<준비>
1. Menu 중 Terminal > New Terminal > Python Version 확인
2. Ctrl + Shift + P --> 명령창에서 exe 엔터
3. 확장에서 Python, Python for VSCode, Python Extension Pack, Python(PyDev) 설치할 것
4. Google에서 'selenium python'으로 검색
5. https://www.seleniumhq.org/download/에서 Browser 중 Google Chrome Driver 설치할 것

1. Web Scraping이란?
  1)Web 문서로부터 필요한 정보만을 추출하여 제공하는 기술을 말한다.
  2)이 기술을 통해 상품 Catalog를 제작하거나 News 기사, Blog나 Cafe의 게시물, 회사의 profile과 금융 data 등을 수집할 수 있다.
  3)그러기 위해서는 먼저 추출하고자 하는 정보들이 구성되어 있는 영역을 확인해야 한다.

2. Internet에서 data 수집하기
  1)Data 수집은 data 준비 절차의 첫 단계로 어떤 형식의 data를 수집할지, 어떤 방법과 경로로 data를 수집할 것인가를 고민해야 한다.
  2)수집된 data의 품질은 data 분석 결과에 많은 영향을 미치며, data가 얼마나 잘 정제되어 있는지에 따라서도 전체 data 분석에 드는 시간과 노력이 좌우되기 때문에 data 수집 과정은 매우 중요한 단계라 할 수 있다.

3. Web site에서 사용된 web 기술 확인
  1)해당 web site에 사용된 web 기술을 확인하는 유용한 도구는 builtwith module이다.
  2)Install
    $ pip install builtwith
  3)이 module은 전달된 URL을 가진 web site를 download하고 분석할 것이다.
  4)분석이 되면 web site에 사용된 기술을 알려준다.

```
import builtwith
builtwith.parse('http://example.wegbscraping.com')
------------------------------------------------------------
{'web-servers': ['Nginx'],
 'web-frameworks': ['Web2py', 'Twitter Bootstrap'],
 'programming-languages': ['Python'],
 'javascript-frameworks': ['jQuery', 'Modernizr', 'jQuery UI']}
```

4. Web site 소유자 찾기
  1)website의 소유자를 찾으려면 해당 website의 domain명에 누가 등록됐는지 확인하는 whois protocol을 사용할 수 있다.
  2)Install
    $ pip install python-whois

```
import whois
print(whois.whois('appspot.com')
------------------------------------------------------------
{'domain_name': ['APPSPOT.COM', 'appspot.com'],
 'registrar': 'MarkMonitor, Inc.',
 'whois_server': 'whois.markmonitor.com',
 'referral_url': None,
 'updated_date': [datetime.datetime(2019, 2, 6, 10, 33, 49),
  datetime.datetime(2019, 2, 6, 2, 33, 49)],
 'creation_date': [datetime.datetime(2005, 3, 10, 2, 27, 55),
```

```
 47         datetime.datetime(2005, 3, 9, 18, 27, 55)],
 48        'expiration_date': [datetime.datetime(2020, 3, 10, 1, 27, 55),
 49         datetime.datetime(2020, 3, 9, 0, 0)],
 50        'name_servers': ['NS1.GOOGLE.COM',
 51         'NS2.GOOGLE.COM',
 52         'NS3.GOOGLE.COM',
 53         'NS4.GOOGLE.COM',
 54         'ns1.google.com',
 55         'ns4.google.com',
 56         'ns3.google.com',
 57         'ns2.google.com'],
 58        'status': ['clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited',
 59         'clientTransferProhibited https://icann.org/epp#clientTransferProhibited',
 60         'clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited',
 61         'serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited',
 62         'serverTransferProhibited https://icann.org/epp#serverTransferProhibited',
 63         'serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited',
 64         'clientUpdateProhibited (https://www.icann.org/epp#clientUpdateProhibited)',
 65         'clientTransferProhibited (https://www.icann.org/epp#clientTransferProhibited)',
 66         'clientDeleteProhibited (https://www.icann.org/epp#clientDeleteProhibited)',
 67         'serverUpdateProhibited (https://www.icann.org/epp#serverUpdateProhibited)',
 68         'serverTransferProhibited
            (https://www.icann.org/epp#serverTransferProhibited)',
 69         'serverDeleteProhibited (https://www.icann.org/epp#serverDeleteProhibited)'],
 70        'emails': ['abusecomplaints@markmonitor.com',
           'whoisrequest@markmonitor.com'],
 71        'dnssec': 'unsigned',
 72        'name': None,
 73        'org': 'Google LLC',
 74        'address': None,
 75        'city': None,
 76        'state': 'CA',
 77        'zipcode': None,
 78        'country': 'US'}
 79
 80  5. Website crawling
 81    1)Web site를 scrap을 하려면 먼저 crawling 이라고 알려진 과정, 즉 관심 있는 data를 가진 web
       page를 download할 필요가 있다.
 82    2)Web site crawling을 할 수 있는 몇 가지 방법이 있으며, 대상 web site 구조에 맞춰 적절히 선택
       한다.
 83    3)Web site download
 84      -Web page를 crawling하려면 우선 web page를 download 해야 한다.
 85      -다음은 web 주소가 URL인 web site를 download하고자 Python의 urllib3 module을 사용하
         는 간단한 Python script이다.
 86
 87        from urllib.request import urlopen
 88
 89        def download(url):
 90          return urlopen(url).read().decode('utf-8')
 91
 92      -이 함수는 URL이 전달됐을 때 web page를 download하고 HTML을 반환한다.
 93      -이 source code는 web page를 download할 때 처리가 안되는 오류에 직면할 수도 있는 문제
```

가 있다.
-이를 테면 요청한 page가 존재하지 않는 것이 그런 사례이다.
-그럴 경우 urllib3는 예외를 발생한 후 script의 실행을 멈춘다.
-그럴 경우를 대비해서 아래의 code로 변경하자.

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

def download(url):
  print('downloading:', url)
  try :
    html = urlopen(url).read().decode('utf-8')
  except HTTPError as e:
    print('Download error:', e.reason)
    html = None
  return html
```

-이제 download 오류 상황이 되면 예외 처리가 되고 None을 반환한다.
-이 기능을 test하려면 500 오류를 반환하는 http://httpstat.us/500 page를 download하면 된다.

```
download('http://httpstat.us/500')
-------------------------------------------
Downloading: http://httpstat.us/500
Download error: Internal Server Error

download('http://www.samsung.com')
-----------------------------------------------
Downloading: http://www.samsung.com
Download error: Forbidden
```

4)Download 재시도
-Download할 때 가끔 발생되는 오류가 일시적인 것들도 있다.
-예를 들어, Web server가 과부하되면 503 Service Unavailable 오류가 발생한다.
-이 오류는 server 상의 문제가 해결되면 더 이상 나타나지 않고 web page를 다시 download할 수 있다.
-그러나 모든 오류에 대해 download를 다시 시도하지 않으려 한다.
-Server에서 404 Not Found 오류가 발생하면 web page가 현재 존재하지 않기 때문에 계속 같은 오류가 발생할 것이다.
-일어날 가능성이 있는 HTTP 오류 전체 list는 IETF(Internet Engineering Task Force)에서 정의했고, http://tools.ietf.org/html/rfc7231#section-6에서 확인할 수 있다.
-이 문서에서 4xx 오류들은 요청할 때 잘못된 것이 있을 때 일어나며, 5xx 오류들은 server에 문제가 있을 때 일어난다는 것을 알 수 있다.
-따라서 5xx 오류가 일어날 때만 download를 재시도하게 download 함수를 만들자.

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

  def download(url, num_retries=2):
    print('downloading:', url)
```

```
            try :
               html = urlopen(url).read().decode('utf-8')
            except HTTPError as e:
               print('Download error:', e.reason)
               html = None
               if num_retries > 0:
                  if hasattr(e, 'code') and 500 <= e.code < 600:
                     #5xx HTTP 오류시 재 시도
                     return download(url, num_retries - 1)
            return html
```

-이제 5xx 오류가 발생하면 download 함수는 계속 자신을 호출하면서 재시도한다.
-또한 download 함수는 재시도할 수 있는 횟수를 알려주는 전달 인자도 갖고 있다.
-이 전달 인자는 기본 2회로 되어 있다.
-Server 오류가 해결되지 않을 수도 있기 때문에 web page를 download하는 시도 횟수를 제한한다.
-이 기능을 test하려면 500 오류를 반환하는 http://httpstat.us/500 page를 download하면 된다.

```
   download('http://httpstat.us/500')
   downloading: http://httpstat.us/500
   Download error: Internal Server Error
   downloading: http://httpstat.us/500
   Download error: Internal Server Error
   downloading: http://httpstat.us/500
   Download error: Internal Server Error
```

-예상대로 download 함수는 web page를 download하려 한다.
-그리고 오류가 발생하면 함수를 끝내기 전에 2번 더 download를 시도한다.

6. Crawling & Scraping

1)Crawling
-Web Site의 data를 그대로 취득하는 것
2)Scraping
-Crawling하여 모든 data에서 필요한 것만 추출하거나 변환하는 처리를 포함.
3)정규식
-정규식은 변경 사항에 대해 유연하게 대처할 수는 있지만 만들기 어렵고 가독성도 떨어진다.
-따라서 적용하기에 너무 취약하고 web page가 바뀌면 사용하는 정규식이 쉽게 무용지물이 된다는 점의 문제점이 있다.

4)pandas.read_html() 함수 사용하기
-다음의 third-party package들의 설치여부를 확인한다.
--html5lib, lxml, BeautifulSoup4
-HTML file의 table 요소를 DataFrame으로 불러온다.
-DataFrame이 들어 있던 list가 return된다.
-Table 요소가 여러 개 있는 경우, 여러 개의 DataFrame이 저장된다.

```
   import pandas as pd

   url = 'http://www.fdic.gov/bank/individual/failed/banklist.html'

   dfs = pd.read_html(url)
```

```
189
190         dfs[0].info()
191         ------------------------------------------------------------------------------
            -
192         <class 'pandas.core.frame.DataFrame'>
193         RangeIndex: 555 entries, 0 to 554
194         Data columns (total 7 columns):
195         Bank Name               555 non-null object
196         City                    555 non-null object
197         ST                      555 non-null object
198         CERT                    555 non-null int64
199         Acquiring Institution   555 non-null object
200         Closing Date            555 non-null object
201         Updated Date            555 non-null object
202         dtypes: int64(1), object(6)
203         memory usage: 30.4+ KB
204
205     5)requests module 이용하기
206       -Python HTTP for Humans'
207       -http://docs.python-requests.org/en/master/
208       -Install
209         pip install requests
210
211         import requests
212         url = 'https://www.naver.com'
213         naver = requests.get(url)
214         print(naver.text)
215
216       -requests module을 사용하여 Web API로 Wikipedia의 정보를 얻는 방법을 소개한다.
217       -이 Wikipedia의 data를 program으로 획득하기 위해 MediaWiki라는 service를 사용할 것이
          다.
218         --https://www.mediawiki.org/wiki/MediaWiki
219
220         import requests, pprint
221         api_url = 'https://en.wikipedia.org/w/api.php'
222         api_params = {'format':'json', 'action':'query', 'titles':'Jack
            Bauer','prop':'revisions', 'rvprop':'content'}
223         #title : 검색하고 싶은 keyword
224         #prop : 검색 결과로 어떤 정보를 반환할지를 지정
225         #rvprop : prop으로 지정한 항목을 더 구체적으로 지정할 수있다.
226
227         wiki_data = requests.get(api_url, params=api_params).json()
228         pprint.pprint(wiki_data)
229         ------------------------------------------------------
230         {'batchcomplete': '',
231           'query': {'pages': {'389903': {'ns': 0,
232                               'pageid': 389903,
233                               'revisions': [{'*': '{{Use mdy '
234                                                   'dates|date=December '
235                                                   '2017}}\n'
236                                                   '{{other people}}\n'
237                                                   '{{Infobox character\n'
```

```
                                        '| color           = '
                                        '#B3001B\n'
                                        '| name            = '
                                        'Jack Bauer\n'
                                        '| series          = '
                                        '[[24 (TV series)|24]]\n'
                                        '| image           = '
                                        'Jack Bauer.jpg\n'
                                        '| caption         = '
    ...
    ...
```

-action에 지정할 수 있는 것
https://en.wikipedia.org/w/api.php

-action으로 query를 지정했을 때 prop에 지정할 수 있는 것
https://en.wikipedia.org/w/api.php?action=help&modules=query

-prop로 revisions를 지정했을 때 rvprop에 지정할 수 있는 것
https://en.wikipedia.org/w/api.php?action=help&modules=query%2Brevisions

-다양한 조건을 지정할 수 있지만, wikipedia 검색 결과를 표시하고 싶다면 위의 예제에서 지정한 설정을 그대로 사용하고, title parameter만 바꾸면 된다.

```
import requests
import codecs
api_base_url = 'https://en.wikipedia.org/w/api.php'
api_params = {'format':'xmlfm', 'action':'query', 'titles':'Jack
Bauer','prop':'revisions', 'rvprop':'content'}
wiki_data = requests.get(api_base_url, params=api_params)
fo = codecs.open('C:/temp/wiki.html', 'w', 'utf-8')
fo.write(wiki_data.text)
fo.close()
```

6)pprint module 이용하기
-대량의 data를 보기 쉽게 표시해주는 표준 module
-pprint(prettyprint)

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

7)BeautifulSoup module 이용하기
-Scraping을 위한 module
-일반적으로 HTML tag들이 start tag와 end tag가 서로 pair 되지 않을 경우가 많다.
-pair 되지 않아도 아름답게 처리해 주는 module
-https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
from bs4 import BeautifulSoup
import requests
```

```
288
289         html_data = requests.get('https://www.naver.com')
290         soup = BeautifulSoup(html_data.text, 'html.parser')
291         soup.title
292         ---------------------------------
293         <title>NAVER</title>
294
295       [한빛미디어 책 제목 읽어오기]
296         hanbit = requests.get('http://www.hanbit.co.kr/media/')
297         soup = BeautifulSoup(hanbit.text, 'html.parser')
298         soup
299         ------------------------------------------------------------
300         <!DOCTYPE html>
301         <html lang="ko">
302         <head>
303         <!--[if lte IE 8]>
304         <script>
305           location.replace('/support/explorer_upgrade.html');
306         </script>
307         <![endif]-->
308         <!-- Google Tag Manager -->
309         <script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
310         new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
311         j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
312         'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefo
            re(j,f);
313         })(window,document,'script','dataLayer','GTM-W9D5PM3');</script>
314         <!-- End Google Tag Manager -->
315         ...
316         ...
317
318         for book in soup.find_all('p', class_='book_tit'):
319           print(book.find('a').text)
320         ------------------------------------------------------------
321         리얼월드 HTTP : 역사와 코드로 배우는 인터넷과 웹 기술
322         이것이 MariaDB다
323         제프리 리처의 Windows via C/C++(복간판)
324         초보자를 위한 유니티 입문(개정판) : 따라 하면서 배우는 2D & 3D 게임 개발
325         회사에서 바로 통하는 실무 엑셀
326         맛있는 디자인 포토샵 CC 2019
327         알고리즘이 욕망하는 것들
328         파이썬 라이브러리를 활용한 머신러닝(번역개정판) : 사이킷런 핵심 개발자가 쓴 머신러닝과 데이
            터 과학 실무서
329         파이썬으로 웹 크롤러 만들기(2판) : 초간단 나만의 웹 크롤러로 원하는 데이터를 가져오는 방법
330         더 나은 세상을 위한 소프트 디지털
331         비됴클래스 하줜의 유튜브 동영상 편집 with 프리미어 프로
332         회사에서 바로 통하는 실무 엑셀+파워포인트+워드&한글
333         맛있는 디자인 포토샵&일러스트레이터 CC 2019
334         맛있는 디자인 프리미어 프로&애프터 이펙트 CC 2019
335         밑바닥부터 시작하는 딥러닝
336         이것이 C#이다
337         핸즈온 머신러닝
```

```
338         소문난 명강의 : 레트로의 유니티 게임 프로그래밍 에센스
339         이것이 자바다
340         이것이 우분투 리눅스다
341
342       [Naver 영화 평점 Scraping 하기]
343
344         from bs4 import BeautifulSoup
345
346         html_data =
            requests.get('https://movie.naver.com/movie/point/af/list.nhn?page=1')
347         soup = BeautifulSoup(html_data.text, 'html.parser')
348         titles = soup.find_all(class_='movie')
349
350         title_list = []
351         for title in titles:
352           print(title.text)
353         ----------------------------------------
354         대학살의 신
355         스타워즈: 라스트 제다이
356         내안의 그놈
357         말모이
358         말모이
359         내안의 그놈
360         언니
361         내안의 그놈
362         존 윅
363         마이 리틀 자이언트
364
365         for title in titles:
366           title_list.append(title.text)
367
368         point_list = []
369         points = soup.find_all(class_='point')
370         for point in points:
371           point_list.append(point.text)
372
373         review_list = []
374         reviews = soup.find_all(class_='title')
375         for review in reviews:
376             rev = review.text
377             rev = rev.strip()
378             rev = rev.replace('\t', '')
379             rev = rev.replace('\n', '')
380             rev = rev.replace('신고', '')
381             review_list.append(rev)
382
383         df = pd.DataFrame(title_list, columns=['Title'])
384         df['Point'] = point_list
385         df['Review'] = review_list
386         df
387         ------------------------------------------------
388           Title                    Point     Review
```

```
0 대학살의 신 7 대학살의 신자식싸움에 부모등 터진다
1 스타워즈: 라스트 제다이 1 스타워즈: 라스트 제다이이거 보느니 로그원 열번 보는게 낫다
2 내안의 그놈 10 내안의 그놈재밌어요 유치해도 빵빵터짐 진영 연기 잘하네요 라미란과 잘...
3 말모이 10 말모이후반부에 보고 울었습니다 감동적임
4 말모이 10 말모이재미를 떠나서 역사는 무조건 10점이다
5 내안의 그놈 10 내안의 그놈뻔한스토리이지만 재밌게 잘 보고왔어요최고의성형은 다이어트
6 언니 5 언니론다 로우지가 언니 역활했으면 그나마 공감이 됐을듯...개연성도 떨어지고 액션도...
7 내안의 그놈 9 내안의 그놈ㅋㅋ재밌었음 생각보다 안정적인 연기력 소소하게 웃기 좋은 영화
8 존 윅 10 존 윅액션의 선두주자 키아누리브스!!
9 마이 리틀 자이언트 7 마이 리틀 자이언트동화보다 더 환상적인 거인
```

[Naver 평점 1page부터 100page까지 scraping 하기]

```
from bs4 import BeautifulSoup

url = 'https://movie.naver.com/movie/point/af/list.nhn?page='

title_list = []
point_list = []
review_list = []

for pge in range(1, 101):
  url = url + str(pge)
  print(url)
  html_data = requests.get(url)
  soup = BeautifulSoup(html_data.text, 'html.parser')
  titles = soup.find_all(class_='movie')
  points = soup.find_all(class_='point')
  reviews = soup.find_all(class_='title')
  for title in titles:
    title_list.append(title.text)
  for point in points:
    point_list.append(point.text)
  for review in reviews:
    rev = review.text
    rev = rev.strip()
    rev = rev.replace('\t', '')
    rev = rev.replace('\n', '')
    rev = rev.replace('신고', '')
    review_list.append(rev)
  url = url.split('=')[0] + '='

df = pd.DataFrame(title_list, columns=['Title'])
df['Point'] = point_list
df['Review'] = review_list

df.info()
```

```
436         ------------------------------------------------
437         <class 'pandas.core.frame.DataFrame'>
438         RangeIndex: 1020 entries, 0 to 1019
439         Data columns (total 3 columns):
440         Title     1020 non-null object
441         Point     1020 non-null object
442         Review    1020 non-null object
443         dtypes: object(3)
444         memory usage: 24.0+ KB
445
446       [Coupang의 상품정보 Scraping]
447         -Web 문서들은 서로 다양한 문서 구조로 출력된다.
448         -따라서 R에서 web 문서로부터 scraping을 하기 위해서는 추출하고자 하는 정보들이 구성되어
            있는 영역을 먼저 확인해야 한다.
449         -Social Commerce의 대표적인 online market인 Coupang의 상품 정보 추출을 해보자.
450         -'여성패션' 중 '여성 크로스백' 목록 item을 살펴보자.
451         -Scraping하려는 web page의 URL 구조와 문서 구조를 파악한다.
452         -URL 구조
453           http://www.coupang.com/np/search?q=여성크로스백
454         -문서 구조
455           --상품명 : class="name"
456           --가격 : class="price-value"
457
458       [한국일보 headline 기사 Scraping하기]
459         -한국일보 첫 page의 기사를 Scraping 해보자.
460         -먼저 scraping 하려는 web page의 URL 구조와 문서 구조를 파악해야 한다.
461         -URL 구조
462           http://www.hankookilbo.com/
463         -문서 구조
464           --기사 제목 : class="firstList"
465
466
467     8)lxml module 이용하기
468       -이 module은 C로 만들어진 libxml2 XML 분석 library를 Python에서 사용할 수 있게 만든
          module이다.
469       -Beautiful Soup보다 더 빠르게 분석하고 scraping할 수 있지만, 어떤 computer에서는 설치가
          좀 어려운 편이다.
470       -Installa
471         $ pip install lxml
472       -필요하다면 cssselect 도 install
473         $ pip install cssselect
474
475         import lxml.html
476
477         url = 'http://www.hanbit.co.kr/media/'
478         html = download(url)
479         tree = lxml.html.fromstring(html)
480         for a in tree.cssselect('a'):
481             #href 속성과 글자를 추출한다.
482             print(a.get('href'), a.text)
483
484     9)requests를 사용하여 API에 접근하기
```

-기상청 RSS(http://www.weather.go.kr/weather/lifenindustry/sevice_rss.jsp)를 이용하자.
-2013년 동네 예보 RSS
http://www.weather.go.kr/images/weather/lifenindustry/dongnaeforecast_rss.pdf
-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;dongCode=1168056500</link>\n'
 '<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;dongCode=1168056500</guid>\n'
 '<description>\n'
 ...
 ...
```

-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;dongCode=1168056500</link>\n<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;dongCode=1168056500</guid>\n<description>\n <header>\n
        <tm>201901151400</tm>\n  <ts>4</ts>\n
        ...
        ...

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

    import xml.etree.ElementTree as ET

    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 18  0    -2.0    -999.0  -999.0  2    0    구름 조금 Partly Cloudy
    1 21  0    -4.0    -999.0  -999.0  2    0    구름 조금 Partly Cloudy
    2 24  0    -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

11)RSS Scraping
```

```
572     -전자신문 RSS
573       import pandas as pd
574       import xml.etree.ElementTree as ET
575       import requests
576
577       api_url = 'http://rss.etnews.com/Section901.xml'
578
579       etnews_data = requests.get(api_url).text
580       etnews_data
581       ------------------------------------------------
582       '<?xml version="1.0" encoding="utf-8" ?>\n<rss version="2.0">\r\n
          <channel>\r\n   ...
583       ...
584
585       xml_data = ET.fromstring(etnews_data)
586       for tag in xml_data.iter('item'):
587         print(tag.find('title').text + "," + tag.find('pubDate').text)
588       ------------------------------------------------------------------
589       애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍,Tue, 15 Jan 2019 17:00:00
          +0900
590       지난해 드론 자격증 취득자 전년비 4배 증가…실효성 지적도,Tue, 15 Jan 2019 17:00:00
          +0900
591       화웨이, 韓 스마트워치 시장 진출,Tue, 15 Jan 2019 17:00:00 +0900
592       ...
593       ...
594
595       list = []
596
597       for tag in xml_data.iter('item'):
598           dic = {'Title':tag.find('title').text,
599                  'Link':tag.find('link').text,
600                  'Author':tag.find('author').text,
601                  'PubDate':tag.find('pubDate').text,
602                  'Guid':tag.find('guid').text}
603           list.append(dic)
604
605       list
606       ------------------------------------
607       [{'Title': "애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍",
608           'Link': 'http://www.etnews.com/20190115000273',
609           'Author': '윤건일',
610           'PubDate': 'Tue, 15 Jan 2019 17:00:00 +0900',
611           'Guid': '20190115000273'},
612           ...
613           ...
614
615       df = pd.DataFrame(list, columns=['Title', 'Link', 'Author', 'PubDate', 'Guid'])
616       df.head()
617       ------------------------------------------------
618       ...
619       ...
620
```

```
    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
```

7. Web site의 data file 읽기
   -Web site에 있는 data set를 R로 loading 하는 방법을 알아보자.
   -다음은 Titanic data file을 CSV file로 저장하는 방법을 소개한다.
   -아래의 site에 접속해 보자.
   -https://github.com/vincentarelbundock/Rdatasets/tree/master/csv/datasets
   -https://vincentarelbundock.github.io/Rdatasets/datasets.html
     --Item(datasets에서 찾는다) 중 'Titanic'을 찾아보자.
     --Link의 속성을 파악하기 위해서는 Google Chrome보다는 Internet Explorer를 이용하는 것이 좋다.
     --Internet Explorer로 해당 site를 방문한 다음, datasets의 Item 중 CSV의 link를 Mouse 오른 Click를 한 후, [속성]을 선택한다.
     --속성 창에서 Data set file의 URL을 확인하자.

```
    import pandas as pd
    url = 'https://vincentarelbundock.github.io/Rdatasets/csv/datasets/Titanic.csv'
    df = pd.read_csv(url)
    df.head()
```

8. Selenium
   1)run.py

```
    # 인터파크 투어 사이트에서 여행지를 입력후 검색 -> 잠시후 -> 결과
    # 로그인시 PC 웹 사이트에서 처리가 어려울 경우  -> 모바일 로그인 진입
    # 모듈 가져오기
    # pip install selenium
    # pip install bs4
    # pip install pymysql
    from selenium import webdriver as wd
    from bs4 import BeautifulSoup as bs
    from selenium.webdriver.common.by import By
    # 명시적 대기를 위해
    from selenium.webdriver.support.ui import WebDriverWait
    from selenium.webdriver.support import expected_conditions as EC
    from DbMgr import DBHelper as Db
    import time
    from Tour import TourInfo

    # 사전에 필요한 정보를 로드 => 디비혹스 쉘, 베치 파일에서 인자로 받아서 세팅
    db       = Db()
    main_url = 'http://tour.interpark.com/'
```

```
671     keyword  = '로마'
672     # 상품 정보를 담는 리스트 (TourInfo 리스트)
673     tour_list = []
674
675     # 드라이버 로드
676     # 맥용
677     # driver = wd.Chrome(executable_path='./chromedriver')
678     # 윈도우용
679     driver = wd.Chrome(executable_path='chromedriver.exe')
680     # 고스트용
681     # driver   = wd.PhantomJS(executable_path='./phantomjs')
682     # 차후 -> 옵션 부여하여 (프록시, 에이전트 조작, 이미지를 배제)
683     # 크롤링을 오래돌리면 => 임시파일들이 쌓인다!! -> 템프 파일 삭제
684
685     # 사이트 접속 (get)
686     driver.get(main_url)
687     # 검색창을 찾아서 검색어 입력
688     # id : SearchGNBText
689     driver.find_element_by_id('SearchGNBText').send_keys(keyword)
690     # 수정할경우 => 뒤에 내용이 붙어버림 => .clear() -> send_keys('내용')
691     # 검색 버튼 클릭
692     driver.find_element_by_css_selector('button.search-btn').click()
693
694     # 잠시 대기 => 페이가 로드되고 나서 즉각적으로 데이터를 획득 하는 행위는
695     # 명시적 대기 => 특정 요소가 로케이트(발결된때까지) 대기
696     try:
697         element = WebDriverWait(driver, 10).until(
698             # 지정한 한개 요소가 올라면 웨이트 종료
699             EC.presence_of_element_located( (By.CLASS_NAME, 'oTravelBox') )
700         )
701     except Exception as e:
702         print( '오류 발생', e)
703     # 암묵적 대기 => DOM이 다 로드 될때까지 대기 하고 먼저 로드되면 바로 진행
704     # 요소를 찾을 특정 시간 동안 DOM 풀링을 지시 예를 들어 10 초이내 라로
705     # 발견 되면 진행
706     driver.implicitly_wait( 10 )
707     # 절대기 대기 => time.sleep(10) -> 클라우드 페어(디도스 방어  솔류션)
708     # 더보기 눌러서 => 게시판 진입
709     driver.find_element_by_css_selector('.oTravelBox>.boxList>.moreBtnWrap>.mor
        eBtn').click()
710
711     # 게시판에서 데이터를 가져올때
712     # 데이터가 많으면 세션(혹시 로그인을 해서 접근되는 사이트일 경우) 관리
713     # 특정 단위별로 로그아웃 로그인 계속 시도
714     # 특정 게시물리 사라질 경우 => 팝업 발생 (없는 ...) => 팝업 처리 검토
715     # 게시판 스캔시 => 임계점을 모름!!
716     # 게시판 스캔 => 메타 정보 획득 => loop 를 돌려서 일괄적으로 방문 접근 처리
717
718     # searchModule.SetCategoryList(1, '') 스크립트 실행
719     # 16은 임시값, 게시물을 넘어갔을때 현상을 확인차
720     for page in range(1, 2):#16):
721         try:
```

```
            # 자바스크립트 구동하기
            driver.execute_script("searchModule.SetCategoryList(%s, '')" % page)
            time.sleep(2)
            print("%s 페이지 이동" % page)
            ##########################################################
            # 여러 사이트에서 정보를 수집할 경우 공통 정보 정의 단계 필요
            # 상품명, 코멘트, 기간1, 기간2, 가격, 평점, 썸네일, 링크(상품상세정보)
            boxItems = driver.find_elements_by_css_selector('.oTravelBox>.boxList>li')
            # 상품 하나 하나 접근
            for li in boxItems:
                # 이미지를 링크값을 사용할것인가?
                # 직접 다운로드 해서 우리 서버에 업로드(ftp) 할것인가?
                print( '썸네임', li.find_element_by_css_selector('img').get_attribute('src') )
                print( '링크', li.find_element_by_css_selector('a').get_attribute('onclick') )
                print( '상품명', li.find_element_by_css_selector('h5.proTit').text )
                print( '코멘트', li.find_element_by_css_selector('.proSub').text )
                print( '가격',   li.find_element_by_css_selector('.proPrice').text )
                area = ''
                for info in li.find_elements_by_css_selector('.info-row .proInfo'):
                    print(  info.text )
                print('='*100)
                # 데이터 모음
                # li.find_elements_by_css_selector('.info-row .proInfo')[1].text
                # 데이터가 부족하거나 없을수도 있으므로 직접 인덱스로 표현은 위험성이 있음
                obj = TourInfo(
                    li.find_element_by_css_selector('h5.proTit').text,
                    li.find_element_by_css_selector('.proPrice').text,
                    li.find_elements_by_css_selector('.info-row .proInfo')[1].text,
                    li.find_element_by_css_selector('a').get_attribute('onclick'),
                    li.find_element_by_css_selector('img').get_attribute('src')
                )
                tour_list.append( obj )
        except Exception as e1:
            print( '오류', e1 )

    print( tour_list, len(tour_list) )
    # 수집한 정보 개수를 루프 => 페이지 방문 => 콘텐츠 획득(상품상세정보) => 디비
    for tour in tour_list:
        # tour => TourInfo
        print( type(tour) )
        # 링크 데이터에서 실데이터 획득
        # 분해
        arr = tour.link.split(',')
        if arr:
            # 대체
            link = arr[0].replace('searchModule.OnClickDetail(','')
            # 슬라이싱 => 앞에 ', 뒤에 ' 제거
            detail_url = link[1:-1]
            # 상세 페이지 이동 : URL 값이 완성된 형태인지 확인 (http~)
            driver.get( detail_url )
            time.sleep(2)
```

```
            # pip install bs4
            # 혖재 페이지를 beautifulsoup 의 DOM으로 구성
            soup = bs( driver.page_source, 'html.parser')
            # 현제 상세 정보 페이지에서 스케줄 정보 획득
            data = soup.select('.tip-cover')
            #print( type(data), len(data), type(data[0].contents)  )
            # 디비 입력 => pip install pymysql
            # 데이터 sum
            content_final = ''
            for c in data[0].contents:
                content_final += str(c)

            # html 콘첸츠 데이터 전처리 (디비에 입력 가능토록)
            import re
            content_final   = re.sub("'", '"', content_final)
            content_final   = re.sub(re.compile(r'\r\n|\r|\n|\n\r+'), '', content_final)

            print( content_final )
            # 콘텐츠 내용에 따라 전처리 => data[0].contents
            db.db_insertCrawlingData(
                tour.title,
                tour.price[:-1],
                tour.area.replace('출발 가능 기간 : ',''),
                content_final,
                keyword
            )

        # 종료
        driver.close()
        driver.quit()
        import sys
        sys.exit()

    2)Tour.py
        # 상품 정보를 담는 클레스
        class TourInfo:
            # 맴버변수 (실제 컬럼보다는 작게 세팅했음)
            title = ''
            price = ''
            area  = ''
            link  = ''
            img   = ''
            contents = ''
            # 생성자
            def __init__(self, title, price, area, link, img, contents=None ):
                self.title = title
                self.price = price
                self.area  = area
                self.link  = link
                self.img   = img
                self.contents = contents

```

3)DbMgr.py

```
# 디비 처리, 연결, 해제, 검색어 가져오기, 데이터 삽입
import pymysql as my

class DBHelper:
    '''
    맴버변수 : 커넥션
    '''
    conn = None
    '''
    생성자
    '''
    def __init__(self):
        self.db_init()
    '''
    맴버 함수
    '''
    def db_init(self):
        self.conn = my.connect(
                    host='localhost',
                    user='root',
                    password='1234',
                    db='pythonDB',
                    charset='utf8',
                    cursorclass=my.cursors.DictCursor )

    def db_free(self):
        if self.conn:
            self.conn.close()

    # 검색 키워드 가져오기 => 웹에서 검색
    def db_selectKeyword(self):
        # 커서 오픈
        # with => 닫기를 처리를 자동으로 처리해준다 => I/O 많이 사용
        rows = None
        with self.conn.cursor() as cursor:
            sql  = "select * from tbl_keyword;"
            cursor.execute(sql)
            rows = cursor.fetchall()
            print(rows)
        return rows

    def db_insertCrawlingData(self, title, price, area, contents, keyword ):
        with self.conn.cursor() as cursor:
            sql = '''
            insert into `tbl_crawlingdata`
            (title, price, area, contents, keyword)
            values( %s,%s,%s,%s,%s )
            '''
            cursor.execute(sql, (title, price, area, contents, keyword) )
        self.conn.commit()

```

```
877     # 단독으로 수행시에만 작동 => 테스트코드를 삽입해서 사용
878     if __name__=='__main__':
879         db = DBHelper()
880         print( db.db_selectKeyword() )
881         print( db.db_insertCrawlingData('1','2','3','4','5') )
882         db.db_free()
```