파이썬을 이용한 웹 데이터 수집(크롤링)

1. 문서 내용 요청 후 읽어오기 urllib 패키지
    - urlopen() 접속
    - read() 데이터 읽어오기
    - text 속성을 사용해서 데이터 읽어올 수도 있음
2. 문서에서 원하는 내용 추출하기(파싱)

BeautifulSoup 패키지 사용 - find()/findAll()

urllib 패키지 - url을 넘겨주면 데이터를 텍스트 형태로 반환(기본 내장 패키지)

- urllib2, urllib3 다른 버전 사용
- requests 패키지를 사용할 수도 있음

In [1]:
```
import requests
# 파이썬 기본 패키지 : HTTP 요청을 보내는 모듈
```

In [2]:
```
url = 'http://www.tistory.com'

# request 패키지의 get(url) 함수 사용 : urllib의 open()과 read() 를 한번에 진행
response = requests.get(url)
# URL 변수에 저장되어 있는 웹주소로 요청신호를 보냄
# 서버(tistory)는 해당 페이지의 소스코드를 클라이언트로 전송하면서 응답하게 됨
# response 변수에는 http://www.tistory.com의 소스코드(html)가 저장되어 있음
```

In [3]:
```
# 정상 응답인지 확인
response.status_code # 응답에대한 상태코드
# 200은 정상응답
# 400번대 코드 : 클라이언트의 요청이 잘못되었다의 의미(url주소가 틀렸거나 권한이 없는 페이지를 요청했거나...)
# 500번대 코드 : 클라이언트는 문법에 맞게 요청을 했는데 서버측에서 인증이 안되었거나, 서버가 망가졌거나의 상태
```

Out[3]:
```
200
```

In [4]:
```
# 요청에 의해 전달된 코드 확인
# 응답객체.text 속성을 통해 확인
# 출력 내용이 너무 길기 때문에 확인을 위해서 일부만 출력
response.text[5036:6983]
# 브라우저가 사이트에 요청해서 받은 결과 코드랑 동일한 코드가 문자열 형태로 반환됨
```

Out[4]:
```
'<title>티스토리</title><meta name="description" content="좀 아는 블로거들의 유용한 이야기, 티스토리. 블로그, 포트폴리오, 웹사이트까지 티스토리에서 나를 표현해 보세요."/><meta property="og:title" content="티스토리"/><meta property="og:description" content="좀 아는 블로거들의 유용한 이야기, 티스토리. 블로그, 포트폴리오, 웹사이트까지 티스토리에서 나를 표현해 보세요."/><meta property="og:image" content="https://t1.daumcdn.net/tistory_admin/static/images/openGraph/tistoryOpengraph.png"/><meta property="og:image:alt" content="좀 아는 블로거들의 유용한 이야기, 티스토리. 블로그, 포트폴리오, 웹사이트까지 티스토리에서 나를 표현해 보세요."/><meta name="twitter:card" content="summary_large_image"/><meta name="twitter:title" content="티스토리"/><meta name="twitter:description" content="좀 아는 블로거들의 유용한 이야기, 티스토리. 블로그, 포트폴리오, 웹사이트까지 티스토리에서 나를 표현해 보세요."/><meta name="twitter:image" content="https://t1.daumcdn.net/tistory_admin/static/images/openGraph/tistoryOpengraph.png"/><meta name="twitter:image:alt" content="좀 아는 블로거들의 유용한 이야기, 티스토리. 블로그, 포트폴리오, 웹사이트까지 티스토리에서 나를 표현해 보세요."/><link rel="shortcut icon" href="https://t1.daumcdn.net/tistory_admin/favicon/tistory_favicon_32x32.ico"/><link rel="icon" href="https://t1.daumcdn.net/tistory_admin/top_v2/bi-tistory-favicon.svg"/><link rel="apple-touch-icon" href="https://t1.daumcdn.net/tistory_admin/top_v2/tistory-apple-touch-favicon.png"/><script>(self.__next_s=self.__next_s||[]).push(["//t1.daumcdn.net/tiara/js/v1/tiara-1.2.0.min.js",{}])</script><script src="/_next/static/chunks/polyfills-42372ed130431b0a.js" noModule=""></script></head><body><div id="kakaoIndex"><a href="#kakaoBody">본문 바로가기</a><a href="#kakaoLnb">메뉴 바로가기</a></div><div id="kakaoWrap"><div class="kakao_head head_type1" id="kakaoHead" role="banner"><div class="inner_header tistory"><div class="group_head"><h1><a class="link_logo" id="kakaoServiceLogo" href="/"><span class="img_common_tistory tit_tistory tit_tistory_black">티스토리</span></a></h1><div class="gnb_tistory" id="kakaoGnb" role="navigation"><h2 class="screen_out">서비스 주요 메뉴</h2><ul class="list_gnb"><li class="on">'
```

In [5]:
```
r = requests.get("https://www.google.co.kr") # 사이트 접속요청 후 응답
r # <Response [200]> : 응답객체이고 정상응답받았다
r.text
print(len(r.text))
# 14278

# 반환된 소스의 일부분 추출
r.text[0:100]
```

```
18806
```

Out[5]:
```
'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ko"><head><meta content'
```

In [6]:
```
url = 'http://www.naver.com'
res = requests.get(url)
res.status_code # 200
```

```
res.text
len(res.text)
```

Out[6]: 223171

파라미터 전달 방법

- 파라미터란? : 사이트의 문서를 요청할 때 서버로 전달되는 정보
- 함수의 파라미터 처럼 문서를 찾기위한 정보나 명령을 수행하기 위한 정보를 같이 전달하게 되는데 그 정보를 파라미터라고 함
- 서버에 파라미터 전송방법
  - 파라미터 전송 방법은 url에 ? 뒤에 파라미터=값&파라미터2=값 으로 전송
    - 사이트에 따라서 잘못된 접속으로 인지하고 에러처리할 수 있음
  - 파라미터를 dict로 구성해서 get(params=dict)
- https://sports.news.naver.com/news?oid=477&aid=0000312064
- params = {'param1': 'value1', 'param2': 'value2'}
- res = requests.get(URL, params=params)

In [7]:
```
# 기본 url
base_url = "https://sports.news.naver.com/news"
# 파라미터가 포함된 url
param_url ="https://sports.news.naver.com/news?oid=079&aid=0004089508"
```

In [8]:
```
# 파라미터 전달 방식 1 : get방식 - url에 파라미터 포함
# 사이트에 따라 접근 거부할 수도 있음
res1 = requests.get(param_url)
res1.status_code
```

Out[8]: 200

In [9]:
```
# 파라미터 전달 방식 2 : get 방식 : params= dict사용
# oid=477&aid=0000312064
param = {'oid':'079', 'aid':'0004089508'}
```

```
res2 = requests.get(base_url, params=param)
res2.status_code
```

Out[9]: 200

urllib 패키지 사용한 소스 추출

In [10]:
```
from urllib.request import urlopen # 사이트에 요청 신호를 보내는 함수
```

In [11]:
```
url ='http://www.naver.com'
html = urlopen(url)
html
#<http.client.HTTPResponse at 0x2a60cc87c70> : 상태코드와 소스코드가 있습니다.
```

Out[11]: <http.client.HTTPResponse at 0x20a6ad662f0>

In [12]:
```
# 응답객체에서 소스코드 읽어오기
# 응답객체.read() - 한번 읽어오면 그 후에는 읽어도 빈 문자열이 나옴
rawtext = html.read()
```

In [13]:
```
type(rawtext)
```

Out[13]: bytes

In [14]:
```
print(type(rawtext),"\n")
print(bytes(rawtext)[:200], "\n")
print(rawtext.decode('utf8')[:200])
```

```
<class 'bytes'>

b'   <!doctype html> <html lang="ko" class="fzoom"> <head> <meta charset="utf-8"> <meta name="Referrer" content="origin"> <meta
http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" cont'

  <!doctype html> <html lang="ko" class="fzoom"> <head> <meta charset="utf-8"> <meta name="Referrer" content="origin"> <meta h
ttp-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" cont
```

문서에서 원하는 내용 추출하기(파싱)

- html 문서에서 원하는 내용 추출

- 필요한 내용만 추출
- BeautifulSoup 라이브러리 사용
  - 태그 형식으로 된 text를 파싱할 때 사용
- find() / findAll() 등 함수 사용

BeautifulSoup

- import bs4
- 데이터를 추출하는데 필요한 기능이 들어 있는 라이브러리 (파싱 라이브러리)
- 외부 라이브러리 : 설치해야 함
- 주피터는 기본 패키지임(설치하지 않아도 됨)
- 파이참에서 설치방법
  - File/Settings
  - Project Interpreter에서 bs4 검색
  - [Install Package]

In [15]:
```
import bs4
```

In [16]:
```
url = 'http://www.naver.com'
html = urlopen(url)
html
```

Out[16]:
```
<http.client.HTTPResponse at 0x20a6addb0a0>
```

In [17]:
```
# 응답객체인 html을 BeautifulSoup(응답객체, 파서기) 함수에 전달
# bs4  파싱객체를 반환
bs_obj = bs4.BeautifulSoup(html,'html.parser')
```

In [18]:
```
type(bs_obj) # bs4.BeautifulSoup
# 출력 내용이 너무 길기 때문에 확인을 위해서 일부만 출력
print(bs_obj.prettify()[:2468])
# html 소스코드를 들여쓰기 하여 계층적인 구조로 표현 (가독성이 높다)
```

```
<!DOCTYPE html>
<html class="fzoom" lang="ko">
 <head>
  <meta charset="utf-8"/>
  <meta content="origin" name="Referrer"/>
  <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
  <meta content="width=1190" name="viewport"/>
  <title>
   NAVER
  </title>
  <meta content="NAVER" name="apple-mobile-web-app-title">
   <meta content="index,nofollow" name="robots">
    <meta content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요" name="description">
     <meta content="네이버" property="og:title"/>
     <meta content="https://www.naver.com/" property="og:url"/>
     <meta content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png" property="og:image"/>
     <meta content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요" property="og:description">
      <meta content="summary" name="twitter:card"/>
      <meta content="" name="twitter:title"/>
      <meta content="https://www.naver.com/" name="twitter:url"/>
      <meta content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png" name="twitter:image"/>
      <meta content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요" name="twitter:description">
       <meta content="uru5NJKa1Bfr5nv5AdQ26Qat7UrPU_02l-PIZRLzI-g" name="google-site-verification">
        <link href="/favicon.ico?1" rel="shortcut icon" type="image/x-icon"/>
        <link href="https://s.pstatic.net/static/www/nFavicon96.png" rel="apple-touch-icon-precomposed">
         <link href="https://s.pstatic.net/static/www/u/2014/0328/mma_204243574.png" rel="apple-touch-icon" sizes="114x114"/>
         <link href="https://s.pstatic.net/static/www/u/2014/0328/mma_20432863.png" rel="apple-touch-icon"/>
         <link href="https://ssl.pstatic.net/sstatic/search/pc/css/sp_autocomplete_251001.css" rel="stylesheet"/>
         <link href="https://www.naver.com/" rel="canonical"/>
         <link href="https://m.naver.com/" media="only screen and (max-width: 767px)" rel="alternate"/>
         <script>
          window.gladsdk=window.gladsdk||{},window.gladsdk.cmd=window.gladsdk.cmd||[],window.ndpsdk=window.ndpsdk||{},window.nd
psdk.cmd=window.ndpsdk.cmd||[],window.ndpsdk.polyfill=window.ndpsdk.polyfill||{cmd:[]};var g_ssc="navertop.v5";window.nsc=g_ss
c,window.nmain=window.nmain||{},window.nmain.jsOrigin="www"
         </script>
         <script>
          window.ntm=window.ntm||[]
         </script>
         <script async="" src="https://ssl.pstatic.net/tveta/libs/ndpsdk/prod/ndp-loader.js">
         </script>
```

BeautifulSoup 패키지의 파싱 함수

- find(태그,[{속성명:속성값}])
  - 지정한 태그 중 첫번째 만나는 태그만 추출 또는 지정한 태그 중 해당 속성과 속상값을 갖고있는 태그의 첫번째 태그
- findAll(태그,[{속성명:속성값}])
  - 지정한 태그 모두 찾아서 추출
  - 첫번째 이외의 태그를 추출할 때 사용
  - list 형태로 반환
- find_all(태그,[{속성명:속성값}])
  - findAll 함수와 동일

In [19]:
```
# test html
html_str = "<html><div>hello</div></html>"

# html코드를 파싱 객체로 변환
bs_obj = bs4.BeautifulSoup(html_str, "html.parser")
print(type(bs_obj))
#<class 'bs4.BeautifulSoup'> bs4 객체 - 관련함수 사용 가능
print(bs_obj)
# html 태그로 추출
```

```
<class 'bs4.BeautifulSoup'>
<html><div>hello</div></html>
```

In [20]:
```
bs_obj.find('div')
# <div>hello</div> : 여는 div 태그부터 닫는 div 태그까지
bs_obj.find('div').text
# html코드에서 첫번째 만나는 div태그의 내부 문자열 반환
```

Out[20]:
```
'hello'
```

In [21]:
```
html_str = """
<html>
    <body>
        <ul>
            <li>hello</li>
            <li>bye</li>
```

```
            <li>welcome</li>
        </ul>
    </body>
</html>
"""
```

In [22]:
```
# bs4 객체 생성
bs_obj = bs4.BeautifulSoup(html_str,'html.parser')
```

In [23]:
```
ul_t = bs_obj.find('ul')
ul_t
```

Out[23]:
```
<ul>
<li>hello</li>
<li>bye</li>
<li>welcome</li>
</ul>
```

In [24]:
```
ul_t
ul_t.find('li')
# ul_t 객체 안에서 첫번째 만나는 li 태그를 반환
```

Out[24]:
```
<li>hello</li>
```

In [25]:
```
uls =ul_t.findAll('li') # ul_t 객체 안의 모든  li태그를 반환
uls # list형태로 반환
```

Out[25]:
```
[<li>hello</li>, <li>bye</li>, <li>welcome</li>]
```

In [26]:
```
type(uls[2]) # tag 객체
```

Out[26]:
```
bs4.element.Tag
```

In [27]:
```
uls[2].text
```

Out[27]:
```
'welcome'
```

In [28]:
```
# 반복문을 이용해서 모든 원소의 text 추출
for li in uls :
```

```
    print(li.text)
```

```
hello
bye
welcome
```

In [29]:
```
ul_t.text
# ul_t 태그 내부에 있는 모든 text를 문자열로 묶어서 반환
```

Out[29]:
```
'\nhello\nbye\nwelcome\n'
```

In [30]:
```
ul_t.text.split('\n')
```

Out[30]:
```
['', 'hello', 'bye', 'welcome', '']
```

In [31]:
```
html_str = """
<html>
    <body>
        <ul class="greet">
            <li>hello</li>
            <li>bye</li>
            <li>welcome</li>
        </ul>
        <ul class="reply">
            <li>ok</li>
            <li>no</li>
            <li>sure</li>
        </ul>
    </body>
</html>
"""
```

In [32]:
```
bs_obj = bs4.BeautifulSoup(html_str,'html.parser')
```

In [33]:
```
# 태그중에 특정 속성을 갖고 있는 태그를 추출
# ul 태그중 class 속성값이 greet인 태그를 추출
# 첫번째 ul 태그를 추출
bs_obj.find('ul')
```

Out[33]:
```
<ul class="greet">
<li>hello</li>
<li>bye</li>
<li>welcome</li>
</ul>
```

In [34]:
```
# ul태그를 찾고 class 속성값이 greet인걸 확인 후에 반환
bs_obj.find('ul',{'class':'greet'})
```

Out[34]:
```
<ul class="greet">
<li>hello</li>
<li>bye</li>
<li>welcome</li>
</ul>
```

In [35]:
```
# ul 태그중 class 속성값이 reply인 태그를 추출
bs_obj.findAll('ul')[1]
bs_obj.find('ul',{'class':'reply'})
# ul 태그중에 class 속성값이 reply인 첫번째 ul태그 반환
bs_obj.findAll('ul',{'class':'reply'}) # list로 반환
# ul 태그중에 class 속성값이 reply인 모든 ul태그 반환
# 해당 ul태그가 여러개면 모두 반환
```

Out[35]:
```
[<ul class="reply">
 <li>ok</li>
 <li>no</li>
 <li>sure</li>
 </ul>]
```

In [36]:
```
# bs_obj 객체의 모든 li태그 추출
bs_obj.find_all('li') # list로 반환
bs_obj.findAll('li') # 6개의 li 태그 객체
```

Out[36]:
```
[<li>hello</li>,
 <li>bye</li>,
 <li>welcome</li>,
 <li>ok</li>,
 <li>no</li>,
 <li>sure</li>]
```

In [37]:
```
bs_obj.find_all('ul') # 2개의 ul 객체
```

Out[37]:
```
[<ul class="greet">
 <li>hello</li>
 <li>bye</li>
 <li>welcome</li>
 </ul>,
 <ul class="reply">
 <li>ok</li>
 <li>no</li>
 <li>sure</li>
 </ul>]
```

In [38]:
```
#  class속성값이 greet 인 ul 태그 내의 모든 li 태그 추출
bs_obj.find('ul').find_all('li')
bs_obj.find_all('ul')[0].find_all('li')
bs_obj.find('ul',{'class':'greet'}).find_all('li')
bs_obj.find_all('ul',{'class':'greet'})[0].find_all('li')
```

Out[38]:
```
[<li>hello</li>, <li>bye</li>, <li>welcome</li>]
```

In [39]:
```
html_str = """
<html>
    <body>
        <h1 id='title'>Hello Python</h1>
        <p id="crawling">웹 크롤링</p>
        <p id="parsing">파싱</p>
    </body>
</html>"""
```

In [40]:
```
bs_obj = bs4.BeautifulSoup(html_str, 'html.parser')
```

In [41]:
```
# h1 태그 중 id가 title인 태그 추출
bs_obj.find('h1',{'id':'title'})
```

Out[41]:
```
<h1 id="title">Hello Python</h1>
```

In [42]:
```
# p태그 중 id가 parsing 인 태그
bs_obj.find('p',{'id':'parsing'})
```

Out[42]:
```
<p id="parsing">파싱</p>
```

bs4 형제 노드 찾기

In [43]:
```
html_str = """
<html>
    <body>
        <h1>파이썬 프로그래밍</h1>
        <p>웹 페이지 분석</p><p>크롤링</p><p>파싱</p>
    </body>
</html>
"""
```

In [44]:
```
bs_obj = bs4.BeautifulSoup(html_str, 'html.parser')
```

In [45]:
```
bs_obj.find('p') # html문서에서 첫번째 나타나는 p 태그
```

Out[45]:
```
<p>웹 페이지 분석</p>
```

In [46]:
```
bs_obj.find_all('p')# html문서에서 나타나는 모든 p 태그
```

Out[46]:
```
[<p>웹 페이지 분석</p>, <p>크롤링</p>, <p>파싱</p>]
```

In [47]:
```
p1 = bs_obj.find('p')
p1.next_sibling # 첫번째 p태그를 기준으로 다음 p태그를 반환
```

Out[47]:
```
<p>크롤링</p>
```

In [48]:
```
p1.next_sibling.next_sibling # 첫번째 p태그를 기준으로 다음 다음 p태그를 반환
```

Out[48]:
```
<p>파싱</p>
```

속성값 추출하기

a 태그의 href 속성

In [49]:
```
html_str = """
<html>
    <body>
        <ul class="ko">
            <li><a href="https://www.naver.com/">네이버</a></li>
            <li><a href="https://www.daum.net/">다음</a></li>
        </ul>
        <ul class="sns">
            <li><a href="https://www.goole.com/">구글</a></li>
            <li><a href="https://www.facebook.net/">페이스북</a></li>
        </ul>
    </body>
</html>
"""
```

In [50]:
```
bs_obj = bs4.BeautifulSoup(html_str,'html.parser')
```

In [51]:
```
bs_obj.find_all('li')
bs_obj.find_all('a')
```

Out[51]:
```
[<a href="https://www.naver.com/">네이버</a>,
 <a href="https://www.daum.net/">다음</a>,
 <a href="https://www.goole.com/">구글</a>,
 <a href="https://www.facebook.net/">페이스북</a>]
```

In [52]:
```
# 태그의 속성값을 추출
# 태그객체['속성명']
bs_obj.find_all('a')[0]['href']
```

Out[52]:
```
'https://www.naver.com/'
```

In [53]:
```
bs_obj.find_all('a')[0].text
```

Out[53]:
```
'네이버'
```

In [54]:
```
hrefs=[]
for a in bs_obj.find_all('a') :
```

```
    print(a['href'])
    hrefs.append(a['href'])
```

```
https://www.naver.com/
https://www.daum.net/
https://www.goole.com/
https://www.facebook.net/
```

In [55]:
```
hrefs=[]
name = []
for a in bs_obj.find_all('a') :
    name.append(a.text)
    hrefs.append(a['href'])
```

In [56]:
```
hrefs
name

import pandas as pd
pd.DataFrame({'site명' : name,
              'siteURL' : hrefs})
```

Out[56]:

	site명	siteURL
0	네이버	https://www.naver.com/
1	다음	https://www.daum.net/
2	구글	https://www.goole.com/
3	페이스북	https://www.facebook.net/

In [57]:
```
import bs4
html_str = """
<html>
   <body>
        <div id="wrap">
                <div id="mainMenuBox">
                <ul>
                    <li><a href="#">패션잡화</a></li>
                    <li><a href="#">주방용품</a></li>
```

```
                    <li><a href="#">생활건강</a></li>
                    <li><a href="#">DIY가구</a></li>
                </ul>
            </div>
                <div>
                <table>
                        <tr><td><img src="shoes1.jpg"></td>
                          <td><img src="shoes2.jpg"></td>
                          <td><img src="shoes3.jpg"></td></tr>
                    <tr id="prdName"><td>솔로이스트<br>걸리쉬 리본단화</td>
                          <td>맥컬린<br>그레이가보시스트랩 펌프스</td>
                          <td>맥컬린<br>섹슈얼인사이드펌프스</td></tr>
                    <tr id="price"><td>100,000원</td><td>200,000원</td><td>120,000원</td></tr>
                </table>
            </div>
            <div id="out_box">
                <div class="box">
                        <h4>공지사항</h4>
                    <hr>
                    <a href="#">[배송] : 무표배송 변경 안내 18.10.20</a><br>
                    <a href="#">[전시] : DIY 가구 전시 안내 18.10.31</a><br>
                    <a href="#">[판매] : 11월 특가 상품 안내 18.11.05</a>
                </div>
                <div class="box">
                        <h4>커뮤니티</h4>
                    <hr>
                    <a href="#">[레시피] : 살 안찌는 야식 만들기</a><br>
                    <a href="#">[가구] : 헌집 새집 베스트 가구</a><br>
                    <a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a><br>
                 </div>
            </div>
        </div>
    </body>
</html>"""
```

In [58]:
```
# bs 객체 생성
bs_obj = bs4.BeautifulSoup(html_str,'html.parser')
```

In [59]:
```
# 모든 a 태그
bs_obj.findAll("a")
```

Out[59]:
```
[<a href="#">패션잡화</a>,
 <a href="#">주방용품</a>,
 <a href="#">생활건강</a>,
 <a href="#">DIY가구</a>,
 <a href="#">[배송] : 무표배송 변경 안내 18.10.20</a>,
 <a href="#">[전시] : DIY 가구 전시 안내 18.10.31</a>,
 <a href="#">[판매] : 11월 특가 상품 안내 18.11.05</a>,
 <a href="#">[레시피] : 살 안찌는 야식 만들기</a>,
 <a href="#">[가구] : 헌집 새집 베스트 가구</a>,
 <a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a>]
```

In [60]:
```
# selector : select(태그) 함수에 인수로 전달
# 해당 태그를 모두 찾아서 list로 반환
bs_obj.select('a')
```

Out[60]:
```
[<a href="#">패션잡화</a>,
 <a href="#">주방용품</a>,
 <a href="#">생활건강</a>,
 <a href="#">DIY가구</a>,
 <a href="#">[배송] : 무표배송 변경 안내 18.10.20</a>,
 <a href="#">[전시] : DIY 가구 전시 안내 18.10.31</a>,
 <a href="#">[판매] : 11월 특가 상품 안내 18.11.05</a>,
 <a href="#">[레시피] : 살 안찌는 야식 만들기</a>,
 <a href="#">[가구] : 헌집 새집 베스트 가구</a>,
 <a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a>]
```

In [61]:
```
bs_obj.select('div')
```

Out[61]:
```
[<div id="wrap">
 <div id="mainMenuBox">
 <ul>
 <li><a href="#">패션잡화</a></li>
 <li><a href="#">주방용품</a></li>
 <li><a href="#">생활건강</a></li>
 <li><a href="#">DIY가구</a></li>
 </ul>
 </div>
 <div>
 <table>
 <tr><td><img src="shoes1.jpg"/></td>
 <td><img src="shoes2.jpg"/></td>
 <td><img src="shoes3.jpg"/></td></tr>
 <tr id="prdName"><td>솔로이스트<br/>걸리쉬 리본단화</td>
 <td>맥컬린<br/>그레이가보시스트랩 펌프스</td>
 <td>맥컬린<br/>섹슈얼인사이드펌프스</td></tr>
 <tr id="price"><td>100,000원</td><td>200,000원</td><td>120,000원</td></tr>
 </table>
 </div>
 <div id="out_box">
 <div class="box">
 <h4>공지사항</h4>
 <hr/>
 <a href="#">[배송] : 무표배송 변경 안내 18.10.20</a><br/>
 <a href="#">[전시] : DIY 가구 전시 안내 18.10.31</a><br/>
 <a href="#">[판매] : 11월 특가 상품 안내 18.11.05</a>
 </div>
 <div class="box">
 <h4>커뮤니티</h4>
 <hr/>
 <a href="#">[레시피] : 살 안찌는 야식 만들기</a><br/>
 <a href="#">[가구] : 헌집 새집 베스트 가구</a><br/>
 <a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a><br/>
 </div>
 </div>
 </div>,
 <div id="mainMenuBox">
 <ul>
 <li><a href="#">패션잡화</a></li>
```

```
<li><a href="#">주방용품</a></li>
<li><a href="#">생활건강</a></li>
<li><a href="#">DIY가구</a></li>
</ul>
</div>,
<div>
<table>
<tr><td><img src="shoes1.jpg"/></td>
<td><img src="shoes2.jpg"/></td>
<td><img src="shoes3.jpg"/></td></tr>
<tr id="prdName"><td>솔로이스트<br/>걸리쉬 리본단화</td>
<td>맥컬린<br/>그레이가보시스트랩 펌프스</td>
<td>맥컬린<br/>섹슈얼인사이드펌프스</td></tr>
<tr id="price"><td>100,000원</td><td>200,000원</td><td>120,000원</td></tr>
</table>
</div>,
<div id="out_box">
<div class="box">
<h4>공지사항</h4>
<hr/>
<a href="#">[배송] : 무표배송 변경 안내 18.10.20</a><br/>
<a href="#">[전시] : DIY 가구 전시 안내 18.10.31</a><br/>
<a href="#">[판매] : 11월 특가 상품 안내 18.11.05</a>
</div>
<div class="box">
<h4>커뮤니티</h4>
<hr/>
<a href="#">[레시피] : 살 안찌는 야식 만들기</a><br/>
<a href="#">[가구] : 헌집 새집 베스트 가구</a><br/>
<a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a><br/>
</div>
</div>,
<div class="box">
<h4>공지사항</h4>
<hr/>
<a href="#">[배송] : 무표배송 변경 안내 18.10.20</a><br/>
<a href="#">[전시] : DIY 가구 전시 안내 18.10.31</a><br/>
<a href="#">[판매] : 11월 특가 상품 안내 18.11.05</a>
</div>,
<div class="box">
<h4>커뮤니티</h4>
```

```
 <hr/>
 <a href="#">[레시피] : 살 안찌는 야식 만들기</a><br/>
 <a href="#">[가구] : 헌집 새집 베스트 가구</a><br/>
 <a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a><br/>
 </div>]
```

In [62]:
```
# id 선택자 : #을 활용
bs_obj.select('div #mainMenuBox') # 리스트로 반환
# id는 유일
bs_obj.select('#mainMenuBox')
```

Out[62]:
```
[<div id="mainMenuBox">
 <ul>
 <li><a href="#">패션잡화</a></li>
 <li><a href="#">주방용품</a></li>
 <li><a href="#">생활건강</a></li>
 <li><a href="#">DIY가구</a></li>
 </ul>
 </div>]
```

In [63]:
```
bs_obj.select('#mainMenuBox ul')
bs_obj.select('#mainMenuBox > ul')
```

Out[63]:
```
[<ul>
 <li><a href="#">패션잡화</a></li>
 <li><a href="#">주방용품</a></li>
 <li><a href="#">생활건강</a></li>
 <li><a href="#">DIY가구</a></li>
 </ul>]
```

In [64]:
```
print(bs_obj.select('#mainMenuBox li')) # 자손 li를 찾음 : ul 태그 내부의 li가 반환
bs_obj.select('#mainMenuBox > li') # 빈 리스트가 나옴
# 소스코드상 나타나는 li는 자손태그임, 자식태그인 li는 없음
```

```
[<li><a href="#">패션잡화</a></li>, <li><a href="#">주방용품</a></li>, <li><a href="#">생활건강</a></li>, <li><a href="#">DIY가구</a></li>]
```

Out[64]:
```
[]
```

In [65]:
```
# id가 wrap인 태그의 모든 자식 div 태그
bs_obj.select("#wrap > div")
```

```
len(bs_obj.select("#wrap > div"))
```

Out[65]: 3

In [66]:
```
# id가 wrap인 태그의 모든 자손 div 태그
bs_obj.select("#wrap  div")
len(bs_obj.select("#wrap  div"))
```

Out[66]: 5

In [67]:
```
# 클래스 선택자(.클래스명)를 이용
# 클래스 속성값은 중복될 수 있음
bs_obj.select('.box') # 클래스 속성값이 box인 태그 모두를 추출
```

Out[67]:
```
[<div class="box">
 <h4>공지사항</h4>
 <hr/>
 <a href="#">[배송] : 무표배송 변경 안내 18.10.20</a><br/>
 <a href="#">[전시] : DIY 가구 전시 안내 18.10.31</a><br/>
 <a href="#">[판매] : 11월 특가 상품 안내 18.11.05</a>
 </div>,
 <div class="box">
 <h4>커뮤니티</h4>
 <hr/>
 <a href="#">[레시피] : 살 안찌는 야식 만들기</a><br/>
 <a href="#">[가구] : 헌집 새집 베스트 가구</a><br/>
 <a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a><br/>
 </div>]
```

In [68]:
```
len(bs_obj.select('.box'))
# 클래스 속성값이 box인 태그 중 두번째 태그안의 모든 a태그
bs_obj.select('.box')[1].select('a')
```

Out[68]:
```
[<a href="#">[레시피] : 살 안찌는 야식 만들기</a>,
 <a href="#">[가구] : 헌집 새집 베스트 가구</a>,
 <a href="#">[후기] : 배송이 잘못 됐어요 ㅠㅠ</a>]
```

In [69]:
```
# 클래스 속성값이 box인 태그 중 두번째 태그안의 모든 a태그중 첫번째 태그
bs_obj.select('.box')[1].select('a')[0]
```

Out[69]:
```
<a href="#">[레시피] : 살 안찌는 야식 만들기</a>
```

In [70]:
```
bs_obj.select('.box')[0].select('a')[0].text
```

Out[70]:
```
'[배송] : 무표배송 변경 안내 18.10.20'
```

In [71]:
```
bs_obj.select('.box')[0].select('a')[0]['href']
```

Out[71]:
```
'#'
```

웹 크롤링 시 주의 사항

- 웹 사이트는 언제든지 변경 될 수 있기 때문에 지금 실행하는 코드가 실행되지 않을 수 있다