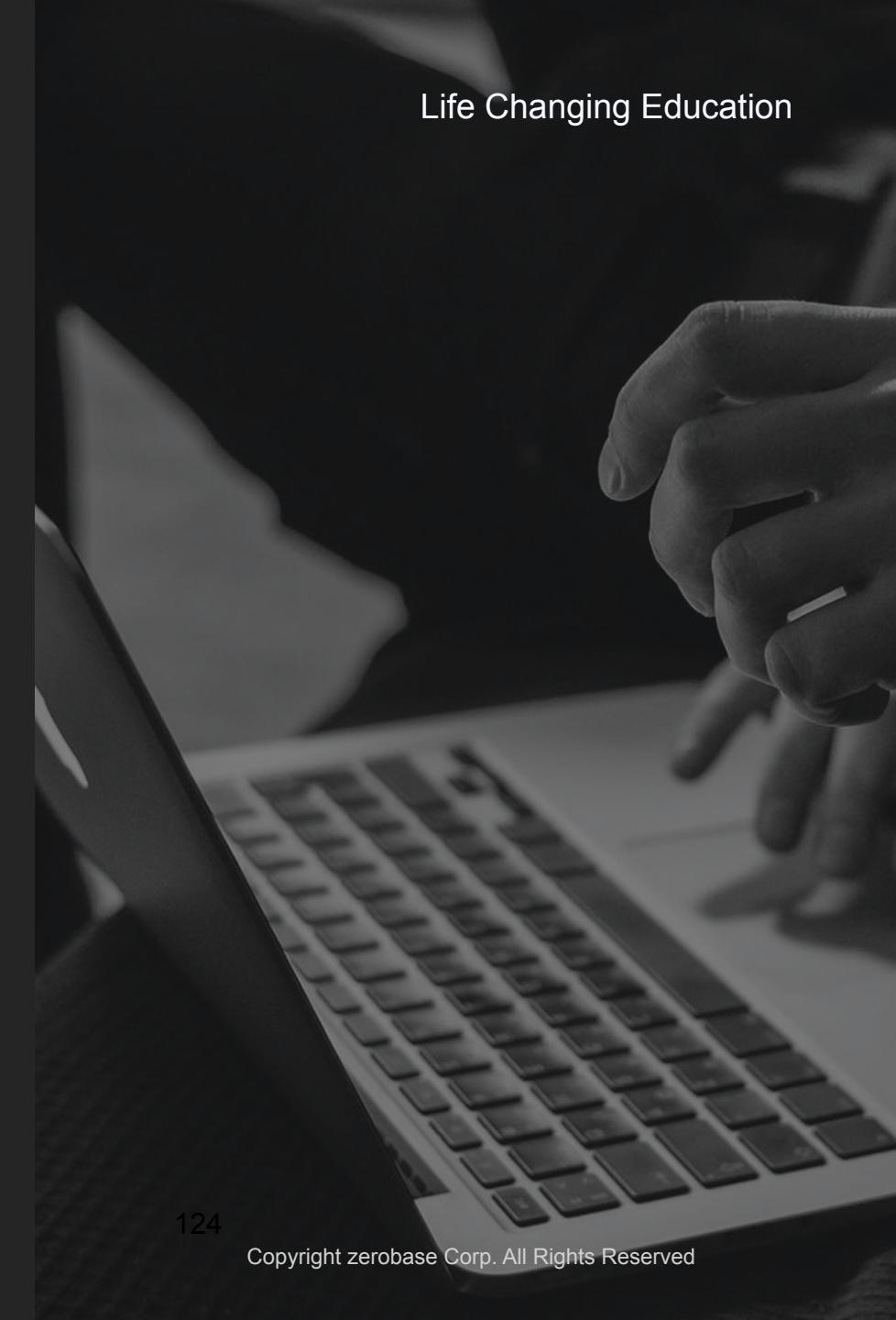
Rank	Cafe	Menu	Price	Address
1	Old Oak Tap	BLT	\$10.	2109 W. Chicago Ave
2	Au Cheval	Fried Bologna	\$9.	800 W. Randolph St
3	Xoco	Woodland Mushroom	\$9.50	445 N. Clark St
4	Al's Deli	Roast Beef	\$9.40	914 Noyes St
5	Publican Quality Meats	PB&L	\$10.	825 W. Fulton Mkt

```
In [63]: df.to_csv(  
    "../data/03. best_sandwiches_list_chicago2.csv", sep=",", encoding="UTF-8"  
)
```

```
In [64]: df
```

Rank	Cafe	Menu	Price	Address
1	Old Oak Tap	BLT	\$10.	2109 W. Chicago Ave
2	Au Cheval	Fried Bologna	\$9.	800 W. Randolph St
3	Xoco	Woodland Mushroom	\$9.50	445 N. Clark St
4	Al's Deli	Roast Beef	\$9.40	914 Noyes St

시카고 맷집 데이터 지도 시각화



```
In [1]: import folium  
import pandas as pd  
import googlemaps  
import numpy as np  
from tqdm import tqdm
```

```
In [2]: df = pd.read_csv("../data/03. best_sandwiches_list_chicago2.csv", index_col=0)  
df.head(5)
```

Out[2]:

Rank	Cafe	Menu	Price	Address
1	Old Oak Tap	BLT	\$10.	2109 W. Chicago Ave
2	Au Cheval	Fried Bologna	\$9.	800 W. Randolph St
3	Xoco	Woodland Mushroom	\$9.50	445 N. Clark St
4	Al's Deli	Roast Beef	\$9.40	914 Noyes St
5	Publican Quality Meats	PB&L	\$10.	825 W. Fulton Mkt

```
In [3]: gmaps_key = "geocoding api key geocoding api key"
gmaps = googlemaps.Client(key=gmaps_key)
```

```
In [9]: lat = []
lng = []

for idx, row in tqdm(df.iterrows()):
    if not row["Address"] == "Multiple location":
        target_name = row["Address"] + ", " + "Chicago"
        gmaps_output = gmaps.geocode(target_name)
        location_output = gmaps_output[0].get("geometry")
        lat.append(location_output["location"]["lat"])
        lng.append(location_output["location"]["lng"])

    else:
        lat.append(np.nan)
        lng.append(np.nan)
```

```
In [69]: len(lat), len(lng)
```

```
Out[69]: (50, 50)
```

```
In [11]: df["lat"] = lat  
df["lng"] = lng  
df.head()
```

Out[11]:

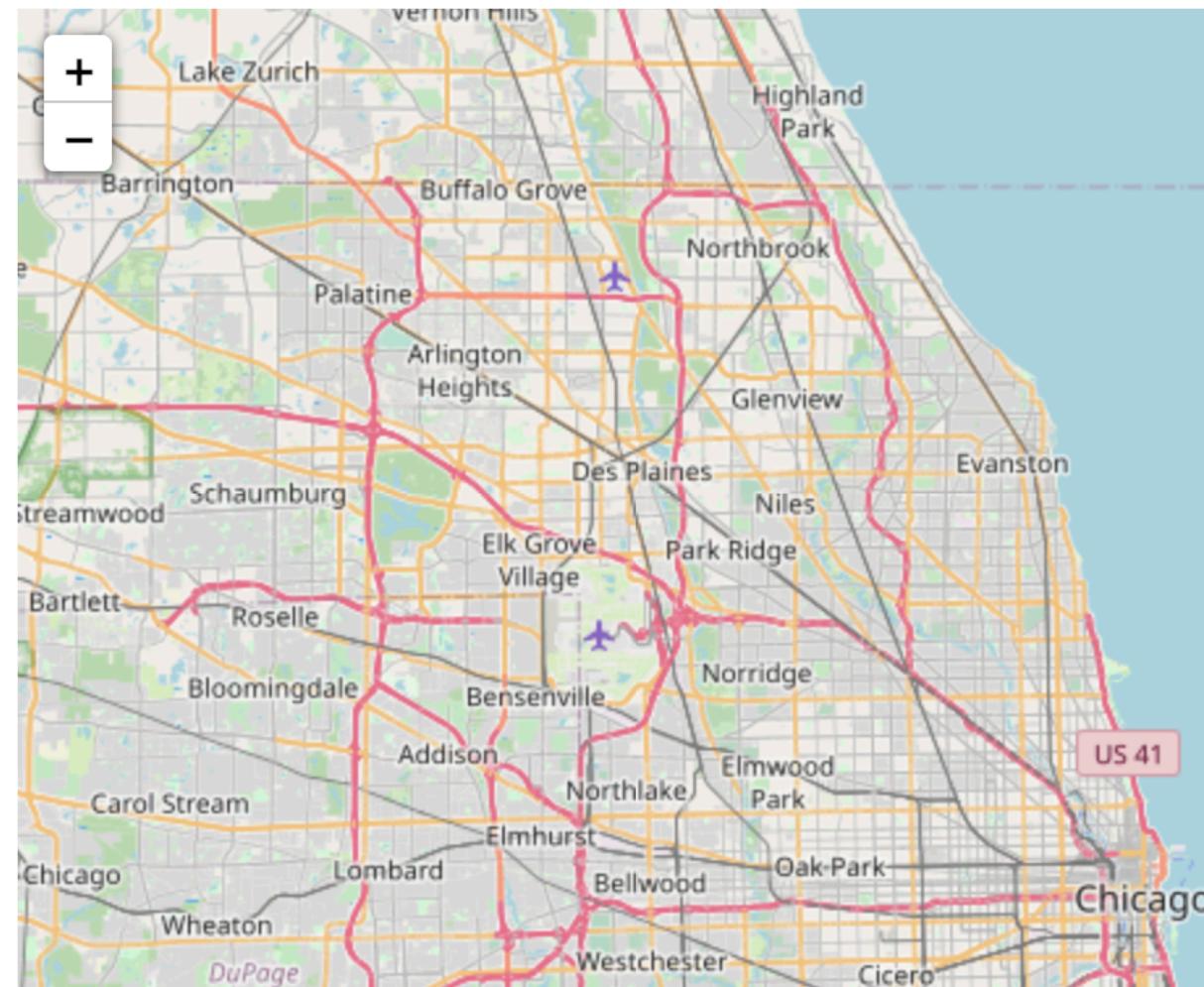
Rank	Cafe	Menu	Price	Address	lat	lng
1	Old Oak Tap	BLT	\$10.	2109 W. Chicago Ave	41.895605	-87.679961
2	Au Cheval	Fried Bologna	\$9.	800 W. Randolph St	41.884639	-87.647590
3	Xoco	Woodland Mushroom	\$9.50	445 N. Clark St	41.890523	-87.630783
4	Al's Deli	Roast Beef	\$9.40	914 Noyes St	42.058322	-87.683748
5	Publican Quality Meats	PB&L	\$10.	825 W. Fulton Mkt	41.886604	-87.648536

Folium을 이용한 지도 시각화

zero-base /

```
In [12]: mapping = folium.Map(location=[41.8781136, -87.6297982], zoom_start=11)  
mapping
```

Out[12]:



```
In [13]: mapping = folium.Map(location=[41.8781136, -87.6297982], zoom_start=11)

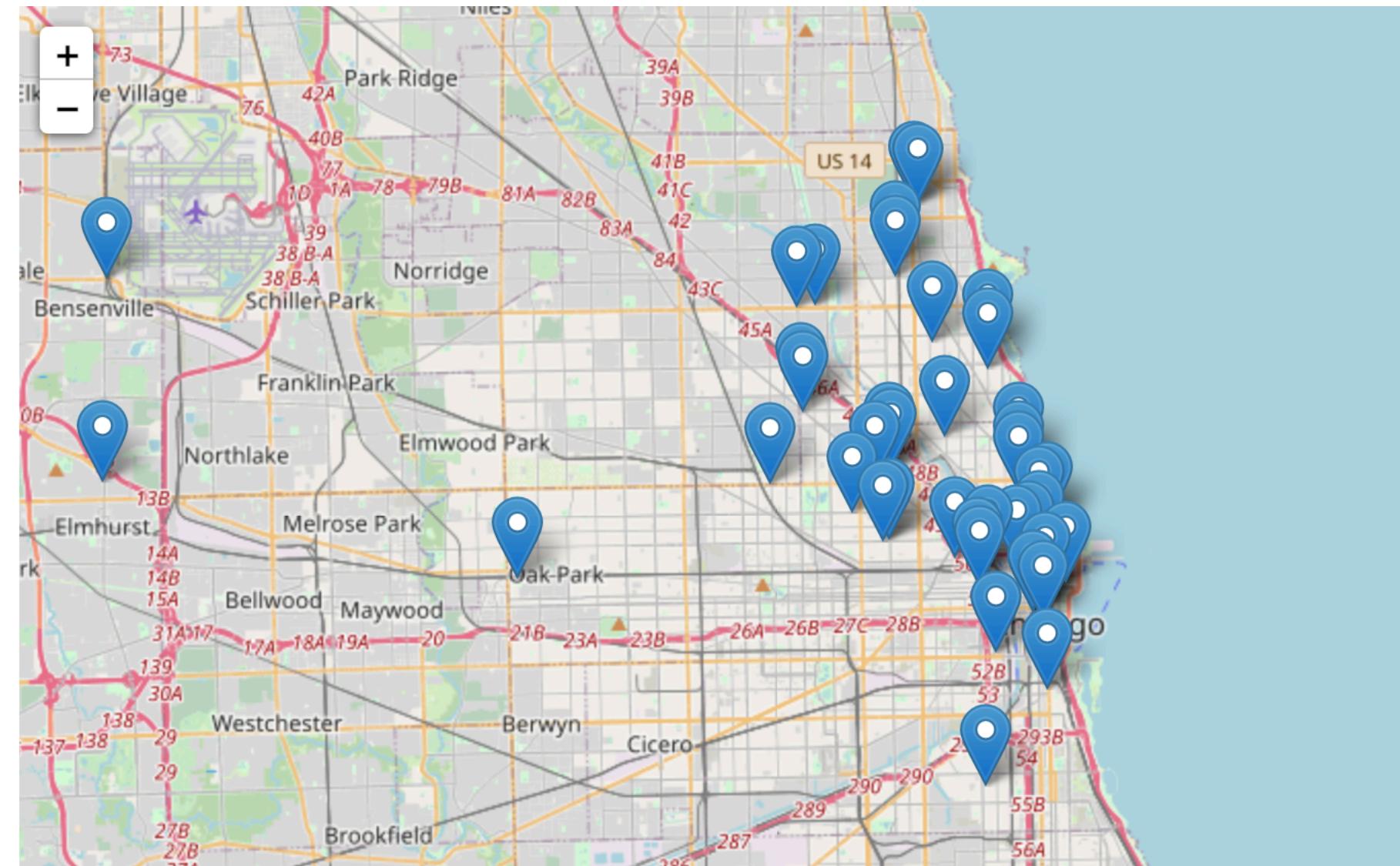
for idx, row in df.iterrows():
    if not row["Address"] == "Multiple location":
        folium.Marker([row["lat"], row["lng"]], popup=row["Cafe"]).add_to(
            mapping
        )

mapping
```

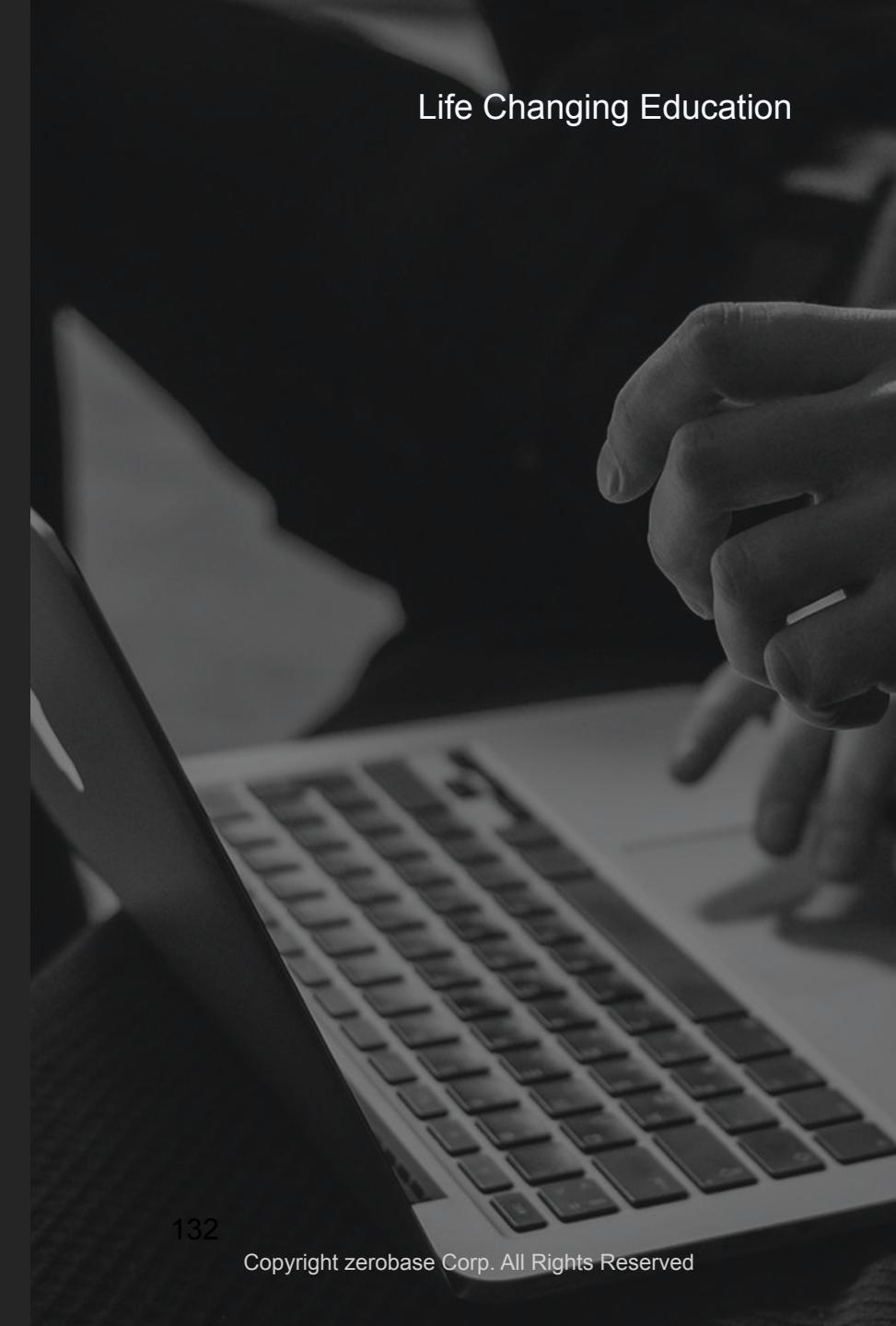
Folium을 이용한 지도 시각화

zero-base /

Out[13]:



네이버 영화 평점 사이트 분석



NAVER 영화

로그인 영화검색 검색

영화홈

예매순 현재상영작 개봉예정작 평점순 박스오피스 다운로드순 전체보기 ▶

○상영작·예정작

영화랭킹

예매

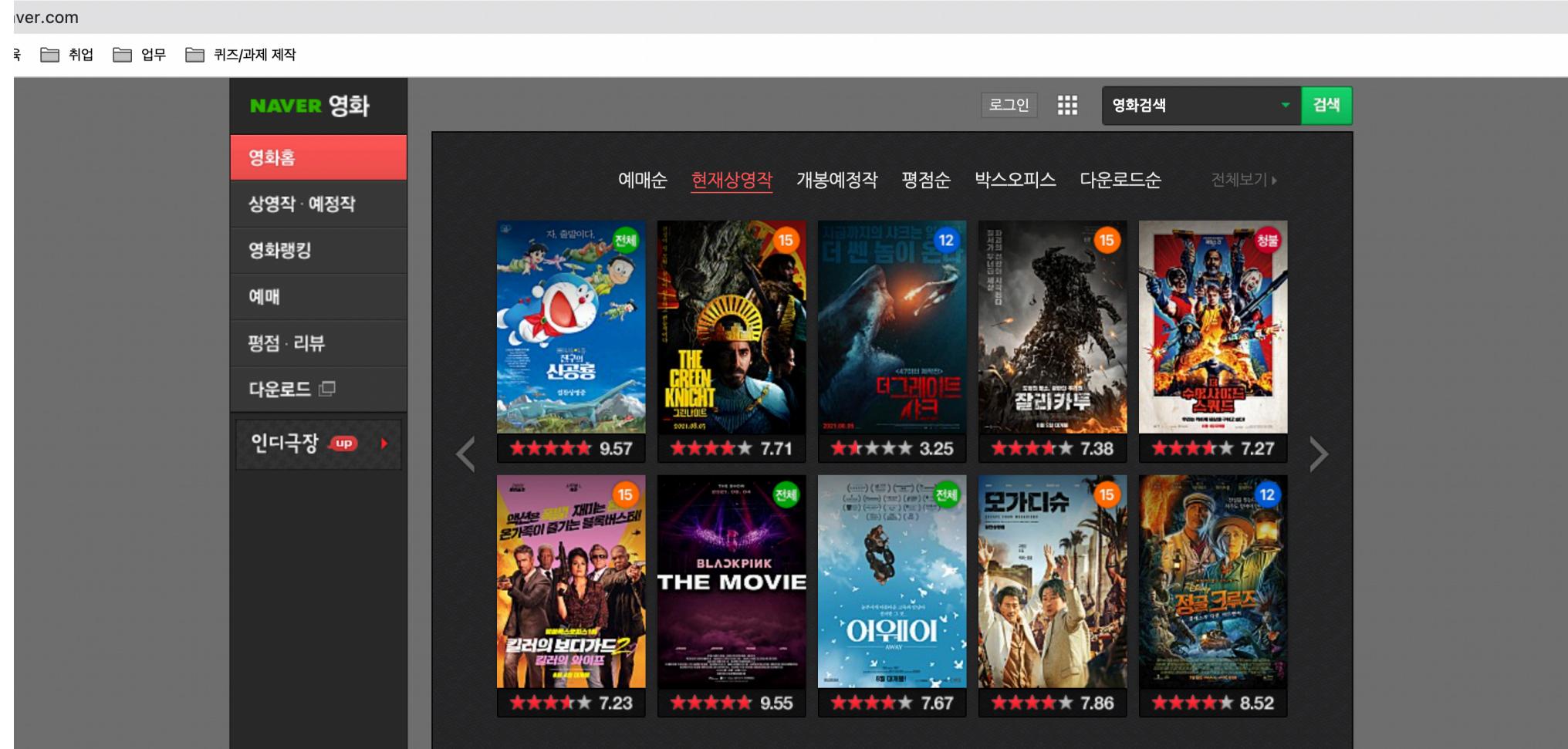
평점·리뷰

다운로드 □

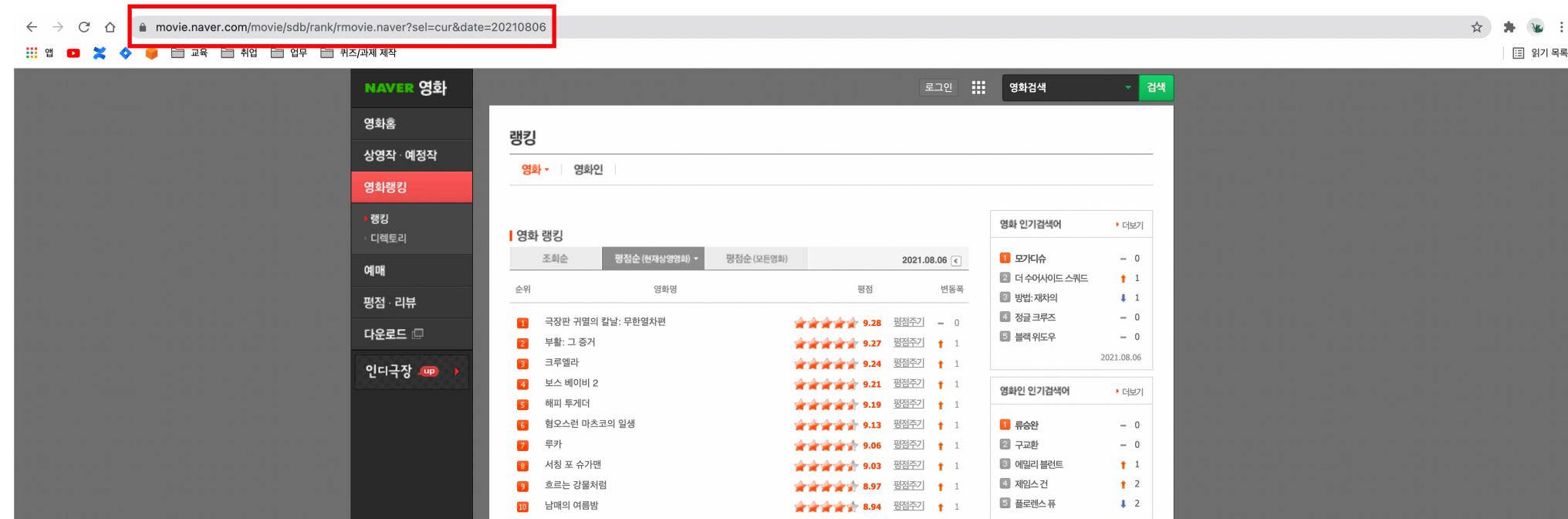
인디극장 up ▶

네이버 영화 평점

순위	영화 제작	평점
1	갈매기	9.79
2	학교 가는 길	9.63
3	자, 춤翩이다.	9.56
4	BLACKPINK THE MOVIE	9.55
5	Shaman Read	9.43
6	교실 안의 마크	9.39
7	유노	9.38
8	아이들로	9.34
9	나투자 않는다!!	9.28
10	부활	9.27



- 네이버 영화 메인 페이지(<https://movie.naver.com/>)
- 영화랭킹 탭 이동



- 영화랭킹에서 평점순(현재상영영화) 선택
- 접근 URL 확인
- <https://movie.naver.com/movie/sdb/rank/rmovie.naver?sel=cur&date=20210806>

<https://movie.naver.com/movie/sdb/rank/rmovie.naver?sel=cur&date=20210806>

- 웹 페이지의 주소에는 많은 정보가 담겨 있어서
- 원하는 정보를 얻기 위해 변화시켜야 하는 주소의 규칙이 보이기도 한다
- 이 경우 날짜 정보를 변경해주면 해당 페이지에 그냥 접근이 가능하다

네이버 영화 평점

```
In [1]: from bs4 import BeautifulSoup
import pandas as pd
```

```
In [2]: from urllib.request import urlopen

url = "http://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20180315"
page = urlopen(url)

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

```
Out[2]:
<!DOCTYPE html>

<html lang="ko">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta content="http://imgmovie.naver.com/today/naverme/naverme_profile.jpg" property="me2:image">
<meta content="네이버영화" property="me2:post_tag">
<meta content="네이버영화" property="me2:category1"/>
<meta content="" property="me2:category2"/>
<meta content="랭킹 : 네이버 영화" property="og:title"/>
<meta content="영화, 영화인, 예매, 박스오피스 랭킹 정보 제공" property="og:description"/>
<meta content="article" property="og:type"/>
<meta content="https://movie.naver.com/movie/sdb/rank/rmovie.naver?sel=cur&date=20180315" property="og:url"/>
<meta content="http://static.naver.net/m/movie/icons/OG_270_270.png" property="og:image"/><!-- http://static.naver.net/m/movie/im/navermovie.jpg -->
```

- 일단 한 페이지만 먼저 접근해 보자
- 날짜는 20180315로 시작

영화 랭킹

조회순

평점순 (현재상영영화) ▾

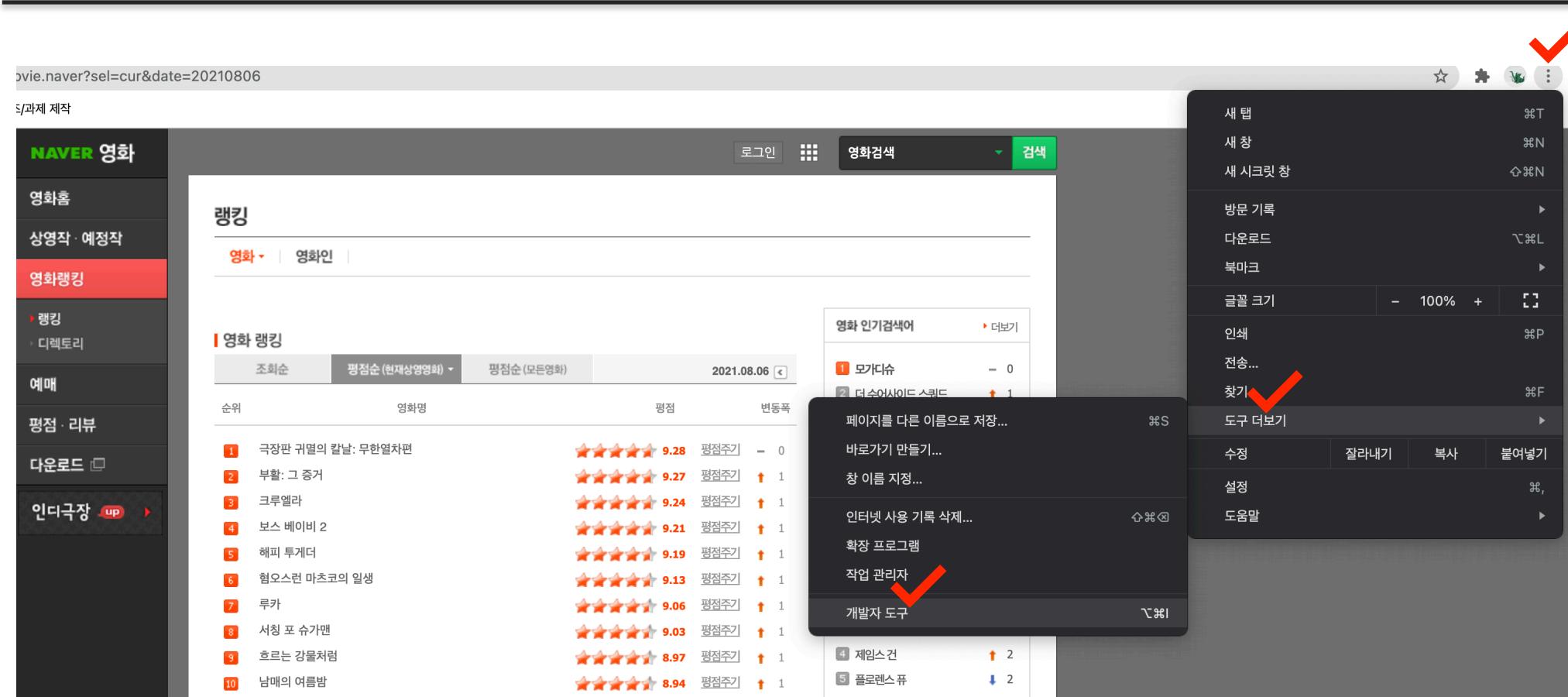
평점순 (모든영화)

2021.08.07

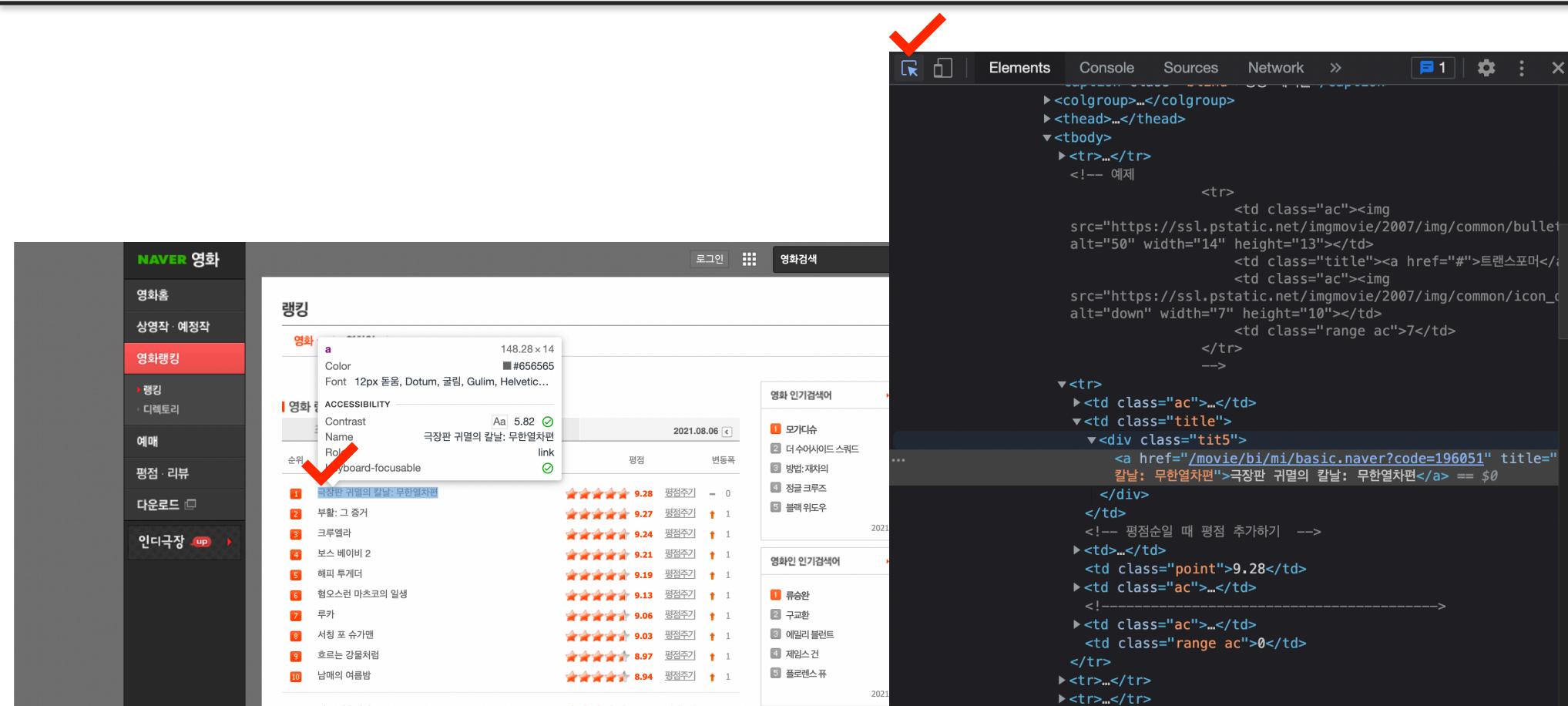


순위	영화명	평점	변동폭
1	극장판 귀멸의 칼날: 무한열차편	★★★★★ 9.28	평점주기 - 0
2	부활: 그 증거	★★★★★ 9.27	평점주기 - 0
3	크루엘라	★★★★★ 9.23	평점주기 - 0
4	보스 베이비 2	★★★★★ 9.21	평점주기 - 0
5	해피 투게더	★★★★★ 9.19	평점주기 - 0
6	혐오스런 마츠코의 일생	★★★★★ 9.13	평점주기 - 0
7	루카	★★★★★ 9.06	평점주기 - 0
8	서칭 포 슈가맨	★★★★★ 9.03	평점주기 - 0

- 여기서 영화 제목과 평점을 가져오고 싶다



- 크롬 개발자 도구 활용
- 크롬 설정 - 도구 더보기 - 개발자 도구



The screenshot shows the Naver Movie Ranking page. The movie '극장판 귀멸의 칼날: 무한열차편' is at the top of the ranking. A red checkmark is placed over the movie title in the browser's developer tools (Elements tab) to indicate it's the target element for extraction. The developer tools also show the HTML structure of the page, including the movie's title, rating, and other details.

- 영화 제목은 <td class = “title”> 안에 <div class = “tit5”> 안에 <a> 태그 안에
- 영화 평점은 <td class = “point”>

```
In [3]: soup.find_all("div", "tit5")
```

```
Out[3]: [<div class="tit5">
    <a href="/movie/bi/mi/basic.naver?code=151196" title="원더">원더</a>
    </div>,
    <div class="tit5">
        <a href="/movie/bi/mi/basic.naver?code=106360" title="위대한 쇼맨">위대한 쇼맨</a>
    </div>,
    <div class="tit5">
        <a href="/movie/bi/mi/basic.naver?code=18847" title="타이타닉">타이타닉</a>
    </div>,
    <div class="tit5">
        <a href="/movie/bi/mi/basic.naver?code=151752" title="온리 더 브레이브">온리 더 브레이브</a>
    </div>,
    <div class="tit5">
        <a href="/movie/bi/mi/basic.naver?code=31013" title="빌리 엘리어트">빌리 엘리어트</a>
    </div>,
    <div class="tit5">
        <a href="/movie/bi/mi/basic.naver?code=172006" title="젝스키스 에이틴">젝스키스 에이틴</a>
    </div>,
    <div class="tit5">
        <a href="/movie/bi/mi/basic.naver?code=151728" title="코코">코코</a>
    </div>]
```

- find_all 명령으로 쉽게 접근
- div 태그의 tit5를 확인해보니 안에 a 태그가 보인다

```
In [4]: soup.find_all("div", "tit5")[0].a
```

```
Out[4]: <a href="/movie/bi/mi/basic.naver?code=151196" title="원더">원더</a>
```

```
In [5]: soup.find_all("div", "tit5")[0].a.string
```

```
Out[5]: '원더'
```

- 영화 제목은 a 태그에서 string으로 글자를 가져오면 알 수 있다

zero-base /

- td 태그에 point 클래스를 확인하면 영화 평점을 얻을 수 있다

```
In [6]: soup.find_all("td", "point")
```

```
Out[6]: [<td class="point">9.40</td>,
          <td class="point">9.38</td>,
          <td class="point">9.29</td>,
          <td class="point">9.28</td>,
          <td class="point">9.28</td>,
          <td class="point">9.25</td>,
          <td class="point">9.22</td>,
          <td class="point">9.19</td>,
          <td class="point">9.19</td>,
          <td class="point">9.17</td>,
          <td class="point">9.12</td>,
          <td class="point">9.10</td>,
          <td class="point">9.01</td>,
          <td class="point">8.96</td>,
          <td class="point">8.89</td>,
          <td class="point">8.85</td>,
          <td class="point">8.81</td>,
          <td class="point">8.80</td>,
          <td class="point">8.78</td>,
          <td class="point">8.78</td>,
          <td class="point">8.69</td>,
          <td class="point">8.59</td>,
          <td class="point">8.49</td>,
```

```
In [7]: len(soup.find_all("td", "point"))
```

```
Out[7]: 46
```

```
In [8]: soup.find_all("td", "point")[0].string
```

```
Out[8]: '9.40'
```

- OK 반복문이 등장할 차례

```
In [9]: end = len(soup.find_all("div", "tit5"))
movie_name = [soup.find_all("div", "tit5")[n].a.string for n in range(0, end)]
movie_name

Out[9]: ['원더',
          '위대한 쇼맨',
          '타이타닉',
          '온리 더 브레이브',
          '빌리 엘리어트',
          '젝스키스 에이틴',
          '코코',
          .
          .
          .
          '사라진 밤',
          '궁합',
          '조선명탐정: 흡혈괴마의 비밀',
          '치즈인더트랩',
          '골든슬럼버',
          '게이트',
          '120BPM']
```

- 매 페이지마다 가져오는 영화의 갯수가 조금씩 다르다…
- 그래서 end 변수에 한 페이지의 영화제목 갯수를 넣고
- 0부터 그 갯수(실제로는 하나 빼기^^) 만큼 for문을 돌려준다…

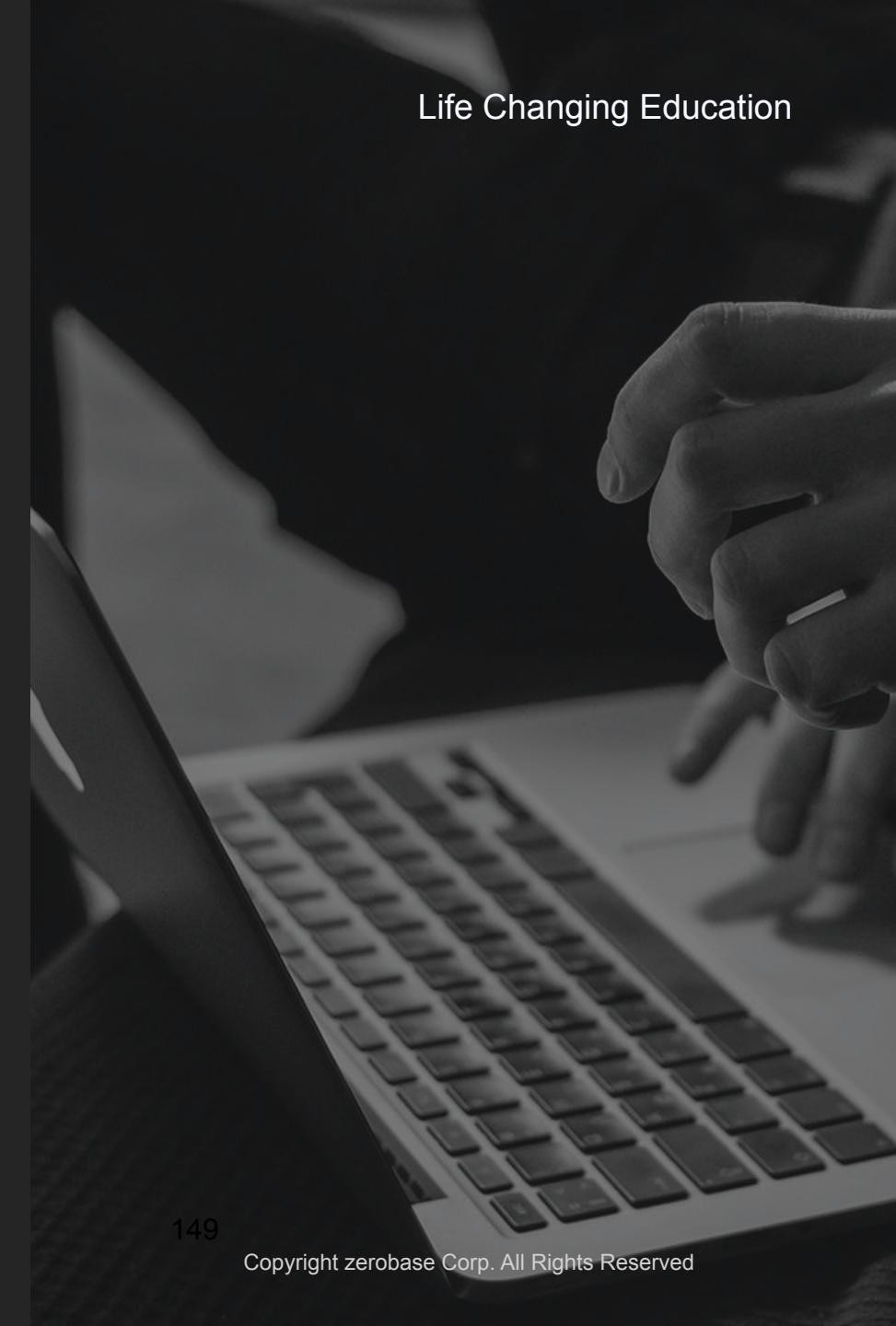
```
In [10]: end = len(soup.find_all("td", "point"))
movie_point = [soup.find_all("td", "point")[n].string for n in range(0, end)]
movie_point
```

```
Out[10]: ['9.40',
 '9.38',
 '9.29',
 '9.28',
 '9.28',
 '9.25',
 '9.22',
 '9.19',
 '9.19',
 '9.17',
 '9.12',
 '9.10',
 '9.01',
 '8.96',
 '8.89',
 '8.85',
 '8.81',
```

- 포인트도 마찬가지!

```
In [11]: len(movie_name), len(movie_point)  
Out[11]: (46, 46)
```

자동화를 위한 코드



<https://movie.naver.com/movie/sdb/rank/rmovie.naver?sel=cur&date=20180315>

- 방금.. 한 웹페이지에서 데이터를 얻을 수 있게 되었다
- 그러면 위의 날짜만 바꿔주면 우리가 원하는 기간 만큼 데이터를 얻겠다
- 그러면 날짜를 만들자!

In [31]:

```
1 date = pd.date_range("2017.12.01", periods=100, freq="D")
2 date
```

Out[31]:

```
DatetimeIndex(['2017-12-01', '2017-12-02', '2017-12-03', '2017-12-04',
                 '2017-12-05', '2017-12-06', '2017-12-07', '2017-12-08',
                 '2017-12-09', '2017-12-10', '2017-12-11', '2017-12-12',
                 .
                 .
                 .
                 '2018-02-27', '2018-02-28', '2018-03-01', '2018-03-02',
                 '2018-03-03', '2018-03-04', '2018-03-05', '2018-03-06',
                 '2018-03-07', '2018-03-08', '2018-03-09', '2018-03-10'],
                dtype='datetime64[ns]', freq='D')
```

- pandas의 date_range를 이용하면, 손쉽게 날짜를 만들 수 있다
- 2017.12.01 부터 100일 생성

```
In [34]: 1 date[0]
```

```
Out[34]: Timestamp('2017-12-01 00:00:00', freq='D')
```

```
In [35]: 1 date[0].strftime("%Y-%m-%d")
```

```
Out[35]: '2017-12-01'
```

```
In [36]: 1 date[0].strftime("%y.%m.%d")
```

```
Out[36]: '17.12.01'
```

- 날짜형 데이터들은 원하는 형태로 출력이 가능하다

```
In [14]: test_string = "Hi, I'm {name}"
print(test_string.format(name="MeRui"))
print(test_string.format(name="Sun"))
```

```
Hi, I'm MeRui
Hi, I'm Sun
```

- 파이썬의 string(문자형) 데이터형은 format이라는 재미난 기능이 있다
- {} 중괄호로 두고, format 옵션으로 손쉽게 내용을 만들 수 있다

```
(ds_study) yongha ➤ ~ conda install tqdm
```

- conda install tqdm

```
In [13]: from tqdm import tqdm_notebook  
# from tqdm.notebook import tqdm  
import time  
  
movie_date = []  
movie_name = []  
movie_point = []
```

- 필요한 모듈 하나 부르고...
- 얻고 싶은 데이터를 저장할 빈 리스트 생성

```
In [*]: for today in tqdm_notebook(date):
    html = "http://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date={date}"
    response = urlopen(html.format(date=today.strftime("%Y%m%d")))
    soup = BeautifulSoup(response, "html.parser")

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

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

    time.sleep(0.5)
```

61%

61/100 [00:52<00:32, 1.21it/s]

- 이제는 100일간의 데이터를 그냥 for문으로 받아 오기만 하면 된다~~

```
In [124]: len(movie_date), len(movie_name), len(movie_point)  
Out[124]: (4655, 4655, 4655)
```

- 방금 꽤 많은 정보를 얻었고, 그 크기가 문제 없음을 확인

```
In [57]: 1 movie = pd.DataFrame({"date": movie_date, "name": movie_name, "point": movie_point})  
2 movie.head()
```

Out[57]:

	date	name	point
0	2017-12-01	뷰티풀 투모로우	9.36
1	2017-12-01	아이 캔 스피크	9.34
2	2017-12-01	다시 태어나도 우리	9.32
3	2017-12-01	록키	9.31
4	2017-12-01	저수지 게임	9.29

- Pandas 데이터프레임으로 만들자…
- 이 데이터는 Raw Data 가 된다

In [58]:

1 movie.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4237 entries, 0 to 4236
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype  
--- 
 0   date      4237 non-null    datetime64[ns]
 1   name      4237 non-null    object  
 2   point     4237 non-null    object  
dtypes: datetime64[ns](1), object(2)
memory usage: 99.4+ KB
```

- 100일치 영화 평점 데이터도 100KB가 채 안된다…
- 그런데… point 가 숫자가 아니네…😭

In [41]:

```
1 movie["point"] = movie["point"].astype(float)
2 movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4237 entries, 0 to 4236
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype  
--- 
 0   date      4237 non-null    datetime64[ns]
 1   name      4237 non-null    object  
 2   point     4237 non-null    float64 
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 99.4+ KB
```

- astype 명령으로 point 를 수자로 바꾸고…
- 확인

```
In [25]: movie.to_csv("../data/04_naver_movie_raw_data.csv", sep=",", encoding="utf-8")
```

- 당연히 정신건강을 위한 저장~~!

영화 평점 데이터 정리



In [59]:

```
1 import numpy as np
2 import pandas as pd
3
4 movie = pd.read_csv("../data/04_naver_movie_raw_data.csv", index_col=0)
5 movie.head()
```

Out[59]:

	date	name	point
0	2017-12-01	뷰티풀 투모로우	9.36
1	2017-12-01	아이 캔 스피크	9.34
2	2017-12-01	다시 태어나도 우리	9.32
3	2017-12-01	록키	9.31
4	2017-12-01	저수지 게임	9.29

In [60]:

```
1 movie_unique = pd.pivot_table(movie, index=["name"], aggfunc=np.sum)
2 movie_best = movie_unique.sort_values(by="point", ascending=False)
3 movie_best.head(10)
```

Out[60]:

name	point
러빙 빈센트	919.11
내 사랑	898.76
라라랜드	799.76
위대한 쇼맨	762.83
뽀로로 극장판 공룡섬 대모험	715.68
너의 췌장을 먹고 싶어	694.91
원더	667.68
1987	666.97
신과함께-죄와 벌	649.34
강철비	607.05

- 영화 이름으로 인덱스를 잡고 점수의 합산을 구한다
- 100일 간 네이버 영화 평점 합산 기준 베스트 10을 뽑을 수 있다

In [45]:

```
1 tmp = movie.query('name == ["1987"]')
2 tmp
```

Out[45]:

		date	name	point
1253	2017-12-27	1987	9.15	
1297	2017-12-28	1987	9.20	
1345	2017-12-29	1987	9.19	
1393	2017-12-30	1987	9.21	
1440	2017-12-31	1987	9.23	
...
3943	2018-03-05	1987	9.26	
3993	2018-03-06	1987	9.26	
4043	2018-03-07	1987	9.26	
4093	2018-03-08	1987	9.26	
4192	2018-03-10	1987	9.26	

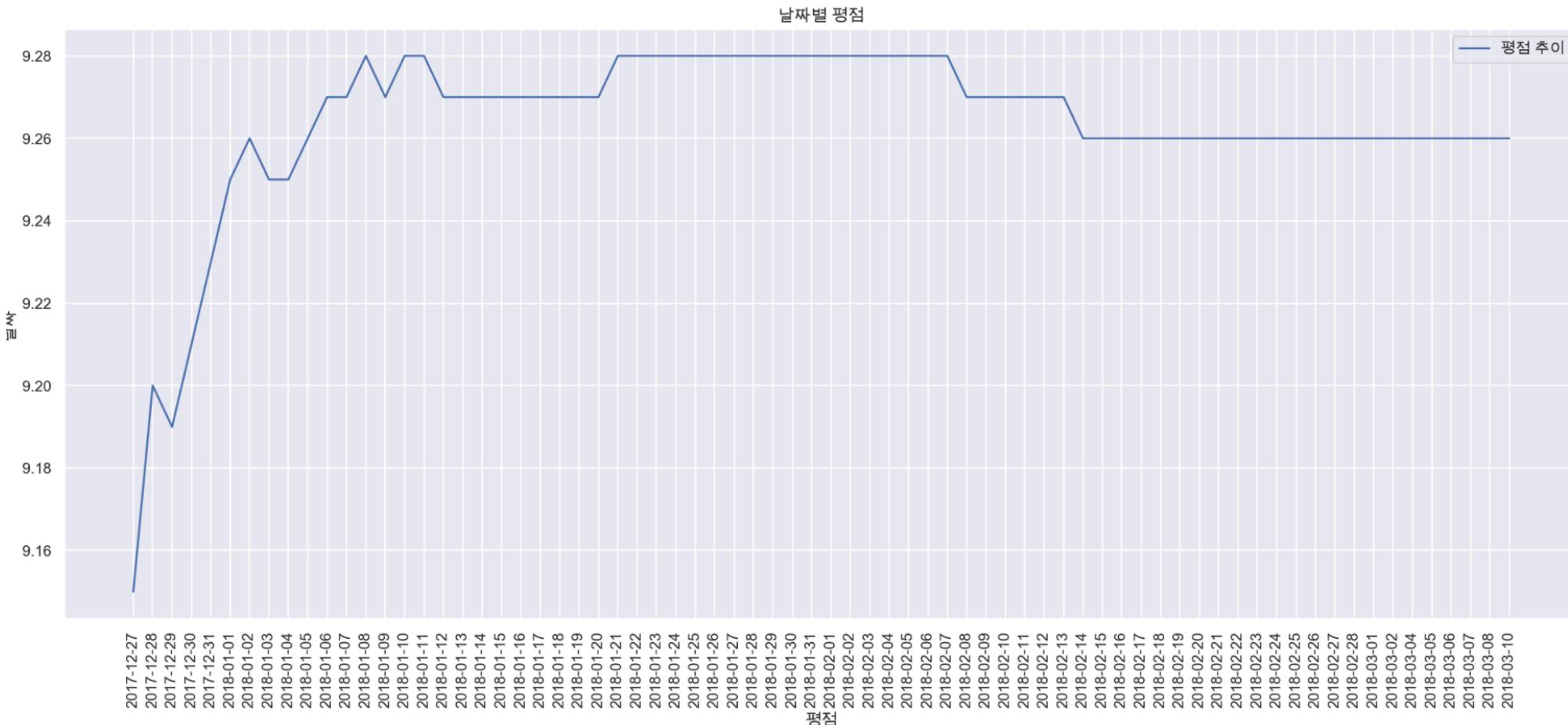
72 rows × 3 columns

- DataFrame의 검색 명령으로 query 명령도 있다

여기 페이지 문의사항!

In [49]:

```
1 import matplotlib.pyplot as plt
2 from matplotlib import rc
3
4 rc("font", family="Arial Unicode MS") # Windows: Malgun Gothic
5
6 #%matplotlib inline
7 get_ipython().run_line_magic("matplotlib", "inline")
8
9 plt.figure(figsize=(20, 8))
10 plt.plot(tmp["date"], tmp["point"])
11 plt.title("날짜별 평점")
12 plt.xlabel("평점")
13 plt.ylabel("날짜")
14 plt.xticks(rotation="vertical")
15 plt.legend(labels=["평점 추이"], loc="best")
16 plt.grid(True)
17 plt.show()
```



```
In [28]: 1 movie_best.head(10)
```

Out[28]:

point

name

러빙 빈센트 919.11

내 사랑 898.76

라라랜드 799.76

위대한 쇼맨 762.83

뽀로로 극장판 공룡섬 대모험 715.68

너의 췌장을 먹고 싶어 694.91

원더 667.68

1987 666.97

신과함께-죄와 벌 649.34

강철비 607.05

```
In [62]: 1 movie_best.tail(10)
```

Out[62]:

point	name
8.33	마야
8.27	메이즈 러너
8.24	언어의 정원
8.08	박열
8.07	더 테이블
7.99	마더
7.60	더 그레이
6.94	매혹당한 사람들
6.83	노 게임 노 라이프 -제로-
5.07	미옥

```
In [63]: 1 movie_pivot = pd.pivot_table(movie, index=["date"], columns=["name"], values=["point"])
2 movie_pivot.head()
```

Out[63]:

name	point																			
	10분 저스	12 솔 제스	1987	1급 기밀	50가지 그림자: 해방	7년-그들 이 없는 언론	7호 실	B급 머느리	가위 손	가장 따 뜻한 색, 블루	...	하이큐!! 승자와 패자	하이큐!! 재능과 센스	해피 데스데 이	행복 을 찾 아서	행복한 사전	환상 의 빛	축집사 : 북 오브 더 아 틀란틱	흥부 클로 세상을 바 꾼자	흥부 히트
date																				
2017-12-01	8.89	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	8.37	8.9	8.67	NaN	NaN	NaN	9.11
2017-12-02	8.89	NaN	NaN	NaN	NaN	NaN	6.96	NaN	NaN	NaN	...	NaN	NaN	8.36	8.9	8.67	NaN	NaN	NaN	9.11
2017-12-03	NaN	NaN	NaN	NaN	NaN	NaN	6.91	NaN	NaN	NaN	...	NaN	NaN	8.36	8.9	8.67	NaN	NaN	NaN	9.11
2017-12-04	NaN	NaN	NaN	NaN	NaN	NaN	6.88	NaN	9.25	NaN	...	NaN	NaN	8.36	8.9	8.67	NaN	NaN	NaN	9.11
2017-12-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	8.35	8.9	8.67	NaN	NaN	NaN	9.11

5 rows x 207 columns

- 100일 간 영화를 모두 정리하자

```
In [64]: 1 movie_pivot.to_excel("../data/04_movie_pivot.xlsx")
```

```
In [65]: 1 movie_pivot.columns = movie_pivot.columns.droplevel()
```

```
In [66]: 1 movie_pivot.head()
```

Out[66]:

	name	10분 저스	12 솔 1987	1급 기밀	50가지 그림자: 해방	7년-그들 이 없는 언론	7호 실	B급 며느리	가위 손	가장 따 뜻한 색, 블루	...	하이큐!! 승자와 패자	하이큐!! 재능과 센스	해피 데스데 이	행복 을 찾 아서	행복한 사전	환상 의 빛	흑집사 : 북 오브 더 아 틀란틱	흥부 총부	흥부 글로 세상을 바 꾼 자	흥부 글로 세상을 바 히트
	date																				
2017-12-01	8.89	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	8.37	8.9	8.67	NaN	NaN	NaN	9.11	
2017-12-02	8.89	NaN	NaN	NaN	NaN	NaN	6.96	NaN	NaN	NaN	...	NaN	NaN	8.36	8.9	8.67	NaN	NaN	NaN	9.11	
2017-12-03	NaN	NaN	NaN	NaN	NaN	NaN	6.91	NaN	NaN	NaN	...	NaN	NaN	8.36	8.9	8.67	NaN	NaN	NaN	9.11	
2017-12-04	NaN	NaN	NaN	NaN	NaN	NaN	6.88	NaN	9.25	NaN	...	NaN	NaN	8.36	8.9	8.67	NaN	NaN	NaN	9.11	
2017-12-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	8.35	8.9	8.67	NaN	NaN	NaN	9.11	

5 rows × 207 columns

- 엑셀로도 저장

A1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ
1	name	0.0MHz	919 유관습4월 이야기 가벼나움	가위손	따뜻한 색, 감시자들	걸캅스	고양이 춤	를 빌려드는 킹 오브	공작	꿈포의 묘지	교회오빠	가부도의	그녀	그래비티	그린 복	#카드: 타이로카봇 : 백!	봇: 음파로	극한직업	기묘한 가족	기생충	김광석	나니엘 블래의 산티아고	특별한 힐극의 웨프	내 사랑	깨워 고양#안의 그늘	을 위한 서의 이름은 현과	8								
2	date																																		
4	2019-03-01			9.6															9.63		8.78	8.66	5.68	8.72											
5	2019-03-02			9.6															9.63		8.78	8.66	5.7	8.72							8.29				
6	2019-03-03			9.6															9.63		8.78	8.66	5.74	8.72							8.29				
7	2019-03-04			9.6															9.63		8.78	8.66	5.75	8.72							8.29				
8	2019-03-05			9.6															9.63		8.78	8.66	5.75	8.72							8.29				
9	2019-03-06			9.6															9.63		8.78	8.66		8.72							8.28				
10	2019-03-07			9.6															9.63		8.77	8.66													
11	2019-03-08			9.6															9.63		8.77	8.66													
12	2019-03-09			9.61															9.63		8.77	8.66													
13	2019-03-10			9.61															9.63		8.77	8.65													
14	2019-03-11			9.61															9.63		8.77	8.65													
15	2019-03-12			9.61															9.62		8.77	8.65													
16	2019-03-13			9.61															9.63		8.77	8.65													
17	2019-03-14			9.61															9.63		8.77	8.63													
18	2019-03-15			9.61															9.63		8.77	8.62													
19	2019-03-16			9.6															9.62		8.77	8.61													
20	2019-03-17			9.6															9.63		8.77	8.6													
21	2019-03-18			9.6															9.63		8.77	8.59													
22	2019-03-19			9.6															9.62		8.77	8.58													
23	2019-03-20			9.6															9.62		8.77	8.58													
24	2019-03-21			9.59															9.62		8.77	8.58													
25	2019-03-22			9.59		8.78	7.76												9.62		8.77	8.57									9.17				
26	2019-03-23			9.59		8.78	7.76												9.62		8.77	8.57									9.17				
27	2019-03-24			9.6		8.78	7.76												9.62		8.77	8.56									9.17				
28	2019-03-25			9.6		8.78													9.62		8.77	8.56									9.17				
29	2019-03-26			9.6		8.78													9.62		8.77	8.56									9.17				
30	2019-03-27			9.59		8.78													9.62		8.77	8.56									9.17				
31	2019-03-28			9.59		8.78													9.62		8.77	8.56									9.17				
32	2019-03-29			9.59		8.78													9.62		8.77	8.56									9.17				
33	2019-03-30			9.6		8.78													9.62		8.77	8.55									9.17				
34	2019-03-31			9.59		8.78													9.62		8.77	8.55									9.17				
35	2019-04-01			9.59		8.78													9.62	9.16	8.77	8.55									9.17				
36	2019-04-02			9.59		8.78													9.62	9.16	8.77	8.55									9.17				
37	2019-04-03			9.59		8.78													9.62	9.16	8.77	8.55									9.17				
38	2019-04-04			9.59		8.78													9.62		8.77	8.55									9.17				
39	2019-04-05	8.58		9.59		8.78													9.62		9.24	8.77	8.55								9.17				
40	2019-04-06	8.58		9.6		8.78													9.62		9.24	8.77	8.55								9.17				
41	2019-04-07	8.59		9.6		8.78													9.62		9.24	8.77	8.55								9.17				
42	2019-04-08	8.58		9.6		8.78													9.62	9.16	9.24	8.77	8.55								9.17				
43	2019-04-09	8.57		9.6		8.78													9.62	9.16	8.77	8.55									9.17				
44	2019-04-10	8.57		9.6		8.78													8.16		9.62	9.16	8.77	8.54							9.17				
45	2019-04-11	8.57		9.6		8.78															9.62	9.16	8.77	8.54								9.17			

```
In [72]: 1 import platform
2 from matplotlib import font_manager, rc
3 import seaborn as sns
4
5 path = "c:/Windows/Fonts/malgun.ttf"
6 if platform.system() == "Darwin":
7     rc("font", family="Arial Unicode MS")
8 elif platform.system() == "Windows":
9     font_name = font_manager.FontProperties(fname=path).get_name()
10    rc("font", family=font_name)
11 else:
12     print("Unknown system... sorry~~~~")
```

- matplotlib 한글 설정

```
In [73]: 1 target_col = ["신과함께-죄와 벌", "1987", "너의 체장을 먹고 싶어", "공범자들", "범죄도시"]
2 plt.figure(figsize=(20, 6))
3 plt.title("날짜별 영화 평점")
4 plt.xlabel("날짜")
5 plt.ylabel("평점")
6 plt.plot(movie_pivot[target_col])
7 plt.legend(target_col, loc="best")
8 plt.xticks(rotation="vertical")
9 plt.grid(True)
10 plt.tick_params(bottom="off", labelbottom="off")
11 plt.show()
```

- 보고 싶은 영화 몇 개만 추려서 그래프로 확인해보자

