웹 스크래핑

Update – 2018.07

Contents

- 웹 데이터 수집
- 웹 스크래핑 개요
- 웹 이해하기
- http 모듈 알아보기
- Requests 설치하기
- Requests 모듈을 이용한 파일 다운로드
- 파싱 개요
- BeautifulSoup 모듈

웹 데이터 수집 단계

데이터 수집의 3단계

1단계 : 대상선정

목적 데이터의 위치를 파악

2단계 : 수집

대상 위치에서 원하는 데이터를 수집

3단계 : 정리

수집된 데이터를 정리

웹 스크래핑 개요

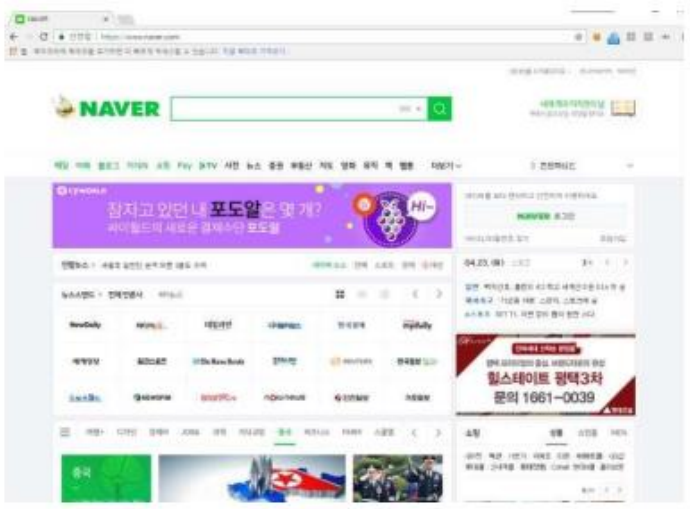
- 스크래핑과 크롤러
  - 스크래핑 - 각각의 페이지에서 정보는 추출하는 행위
  - 크롤러 - 자동으로 정보수집을 반복하는 프로그램

웹 데이터 수집 - 주의사항

- **수집 데이터의 처리와 저작권**
  - 웹 사이트의 정보는 기본적으로 저작물
  - 2016년 재정된 저작권법 제 30조 : 정보 해석을 목적으로 저작물을 복제/번안 가능
- **웹 사이트의 리소스 압박과 업무 방해**
  - 웹 사이트의 자원을 독점하게 되면 다른 사람이 웹 사이트를 이용할 수 없음
  - 무한 크롤러 사용 시 업무방해 혐의 적용 가능

웹의 이해

- **웹 페이지 Web Page**
  - 웹 상의 문서
  - 우리가 보고 있는 웹 사이트들은 문서로 이루어져 있다.
  - 텍스트, 그림, 소리, 동영상 등을 표현 가능
  - 대부분 HTML 이라는 언어로 이루어져 있음



웹의 이해

GET

- Body 없이 Header만으로 전송된다.
- 링크 / 북마크 가 가능하다.
- 요청에 길이 제한이 있다.

/test/demo_form.php**?name1=value1&name2=value2**

- URL의 ? 뒤에 쿼리 문자열이 올 수 있다.
- 쿼리 문자열은 key와 value를 가지고 있으며, 각 쿼리는 & 로 구분한다.

POST

- Body에 query data가 들어간다.
- 링크 / 북마크가 불가능하다.
- 데이터 길이에 제한이 없다.
- URL을 가지지 않으므로 주로 중요한 데이터를 다룰 때 사용한다.

http 모듈

- **urllib**
  - Python built-in module
  - 간편하게 HTTP request를 보낼 수 있음
  - 로그인 및 세션을 유지하기가 번거로움

- **requests**
  - 간편하게 HTTP request를 보낼 수 있음
  - 세션을 유지하기가 용이함
  - python2 / python3 완벽 지원
  - 코드가 간결하고 documentation이 잘 되어 있음

requests 모듈 설치하기

```
pip install requests
```

Requests: HTTP for Humans

Release v2.18.4. (Installation)

license Apache 2.0 | wheel yes | python 2.6, 2.7, 3.4, 3.5, 3.6 | codecov 88% | Say Thanks!

Requests is the only *Non-GMO* HTTP library for Python, safe for human consumption.

Html 파일 읽기

```
from urllib.request import urlopen

html = urlopen('http://google.com')
print(html.read())
```

Html 소스 출력

예외처리해서 Html 파일 읽기

```
from urllib.request import urlopen
from urllib.error import HTTPError
from urllib.error import URLError

try:
    html=urlopen('http://naver.on')
except HTTPError as e:
    print("http 에러")
except URLError as e:
    print('존재하지 않는 사이트. 주소 오류')
else:
    print(html.read())
```

존재하지 않는 사이트. 주소 오류

웹상의 이미지 저장하기

```
import urllib.request

# 웹상의 이미지 주소 복사를 url 변수에 삽입하기
url =
'https://www.google.co.kr/images/branding/googlelogo/2x/googlelogo_color
_272x92dp.png'

# 이미지 저장 위치 및 파일 이름 지정
savename = 'C:/__StudyPython/workspace/python_Basic/google.png'

# url이 가리키는 주소에 접근해서 해당 자원을 로컬 컴퓨터에 저장
urllib.request.urlretrieve(url,savename)
print('저장완료')
```

결과 확인은 저장폴더에서 확인

Parsing 개요

- 파싱 (parsing)
  - 가공되지 않은 문자열에서 필요한 부분을 추출하여 의미있는 (구조화된) 데이터로 만드는 과정

BeautifulSoup 모듈

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/

  - 파싱을 도와주는 강력한 python 라이브러리
  - 정규식을 작성할 필요 없이 tag, id, class 등의 이름으로 쉽게 파싱 가능
  - 쉽고 간결하며, documentation이 매우 잘 되어 있음

```
pip install beautifulsoup4
```

BeautifulSoup 모듈

- BeautifulSoup 객체 만들기

```
import urllib.request
from bs4 import BeautifulSoup

# url에 접속하여 연결된 후 HttpResponse 객체 생성
html = urlopen('http://naver.com')

# BeautifulSoup의 파서 기능이용하여 html 분석 객체 생성
bs = BeautifulSoup(html.read(),'html.parser')
print(type(bs)) #<class 'bs4.BeautifulSoup'>
```

BeautifulSoup 모듈

- 태그로 찾기 – 도트(.) 이용

bs = Beautifulsoup 객체

bs.태그1.태그2

bs.태그.next_sibling : 태그의 다음 요소 추출

bs.태그.string : 태그의 텍스트가 추출

```
import urllib.request
from bs4 import BeautifulSoup

html = urlopen('http://naver.com')
bs = BeautifulSoup(html.read(),'html.parser')

print(bs)  # url 전체 페이지 읽기
print(bs.h1) # h1 태그 소스만 읽어오기
print(bs.div) # 첫번째 div 태그 소스만 읽어오기
```

웹상의 html 페이지

BeautifulSoup 모듈

- 태그로 찾기 – 도트(.) 이용

Html을 변수로 지정해서 찾기

```
from bs4 import BeautifulSoup
html = """
<html>
    <body>
        <h1>Hello Python</h1>
        <p>웹페이지 분석
            <span>판다스</span>, <span>넘파이</span>
        </p>
        <p>웹스크래핑</p>
    </body>
</html>
"""
print(html)
soup = BeautifulSoup(html,"html.parser")
```

BeautifulSoup 모듈

- 태그로 찾기 – 도트(.) 이용

```
print('h1 태그 출력 : ',bs.html.body.h1)
print('첫번째 p 태그 출력 : ',bs.html.body.p)
print('공백 출력 : ', bs.html.body.p.next_sibling)
print('두번째 p 태그 출력 : ', bs.html.body.p.next_sibling.next_sibling.next_sibling)
print('첫번째 span 태그 출력 : ', bs.html.body.p.span)
print('첫번째 span 태그의 텍스트 출력 : ', bs.html.body.p.span.string)
print('두번째 span 태그 출력 : ', bs.html.body.p.span.next_sibling.next_sibling)
```

BeautifulSoup 모듈

- 아이디로 찾기 – find()

bs = Beautifulsoup 객체

bs.find(id='아이디명') : 아이디명의 첫번째 태그

bs.find(id='아이디명').string : 아이디명의 첫번째 태그 안의 텍스트

```
from bs4 import BeautifulSoup
html = """
<html>
    <body>
        <h1 id='title'>Hello Python</h1>
        <p id='body'>웹페이지 분석</p>
        <p>웹스크래핑</p>
    </body>
</html>
"""
bs = BeautifulSoup(html,"html.parser")
```

BeautifulSoup 모듈

- 아이디로 찾기 – find()

```
title = bs.find(id='title')
body = bs.find(id='body')
print('title 아이디의 텍스트 - ',title.string)
print('body 아이디의 텍스트 - ',body.string)
```

```
title 아이디의 텍스트 - Hello Python
body 아이디의 텍스트 - 웹페이지 분석
```

BeautifulSoup 모듈

- 클래스 이름으로 찾기 – find_all(class_=클래스이름)

bs = Beautifulsoup 객체

bs.find_all(class_='클래스이름')

리스트 구조로 추출

리스트변수.string으로

텍스트 추출 가능

```
from bs4 import BeautifulSoup
html = """
<html>
    <body>
        <ul>
            <li class='fruit'>사과</li>
            <li>양파</li>
            <li>고구마</li>
            <li class='fruit'>바나나</li>
        </ul>
    </body>
</html>
"""
```

BeautifulSoup 모듈

- 클래스 이름으로 찾기 – find_all(class_=클래스이름)

```
bs = BeautifulSoup(html,"html.parser")
fruits_list = bs.find_all(class_='fruit')
print(fruits_list,type(fruits_list))
for i in fruits_list:
    print(i)
for i in fruits_list:
    print(i.string)
```

BeautifulSoup 모듈

- 태그 여러 개 찾기 – find_all('태그')

  - bs = Beautifulsoup 객체
  - bs.find_all('태그')
  - 페이지에 삽입된 태그를 리스트 구조로 추출
  - 리스트변수.string으로 텍스트 추출 가능

BeautifulSoup 모듈

- 태그 여러 개 찾기 – find_all('태그')

```
from bs4 import BeautifulSoup
html = """
<html>
    <body>
        <ul>
            <li><a href='http://naver.com'>Naver</a></li>
            <li><a href='http://nate.com'>Nate</a></li>
            <li><a href='http://google.com'>Google</a></li>
            <li><a href='http://yahoo.com'>Yahoo</a></li>
            <li><a href='http://daum.com'>Daum</a></li>
        </ul>
    </body>
</html>
"""
```

BeautifulSoup 모듈

- 태그 여러 개 찾기 – find_all('태그')

```
bs= BeautifulSoup(html,"html.parser")
links = bs.find_all('a') #모든 a 태그 목록
print(type(links))
for i in links:
    print(i)
```

BeautifulSoup 모듈

- 태그의 속성 추출 – find_all('태그')

- bs = Beautifulsoup 객체
- bs.find_all('태그')
- For 문 사용시 리스트변수.attrs['속성']으로 태그 내부의 속성 추출 가능
- 리스트 구조

BeautifulSoup 모듈

- 태그의 속성 추출 – find_all('태그')

```
from bs4 import BeautifulSoup
html = """
<html>
    <body>
        <ul>
            <li><a href='http://naver.com'>Naver</a></li>
            <li><a href='http://nate.com'>Nate</a></li>
            <li><a href='http://google.com'>Google</a></li>
            <li><a href='http://yahoo.com'>Yahoo</a></li>
            <li><a href='http://daum.com'>Daum</a></li>
        </ul>
    </body>
</html>
"""
```

BeautifulSoup 모듈

- 태그의 속성 추출 – find_all('태그')

```
bs= BeautifulSoup(html,"html.parser")
links = bs.find_all('a')
for i in links :
    href=i.attrs["href"] #태그내부의 속성값
    print(href)
```

BeautifulSoup 모듈

- 태그의 속성 추출 – find_all(태그리스트)

  - bs = Beautifulsoup 객체
  - Bs.findAll(['태그1','태그2'...])
  - 리스트에 있는 태그만 모아서 리스트 생성
  - 리스트 구조

BeautifulSoup 모듈

- 태그의 속성 추출 – find_all(태그리스트)

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

html = urlopen('http://pythonscraping.com/pages/warandpeace.html')
bs = BeautifulSoup(html,"html.parser")

# h1, h2, h3 태그의 리스트 생성
allText = bs.find_all(['h1','h2','h3'])
print(allText)
```

BeautifulSoup 모듈

- 여러개의 클래스가 적용된 태그 리스트 추출하기
  find_all('태그명',{'class':{'클래스이름','클래스이름'}})

- bs = Beautifulsoup 객체
- bs.find_all('태그명',{'class','클래스이름'})
- 태그만 모아서 리스트 생성
- 리스트 구조

```
# span 태그 중에서 class가 green, red 인 태그
allText = bs.find_all('span',{'class':{'green','red'}})
```

BeautifulSoup 모듈

- 여러개의 클래스가 적용된 태그 리스트 추출하기
  find_all('태그명',{'class':{'클래스이름','클래스이름'}})

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

html = urlopen('http://pythonscraping.com/pages/warandpeace.html')
bs = BeautifulSoup(html,"html.parser")

# span 태그 중에서 class가 green, red 인 태그
allText = bs.find_all('span',{'class':{'green','red'}})

for i in allText:
    print(i.get_text())
```

BeautifulSoup 모듈

- 내용으로 찾기

```
bs.find_all(text='내용')
```

- bs = Beautifulsoup 객체
- bs.find_all(text='내용')
- 찾는 내용요소로 리스트 생성
- 단어 개수 구하기 – len() 이용

BeautifulSoup 모듈

- 내용으로 찾기

```
bs.find_all(text='내용')
```

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

html = urlopen('http://pythonscraping.com/pages/warandpeace.html')
bs = BeautifulSoup(html,"html.parser")

nameList = bs.find_all(text='the prince')
print(nameList)
print('갯수는? ',len(nameList))
```

BeautifulSoup 모듈

- 같은 클래스가 적용된 태그 리스트 추출하기
  findAll('태그명',{'class','클래스명')

- bs = Beautifulsoup 객체
- bs.findAll('태그명',{'class','클래스이름'})
- 리스트에 있는 태그만 모아서 리스트 생성
- 리스트 구조

BeautifulSoup 모듈

- 같은 클래스가 적용된 태그 리스트 추출하기
  findAll('태그명',{'class','클래스명')

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

html = urlopen('http://pythonscraping.com/pages/warandpeace.html')
bs = BeautifulSoup(html,"html.parser")

nameList = bs.findAll('span',{'class','green'})
print(nameList)
```

BeautifulSoup 모듈

- CSS 셀렉터로 추출 – select_one('CSS셀렉터')

  - bs = Beautifulsoup 객체
  - bs.select_one('css 셀렉터')
  - CSS 셀렉터 1개만 추출

BeautifulSoup 모듈

- CSS 셀렉터로 추출 – select_one('CSS셀렉터')

```
from bs4 import BeautifulSoup
# 분석 대상 HTML
html = """
<html>
 <body>
    <div id="main">
       <h1>도서목록</h1>
       <ul class="items">
        <li>자바 프로그래밍 입문</li>
        <li>HTML5</li>
        <li>Python</li>
       </ul>
    </div>
 </body>
</html>
"""
```

```
#HTML 분석
soup = BeautifulSoup(html, 'html.parser')
# 필요한 부분을 CSS 셀렉터로 추출
# 타이블 부분만 추출하기
# div#main > h1 공백 주의
h1 = soup.select_one("div#main > h1")
print(h1)
print(h1.string)
```

BeautifulSoup 모듈

- CSS 셀렉터로 추출 – select('CSS셀렉터')

- bs = Beautifulsoup 객체
- bs.select('css 셀렉터')
- CSS 셀렉터 조건으로 다중 추출
- 리스트 구조

BeautifulSoup 모듈

- CSS 셀렉터로 추출 – select('CSS셀렉터')

```
from bs4 import BeautifulSoup
# 분석 대상 HTML
html = """
<html>
 <body>
    <div id="main">
       <h1>도서목록</h1>
       <ul class="items">
        <li>자바 프로그래밍 입문
</li>
        <li>HTML5</li>
        <li>Python</li>
       </ul>
    </div>
 </body>
</html>
"""
```

```
#HTML 분석
soup = BeautifulSoup(html, 'html.parser')
# 필요한 부분을 CSS 셀렉터로 추출
# 목록 부분 추출
li_list = soup.select("div#main > ul.items > li")
print(li_list)
for li in li_list:
    print('li = ',li.string)
```