

# 파이썬을 통한 웹페이지 크롤링 살펴보기

파이썬 초심자를 위한 열네 번째 공감토크

# 안녕하세요. **AskDjango** 이진석입니다.



# 발표자: 이진석

- 파이썬/장고 프로 답변러
  - 주 서식지: [nomade.kr](http://nomade.kr)
  - 페이스북 그룹: [AskDjango](#), [Python Korea](#), [Django](#)
- 파이썬 중독자 (2004년 v2.4~), 장고 중독자 (2008년 v1.0 ~)
- 파이썬/장고 교육 서비스 AskDjango 운영자
  - 질답채널로서 [페이스북 AskDjango 그룹](#) 및 [VOD 서비스](#) 운영
- Microsoft Azure MVP (2016년~)
- (전) 서울대학교, 벤처경영학, "벤처창업 웹프로그래밍" 파이썬/장고 강사

**진행 중에 질문이 있으시면  
참지말고 바로 질문주세요. :)**

참으면 병나요.

# Agenda

## 1. 파이썬/크롤링 소개

## 2. 웹페이지 유형별 사례분석

- 단순 HTML 크롤링
- Ajax 렌더링 크롤링
- AngularJS, Vue.js, React.JS 류의 자바스크립트 렌더링 크롤링

## 3. 장고를 활용하여 크롤링 데이터 저장 및 조회화면 만들기

# 파이썬/크롤링 소개

# 크롤링이란 ?

- 정제되지 않은 웹페이지에서 필요한 데이터만 추출해내는 행위
- robots.txt 를 지켜주세요. => 상도
  - 사이트의 루트에 위치하며 사이트에서 검색엔진 크롤러가 액세스하지 않기를 바라는 부분을 표시
  - medium의 robots.txt
  - 네이버 블로그의 robots.txt
  - google.com의 robots.txt
  - requests에서는 requests-robotstxt를 통해 지원



# 웹페이지는 ... **HTML/CSS/JavaScript** 가 뒤엉킨 혼돈의 카오스

# 업무에 필요한 데이터가 ...

바로 쓸 수 있는 형태로 데이터가  
공개되어있으면 좋겠지만 ...

그것이 바로 **API** 서비스

우리나라에는 **API** 서비스가 잘 없어요.

**심지어 네이버 등에서도 잘 유지가 안 되요.**

자 !!!

우리는 국회의원 명단이 필요합니다.



# 웹페이지에 있으니 가져오면 되겠네?

## 의원활동

국회의장

국회부의장


국회의원현황

의원실행사

의원실재용

의원연구단체

의원외교단체

 국회의원 검색

## 국회의원현황

메인 > 의원활동 > 국회의원현황



국회의원을 정당별, 위원회별, 지역별 등으로 다양하게 검색해보실 수 있습니다.

· 분류선택

· 지역선택

· 이름입력

총 297명의 의원이 있습니다.

의석수 현황

역대 국회의원



**강길부**  
(姜吉夫)  
KANG GHILBOO  
4선  
울산 울주군



**강병원**  
(姜炳遠)  
KANG BYUNGWON  
초선  
서울 은평구을



**강석진**  
(姜錫振)  
KANG SEOKJIN  
초선  
경남 산청군함양군거창  
군합천군



**강석호**  
(姜碩鎬)  
KANG SEOKHO  
3선  
경북 영양군영덕군봉화  
군울진군



**강창일**  
(姜昌一)  
KANG CHANGIL  
4선  
제주 제주시갑



**강효상**  
(姜孝祥)  
KHANG HYOSHANG  
초선  
비례대표

국회의원 검색

검색

총 297명의 의원이 있습니다.

강길부

(姜吉夫)

KANG GHILBOO

4선

울산 울주군

강병원

(姜炳遠)

KANG BYUNGWON

초선

서울 은평구을

강석호

(姜碩鎬)

KANG SEOKHO

3선

경북 영양군영덕군봉화군을진군

강창일

(姜昌一)

KANG CHANGIL

4선

제주 제주시갑

검사기

콘솔

디버거

스타일 편집기

성능

메모리

네트워크

장소

HTML 검색

<div class="content\_body">

<!--게시물 검색-->

<div class="bbs\_search">

<!--//게시물 검색-->

<div id="listArea">

<div class="bodrd\_top\_msch">

<!--국회의원현황 리스트-->

<div class="memberna\_list">

<dl>

<dt>

<strong>

<a href="javascript:jsMemPop('9770276')" title="강길부의원정보 새창에서 열림">강길부</a>

</strong>

<br>

<span class="chi">(姜吉夫)</span>

<br>

KANG GHILBOO

</dt>

<dd class="img">

<a href="#" onclick="jsMemPop('9770276')" title="강길부의원정보 새창에서 열림">



</a>

</dd>

<dd class="mt">4선</dd>

<dd class="ht">울산 울주군</dd>

HTML/CSS/JavaScript가 혼재

여러분의 파이썬/장고 페이스메이커가 되겠습니다. / © AskDjango

19

```

1 <div class="bodrd_top_msch">
2   <!-- 검색결과 리스트 -->
3   <div class="numlist_txt">총 <em class="cr">297</em>명의 의원이 있습니다.</div>
4   <div class="btn_tr">
5     <a href="#" onclick="window.open('/memCond/hnumseat.do','hnumseat','left=0, top=0, resizable=no, scrollbars=no, width=515, height=500');return false" title="새창으로
6       <a href="http://www.rokps.or.kr/profile/profile_number.asp" target="_blank" title="새창으로 열림" class="btntxt02">역대 국회의원</a>
7     </div>
8 </div>
9
10 <!-- 국회의원현황 리스트 -->
11 <div class="memberna_list">
12   <dl>
13     <dt>
14       <strong><a href="javascript:jsMemPop('9770276')" title="강길부의원정보 새창에서 열림">강길부</a></strong><br />
15       <span class="chi">
16         (姜吉夫)
17       </span><br />
18       KANG GHILBOO
19     </dt>
20     <dd class="img">
21       <a href="#" onclick="jsMemPop('9770276')" title="강길부의원정보 새창에서 열림">
22         
23       </a>
24     </dd>
25     <dd class="mt">4선</dd>
26     <dd class="ht">울산 울주군</dd>
27   </dl>
28   <dl>
29     <dt>
30       <strong><a href="javascript:jsMemPop('9770933')" title="강병원의원정보 새창에서 열림">강병원</a></strong><br />
31       <span class="chi">
32         (姜炳遠)
33       </span><br />
34       KANG BYUNGWON
35     </dt>
36     <dd class="img">
37       <a href="#" onclick="jsMemPop('9770933')" title="강병원의원정보 새창에서 열림">
38         
39       </a>
40     </dd>
41     <dd class="mt">초선</dd>
42     <dd class="ht">서울 은평구을</dd>
43   </dl>
44   <dl>
45     <dt>
46       <strong><a href="javascript:jsMemPop('9771036')" title="강석진의원정보 새창에서 열림">강석진</a></strong><br />
47       <span class="chi">
48         (姜錫振)
49       </span><br />
50       KANG SEOKJIN
51     </dt>

```

눈이 돌아갑니다. @\_@!!!

우리 열심히 **Copy&Paste**를 해봅시다.  
그... 그러면 되겠죠? *!!!*

우리. 그러지말아요.

# 컴퓨터에게 ... 파이썬에게 시켜보면 ~

단 몇 줄만으로 ~~~

```
import requests
from bs4 import BeautifulSoup

url = 'http://www.assembly.go.kr/assm/memact/congressman/memCond/memCondListAjax.do'
params = {'currentPage': 1, 'rowPerPage': 300}
html = requests.get(url, params=params).text
soup = BeautifulSoup(html, 'html.parser')

for idx, tag in enumerate(soup.select('a[href*=jsMemPop]'), 1):
    print('[{}] {}'.format(idx, tag.text))
```

# 이렇게 짤~~~~

- [1] 강길부
- [2] 강병원
- [3] 강석진
- [4] 강석호
- [5] 강창일
- [6] 강효상
- [7] 강훈식

중략 ...

- [293] 홍일표
- [294] 홍철호
- [295] 황영철
- [296] 황주홍
- [297] 황희



물론 다른 언어로도 할 수 있습니다만,

- nodejs
- java
- c++
- ruby

파이썬에는 ~~~

**Pandas/TensorFlow**  
가 있습니다.

여러분의 파이썬/장고 페이스메이커가 되겠습니다. / © AskDjango



## 다음 코드를 이어 적용하면 ...

```
names = []
for tag in soup.select('a[href*=jsMemPop]'):
    names.append(tag.text)

import pandas as pd

series = pd.Series(names)
series.str.slice(0, 1).value_counts().plot(kind='bar', figsize=(12, 3))
```

# 국회의원 성씨 분포를 그래프로 손쉽게 ;)

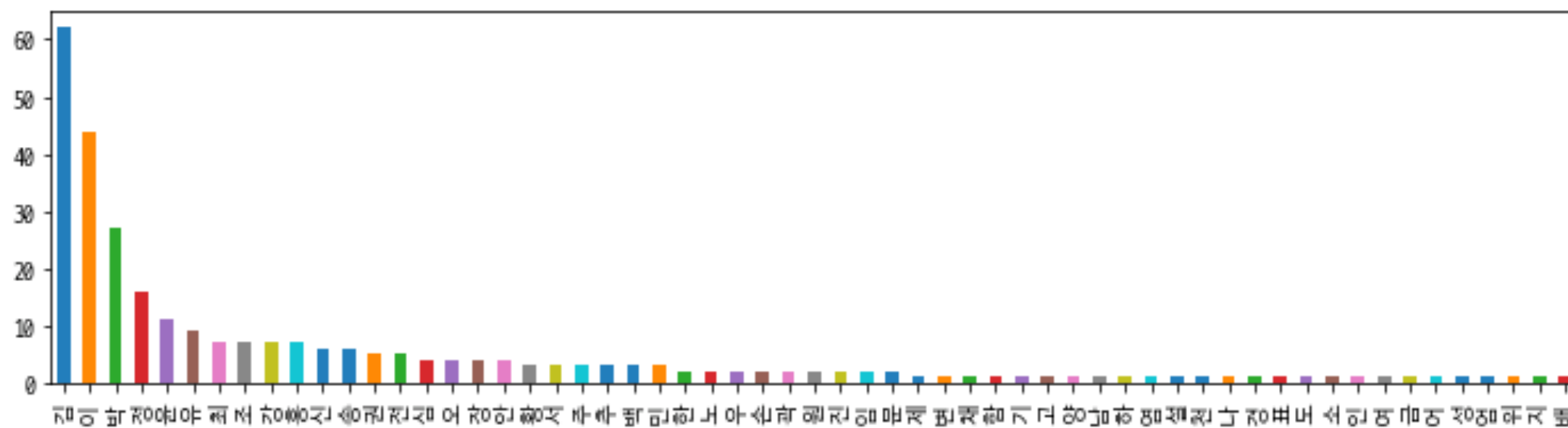
```
In [32]: import pandas as pd
```

```
series = pd.Series(names)
```

```
In [33]: %matplotlib inline
```

```
series.str.slice(0, 1).value_counts().plot(kind='bar', figsize=(12, 3))
```

```
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x11cc384a8>
```



# 요약

- 너무 부지런해지지마세요. 게으러지세요.
  - 게으른 자는 효율적으로 움직입니다.
- 그리고, 단순반복작업은 파이썬에게 양보하세요.

# 웹페이지 유형별 사례분석

# 웹페이지 크롤링은 크게 3가지

1. 단순 HTML 크롤링
2. Ajax 렌더링 크롤링
3. AngularJS, Vue.js, React.JS 류의 자바스크립트 렌더링 크롤링

# 파이썬으로 크롤링할 때, 주로 사용하는 라이브러리

- Requests : 파이썬에서 동작하는 작고 빠른 브라우저
  - 웹서버로부터 초기 HTML만 받을 뿐, 추가 CSS/JavaScript 처리 X
  - 거의 모든 플랫폼에서 구동 가능
- Selenium : 브라우저 X, 브라우저를 원격 컨트롤하는 테스트 라이브러리
  - Chrome, Firefox, IE, PhantomJS 등
  - 기존 브라우저를 사용하므로, 추가 CSS/JavaScript 처리 지원
  - 리소스를 많이 먹습니다. 사이트에 따라 죽기도 합니다.
- BeautifulSoup4 : HTML 파서
  - 지정 HTML로부터 원하는 위치/형식의 문자열을 획득
  - 주로 Requests에 의해 많이 사용되지만, Selenium에서도 사용할 수 있습니다.

모든 웹페이지는 사용된  
웹프론트엔드 기술과  
구현방식이 모두 모두 다릅니다.



다음 내용의 웹페이지가 있습니다.

# Hello, AskDjango!!!

크롤링으로 하고자 하는 일은 ... **h1** 태그의 text를 가져오는 것 !!!

훗, 쉽지.

나는 파이썬 크롤링을  
열심히 수련했다구. 다다닥 ~~~

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

```
html = requests.get('http://웹페이지주소').text  
soup = BeautifulSoup(html, 'html.parser')  
print(soup.find('h1').text)
```

## 그런데 ...

```
4 html = requests.get(url).text
5 soup = BeautifulSoup(html, 'html.parser')
----> 6 print(soup.find('h1').text)
```

AttributeError: 'NoneType' object has no attribute 'text'

h1 태그를 찾을 수 없어서

soup.find('h1')에서 None을 반환한 것입니다.

# 나의 예상, 페이지 소스

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>react helloworld</title>
  </head>
  <body>
    <h1>Hello, AskDjango!!!</h1>
  </body>
</html>
```

하지만, "페이지 소스"를 까보니 ... 엉 ???

**h1** 태그는 도대체 어디에 ???

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>react helloworld</title>
  </head>
  <body>
    <div id="app"></div>
    <script src="index.js" charset="utf-8"></script>
  </body>
</html>
```

그런데, 브라우저 개발자도구를 통해서, 분명히 **h1**이 있는데 ...

귀신이 곡할 노릇 ... :(



# 뭐가 다른 거지 ???

- "페이지 소스" 메뉴를 통해 본 HTML은 웹서버로부터 받은 최초의 **HTML**
  - 이 HTML 구조는 로딩 후에 변경될 수 있습니다. **by JavaScript**
- "브라우저 개발자도구"를 통해 확인한 구조
  - 최초의 구조로부터 JavaScript를 통해 변경된 구조

우리는 ReactJS라는 JavaScript 코드를 통해 내용이 그려진 웹페이지를 본 것 입니다.



# 그렇다면 어떻게 해야 ???

- 본 페이지는 requests로는 포기하세요.

# Selenium으로 해보시죠.<sup>1</sup>

feat. Firefox

```
from bs4 import BeautifulSoup
from selenium import webdriver

browser = webdriver.Firefox()
browser.get('http://웹페이지주소')

soup = BeautifulSoup(browser.page_source, 'html.parser')
print(soup.find('h1').text)          # 'Hello, AskDjango!!!'
```

어? 된다 !!!

---

<sup>1</sup> geckodriver 다운로드해서, 환경변수 PATH 상의 디렉토리로의 복사가 필요합니다.

크롤링할 때에는 무조건 **selenium**으로  
쓰면 되겠네요?

예. 그렇습니다.

아무 생각없이 쓰기에는  
**Selenium**이 아주 좋습니다.

# 다음 경우에 **Selenium**이 좋아요.

- 고민하기 싫다.
- 내가 요청할 웹페이지가 몇 개 안된다.
- GUI가 있는 컴퓨터에서 수행한다.<sup>2</sup>
- 컴퓨터 사양이 넉넉하다.
- 웹페이지가 JavaScript로 동작한다.

---

<sup>2</sup> 각 브라우저 별로 구동할 수 있는 환경의 제약이 있기 때문입니다.

# 바꿔말하면, 다음 경우에는 **Selenium**을 안 쓰고 싶지만 ...

- 요청할 웹페이지가 엄청 많고 ...
- 컴퓨터 사양이 넉넉하지 않고 ...
- 자바스크립트 동작이 필요없다면 ...

쓸 수만 있다면 Requests가 성능이 수백배 좋아요.

그래도, 고민하기 싫다면 Selenium을 쓰면 꾸역꾸역 돌아가겠죠.

가급적 **Requests**를 통해 처리할 수 있다면  
처리하는 것이 효율이 훨씬 좋습니다.



# 왜냐하면 ...

- 추가 이미지/JavaScript/CSS 등의 리소스를 로딩/실행하지 않기  
때문입니다.

뭘 쓸지 헛갈려요.  
정리해주세요요.

# 정리들어갑니다.

# 먼저 첫번째 기준

"페이지 소스" 메뉴로 봤을 때, 내가 원하는 콘텐츠가 있느냐?

1. 있다면 : Requests 쓰세요. 끝 !!!
2. 없다면 : 다음 페이지로 ...

## 두번째 기준

내가 원하는 콘텐츠가 별도 **Ajax**요청을 통해 받아오진 않는지?

이는 브라우저 개발자 도구의 Network 탭을 통해 확인 가능

1. 받아오고 있다면 : Requests로 받아오세요. 끝 !!!
2. 내역을 못 찾았다면 ... : 다음 페이지로 ...

# 세번째 기준

내가 원하는 콘텐츠가 별도 자바스크립트 로직을 통해 그려지는 지?

1. 자바스크립트 로직을 통해 그려진다면 ... 로직으로 파이썬 로직으로 변환가능한지 체크
  1. 변환가능하다면 ... 파이썬 로직으로 변경해서 처리. 끝.
  2. 불가하다면 ... : Selenium 쓰세요. 끝.
2. 모르겠거나, 그렇지 아니하다면 ... : Selenium 쓰세요. 끝.

# 위에서 나눈 분류를 다시 정리해보면 !!!

## 1. 단순 HTML 크롤링

- Requests 로 고고 !!!

## 2. Ajax 렌더링 크롤링

- Requests로 해볼 수 있으면 해보고, 어려우면 Selenium

## 3. AngularJS, Vue.js, React.JS 류의 자바스크립트 렌더링 크롤링

- Selenium 으로 고고 !
- 분석하기에 따라 requests로 가능할 수도 있습니다.

이도저도 생각하기 귀찮으면 **Selenium**

또 ... 우리 **ActiveX** 사이트가 있죠.

- Selenium으로 IE를 통해 자동화를 해볼 수는 있어요.
- 하지만, 매번 ActiveX 설치 이벤트까지 처리하기가 까다로워요.
- 가급적이면 피해가기.



# 요약

- 웹서비스를 만들어본 경험이 있다면, 그 인사이트를 크롤링에 적용해볼 수 있습니다.
- 크롤링은 서비스 개발자의 마음으로 웹페이지를 바라봐야 합니다.
- 그래야 보여요.

# Requests 크롤링 Tip

- 요청 시에 종종 추가 헤더 설정이 필요할 수 있습니다. 헤더를 설정하지 않으면 응답을 주지 않는 서비스가 있어요.
  - Referer
  - User-Agent
  - Accept-Language
- 모바일 페이지가 있다면 모바일 페이지를 크롤링하세요. HTML 마크업이 보다 심플합니다.

# 크롤링 라이브 데모

- 네이버 실시간 검색어 : 정적 HTML
- 멜론 검색 : Ajax 렌더링
- 또 ... 어떤 페이지가 궁금하나요?

# 장고를 활용하여 크롤링 데이터 저장 및 조회하면 만들기

# 크롤링한 데이터를 저장하고 싶어요.

- 파일에 저장하실 수 있어요. (CSV, JSON 포맷 등)
- 데이터베이스에 저장하실 수 있어요.
  - 클라우드 스토리지에 저장하거나 <sup>3</sup>
  - 로컬에서는 SQLite 데이터베이스가 쉬워요.
  - 파이썬은 SQLite를 기본지원합니다.

---

<sup>3</sup> Python Korea 2017년 12월 세미나 : [Azure Function 활용한 파이썬 크롤링 스케줄링](#) 참고

# 파이썬으로 SQLite3 DB에 데이터 저장하기

```
import sqlite3
conn = sqlite3.connect('assembly.db')

cursor = conn.cursor()

# Create table if not exists ...
cursor.execute('CREATE TABLE IF NOT EXISTS member(name text, code text)')

# Insert a row of data
members = [('홍길동', '12345'), ('김철수', '12346'), ('이영희', '12347')]
cursor.executemany("INSERT INTO member VALUES (?, ?)", members)

# Save (commit) the changes
conn.commit()

# Fetch data from Table
for row in cursor.execute('SELECT * FROM member'):
    print(row)

# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
conn.close()
```

잘 동작합니다만 ...

로직이 조금만 복잡해져도 @\_@!!!



중간에 다른 **DB**로 바꿀려면 @\_@!!!

**ORM을 써보려합니다.**

# 국내에서 유명한 파이썬 **ORM**

- Django Model
  - Django는 웹서비스 개발 목적으로만 쓴다는 선입견은 No No
  - 웹화면없이도 Model만 사용하실 수 있어요.
- SQLAlchemy

# 라이브 코딩쇼 Part 1

1. 장고 프로젝트 생성, superuser 계정 생성
2. 모델 생성/마이그레이션
3. 크롤링 결과를 모델을 통해 DB에 저장

# 라이브 코딩쇼 Part 2

1. 모델에 저장된 데이터를 웹을 통한 조회

2. 익명 댓글 구현하기

어? 국회의원 평가 서비스 ???

인생은 짧습니다.  
파이썬3와 장고를 쓰세요.

# 감사합니다.<sup>4</sup>

이진석 / [me@nomade.kr](mailto:me@nomade.kr)

---

<sup>4</sup> 크롤링/파이썬/장고 궁금증은 [AskDjango](#) 페이스북그룹에서 해결하세요.  
크롤링/파이썬/장고 학습은 [AskDjango VOD](#)를 통해 시작하세요.