Week 8. 웹 정보 수집(스크래핑, 크롤링)

웹 스크래핑(scraping)

비 정형화 되어있는 웹페이지의 데이터들을 긁어모아 정형화된 정보 형식으로 바꾸는 작업

(ex. 쇼핑몰 가격비교 - 각 쇼핑몰 상품 페이지에서 상품 이름 가격 등을 추출)

웹 크롤링(crawling)

웹의 링크 형태를 이해하고 웹 페이지를 수집하는 작업

(ex. google bot이 전 세계 웹사이트를 수집하여 검색 서비스를 제공)

HTML

웹 페이지를 표현하는 `마크업 언어`

태그, 요소, 속성 등의 구성요소를 이용해 문서 구성을 구조적으로 표현한다.

이렇게 구조화된 문서는 효율적으로 parsing(탐색)하고 원하는 정보를 mining(찾아)낼 수 있고 심지어는 스택이나 큐 등의 자료구조를 이용해서 문서를 이루는 내용들을 일일이 파싱할 것도 없이 파이썬 기본 모듈로 제공되는 `HTMLParser`를 이용하거나 `BeautifulSoup` 등의 파이썬 라이브러리를 통해서 더욱 손쉽게 파싱할 수도 있다.

urllib(python 기본) 패키지 사용해보기

- urlencode : URL 인수 문자열 생성
- urlopen : 웹서버 연결
- urlretrieve : 웹서버 연결 및 HTML 문서 저장

Import

```
In [25]: import urllib.request
```

url에서 데이터를 받아오기

일반적으로 `type()` 함수를 찍어보고 `str` 이 아닌 경우 (한글 페이지 인 경우) 뒤에 `decode("utf-8")` 함수를 이용하여 위 형식으로 디코딩 해야 함

In [26]:
```
url = 'http://konanacademy.github.io/da/'
# 핸들 객체 얻기
data = urllib.request.urlopen(url)
text = data.read().decode("utf-8")
print(text[:100])
```

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" cont
```

또한 보안이 철저한 웹사이트의 경우, 접속자의 header, 즉 접속자의 정보를 확인하는 것이 보통이므로 아래 코드를 추가할 필요가 있음.

```
req = urllib.Request(url)
req.add_header('User-agent', 'Mozila/5.0 (compatible; MSIE 5.5; Windows NT)')
response = urllib2.urlopen(req)
```

(노가다)파싱

In [3]:
```
# 특정 기준으로 잘라내서 리스트로 만든 후 https 링크만 추출해보자.
words = text.split('https://')
words[1]
```

Out[3]:
```
'github.com/KonanAcademy/da"> GitHub </a></p>\n    </header>\n    <section>\n
<h3>\n            <a name="welcome-to-github-pages" class="anchor" href="#welcome-to
-github-pages"><span\n                    class="octicon octicon-link"></span></a>We
lcome to KonanAcademy\'s Data Analysis Baisc Pages.</h3>\n\n        <p>코난아카데미
 데이터분석 기초</p>\n        <pre><code>\n            데이터 분석 기초 (2015.2~2015.3)
\n            2015년 2월 24일 - 2015년 3월\n            매주 화, 목 저녁. 7시~9시\n
        장소: 코난테크놀로지 \n        </code></pre>\n        <table>\n          <thea
d>\n            <tr>\n              <th align="left"></th>\n              <th align
="left">날짜</th>\n              <th align="left">제목</th>\n              <th align
="left">세부사항</th>\n            </tr>\n          </thead>\n          <tbody>\n
        <tr>\n              <td align="left">0</td>\n              <td align="left">
2/24</td>\n              <td align="left"><ui><li><a href="http://nbviewer.ipython.o
rg/github/KonanAcademy/da/blob/master/part1/00_what_is_da.ipynb">데이터분석이란?</a>
</li><li><a href="http://nbviewer.ipython.org/github/KonanAcademy/da/blob/master/par
t1/00_1_python_example.ipynb">데이터 분석 예제1(Python)</a></li><li><a href="'
```

In [4]:
```
words[1].split('">')[0]
```

Out[4]:
```
'github.com/KonanAcademy/da'
```

a little bit of 자동화

In [5]:
```
links = []

for word in words[1:5] :
    dummy_link = "https://" + word.split('">')[0]
    links.append(dummy_link)

print(links)
```

```
['https://github.com/KonanAcademy/da', 'https://github.com/KonanAcademy/da/blob/mast
er/part1/00_2_R_example.md', 'https://github.com/KonanAcademy/da/blob/master/part1/0
0_4_R_example.md', 'https://view.officeapps.live.com/op/view.aspx?src=http%3A%2F%2Fw
ww.fil.ion.ucl.ac.uk%2Fspm%2Fdoc%2Fmfd%2F2010%2Fpage1%2FLinearAlgebra.ppt']
```

그럼 주가 데이터는?

In [6]:
```
import urllib

params = urllib.parse.urlencode({"a": 4, "b": 20, "c": 2016, "d": 6, "e": 30, "f": 20
16, "s": "^KS11"})
print (params)
```

```
f=2016&s=%5EKS11&b=20&a=4&e=30&c=2016&d=6
```

In [7]:
```
url = 'http://ichart.finance.yahoo.com/table.csv?g=d&ignore=.csv&%s' % params
data = urllib.request.urlopen(url).read().decode('utf-8')
print(data[:200])
```

```
Date,Open,High,Low,Close,Volume,Adj Close
2016-07-29,2023.23999,2028.98999,2016.189941,2016.189941,366400,2016.189941
2016-07-28,2029.069946,2029.380005,2014.359985,2021.099976,350900,2021.099976
2016
```

requests 패키지 사용해보기

- HTTP protocols (`get` , `post` , `put` , `delete` , `head` , `options`) 을 더욱 쉽게 사용할 수 있도록 한 third-party 패키지
- anaconda를 설치하면 기본적으로 설치되어 있음

Import

In [1]:
```
import requests
```

url에서 데이터를 받아오기

In [29]:
```
url = 'http://konanacademy.github.io/da/'

# `get`을 이용하여 url 요청
req = requests.get(url)
print(req.content.decode('utf-8')[:500])
```

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="chrome=1">

    <meta property="og:title" content="코난아카데미" />
    <meta property="og:description" content="코난아카데미 데이터분석 기초" />
    <meta property="og:type" content="website" />
    <meta property="og:url" content="http://konanacademy.github.io/da" />
    <title>데이터분석 기초</title>

    <link rel="stylesheet" href="stylesheets/styles.css">
    <link rel="stylesheet" href="stylesheets/pygment_trac
```

html 소스 가져오기

In [3]:
```
# HTML 소스 가져오기
html = req.content.decode('utf-8')
print(html[:300])

# HTTP Header 가져오기
header = req.headers

# HTTP Status 가져오기 (200: 정상)
status = req.status_code

# HTTP가 정상적으로 되었는지 (True/False)
is_ok = req.ok
```

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="chrome=1">

    <meta property="og:title" content="코난아카데미" />
    <meta property="og:description" content="코난아카데미 데이터분석 기초" />
    <meta property="og:type" content="website" />
    <meta property
```

이제 위에서 가져온 html text에서 좋은 정보를 뽑아내려면 어떻게 해야할까?

> 각각 정보의 단위로 잘라내는 것을 parsing 이라 하고 이를 엄청나게 도와주는 또다른 third-party library가 있다

beautifulSoup 패키지 사용해보기

- HTML 문서 파싱 및 태그 검색
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

BeautifulSoup 은 html 태그를 `Tag object` 로 저장한다

Import

In [7]:
```
from bs4 import BeautifulSoup as bs
```

HTML 데이터 parsing

- `html.parser` : html 데이터
- `lxml-xml` : xml 데이터

위의 에제에서 `https` 링크 자료들을 가져오고 싶었으니, html 에서 하이퍼링크를 담당하는 `<a>` 태그를 가져오려고 한다

- `find_all()` : 대상 태그를 가지는 html 데이터를 list객체로 반환한다
- `parent` : 계층구조상 한칸 위에 있는 태그를 지칭한다
- `content` : 계층구조상 한칸 아래에 있는 태그 목록을 반환한다
- `nextSibling` , `previousSibling` : 계층구조상 같은 위치에 있는 바로 앞뒤 태그를 지칭한다.

In [8]:
```
navigator = bs(html, 'html.parser')
```

In [10]:
```
a_tag_list = navigator.find_all('a')

print(a_tag_list[:5])
```

```
[<a href="http://www.konantech.com/">코난테크놀로지</a>, <a href="http://KonanAcadem
y.github.io"> 코난아카데미 메인</a>, <a href="https://github.com/KonanAcademy/da"> Git
Hub </a>, <a class="anchor" href="#welcome-to-github-pages" name="welcome-to-github-
pages"><span class="octicon octicon-link"></span></a>, <a href="http://nbviewer.ipyt
hon.org/github/KonanAcademy/da/blob/master/part1/00_what_is_da.ipynb">데이터분석이란?
</a>]
```

> 아직 부족함. 정확하게는 `a` 태그 안의 `href` 컴포넌트의 값을 받아오고 싶으니까

In [23]:
```
real_a_tag_lists = []

for link in a_tag_list:
    real_a_tag_lists.append(link.get('href')) # 실제 value는 link.string

print(real_a_tag_lists[:5])
```

```
['http://www.konantech.com/', 'http://KonanAcademy.github.io', 'https://github.com/K
onanAcademy/da', '#welcome-to-github-pages', 'http://nbviewer.ipython.org/github/Kon
anAcademy/da/blob/master/part1/00_what_is_da.ipynb']
```

다 한건가? .. 여기 보니까 `#welcome-to-github-pages` 라는 게 있네요.. 이건 실제로 `http` 웹 링크가 아니니까... 조금 더 수정을 해 보면

In [61]:
```
real_a_tag_lists = []

# startswith : `http`라는 단어로 시작하면 True를 반환하여 if문을 실행함
for link in a_tag_list:
    if link.get('href').startswith("http"):
        real_a_tag_lists.append(link.get('href'))
    else:
        print('http 가 아닌것들:', link.get('href'))

print(real_a_tag_lists[:5])

# with open('data/links.txt', 'w') as f:
#     for i in real_a_tag_lists:
#         f.write(i+'\n')

import pandas as pd

df = pd.DataFrame(real_a_tag_lists)

df.to_csv('data/lists.csv', header=False)
```

```
http 가 아닌것들: #welcome-to-github-pages
['http://www.konantech.com/', 'http://KonanAcademy.github.io', 'https://github.com/K
onanAcademy/da', 'http://nbviewer.ipython.org/github/KonanAcademy/da/blob/master/par
t1/00_what_is_da.ipynb', 'http://nbviewer.ipython.org/github/KonanAcademy/da/blob/ma
ster/part1/00_1_python_example.ipynb']
```

`#welcome-to-github-pages` 가 빠졌다. 끝!

다른 방법 (정규화 포맷 re 라이브러리)

In [15]:
```
import re

links = []

for link in navigator.find_all(href=re.compile("http")):
    links.append(link.get('href'))

print(links[:5])
```

```
['http://www.konantech.com/', 'http://KonanAcademy.github.io', 'https://github.com/KonanAcademy/da', 'http://nbviewer.ipython.org/github/KonanAcademy/da/blob/master/part1/00_what_is_da.ipynb', 'http://nbviewer.ipython.org/github/KonanAcademy/da/blob/master/part1/00_1_python_example.ipynb']
```

다시 주가 데이터

url 데이터 받기

In [64]:
```
import requests
from bs4 import BeautifulSoup

url = "https://www.google.com/finance/historical?q=KRX%3AKOSPI200"
req = requests.get(url)
print(req.text[:1000])
```

```
<!DOCTYPE html><html><head><script>(function(){(function(){function e(a){this.t={};this.tick=function(a,c,b){var d=void 0!=b?b:(new Date).getTime();this.t[a]=[d,c];if(void 0==b)try{window.console.timeStamp("CSI/"+a)}catch(h){}};this.tick("start",null,a)}var a;if(window.performance)var d=(a=window.performance.timing)&&a.responseStart;var f=0<d?new e(d):new e;window.jstiming={Timer:e,load:f};if(a){var c=a.navigationStart;0<c&&d>=c&&(window.jstiming.srt=d-c)}if(a){var b=window.jstiming.load;0<c&&d>=c&&(b.tick("_wtsrt",void 0,c),b.tick("wtsrt_","_wtsrt",d),b.tick("tbsd_","wtsrt_"))}try{a=null,window.chrome&&window.chrome.csi&&(a=Math.floor(window.chrome.csi().pageT),b&&0<c&&(b.tick("_tbnd",void 0,window.chrome.csi().startE),b.tick("tbnd_","_tbnd",c))),null==a&&window.gtbExternal&&(a=window.gtbExternal.pageT()),null==a&&window.external&&(a=window.external.pageT,b&&0<c&&(b.tick("_tbnd",void 0,window.external.startE),b.tick("tbnd_","_tbnd",c))),a&&(window.jstiming.pt=a)}catch(g){}})();}).cal
```

BeautifulSoup 로 text 데이터를 xml 형식으로 구조화

In [72]:
```
navigator = BeautifulSoup(req.text, 'lxml')
import pandas as pd
```

In [73]:
```
import dateutil

list_records = []

table = navigator.find("table", class_="historical_price")

for i, r in enumerate(table.find_all('tr')):
    for j, c in enumerate(r.find_all('td')):
        if j == 0:
            record = {"date": dateutil.parser.parse(c.text.strip())}
        elif j == 1:
            record["open"] = float(c.text.strip())
        elif j == 2:
            record.update({"high": float(c.text.strip())})
        elif j == 3:
            record.update({"low": float(c.text.strip())})
        elif j == 4:
            record.update({"close": float(c.text.strip())})
        elif j == 5:
            record.update({"volume": int(c.text.strip().replace(',',''))})
    try:
        list_records.append(record)
    except:
        pass

list_records[:3]
```

Out[73]:
```
[{'close': 301.71,
  'date': datetime.datetime(2017, 5, 24, 0, 0),
  'high': 302.79,
  'low': 301.23,
  'open': 302.32,
  'volume': 111991000},
 {'close': 311.77,
  'date': datetime.datetime(2017, 7, 5, 0, 0),
  'high': 312.1,
  'low': 309.64,
  'open': 309.87,
  'volume': 68312000},
 {'close': 310.46,
  'date': datetime.datetime(2017, 7, 4, 0, 0),
  'high': 312.82,
  'low': 309.78,
  'open': 312.42,
  'volume': 89184000}]
```

구조화 된 데이터를 pandas dataframe으로

In [19]:
```
df = pd.DataFrame(list_records,
                  columns=["date", "open", "high", "low", "close", "volume"])
df.tail()
```

Out[19]:

	date	open	high	low	close	volume
25	2017-03-02	272.14	273.34	271.35	272.65	105762000
26	2017-02-28	269.27	270.62	269.10	270.06	88888000
27	2017-02-27	270.35	270.62	268.72	268.97	81075000
28	2017-02-24	272.61	272.84	269.76	270.38	116666000
29	2017-02-23	272.61	273.18	272.18	272.89	64488000

read_html 을 이용하여 dataframe 바로 생성

In [10]:
```
import requests as rs
from bs4 import BeautifulSoup
import html5lib
import pandas as pd

url = "https://www.google.com/finance/historical?q=KRX%3AKOSPI200"
req = rs.get(url)

soup = BeautifulSoup(req.text, 'html.parser')
table = soup.find("table", class_="historical_price")

df_table = pd.read_html(str(table), header=0)

df_table[0]
```

Out[10]:

	Date	Open	High	Low	Close	Volume
0	Jul 5, 2017	309.87	312.10	309.64	311.77	68312000
1	Jul 4, 2017	312.42	312.82	309.78	310.46	89184000
2	Jul 3, 2017	312.50	312.93	310.85	312.39	71751000
3	Jun 30, 2017	310.61	311.76	310.17	311.76	73223000
4	Jun 29, 2017	312.86	313.79	312.26	312.56	103106000
5	Jun 28, 2017	310.71	312.03	310.47	310.84	94923000
6	Jun 27, 2017	311.72	312.95	311.05	312.12	131745000
7	Jun 26, 2017	310.53	312.22	310.13	311.89	71899000
8	Jun 23, 2017	309.36	310.58	308.96	310.26	81460000
9	Jun 22, 2017	308.60	309.47	307.28	309.47	88484000
10	Jun 21, 2017	307.61	308.00	306.13	307.52	100934000
11	Jun 20, 2017	310.55	310.63	308.72	309.31	122387000
12	Jun 19, 2017	307.14	309.29	306.19	308.61	87803000
13	Jun 16, 2017	306.97	307.40	305.82	306.79	87835000
14	Jun 15, 2017	308.02	308.83	305.04	306.69	95414000
15	Jun 14, 2017	309.53	310.05	307.03	307.95	113446000
16	Jun 13, 2017	306.25	308.29	306.25	308.02	94388000
17	Jun 12, 2017	307.74	308.31	305.59	306.24	95851000
18	Jun 9, 2017	307.51	310.02	307.17	309.38	118710000
19	Jun 8, 2017	305.70	306.87	304.17	306.25	126075000
20	Jun 7, 2017	306.94	307.59	305.42	305.68	87994000
21	Jun 5, 2017	308.52	308.59	306.79	307.33	76116000
22	Jun 2, 2017	305.34	308.01	305.27	307.83	94028000
23	Jun 1, 2017	304.74	305.30	302.99	304.03	77246000
24	May 31, 2017	303.79	306.18	303.64	304.67	114684000
25	May 30, 2017	306.89	306.99	303.11	304.59	94122000
26	May 29, 2017	307.98	309.32	305.13	306.52	105979000
27	May 26, 2017	305.41	308.51	305.07	306.96	98095000
28	May 25, 2017	302.83	305.34	302.18	305.22	100091000
29	May 24, 2017	302.32	302.79	301.23	301.71	111991000