Lab. request 사용하기

1. requests module
   1)Python HTTP for Humans'
   2)Python에서 웹 데이터를 받아올 때 가장 많이 사용하는 module이다.
   3)http://docs.python-requests.org/en/master/
   4)Install
      $ pip install requests

   5)웹 페이지에 접속하기
      -요청하면 서버가 응답한 값을 반환한다.
      -바로 응답 코드를 확인할 수 있다.
      -또는 status__code 속성을 통해서도 응답 코드를 확인할 수 있다.

```
import requests
url = 'https://www.naver.com'
naver = requests.get(url)
print(naver)
------------------------------
<Response [200]>

import requests
url = 'https://www.naver.com'
naver = requests.get(url)
print(naver)
print(naver.status_code)
------------------------------
<Response [200]>
200
```

   6)요청 페이지를 찾을 수 없을 때 404코드를 반환한다.

```
import requests

def url_check(url):
    res = requests.get(url)

    print(res)

    sc = res.status_code

    if sc == 200:
        print("%s 요청성공"%(url))
    elif sc == 404:
        print("%s 찾을 수 없음" %(url))
    else:
        print("%s 알수 없는 에러 : %s"%(url, sc))


url_check("https://www.naver.com")
url_check("https://www.naver.com//a")
------------------------------------------------------
<Response [200]>
https://www.naver.com 요청성공
<Response [404]>
https://www.naver.com//a 찾을 수 없음
```

   7)header 가져오기

```
import requests

url = "https://www.naver.com"

res = requests.get(url)

print(res)
print(res.headers)
----------------------------------------
<Response [200]>
{'Server': 'NWS', 'Date': 'Tue, 27 Aug 2019 11:28:32 GMT', 'Content-Type': 'text/html; charset=UTF-8',
'Transfer-Encoding': 'chunked', 'Connection': 'keep-alive', 'Set-Cookie':
'PM_CK_loc=e28d849e90b7a5d7c3ee6e69e31d9ba3a3ac7923a7b342c194bb9591a177e4a7; Expires=Wed, 28 Aug 2019
11:28:32 GMT; Path=/; HttpOnly', 'Cache-Control': 'no-cache, no-store, must-revalidate', 'Pragma': 'no-cache', 'P3P':
'CP="CAO DSP CURa ADMa TAIa PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"',
'X-Frame-Options': 'DENY', 'X-XSS-Protection': '1; mode=block', 'Content-Encoding': 'gzip', 'Strict-Transport-Security':
'max-age=63072000; includeSubdomains', 'Referrer-Policy': 'unsafe-url'}
```

   8)응답 객체에서 header를 dict 형태로 가져온다.

```
import requests
```

```
url = "https://www.naver.com"
res = requests.get(url)
print(res)

headers = res.headers
print(headers['Set-Cookie'])
-------------------------------------------------
<Response [200]>
PM_CK_loc=e28d849e90b7a5d7c3ee6e69e31d9ba3a3ac7923a7b342c194bb9591a177e4a7; Expires=Wed, 28 Aug 2019
11:30:28 GMT; Path=/; HttpOnly


import requests

url = "https://www.naver.com"
res = requests.get(url)
print(res)

headers = res.headers

for header in headers:
    print(headers[header])
-------------------------------------------------
<Response [200]>
NWS
Tue, 27 Aug 2019 11:31:58 GMT
text/html; charset=UTF-8
chunked
keep-alive
PM_CK_loc=e28d849e90b7a5d7c3ee6e69e31d9ba3a3ac7923a7b342c194bb9591a177e4a7; Expires=Wed, 28 Aug 2019
11:31:58 GMT; Path=/; HttpOnly
no-cache, no-store, must-revalidate
no-cache
CP="CAO DSP CURa ADMa TAIa PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"
DENY
1; mode=block
gzip
max-age=63072000; includeSubdomains
unsafe-url
```

9)HTML code 보기

```
import requests
url = 'https://www.naver.com'
naver = requests.get(url)
print(naver.text)
```

10)HTML code 보기2

-content 속성을 사용하면 한글을 binary 형태(인코딩)로 바꿔서 가져온다.

-binary 형태로 HTML을 가져올 경우 text속성을 이용하였을 때 발생하는 한글 문자 깨지는 현상을 방지할 수 있다.

```
import requests
url = 'https://www.naver.com'
naver = requests.get(url)
print(naver.content)
```

11)인코딩 확인

```
import requests
url = 'https://www.naver.com'
naver = requests.get(url)
print(naver.encoding)
------------------------------
UTF-8
```

12)JSON 형식 처리

-Response 객체의 json()을 통해 JSON 형식의 응답을 간단하게 decoding 해서 dict 또는 list 추출 가능

```
r = requests.get('http://weather.livedoor.com/forecast/webservice/json/v1?city=130010')
r.json()
-----------------------------------------------------------------
{'pinpointLocations': [{'link': 'http://weather.livedoor.com/area/forecast/1310100', 'name': '千代田区'}, {'link':
'http://weather.livedoor.com/area/forecast/1310200'....
```

13)Data 보내기

-requests로 요청할 때 data를 실어 보낼 수 있다.

-querystring 같은 경우 URL에 직접 표현할 수 있지만, querystring을 만들어야 하는 번거로움이 있다.

-하지만 data를 dict 형태로 만들어 보내는 방식으로 번거로움을 줄일 수 있다.

-data 뿐만 아니라 header, cookie 같은 data도 원하는 값으로 변경하여 요청 가능하다.
-특정 page는 header의 user-agent가 비었거나, cookie가 비어있을 경우 정상적으로 HTML 처리에 문제가 있을 수 있다.
-이럴 때는 header나 cookie를 직접 만들어야 한다.
-querystring data를 만들어 요청하기

```
import requests
url = "https://pjt3591oo.github.io/"
res = requests.get(url, params={"key1": "value1", "key2": "value2"})
print(res.url)
--------------------------------------------------
https://pjt3591oo.github.io/?key1=value1&key2=value2
```

14)get()함수는 HTTP method의 GET에 대응된다.
-post(), put(), delete(), head(), options()는 각각 POST, PUT, DELETE, HEAD, OPTIONS에 대응된다.

15)Session 객체
-여러 개의 page를 연속으로 crawling할 때는 효과적이다.
-HTTP header 또는 Basic 인증 등의 설정을 한 번만 하고 여러번 재사용 가능하다.

2. pprint module 이용하기
1)대량의 data를 보기 쉽게 표시해주는 표준 module
2)pprint(prettyprint)

```
import requests
import pprint
url = 'https://www.naver.com'
naver = requests.get(url)
pprint.pprint(naver.text)
```

3. requests를 사용하여 API에 접근하기
1)기상청 RSS(http://www.weather.go.kr/weather/lifenindustry/sevice_rss.jsp)를 이용하자.
2)2013년 동네 예보 RSS
http://www.weather.go.kr/images/weather/lifenindustry/dongnaeforecast_rss.pdf
3)주소선택후 rss button click하면 zone을 알 수 있다.
-예:서울특별시 강남구 청담동
http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1168056500

```
import requests
import pprint
api_uri = 'http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1168056500'
weather_data = requests.get(api_uri).text
pprint.pprint(weather_data)
--------------------------------------------------------------------------------
('<?xml version="1.0" encoding="UTF-8" ?>\n'
 '<rss version="2.0">\n'
 '<channel>\n'
 '<title>기상청 동네예보 웹서비스 - 서울특별시 강남구 청담동 도표예보</title>\n'
 '<link>http://www.kma.go.kr/weather/main.jsp</link>\n'
 '<description>동네예보 웹서비스</description>\n'
 '<language>ko</language>\n'
 '<generator>동네예보</generator>\n'
 '<pubDate>2019년 01월 15일 (화)요일 14:00</pubDate>\n'
 ' <item>\n'
 '<author>기상청</author>\n'
 '<category>서울특별시 강남구 청담동</category>\n'
 '<title>동네예보(도표) : 서울특별시 강남구 청담동 '

 '[X=61,Y=126]</title><link>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;dongCod
 e=1168056500</link>\n'

 '<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;dongCode=1168056500</guid
 >\n'
 '<description>\n'
 ...
 ...
```

4)get method의 params option을 활용하기

```
api_url = 'http://www.kma.go.kr/wid/queryDFSRSS.jsp'

payload = {'zone':'1168056500'}

weather_data = requests.get(api_url, payload).text

weather_data
-----------------------------------------------
'<?xml version="1.0" encoding="UTF-8" ?>\n<rss version="2.0">\n<channel>\n<title>기상청 동네예보 웹서비스 - 서울특별시
강남구 청담동 도표예보</title>\n<link>http://www.kma.go.kr/weather/main.jsp</link>\n<description>동네예보
웹서비스</description>\n<language>ko</language>\n<generator>동네예보</generator>\n<pubDate>2019년 01월 15일
(화)요일 14:00</pubDate>\n <item>\n<author>기상청</author>\n<category>서울특별시 강남구
```

```
청담동</category>\n<title>동네예보(도표) : 서울특별시 강남구 청담동
[X=61,Y=126]</title><link>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;dongCo
de=1168056500</link>\n<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;do
ngCode=1168056500</guid>\n<description>\n <header>\n  <tm>201901151400</tm>\n  <ts>4</ts>\n
...
...
```

5)xml.etree.ElementTree module 사용하기
-Python에서 xml data를 다루기 위한 module

```
import xml.etree.ElementTree as ET
import pandas as pd

xml_data = ET.fromstring(weather_data)

for tag in xml_data.iter('data'):
    print(tag.find('hour').text + "/" + tag.find('temp').text)
-----------------------------------------------------------------------
18/-2.0
21/-4.0
24/-5.0
...
...

list = []
for tag in xml_data.iter('data'):
    dic = {'hour': tag.find('hour').text,
   'day' : tag.find('day').text,
   'temp' : tag.find('temp').text,
   'tmx' : tag.find('tmx').text,
   'tmn' : tag.find('tmn').text,
   'sky' : tag.find('sky').text,
   'pty' : tag.find('pty').text,
   'wfKor' : tag.find('wfKor').text,
   'wfEn' : tag.find('wfEn').text}
list.append(dic)

df = pd.DataFrame(list, columns=['hour', 'day', 'temp','tmx','tmn','sky', 'pty', 'wfKor', 'wfEn'])

df
-----------------------------------------------------------------
   hour day  temp    tmx     tmn     sky  pty  wfKor       wfEn
0  180     -2.0    -999.0  -999.0  2    0    구름 조금  Partly Cloudy
1  210     -4.0    -999.0  -999.0  2    0    구름 조금  Partly Cloudy
2  240     -5.0    -999.0  -999.0  1    0    맑음       Clear
3  3    1     -6.0      -1.0    -8.0    1    0     맑음        Clear
4  6    1     -7.0      -1.0    -8.0    1    0     맑음        Clear
```

4. Lab : RSS Scraping

1)전자신문 RSS

```
import pandas as pd
import xml.etree.ElementTree as ET
import requests

api_url = 'http://rss.etnews.com/Section901.xml'

etnews_data = requests.get(api_url).text
etnews_data
--------------------------------------------------
'<?xml version="1.0" encoding="utf-8" ?>\n<rss version="2.0">\r\n    <channel>\r\n   ...
...

xml_data = ET.fromstring(etnews_data)
for tag in xml_data.iter('item'):
    print(tag.find('title').text + "," + tag.find('pubDate').text)
-----------------------------------------------------------------------
애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍,Tue, 15 Jan 2019 17:00:00 +0900
지난해 드론 자격증 취득자 전년비 4배 증가…실효성 지적도,Tue, 15 Jan 2019 17:00:00 +0900
화웨이, 韓 스마트워치 시장 진출,Tue, 15 Jan 2019 17:00:00 +0900
...
...

list = []

for tag in xml_data.iter('item'):
    dic = {'Title':tag.find('title').text,
           'Link':tag.find('link').text,
           'Author':tag.find('author').text,
           'PubDate':tag.find('pubDate').text,
           'Guid':tag.find('guid').text}
    list.append(dic)
```

```
list
----------------------------------------
[{'Title': "애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍",
  'Link': 'http://www.etnews.com/20190115000273',
  'Author': '윤건일',
  'PubDate': 'Tue, 15 Jan 2019 17:00:00 +0900',
  'Guid': '20190115000273'},
  ...
  ...

df = pd.DataFrame(list, columns=['Title', 'Link', 'Author', 'PubDate', 'Guid'])
df.head()
-----------------------------------------------
...
...

df.info()
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 5 columns):
Title      30 non-null object
Link       30 non-null object
Author     30 non-null object
PubDate    30 non-null object
Guid       30 non-null object
dtypes: object(5)
memory usage: 1.2+ KB
```

5. Login이 필요한 site에서 download하기

1)한빛출판네트워크 login page

http://www.hanbit.co.kr/member/login.html

2)마이한빛

http://www.hanbit.co.kr/myhanbit/myhanbit.html

3)로그인 폼

```
<form name="frm" id="frm" action="#" method="post">
<input name="retun_url" id="retun_url" type="hidden" value="http://www.hanbit.co.kr/myhanbit/myhanbit.html" class="i_text" size="100">
<div class="login_left">
   <fieldset>
      <legend>한빛출판네트워크 로그인</legend>

      <label class="i_label" for="login_id"><strong style="position: absolute; visibility: visible;"></strong>
         <input name="m_id" id="m_id" type="text" value="" class="i_text" placeholder="아이디"
         onkeydown="javascript:if(event.keyCode==13){login_proc(); return false;}">
      </label>

      <label class="i_label" for="login_pw"><strong style="position: absolute;"></strong>
         <input name="m_passwd" id="m_passwd" type="password" value="" class="i_text" placeholder="비밀번호"
         onkeydown="javascript:if(event.keyCode==13){login_proc(); return false;}">
      </label>

      <label>
         <input type="button" name="login_btn" id="login_btn" value="로그인" class="btn_login">
      </label>

      <label class="i_label2">
         <input type="checkbox" name="keepid" id="keepid" value="1" class="i_check"><strong>아이디 저장</strong>
      </label>
   </fieldset>

   <ul class="login_btn">
      <li><a href="/member/find_id.html" class="btn_idc">아이디 찾기</a></li>
      <li><a href="/member/find_pw.html" class="btn_pwc">비밀번호 찾기</a></li>
      <li><a href="/member/member_agree.html" class="btn_joinc">회원가입</a></li>
   </ul>
</div>
</form>
```

4)m_id, m_passwd라는 값(name 속성의 값)을 입력하고, 입력 양식을 제출하면 즉 submit하면 login되는 구조이다.

5)Login 과정 분석

-Chrome의 Network tab

-상단의 filter중에서 'Doc'를 클릭한다.

-그 위의 [Preserve log] check

--원래 [Network] tab은 page가 이동할 때 기존 page와 관련된 내용을 지우고, 새로운 page의 내용만 띄운다.

--하지만 이것을 check하면 내용을 지우지 않고 유지해준다.

--login 과정을 분석하려면 web page를 어떻게 이동하는지 알아야하므로 반드시 체크한다.

-로그인을 수행한다.
-그러면, login.html -> login_pro.php -> myhanbit.html의 과정으로 보인다.
-하나하나의 과정을 클릭하면 자세한 내용을 볼 수 있다.
-login_proc.php를 클릭해보자.
-Request Method가 POST임을 알 수 있다.
-그 아래로 계속 내려가면 Form Data 섹션의 m_id와 m_passwd의 값을 확인할 수 있다.
-다시 말해, login_proc.php페이지에 입력 양식 data를 POST로 전달하면 로그인한다는 것을 알 수 있다.

6)Python으로 login하기

```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin

# 아이디와 비밀번호 지정하기
USER = "devexpert"
PASS = "P@$$W0rd"

# 세션 시작하기
session = requests.session()

# 로그인하기
login_info = {
    "m_id": USER,  # 아이디 지정
    "m_passwd": PASS  # 비밀번호 지정
}

url_login = "http://www.hanbit.co.kr/member/login_proc.php"
res = session.post(url_login, data=login_info)
res.raise_for_status() # 오류가 발생하면 예외가 발생.

# 마이페이지에 접근하기
url_mypage = "http://www.hanbit.co.kr/myhanbit/myhanbit.html"
res = session.get(url_mypage)
res.raise_for_status()

# 마일리지와 이코인 가져오기
soup = BeautifulSoup(res.text, "html.parser")
mileage = soup.select_one(".mileage_section1 span").get_text()
ecoin = soup.select_one(".mileage_section2 span").get_text()
print("마일리지: " + mileage)
print("이코인: " + ecoin)
```

```
------------------------------------------------------------------
마일리지: 0
이코인: 0
```

6. Web page image 추출하기

```
import requests
r = requests.get("http://wikibook.co.kr/logo.png")

# Binary 형식으로 데이터 저장하기
with open("test.png", "wb") as f:
    f.write(r.content)
print("saved")
```