

The 50 Best Sandwiches in Chicago

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

PUBLISHED OCT. 9, 2012



19 COMMENTS

Beautiful Soup을 통해 익히는

웹 스크래핑의 기초

BLT at Old Oak Tap, the No. 1 sandwich in Chicago. PHOTOGRAPH: ANNA KNOTT; FOOD STYLIST: LISA KUEHL



네이버 영화 평점

영화홈

상영작·예정작

영화랭킹

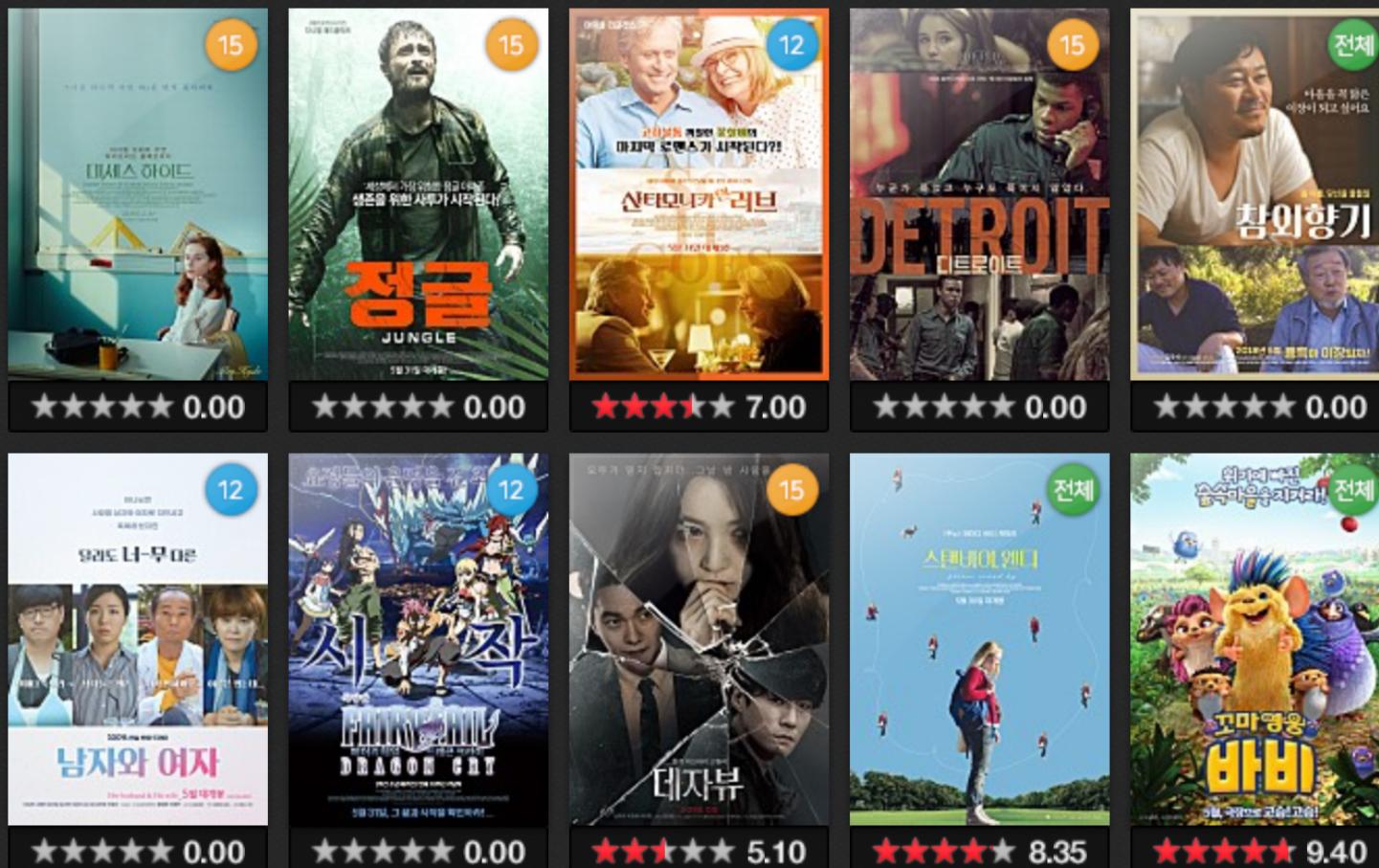
예매

평점·리뷰

다운로드 □

인디극장

up



예매순

현재상영작

개봉예정작

평점순

박스오피스

다운로드순

전체보기 ▶

NAVER 영화

영화홈

상영작 · 예정작

영화랭킹

▶ 랭킹

▶ 디렉토리

예매

평점 · 리뷰

다운로드

인디극장 up ▶

로그인

영화검색

검색

랭킹

영화 ▾ | 영화인 | 예매 | 박스오피스

영화 랭킹

조회순	평점순 (현재상영영화) ▾	평점순 (모든영화)	2018.05.29
순위	영화명	평점	변동폭
1	당갈	★★★★★ 9.62	평점주기 - 0
2	위대한 쇼맨	★★★★★ 9.37	평점주기 - 0
3	킹 오브 프리즘 프라이드 더 히어로	★★★★★ 9.26	평점주기 - 0
4	러빙 빈센트	★★★★★ 9.19	평점주기 - 0
5	피터 래빗	★★★★★ 9.13	평점주기 - 0

영화 인기검색어 ▶ 더보기

1 독전 - 0

2 데드풀 2 - 0

3 버닝 - 0

4 트루스 오어 데어 - 0

5 한 솔로: 스타워즈 .. ↑ 1

2018.05.29

영화인 인기검색어 ▶ 더보기

<https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20180529>

- 웹 페이지의 주소에는 많은 정보가 담겨있어서
- 원하는 정보를 얻기 위해 변화시켜야하는 주소의 규칙이 보이기도 한다
- 이 경우 날짜 정보를 변경해주면 해당 페이지에 그냥 접근이 가능하다

In [2]:

```
from bs4 import BeautifulSoup
import pandas as pd
from urllib.request import urlopen

url = "http://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20180501"
page = urlopen(url)

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

<link href="/common/css/old_default.css?20180418140529" rel="stylesheet" type="text/cs
s"/>
<link href="/common/css/old_layout.css?20180418140529" rel="stylesheet" type="text/cs
s"/>
```

- 일단 한 페이지만 먼저 접근해 보자

| 영화 랭킹

조회순	평점순 (현재상영영화) ▾	평점순 (모든영화)	2018.05.01	<	>
-----	----------------	------------	------------	---	---

순위	영화명	평점	변동폭
1	당갈	★★★★★ 9.53	평점주기 - 0
2	덕구	★★★★★ 9.50	평점주기 - 0
3	원더	★★★★★ 9.40	평점주기 - 0
4	위대한 쇼맨	★★★★★ 9.38	평점주기 - 0
5	지금, 만나러 갑니다	★★★★★ 9.33	평점주기 - 0
6	킹 오브 프리즘 프라이드 더 히어로	★★★★★ 9.31	평점주기 - 0
7	나, 다니엘 블레이크	★★★★★ 9.23	평점주기 ↑ 1
8	우리들	★★★★★ 9.21	평점주기 ↑ 1
9	내 사랑	★★★★★ 9.18	평점주기 ↑ 2
10	빠삐용	★★★★★ 9.18	평점주기 ↑ 2

- 여기서 영화 제목과 평점을 가져오고 싶다

랭킹

영화 | 영화인 | 예매 | 박스오피스

영화 랭킹

순위	영화명	평점
1	당갈	9.53
2	덕구	9.50
3	원더	9.40
4	위대한 쇼맨	9.38

```

<tbody>
  <tr>...</tr>
  <!-- 예제
        <tr>
          <td class="ac"></td>
          <td class="title"><a href="#">트랜스포머
</a></td>
          <td class="ac"></td>
          <td class="range ac">7</td>
        </tr>
        -->
  <tr>
    <td class="ac">...</td>
    <td class="title">
      <div class="tit5"> == $0
        <a href="/movie/bi/mi/basic.nhn?code=157243" title="당
갈">당갈</a>
      </div>
    </td>
    <!-- 평점순일 때 평점 추가하기 -->
    <td>...</td>
    <td class="point">9.53</td>
  </tr>
</tbody>

```

- 크롬 개발자 도구로 확인해 본 결과, div 태그에 tit5 클래스를 확인하면 제목

In [3]:

```
soup.find_all('div', 'tit5')
```

```
[<div class="tit5">
<a href="/movie/bi/mi/basic.nhn?code=157243" title="당갈">당갈</a>
</div>, <div class="tit5">
<a href="/movie/bi/mi/basic.nhn?code=154667" title="덕구">덕구</a>
</div>, <div class="tit5">
<a href="/movie/bi/mi/basic.nhn?code=151196" title="원더">원더</a>
</div>, <div class="tit5">
```

- **find_all** 명령으로 쉽게 접근했다

```
In [4]:
```

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

```
<a href="/movie/bi/mi/basic.nhn?code=157243" title="당갈">당갈</a>
```

```
In [5]:
```

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

```
'당갈'
```

- **OK** 이렇게 접근하면 영화 제목을 알 수 있다

In [5]:

```
movie_name = [each.a.string for each in soup.find_all('div', 'tit5')]  
movie_name
```

```
[ '당갈' ,  
  '덕구' ,  
  '원더' ,  
  '위대한 쇼맨' ,  
  '지금, 만나러 갑니다' ,
```

- 이렇게 간단하게 접근하면 웹페이지 하나에 대해 제목을 얻을 수 있다

The screenshot shows a movie ranking page with two main entries. The first entry is for the movie '당갈' (Dang Gal) at the top of the list. The second entry is for the movie '덕구' (Deokgu) in second place. Both entries show a 5-star rating system. The developer tools are used to inspect the HTML code for the first movie's rating, specifically focusing on the 'td.point' element which contains the value '9.53'. The code also includes logic for adding points when the ranking order changes.

```

<td class="ac"></td>
<td class="range ac">7</td>
</tr>
-->
<tr>
  <td class="ac">...</td>
  <td class="title">
    <div class="tit5">
      <a href="/movie/bi/mi/basic.nhn?code=157243" title="당
      갈">당갈</a>
    </div>
  </td>
  <!-- 평점순일 때 평점 추가하기 -->
  <td>...</td>
  <td class="point">9.53</td> == $0
  <td class="ac">...</td>
  <!--
  <td class="ac">...</td>
  <td class="range ac">0</td>
  </tr>

```

- **td** 태그에 **point** 클래스를 확인하면 영화 평점을 얻을 수 있다

In [6]:

```
soup.find_all('td', 'point')
```

```
[<td class="point">9.53</td>,
 <td class="point">9.50</td>,
 <td class="point">9.40</td>,
 <td class="point">9.38</td>,
 <td class="point">9.33</td>,
 <td class="point">9.31</td>,
 <td class="point">9.23</td>,
 <td class="point">9.21</td>,
```

- 이제 이 정도는 쉽게 얻을 수 있다. (단, 2~3시간만에.. 우와~)

```
In [7]: soup.find_all('td', 'point')[0].string
```

```
'9.53'
```

- **OK** 반복문이 등장할 차례

```
In [8]: movie_point = [each.string for each in soup.find_all('td', 'point')]
movie_point
```

```
[ '9.53',
  '9.50',
  '9.40',
  '9.38',
  '9.33',
```

- 이쁘게 잘 나왔다

<https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20180529>

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

```
In [10]:
```

```
date = pd.date_range('2018.07.01', periods=35, freq='D')  
date
```

```
DatetimeIndex(['2018-07-01', '2018-07-02', '2018-07-03', '2018-07-04',  
                '2018-07-05', '2018-07-06', '2018-07-07', '2018-07-08',  
                '2018-07-09', '2018-07-10', '2018-07-11', '2018-07-12',  
                '2018-07-13', '2018-07-14', '2018-07-15', '2018-07-16',  
                '2018-07-17', '2018-07-18', '2018-07-19', '2018-07-20',  
                '2018-07-21', '2018-07-22', '2018-07-23', '2018-07-24',  
                '2018-07-25', '2018-07-26', '2018-07-27', '2018-07-28',  
                '2018-07-29', '2018-07-30', '2018-07-31', '2018-08-01',  
                '2018-08-02', '2018-08-03', '2018-08-04'],  
               dtype='datetime64[ns]', freq='D')
```

- pandas의 `date_range`를 이용하면, 손쉽게 날짜를 만들 수 있다.

```
In [11]:
```

```
date[0]
```

```
Timestamp('2018-07-01 00:00:00', freq='D')
```

- 날짜형 데이터는 이렇게 생겼다.

```
In [12]:
```

```
date[0].strftime( '%y-%m-%d' )
```

```
'18-07-01'
```

```
In [13]:
```

```
date[0].strftime( '%y. %m. %d' )
```

```
'18. 07. 01'
```

- 이렇게 날짜형 데이터는 **strftime** 명령으로 원하는 형태로 문자열로 잡을 수 있다.

In [15]:

```
date[0].strftime( '%y%m%d' )
```

'180701'

In [16]:

```
date[0].strftime( '%Y%m%d' )
```

'20180701'

- 네이버 영화 사이트에서 필요한 형태는 마지막형태이다

In [25]:

```
import time

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

- 얻고 싶은 데이터를 저장할 빈 리스트를 만들어 두자

In [18]:

```
for today in 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")  
  
    movie_date.extend([today]*len(soup.find_all('td', 'point')))  
    movie_name.extend([each.a.string for each in soup.find_all('div', 'tit5')])  
    movie_point.extend([each.string for each in soup.find_all('td', 'point')])  
  
    print(str(today))  
    time.sleep(0.5)
```

- 이제 이정도 코드를 이해하기 위해 노력하자

In [19]:

```
len(movie_date), len(movie_name), len(movie_point)
```

```
(1650, 1650, 1650)
```

- 방금 꽤 많은 정보를 얻었고, 그 크기가 문제 없음을 알았다

```
In [28]:
```

```
movie = pd.DataFrame({ 'date':movie_date, 'name':movie_name, 'point':movie_point})  
movie.head( )
```

	date	name	point
0	2018-05-01	당갈	9.53
1	2018-05-01	덕구	9.50
2	2018-05-01	원더	9.40
3	2018-05-01	위대한 쇼맨	9.38
4	2018-05-01	지금, 만나러 갑니다	9.33

- 역시.. 잘 만들어 졌다

In [29]:

```
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1346 entries, 0 to 1345
Data columns (total 3 columns):
date      1346 non-null datetime64[ns]
name      1346 non-null object
point     1346 non-null object
dtypes: datetime64[ns](1), object(2)
memory usage: 31.6+ KB
```

- 평점은 숫자형(**float**)으로 만들자

```
In [30]: movie['point'] = movie['point'].astype(float)  
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1346 entries, 0 to 1345  
Data columns (total 3 columns):  
date      1346 non-null datetime64[ns]  
name       1346 non-null object  
point      1346 non-null float64  
dtypes: datetime64[ns](1), float64(1), object(1)  
memory usage: 31.6+ KB
```

- OK~~

```
In [31]: movie.to_csv("../data/04_naver_movie_raw_data.csv", sep=",", encoding="utf-8")
```

- 당연히 정신건강을 위한 저장~~~~

저장해둔 네이버 영화 평점 데이터를 다듬자

```
In [1]:
```

```
import numpy as np
import pandas as pd

movie = pd.read_csv("../data/04_naver_movie_raw_data.csv", index_col=0)
movie.head()
```

	date	name	point
0	2018-05-01	당갈	9.53
1	2018-05-01	덕구	9.50
2	2018-05-01	원더	9.40
3	2018-05-01	위대한 쇼맨	9.38
4	2018-05-01	지금, 만나러 갑니다	9.33

```
In [2]: movie_unique = pd.pivot_table(movie, index=['name'], aggfunc=np.sum)  
movie_unique.head()
```

	point
name	
12 솔져스	33.30
4등	42.65
B급 며느리	33.64
곤지암	196.71
그 후	123.35

- 영화 이름으로 인덱스를 잡고 점수의 합산을 해볼 수 있다

In [3]:

```
movie_best = movie_unique.sort_values(by='point', ascending=False)
movie_best.head(10)
```

	point
	name
당갈	288.49
위대한 쇼맨	281.29
킹 오브 프리즘 프라이드 더 히어로	278.44
박하사탕	272.23
비밥바룰라	271.04
그날, 바다	269.97
어벤져스: 인피니티 워	269.34
소공녀	267.51
안녕, 나의 소울메이트	265.08
레디 플레이어 원	258.11

- 한달간 네이버 영화 평점 합산 기준 베스트10을 뽑을 수 있다

In [4]:

```
movie.query('name == ["소공녀"]')
```

		date	name	point
15		2018-05-01	소공녀	8.95
65		2018-05-02	소공녀	8.94
115		2018-05-03	소공녀	8.94
165		2018-05-04	소공녀	8.94
215		2018-05-05	소공녀	8.93
264		2018-05-06	소공녀	8.93
308		2018-05-07	소공녀	8.93

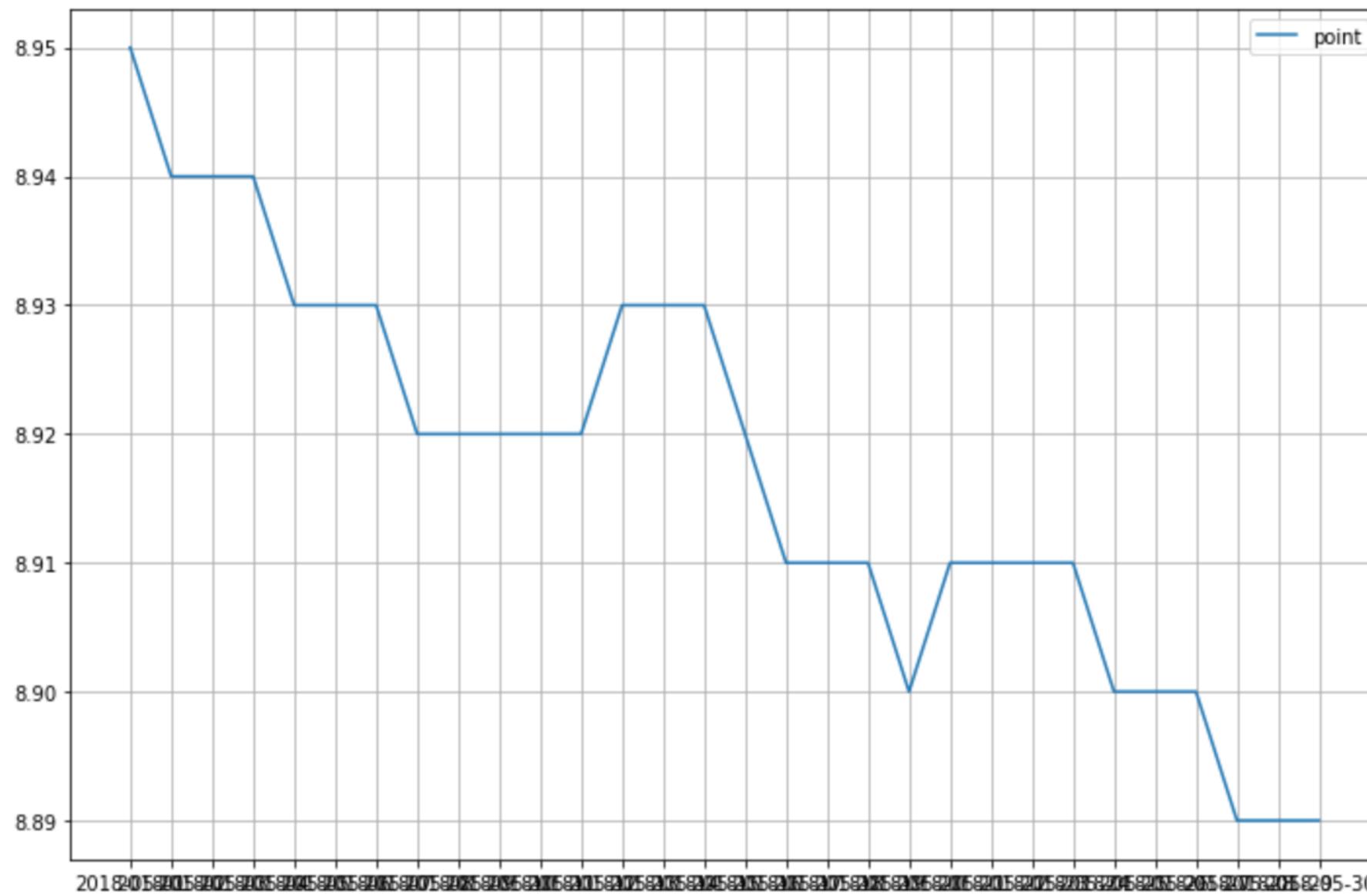
- **DataFrame**의 검색 명령으로 **query** 명령도 있다

In [5]:

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(12,8))
plt.plot(movie.query('name == ["소공녀"]')[['date'],
                                             movie.query('name == ["소공녀"]')[['point']])
plt.legend(loc='best')
plt.grid()
plt.show()
```

- 날짜별 점수를 한 영화에 대해 그려볼 수 있다.



In [6]:

```
movie_best.head(10)
```

	point
	name
당갈	288.49
위대한 쇼맨	281.29
킹 오브 프리즘 프라이드 더 히어로	278.44
박하사탕	272.23
비밥바를라	271.04
그날, 바다	269.97
어벤져스: 인피니티 워	269.34
소공녀	267.51
안녕, 나의 소울메이트	265.08
레디 플레이어 원	258.11

In [7]:

```
movie_best.tail(10)
```

	point
	name
자전거 탄 소년	42.90
4등	42.65
두 번의 결혼식과 한 번의 장례식	40.40
마징가 Z: 인피니티	40.02
B급 며느리	33.64
12 솔져스	33.30
서서평, 천천히 평온하게	27.66
샤인	27.25
사라진 밤	14.05
델마와 루이스	8.92

In [31]:

```
movie_pivot = movie.pivot('date','name','point')
movie_pivot.head()
```

name	500 일의 썸머	개들의 섬	건축학개론	걸어도 걸어도	게이트	경성 학교: 사라진 소녀들	그렇게 아버지가 된다	극장판 포켓몬스터 DP - 디아루가 VS 펄기아 VS 블랙 앤드 웨이트 크라이	극장판 헬로카봇: 백악기 시대	나라타주	...	파리로 가는 길	파수꾼	판의 미로 - 오필리아와 세 개의 열쇠	팬텀스 레드	피아니스트	피터 래빗	한여름의 판타지아	호텔아르테미스	허스토리	환상의 빛
date																					
2018-07-01	NaN	9.03	NaN	8.66	3.75	NaN	8.83	7.99	NaN	NaN	...	7.91	NaN	NaN	8.44	NaN	8.97	NaN	9.55	NaN	8.01
2018-07-02	NaN	9.04	NaN	8.66	3.75	NaN	8.83	8.00	NaN	NaN	...	7.91	NaN	NaN	8.41	NaN	8.97	NaN	9.52	NaN	8.01
2018-07-04	NaN	8.98	NaN	8.66	NaN	NaN	8.83	7.99	NaN	NaN	...	7.91	NaN	NaN	8.41	NaN	8.97	NaN	9.49	NaN	8.01

- 한 달 간 영화를 모두 정리하자

```
In [13]: movie_pivot.to_excel("../data/04_movie_pivot.xlsx")
```

- 이번에는 엑셀로 저장하자

A1

fx

date

1	A	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		
2	date	00일의 썬기	개들의 삶	건축학개론	어도 걸어	케이트	교: 사라진	아버지기	디아루가	로카봇 : 백	나라타주	의 산티아	키스트	아지	세상의	당갈	댄서	더 포스트	덕구	데드풀 2	독전	동주	레옹	레이	레이디버그	틀 포레스	마녀	방을 나온	마지막	황제드나잇	선	미션	임파서블: 흐
3	2018-07-01		9.03		8.66	3.75		8.83	7.99			8.13				9.6	9.11			8.77	7.75						8.19	9.23	8.95				
4	2018-07-02		9.04		8.66	3.75		8.83	8			8.13					9.11			8.77	7.74						6	8.15	9.23	8.94			
5	2018-07-03		9.01		8.66	3.75		8.83	7.99			8.13				9.6			8.76	7.74						6	8.12	9.23	8.93				
6	2018-07-04		8.98		8.66			8.83	7.99			8.13							8.76	7.73						6	8.1	9.23	8.9				
7	2018-07-05		8.97		8.66			8.83	7.99			8.13						8.76	7.73						6	8.1	9.23	8.9					
8	2018-07-06		8.96		8.66	3.75		8.83	7.99			8.13						8.76	7.72						6	8.11	9.23	8.9					
9	2018-07-07		8.95		8.65	3.75		8.83	7.99			8.13						8.76	7.72						6	8.13	9.23	8.91					
10	2018-07-08		8.91		8.65	3.75		8.83	7.99			8.13						8.76	7.71						6	8.15	9.23	8.92					
11	2018-07-09		8.89		8.64	3.75		8.83	7.98			8.13						8.76	7.7						6	8.16	9.23	8.91					
12	2018-07-10		8.8		8.64		6.12	8.83	7.98			8.13						8.76	7.7						6	8.17		8.9					
13	2018-07-11		8.79		8.64	6.12	8.83	7.99				8.13						8.76	7.7						6	8.17		8.91					
14	2018-07-12		8.79		8.64	6.12	8.83				8.13						8.76							6	8.18		8.91						
15	2018-07-13		8.76		8.64	6.12	8.83			6.48	8.13			9.58		8.73		8.76							6	8.18	8.88	8.9					
16	2018-07-14		8.75		8.64	6.12	8.83			6.48	8.13			9.57		8.73		8.76							6	8.19	8.88	8.9					
17	2018-07-15		8.73		8.64	6.12	8.83			6.48	8.13	9.13		9.57		8.73		8.76							6	8.21	8.88	8.9					
18	2018-07-16		8.72		8.64	6.12	8.83	7.99		6.48	8.13			9.57		8.73		8.76							6	8.2	8.88	8.9					
19	2018-07-17		8.72		8.64	6.12	8.83			6.48	8.13			9.57		8.73			7.67							6	8.21	8.88	8.89				
20	2018-07-18		8.71		8.65	6.12	8.83			6.48	8.13			9.57		8.73			7.67							8.21	8.88	8.89					
21	2018-07-19		8.71		8.65	6.12	8.83			6.48	8.13			9.56		8.73	9.48	7.66	9.37						8.74	8.2	8.88	8.9					
22	2018-07-20		8.7		8.65	6.12	8.83			6.48	8.13			9.56		8.73	9.48	7.66	9.37						8.74	8.2	8.88	8.89					
23	2018-07-21		8.69		8.65	6.12	8.83			6.48	8.13			9.56					7.65	9.37						8.74	8.2	8.88	8.9				
24	2018-07-22		8.66		8.65	6.12	8.83			6.48	8.13			9.55					7.64	9.37						8.74	8.21	8.88	8.9				
25	2018-07-23		8.65		8.65	6.12	8.83			6.48	8.13			9.55					7.64	9.37	9.03					8.74	8.2	8.88	8.9				
26	2018-07-24		8.64		8.65	6.12	8.83			6.48	8.13			9.55					7.63	9.37	9.03					8.74	8.2	8.88	8.9				
27	2018-07-25		8.63		8.65	6.12	8.83			6.48	8.13			9.55					7.63	9.37	9.03					8.74	8.21		8.88	9.22			
28	2018-07-26		8.63		8.65	3.74	6.12	8.83			6.48	8.13			9.56					7.63	9.37						8.74	8.21		8.89	9.23		
29	2018-07-27		8.64		8.65	8.65			8.83			8.13		7.45	9.56				7.63	9.35	9.37					8.74	8.21		8.88	9.19			
30	2018-07-28	8.38	8.63	8.65	8.65				8.83			8.13		7.45	9.56				7.63	9.35	9.37					8.74	8.21		8.88		9.17		
31	2018-07-29	8.38	8.63	8.65	8.65				8.83			8.13		7.45	9.56				7.62	9.35	9.37					8.74	8.22		8.88		9.17		
32	2018-07-30	8.38	8.61	8.65	8.65				8.83			8.13		7.45	9.55				7.62	9.35	9.37					8.74	8.22		8.86		9.16		
33	2018-08-01	8.38	8.6	8.65	8.65				8.83		9.29		8.13		7.45	9.55				7.62	9.35	9.37	9.03				8.74	8.21		8.86		9.16	
34	2018-08-02	8.38	8.6	8.65	8.65				8.83		9.3		8.13		7.45	9.55				7.61	9.35	9.37	9.03				8.74	8.22		8.84		9.15	
35	2018-08-03	8.38	8.6	8.65	8.65				8.83			8.13			9.55					7.61	9.35	9.37	9.03				8.74	8.22		8.84		9.15	
36	2018-08-04	8.38	8.6	8.65	8.65				8.83		9.27		8.13		7.45	9.55				7.61	9.35	9.37	9.03				8.74	8.22		8.83		9.16	

In [33]:

```
movie_pivot.columns
```

Index(['500일의 썸머', '개들의 섬', '건축학개론', '걸어도 걸어도', '게이트', '경성학교: 사라진 소녀들', '그렇게 아버지가 된다', '극장판 포켓몬스터DP - 디아루가 VS 펄기아 VS 다크라이', '극장판 헬로카봇 : 백 악기 시대', '나라타주', '나의 산티아고', '다키스트 아워', '단지 세상의 끝', '당갈', '댄서', '더 포스트', '덕 구', '데드풀 2', '독전', '동주', '레옹', '레이', '레이디버그', '리틀 포레스트', '마녀', '마당을 나온 암탉', '마지막 황제', '미드나잇 선', '미션', '미션 임파서블: 폴아웃', '바그다드 카페 : 디렉터스컷', '바닷마을 다이어리', '바람과 함께 사라지다', '버닝', '벤허', '변산', '부르고뉴, 와인에서 찾은 인생', '블레이드 2', '사랑과 영혼', '색, 계', '서서평, 천천히 평온하게', '세 번째 살인', '셰이프 오브 워터: 사랑의 모양', '소공녀', '속닥속닥', '스카이스크래퍼', '스탠바이, 웬디', '시카리오: 데이 오브 솔다도', '시카리오: 암살자의 도시', '신과함께-인과 연', '신비아파트: 금빛 도깨비와 비밀의 동굴', '싱 스트리트', '싸이코', '아무도 모른다', '아바타', '아이 엠 러브', '아이 필 프리티', '아일라', '안녕, 나의 소녀', '알로, 슈티', '앤틴맨', '앤틴맨과 와스프', '어거스트 : 가족의 초상', '어느 가족', '어벤져스: 인피니티 워', '여고괴담 두번째 이야기', '여중생A', '오늘 밤, 로맨스 극장에서', '오션스8', '원더', '유전', '이별의 아침에 약속의 꽃을 장식하자', '인랑', '인크레더블 2', '제로 다크 서티', '조용한 가족', '족구왕', '죽은 시인의 사회', '쥬라기 월드: 폴른 킹']

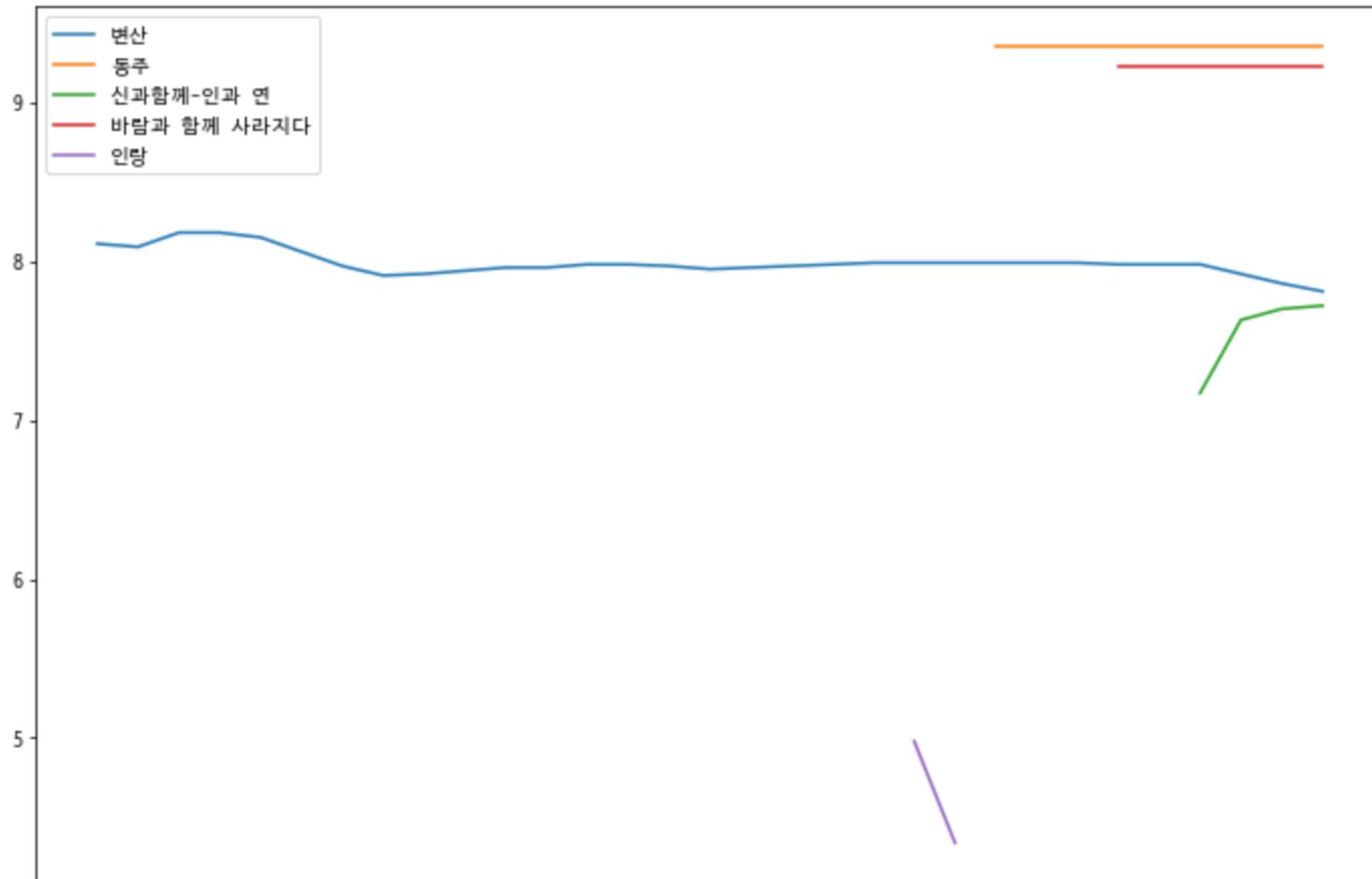
```
In [15]:  
from matplotlib import font_manager, rc  
plt.rcParams['axes.unicode_minus'] = False  
import seaborn as sns  
  
# f_path = "c:/Windows/Fonts/malgun.ttf"  
f_path = "/Users/pinkwink/Library/Fonts/D2Coding-Ver1.3-20171129.ttf"  
# f_path = "/Library/Fonts/AppleGothic.ttf"  
font_name = font_manager.FontProperties(fname=f_path).get_name()  
rc('font', family=font_name)
```

- 한글 설정~~~

In [35]:

```
target_col = ['변산', '동주', '신과함께-인과 연', '바람과 함께 사라지다', '인랑']
plt.figure(figsize=(12,8))
plt.plot(movie_pivot[target_col])
plt.legend(target_col, loc='best')
plt.tick_params(bottom=False, labelbottom=False)
plt.show()
```

- 보고 싶은 영화 몇 개만 추려서 그래프로 확인해보자



시카고 샌드위치 맛집 정보 추출



chicago magazine the 50 best sandwiches



전체

이미지

뉴스

동영상

지도

더보기

설정

도구

검색결과 약 8,330,000개 (0.65초)

The 50 Best Sandwiches in Chicago | Chicago magazine | November ...

www.chicagomag.com/Chicago-Magazine/.../Best-Sandwiches-Ch... ▾ 이 페이지 번역하기

2012. 10. 9. - Our list of Chicago's 50 best sandwiches, ranked in order of deliciousness.

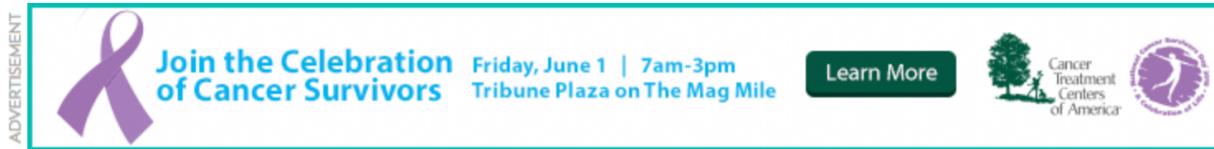
이 페이지를 여러 번 방문했습니다. 최근 방문 날짜: 17. 12. 2

Chicago Magazine's November Issue: 50 Best Sandwiches | Chicago ...

www.chicagomag.com/Chicago-Magazine/.../Chicago-Magazines-... ▾ 이 페이지 번역하기

2012. 10. 8. - Chicago magazine's November issue hits newsstands Thursday, ... 50 Best Sandwiches (cover story) - If you've been visiting Chicago's top ...

ADVERTISEMENT



 **Join the Celebration
of Cancer Survivors** Friday, June 1 | 7am-3pm
Tribune Plaza on The Mag Mile [Learn More](#)



The 50 Best Sandwiches in Chicago

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

PUBLISHED OCT. 9, 2012



19 COMMENTS



In our research, we learned that the sandwich is a wily chameleon, soaking up and synthesizing every trend, be it the resurgence of house-cured charcuterie or the sudden ubiquity of arugula. We learned to ask for extra napkins ahead of time. And we learned, above all, that quality and quantity can intersect in restaurants, and there's no shame in that. Only joy.

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](#)

→ [WHERE TO BUY NOW](#)

- How to Mend a Broken Heart
- How Jackie Spinner's Sons Inspired Her Documentary About Autism in Morocco

ADVERTISEMENT



2012 BEST SANDWICHES

1. Old Oak Tap BLT

"Truly inspired."

PUBLISHED OCT. 9, 2012



1 COMMENT





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

-
- 메인 페이지에는 시카고의 **50**개 맛집 샌드위치 가게에 대해
 - 메뉴와 가게 이름이 정리되어 있고
 - 각각을 소개한 **50**개의 페이지에 들어가보면
 - 가게 주소와 대표 메뉴의 가격이 있다
 - 총 **51**개 페이지에서 각 가게의
 - 가게이름
 - 대표메뉴
 - 대표메뉴의 가격
 - 가게 주소를 수집

한 페이지에서 접근해야 할 50개 URL을 찾자

In [18]:

```
from bs4 import BeautifulSoup
import pandas as pd
from urllib.request import urlopen
```

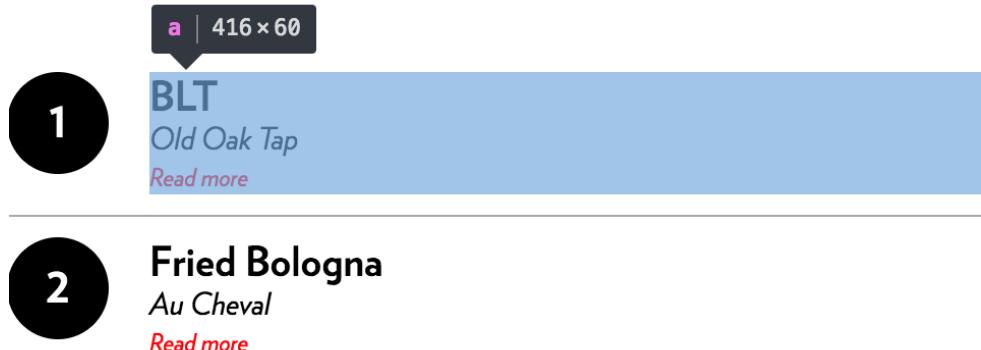
- 실습 파일을 따로 관리하고 있는 분들을 위해 필요 모듈을 다시 **import**

In [19]:

```
url_base = 'http://www.chicagomag.com'  
url_sub = '/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/'  
url = url_base + url_sub  
  
html = urlopen(url)  
soup = BeautifulSoup(html, "html.parser")  
  
soup
```

```
<!DOCTYPE doctype html>  
  
<html lang="en">  
<head>  
<!-- Urbis maqnitudo. Fabulas magnitudo. -->
```

In our research, we learned that the sandwich is a wily chameleon, soaking up and synthesizing every trend, be it the resurgence of house-cured charcuterie or the sudden ubiquity of arugula. We learned to ask for extra napkins ahead of time. And we learned, above all, that quality and quantity can intersect in restaurants, and there's no shame in that. Only joy.



```
Elements Console Sources Network Performance > ⚠ 9 | : X
"app_id=432224116825361&container_width=1&href=http%3A%2F%2Fwww.chic
agomag.com%2FChicago-Magazine%2FNovember-2012%2FBest-Sandwiches-
Chicago%2F&layout=box_count&locale=en_US&sdk=joey&send=false&share=t
rue&show_faces=false&width=50">...</div>
► <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/Best-Sandwiches-in-
      Chicago-Old-Oak-Tap-BLT/">...</a> = $0
    </div>
  </div>
  ...
  ► <div class="sammy" style="position: relative;">...</div>
  ► <div class="sammy" style="position: relative;">...</div>
```

- **div의 sammy 클래스가 눈에 보인다**

In [3]:

```
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-Sandwiches-in-C
hicago-Old-Oak-Tap-BLT/"><b>BLT</b><br>
Old Oak Tap<br>
```

- **div**의 **sammy** 클래스를 읽으면 될 것 같다

```
In [4]: len(soup.find_all('div', 'sammy'))
```

50

- OK~~

In [5]:

```
print(soup.find_all('div', 'sammy')[0])
```

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

- 한 항목을 보면 원하는 정보 중에서 랭킹, 가게이름, 메뉴가 보인다.

In [6]:

```
tmp_one = soup.find_all('div', 'sammy')[0]  
type(tmp_one)
```

bs4.element.Tag

- **type⁰이 bs4.element.Tag**라는 것은 **find** 명령을 사용할 수 있다는 뜻

In [7]:

```
tmp_one.find(class_='sammyRank')
```

```
<div class="sammyRank">1</div>
```

In [8]:

```
tmp_one.find(class_='sammyRank').get_text()
```

```
'1'
```

- **Randing 확보**

```
In [9]:
```

```
tmp_one.find(class_='sammyListing').get_text()
```

```
'BLT\r\nOld Oak Tap\nRead more '
```

```
In [10]:
```

```
tmp_one.find('a')['href']
```

```
'/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/'
```

- 가게 이름과 메뉴는 한번에 있다.ㅠㅠ.
- 연결되는 홈페이지 주소가 “상대경로”이다.

```
In [11]: import re

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

['BLT', 'Old Oak Tap', 'Read more ']
```

- 가게 이름과 메뉴는 `re` 모듈의 `split`으로 쉽게 구분할 수 있다.

```
In [12]:  
    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 있는 거지

In [15]:

```
print(url_base)
print(tmp_one.find('a')['href'])
```

http://www.chicagomag.com
/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/

- 처음 나눴던 **url_base**를 사용할 때가 되었다
- 어떤 주소는 상대주소로 어떤 주소는 절대주소로 나타난다

In [17]:

```
from urllib.parse import urljoin  
urljoin(url_base, tmp_one.find('a')['href'])
```

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

- 이럴때 **urljoin**을 이용하면
- **url_base**를 이용해서 상대주소든, 절대주소든 모두 절대주소로 변경해준다

In [26]:

```
tmp_one = soup.find_all('div', 'sammy')[10].find('a')['href']
tmp_one
```

'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Lul
a-Cafe-Ham-and-Raclette-Panino/'

In [27]:

```
urljoin(url_base, tmp_one)
```

'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Lul
a-Cafe-Ham-and-Raclette-Panino/'

In [28]:

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

for item in soup.find_all('div', 'sammy'):
    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']))
```

- 50개 메뉴에 다 적용하자...

```
In [29]:
```

```
main_menu[:5]
```

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

```
In [30]:
```

```
cafe_name[:5]
```

```
['Old Oak Tap', 'Au Cheval', 'Xoco', 'Al's Deli', 'Publican Quality Meats']
```

```
In [31]:
```

```
url_add[:5]
```

```
['http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Ol  
d-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-Xo  
co-Woodland-Mushroom/','  
 'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Al  
s-Deli-Roast-Beef/'],
```

```
In [32]: len(rank), len(main_menu), len(cafe_name), len(url_add)  
(50, 50, 50, 50)
```

In [33]:

```
import pandas as pd

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

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

In [34]:

```
df = pd.DataFrame(data, columns=[ 'Rank' , 'Cafe' , 'Menu' , 'URL' ] )  
df.head(5)
```

	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...

- 50개 자료에 대해 이름, 메뉴 또 더 접근해야할 URL까지 모두 정리되었다.

```
In [35]: df.to_csv('..../data/04_best_sandwiches_list_chicago.csv', sep=',', encoding='UTF-8')
```

50개 URL에서 원하는 정보를 얻자

In [36]:

```
df = pd.read_csv('..../data/04_best_sandwiches_list_chicago.csv', index_col=0)
df.head()
```

	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 [37]: html = urlopen(df['URL'][0])
soup_tmp = BeautifulSoup(html, "html.parser")
soup_tmp
```

```
<!DOCTYPE doctype html>

<html lang="en">
<head>
<!-- Urbis magnitudo. Fabulas magnitudo. -->
```

- 먼저 하나의 URL에 대해서 테스트해보자

2012 BEST SANDWICHES

1. Old Oak Tap BLT

“Truly inspired.”

PUBLISHED OCT. 9, 2012



1 COMMENT





PHOTO: ANNA KNOTT; FOOD STYLIST: LISA KUEHL

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

MARKETPLACE

EVENTS & PARTY PIX

RESOURCE GUIDE

ADVERTISEMENT





PHOTO: ANNA KNOTT; FOOD STYLIST: LISA KUEHL

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.

em | 329.83 x 47

\$10. 2109 W. Chicago Ave., 773-772-0406,
[theoldoaktap.com](http://www.theoldoaktap.com)

The screenshot shows a browser's developer tools element inspector. The element currently selected is an *em* tag located within a

element with class "addy". The element's value is '\$10. 2109 W. Chicago Ave., 773-772-0406, " [theoldoaktap.com](http://www.theoldoaktap.com/)". The browser's address bar at the top also displays this URL. Below the element tree, the browser's navigation bar is visible, showing the current page's title and other navigation controls. At the bottom of the developer tools, there are tabs for Styles, Event Listeners, DOM Breakpoints, Properties, and Accessibility, with the Styles tab currently active. A CSS selector filter is present, and the element's style definition is shown as:

```
element.style {
```

On the right side of the developer tools, there is a margin/border/padding panel with a dashed orange border around the 'margin' and 'border' fields.

```
show_faces=false&width=50">...</div>
▶ <p>...</p>
▼ <p class="addy">
  ▼ <em> == $0
    "$10. 2109 W. Chicago Ave., 773-772-0406, "
    <a href="http://www.theoldoaktap.com/">theoldoaktap.com</a>
    </em>
  </p>
▶ <aside class="story-nav-aside">...</aside>
▶ <footer>...</footer>
▶ <div class="clearfix">...</div>
<script>
  var newsletterChoice = "Dish";
</script>
▶ <div id="footer-newsletter-subscribe" class="margin-top-10 margin-bottom-40">...</div>
▶ <script>...</script>
▶ <section id="share" class="clearfix">...</section>
```

html #page_htmlid_19495 #content-wrap #page #article-19495 div div p.addy em

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

```
element.style {
```

margin -
border -

```
In [38]:
```

```
print(soup_tmp.find('p', 'addy'))
```

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

- 어딜 뒤져야 하는지 알았다

In [39]:

```
price_tmp = soup_tmp.find('p', 'addy').get_text()  
price_tmp
```

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

- OK... 그런데 가격과 주소가 같이 있네ㅠㅠ.

```
In [40]: re.split('.,', price_tmp)
```

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

```
In [41]: price_tmp = re.split('.,', price_tmp)[0]  
price_tmp
```

```
'\n$10. 2109 W. Chicago Ave'
```

- 주소의 끝이 .,로 끝나는 미국식 습관에 맞춰 주소 뒤를 제거

In [42]:

```
re.search( '\$\d+\.( \d+)?', price_tmp).group()
```

```
'$10.'
```

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

In [43]:

```
end = re.search('\$\d+\.( \d+)?', price_tmp).end()
price_tmp[end+1:]
```

'2109 W. Chicago Ave'

- 가격이 끝나는 지점의 위치를 이용해서 그 뒤는 주소로 생각한다

In [44]:

```
price = []
address = []

for idx, row in df[:3].iterrows():
    html = urlopen(row['URL'])
    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)

    end = re.search('\$\d+\.\(\d+\)?', price_tmp).end()
    address.append(price_tmp[end+1:])

print(row['Rank'])
```

In [45]:

price

```
[ '$10.', '$9.', '$9.50' ]
```

In [46]:

address

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

- **OK** 이제 전체로 다 돌리자

In [*]:

```
import time
price = []
address = []

for idx, row in df.iterrows():
    html = urlopen(row['URL'])
    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)

    end = re.search('\$\d+\.\d+', price_tmp).end()
    address.append(price_tmp[end+1:])

    time.sleep(0.5)
    print(row['Rank'], ' / ', '50')
```

```
In [50]:
```

```
len(price), len(address), len(df)
```

```
(50, 50, 50)
```

In [51]:

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

	Cafe	Menu	Price	Address
Rank				
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

- 일은 PC가 하고 열매는 우리가 먹는다 ~~~~

```
In [52]: df.to_csv('..../data/04_best_sandwiches_list_chicago2.csv', sep=',', encoding='UTF-8')
```