

Python3 + BeautifulSoup4 와 함께하는 즐거운 웹크롤링

발표자 : 김다인

오늘의 주제

- Python3와 BeautifulSoup4를 이용한 정적 웹페이지 크롤링
 - 선행지식: Python3 기초 (특히 **문자열**, **파일 입출력**에 관한 함수)
 - 필요한 것: Visual Studio Code 설치
- 이 시간에 하는 것
 - 파이썬 간단 복습 (for beginners or who just can't remember python syntax)
 - 간단한 HTML 문서 구조와 태그, 클래스, id 이해 점검
 - 15분 초간단 BeautifulSoup4 기초 부분 학습
 - 웹페이지로 실습해보자!
- 심화: Selenium을 이용한 동적 웹페이지 크롤링

Python3 Summary (1)

- len(문자열) : 문자열의 문자 개수를 세서 정수로 반환
 - classic='The great gatsby'
 - print(len(classic)) #16 반환
- len(배열) : 배열 내 원소의 개수를 세서 정수로 반환
 - book=['The great Gatsby', 'Le Petit Prince', 'A steady rain', 'A night flight', 'The beginners', 'The cathedral']
 - print(len(book)) #6 반환
- 배열.append(데이터) : 배열에 데이터를 맨뒤에 추가함
 - book.append('The big short')
 - Print(len(book)) #7

Python3 Summary (2)

- `open(' 경로/파일 이름 ' , ' 모드 ' , encoding= ' utf-8 ')`
 - 이 함수는 파일을 읽어 `_io.TextIOWrapper` 타입으로 반환한다.
 - 파일 이름 : 확장자가 함께 붙음. 같은 디렉토리 내에 있으면 경로 필요
X
 - 모드 : 이 파일을 읽을 것인지 수정할 것인지 정함 (w/r)
 - Encoding : 파일에 영문 외 다른 글자가 있을 때 읽기 위해 씀.
 - 보통 텍스트 파일일 경우 utf-8
 - Ex) `fp = open('test.txt', 'w', encoding='utf-8')`
- 파일 이름.`close()`
 - 한 번 연 파일은 닫아야 한다.
 - Ex) `fp.close()`

Python3 Summary (3)

- `fp.writelines(배열)`
 - 배열을 텍스트 파일에 입력한다.
 - 원소마다 개행을 주고 싶다면, `fp.writelines(book+'\\n')`
 - 앞에서부터 덮어쓰게 된다.
 - Ex) `fp.writelines(book)` # 실행결과 직접 확인
 - `fp.write(문자열)` 함수는 개행을 추가하지 않고 텍스트만 입력함
- `fp.readlines(문자열/배열)`
 - 파일로부터 문자열을 읽어서 배열로 반환한다.
 - For 함수와 함께 쓰면 시너지 효과
 - Ex) `temp_list = []`
 - `for line in fp.readlines():`
 - `temp_list.append(line)`

Python3 Summary (4)

- `.strip('token')`
 - 문자열에서 앞뒤로 들어가는 token을 자른 문자열 반환
 - Ex) `wow = '---Wow! You're the best student ever!---`
 - `print(wow.strip('-'))` #Wow! You're the best student ever!
- `.split('token')`
 - Token을 기준으로 문자열을 잘라 배열로 반환
 - Ex) `print(wow.split(' '))` # ['Wow!', 'You're', 'the', 'best', 'student', 'ever!']
- `.replace('a', 'b')`
 - 문자열에서 모든 a를 b로 변환
 - Ex) `print('tacocat'.replace('c', 'r'))`

Python3 Summary (5)

- 리스트 컴프리헨션
- Just look at it
 - `Temp_list = [tmp for tmp in book if tmp]`
 - 무슨 뜻인지?
- [반환할 원소 // for 원소 in 배열 // if 조건] 이렇게 끊어 보면 쉽다!
- 이 때, 반환할 원소는 전체 원소 중 if 조건에 맞는 원소들만 해당된다.
- Ex) `even_list = [x for x in range(10) if x%2==0]`
- `Print(even_list) # [0, 2, 4, 6, 8]`

Python3 Summary (6)

- 리스트 컴프리헨션이 왜 필요한가?
 - 웹크롤링으로 갓 따온 내용물들은 빈 원소("")를 많이 포함한다. 이것은 필요하지 않은 내용물(dummy)이다.
 - Dummy를 없애고 싶다!
 - Ex) `old_list = ['My', ' ', 'textbook', '', 'is', '', 'quite', '', 'old']`
 - `new_list = [word for word in old_list if word]`
 - `Print(new_list) # ['My', 'textbook', 'is', 'quite', 'old']`
- 어떻게 이럴 수 있을까?
- if 조건에 그냥 원소가 들어가면 화이트스페이스(공백) 원소는 언제나 false를 반환하기 때문이다. 즉, 유효한 원소만 체크하게 된다.
- 단, '\n'나 ' ' 등은 true값을 가진다. 이 경우는 replace를 하면 된다.

HTML 문서 구조

- 그 어떤 복잡한 웹 문서라도, 결국은 다음과 같은 큰 뼈대를 가진다.

```
<html>
```

```
  <head> </head>
```

```
    <body>
```

```
      <!-- -->
```

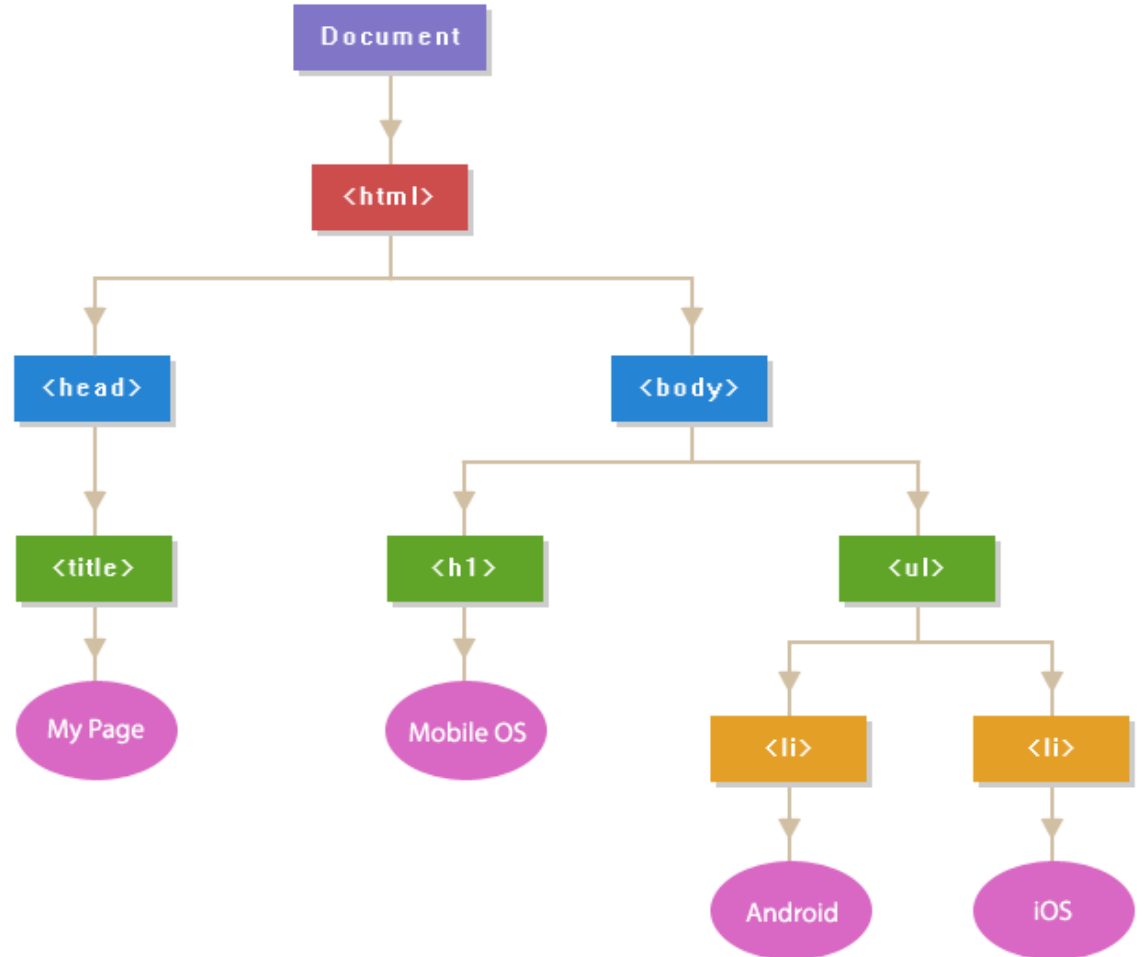
```
    </body>
```

```
</html>
```

HTML 문서 구조

- 모든 HTML 문서는 트리 구조로 나타낼 수 있다.
- 다음과 같은 것을 DOM Node 혹은 DOM Tree 라고 부른다.

- What's DOM?
- [Document Object Model](#)



BeautifulSoup4 사용과 관련한 태그, 속성

- **Tag attribute**


- Html5로 넘어가면서 html 태그에서 `<a>` 태그의 href나 target 같은 기능적 요소를 제외한 align이나 text-color 같은 디자인적 요소는 모두 css style로 처리하게끔 권장된다.
- 하지만, attribute selector로 탐색할 때도 있으므로 잘 봐둔다.

- **Class**

- Css에서 속성 부여를 위해 쓴다. 같은 클래스들끼리 전부 같은 css 효과를 받게 된다.
- `<p class="my_class"> </p>`

- **Id**

- Html 문서에서 딱 하나의 속성에만 부여할 수 있다. Class보다 영향력 있음
- `<div id="only_one"> </div>`



더 이상의
자세한 설명은
생략한다.

BeautifulSoup4를 하기 전에

- 사실 BeautifulSoup4를 더 정확하게 활용하고 싶다면, 그 전에 HTML 문서에 대한 이해와 HTTP 헤더에 대한 이해가 필요하다.
- 하지만 우리는 일단 맨땅에 헤딩을 하기 때문에 지금 당장은 많은 지식이 필요하지 않으니, 일단 한 다음 이해해보자!

BeautifulSoup4 문법

```
from bs4 import BeautifulSoup
from urllib.request import urlopen
```

```
from bs4 import BeautifulSoup
from urllib.request import urlopen
```

일단 BeautifulSoup 라이브러리를 import하고, 그 다음 url library-request에서 urlopen을 import한다. (라이브러리 내 다른 기능은 필요 없다.)

스피드웨건 : 파이썬에서 외부 라이브러리나 모듈(다른 파이썬 파일)을 가져와서 쓰려면 import를 하면 된다.

BeautifulSoup4 문법

- `Url = '문자열'` #파싱하고 싶은 웹페이지 주소
- `Html = urlopen(url)` #웹페이지를 연다
- `Source = html.read()` #연 웹페이지를 읽어 소스코드를 반환
- `Html.close()`

```
url = 'https://www.melon.com/chart/index.htm'  
html = urlopen(url)  
source = html.read()  
html.close()
```

BeautifulSoup4 문법- 고난과 해결 (urllib 모듈의 한계)

```
raise HTTPError(req.full_url, code, msg, hdrs, fp)
urllib.error.HTTPError: HTTP Error 406: Not Acceptable
```

- 실제로, 신경 써서 짠 홈페이지는 이대로 들어가면 406 코드를 뱉으며 접속 거부를 하는 경우가 꽤 있다. (우리 학교는 안 그럼)
- 이 경우 클라이언트가 사람이 아니라 컴퓨터 프로그램인 걸 알기 때문에 거부하는 것이다.
- What can we do to fix this problem then?
 - path 1. urllib 대신 requests 모듈 import, User-Agent 설정
 - path 2. 아예 홈페이지 소스 긁어버리기

Path 1. User-Agent 설정, 왜 할까?

- Urllib 모듈을 import해서 쓸 경우, 서버 사이트에서 urllib를 통해 접속했음을 알 수 있다.
- urllib는 HTTP 요청 헤더를 수정할 수 없기 때문에, urllib 접속 방식을 막으면 소스 코드 전체를 텍스트 파일로 굶는 방법밖에 없다.
- 대신, requests 모듈을 import해와서 Header(클라이언트 유저 정보)를 설정하면 사용자로 인식해서 접속이 승인된다.
- Urllib 완전 쓸모없는데? 왜 아직까지 남아 있는 거임?
 - 이미지 다운로드를 할 때 괴물 같은 저력을 발휘함. 참고로 순수 requests로는 이미지 링크는 받을 수 있어도 다운로드 못 한다.

Path 1. User-Agent 설정하기

- 위의 코드를 전부 지우고 다음과 같이 바꾼다.
- (github에 코드 있음)

```
from bs4 import BeautifulSoup
import requests
```

```
headers = {'User-  
Agent': 'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36'} #headers=부터 여기까지 한 줄임  
url = 'https://www.melon.com/chart/'  
html = requests.get(url, headers = headers).text
```

Path 2. 소스 코드 긁기

- 이것은 Path 1에서 세 줄만 추가하면 된다.
- 우리는 이 방법을 쓰지 않는다.
- 심심풀이로 코드 분석이 필요하거나, 소스 코드가 자주 변경되는 페이지라 버전 아카이빙이 필요할 때 쓰기 좋다.
- 다음 코드 추가

```
out_fp = open("page.html", "w", encoding="utf-8")
out_fp.write(html)
out_fp.close()
```

예제 HTML 코드 분석의 시간

- 이 파일은 잠시 접어두고, 간단히 문법을 학습해보자.
- Github에 올라와있는 example.html 파일을 이용해 파싱해보자.
 - 허접해 보이지만, 나름대로 콘텐츠가 있는 문서다.
- 각자 노트에 DOM TREE를 그려본다.
- 이 때, id나 class, attribute 등은 밑에 조그맣게 표시한다.

BeautifulSoup 진짜 문법 (1)

- Bs4_ex_template.py 파일을 연다.
- 웹페이지를 소스 코드 텍스트로 받아둔 html 변수가 있다. 이 변수를 BeautifulSoup 라이브러리가 사용할 수 있게 적절하게 파싱해줘야 한다.
- Soup = BeautifulSoup(html, 'html.parser')
 - 여담: html.parser 말고 lxml으로 해도 된다. 나 같은 초보 입장에서 퍼포먼스 차이를 느낄 수 없음
 - 참고로, 이 때 html은 **문자열(str)**이다. 즉, <html>로 시작해서 </html>로 끝나는 하나의 거대한 문자열이다. 쉬운 예제로 시험해보고 싶다면, 간단한 html 코드를 자작으로 만들어 문자열에 저장하고 시험해보기 바란다.

BeautifulSoup 진짜 문법 (2)

- 탐색은 크게 find와 find_all로 한다.
- Soup.find(): 가장 먼저 검색되는 **태그 1개**를 Tag 형태로 반환
 - Find한 결과를 그대로 출력하면 html 태그와 내용물이 함께 출력된다.
 - 태그를 제거하고 싶다면, 결과물에 .text나 .get_text()를 붙인다.
- Soup.find_all(): 검색되는 **태그 전체를 ResultSet 형태**로 반환
 - Find한 결과가 list로 반환되기 때문에, 그대로 .get_text()가 불가능하다.
 - So how can we utilize this set?
 - For문 이용!
- 직접 실습해보자 -> bs4_ex_template.py

BeautifulSoup 진짜 문법 (3)

1. 태그로 검색하기

- `Soup.find('p')` : 소스 코드의 첫 번째 `<p>` 태그를 반환한다. (=soup.p)
- `Soup.find_all('div')`: 소스 코드의 모든 `<div>` 태그를 찾아 태그들의 리스트 형태로 반환한다. Find_all을 할 경우, 굉장히 많은 태그들이 포함될 가능성이 아주 높으므로 id나 class 등의 조건을 걸어 검색한다.

2. Id로 검색하기

- `Soup.find(id='myspecial')`: 소스 코드에서 어떤 태그든지 id가 'myspecial'인 태그를 찾아 반환한다.
- `Soup.find_all(id='myspecial')`: 소스 코드의 태그들 중 id가 'myspecial' (id인 태그들을 모두 찾아 반환한다.

BeautifulSoup 진짜 문법 (4)

3. 태그 + 클래스로 검색하기

- `Soup.find('p', 'class_name')`: 소스 코드의 p 태그 중 class가 'class_name'인 첫 번째 태그를 찾아 반환한다.
- `Soup.find_all('p', class='class_name')`: 소스 코드의 모든 p 태그 중 class가 'class_name'인 모든 태그들을 찾아 리스트로 반환한다.

4. 태그 + 태그 속성으로 검색하기

- `Soup.find('p', {'align':'center'})`: 소스 코드의 p 태그 중 align 속성이 center인 첫 번째 태그를 찾아 반환한다.
- `Soup.find_all('p', attrs={'align':'center'})`: 소스 코드의 p 태그들 중 align 속성이 center인 태그들을 모두 찾아 반환한다.
- 번외: align뿐 아니라 class나 id도 사실 이렇게 검색 가능하다(...)

BeautifulSoup 진짜 문법 (5)

5. 중첩 태그 검색하기

- `Soup.select('tag1 tag2 tag3 ...')` : `tag1`, `tag2`, `tag3` 순으로 중첩된 모든 `tag3` 태그들을 찾아준다. `Find_all()`과 동작 논리는 같지만, 중첩 태그만으로 꼭 찾는다는 점이 조금 다르다.
- 왜 다른가?
 - 가령, `body > p > div > a` 순으로 되어 있는 태그와 `body > p > span > a` 순으로 되어 있는 태그를 생각해보자.
 - `Soup.find_all('a')`를 하면 당연히사 `div` 밑의 `a`와 `span` 밑의 `a`가 모두 검색될 것
 - 하지만, `div` 밑의 `a`만 필요하다면 `soup.select('p div a')`만 하면 된다.
- 좋은데? 근데 왜 다른 방식으로 검색하는 거임?
 - 중첩이 아주 많을 경우 불리하다.
 - 우리가 실습할 차트 페이지는 중첩이 어마무지하다. 잠시 뒤 살펴본다.

BeautifulSoup 진짜 문법 (6)

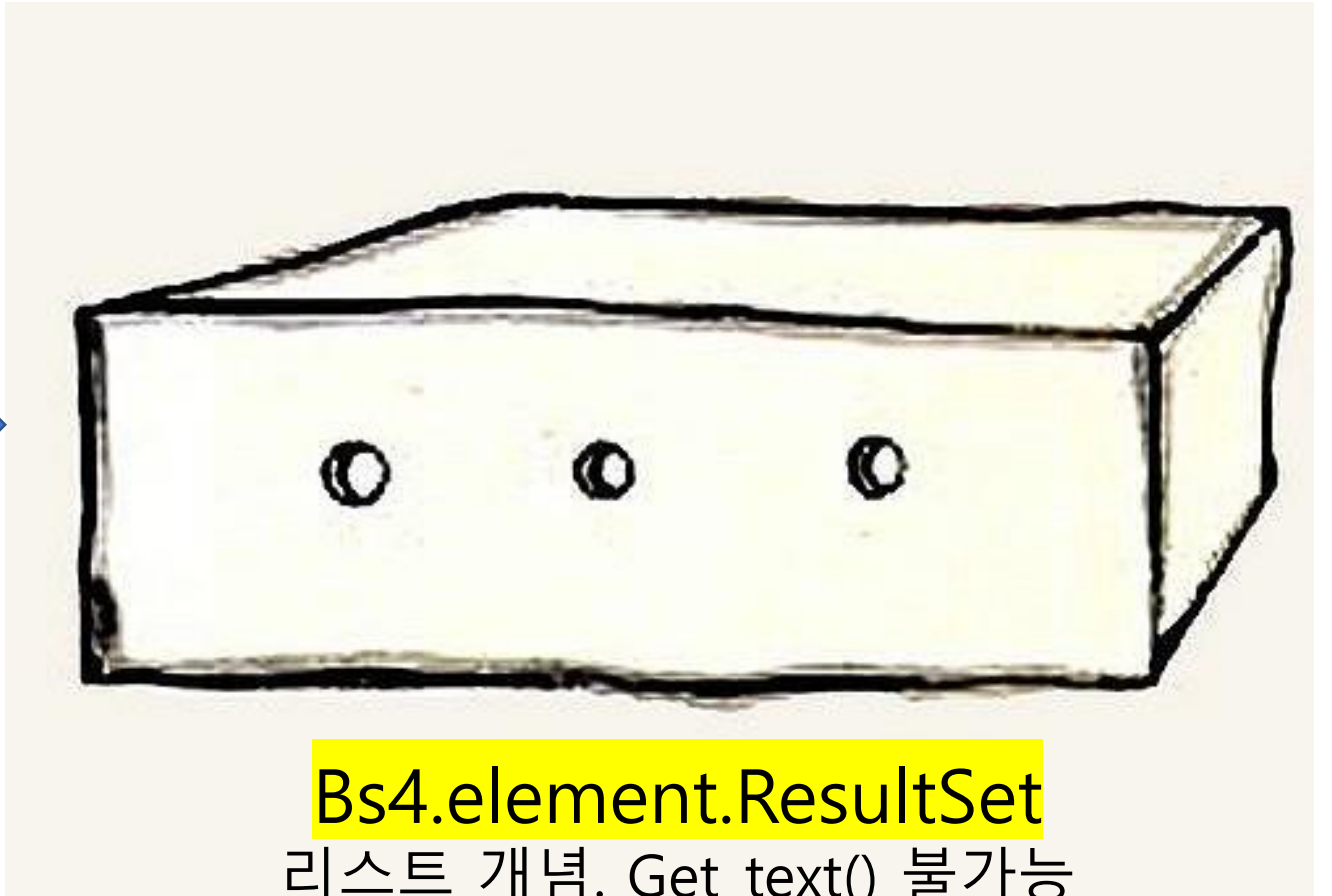
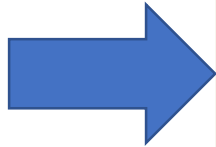
- When it comes to Find_all()
 - Find_all()은 기본적으로 'bs4.element.Tag'들을 원소로 갖는 'bs4.element.ResultSet'을 반환한다.
 - 이것은 바로 문자열로 반환될 수 없지만, for 문을 이용해 각개 원소들을 분리해서 사용할 수 있다.
- 만약, 특정 몇 번째의 태그가 필요하다면 다음과 같이 설정이 가능하다.
- ResultSet[3].get_text()
- 기억하자. Find_all()은 태그 리스트들을 반환하며, 원소가 태그이기 때 문에 당연히 for 문으로 분리가 가능하고 심지어 원소 번호를 지정해서 사용도 가능하다!

BeautifulSoup 진짜 문법 (6)

- To sum up...



`bs4.element.Tag`
get_text() 가능

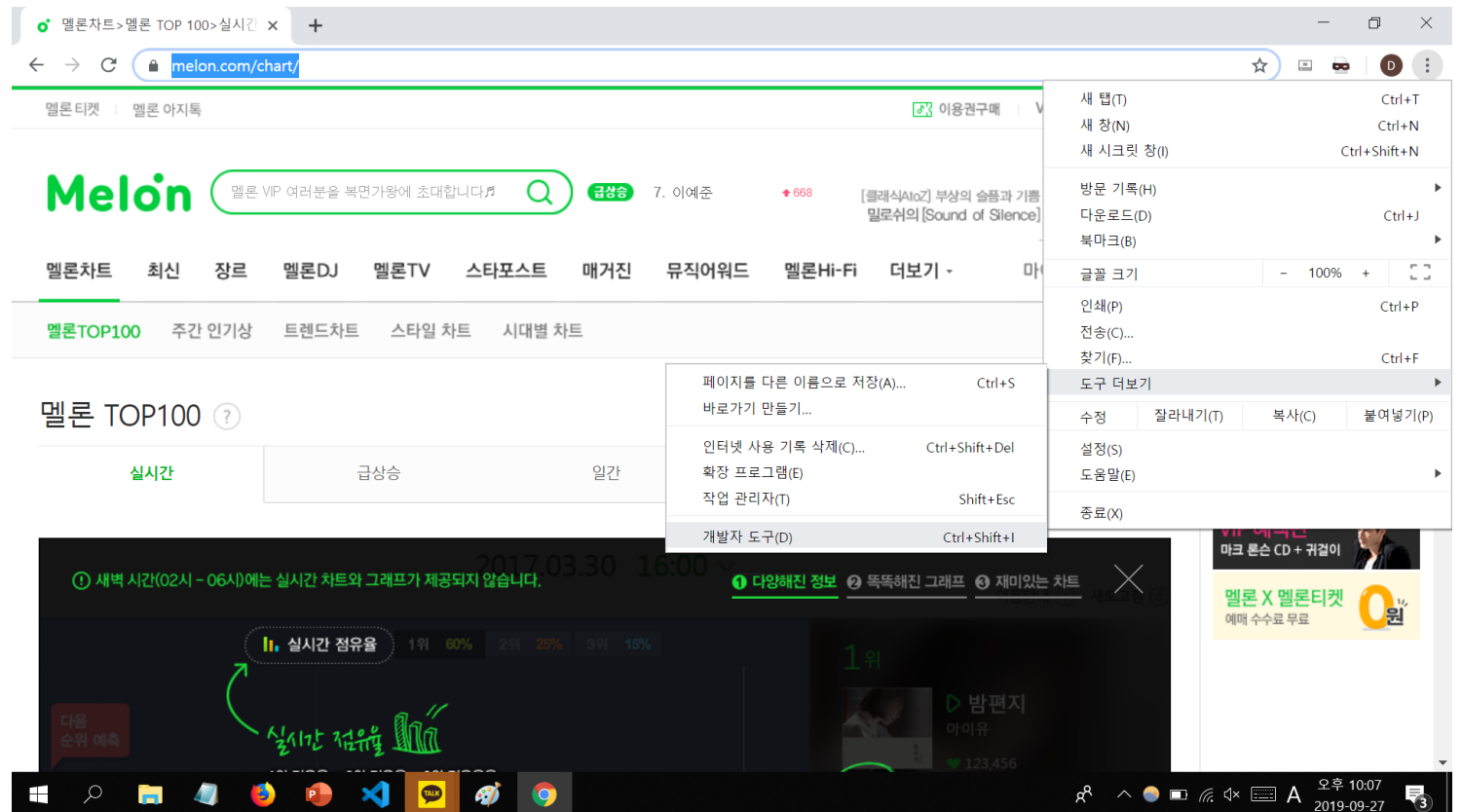


`Bs4.element.ResultSet`
리스트 개념. Get_text() 불가능

타킷 페이지 분석 (1)

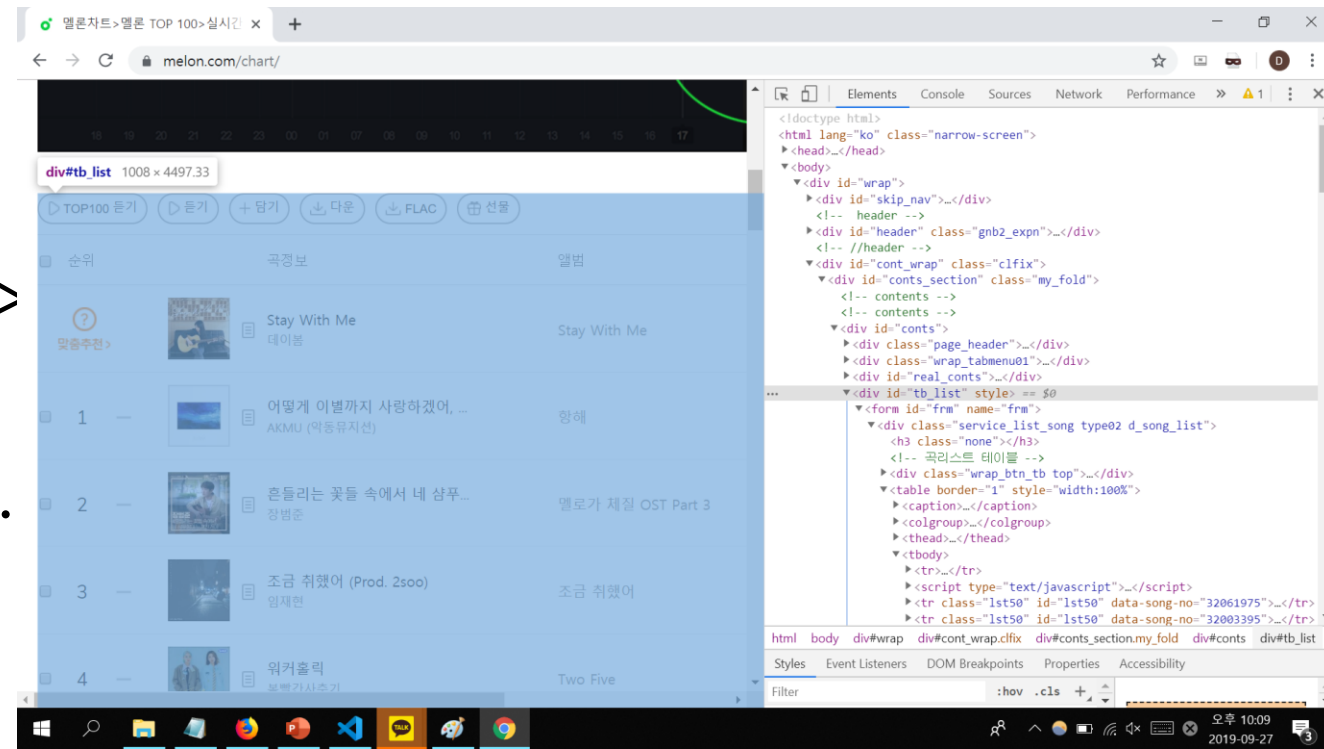
- 잠시 돌아와서, 홈페이지 소스 코드를 분석해보자.
- 우리의 목적 : 멜론 차트 50위까지 타이틀을 가져오기

- 크롬 브라우저를 열고,
타킷 페이지에 접속,
개발자 도구를 연다.
- (?) 개발자 도구?



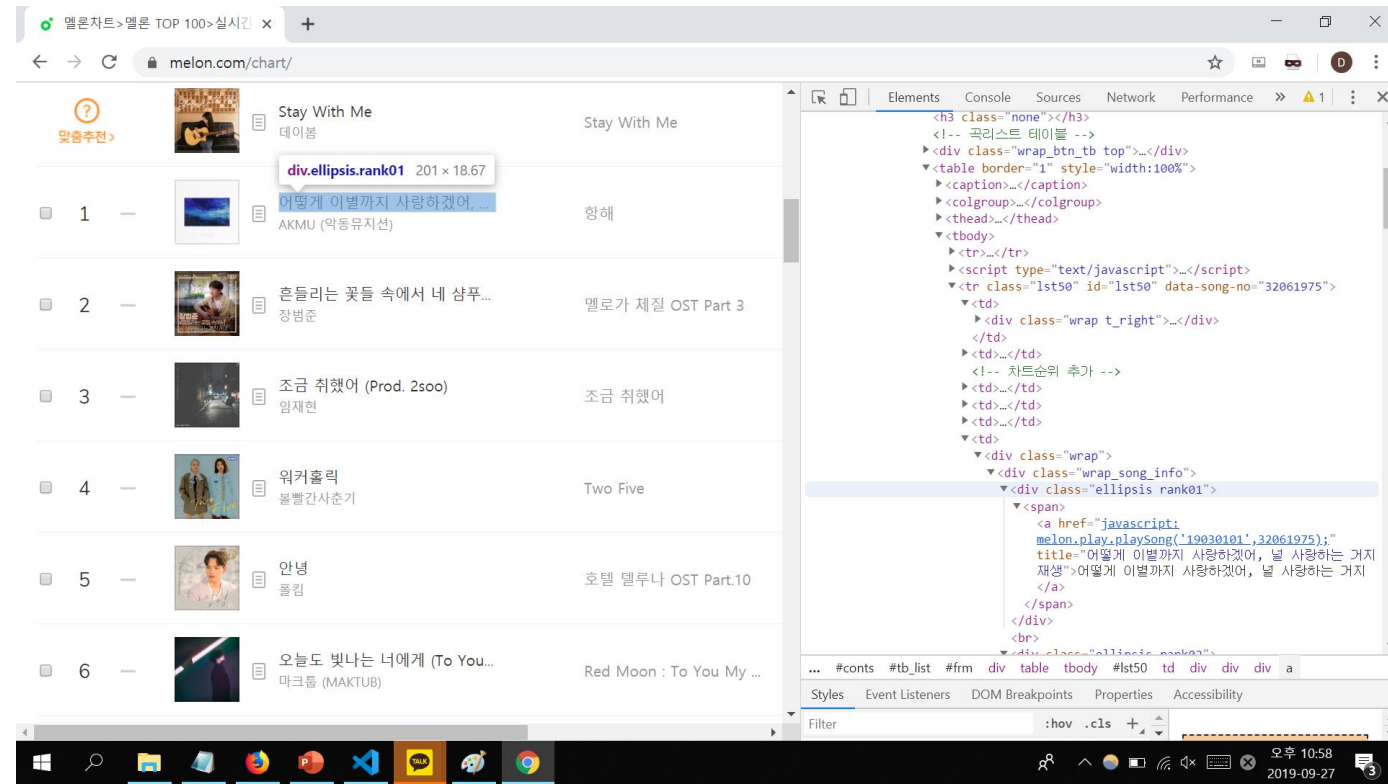
타킷 페이지 분석 (2)

- Elements에서 소스 코드에 마우스를 가져가면, 해당 코드가 가리키는 페이지 영역을 파랗게 표시한다.
- 이 경우, 차트 테이블은 `id=tb_list` div에 있다.
- 조금 더 하부로 내려가보면,
`<div class="wrap_song_info">`
태그가 보일 것이다.
이것이 우리가 필요한 정보다.



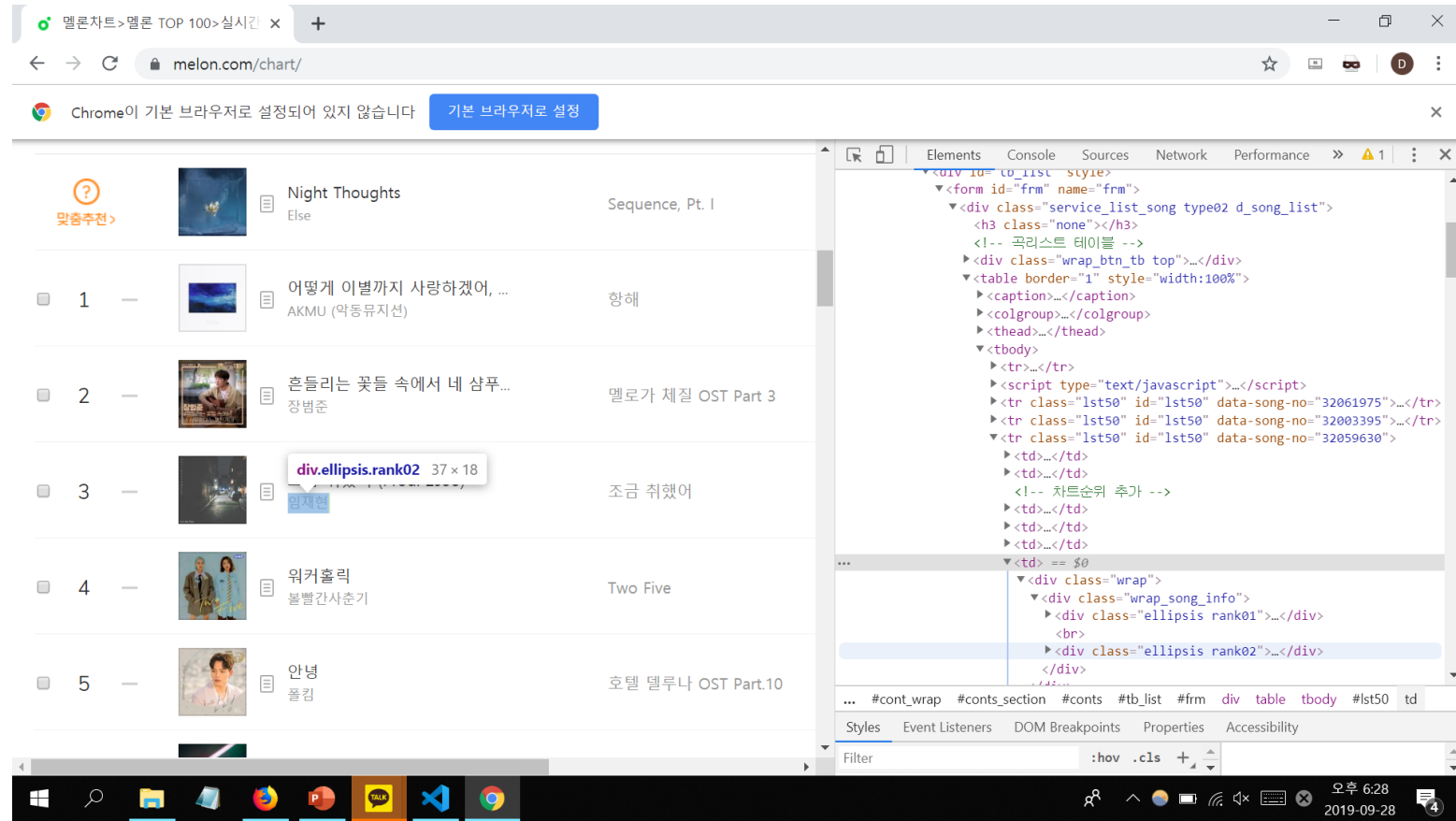
타킷 페이지 분석 (3)

- Title과 artist를 긁어보자
- (?) 먼저, title과 바로 연결되는 태그를 찾아보면...
 - `<div class="ellipsis rank01">`
 - 이 때 기억할 것은, class나 id가 붙은 태그가 추적이 편하다는 것이다.
 - 물론, span 태그가 있지만 별다른 특징이 없으므로 그 상위의 div 태그를 선택한다.



타킷 페이지 분석 (4)

- 같은 방법으로 artist를 찾아보자
 - <div class="ellipsis rank02">



타킷 페이지 분석 (5)

- 정리
 - 타이틀 : <div class="ellipsis rank01">
 - 아티스트 : <div class="ellipsis rank02">

멜론 차트 크롤링 실습 (1)

- 다시 맨 처음 켜둔 파일로 돌아가자.
- `Soup = BeautifulSoup(html, 'html.parser')`
- What next?
- 아까 봐둔 타이틀과 아티스트 텍스트를 담은 태그를 찾는다.
- Div + class 조합이라면?
 - 태그 + 클래스 조합으로 찾는 게 적격!
 - 만능키인 태그 + 태그 속성으로 찾아도 좋다.

멜론 차트 크롤링 실습 (2)

생각하는 시간

멜론 차트 크롤링 실습 (3)

타이틀 : <div class="ellipsis rank01">

아티스트 : <div class="ellipsis rank02">

- Titles = soup.find_all('div', {'class': 'ellipsis rank01'})
 - (=titles = soup.find_all('div', 'ellipsis rank01'))
- Artists = soup.find_all('div', {'class': 'ellipsis rank02'})
 - (=artists = soup.find_all('div', 'ellipsis rank02'))
- 여기까지 했다면 다 한 것이나 마찬가지다.

멜론 차트 크롤링 실습 (4)

- Titles와 artists는 find_all로 찾았기 때문에, 이 변수의 타입은 bs4.element.ResultSet이다.
- 이 ResultSet의 원소들은 까보면 알겠지만, div 태그들 모임이다.
- As you know...
 - ResultSet은 바로 텍스트 출력이 불가능하다. (리스트이기 때문에)
 - 그럼 어떻게 할까? -> for문 사용
 - 일단, 빈 리스트 두 개를 만든다. (하나는 title, 하나는 artist)
- For문을 작성하기 전, 코드를 다시 한 번 살펴보자.
- div 안에는 span 태그가 있고, 이 안에 실질적인 a 태그의 text가 우리가 찾는 타이틀과 아티스트 이름이 있다.
- For문 안에서 find를 또 해준 다음, get_text()를 해줘야 한다.

멜론 차트 크롤링 실습 (5)

- for t in titles:
 title.append(t.find('a').get_text())
- for a in artists:
 artist.append(a.find('a').get_text())
- 병합해서 출력해보자!

멜론 차트 크롤링 실습 끝

```
32 for i in range(rank):
33     print(str(i+1)+"위 : "+artist[i]+" - "+title[i])
```

문제 출력 디버그 콘솔 터미널

2: Python Debug Consc ▼

rawling-example\bs4_do.py "

```
1위 : AKMU (악동뮤지션) - 어떻게 이별까지 사랑하겠어, 널 사랑하는 거지
2위 : 장범준 - 흔들리는 꽃들 속에서 네 샴푸향이 느껴진거야
3위 : 임재현 - 조금 취했어 (Prod. 2soo)
4위 : 불빨간사춘기 - 워커홀릭
5위 : 폴킴 - 안녕
6위 : 마크툽 (MAKTUB) - 오늘도 빛나는 너에게 (To You My Light) (Feat.이라운)
7위 : 케이시 (Kassy) - 가을밤 떠난 너
8위 : HYNN(박혜원) - 시든 꽃에 물을 주듯
9위 : 펀치 (Punch) - 가끔 이러다
10위 : 휘인 (Whee In) - 헤어지자 (Prod. 정키)
11위 : 거미 - 기억해줘요 내 모든 날과 그때를
12위 : 전상근 - 사랑이란 멜로는 없어
```

멜론 차트 크롤링 복습

- 클라이언트가 멜론 차트 노래를 듣고 너무 좋아서 앨범 이름까지 다 긁어오라고 한다. 어떻게 해야 할까?
- 각자 실습!