

```
1 Lab. request 사용하기
2
3 1. requests module
4 1)Python HTTP for Humans'
5 2)Python에서 웹 데이터를 받아올 때 가장 많이 사용하는 module이다.
6 3)http://docs.python-requests.org/en/master/
7 4)Install
8     $ pip install requests
9
10
11 5)웹 페이지에 접속하기
12     -요청하면 서버가 응답한 값을 반환한다.
13     -바로 응답 코드를 확인할 수 있다.
14     -또는 status__code 속성을 통해서도 응답 코드를 확인할 수 있다.
15
16     import requests
17     url = 'https://www.naver.com'
18     naver = requests.get(url)
19     print(naver)
20     -----
21     <Response [200]>
22
23     import requests
24     url = 'https://www.naver.com'
25     naver = requests.get(url)
26     print(naver)
27     print(naver.status_code)
28     -----
29     <Response [200]>
30     200
31
32
33 6)요청 페이지를 찾을 수 없을 때 404코드를 반환한다.
34
35     import requests
36
37     def url_check(url):
38         res = requests.get(url)
39
40         print(res)
41
42         sc = res.status_code
43
44         if sc == 200:
45             print("%s 요청성공"%(url))
46         elif sc == 404:
47             print("%s 찾을 수 없음" %(url))
48         else:
49             print("%s 알수 없는 에러 : %s"%(url, sc))
50
51
52     url_check("https://www.naver.com")
53     url_check("https://www.naver.com//a")
54     -----
```

```
55 <Response [200]>
56 https://www.naver.com 요청성공
57 <Response [404]>
58 https://www.naver.com//a 찾을 수 없음
59
60
61 7)header 가져오기
62
63 import requests
64
65 url = "https://www.naver.com"
66
67 res = requests.get(url)
68
69 print(res)
70 print(res.headers)
71 -----
72 <Response [200]>
73 {'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'}
74
75
76 8)응답 객체에서 header를 dict 형태로 가져온다.
77
78 import requests
79
80 url = "https://www.naver.com"
81 res = requests.get(url)
82 print(res)
83
84 headers = res.headers
85 print(headers['Set-Cookie'])
86 -----
87 <Response [200]>
88 PM_CK_loc=e28d849e90b7a5d7c3ee6e69e31d9ba3a3ac7923a7b342c194bb9591a177e4a7;
Expires=Wed, 28 Aug 2019 11:30:28 GMT; Path=/; HttpOnly
89
90
91 import requests
92
93 url = "https://www.naver.com"
94 res = requests.get(url)
95 print(res)
96
97 headers = res.headers
98
99 for header in headers:
```

```
100     print(headers[header])
101     -----
102     <Response [200]>
103     NWS
104     Tue, 27 Aug 2019 11:31:58 GMT
105     text/html; charset=UTF-8
106     chunked
107     keep-alive
108     PM_CK_loc=e28d849e90b7a5d7c3ee6e69e31d9ba3a3ac7923a7b342c194bb9591a177e4a7;
109     Expires=Wed, 28 Aug 2019 11:31:58 GMT; Path=/; HttpOnly
110     no-cache, no-store, must-revalidate
111     no-cache
112     CP="CAO DSP CURa ADMa TAIA PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM
113     STA PRE"
114     DENY
115     1; mode=block
116     gzip
117     max-age=63072000; includeSubdomains
118     unsafe-url
119
120 9)HTML code 보기
121
122     import requests
123     url = 'https://www.naver.com'
124     naver = requests.get(url)
125     print(naver.text)
126
127 10)HTML code 보기2
128     -content 속성을 사용하면 한글을 binary 형태(인코딩)로 바꿔서 가져온다.
129     -binary 형태로 HTML을 가져올 경우 text속성을 이용하였을 때 발생하는 한글 문자 깨지는 현상을 방지할 수 있다.
130
131     import requests
132     url = 'https://www.naver.com'
133     naver = requests.get(url)
134     print(naver.content)
135
136
137 11)인코딩 확인
138
139     import requests
140     url = 'https://www.naver.com'
141     naver = requests.get(url)
142     print(naver.encoding)
143     -----
144     UTF-8
145
146
147 12)JSON 형식 처리
148     -Response 객체의 json()을 통해 JSON 형식의 응답을 간단하게 decoding 해서 dict 또는 list 추출 가능
149
150     r = requests.get('http://weather.livedoor.com/forecast/webservice/json/v1?city=130010')
151     r.json()
```

```
152 -----
153 {'pinpointLocations': [{'link': 'http://weather.livedoor.com/area/forecast/1310100', 'name':
154   '千代田区'}, {'link': 'http://weather.livedoor.com/area/forecast/1310200'....
155
156 13)Data 보내기
157   -requests로 요청할 때 data를 실어 보낼 수 있다.
158   -querystring 같은 경우 URL에 직접 표현할 수 있지만, querystring을 만들어야 하는 번거로움이 있다.
159   -하지만 data를 dict 형태로 만들어 보내는 방식으로 번거로움을 줄일 수 있다.
160   -data 뿐만 아니라 header, cookie 같은 data도 원하는 값으로 변경하여 요청 가능하다.
161   -특정 page는 header의 user-agent가 비었거나, cookie가 비어있을 경우 정상적으로 HTML 처리에 문제가 있을
162     수 있다.
163   -이럴 때는 header나 cookie를 직접 만들어야 한다.
164   -querystring data를 만들어 요청하기
165
166   import requests
167   url = "https://pjt3591oo.github.io/"
168   res = requests.get(url, params={"key1": "value1", "key2": "value2"})
169   print(res.url)
170   -----
171   https://pjt3591oo.github.io/?key1=value1&key2=value2
172
173 14)get()함수는 HTTP method의 GET에 대응된다.
174   -post(), put(), delete(), head(), options()는 각각 POST, PUT, DELETE, HEAD, OPTIONS에 대응된다.
175
176 15)Session 객체
177   -여러 개의 page를 연속으로 crawling할 때는 효과적이다.
178   -HTTP header 또는 Basic 인증 등의 설정을 한 번만 하고 여러번 재사용 가능하다.
179
180 2. pprint module 이용하기
181   1)대량의 data를 보기 쉽게 표시해주는 표준 module
182   2)pprint(prettyprint)
183
184   import requests
185   import pprint
186   url = 'https://www.naver.com'
187   naver = requests.get(url)
188   pprint.pprint(naver.text)
189
190
191
192 3. requests를 사용하여 API에 접근하기
193   1)기상청 RSS(http://www.weather.go.kr/weather/lifenindustry/sevice\_rss.jsp)를 이용하자.
194   2)2013년 동네 예보 RSS
195     http://www.weather.go.kr/images/weather/lifenindustry/dongnaeforecast\_rss.pdf
196   3)주소선택후 rss button click하면 zone을 알 수 있다.
197     -예:서울특별시 강남구 청담동
198     http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1168056500
199
200   import requests
201   import pprint
202   api_uri = 'http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1168056500'
203   weather_data = requests.get(api_uri).text
```

```

204     pprint.pprint(weather_data)
205     -----
206     ('<?xml version="1.0" encoding="UTF-8" ?>\n'
207      '<rss version="2.0">\n'
208      '<channel>\n'
209      '<title>기상청 동네예보 웹서비스 - 서울특별시 강남구 청담동 도표예보</title>\n'
210      '<link>http://www.kma.go.kr/weather/main.jsp</link>\n'
211      '<description>동네예보 웹서비스</description>\n'
212      '<language>ko</language>\n'
213      '<generator>동네예보</generator>\n'
214      '<pubDate>2019년 01월 15일 (화)요일 14:00</pubDate>\n'
215      '<item>\n'
216      '<author>기상청</author>\n'
217      '<category>서울특별시 강남구 청담동</category>\n'
218      '<title>동네예보(도표) : 서울특별시 강남구 청담동 '
219      '[X=61,Y=126]</title><link>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&dongCode=1168056500</link>\n'
220      '<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&dongCode=1168056500</guid>\n'
221      '<description>\n'
222      ...
223      ...
224
225
226 4)get method의 params option을 활용하기
227
228     api_url = 'http://www.kma.go.kr/wid/queryDFSRSS.jsp'
229
230     payload = {'zone':'1168056500'}
231
232     weather_data = requests.get(api_url, payload).text
233
234     weather_data
235     -----
236     '<?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&dongCode=1168056500</link>\n<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&dongCode=1168056500</guid>\n<description>\n<header>\n<tm>201901151400</tm>\n<ts>4</ts>\n'
237     ...
238     ...
239
240
241 5)xml.etree.ElementTree module 사용하기
242     -Python에서 xml data를 다루기 위한 module
243
244     import xml.etree.ElementTree as ET
245     import pandas as pd

```

```

246
247     xml_data = ET.fromstring(weather_data)
248
249     for tag in xml_data.iter('data'):
250         print(tag.find('hour').text + "/" + tag.find('temp').text)
251     -----
252     18/-2.0
253     21/-4.0
254     24/-5.0
255     ...
256     ...
257
258     list = []
259     for tag in xml_data.iter('data'):
260         dic = {'hour': tag.find('hour').text,
261               'day' : tag.find('day').text,
262               'temp' : tag.find('temp').text,
263               'tmx' : tag.find('tmx').text,
264               'tmn' : tag.find('tmn').text,
265               'sky' : tag.find('sky').text,
266               'pty' : tag.find('pty').text,
267               'wfKor' : tag.find('wfKor').text,
268               'wfEn' : tag.find('wfEn').text}
269     list.append(dic)
270
271     df = pd.DataFrame(list, columns=['hour', 'day', 'temp', 'tmx', 'tmn', 'sky', 'pty', 'wfKor',
272                                     'wfEn'])
273
274     df
275     -----
276     hour  day  temp    tmx    tmn  sky  pty  wfKor  wfEn
277     0  18   0   -2.0   -999.0 -999.0  2    0   구름 조금  Partly Cloudy
278     1  21   0   -4.0   -999.0 -999.0  2    0   구름 조금  Partly Cloudy
279     2  24   0   -5.0   -999.0 -999.0  1    0   맑음      Clear
280     3   3   1   -6.0    -1.0  -8.0  1    0   맑음      Clear
281     4   6   1   -7.0    -1.0  -8.0  1    0   맑음      Clear
282
283
284 4. Lab : RSS Scraping
285 1)전자신문 RSS
286     import pandas as pd
287     import xml.etree.ElementTree as ET
288     import requests
289
290     api_url = 'http://rss.etnews.com/Section901.xml'
291
292     etnews_data = requests.get(api_url).text
293     etnews_data
294     -----
295     '<?xml version="1.0" encoding="utf-8" ?>\n<rss version="2.0">\r\n    <channel>\r\n    ...
296     ...
297
298     xml_data = ET.fromstring(etnews_data)

```

```
299     for tag in xml_data.iter('item'):
300         print(tag.find('title').text + "," + tag.find('pubDate').text)
301         -----
302     애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍,Tue, 15 Jan 2019 17:00:00 +0900
303     지난해 드론 자격증 취득자 전년비 4배 증가...실효성 지적도,Tue, 15 Jan 2019 17:00:00 +0900
304     화웨이, 韓 스마트워치 시장 진출,Tue, 15 Jan 2019 17:00:00 +0900
305     ...
306     ...
307
308     list = []
309
310     for tag in xml_data.iter('item'):
311         dic = {'Title':tag.find('title').text,
312               'Link':tag.find('link').text,
313               'Author':tag.find('author').text,
314               'PubDate':tag.find('pubDate').text,
315               'Guid':tag.find('guid').text}
316         list.append(dic)
317
318     list
319     -----
320     [{'Title': "애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍",
321       'Link': 'http://www.etnews.com/20190115000273',
322       'Author': '윤건일',
323       'PubDate': 'Tue, 15 Jan 2019 17:00:00 +0900',
324       'Guid': '20190115000273'},
325       ...
326       ...
327
328     df = pd.DataFrame(list, columns=['Title', 'Link', 'Author', 'PubDate', 'Guid'])
329     df.head()
330     -----
331     ...
332     ...
333
334     df.info()
335     -----
336     <class 'pandas.core.frame.DataFrame'>
337     RangeIndex: 30 entries, 0 to 29
338     Data columns (total 5 columns):
339     Title      30 non-null object
340     Link       30 non-null object
341     Author     30 non-null object
342     PubDate    30 non-null object
343     Guid       30 non-null object
344     dtypes: object(5)
345     memory usage: 1.2+ KB
346
347
348
349     5. Login이 필요한 site에서 download하기
350     1)한빛출판네트웍 로그인 page
351     http://www.hanbit.co.kr/member/login.html
352
```

```

353 2)마이한빛
354 http://www.hanbit.co.kr/myhanbit/myhanbit.html
355
356 3)로그인 폼
357 <form name="frm" id="frm" action="#" method="post">
358 <input name="retun_url" id="retun_url" type="hidden"
value="http://www.hanbit.co.kr/myhanbit/myhanbit.html" class="i_text" size="100">
359 <div class="login_left">
360 <fieldset>
361 <legend>한빛출판네트워크 로그인</legend>
362
363 <label class="i_label" for="login_id"><strong style="position: absolute; visibility:
visible;"></strong>
364 <input name="m_id" id="m_id" type="text" value="" class="i_text" placeholder="아이
디" onkeydown="javascript:if(event.keyCode==13){login_proc(); return false;}">
365 </label>
366
367 <label class="i_label" for="login_pw"><strong style="position: absolute;"></strong>
368 <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;}">
369 </label>
370
371 <label>
372 <input type="button" name="login_btn" id="login_btn" value="로그인"
class="btn_login">
373 </label>
374
375 <label class="i_label2">
376 <input type="checkbox" name="keepid" id="keepid" value="1"
class="i_check"><strong>아이디 저장</strong>
377 </label>
378 </fieldset>
379
380 <ul class="login_btn">
381 <li><a href="/member/find_id.html" class="btn_idc">아이디 찾기</a></li>
382 <li><a href="/member/find_pw.html" class="btn_pwc">비밀번호 찾기</a></li>
383 <li><a href="/member/member_agree.html" class="btn_joinc">회원가입</a></li>
384 </ul>
385 </div>
386 </form>
387
388
389 4)m_id, m_passwd라는 값(name 속성의 값)을 입력하고, 입력 양식을 제출하면 즉 submit하면 login되는 구조이
다.
390
391 5)Login 과정 분석
392 -Chrome의 Network tab
393 -상단의 filter중에서 'Doc'를 클릭한다.
394 -그 위의 [Preserve log] check
395 --원래 [Network] tab은 page가 이동할 때 기존 page와 관련된 내용을 지우고, 새로운 page의 내용만 띄운다.
396 --하지만 이것을 check하면 내용을 지우지 않고 유지해준다.
397 --login 과정을 분석하려면 web page를 어떻게 이동하는지 알아야하므로 반드시 체크한다.
398 -로그인을 수행한다.

```



399 -그러면, login.html -> login\_pro.php -> myhanbit.html의 과정으로 보인다.  
400 -하나하나의 과정을 클릭하면 자세한 내용을 볼 수 있다.  
401 -login\_proc.php를 클릭해보자.  
402 -Request Method가 POST임을 알 수 있다.  
403 -그 아래로 계속 내려가면 Form Data 섹션의 m\_id와 m\_passwd의 값을 확인할 수 있다.  
404 -다시 말해, login\_proc.php페이지에 입력 양식 data를 POST로 전달하면 로그인한다는 것을 알 수 있다.  
405

#### 406 6)Python으로 login하기

```
407
408 import requests
409 from bs4 import BeautifulSoup
410 from urllib.parse import urljoin
411
412 # 아이디와 비밀번호 지정하기
413 USER = "devexpert"
414 PASS = "P@$$W0rd"
415
416 # 세션 시작하기
417 session = requests.session()
418
419 # 로그인하기
420 login_info = {
421     "m_id": USER, # 아이디 지정
422     "m_passwd": PASS # 비밀번호 지정
423 }
424
425 url_login = "http://www.hanbit.co.kr/member/login_proc.php"
426 res = session.post(url_login, data=login_info)
427 res.raise_for_status() # 오류가 발생하면 예외가 발생.
428
429 # 마이페이지에 접근하기
430 url_mypage = "http://www.hanbit.co.kr/myhanbit/myhanbit.html"
431 res = session.get(url_mypage)
432 res.raise_for_status()
433
434 # 마일리지와 이코인 가져오기
435 soup = BeautifulSoup(res.text, "html.parser")
436 mileage = soup.select_one(".mileage_section1 span").get_text()
437 ecoin = soup.select_one(".mileage_section2 span").get_text()
438 print("마일리지: " + mileage)
439 print("이코인: " + ecoin)
440 -----
441 마일리지: 0
442 이코인: 0
443
444
```

#### 445 6. Web page image 추출하기

```
446
447 import requests
448 r = requests.get("http://wikibook.co.kr/logo.png")
449
450 # Binary 형식으로 데이터 저장하기
451 with open("test.png", "wb") as f:
452     f.write(r.content)
```

```
453     print("saved")
```