

웹 스크레이핑

웹 브라우저로 웹 사이트 접속하기

하나의 웹 사이트에 접속하기

- 사이트를 하나 지정한 후에 웹 브라우저를 열어서 접속하는 방법

```
In [1]: import webbrowser
```

```
url = 'www.naver.com'  
webbrowser.open(url, new=0)
```

```
Out[1]: True
```

- 네이버에서 특정 검색어를 입력해 결과 얻기

```
In [2]: import webbrowser
```

```
naver_search_url = "http://search.naver.com/search.naver?query="  
search_word = '파이썬'  
url = naver_search_url + search_word  
  
webbrowser.open_new(url)
```

```
Out[2]: True
```

- 구글(Google)에서도 검색을 위한 웹 사이트 주소(www.google.com)와 검색어를 연결 해 입력하면 검색 결과를 얻을 수 있음

```
In [3]: import webbrowser

google_url = "www.google.com/search?q="
search_word = 'python'
url = google_url + search_word

webbrowser.open_new(url)
```

Out[3]: True

여러 개의 웹 사이트에 접속하기

- url 주소 리스트와 for 문을 이용

```
In [4]: import webbrowser

urls = ['www.naver.com', 'www.daum.net', 'www.google.com']

for url in urls:
    webbrowser.open_new(url)
```

- 여러 단어 리스트와 for문 이용

```
In [5]: import webbrowser

google_url = "www.google.com/search?q="
search_words = ['python web scraping', 'python webbrowser']

for search_word in search_words:
    webbrowser.open_new(google_url + search_word)
```

웹 스크레이핑을 위한 기본 지식

데이터의 요청과 응답 과정

HTML의 기본 구조

- HTML 생성

```
In [6]: import os
```

```
# 폴더 생성  
folder = "C:/Myexam"  
os.makedirs(folder, exist_ok=True)
```

```
In [7]:
```

```
%%writefile C:/Myexam/HTML_example.html  
<!doctype html>  
<html>  
<head>  
<meta charset="utf-8">  
<title>이것은 HTML 예제</title>  
</head>  
<body>  
<h1>출간된 책 정보</h1>  
<p id="book_title">이해가 쑥쑥 되는 파일</p>  
<p id="author">홍길동</p>  
<p id="publisher">위키북스 출판사</p>  
<p id="year">2018</p>  
</body>  
</html>
```

Overwriting C:/Myexam/HTML_example.html

- HTML 생성

```
In [8]:
```

```
%%writefile C:/Myexam/HTML_example2.html  
<!doctype html>  
<html>  
<head>  
<meta charset="utf-8">  
<title>이것은 HTML 예제</title>
```

```
</head>
<body>
  <h1>출간된 책 정보</h1>
  <p>이해가 쏙쏙 되는 파이썬</p>
  <p>홍길동</p>
  <p>위키북스 출판사</p>
  <p>2018</p>
</body>
</html>
```

Overwriting C:/Myexam/HTML_example2.html

웹 페이지의 HTML 소스 갖고 오기

- 구글 웹 페이지(<https://www.google.co.kr>)의 소스코드

In [9]: `import requests`

```
r = requests.get("https://www.google.co.kr")
r
```

Out[9]: <Response [200]>

- 응답 객체를 잘 가져왔는지 확인만 하면 되므로 HTML 파일 전체 중 일부분 출력

In [10]: `r.text[0:100]`

Out[10]: '`<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ko"><head><meta content'`

- 한번에 수행 가능

In [11]: `import requests`

```
html = requests.get("https://www.google.co.kr").text
html[0:100]
```

```
Out[11]: '<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ko"><head><meta content'
```

HTML 소스코드를 분석하고 처리하기

데이터 찾고 추출하기

- HTML 코드를 분석해 원하는 데이터를 추출하는 방법
- HTML 코드를 분석하기 위해서는 HTML 코드 구문을 이해하고 요소별로 HTML 코드를 분류
- BeautifulSoup 라이브러리를 이용해 HTML 소스를 파싱하고
- 태그나 속성을 통해 원하는 데이터를 추출

```
In [12]: from bs4 import BeautifulSoup

# 테스트용 html 코드
html = """<html><body><div><span>\n    <a href=http://www.naver.com>naver</a>\n    <a href=https://www.google.com>google</a>\n    <a href=http://www.daum.net/>daum</a>\n</span></div></body></html>"""

# BeautifulSoup을 이용해 HTML 소스를 파싱
soup = BeautifulSoup(html, 'lxml')
soup
```

```
Out[12]: <html><body><div><span> <a href="http://www.naver.com">naver</a> <a href="https://www.google.com">google</a> <a href="http://www.daum.net/">daum</a> </span></div></body></html>
```

- 파싱 결과를 좀 더 보기 편하게 HTML 구조의 형태로 확인

```
In [13]: print(soup.prettify())
```

```
<html>
<body>
<div>
<span>
<a href="http://www.naver.com">
naver
</a>
<a href="https://www.google.com">
google
</a>
<a href="http://www.daum.net/">
daum
</a>
</span>
</div>
</body>
</html>
```

- 파싱한 결과에서 BeautifulSoup.find('태그')를 수행하면
- HTML 소스코드에서 해당 '태그'가 있는 첫 번째 요소를 찾아서 반환

In [14]: `soup.find('a')`

Out[14]: `naver`

- get_text()는 HTML 소스코드의 요소에서 태그와 속성을 제거하고 텍스트 문자열만 반환
- get_text()는 원하는 HTML 요소를 가져온 후에 마지막 단계에서 요소의 텍스트 부분만 추출할 때 이용

In [15]: `soup.find('a').get_text()`

Out[15]: `'naver'`

- HTML 코드안의 모든 a 태그를 찾아서 a 태그로 시작하는 모든 요소를 다 반환하려면
- BeautifulSoup.find_all('태그')를 이용

```
In [16]: soup.find_all('a')
```

```
Out[16]: [<a href="http://www.naver.com">naver</a>,
<a href="https://www.google.com">google</a>,
<a href="http://www.daum.net/">daum</a>]
```

- 태그 이름의 모든 요소를 반환하는 `find_all()`의 결과는 리스트 형태로 반환
- `get_text()`는 리스트에 적용할 수 없으므로
- `for`문을 이용해 항목별로 `get_text()`를 적용

```
In [17]: site_names = soup.find_all('a')
for site_name in site_names:
    print(site_name.get_text())
```

```
naver
google
daum
```

- HTML 파일을 작성한 후에 `html2` 변수에 할당

```
In [18]: from bs4 import BeautifulSoup
```

```
# 테스트용 HTML 코드
html2 = """
<html>
  <head>
    <title>작품과 작가 모음</title>
  </head>
  <body>
    <h1>책 정보</h1>
    <p id="book_title">토지</p>
    <p id="author">박경리</p>

    <p id="book_title">태백산맥</p>
    <p id="author">조정래</p>

    <p id="book_title">감옥으로부터의 사색</p>
```

```

<p id="author">신영복</p>
</body>
</html>
"""

soup2 = BeautifulSoup(html2, "lxml")

```

- BeautifulSoup의 다양한 기능을 활용해 HTML 소스로부터 필요한 데이터를 추출
- HTML 소스에서 title 태그의 요소는 'BeautifulSoup.title'을 이용해 가져올 수 있음

In [19]: `soup2.title`

Out[19]: <title>작품과 작가 모음</title>

- HTML 소스의 body 태그의 요소는 'BeautifulSoup.body'를 이용해 가져올 수 있음

In [20]: `soup2.body`

```

Out[20]: <body>
<h1>책 정보</h1>
<p id="book_title">토지</p>
<p id="author">박경리</p>
<p id="book_title">태백산맥</p>
<p id="author">조정래</p>
<p id="book_title">감옥으로부터의 사색</p>
<p id="author">신영복</p>
</body>

```

- body 태그 요소 내에 h1 태그의 요소는 'BeautifulSoup.body.h1'로 가져올 수 있음

In [21]: `soup2.body.h1`

Out[21]: <h1>책 정보</h1>

- find_all()을 이용하면

- 변수 html2에 있는 HTML 소스코드에서 p 태그가 들어 간 요소를 모두 가져올 수 있음

```
In [22]: soup2.find_all('p')
```

```
Out[22]: [<p id="book_title">토지</p>,
<p id="author">박경리</p>,
<p id="book_title">태백산맥</p>,
<p id="author">조정래</p>,
<p id="book_title">감옥으로부터의 사색</p>,
<p id="author">신영복</p>]
```

- p 태그 중 책 제목과 작가를 분리해서 가져오려면
- find()나 find_all()을 이용할 때 '태그' 뿐만 아니라
- 태그 내의 '속성'도 함께 지정
- BeautifulSoup.find_all('태그','속성')
- BeautifulSoup.find('태그','속성')
- html2의 HTML 코드의 p 태그 요소 중 id가 book_title 인 속성을 갖는 첫 번째 요소만 반환

```
In [23]: soup2.find('p', {"id": "book_title"})
```

```
Out[23]: <p id="book_title">토지</p>
```

- p 태그 요소 중 id가 author인 속성을 갖는 첫번째 요소만 반환

```
In [24]: soup2.find('p', {"id": "author"})
```

```
Out[24]: <p id="author">박경리</p>
```

- 조건을 만족하는 요소 전체를 가지고 오려면 find_all()을 이용

```
In [25]: soup2.find_all('p', {"id": "book_title"})
```

```
Out[25]: [<p id="book_title">토지</p>,
<p id="book_title">태백산맥</p>,
<p id="book_title">감옥으로부터의 사색</p>]
```

```
In [26]: soup2.find_all('p', {"id": "author"})
```

```
Out[26]: [<p id="author">박경리</p>, <p id="author">조정래</p>, <p id="author">신영복</p>]
```

- 책 제목과 작가를 포함한 요소를 각각 추출한 후에 텍스트만 뽑는 코드

```
In [27]: from bs4 import BeautifulSoup
```

```
soup2 = BeautifulSoup(html2, "lxml")

book_titles = soup2.find_all('p', {"id": "book_title"})
authors = soup2.find_all('p', {"id": "author"})

for book_title, author in zip(book_titles, authors):
    print(book_title.get_text() + '/' + author.get_text())
```

토지/박경리
태백산맥/조정래
감옥으로부터의 사색/신영복

- CSS 선택자(selector)를 이용
- CSS 선택자는 CSS에서 원하는 요소를 선택 하는 것으로서 파이썬뿐만 아니라 다른 프로그래밍 언어에서도 HTML 소스를 처리할 때 많이 이용
- Beautiful Soup도 'BeautifulSoup.select('태그 및 속성')'를 통해 CSS 선택자를 지원
- 'BeautifulSoup.select()'의 인자로 '태그 및 속성'을 단계적으로 입력하면
- 원하는 요소를 찾을 수 있음

- html2 변수에 할당된 HTML 소스에서 body 태그 요소 내에 M 태그 요소를 가지고 오기

```
In [28]: soup2.select('body h1')
```

```
Out[28]: [<h1>책 정보</h1>]
```

- body 태그 요소 중에 p 태그를 포함한 요소를 모두 갖고 오기

```
In [29]: soup2.select('body p')
```

```
Out[29]: [<p id="book_title">토지</p>,
<p id="author">박경리</p>,
<p id="book_title">태백산맥</p>,
<p id="author">조정래</p>,
<p id="book_title">감옥으로부터의 사색</p>,
<p id="author">신영복</p>]
```

- 변수 html2의 HTML 소스에서 p 태그는 body 태그 요소 내에서만 있음

```
In [30]: soup2.select('p')
```

```
Out[30]: [<p id="book_title">토지</p>,
<p id="author">박경리</p>,
<p id="book_title">태백산맥</p>,
<p id="author">조정래</p>,
<p id="book_title">감옥으로부터의 사색</p>,
<p id="author">신영복</p>]
```

- 태그 안의 속성과 속성값을 이용해 요소를 세밀하게 구분해 추출
- 태그 안의 속성이 class인 경우 '태그.class_속성값'으로 입력하고
- 속성이 id인 경우에는 '태그#id_속성값'으로 입력해 추출
- 태그 안에 있는 속성이 id이므로 'p#id_속성값'으로 원하는 요소를 추출

```
In [31]: soup2.select('p#book_title')
```

```
Out[31]: [<p id="book_title">토지</p>,
<p id="book_title">태백산맥</p>,
<p id="book_title">감옥으로부터의 사색</p>]
```

```
In [32]: soup2.select('p#author')
```

```
Out[32]: [<p id="author">박경리</p>, <p id="author">조정래</p>, <p id="author">신영복</p>]
```

- class 속성이 있는 HTML 소스

```
In [33]: %%writefile C:/Myexam/HTML_example_my_site.html
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>사이트 모음</title>
  </head>
  <body>
    <p id="title"><b>자주 가는 사이트 모음</b></p>
    <p id="contents">이곳은 자주 가는 사이트를 모아둔 곳입니다.</p>
    <a href="http://www.naver.com" class="portal" id="naver">네이버</a> <br>
    <a href="https://www.google.com" class="search" id="google">구글</a> <br>
    <a href="http://www.daum.net" class="portal" id="daum">다음</a> <br>
    <a href="http://www.nl.go.kr" class="government" id="nl">국립중앙도서관</a>
  </body>
</html>
```

Overwriting C:/Myexam/HTML_example_my_site.html

- 'BeautifulSoup.select('태그 몇 속성')'에서 태그 안의 속성이 class인 경우
- '태그.class_속성값'으로 원하는 요소를 추출
- HTML 소스 파일은 이미 저장돼 있으므로 텍스트 파일을 읽어와서 변수 html3에 할당

```
In [34]: f = open('C:/Myexam/HTML_examle_my_site.html', encoding='utf-8')

html3 = f.read()
f.close()

soup3 = BeautifulSoup(html3, "lxml")
```

- 읽어온 HTML 소스에서 태그가 a인 요소를 모두 가져오기

```
In [35]: soup3.select('a')
```

```
Out[35]: [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>,
<a class="search" href="https://www.google.com" id="google">구글</a>,
<a class="portal" href="http://www.daum.net" id="daum">다음</a>,
<a class="government" href="http://www.nl.go.kr" id="nl">국립중앙도서관</a>]
```

- HTML 소스에서 태그가 a이며 class 속성값이 "portal"인 요소만 가져오기

```
In [36]: soup3.select('a.portal')
```

```
Out[36]: [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>,
<a class="portal" href="http://www.daum.net" id="daum">다음</a>]
```

웹 브라우저의 요소 검사

- soup3.select('html body a')
- soup3.select('body a')
- soup3.select('html a')
- soup3.select('a')
- 'BeautifulSoup.select('태그 및 속성')'의 인자로 a만 입력해 태그 a를 포함하는 모든 요소를 추출

```
In [37]: soup3.select('a')
```

```
Out[37]: [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>,
<a class="search" href="https://www.google.com" id="google">구글</a>,
<a class="portal" href="http://www.daum.net" id="daum">다음</a>,
<a class="government" href="http://www.nl.go.kr" id="nl">국립중앙도서관</a>]
```

- HTML 소스에서 태그 a를 포함하는 요소 중 class 속성이 "portal"인 요소만 선택

```
In [38]: soup3.select('a.portal')
```

```
Out[38]: [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>,
<a class="portal" href="http://www.daum.net" id="daum">다음</a>]
```

- 태그를 포함하는 요소 중 id 속성이 "naver" 인 요소를 선택

```
In [39]: soup3.select('a#naver')
```

```
Out[39]: [<a class="portal" href="http://www.naver.com" id="naver">네이버</a>]
```

줄 바꿈으로 가독성 높이기

- HTML 소스 코드를 파일('br_example_constitution.html')로 저장

```
In [40]: %%writefile C:/Myexam/br_example_constitution.html
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>줄 바꿈 테스트 예제</title>
  </head>
  <body>
    <p id="title"><b>대한민국헌법</b></p>
    <p id="content">제1조 <br/>①대한민국은 민주공화국이다.<br/>②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.<br/>
    <p id="content">제2조 <br/>①대한민국의 국민이 되는 요건은 법률로 정한다.<br/>②국가는 법률이 정하는 바에 의하여 재외국민을 보호
  </body>
</html>
```

Overwriting C:/Myexam/br_example_constitution.html

- HTML 파일('br_example_constitution.html')을 읽어서 변수 html_source에 할당한 후
- 요소에서 텍스트를 주출하고 출력

```
In [41]: from bs4 import BeautifulSoup

f = open('C:/Myexam/br_example_constitution.html', encoding='utf-8')

html_source = f.read()
f.close()

soup = BeautifulSoup(html_source, "lxml")

title = soup.find('p', {"id": "title"})
contents = soup.find_all('p', {"id": "content"})

print(title.get_text())
for content in contents:
    print(content.get_text())
```

대한민국헌법

제1조 ①대한민국은 민주공화국이다. ②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.

제2조 ①대한민국의 국민이 되는 요건은 법률로 정한다. ②국가는 법률이 정하는 바에 의하여 재외국민을 보호할 의무를 진다.

- 추출된 HTML 코드에서 줄 바꿈 태그를 파이썬의 개행 문자(\n)로 바꿈
- BeautifulSoup의 'replace_with(새로운 문자열)'를 이용해
- 기존의 태그나 문자열을 새로운 태그나 문자열로 바꿀

- find_result = BeautifulSoup.find ('태그')
- find_result.replace_with('새 태그나 문자열')

- HTML 코드에서 br 태그를 파이썬의 개행문자로 바꾸고 싶으면

```
In [42]: html1 = '<p id="content">제1조 <br/>①대한민국은 민주공화국이다.<br/>②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.</p>'  
  
soup1 = BeautifulSoup(html1, "lxml")  
  
print("==> 태그 p로 찾은 요소")  
content1 = soup1.find('p', {"id": "content"})  
print(content1, "\n")  
  
br_content = content1.find("br")  
print("==> 결과에서 태그 br로 찾은 요소:", br_content, "\n")  
  
br_content.replace_with("\n")  
print("==> 태그 br을 개행문자로 바꾼 결과")  
print(content1)
```

==> 태그 p로 찾은 요소
<p id="content">제1조
①대한민국은 민주공화국이다.
②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.</p>

==> 결과에서 태그 br로 찾은 요소:

==> 태그 br을 개행문자로 바꾼 결과
<p id="content">제1조
①대한민국은 민주공화국이다.
②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.</p>

- 추출된 요소 전체에 적용

```
In [43]: soup2 = BeautifulSoup(html1, "lxml")  
content2 = soup2.find('p', {"id": "content"})  
  
br_contents = content2.find_all("br")  
for br_content in br_contents:  
    br_content.replace_with("\n")  
print(content2)
```

<p id="content">제1조
①대한민국은 민주공화국이다.
②대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.</p>

- 함수 사용

```
In [44]: def replace_newline(soup_html):
    br_to_newlines = soup_html.find_all("br")
    for br_to_newline in br_to_newlines:
        br_to_newline.replace_with("\n")
    return soup_html
```

- BeautifulSoup로 파싱된 HTML 소스에서 br 태그를 개행문자(\n)로 변경
- 함수를 이용한 결과에서 요소의 내용만 추출하기 위해 get_text()를 적용

```
In [45]: soup2 = BeautifulSoup(html1, "lxml")
content2 = soup2.find('p', {"id": "content"})
content3 = replace_newline(content2)
print(content3.get_text())
```

제1조

- ① 대한민국은 민주공화국이다.
② 대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.

- HTML 소스코드를 할당한 변수 html_source에 위의 파일 코드를 적용

```
In [46]: from bs4 import BeautifulSoup

soup = BeautifulSoup(html_source, "lxml")

title = soup.find('p', {"id": "title"})
contents = soup.find_all('p', {"id": "content"})

print(title.get_text(), '\n')

for content in contents:
    content1 = replace_newline(content)
    print(content1.get_text(), '\n')
```

대한민국헌법

제1조

- ① 대한민국은 민주공화국이다.
- ② 대한민국의 주권은 국민에게 있고, 모든 권력은 국민으로부터 나온다.

제2조

- ① 대한민국의 국민이 되는 요건은 법률로 정한다.
- ② 국가는 법률이 정하는 바에 의하여 재외국민을 보호할 의무를 진다.

- 줄을 바꾸어 문단을 구분하는 p 태그를 표기하기 위해
- 'content1.get_text()'를 print()로 출력할 때 개행문자(\n)를 추가

웹 사이트에서 데이터 가져오기

웹 스크레이핑 시 주의 사항

- 웹 페이지의 소스코드에서 데이터를 얻기 위한 규칙을 발견
- 파이썬 코드를 이용해 웹 스크레이핑을 할 경우 해당 웹 사이트에 너무 빈번하게 접근 금지
- 사이트는 언제든지 예고 없이 변경될 수 있음
- 인터넷 상에 공개된 데이터라고 하더라도 저작권(copyright)이 있는 경우가 있음

순위 데이터를 가져오기

웹 사이트 순위

- 인터넷 사용자들이 방문하는 웹 사이트의 방문 정보(접속한 사용자 수, 페이지 뷰 정보 등) 및 웹 트래픽을 분석해서
- 웹 사이트의 순위를 제공하는 웹 사이트가 있음
- 선정한 select()의 인자 'p a'를 이용해 웹 사이트의 트래픽 순위를 추출

- 순위 결과가 잘 추출됐는지 알아보기 위해 변수 website_ranking에 저장된 내용 중에서 앞의 일부만 출력
- 첫번째 항목을 제외한 리스트 website_ranking[1:]의 각 요소에서 웹 사이트 주소
 - 리스트의 모든 항목에 대해 get_text()를 적용하기 위해
 - 한 줄 for 문을 적용한 리스트 컴프리 헨션을 이용
- website_ranking_address 중 앞의 일부만 출력
- 코드를 통합
- pandas의 DataFrame을 이용하면 위의 출력 결과를 좀 더 보기 좋게 만들 수 있음

웹 페이지에서 이미지 가져오기

하나의 이미지 내려받기

- requests 라이브러리를 이용해 이미지 파일을 위한 응답 객체 가져오기

```
In [47]: import requests
```

```
url = 'https://www.python.org/static/img/python-logo.png'
html_image = requests.get(url)
html_image
```

```
Out[47]: <Response [200]>
```

- 이미지 주소에서 이미지 파일명만 추출해 이용

- 이미지 파일의 전체 경로에서 파일 이름만 추출한 것

```
In [48]: import os
```

```
image_file_name = os.path.basename(url)
image_file_name
```

```
Out[48]: 'python-logo.png'
```

- 이미지 파일을 내 컴퓨터로 내려받을 폴더를 생성

- os.makedirs(folder)
- os.path.exists(folder)

```
In [49]: folder = 'C:/Myexam/download'
```

```
if not os.path.exists(folder):
    os.makedirs(folder)
```

- 생성된 폴더와 추출한 이미지 파일명을 합치기 위해서 OS모듈의 메서드를 이용
- os.path.join(path1,[path2,...])
- 파일을 저장 하려는 폴더가 folder이고 파일 이름이 file이라면
- 'os.path.join(folder, file)'로 파일의 전체 경로를 생성
- 생성한 이미지 파일을 위한 폴더와 추출한 이미지 파일을 통합하는 코드

```
In [50]: image_path = os.path.join(folder, image_file_name)
image_path
```

Out[50]: 'C:/Myexam/download\\python-logo.png'

- 이미지 파일을 저장하기 전에 우선 `open('file_name','mode')`을 이용해 파일을 오픈
- `file_name`에는 앞에서 지정한 경로 이름을 넣고
- `mode`에는 쓰기 모드와 바이너리 파일 모드를 지정
- 저장하려는 파일이 텍스트 파일이 아니고
- 이미지 파일이므로 바이너리 파일 모드로 지정

In [51]: `imageFile = open(image_path, 'wb')`

- `requests` 라이브러리의 `iter_content(chunk_size)`를 이용해
- 전체 이미지를 `chunk_size [bytes]` 만큼나눠서 내려옴
- 전체 파일의 마지막까지 나눠서 내려받은 데이터를 차례대로 파일 쓰기를 하면
- 최종적으로 완전한 하나의 이미지 파일을 내려받을 수 있음

In [52]: `# 이미지 데이터를 1000000 바이트씩 나눠서 내려받고 파일에 순차적으로 저장`
`chunk_size = 1000000`
`for chunk in html_image.iter_content(chunk_size):`
 `imageFile.write(chunk)`
`imageFile.close()`

- 지정된 폴더의 파일 목록을 보여주는 `'os.listdir(folder)'`를 수행

In [53]: `os.listdir(folder)`

Out[53]: `['external-link-arrow-9fdf3fbf53535dc5eb554a9a5d5601329bb7bb7169ee72746a37fac0527b2967.svg',`
`'icon-no-photos-034c2399566c35bc56688f11795abdadf83f3c88fd167216a17ed55c2d65c73e.svg',`
`'python-logo.png',`
`'reshot-logo--mark-f8dfafbc1cc8fbf4dfa0e2f210265735aefa6e32f883b5a1fe27fd94f84719b3.svg']`

- 이미지 주소를 알 경우 이미지 파일을 컴퓨터로 내려받는 방법

```
In [54]: import requests
import os

url = 'https://www.python.org/static/img/python-logo.png'
html_image = requests.get(url)
image_file_name = os.path.basename(url)

folder = 'C:/Myexam/download'

if not os.path.exists(folder):
    os.makedirs(folder)

image_path = os.path.join(folder, image_file_name)

imageFile = open(image_path, 'wb')
# 이미지 데이터를 1000000 바이트씩 나눠서 저장
chunk_size = 1000000
for chunk in html_image.iter_content(chunk_size):
    imageFile.write(chunk)
imageFile.close()
```

여러 이미지 내려받기

- select('a img')를 수행하면 해당 이미지의 요소가 추출

```
In [55]: import requests
from bs4 import BeautifulSoup

URL = 'https://reshot.com/search/animal'

html_reshot_image = requests.get(URL).text
soup_reshot_image = BeautifulSoup(html_reshot_image, "lxml")
reshot_image_elements = soup_reshot_image.select('a img')
reshot_image_elements[0:4]
```

```
Out[55]: [,
,
,
]
```

- 출력 결과를 보면 img 태그가 포함된 이미지가 있는 요소가 추출
- 리스트 reshot_image_elements의 제일 첫 번째 요소는 reshot의 로고 이미지
- 동물 이미지만 가져오기 위해서는 reshot_image_elements[1:]을 이용
- BeautifulSoup에서 get('속성')은 '속성'에 들어간 '속성값'을 반환
- 추출된 요소에서 src 의 속성값인 이미지 주소를 구하려면 get('src')을 수행

```
In [56]: reshot_image_url = reshot_image_elements[1].get('src')
reshot_image_url
```

```
Out[56]: 'https://www.reshot.com/build/photos/external-link-arrow-9fdf3fbf53535dc5eb554a9a5d5601329bb7bb7169ee72746a37fac0527b2967.svg'
```

- 이미지의 주소를 알고 있을 때 이미지를 내려받는 방법

```
In [57]: html_image = requests.get(reshot_image_url)

folder = "C:/Myexam/download"

# os.path.basename(URL)는 웹사이트나 폴더가 포함된 파일명에서 파일명만 분리하는 방법
imageFile = open(os.path.join(folder, os.path.basename(reshot_image_url)), 'wb')

# 이미지 데이터를 1000000 바이트씩 나눠서 저장하는 방법
chunk_size = 1000000
for chunk in html_image.iter_content(chunk_size):
    imageFile.write(chunk)
imageFile.close()
```

- 함수로 만들고 반복문으로 지정한 개수만큼 이미지를 내려받는 코드를 작성

```
In [58]: import requests
from bs4 import BeautifulSoup
import os

# URL(주소)에서 이미지 주소 추출
def get_image_url(url):
    html_image_url = requests.get(url).text
    soup_image_url = BeautifulSoup(html_image_url, "lxml")
    image_elements = soup_image_url.select('img')
    if(image_elements != None):
        image_urls = []
        for image_element in image_elements:
            image_urls.append(image_element.get('src'))
        return image_urls
    else:
        return None

# 폴더를 지정해 이미지 주소에서 이미지 내려받기
def download_image(img_folder, img_url):
    if(img_url != None):
        html_image = requests.get(img_url)
        # os.path.basename(URL)는 웹사이트나 폴더가 포함된 파일명에서 파일명만 분리
        imageFile = open(os.path.join(img_folder, os.path.basename(img_url)), 'wb')
```

```

chunk_size = 1000000 # 이미지 데이터를 1000000 바이트씩 나눠서 저장
for chunk in html_image.iter_content(chunk_size):
    imageFile.write(chunk)
    imageFile.close()
    print("이미지 파일명: '{0}'. 내려받기 완료!".format(os.path.basename(img_url)))
else:
    print("내려받을 이미지가 없습니다.")

# 웹 사이트의 주소 지정
reshot_url = 'https://www.reshot.com/search/animal'

figure_folder = "C:/Myexam/download" # 이미지를 내려받을 폴더 지정

reshot_image_urls = get_image_url(reshot_url) # 이미지 파일의 주소 가져오기

num_of_download_image = 3 # 내려받을 이미지 개수 지정
# num_of_download_image = len(reshot_image_urls)

for k in range(num_of_download_image):
    download_image(figure_folder, reshot_image_urls[k])
print("====")
print("선택한 모든 이미지 내려받기 완료!")

```

이미지 파일명: 'reshot-logo--mark-f8dfafbc1cc8fbf4dfa0e2f210265735aefa6e32f883b5a1fe27fd94f84719b3.svg'. 내려받기 완료!
 이미지 파일명: 'icon-no-photos-034c2399566c35bc56688f11795abdadf83f3c88fd167216a17ed55c2d65c73e.svg'. 내려받기 완료!
 이미지 파일명: 'external-link-arrow-9fdf3fbf53535dc5eb554a9a5d5601329bb7bb7169ee72746a37fac0527b2967.svg'. 내려받기 완료!
 =====
 선택한 모든 이미지 내려받기 완료!

- len(reshot_image_urls)로 이미지가 몇 개인지 확인

In [59]: num_of_download_image = len(reshot_image_urls)
 num_of_download_image

Out[59]: 7

정리

- webbrowser 라이브러리를 이용해 원하는 웹 사이트를 웹 브라우저로 열어서 접속하는 방법
- HTML 코드를 분석하고 requests 라이브러리를 이용해 HTML 소스를 가져오는 방법
- HTML 소스를 BeautifulSoup 라이브러리를 이용해 파싱하고 원하는 결과를 추출하는 방법
- 웹 사이트에 있는 이미지 파일을 내려받는 방법

In []: