

17장 웹 스크레이핑

- 신문이나 잡지에서 원하는 글이나 사진을 오려서 모아두는 활동을 스크랩이라고 한다.
- 웹 스크레이핑(Web scraping)은 컴퓨터 소프트웨어 기술을 활용해 웹 사이트 내에 있는 정보를 추출하는 것이다.

17.1 웹 브라우저로 웹 사이트 접속하기

17.1.1 하나의 웹 사이트에 접속하기

- webbrowser 모듈을 이용해 웹 사이트에 접속한다.

[ch17_web scraping/ex01_webbrowser.py]

```
01 import webbrowser
02
03 url = 'www.naver.com'
04 webbrowser.open(url)
05
06
07 naver_search_url = "http://search.naver.com/search.naver?query="
08 search_word = '파이썬'
09 url = naver_search_url + search_word
10
11 webbrowser.open_new(url)
12
13
14 google_url = "https://www.google.com/search?q="
15 search_word = 'python'
16 url = google_url + search_word
17
18 webbrowser.open_new(url)
```

// 명령 프롬프트로 실행한다.

D:\dev\workspace\python\ch17_web scraping>python ex01_webbrowser.py

17.1.2 여러 개의 웹 사이트에 접속하기

- 한 번에 여러 개의 웹 사이트에 접속하려면 url 주소 리스트와 for 문을 이용하면 된다.

[ch17_web scraping/ex02_open_new.py]

```
01 import webbrowser
02
03 urls = ['www.naver.com', 'www.daum.net', 'www.google.com']
04
05 for url in urls:
06     webbrowser.open_new(url)
07
08
09 google_url = "www.google.com/search?q="
10 search_words = ['python web scraping', 'python webbrowser']
```

```

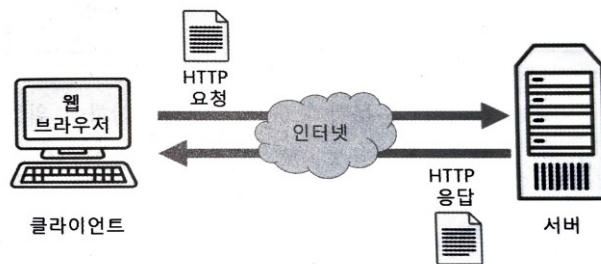
11
12 for search_word in search_words:
13     webbrowser.open_new(google_url + search_word)

```

17.2 웹 스크레이핑을 위한 기본 지식

17.2.1 데이터의 요청과 응답 과정

- 웹 브라우저를 통해 웹 사이트의 데이터를 가져오는 과정



17.2.2 HTML의 기본 구조

- 아래 두개 html 파일의 소스코드는 다르지만 출력 결과는 같다.

[ch17_webscraping/ex03_HTML_example.html]

```

01 <!doctype html>
02 <html>
03 <head>
04 <meta charset="utf-8">
05 <title>이것은 HTML 예제</title>
06 </head>
07 <body>
08 <h1>출간된 책 정보</h1>
09 <p id="book_title">이해가 쏙쏙 되는 파이썬</p>
10 <p id="author">홍길동</p>
11 <p id="publisher">위키북스 출판사</p>
12 <p id="year">2018</p>
13 </body>
14 </html>

```

[ch17_webscraping/ex04_HTML_example2.html]

```

01 <!doctype html>
02 <html>
03 <head>
04 <meta charset="utf-8">
05 <title>이것은 HTML 예제</title>
06 </head>
07 <body>
08 <h1>출간된 책 정보</h1>
09 <p>이해가 쏙쏙 되는 파이썬</p>
10 <p>홍길동</p>
11 <p>위키북스 출판사</p>
12 <p>2018</p>

```

```
13 </body>
14 </html>
```

17.2.3 웹 페이지의 HTML 소스 갖고 오기

- 웹 페이지의 소스코드를 가져오는 방법을 알아본다.

[ch17_web scraping/ex05_requests_get.py]

```
01 import requests
02
03 r = requests.get("https://www.google.co.kr")
04 print(r)
05 # <Response [200]>
06
07 print(r.text[0:100])
08 '''
09 <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ko"><head><meta content
10 '''
11
12 html = requests.get("https://www.google.co.kr").text
13 print(html[0:100])
14 '''
15 <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ko"><head><meta content
16 '''
```

17.2.4 HTML 소스코드를 분석하고 처리하기

- HTML 코드를 분석해 원하는 데이터를 추출하는 작업을 파싱(Parsing)이라고 한다. BeautifulSoup 라이브러리를 이용하면 HTML 소스를 파싱하고 태그나 속성을 통해 원하는 데이터를 추출할 수 있다.

(1) 데이터 찾고 추출하기

[ch17_web scraping/ex06_soup.py]

```
01 from bs4 import BeautifulSoup
02
03 # 테스트용 html 코드
04 html = """<html><body><div><span>\
05     <a href=http://www.naver.com>naver</a>\
06     <a href=https://www.google.com>google</a>\
07     <a href=http://www.daum.net/>daum</a>\
08     </span></div></body></html>"""
09
10 # BeautifulSoup를 이용해 HTML 소스를 파싱
11 soup = BeautifulSoup(html, 'lxml')
12 print(soup)
13 '''
14 <html><body><div><span>                <a                href="http://www.naver.com">naver</a>                <a
15 href="https://www.google.com">google</a>                <a                href="http://www.daum.net/">daum</a>
16 </span></div></body></html>
17 '''
18
```

```

19 print(soup.prettify())
20 '''
21 <html>
22 <body>
23 <div>
24 <span>
25 <a href="http://www.naver.com">
26     naver
27 </a>
28 <a href="https://www.google.com">
29     google
30 </a>
31 <a href="http://www.daum.net/">
32     daum
33 </a>
34 </span>
35 </div>
36 </body>
37 </html>
38 '''
39
40 print(soup.find('a'))
41 # <a href="http://www.naver.com">naver</a>
42
43 print(soup.find('a').get_text())
44 # naver
45
46 print(soup.find_all('a'))
47 '''
48 [<a href="http://www.naver.com">naver</a>, <a href="https://www.google.com">google</a>, <a
49 href="http://www.daum.net/">daum</a>]
50 '''
51
52 site_names = soup.find_all('a')
53 for site_name in site_names:
54     print(site_name.get_text())
55 ...
56 naver
57 google
58 daum
59 ...
60
61
62 # 테스트용 HTML 코드
63 html2 = """
64 <html>
65 <head>
66 <title>작품과 작가 모음</title>
67 </head>
68 <body>
69 <h1>책 정보</h1>
70 <p id="book_title">토지</p>
71 <p id="author">박경리</p>
72
73 <p id="book_title">태백산맥</p>
74 <p id="author">조정래</p>
75
76 <p id="book_title">감옥으로부터의 사색</p>
77 <p id="author">신영복</p>
78 </body>
79 </html>
80 """
81
82 soup2 = BeautifulSoup(html2, "lxml")
83 print(soup2.title)

```

```

84 # <title>작품과 작가 모음</title>
85
86 print(soup2.body)
87 '''
88 <body>
89 <h1>책 정보</h1>
90 <p id="book_title">토지</p>
91 <p id="author">박경리</p>
92 <p id="book_title">태백산맥</p>
93 <p id="author">조정래</p>
94 <p id="book_title">감옥으로부터의 사색</p>
95 <p id="author">신영복</p>
96 </body>
97 '''
98
99 print(soup2.body.h1)
100 # <h1>책 정보</h1>
101
102 print(soup2.find_all('p'))
103 '''
104 [<p id="book_title">토지</p>, <p id="author">박경리</p>, <p id="book_title">태백
105 산맥</p>, <p id="author">조정래</p>, <p id="book_title">감옥으로부터의 사색</p>,
106 <p id="author">신영복</p>]
107 '''
108
109 print(soup2.find('p', {"id": "book_title"}))
110 # <p id="book_title">토지</p>
111
112 print(soup2.find('p', {"id": "author"}))
113 # <p id="author">박경리</p>
114
115 print(soup2.find_all('p', {"id": "book_title"}))
116 '''
117 [<p id="book_title">토지</p>, <p id="book_title">태백산맥</p>, <p id="book_title">감옥으로부터의 사색
118 </p>]
119 '''
120
121 print(soup2.find_all('p', {"id": "author"}))
122 '''
123 [<p id="author">박경리</p>, <p id="author">조정래</p>, <p id="author">신영복</p>]
124 '''
125
126 soup2 = BeautifulSoup(html2, "lxml")
127
128 book_titles = soup2.find_all('p', {"id": "book_title"})
129 authors = soup2.find_all('p', {"id": "author"})
130
131 for book_title, author in zip(book_titles, authors):
132     print(book_title.get_text() + '/' + author.get_text())
133 '''
134 토지/박경리
135 태백산맥/조정래
136 감옥으로부터의 사색/신영복
137 '''
138
139 print(soup2.select('body h1'))
140 # [<h1>책 정보</h1>]
141
142 print(soup2.select('body p'))
143 # print(soup2.select('p'))
144 '''
145 [<p id="book_title">토지</p>, <p id="author">박경리</p>, <p id="book_title">태백
146 산맥</p>, <p id="author">조정래</p>, <p id="book_title">감옥으로부터의 사색</p>,
147 <p id="author">신영복</p>]
148 '''

```

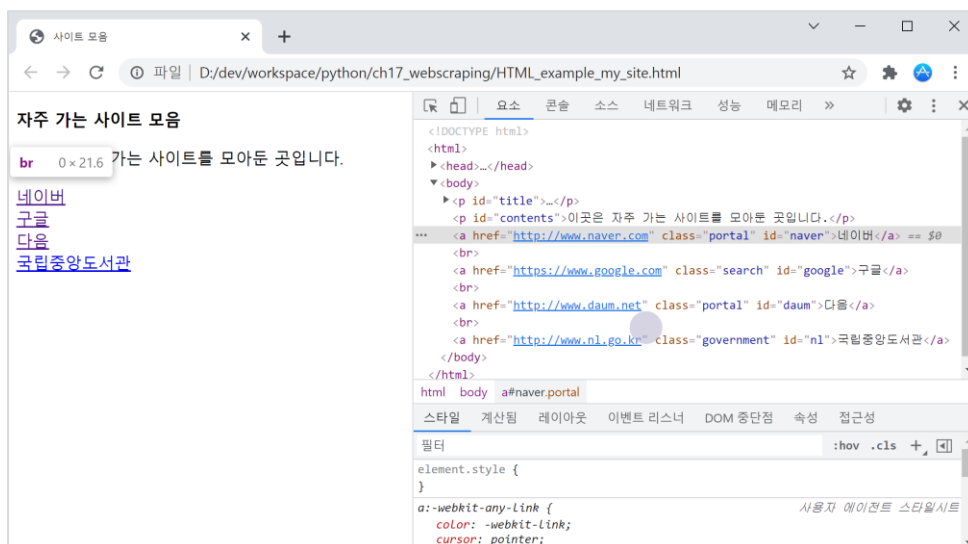
```

149
150 print(soup2.select('p#book_title'))
151 '''
152 [<p id="book_title">토지</p>, <p id="book_title">태백산맥</p>, <p id="book_title">감옥으로부터의 사색
153 </p>]
154 '''
155
156 print(soup2.select('p#author'))
157 '''
158 [<p id="author">박경리</p>, <p id="author">조정래</p>, <p id="author">신영복</p>]
159 '''
160
161 f = open('./ch17_webscraping/HTML_example_my_site.html', encoding='utf-8')
162 html3 = f.read()
163 f.close()
164 soup3 = BeautifulSoup(html3, "lxml")
165 print(soup3.select('a'))
166 '''
167 [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>, <a class="search"
168 href="https://www.google.com" id="google">구글</a>, <a class="portal" href="http://www.daum.net"
169 id="daum">다음</a>, <a class="government" href="http://www.nl.go.kr" id="nl">국립중앙도서관</a>]
170 '''
171
172 print(soup3.select('a.portal'))
173 '''
174 [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>, <a class="portal"
175 href="http://www.daum.net" id="daum">다음</a>]
176 '''

```

(2) 웹 브라우저의 요소 검사

- 웹 브라우저에서 제공하는 요소 검사 기능을 이용하면 관심 위치에서 HTML 소스코드가 어떠한 구조로 구성돼 있는지 좀 더 쉽게 분석할 수 있다.



- BeautifulSoup.select('태그 및 속성')의 인자로 a만 입력해 태그 a를 포함하는 모든 요소를 추출한다.
 - 21: a 태그를 포함하는 요소 중 id 속성이 'naver'인 요소를 선택한다.

[ch17_webscraping/ex07_select.py]

```

01 from bs4 import BeautifulSoup
02
03 f = open('./ch17_web scraping/HTML_example_my_site.html', encoding='utf-8')
04 html3 = f.read()
05 f.close()
06 soup3 = BeautifulSoup(html3, "lxml")
07
08 print(soup3.select('a'))
09 '''
10 [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>, <a class="search"
11 href="https://www.google.com" id="google">구글</a>, <a class="portal" href="http://www.daum.net"
12 id="daum">다음</a>, <a class="government" href="http://www.nl.go.kr" id="nl">국립중앙도서관</a>]
13 '''
14
15 print(soup3.select('a.portal'))
16 '''
17 [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>, <a class="portal"
18 href="http://www.daum.net" id="daum">다음</a>]
19 '''
20
21 print(soup3.select('a#naver'))
22 '''
23 [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>]
24 '''

```

(3) 줄 바꿈으로 가독성 높이기

- BeautifulSoup로 웹 사이트의 HTML 소스를 가져온 후에 `get_text()`를 이용해 요소의 텍스트를 가져와서 출력하면 웹 브라우저에서 보는 것과 달리 줄 바꿈이 없어서 가독성이 떨어질 때가 있다. 이런 문제점을 해결해 가독성을 높이는 방법을 살펴보자.
 - 36~45: 첫 번째 `br` 태그만 문자열 교환이 이뤄졌다.
 - 62~66: 추출된 요소에 대해 모든 `br` 태그가 개행문자(`\n`)로 바뀌는 함수이다.

[ch17_web scraping/ex08_replace.py]

```

01 from bs4 import BeautifulSoup
02
03 f = open('./ch17_web scraping/br_example_constitution.html', encoding='utf-8')
04
05 html_source = f.read()
06 f.close()
07
08 soup = BeautifulSoup(html_source, "lxml")
09
10 title = soup.find('p', {"id": "title"})
11 contents = soup.find_all('p', {"id": "content"})
12
13 print(title.get_text())
14 for content in contents:
15     print(content.get_text())
16 ...
17 제1조 ①대한민국은 민주공화국이다.②대한민국의 주권은 국민에게 있고, 모든 권력은 국
18 민으로부터 나온다.
19 제2조 ①대한민국의 국민이 되는 요건은 법률로 정한다.②국가는 법률이 정하는 바에 의
20 하여 재외국민을 보호할 의무를 진다.
21 ...
22
23
24 html1 = '<p id="content">제1조 <br/>①대한민국은 민주공화국이다.<br/>②대한민국의 주권은 국민에게 있고,

```

```

25 모든 권력은 국민으로부터 나온다.</p>'
26 soup1 = BeautifulSoup(html1, "lxml")
27
28 print("==> 태그 p로 찾은 요소")
29 content1 = soup1.find('p', {"id": "content"})
30 print(content1)
31 '''
32 <p id="content">제1조 <br/>①대한민국은 민주공화국이다.<br/>②대한민국의 주권은 국
33 민에게 있고, 모든 권력은 국민으로부터 나온다.</p>
34 '''
35
36 br_content = content1.find("br")
37 print("==> 결과에서 태그 br로 찾은 요소:", br_content)
38 br_content.replace_with("\n")
39 print("==> 태그 br을 개행문자로 바꾼 결과")
40 print(content1)
41 '''
42 <p id="content">제1조
43 ①대한민국은 민주공화국이다.<br/>②대한민국의 주권은 국민에게 있고, 모든 권력은 국
44 민으로부터 나온다.</p>
45 '''
46
47
48 soup2 = BeautifulSoup(html1, "lxml")
49 content2 = soup2.find('p', {"id": "content"})
50
51 br_contents = content2.find_all("br")
52 for br_content in br_contents:
53     br_content.replace_with("\n")
54 print(content2)
55 '''
56 <p id="content">제1조
57 ①대한민국은 민주공화국이다.
58 ②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.</p>
59 '''
60
61
62 def replace_newline(soup_html):
63     br_to_newlines = soup_html.find_all("br")
64     for br_to_newline in br_to_newlines:
65         br_to_newline.replace_with("\n")
66     return soup_html
67
68 soup2 = BeautifulSoup(html1, "lxml")
69 content2 = soup2.find('p', {"id": "content"})
70 content3 = replace_newline(content2)
71 print(content3.get_text())
72 '''
73 제1조
74 ①대한민국은 민주공화국이다.
75 ②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.
76 '''

```

17.3 웹 사이트에서 데이터 가져오기

17.3.1 웹 스크레이핑 시 주의사항

- 웹 페이지의 소스코드에서 데이터를 얻기 위한 규칙을 발견할 수 있어야 한다.
- 파이썬 코드를 이용해 웹 스크레이핑을 할 경우 해당 웹 사이트에 너무 빈번하게 접근하지

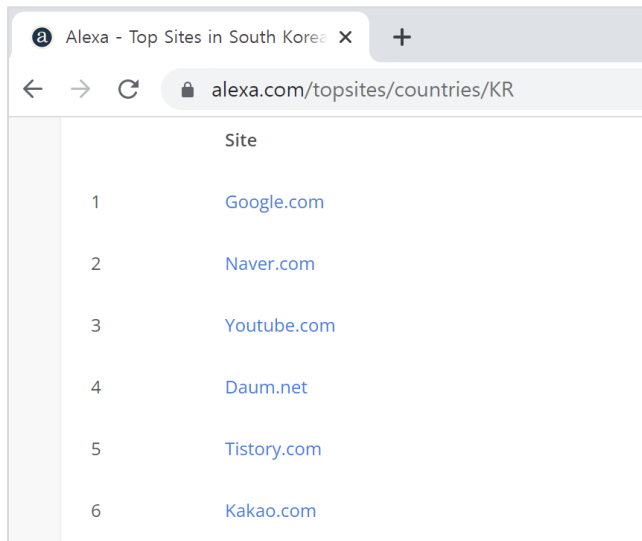
말아야 한다.

- 웹 사이트는 언제든지 예고 없이 변경될 수 있다. 그러므로 웹 스크레이핑을 위한 코드는 한번 만들고 끝나는 것이 아니라 지속해서 관리해야 한다.
- 웹 사이트에서 얻은 데이터를 활용하기 전에 저작권 침해 여부를 미리 확인해야 한다.

17.3.2 순위 데이터를 가져오기

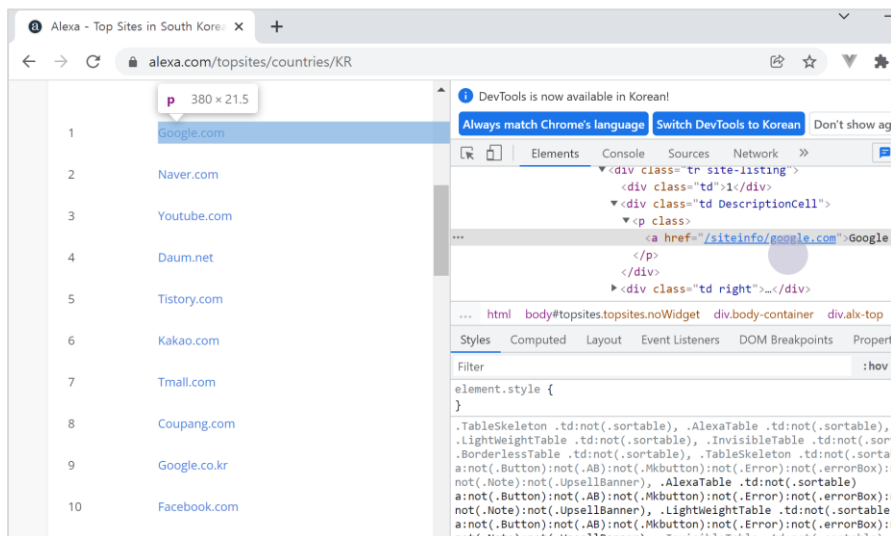
(1) 웹 사이트 순위

- 아마존의 계열사인 알렉사(Alexa Internet Inc.)에서는 전 세계적으로 혹은 나라별로 웹 사이트의 순위 정보를 제공한다.
 - <https://www.alexa.com/topsites/countries/KR>



	Site
1	Google.com
2	Naver.com
3	Youtube.com
4	Daum.net
5	Tistory.com
6	Kakao.com

- 웹 사이트 화면의 관심 위치에서 요소 검사를 수행하여 문서의 구조를 파악한다.



DevTools is now available in Korean!

Always match Chrome's language | Switch DevTools to Korean | Don't show ag

Elements | Console | Sources | Network

```
<div class="tr site-listing">
  <div class="td">1</div>
  <div class="td DescriptionCell">
    <p class="site-name">
      <a href="/siteinfo/google.com">Google.
    </p>
  </div>
  <div class="td right">...</div>
</div>
```

html body#topsites.topsites.noWidget div.body-container div.alx-top

Filter | Styles | Computed | Layout | Event Listeners | DOM Breakpoints | Property

element.style {

```
.TableSkeleton .td:not(.sortable), .AlexaTable .td:not(.sortable),
.LightWeightTable .td:not(.sortable), .InvisibleTable .td:not(.sort
.BorderlessTable .td:not(.sortable), .TableSkeleton .td:not(.sorta
a:not(.Button):not(.AB):not(.Mkbutton):not(.Error):not(.errorBox):f
not(.Note):not(.UpsellBanner), .AlexaTable .td:not(.sortable)
a:not(.Button):not(.AB):not(.Mkbutton):not(.Error):not(.errorBox):f
not(.Note):not(.UpsellBanner), .LightWeightTable .td:not(.sortable)
a:not(.Button):not(.AB):not(.Mkbutton):not(.Error):not(.errorBox):f
not(.Note):not(.UpsellBanner), .InvisibleTable .td:not(.sortable)
```

- select()의 인자 'p a'를 이용해 웹 사이트의 트래픽 순위를 추출하는 코드를 작성한다.

[ch17_webscraping/ex09_alexa.py]

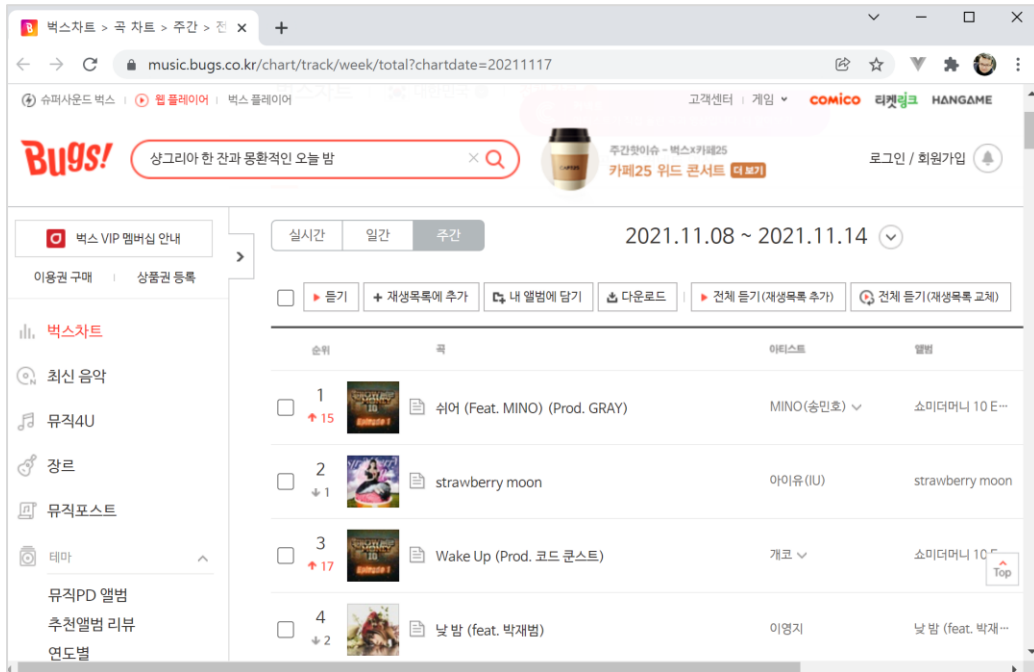
```
01 import requests
02 from bs4 import BeautifulSoup
03
04 url = "https://www.alexa.com/topsites/countries/KR"
05
06 html_website_ranking = requests.get(url).text
07 soup_website_ranking = BeautifulSoup(html_website_ranking, "lxml")
08
09 # p 태그의 요소 안에서 a 태그의 요소를 찾음
10 website_ranking = soup_website_ranking.select('p a')
11
12 print(website_ranking[0:6])
13 '''
14 [<a href="https://support.alexa.com/hc/en-us/articles/200444340" target="_blank">this explanation</a>,
15 <a href="/siteinfo/google.com">Google.com</a>, <a href="/siteinfo/naver.com">Naver.com</a>, <a
16 href="/siteinfo/youtube.com">Youtube.com</a>, <a href="/siteinfo/daum.net">Daum.net</a>, <a
17 href="/siteinfo/tistory.com">Tistory.com</a>]
18 '''
19
20 print(website_ranking[1].get_text())
21 # Google.com
22
23 website_ranking_address = [website_ranking_element.get_text(
24 ) for website_ranking_element in website_ranking[1:]]
25 print(website_ranking_address[0:6])
26 '''
27 ['Google.com', 'Naver.com', 'Youtube.com', 'Daum.net', 'Tistory.com', 'Kakao.com']
28 '''
29
30
31 url = "https://www.alexa.com/topsites/countries/KR"
32
33 html_website_ranking = requests.get(url).text
34 soup_website_ranking = BeautifulSoup(html_website_ranking, "lxml")
35
36 # p 태그의 요소 안에서 a 태그의 요소를 찾음
37 website_ranking = soup_website_ranking.select('p a')
38 website_ranking_address = [website_ranking_element.get_text(
39 ) for website_ranking_element in website_ranking]
40
41 print("[Top Sites in South Korea]")
42 for k in range(6):
43     print("{0}: {1}".format(k+1, website_ranking_address[k]))
44 '''
45 [Top Sites in South Korea]
46 1: this explanation
47 2: Google.com
48 3: Naver.com
49 4: Youtube.com
50 5: Daum.net
51 6: Tistory.com
52 '''
```

(2) 주간 음악 순위

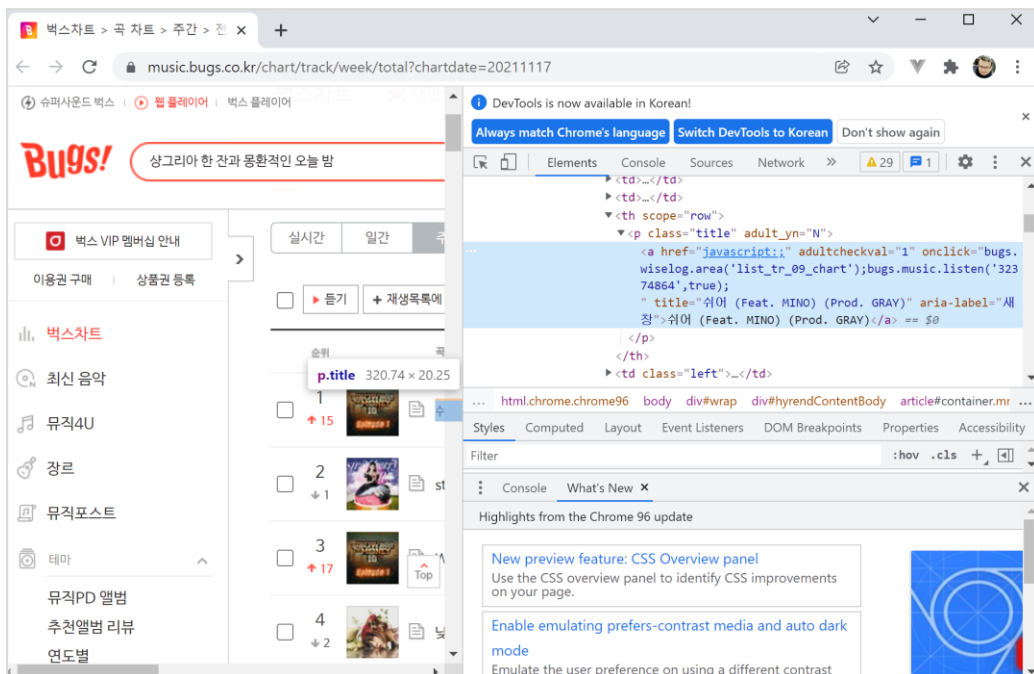
- 파이썬을 이용해 음악 서비스를 제공하는 벅스(Bugs)에서 음악 순위 정보(곡명과 아티스트)를 가져오는 코드를 작성한다.

■ 벅스차트의 주간 음악 순위

- <https://music.bugs.co.kr/chart/track/week/total?chartdate=20211117>



■ HTML 소스코드를 분석하기 위해 1위의 노래 제목에서 마우스 오른쪽 버튼을 클릭한 다음 [요소 검사]를 클릭한다.



■ 데이터 추출이 잘 되는지 다음 코드를 확인해 보자.

[ch17_webscraping/ex10_bugs.py]

```

01 import glob
02 import requests
03 from bs4 import BeautifulSoup
04
05 # 주간 뮤직 차트 (날짜 지정)
06 url = "https://music.bugs.co.kr/chart/track/week/total?chartdate=20200921"
07 # url = "https://music.bugs.co.kr/chart/track/realtime/total" # 실시간 뮤직 차트
08 # url = "https://music.bugs.co.kr/chart/track/day/total" # 일간 뮤직 차트
09 # url = "https://music.bugs.co.kr/chart/track/week/total" # 주간 뮤직 차트
10
11 html_music = requests.get(url).text
12 soup_music = BeautifulSoup(html_music, "lxml")
13
14 # p 태그의 요소 중에서 class 속성값이 "title" 인 것을 찾고
15 # 그 안에서 a 태그의 요소를 추출
16 titles = soup_music.select('p.title a')
17 print(titles[0:7])
18 '''
19 [<a          adultcheckval="1"          aria-label="새창"          href="javascript:;"
20 onclick="bugs.wiselog.area('list_tr_09_chart');bugs.music.listen('32374864',true);
21 " title="쉬어 (Feat. MINO) (Prod. GRAY)">쉬어 (Feat. MINO) (Prod. GRAY)</a>, <a adultcheckval="1" aria-
22 label="새창"          href="javascript:;"
23 onclick="bugs.wiselog.area('list_tr_09p_chart');bugs.music.listen('6132120',true);
24 " title="strawberry moon">strawberry moon</a>, <a adultcheckval="1" aria-label="새창"
25 href="javascript:;" onclick="bugs.wiselog.area('list_tr_09_chart');bugs.music.listen('32374863',true);
26 " title="Wake Up (Prod. 코드 쿤스트)">Wake Up (Prod. 코드 쿤스트)</a>, <a adultcheckval="1" aria-label="
27 새창"          href="javascript:;"
28 onclick="bugs.wiselog.area('list_tr_09_chart');bugs.music.listen('6133448',true);
29 " title="낮 밤 (feat. 박재범)">낮 밤 (feat. 박재범)</a>, <a adultcheckval="1" aria-label="새창"
30 href="javascript:;" onclick="bugs.wiselog.area('list_tr_09_chart');bugs.music.listen('6135536',true);
31 " title="어제 너는 나를 버렸어">어제 너는 나를 버렸어</a>, <a adultcheckval="1" aria-label="새창"
32 href="javascript:;" onclick="bugs.wiselog.area('list_tr_09_chart');bugs.music.listen('32349594',true);
33 " title="Savage">Savage</a>, <a adultcheckval="1" aria-label="새창" href="javascript:;"
34 onclick="bugs.wiselog.area('list_tr_09_chart');bugs.music.listen('32374861',true);
35 " title="TROUBLE (Prod. Slom)">TROUBLE (Prod. Slom)</a>]
36 '''
37
38 music_titles = [title.get_text() for title in titles]
39 print(music_titles[0:7])
40 '''
41 ['쉬어 (Feat. MINO) (Prod. GRAY)', 'strawberry moon', 'Wake Up (Prod. 코드 쿤스트)', '
42 낮 밤 (feat. 박재범)', '어제 너는 나를 버렸어', 'Savage', 'TROUBLE (Prod. Slom)']
43 '''
44
45 # p 태그의 요소 중에서 class 속성값이 "artist" 인 것을 찾고
46 # 그 안에서 a 태그의 요소를 추출
47 artists = soup_music.select('p.artist a')
48 print(artists[0:7])
49 '''
50 [<a class="artistTitle" href="https://music.bugs.co.kr/artist/80185895?w1_ref=list_tr_10_chart"
51 title="MINO(송민호)">MINO(송민호)</a>, <a class="more" href="javascript:void(0);"
52 name="atag_martist_list" onclick="bugs.layermenu.openMultiArtistSearchResultPopLayer(this, 'MINO(송민
53 호)||MINO(송민호)||80185895||OK\\nAnandelight||아년딜라이트
54 (Anandelight)||20070271||OK\\nunofficialboy||unofficialboy||20036043||OK\\nBE\0 (비오)||BE\0
55 (비오)||80337001||OK\\n지구인||지구인||80126124||OK\\nMudd the student||Mudd the student||20096572||OK',
56 ''); return false;" title="아티스트 전체보기" wise_log_str="?w1_ref=list_tr_10_chart">
57 MINO(송민호)
58 </a>, <a href="https://music.bugs.co.kr/artist/80049126?w1_ref=list_tr_10_chart" onclick="
59 " title="아이유(IU)">아이유(IU)</a>, <a class="artistTitle"
60 href="https://music.bugs.co.kr/artist/2916?w1_ref=list_tr_10_chart" title="개코">개코</a>, <a
61 class="more" href="javascript:void(0);" name="atag_martist_list"
62 onclick="bugs.layermenu.openMultiArtistSearchResultPopLayer(this, '개코||개코
63 ||2916||OK\\n0urealcoat||0urealcoat (아우릴코트)||20076067||OK\\nSINCE||SINCE||20056615||OK\\n안병웅||
64 안병웅||20086251||OK\\nTabber||Tabber||80212388||OK\\n조광일||조광일||20087231||OK', ''); return
65 false;" title="아티스트 전

```

```

66 체보기" wise_log_str="?wl_ref=list_tr_10_chart">
67 개코
68 </a>, <a href="https://music.bugs.co.kr/artist/20079471?wl_ref=list_tr_10_chart" onclick="
69 " title="이영지">이영지</a>, <a href="https://music.bugs.co.kr/artist/80067149?wl_ref=list_tr_10_chart"
70 onclick="
71 " title="10CM">10CM</a>]
72 '''
73
74 artists = soup_music.select('p.artist a:not(.more)')
75 print(artists[0:7])
76 '''
77 [<a class="artistTitle" href="https://music.bugs.co.kr/artist/80185895?wl_ref=list_tr_10_chart"
78 title="MINO(송민호)">MINO(송민호)</a>, <a
79 href="https://music.bugs.co.kr/artist/80049126?wl_ref=list_tr_10_chart" onclick="
80 " title="아이유(IU)">아이유(IU)</a>, <a class="artistTitle"
81 href="https://music.bugs.co.kr/artist/2916?wl_ref=list_tr_10_chart" title="개코">개코</a>, <a
82 href="https://music.bugs.co.kr/artist/20079471?wl_ref=list_tr_10_chart" onclick="
83 " title="이영지">이영지</a>, <a href="https://music.bugs.co.kr/artist/80067149?wl_ref=list_tr_10_chart"
84 onclick="
85 " title="10CM">10CM</a>, <a href="https://music.bugs.co.kr/artist/80347326?wl_ref=list_tr_10_chart"
86 onclick="
87 " title="aespa">aespa</a>, <a class="artistTitle"
88 href="https://music.bugs.co.kr/artist/80085859?wl_ref=list_tr_10_chart" title="Zion.T">Zion.T</a>]
89 '''
90
91 music_artists = [artist.get_text() for artist in artists]
92 print(music_artists[0:7])
93 # ['MINO(송민호)', '아이유(IU)', '개코', '이영지', '10CM', 'aespa', 'Zion.T']
94
95
96 url = "https://music.bugs.co.kr/chart/track/week/total?chartdate=20200921"
97 html_music = requests.get(url).text
98 soup_music = BeautifulSoup(html_music, "lxml")
99
100 titles = soup_music.select('p.title a')
101 artists = soup_music.select('p.artist a:not(.more)')
102
103 music_titles = [title.get_text() for title in titles]
104 music_artists = [artist.get_text().strip() for artist in artists]
105
106 for k in range(7):
107     print("{0}: {1} / {2}".format(k+1, music_titles[k], music_artists[k]))
108 '''
109 1: Dynamite / 방탄소년단
110 2: Bad Boy / 청하
111 3: 취기를 빌려 (취향저격 그녀 X 산들) / 산들
112 4: Tight / 10CM
113 5: 그리워하면 그댈 만날까봐 / 김나영
114 6: 숲의 아이 (Bon voyage) / 유아 (오마이걸)
115 7: 밤새 (취향저격 그녀 X 카더가든) / 카더가든
116 '''
117
118 music_titles_artists = {}
119 order = 0
120
121 for (music_title, music_artist) in zip(music_titles, music_artists):
122     order = order + 1
123     music_titles_artists[order] = [music_title, music_artist]
124
125 print(music_titles_artists[1])
126 # ['Dynamite', '방탄소년단']
127
128 print(music_titles_artists[2])
129 # ['Bad Boy', '청하']
130

```

```

131
132 # 날짜를 입력하면 벅스 차트에서 주간 음악 순위(1~100위)의 곡명과 아티스트를 반환
133 def bugs_music_week_top100(year, month, day):
134
135     # 월과 일의 경우는 항상 두 자리로 맞춤
136     month = "{0:02d}".format(month)
137     day = "{0:02d}".format(day)
138
139     base_url = 'https://music.bugs.co.kr/chart/track/week/total?'
140     url = base_url + 'chartdate={0}{1}{2}'.format(year, month, day)
141
142     html_music = requests.get(url).text
143     soup_music = BeautifulSoup(html_music, "lxml")
144
145     titles = soup_music.select('p.title a')
146     artists = soup_music.select('p.artist a:not(.more)')
147
148     music_titles = [title.get_text() for title in titles]
149     music_artists = [artist.get_text().strip() for artist in artists]
150
151     return music_titles, music_artists
152
153
154 # 날짜를 지정해 bugs_music_week_top100() 함수 호출
155 bugs_music_titles, bugs_music_artists = bugs_music_week_top100(2020, 9, 21)
156
157 # 곡명과 아티스트를 저장할 파일 이름을 폴더와 함께 지정
158 file_name = './ch17_webscraping/bugs_week_top100.txt'
159
160 f = open(file_name, 'w', encoding="utf-8") # 파일 열기
161
162 # 추출된 노래 제목과 아티스트를 파일에 저장
163 for k in range(len(bugs_music_titles)):
164     f.write("{0:2d}: {1}/{2}\n".format(k+1,
165                                     bugs_music_titles[k], bugs_music_artists[k]))
166
167 f.close() # 파일 닫기
168
169 glob.glob(file_name) # 생성된 파일 확인

```

17.3.3 웹 페이지에서 이미지 가져오기

(1) 하나의 이미지 내려받기

[ch17_webscraping/ex11_imageFile.py]

```

01 import requests
02
03 url = 'https://www.python.org/static/img/python-logo.png'
04
05 html_image = requests.get(url)
06 print(html_image)
07
08
09 import os
10
11 image_file_name = os.path.basename(url)
12 print(image_file_name)
13 # python-logo.png
14
15

```

```

16 folder = './ch17_webscraping/download'
17
18 if not os.path.exists(folder):
19     os.makedirs(folder)
20
21
22 image_path = os.path.join(folder, image_file_name)
23 print(image_path)
24 # ./ch17_webscraping/download/python-logo.png
25
26 imageFile = open(image_path, 'wb')
27
28 # 이미지 데이터를 1000000 바이트씩 나눠서 내려받고 파일에 순차적으로 저장
29 chunk_size = 1000000
30 for chunk in html_image.iter_content(chunk_size):
31     imageFile.write(chunk)
32 imageFile.close()
33
34 print(os.listdir(folder))
35 # ['python-logo.png']
36
37
38
39 import requests
40 import os
41
42 url = 'https://www.python.org/static/img/python-logo.png'
43 html_image = requests.get(url)
44 image_file_name = os.path.basename(url)
45
46 folder = './ch17_webscraping/download'
47
48 if not os.path.exists(folder):
49     os.makedirs(folder)
50
51 image_path = os.path.join(folder, image_file_name)
52
53 imageFile = open(image_path, 'wb')
54 # 이미지 데이터를 1000000 바이트씩 나눠서 저장
55 chunk_size = 1000000
56 for chunk in html_image.iter_content(chunk_size):
57     imageFile.write(chunk)
58 imageFile.close()

```

(2) 여러 이미지 내려받기

[ch17_webscraping/ex12_imageFiles.py]

```

01 import requests
02 import os
03 from bs4 import BeautifulSoup
04
05 URL = 'https://www.reshot.com/search/animal'
06
07 html_reshot_image = requests.get(URL).text
08 soup_reshot_image = BeautifulSoup(html_reshot_image, "lxml")
09 reshot_image_elements = soup_reshot_image.select('a img')
10 reshot_image_elements[0:4]
11
12 reshot_image_url = reshot_image_elements[1].get('src')
13 print(reshot_image_url)
14 '''

```

```

15 https://res.cloudinary.com/twenty20/private_images/t_resnet-400/v1521838685/photosp/bae96789-a5ab-4471-
16 b54f-9686ace09e33/bae96789-a5ab-4471-b54f-9686ace09e33.jpg
17 '''
18
19
20 html_image = requests.get(reshot_image_url)
21
22 folder = './ch17_webscraping/download' # 이미지를 내려받을 폴더를 지정
23
24 # os.path.basename(URL)은 웹사이트나 폴더가 포함된 파일명에서 파일명만 분리
25 imageFile = open(os.path.join(
26     folder, os.path.basename(reshot_image_url)), 'wb')
27
28 # 이미지 데이터를 1000000 바이트씩 나눠서 저장
29 chunk_size = 1000000
30 for chunk in html_image.iter_content(chunk_size):
31     imageFile.write(chunk)
32 imageFile.close()
33
34
35 import requests
36 from bs4 import BeautifulSoup
37 import os
38
39 # URL(주소)에서 이미지 주소를 추출
40 def get_image_url(url):
41     html_image_url = requests.get(url).text
42     soup_image_url = BeautifulSoup(html_image_url, "lxml")
43     image_elements = soup_image_url.select('a img')
44     if(image_elements != None):
45         image_urls = []
46         for image_element in image_elements[1:]:
47             image_urls.append(image_element.get('src'))
48         return image_urls
49     else:
50         return None
51
52 # 폴더를 지정해 이미지 주소에서 이미지 내려받기
53 def download_image(img_folder, img_url):
54     if(img_url != None):
55         html_image = requests.get(img_url)
56         # os.path.basename(URL)은 웹사이트나 폴더가 포함된 파일명에서 파일명만 분리
57         imageFile = open(os.path.join(img_folder, os.path.basename(img_url)), 'wb')
58
59         chunk_size = 1000000 # 이미지 데이터를 1000000 바이트씩 나눠서 저장
60         for chunk in html_image.iter_content(chunk_size):
61             imageFile.write(chunk)
62             imageFile.close()
63         print("이미지 파일명: '{0}'. 내려받기 완료!".format(os.path.basename(img_url)))
64     else:
65         print("내려받을 이미지가 없습니다.")
66
67 # 웹 사이트의 주소를 지정
68 reshot_url = 'https://www.reshot.com/search/animal'
69
70 figure_folder = "./ch17_webscraping/download" # 이미지를 내려받을 폴더를 지정
71
72 reshot_image_urls = get_image_url(reshot_url) # 이미지 파일의 주소 가져오기
73
74 num_of_download_image = 7 # 내려받을 이미지 개수를 지정
75 # num_of_download_image = len(reshot_image_urls) # 전체 이미지 개수
76
77 for k in range(num_of_download_image):
78     download_image(figure_folder, reshot_image_urls[k])
79 print("=====")

```



```
80 print("선택한 모든 이미지 내려받기 완료!")
81
82 num_of_download_image = len(reshot_image_urls)
83 print(num_of_download_image)
84 # 50
```