[python] BeautifulSoup를 통한 영화리뷰 scraping 하기 — 나무늘보의 개발 블로그

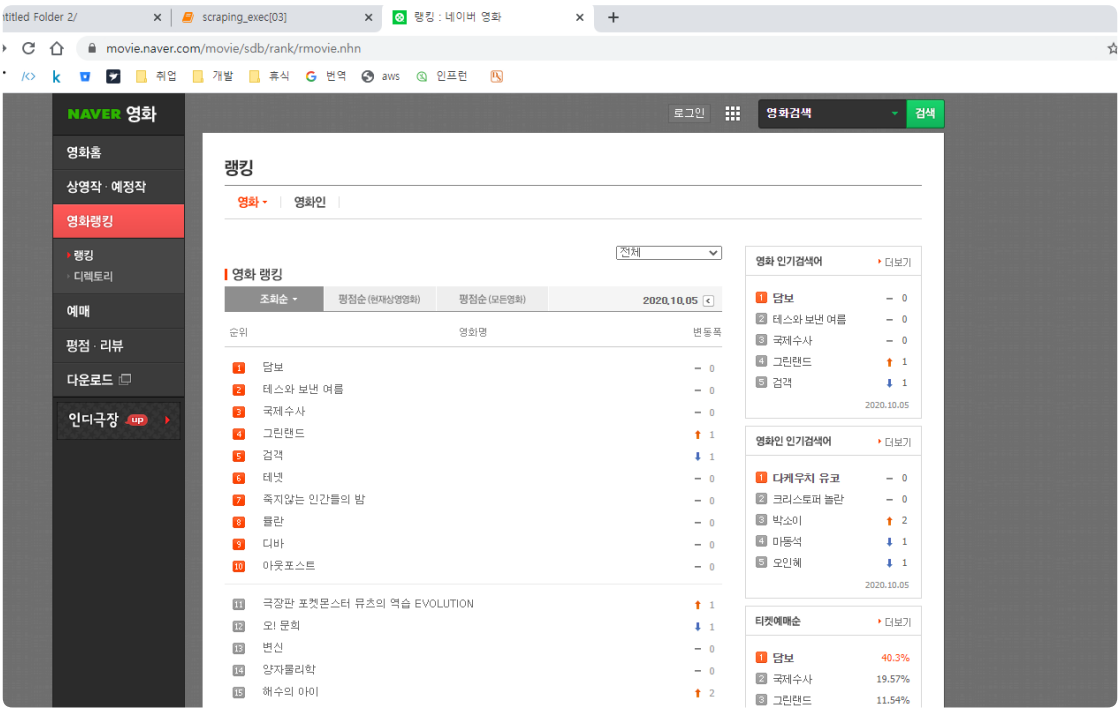
노트북: 첫 번째 노트북
만든 날짜: 2020-10-25 오후 5:13
URL: https://continuous-development.tistory.com/106?category=736681

Python

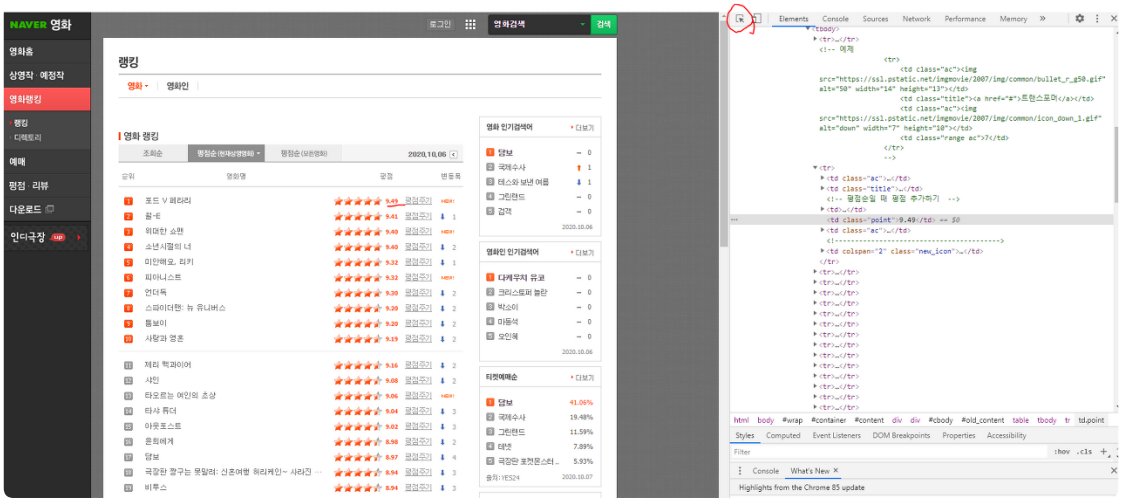
[python] BeautifulSoup를 통한 영화리뷰 scraping 하기

2020. 10. 7. 09:06 수정 삭제 공개

제일 흔한 영화 리뷰를 가져오기로 한다. 이 사이트는 네이버 영화 사이트이다.
movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=2017-05-01



크롤링하기 위해서는 html의 구조를 봐야 한다. 크롬에서 F12 버튼을 통해 해당 페이지의 소스를 본다. 이걸 통해 해당 위치가 어디인지 찾을 수 있다.



아래 환경은 jupyter에서 실행하였다. 크롤링할 때 대표적으로 쓰는 BeautifulSoup을 사용하였다.

영화 평점에 대한 평점 변화(시각화)

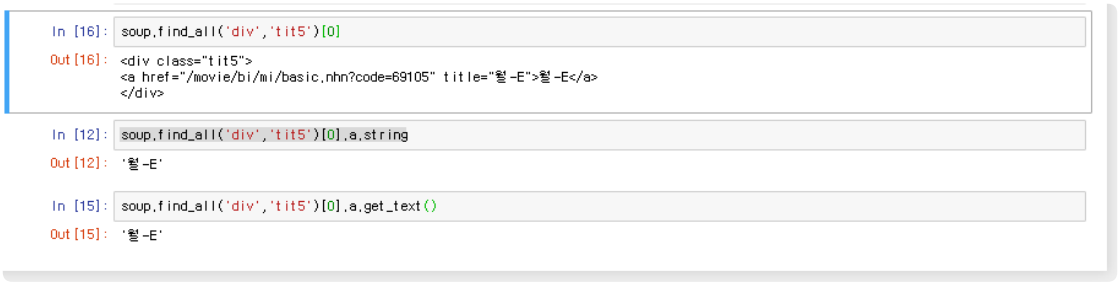
In [3]:
```
from urllib.request import urlopen
from bs4 import BeautifulSoup
from urllib.error import HTTPError
from urllib.error import URLError
```

In [4]:
```
base_url='https://movie.naver.com/'
sub_url = 'movie/sdb/rank/rmovie.nhn?sel=cur&date=20201005'
try:
    html = urlopen(base_url+sub_url)
except HTTPError as he:
    print('http error')
except URLError as ue:
    print('url error')
else:
    soup = BeautifulSoup(html.read(),'html.parser')
```

기본적인 필요 로직이다. base_url에 기본 메인 home url을 넣고 sub에 우리가 이동한 곳에 대한 url을 넣는다.

그리고 try except를 통해 에러가 났을 때 해당 부분으로 보낸다.

그게 아닐 경우 BeautifulSoup를 이용해서 html을 읽어 soup에 저장한다.



그 후 soup를 통해 find_all이라는 명령어를 통해 모든 것을 가져오고 그 중에 [0] 번째 인덱스의 값을 가져온다.
거기서 a태그의 string 값을 가져오거나
또는 a태그를 get_text로 값을 가져온다.

이번엔 point 값을 가져온다.


```
In [27]: soup.find_all('td','point')[0].get_text()
Out[27]: '9.41'
```

이렇게 값을 가져오고 가져온 값을
리스트에 넣어주는 작업을 하자.

```
In [23]: move_names = [soup.find_all('div','tit5')[n].a.get_text() for n in range(0,50)]
         print(len(move_names))
         print(move_names)

50
['헬-E', '소년시절의 너', '브레이크 더 사일런스: 더 무비', '미안해요, 리키', '언더독', '스파이더맨: 뉴 유니버스', '톰보이', '사랑과 영혼', '제리 맥과이어', '사인', '타샤 튜더', '아웃포스트', '담보', '윤희에게', '극장판 짱구는 못말려: 신혼여행 허리케인~ 사라진 아빠!', '비투스', '여행자', '하녀', '아무도 모른다', '너의 이름은.', '마미', '브리짓 존스의 일기', '찬실이는 복도 많지', '할거:유관순 이야기', '69세', '레이디 버드', '기생충', '아무르', '보리밭을 흔드는 바람', '로렌스 애니웨이 ', '500일의 썸머', '겁쟁', '주디', '그래비티', '테넷', '경계선', '프란시스 하', '신문기자', '나의 산티아고', '드라이브', '블레이드 러너 2049', '설국열차', '날씨의 아이', '라붐', '오! 문희', '반교: 디텐션', '다만 악에서 구하소서', '갑쪽같은 그녀', '홀리 모터스', '애드 아스트라']

In [28]: move_points = [soup.find_all('td','point')[n].get_text() for n in range(0,50)]
         print(len(move_points))
         print(move_points)

50
['9.41', '9.39', '9.35', '9.32', '9.30', '9.20', '9.20', '9.19', '9.16', '9.08', '9.04', '9.02', '9.00', '8.98', '8.94', '8.94',
'8.91', '8.90', '8.87', '8.78', '8.68', '8.68', '8.65', '8.64', '8.63', '8.50', '8.49', '8.48', '8.47', '8.44', '8.42', '8.35',
'8.34', '8.29', '8.26', '8.17', '8.13', '8.11', '8.10', '8.07', '8.00', '7.98', '7.97', '7.94', '7.93', '7.80', '7.65', '7.57',
'7.52', '7.26']
```

이제 하루의 날이 아닌 여러 개의 날의 데이터를 가져오는 작업을 해보자 일단 우리가 필요한 날짜 데이터를 만든다.

```
In [48]: date = pd.date_range('2017-5-1',periods=100,freq='D')
         date

Out[48]: DatetimeIndex(['2017-05-01', '2017-05-02', '2017-05-03', '2017-05-04',
                        '2017-05-05', '2017-05-06', '2017-05-07', '2017-05-08',
                        '2017-05-09', '2017-05-10', '2017-05-11', '2017-05-12',
                        '2017-05-13', '2017-05-14', '2017-05-15', '2017-05-16',
                        '2017-05-17', '2017-05-18', '2017-05-19', '2017-05-20',
                        '2017-05-21', '2017-05-22', '2017-05-23', '2017-05-24',
                        '2017-05-25', '2017-05-26', '2017-05-27', '2017-05-28',
                        '2017-05-29', '2017-05-30', '2017-05-31', '2017-06-01',
                        '2017-06-02', '2017-06-03', '2017-06-04', '2017-06-05',
                        '2017-06-06', '2017-06-07', '2017-06-08', '2017-06-09',
                        '2017-06-10', '2017-06-11', '2017-06-12', '2017-06-13',
                        '2017-06-14', '2017-06-15', '2017-06-16', '2017-06-17',
                        '2017-06-18', '2017-06-19', '2017-06-20', '2017-06-21',
                        '2017-06-22', '2017-06-23', '2017-06-24', '2017-06-25',
                        '2017-06-26', '2017-06-27', '2017-06-28', '2017-06-29',
                        '2017-06-30', '2017-07-01', '2017-07-02', '2017-07-03',
                        '2017-07-04', '2017-07-05', '2017-07-06', '2017-07-07',
                        '2017-07-08', '2017-07-09', '2017-07-10', '2017-07-11',
                        '2017-07-12', '2017-07-13', '2017-07-14', '2017-07-15',
                        '2017-07-16', '2017-07-17', '2017-07-18', '2017-07-19',
                        '2017-07-20', '2017-07-21', '2017-07-22', '2017-07-23',
                        '2017-07-24', '2017-07-25', '2017-07-26', '2017-07-27',
                        '2017-07-28', '2017-07-29', '2017-07-30', '2017-07-31',
                        '2017-08-01', '2017-08-02', '2017-08-03', '2017-08-04',
                        '2017-08-05', '2017-08-06', '2017-08-07', '2017-08-08'],
                       dtype='datetime64[ns]', freq='D')
```

이건 안 해도 되지만 시간이 걸릴 때 잘 돌아가고 있는지 확인을 하기 위해 사용한다.

```
In [49]: import urllib
         from tqdm import tqdm_notebook # loop 에 대한 진행 상태에 대해 보여준다.
         import time

In [*]: for n in tqdm_notebook(range(100)):
            time.sleep(0.1)
```

40% 40/100 [00:04<00:06, 9.38it/s]

tqdm_notebook은 이렇게 loop에 대한 진행사항을 볼 수 있다.

```
In [*]: for x in tqdm_notebook(range(5),desc='outer'):
            for y in tqdm_notebook(range(20),desc='innter'):
                time.sleep(0.1)
```

outer 60% 3/5 [00:06<00:04, 2.18s/it]

innter 100% 20/20 [00:02<00:00, 9.35it/s]

innter 100% 20/20 [00:02<00:00, 9.30it/s]

innter 100% 20/20 [00:02<00:00, 9.43it/s]

innter 15% 3/20 [00:00<00:02, 8.41it/s]

이렇게도 가능하다.

이제 date와 names와 point를 가져와 써보자

```
In [75]: names_result=[]
         points_result=[]
         date_result=[]
```

```
In [76]: base_url='https://movie.naver.com/'
         sub_url = 'movie/sdb/rank/rmovie.nhn?sel=cur&date='
         for day in tqdm_notebook(date):
         #     print(day)
             html = base_url+sub_url+ '{date}' # 가변 변수
             response = urlopen(html.format(date=urllib.parse.quote(day.strftime('%Y%m%d')))) # 여기서 오픈한다. 가변변수값은 이렇게 넣
             soup = BeautifulSoup(response,'html.parser')
             end = len(soup.find_all('td','point'))
             names_result.extend([soup.find_all('div','tit5')[n].a.string for n in range(0,end)])
             points_result.extend([soup.find_all('td','point')[n].string for n in range(0,end)])
             date_result.extend([day for n in range(0,end)])
```

100% 100/100 [02:01<00:00, 1.32s/it]

```
In [77]: print(len(names_result))
         print(len(points_result))
         print(len(date_result))
```

```
4723
4723
4723
```

이 값을 DataFrame 형태로 만들어 주자

```
In [78]: movieDF = pd.DataFrame({'date':date_result, 'name':names_result, 'point': points_result})
```

```
In [80]: movieDF.head()
```

Out[80]:

	date	name	point
0	2017-05-01	히든 피겨스	9.38
1	2017-05-01	사운드 오브 뮤직	9.36
2	2017-05-01	시네마 천국	9.29
3	2017-05-01	미스 슬로운	9.26
4	2017-05-01	잉여들의 히치하이킹	9.25

우리는 point를 사용하기 위해 데이터 타입을 바꿔줄 필요가 있다.

```
In [81]: movieDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4723 entries, 0 to 4722
Data columns (total 3 columns):
date     4723 non-null datetime64[ns]
name     4723 non-null object
point    4723 non-null object
dtypes: datetime64[ns](1), object(2)
memory usage: 110.8+ KB
```

- astype 타입변환

```
In [84]: type(movieDF['point'])
Out[84]: pandas.core.series.Series

In [85]: movieDF['point'] = movieDF['point'].astype(float)

In [86]: movieDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4723 entries, 0 to 4722
Data columns (total 3 columns):
date     4723 non-null datetime64[ns]
name     4723 non-null object
point    4723 non-null float64
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 110.8+ KB
```

astype을 이용해서 데이터 타입을 바꿔준다.

그룹으로 묶어서 보기 위해서는

- 내가 원하는 영화의 평점을 총점으로 확인하고 싶다면?
- 피봇테이블을 이용할 수 있습니다.

```
In [94]: movie_pivot = pd.pivot_table(movieDF, index=['name'], aggfunc=np.sum) # 이런식으로 집계를 할 수 있다.
         movie_pivot
```

Out[94]:

name	point
10분	124.46
47 미터	149.23
500일의 썸머	75.51
7년-그들이 없는 언론	137.28
7번째 내가 죽던 날	407.48
7인의 사무라이	36.60
8 마일	195.36
가디언즈 오브 갤럭시	34.22
가디언즈 오브 갤럭시 VOL. 2	484.45

결과에 대해 정렬을 하기 위해서

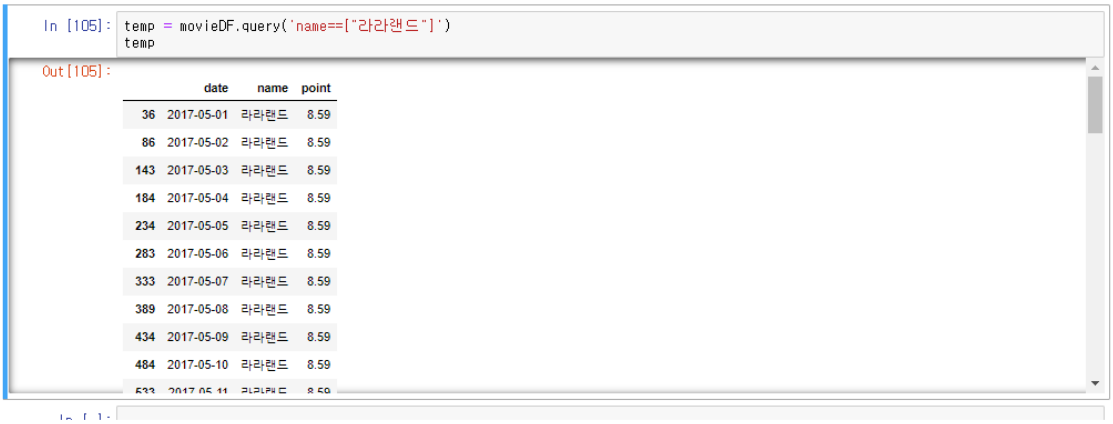
```
- 결과에 대한 정렬이 필요 할 경우 sort_values(by= , ascending=)

In [99]: movie_pivot_sort = movie_pivot.sort_values(by='point',ascending=False)

In [100]: movie_pivot_sort
```

Out[100]:

name	point
댄서	914.60
서서평, 천천히 평온하게	889.64
오두막	861.65
라라랜드	858.89
너의 이름은.	738.42
노무현입니다	682.24
보스 베이비	644.21
갯 아웃	630.62
기쿠지로의 여름	613.43

칼럼에 where 조건절이다.

- 컬럼에 where 조건절



```
In [105]: temp = movieDF.query('name=="라라랜드"')
          temp
```

Out[105]:

	date	name	point
36	2017-05-01	라라랜드	8.59
86	2017-05-02	라라랜드	8.59
143	2017-05-03	라라랜드	8.59
184	2017-05-04	라라랜드	8.59
234	2017-05-05	라라랜드	8.59
283	2017-05-06	라라랜드	8.59
333	2017-05-07	라라랜드	8.59
389	2017-05-08	라라랜드	8.59
434	2017-05-09	라라랜드	8.59
484	2017-05-10	라라랜드	8.59
533	2017-05-11	라라랜드	8.59

이런 식으로 내가 원하는 데이터를 가져올 수 있다.

'**Python**' 카테고리의 다른 글□

[Python] BeautifulSoup을 통한 이미지 스크래핑 하기□

[python] 영화 리뷰에 대한 자연어 처리분석/ 감성분석하기 feat. 스크래핑□

[python] BeautifulSoup를 통한 영화리뷰 scraping 하기□

[Python] 파이썬 기초 14 - 아주 기초적인 pandas 사용법과 예제□

[Python] 파이썬 기초 13 - 파이썬을 통한 파일 입출력 사용법□

[Python] 파이썬 기초 12 - 예외처리□

Beautiful soup 사용 python Beautiful soup python scraping 파이썬 scraping 파이썬 스크래핑

나무늘보스

혼자 끄적끄적하는 블로그 입니다.