

■ 크롤링(Crawling)

- 조직적 / 자동화된 방법으로 데이터를 탐색 / 수집 하는 것
- 데이터 수집 절차
 - 원하는 URL에 request를 보내고 결과를 받는다
 - 받은 결과물(HTML / JSON / XML)을 파싱(Parsing)한다
 - 필요한 정보만 추출한다
- 파이썬에서 크롤링을 하기 위해 필요한 라이브러리
 - 데이터 통신 : requests / urllib / urllib2
 - 파싱 : bs4 (BeautifulSoup)

■ 크롤링(Crawling)

● requests

```
import requests
```

● URL 호출 후 응답코드 확인

```
result = requests.get('http://ggoreb.com/http')  
result.status_code
```

200

● 응답결과 확인

```
result.text
```

```
"<html lang='ko'>\n<head>\n  <meta charset='utf-8' >\n</head>\n<body>\n  <h3>\n    <a href='#'>이웃을 돕는 방법</a>\n  </h3>\n</body>\n</html>"
```

● Encoding 처리

```
result.encoding = 'utf-8'  
result.text
```

```
"<html lang='ko'>\n<head>\n  <meta charset='utf-8' >\n</head>\n<body>\n  <h3>\n    <a href='#'>허니비</a>\n  </h3>\n</body>\n</html>"
```

■ 크롤링(Crawling)

● 공통으로 사용할 함수 작성 (인코딩 처리 / 응답결과 text)

```
def get_html(url):  
    html = ''  
    res = requests.get(url)  
    if res.status_code == 200:  
        res.encoding = 'utf-8'  
        html = res.text  
    return html
```

```
get_html('http://ggoreb.com/http')
```

```
"<html lang='ko'>\n<head>\n    <meta charset='utf-8' >\n</head>\n<body>\n<h3>\n    <a href='#'>허니비</a>\n    </h3>\n</body>\n</html>"
```

■ requests 모듈

● 기본 사용

```
import requests
res = requests.get('http://ggoreb.com/python/request.jsp')
print(res.status_code)
print(res.text)
```

200

```
method : GET<br>
query string<br>
<br><br>
header<br>
key : accept, value : */*<br>
key : Accept-Encoding, value : gzip, deflate<br>
key : connection, value : close<br>
key : host, value : ggoreb.com<br>
key : HOSTING_CONTINENT_CODE, value : AS<br>
key : HOSTING_COUNTRY_CODE, value : KR<br>
key : HOSTING_WHITE_IP, value : false<br>
key : user-agent, value : python-requests/2.22.0<br>
key : X-Forwarded-Proto, value : http<br>
key : X-SERVER_PORT, value : 80<br>
key : X-SERVER_PROTOCOL, value : HTTP/1.1<br>
key : X-SIMPLEXI, value : 14.138.9.145<br>
key : content-length, value : 0<br>
```

■ requests 모듈

● 파라미터 사용

```
import requests
param = { 'page': 1, 'search': '검색어' }
res = requests.get('http://ggoreb.com/python/request.jsp', params=param)
print(res.text)
```

```
method : GET<br>
query string<br>
key : search, value : 검색어<br>
key : page, value : 1<br>
<br><br>
header<br>
key : accept, value : */*<br>
key : Accept-Encoding, value : gzip, deflate<br>
key : connection, value : close<br>
key : host, value : ggoreb.com<br>
key : HOSTING_CONTINENT_CODE, value : AS<br>
key : HOSTING_COUNTRY_CODE, value : KR<br>
key : HOSTING_WHITE_IP, value : false<br>
key : user-agent, value : python-requests/2.22.0<br>
key : X-Forwarded-Proto, value : http<br>
key : X-SERVER_PORT, value : 80<br>
key : X-SERVER_PROTOCOL, value : HTTP/1.1<br>
key : X-SIMPLEXI, value : 14.138.9.145<br>
key : content-length, value : 0<br>
```

■ requests 모듈

● 헤더 사용

```
import requests
header = { 'user-agent': 'android', 'accept-language': 'en' }
res = requests.get('http://ggoreb.com/python/request.jsp', headers=header)
print(res.text)
```

```
method : GET<br>
query string<br>
<br><br>
header<br>
key : accept, value : */*<br>
key : Accept-Encoding, value : gzip, deflate<br>
key : accept-language, value : en<br>
key : connection, value : close<br>
key : host, value : ggoreb.com<br>
key : HOSTING_CONTINENT_CODE, value : AS<br>
key : HOSTING_COUNTRY_CODE, value : KR<br>
key : HOSTING_WHITE_IP, value : false<br>
key : user-agent, value : android<br>
key : X-Forwarded-Proto, value : http<br>
key : X-SERVER_PORT, value : 80<br>
key : X-SERVER_PROTOCOL, value : HTTP/1.1<br>
key : X-SIMPLEXI, value : 14.138.9.145<br>
key : content-length, value : 0<br>
```

■ Parsing (반복문 / find / indexing)

● 응답결과 확인

```
import requests
result = get_html('http://ggoreb.com/python/html/data1.html')
result
```

```
'<!DOCTYPE html>\n<html>\n<body>\n\n<h2>Basic HTML Table</h2>\n\n<table style\n=\n"width:100%">\n  <tr>\n    <th>Firstname</th>\n    <th>Lastname</th>\n    <th>Age</th>\n  </tr>\n  <tr>\n    <td>Jill</td>\n    <td>Smith</td>\n    <td>50</td>\n  </tr>\n  <tr>\n    <td>Eve</td>\n    <td>Jackson</td>\n    <td>94</td>\n  </tr>\n  <tr>\n    <td>John</td>\n    <td>Doe</td>\n    <td>80</td>\n  </tr>\n</table>\n\n</body>\n</html>\n'
```

Basic HTML Table

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

```
<body>
  <h2>Basic HTML Table</h2>
  <table style="width:100%" == $0
    <tbody>
      <tr>...</tr>
      <tr>
        <td>Jill</td>
        <td>Smith</td>
        <td>50</td>
      </tr>
      <tr>
        <td>Eve</td>
        <td>Jackson</td>
        <td>94</td>
      </tr>
      <tr>
        <td>John</td>
        <td>Doe</td>
        <td>80</td>
      </tr>
    </tbody>
  </table>
</body>
```

■ Parsing (반복문 / find / indexing)

● 원하는 위치를 찾은 후 문자열 잘라내기

```
s_idx = 0
e_idx = 0
while True:
    s_idx = result.find('<td>', e_idx)
    if s_idx == -1:
        break
    e_idx = result.find('</td>', s_idx)
    print(result[s_idx + 4 : e_idx])
```

```
Jill
Smith
50
Eve
Jackson
94
John
Doe
80
```


■ Parsing (반복문 / find / indexing)

● 응답결과 확인

```
import requests
result = get_html('http://ggoreb.com/python/html/data2.html')
result
```

```
'<!DOCTYPE html>\n<html>\n<head>\n<style>\n\ttable {\n\t\tfont-family: arial, sans-serif;\n\t\tborder-collapse: collapse;\n\t\twidth: 100%;\n\t}\n\ttd, th {\n\t\tborder: 1px solid #dddddd;\n\t\ttext-align: left;\n\t\tpadding: 8px;\n\t}\n\ttr:nth-child(even) {\n\t\tbackground-color: #dddddd;\n\t}\n</style>\n</head>\n<body>\n\n<h2>HTML Table</h2>\n\n<table>\n  <tr>\n    <th>Company</th>\n    <th>Contact</th>\n    <th>Country</th>\n  </tr>\n  <tr>\n    <td>Alfreds Futterkiste</td>\n    <td>Maria Anders</td>\n    <td>Germany</td>\n  </tr>\n  <tr>\n    <td>Centro comercial Moctezuma</td>\n    <td>Francisco Chang</td>\n    <td>Mexico</td>\n  </tr>\n  <tr>\n    <td>Ernst Handel</td>\n    <td>Roland Mendel</td>\n    <td>Austria</td>\n  </tr>\n  <tr>\n    <td>Island Trading</td>\n    <td>Helen Bennett</td>\n    <td>UK</td>\n  </tr>\n  <tr>\n    <td>Laughing Bacchus Winecellars</td>\n    <td>Yoshi Tannamuri</td>\n    <td>Canada</td>\n  </tr>\n  <tr>\n    <td>Magazzini Alimentari Riuniti</td>\n    <td>Giovanni Rovelli</td>\n    <td>Italy</td>\n  </tr>\n</table>\n\n</body>\n</html>\n'
```

■ Parsing (반복문 / find / indexing)

● 응답결과 확인

HTML Table

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

```
... ▼ <table> == $0
  ▼ <tbody>
    ▼ <tr>
      <th>Company</th>
      <th>Contact</th>
      <th>Country</th>
    </tr>
    ▼ <tr>
      <td>Alfreds Futterkiste</td>
      <td>Maria Anders</td>
      <td>Germany</td>
    </tr>
    ▶ <tr>...</tr>
    ▶ <tr>...</tr>
    ▶ <tr>...</tr>
    ▶ <tr>...</tr>
  </tbody>
</table>
```

■ Parsing (반복문 / find / indexing)

● 원하는 위치를 찾은 후 문자열 잘라내기

```
s_idx = 0
e_idx = 0
while True:
    s_idx = result.find('<td>', e_idx)
    if s_idx == -1:
        break
    e_idx = result.find('</td>', s_idx)
    print(result[s_idx + 4 : e_idx])
```

```
Alfreds Futterkiste
Maria Anders
Germany
Centro comercial Moctezuma
Francisco Chang
Mexico
Ernst Handel
Roland Mendel
Austria
Island Trading
Helen Bennett
UK
Laughing Bacchus Winecellars
Yoshi Tannamuri
Canada
Magazzini Alimentari Riuniti
Giovanni Rovelli
Italy
```

■ Parsing (BeautifulSoup)

● 네이버 주식 정보 코스피 / 코스닥 지수 추출

[←](#) [→](#) [↺](#) [🔒](#) [finance.naver.com](#) [☆](#) [🔍](#) [☰](#)

[\[금감원\] 전자공시시스템](#) [\[한국거래소\] 공매도 종합 포털](#) [\[한국은행\] 100대 통계지표](#) [\[뉴스\] 글로벌 경제 리포트](#) [투자자보호 해외동향 2018-9](#)

[최근조회종목](#)
MYSTOCK

최근조회종목이 없습니다.

"코스피, 상승 제한적 전망...종목별 ...

반등 조짐을 보이는 코스피지수가 2300선을 돌파할 지 시장의 관심이 쏠리고 있다. 신종국 리스크와 미중 무역분쟁 확대..

자진 상장폐지한다는 한국유리, 주가 급등 이유는 증시 위축? ... '고위험ELS' 쓸어담는 큰손

中과 수출경쟁업종, 무역전쟁이 기회?

'수출 모멘텀' ... 상승세 탄 방산株

[월가시각] "관세·기술주 등에 주목하며 신중 모드"

▶ [주요뉴스 더보기](#)

투자전략

[현문학 기자의 돈되는 중국경제] '개혁개방 ... 매일경제

불안한 외부환경보다 대내정책에서 투자기 ... 미래에셋대우

기술주 반등에 S&P 500 강보합 마감 한국투자증권

무역갈등 우려 지속, 증시 혼조, ETF 자산유, ... 하나금융투자

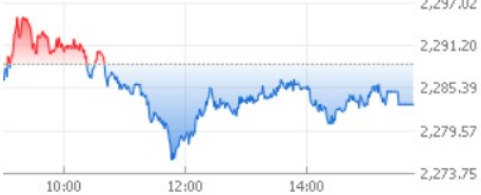
위험 확인 구간 신한금융투자

▶ [더보기](#)

오늘의증시

실시간 2018.09.11 장마감

코스피 **2,283.20** ▼5.46 -0.24% [↕](#)



2,297.02
2,291.20
2,285.39
2,279.57
2,273.75

10:00 12:00 14:00

개인 +1,926	외국인 -1,893	기관 -517	(억원)	
↑ 0	▲ 462	- 91	▼ 342	↓ 0

코스닥 **820.23** ▲4.07 +0.50% [↕](#)

코스피 200 **292.46** ▼1.24 -0.42% [↕](#)

해외증시

▶ [더보기](#)

다우산업(09, 10)	25,857.07	▼ 59.47
나스닥(09, 10)	7,924.16	▲ 21.62
홍콩H(09, 11)	10,359.05	▼ 74.57
상해종합(09, 11)	2,658.02	▼ 11.47


주식수수료

100년 무료

유관기관 제비용 제외 온라인 상속주

무조건 3년간

신용대출 **4.9%**



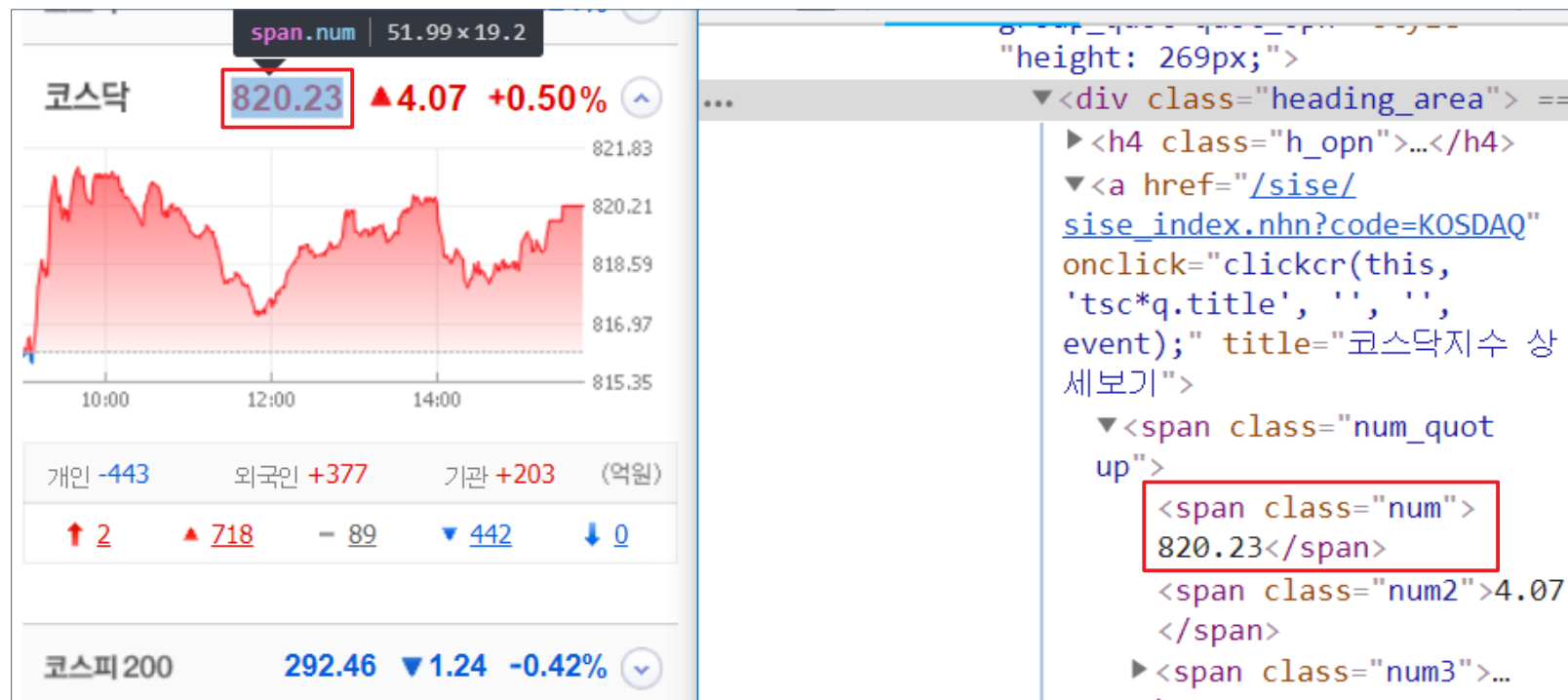
■ Parsing (BeautifulSoup)

● 네이버 주식 정보 코스피 / 코스닥 지수 추출



■ Parsing (BeautifulSoup)

● 네이버 주식 정보 코스피 / 코스닥 지수 추출



■ Parsing (BeautifulSoup)

● 응답결과 확인

```
import requests
result = get_html('https://finance.naver.com')
result
```

```
'<html lang="ko">\n <head> \n <title>?????</title> \n <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> \n <meta http-equiv="Content-Script-Type" content="text/javascript" /> \n <meta http-equiv="Content-Style-Type" content="text/css" /> \n <meta property="og:title" content="?????" /> \n <meta property="og:image" content="https://ssl.pstatic.net/static/m/stock/im/2016/08/og_stock-200.png" /> \n <meta property="og:url" content="https://finance.naver.com" /> \n <meta property="og:description" content="????? ???? , ???? ????g, ???? , ???? , ????â? ????g ???? ?????" /> \n <meta property="og:type" content="article" /> \n <meta property="og:article:thumbnailUrl" content="" /> \n <meta property="og:article:author" content="?? ?????" /> \n <meta property="og:article:author:url" content="http://FINANCE.NAVER.COM" /> \n <link rel="stylesheet" type="text/css" href="/css/finance_header.css?20191212151800" /> \n <link rel="stylesheet" type="text/css" href="/css/finance.css?20191212151800" /> \n <link rel="stylesheet" type="text/css" href="/css/newstock3.css?20191212151800" /> \n <script type="text/javascript" src="/js/jindo.min.ns.1.5.3.euckr.js?20191212151800"></script> \n <script type="text/javascript" src="/js/newstock3.js?20191212151800"></script>
```

■ Parsing (BeautifulSoup)

● get_html 함수 수정

```
def get_html(url):  
    html = ''  
    res = requests.get(url)  
    if res.status_code == 200:  
        res.encoding = None  
        html = res.text  
    return html
```

```
'<html lang="ko">₩₩ <head> ₩₩ <title>네이버 금융</title> ₩₩ <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> ₩₩ <meta http-equiv="Content-Script-Type" content="text/javascript" /> ₩₩ <meta http-equiv="Content-Style-Type" content="text/css" /> ₩₩ <meta property="og:title" content="네이버 금융" /> ₩₩ <meta property="og:image" content="https://ssl.pstatic.net/static/m/stock/im/2016/08/og_stock-200.png" /> ₩₩ <meta property="og:url" content="https://finance.naver.com" /> ₩₩ <meta property="og:description" content="국내 해외 증시 지수, 시장지표, 펀드, 뉴스, 증권사 리서치 등 제공" /> ₩₩ <meta property="og:type" content="article" /> ₩₩ <meta property="og:article:thumbnailUrl" content="" /> ₩₩ <meta property="og:article:author" content="네이버금융" /> ₩₩ <meta property="og:article:author:url" content="http://FINANCE.NAVER.COM" /> ₩₩ <link rel="stylesheet" type="text/css" href="/css/finance_header.css?20191212151800" /> ₩₩ <link rel="stylesheet" type="text/css" href="/css/finance.css?20191212151800" /> ₩₩ <link rel="stylesheet" type="text/css" href="/css/newstock3.css?20191212151
```


■ Parsing (BeautifulSoup) – select (select_one)

● CSS 선택자를 이용하여 요소 찾기

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(result, 'html.parser')
tags = soup.select('span.num')
num = tags[0].getText()
print(num)

num = tags[1].getText()
print(num)
```

```
2,203.71
647.62
```

● 반복문을 이용한 문자열 추출

```
for tag in tags:
    print(tag.getText())
```

```
2,203.71
647.62
294.87
```

■ Parsing (BeautifulSoup) – find (find_all)

● 태그명을 사용하여 원하는 요소 찾기

```
import requests
from bs4 import BeautifulSoup
res = requests.get('http://ggoreb.com/python/html/example.html')
soup = BeautifulSoup(res.text, 'html.parser')
print(soup.find('div'))
print(soup.find('div').getText())
```

```
<div>
<p>a</p>
<p>b</p>
<p>c</p>
</div>
```

a
b
c

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <div>
      <p>a</p>
      <p>b</p>
      <p>c</p>
    </div>
    <div class="ex_class">
      <p>d</p>
      <p>e</p>
      <p>f</p>
    </div>
    <div id="ex_id">
      <p>g</p>
      <p>h</p>
      <p>i</p>
    </div>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

■ Parsing (BeautifulSoup) – find (find_all)

● 태그명을 사용하여 원하는 요소 찾기

```
import requests
from bs4 import BeautifulSoup
res = requests.get('http://ggoreb.com/python/html/example.html')
soup = BeautifulSoup(res.text, 'html.parser')
print(soup.find_all('div'))
```

```
[<div>
<p>a</p>
<p>b</p>
<p>c</p>
</div>, <div class="ex_class">
<p>d</p>
<p>e</p>
<p>f</p>
</div>, <div id="ex_id">
<p>g</p>
<p>h</p>
<p>i</p>
</div>]
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <div>
      <p>a</p>
      <p>b</p>
      <p>c</p>
    </div>
    <div class="ex_class">
      <p>d</p>
      <p>e</p>
      <p>f</p>
    </div>
    <div id="ex_id">
      <p>g</p>
      <p>h</p>
      <p>i</p>
    </div>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

■ Parsing (BeautifulSoup) – find (find_all)

● id 속성을 사용하여 원하는 요소 찾기

```
soup.find('div', {'id': 'ex_id'})
```

```
soup.find(attrs={'id': 'ex_id'})
```

```
<div id="ex_id">  
<p>g</p>  
<p>h</p>  
<p>i</p>  
</div>
```

● class 속성을 사용하여 원하는 요소 찾기

```
soup.find(attrs={'class': 'ex_class'})
```

```
<div class="ex_class">  
<p>d</p>  
<p>e</p>  
<p>f</p>  
</div>
```

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Page title</title>  
  </head>  
  <body>  
    <div>  
      <p>a</p>  
      <p>b</p>  
      <p>c</p>  
    </div>  
    <div class="ex_class">  
      <p>d</p>  
      <p>e</p>  
      <p>f</p>  
    </div>  
    <div id="ex_id">  
      <p>g</p>  
      <p>h</p>  
      <p>i</p>  
    </div>  
    <h1>This is a heading</h1>  
    <p>This is a paragraph.</p>  
    <p>This is another paragraph.</p>  
  </body>  
</html>
```

■ Parsing (BeautifulSoup) – find (find_all)

● 그외 여러가지 속성을 사용하여 원하는 요소 찾기

```
import requests
from bs4 import BeautifulSoup
res = requests.get('http://ggoreb.com/python/html/example2.html')
soup = BeautifulSoup(res.text, 'html.parser')
```

– 개수 제한

```
print(soup.find_all('p', limit=2))
```

```
[<p class="desc">a</p>, <p>b</p>]
```

– Text 매칭

```
print(soup.find_all(string='a'))
```

```
['a']
```

– Text 매칭 (정규식)

```
import re
print(soup.find_all(string=re.compile('This.*')))
```

```
['This is a heading 1', 'This is a paragraph.', 'This is another paragraph.',
'This is a heading 2']
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <div>
      <p class='desc'>a</p>
      <p>b</p>
      <p data-role='click'>c</p>
    </div>
    <hr>
    <h1>This is a heading 1</h1>
    <p title='a'>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <h1 class='sub'>This is a heading 2</h1>
  </body>
</html>
```

■ Parsing (BeautifulSoup) – find (find_all)

● 그외 여러가지 속성을 사용하여 원하는 요소 찾기

– 찾은 요소의 지정 속성 확인

```
soup.find('h1', {'class': 'sub'})['class']
```

```
['sub']
```

– 찾은 요소의 모든 속성 확인

```
soup.find('p', {'data-role': 'click'}).attrs
```

```
{'data-role': 'click'}
```

– 찾은 요소의 다음 요소 확인

```
soup.find(attrs={'class': 'desc'}).find_next()
```


```
<p>b</p>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <div>
      <p class='desc'>a</p>
      <p>b</p>
      <p data-role='click'>c</p>
    </div>
    <hr>
    <h1>This is a heading 1</h1>
    <p title='a'>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <h1 class='sub'>This is a heading 2</h1>
  </body>
</html>
```

■ Parsing (BeautifulSoup)

● 네이버 웹툰 제목 추출

← → ↻ comic.naver.com/webtoon/list.nhn?titleId=557672 ☆ ⓘ ⚙









기괴괴괴

오성대

<절벽귀> 오성대 작가의 신작 옴니버스 미스터리 스

기묘하고 괴상한 이야기들.

[+ 관심웹툰](#) [첫화보기](#) [작가의 다른 작품](#) [❤](#)

이미지	제목
	 다음화를 미리 만나보
 <div>100.04 x 16</div> 229화 인간현관 #7	
	228화 인간현관 #6
	227화 인간현관 #5
	226화 인간현관 #4
	225화 인간현관 #3

DevTools - comic.naver.com/webtoon/list.nhn...

Elements Console Sources >>

```
class="viewList">
  ▶<caption>...</caption>
  ▶<colgroup>...</colgroup>
  ▶<thead>...</thead>
  <!-- 게임배너 or 광고배너 -->
  ▼<tbody>
    ▶<tr class="band_banner v2">...</tr>
    ▼<tr>
      ▶<td>...</td>
      ▼<td class="title">
        ...
        <a href="/webtoon/detail.nhn?titleId=557672&no=244&weekday=thu" onclick="clickcr(this,'lst.title','557672','244',event)">229화 인간현관 #7</a> == $0
      </td>
      ▶<td>...</td>
      <td class="num">2018.09.05</td>
    </tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
```

■ Parsing (BeautifulSoup)

● 네이버 웹툰 제목 추출

```
result = get_html('https://comic.naver.com/webtoon/list.nhn?titleId=557672')
parse = BeautifulSoup(result, 'html.parser')

titles = parse.select('td.title > a')
for title in titles:
    print(title.getText().replace('₩n', ''))
```

```
229화 인간현관 #7
228화 인간현관 #6
227화 인간현관 #5
226화 인간현관 #4
225화 인간현관 #3
224화 인간현관 #2
223화 인간현관 #1
222화 성장산 #3
221화 성장산 #2
220화 성장산 #1
```


■ Parsing (BeautifulSoup)

● 네이버 웹툰 제목 / 날짜 추출

9+	
별점	등록일
★★★★★ 9.94	2018.09.05
★★★★★ 9.87	2018.08.29
★★★★★ 9.93	2018.08.22
★★★★★ 9.95	2018.08.15
★★★★★ 9.96	2018.08.08

[illegible]

■ Parsing (BeautifulSoup)

● 네이버 웹툰 제목 / 날짜 추출

```
result = get_html('https://comic.naver.com/webtoon/list.nhn?titleId=557672')
table = parse.select_one('table.viewList')
trs = table.select('tr')
for tr in trs:
    title = tr.select_one('td.title > a')
    if title:
        print(title.get('href')) # print(title['href'])
        print(title.getText().replace('\n', ''), end=' // ')
    num = tr.select_one('.num')
    if num:
        print(num.getText())
```

```
/webtoon/detail.nhn?titleId=557672&no=244&weekday=thu
229화 인간현관 #7 // 2018.09.05
/webtoon/detail.nhn?titleId=557672&no=243&weekday=thu
228화 인간현관 #6 // 2018.08.29
/webtoon/detail.nhn?titleId=557672&no=242&weekday=thu
227화 인간현관 #5 // 2018.08.22
/webtoon/detail.nhn?titleId=557672&no=241&weekday=thu
226화 인간현관 #4 // 2018.08.15
/webtoon/detail.nhn?titleId=557672&no=240&weekday=thu
225화 인간현관 #3 // 2018.08.08
/webtoon/detail.nhn?titleId=557672&no=239&weekday=thu
```

■ Parsing (BeautifulSoup)

● XML 문서 정보 추출 - 1

- 데이터 로드 함수 작성

```
def get_xml(url):  
    xml = ''  
    resp = requests.get(url)  
    if resp.status_code == 200:  
        resp.encoding = None  
        xml = resp.text  
    return xml
```

- 데이터 파싱

```
from bs4 import BeautifulSoup  
  
result = get_xml('http://ggoreb.com/python/xml/data1.xml')  
  
parse = BeautifulSoup(result, 'xml')  
person = parse.find('person')  
number = person.find('number')  
name = person.find('name')  
print(number.text)  
print(name.text)
```

```
▼ <person>  
    <number>1</number>  
    <name>kim</name>  
</person>
```

■ Parsing (BeautifulSoup)

● XML 문서 정보 추출 - 2

- 데이터 파싱

```
from bs4 import BeautifulSoup

result = get_xml('http://ggoreb.com/python/xml/data2.xml')

parse = BeautifulSoup(result, 'xml')
persons = parse.findAll('person')
for person in persons:
    number = person.find('number')
    name = person.find('name')
    print(number.text)
    print(name.text)
```

```
1
kim
2
lee
3
park
```

```
▼ <persons>
  ▼ <person>
    <number>1</number>
    <name>kim</name>
  </person>
  ▼ <person>
    <number>2</number>
    <name>lee</name>
  </person>
  ▼ <person>
    <number>3</number>
    <name>park</name>
  </person>
</persons>
```

■ Parsing (BeautifulSoup)

● XML 문서 정보 추출 - 3

- 데이터 파싱 / 클래스 활용

```
from bs4 import BeautifulSoup

class Data:
    number = 0
    name = ''
    def __str__(self):
        return 'Data [number=' + str(self.number) + ', name=' + self.name + ']'

data_list = []
result = get_xml('http://ggoreb.com/python/xml/data2.xml')
parse = BeautifulSoup(result, 'xml')
persons = parse.findAll('person')
for person in persons:
    number = person.find('number')
    name = person.find('name')
    d = Data()
    d.number = number.text
    d.name = name.text
    data_list.append(d)

for data in data_list:
    print(data)
```

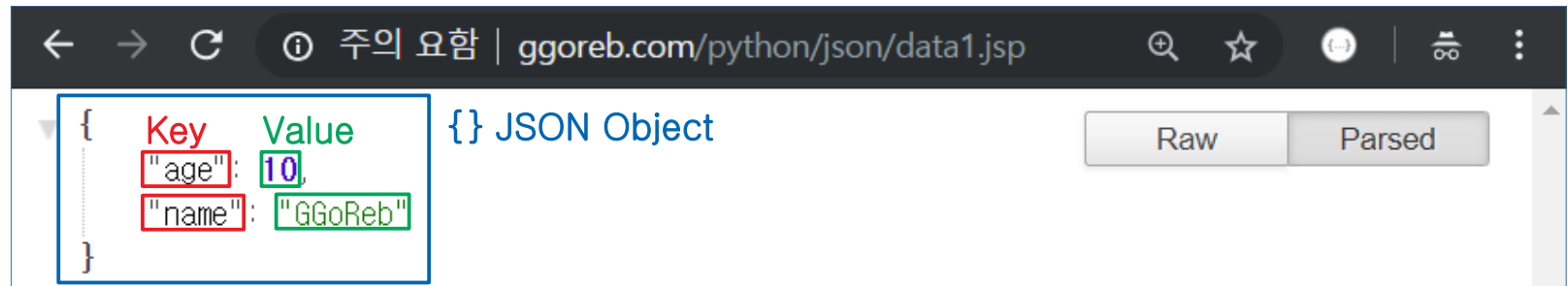
```
Data [number=1, name=kim]
Data [number=2, name=lee]
Data [number=3, name=park]
```

```
▼<persons>
  ▼<person>
    <number>1</number>
    <name>kim</name>
  </person>
  ▼<person>
    <number>2</number>
    <name>lee</name>
  </person>
  ▼<person>
    <number>3</number>
    <name>park</name>
  </person>
</persons>
```

■ Parsing (BeautifulSoup)

● JSON 정보 추출 - 1

- JSON 정보 확인 (JSON Object)



- 데이터 파싱

```
import requests

result = requests.get('http://ggoreb.com/python/json/data1.jsp')

parse = result.json() # JSON => Dictionary

print(type(parse))
print(parse['age'])
print(parse['name'])
```

```
<class 'dict'>
10
GGoReb
```

■ Parsing (BeautifulSoup)

● JSON 정보 추출 - 2

- JSON 정보 확인 (JSON Array)

The screenshot shows a web browser window with the address bar displaying 'ggoreb.com/python/json/data2.jsp'. The main content area shows a JSON array of three objects, each with 'age' and 'name' fields. The objects are highlighted in blue boxes and labeled as 'JSON Object'. The entire array is labeled as 'JSON Array'.

```
[  
  {  
    "age": 10,  
    "name": "A"  
  },  
  {  
    "age": 11,  
    "name": "B"  
  },  
  {  
    "age": 12,  
    "name": "C"  
  }  
]
```

Raw Parsed

{} JSON Object

{} JSON Object

{} JSON Object

[] JSON Array

■ Parsing (BeautifulSoup)

● JSON 정보 추출 - 2

- 데이터 파싱

```
import requests

result = requests.get('http://ggoreb.com/python/json/data2.jsp')

parse = result.json()  # JSON => List

print(type(parse))
for data in parse:
    age = data.get('age')
    name = data.get('name')
    print(age, name)
```

```
<class 'list'>
```

```
10 A
```

```
11 B
```

```
12 C
```


■ Parsing (BeautifulSoup)

● JSON 정보 추출 - 3

- JSON 정보 확인 (JSON Array)

Browser address bar: < > ↻ ⓘ 주의 요함 | ggoreb.com/python/json/data3.jsp 🔍 ☆ ⌂ ⋮

Buttons: Raw Parsed

```
[
  {
    "address": [
      "서울",
      "신림"
    ],
    "age": 10,
    "name": "A"
  },
  { ... }, // 3 items
  { ... } // 3 items
]
```

Annotations:

- [] JSON Array (pointing to the "address" array)
- { } JSON Object (pointing to the first object)
- [] JSON Array (pointing to the entire array structure)

■ Parsing (BeautifulSoup)

● JSON 정보 추출 - 3

- 데이터 파싱

```
import requests

result = requests.get('http://ggoreb.com/python/json/data3.jsp')

parse = result.json()

for data in parse:
    address = data.get('address')
    separator = ''
    for add in address:
        print(separator + add, end='')
        separator = ', '
    print()
    age = data.get('age')
    print(age)
    name = data.get('name')
    print(name)
```

서울, 신림

10

A

대전, 탄방

11

B

부산, 해운대

12

C

■ Selenium

- 브라우저 자동화 라이브러리
- 브라우저를 제어 할 수 있는 웹 드라이버를 통해 매크로와 같은 기능 사용
- 지원 브라우저 : Chrome, Fire Fox, Edge, Safari 등
- 지원 언어 : Python, Java, C#, JavaScript 등
- 반드시 사용해야 하는 경우
 - 자바스크립트를 통해 웹 페이지의 내용을 동적으로 생성하는 사이트
 - 프로그래밍 언어의 코드를 이용한 접속이 아닌 실제 브라우저를 통해서만 접속을 허용하는 사이트

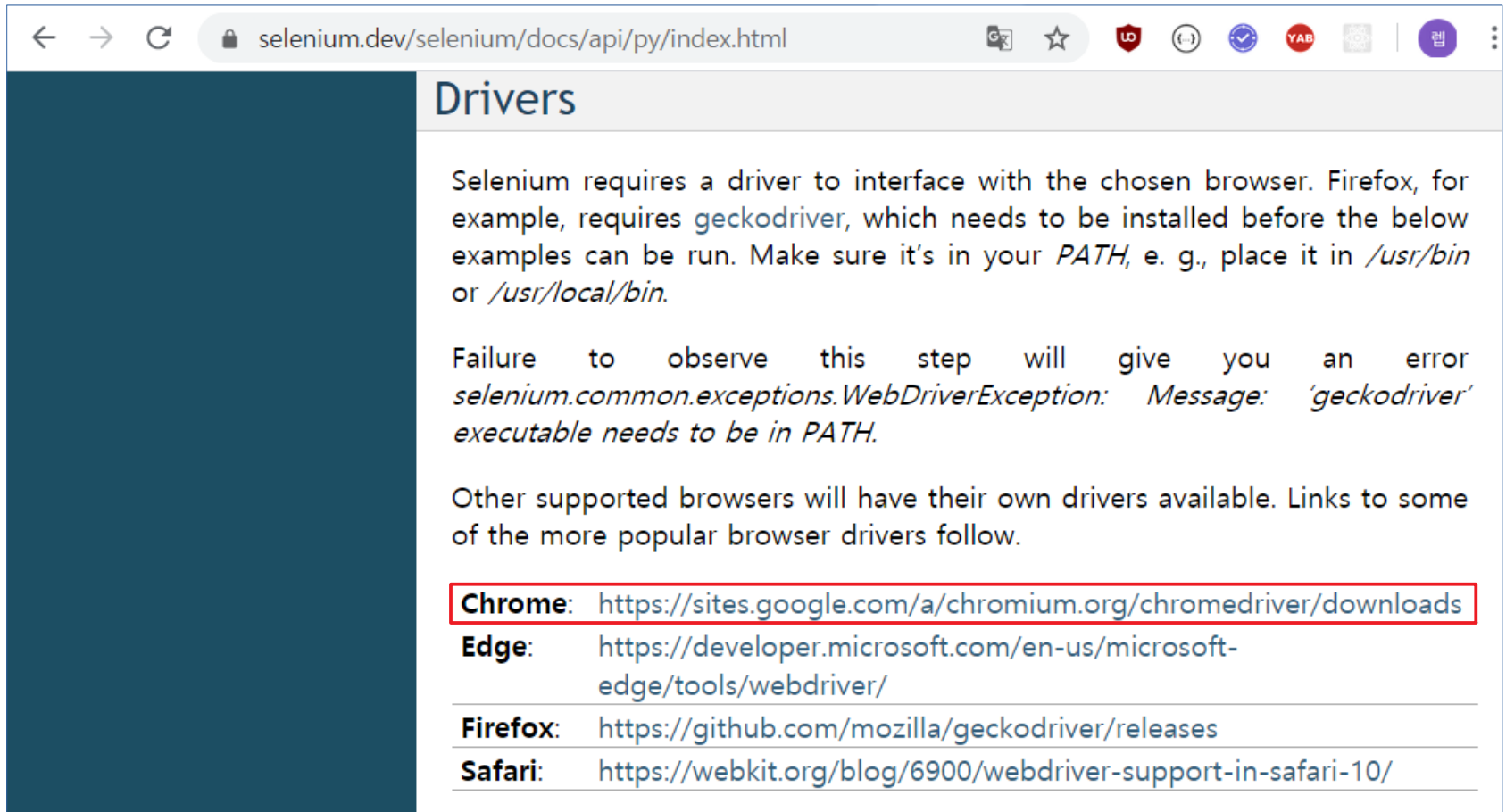
ex) 네이버 뽐 : <http://m.bbboom.naver.com>

리니지M 공지사항 : <https://lineagem.plaync.com/board/notice/list>

■ Selenium – Web Driver 설치

● Selenium 공식 사이트

– <https://selenium.dev/selenium/docs/api/py/index.html>



Drivers

Selenium requires a driver to interface with the chosen browser. Firefox, for example, requires [geckodriver](#), which needs to be installed before the below examples can be run. Make sure it's in your *PATH*, e. g., place it in */usr/bin* or */usr/local/bin*.

Failure to observe this step will give you an error *selenium.common.exceptions.WebDriverException: Message: 'geckodriver' executable needs to be in PATH*.

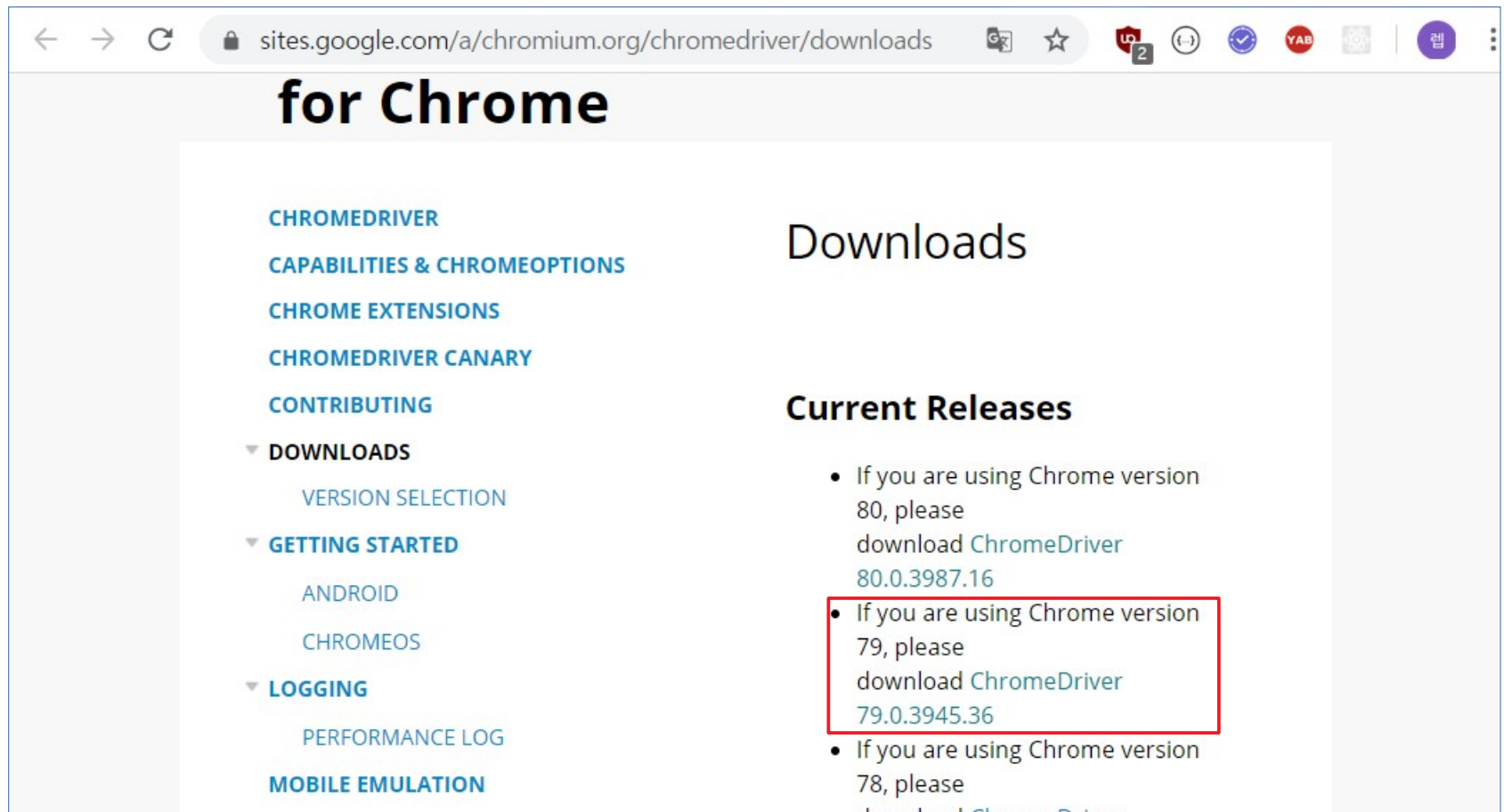
Other supported browsers will have their own drivers available. Links to some of the more popular browser drivers follow.

Chrome:	https://sites.google.com/a/chromium.org/chromedriver/downloads
Edge:	https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
Firefox:	https://github.com/mozilla/geckodriver/releases
Safari:	https://webkit.org/blog/6900/webdriver-support-in-safari-10/

■ Selenium 설치

● Web Driver 다운로드 사이트

– <https://sites.google.com/a/chromium.org/chromedriver/downloads>



The screenshot shows a web browser window with the address bar displaying sites.google.com/a/chromium.org/chromedriver/downloads. The page content is for Chromedriver downloads for Chrome. On the left is a navigation menu with links: CHROMEDRIVER, CAPABILITIES & CHROMEOPTIONS, CHROME EXTENSIONS, CHROMEDRIVER CANARY, CONTRIBUTING, DOWNLOADS (expanded), GETTING STARTED (expanded), LOGGING, and MOBILE EMULATION. Under DOWNLOADS are links for VERSION SELECTION, ANDROID, and CHROMEOS. Under GETTING STARTED is a link for PERFORMANCE LOG. The main content area has a heading "Downloads" and a section "Current Releases" with a bulleted list of instructions for different Chrome versions. The second bullet point, for Chrome version 79, is highlighted with a red box.

for Chrome

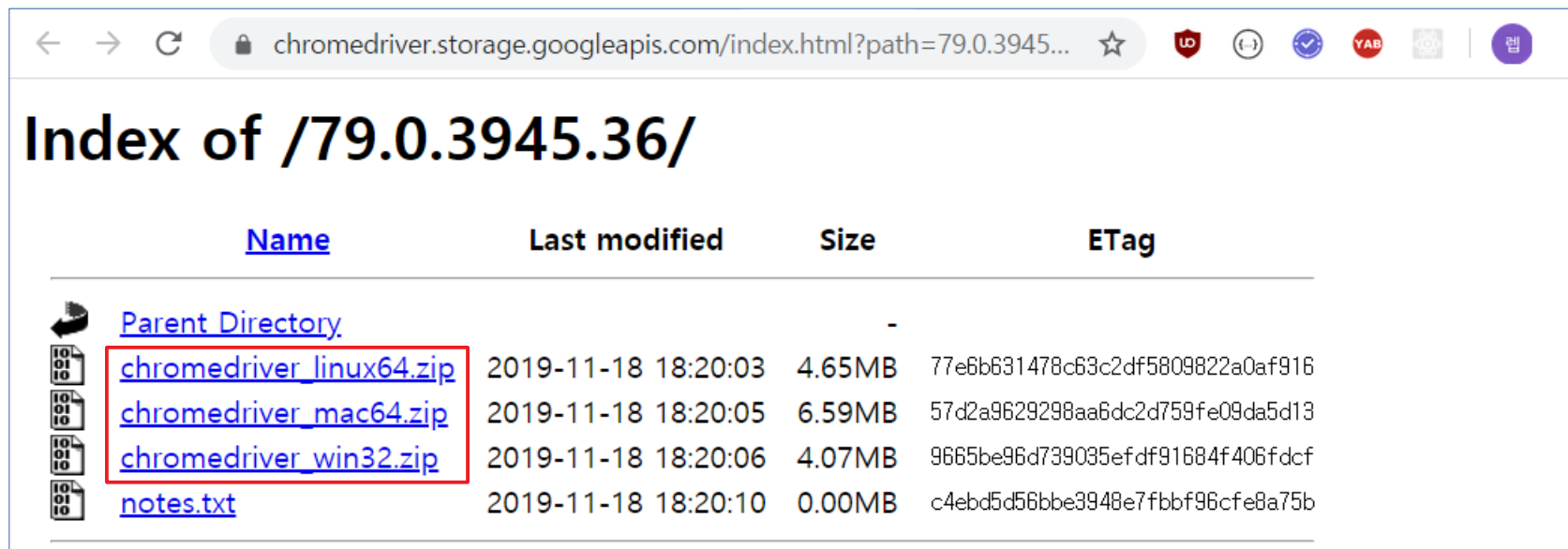
Downloads

Current Releases

- If you are using Chrome version 80, please download [ChromeDriver 80.0.3987.16](#)
- If you are using Chrome version 79, please download [ChromeDriver 79.0.3945.36](#)
- If you are using Chrome version 78, please download [ChromeDriver 78.0.3930.162](#)






■ Selenium 설치

● Web Driver 다운로드

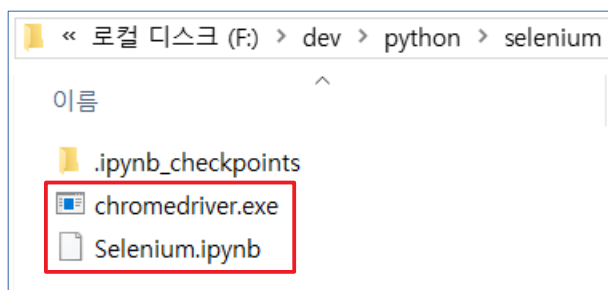


← → ↻ chromedriver.storage.googleapis.com/index.html?path=79.0.3945... ☆ 🔒 (1) ✓ YAB

Index of /79.0.3945.36/



Name	Last modified	Size	ETag
 Parent Directory		-	
 chromedriver_linux64.zip	2019-11-18 18:20:03	4.65MB	77e6b631478c63c2df5809822a0af916
 chromedriver_mac64.zip	2019-11-18 18:20:05	6.59MB	57d2a9629298aa6dc2d759fe09da5d13
 chromedriver_win32.zip	2019-11-18 18:20:06	4.07MB	9665be96d739035efdf91684f406fdcf
 notes.txt	2019-11-18 18:20:10	0.00MB	c4ebd5d56bbe3948e7fbbf96cfe8a75b

– 코드를 작성중인 경로로 복사



« 로컬 디스크 (F:) > dev > python > selenium

이름

- .ipynb_checkpoints
-  chromedriver.exe
-  Selenium.ipynb

■ Selenium 설치

● Selenium 다운로드

- pip install selenium
- conda install selenium

```
F:\dev\python\selenium>conda install selenium
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\GGoReb\Miniconda3

  added / updated specs:
    - selenium

The following NEW packages will be INSTALLED:

  selenium                    pkgs/main/win-64::selenium-3.141.0-py37he774
522_0

Proceed ([y]/n)?
```

■ Selenium 활용

● <http://ggoreb.com/http/wait.jsp>



The screenshot shows a web browser window with the address bar displaying `http://ggoreb.com/http/wait.jsp`. The page content on the left lists:

- 내용
- 5초 뒤 추가 내용
- 10초 뒤 추가 내용

The developer tools console on the right shows the HTML structure of the page:

```
...<!doctype html> == $0
<html>
  <head>...</head>
  <body>
    <script>...</script>
    <p>내용</p>
    <p>5초 뒤 추가 내용</p>
    <p>10초 뒤 추가 내용</p>
  </body>
</html>
```

– p 태그 확인

```
import requests
from bs4 import BeautifulSoup
result = requests.get('http://ggoreb.com/http/wait.jsp')
soup = BeautifulSoup(result.text, 'html.parser')

print(len(soup.find_all('p'))) # → 0
```


■ Selenium 활용

● 웹 드라이버 로드

```
from selenium import webdriver as wd  
  
driver = wd.Chrome(executable_path='chromedriver.exe')
```

– 필요시 여러 옵션 사용 가능 (proxy, user-agent, image 생략 등)

● 웹 사이트 접속

```
driver.get('http://ggoreb.com/http/wait.jsp')
```

● 브라우저에 표시된 요소 찾기

```
print(driver.find_elements_by_tag_name('p'))
```

```
[<selenium.webdriver.remote.webelement.WebElement (session="2d31408a49702047badcb960b1c7e65f", element="5c052df8-a7c1-4486-b32d-7bb6139a9644")>, <selenium.webdriver.remote.webelement.WebElement (session="2d31408a49702047badcb960b1c7e65f", element="6712a066-80dd-43ee-afcb-0f05a17779ed")>, <selenium.webdriver.remote.webelement.WebElement (session="2d31408a49702047badcb960b1c7e65f", element="58f88c54-6420-44eb-831a-2ba785c74fe4")>]
```

● 한개 요소 지정 후 내용 확인

```
print(driver.find_elements_by_tag_name('p')[2].text)
```

■ Selenium 활용

● 코드가 한꺼번에 실행되는 경우

- 자바스크립트로 생성되는 요소를 찾는 경우 오류 발생

```
import requests
from bs4 import BeautifulSoup
from selenium import webdriver as wd

driver = wd.Chrome(executable_path='chromedriver.exe')
driver.get('http://ggoreb.com/http/wait.jsp')
print(driver.find_elements_by_tag_name('p'))
print(driver.find_elements_by_tag_name('p')[2].text)
```

```
[<selenium.webdriver.remote.webelement.WebElement (session="900d093e5346fbb6bdcd33ce56976fea", element="c529d1dc-bc2a-43f8-92ce-49388e51e7a9")>]
```

IndexError Traceback (most recent call last)

```
<ipython-input-20-9623e5757242> in <module>
      8 driver.get('http://ggoreb.com/http/wait.jsp')
      9 print(driver.find_elements_by_tag_name('p'))
--> 10 print(driver.find_elements_by_tag_name('p')[2].text)
```

IndexError: list index out of range

■ wait

- 브라우저에 HTML 요소가 보여진 후 코드 실행
- Explicit Waits (명시적 대기)
 - 지정된 요소가 발견될 때까지 대기
 - 지정된 시간을 초과하는 경우 예외 발생
- Implicit Waits (암시적 대기)
 - 브라우저에 기본 요소(DOM)가 다 로드될 때까지 대기
 - 지정된 시간보다 일찍 로드가 된 경우 먼저 진행
 - 지정된 시간을 초과해도 모든 요소가 로드될 때까지 대기

■ wait – Explicit(명시적)

● 웹 드라이버 로드

```
from selenium import webdriver as wd  
driver = wd.Chrome(executable_path='chromedriver.exe')
```

● 필요 모듈 로드

```
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected_conditions as EC
```

● 웹 사이트 접속

```
driver.get('http://ggoreb.com/http/wait.jsp')
```

● 브라우저에 표시된 요소 찾기

```
print(driver.find_elements_by_tag_name('p'))
```

● 요소가 발견될 때까지 대기 후 찾기

```
try :  
    element = WebDriverWait(driver, 10).until(  
        EC.presence_of_element_located((By.CSS_SELECTOR, 'p:nth-of-type(2)'))  
    )  
    print(driver.find_elements_by_tag_name('p')[1].text)  
except Exception as e:  
    print('오류 발생', e)
```

■ wait – Implicit(암시적)

● 웹 드라이버 로드

```
from selenium import webdriver as wd
driver = wd.Chrome(executable_path='chromedriver.exe')

from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

driver.get('http://ggoreb.com/http/wait.jsp')

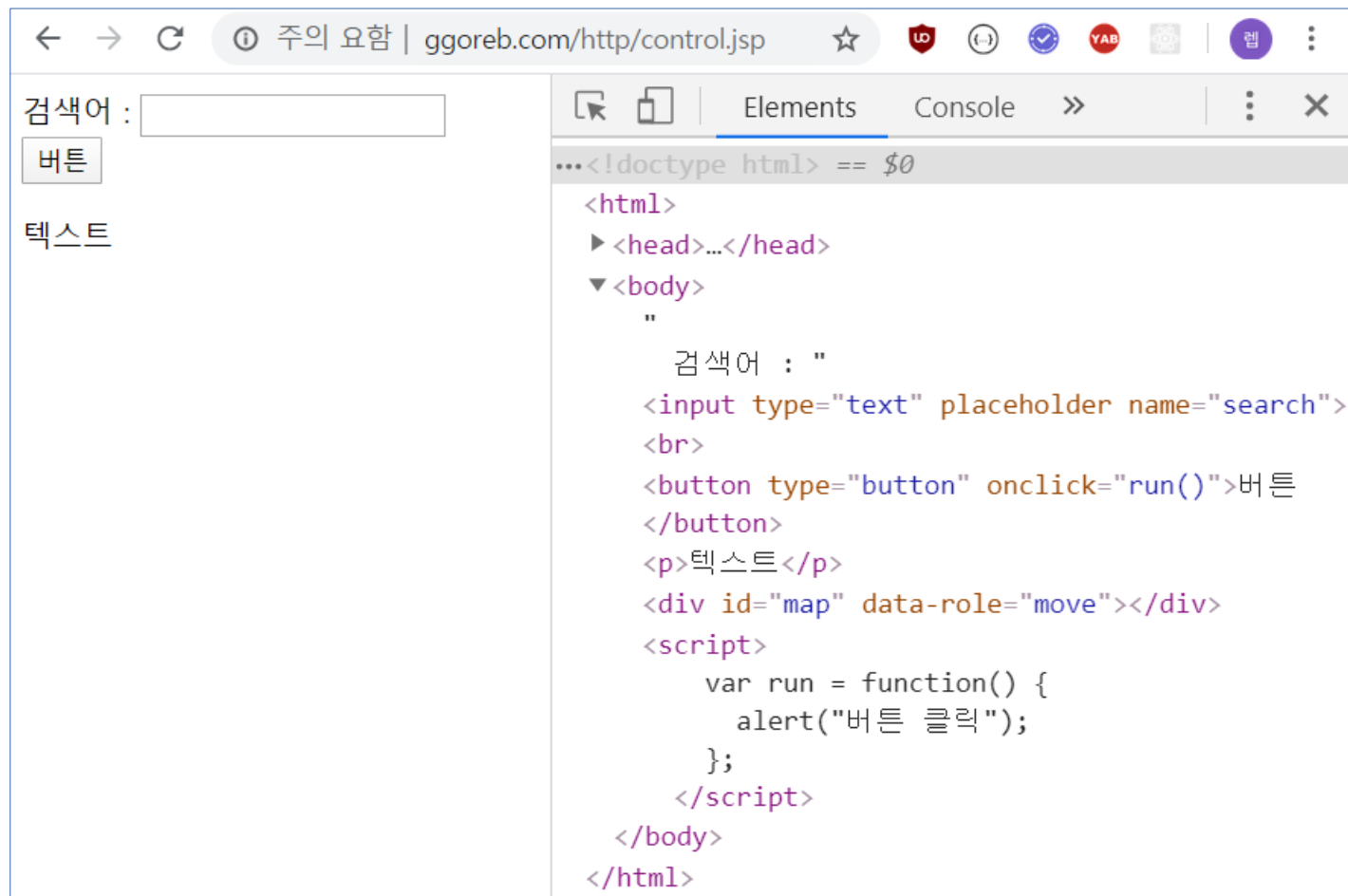
driver.implicitly_wait(10)
print(driver.find_elements_by_tag_name('p')[1].text)
```

```
IndexError                                Traceback (most recent call last)
<ipython-input-11-7f7a5ef18559> in <module>
      9
     10 driver.implicitly_wait(10)
--> 11 print(driver.find_elements_by_tag_name('p')[1].text)

IndexError: list index out of range
```

■ 요소 제어

● <http://ggoreb.com/http/control.jsp>



The screenshot shows a web browser window with the address bar displaying `http://ggoreb.com/http/control.jsp`. The page content includes a search form with a text input field labeled "검색어 :", a button labeled "버튼", and a text area labeled "텍스트". The browser's developer tools are open, showing the HTML source code. The code defines a search form with a text input, a button, and a text area, along with a JavaScript function `run()` that triggers an alert when the button is clicked.

```
...<!doctype html> == $0
<html>
  <head>...</head>
  <body>
    "
    검색어 : "
    <input type="text" placeholder name="search">
    <br>
    <button type="button" onclick="run()">버튼
    </button>
    <p>텍스트</p>
    <div id="map" data-role="move"></div>
    <script>
      var run = function() {
        alert("버튼 클릭");
      };
    </script>
  </body>
</html>
```

■ 요소 제어 – send_keys() / click()

● 아이디 / 비밀번호 / 검색어 등 데이터 입력 – send_keys()

```
driver.find_element_by_tag_name('input').send_keys('검색어 입력')
```

● 버튼 등 요소 클릭 – click()

```
driver.find_element_by_tag_name('button').click()
```



■ 요소 제어 – send_keys() / click() / execute_script()

● 요소의 텍스트 가져오기 – text

```
print(driver.find_element_by_tag_name('p').text)
```

텍스트

● 요소의 속성 가져오기 – get_attribute()

```
print(driver.find_element_by_tag_name('div').get_attribute('data-role'))  
print(driver.find_element_by_id('map').get_attribute('data-role'))
```

move
move

● 자바스크립트 실행 – execute_script()

```
driver.execute_script('alert("run script")')
```

