웹 데이터 수집 기본 다지기

학습 목표

- 웹 데이터 수집을 위한 기본 내용을 이해해 본다.

urlopen함수와 BeautifulSoup함수의 이해

학습내용

- urlopen() 함수 기본
- BeautifulSoup() 함수 기본
- lxml등을 이용한 html 정보를 구조화 시키기
- 정보 가져오는 여러가지 방법

01 urlopen 함수 이해

- urllib.request - url 처리와 관련된 모듈 패키지.
- [패키지].[모듈].[함수()]의 형식으로 실행.
- urlopen함수는 URL을 여는 함수이다.
- 국내증시 html 정보 가져오기
- url : https://finance.naver.com/sise/ (https://finance.naver.com/sise/)

In [1]:

```
from urllib.request import urlopen
```

In [2]:

```
## 정보 가져오기
url = "https://finance.naver.com/sise/"
page = urlopen(url)
page
```

Out[2]:

```
<http.client.HTTPResponse at 0x1a54b2b72e0>
```

list() 함수를 활용한 획득 정보 일부를 확인

In [3]:

```
listpage = list(page)
listpage[0:20]
```

Out[3]:

```
[b"<script language='javascript'>₩n",
 b'₩n',
 b'function main_tab(tab_title, pst, tab_cnt)₩n',
 b'{₩n',
 b'₩tfor(var i=0 ; i<tab_cnt ; i++)₩n',
 b'₩t{₩n',
 b'₩t₩tif (i == pst)₩n',
 b"₩t₩t₩tdocument.getElementById(tab_title+'_title_tab_'+i).style.display = '';₩n",
 b'₩t₩telse₩n',
 b"₩t₩t₩tdocument.getElementById(tab_title+'_title_tab_'+i).style.display = 'non
e';₩n",
 b'₩n',
 b'₩n',
 b'₩t₩tif (i == pst)₩n',
 b"₩t₩t₩tdocument.getElementById(tab_title+'_tab_'+i).style.display = '';₩n",
 b'₩t₩telse₩n',
 b"₩t₩t₩tdocument.getElementById(tab_title+'_tab_'+i).style.display = 'none';₩n",
 b'₩n',
 b'₩t}₩n',
 b'}₩n',
 b'₩n']
```

02 BeautifulSoup 함수를 활용한 html 정보 구조화

- BeautifulSoup 는 파이썬 라이브러리입니다.
- HTML과 XML파일로부터 데이터를 쉽게 가져오기 위한 파이썬 라이브러리이다.
- bs4의 모듈 안에 있음.

In [4]:

```
from bs4 import BeautifulSoup
```

In [5]:

```
page = '''
<html>
<title>나의 홈페이지</title>
<body>
<div>
<a href="https://www.naver.com/">naver</a>
<a href="https://www.google.com">google</a>
<img height="300" src="dog_01.jpg" width="500"/>
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
<p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>
<p class="p3"> 강아지 사진과 네이버 링크 </p>
<p id="p4"> 간단한 나의 홈페이지를 만들다.</p>
<p class="p3"> 강아지 사진과 네이버 링크222 </p>
</div>
</body>
</html>
'''
```

2-1 html을 파일로부터 읽을 때의 경우

```
page = open("mypage.html", 'r', encoding="utf-8").read()
page
```

- HTML/XHTML을 텍스트 파일을 구문분석하기 몇가지 파서를 사용합니다.
  - HTML Parse란 : HTML 문법 규칙을 따른 문자열을 다른 문법을 기준으로 단어의 의미나 구조를 분석하는 것을 말함. 이를 수행하는 프로그램을 HTML Parser라 한다.

- HTML 파서의 종류
  - lxml : c로 구현된 가장 빠름.
  - html5lib : 웹 브라우저 방식으로 HTML 해석
  - html.parser

- 특정 parser의 경우, 특정 태그가 포함되지 않는 오류가 있을 수 있음.

In [6]:

```
soup = BeautifulSoup(page, 'lxml')
soup
```

Out[6]:

```
<html>
<head><title>나의 홈페이지</title>
</head><body>
<div>
<a href="https://www.naver.com/">naver</a>
<a href="https://www.google.com">google</a>
<img height="300" src="dog_01.jpg" width="500"/>
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
<p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>
<p class="p3"> 강아지 사진과 네이버 링크 </p>
<p id="p4"> 간단한 나의 홈페이지를 만들다.</p>
<p class="p3"> 강아지 사진과 네이버 링크222 </p>
</div>
</body>
</html>
```

In [7]:

```
## soup 정보에서 head 부분 가져오기
soup.head
```

Out[7]:

```
<head><title>나의 홈페이지</title>
</head>
```

In [8]:

```
## soup 정보에서 body 부분 가져오기
soup.body
```

Out[8]:

```
<body>
<div>
<a href="https://www.naver.com/">naver</a>
<a href="https://www.google.com">google</a>
<img height="300" src="dog_01.jpg" width="500"/>
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
<p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>
<p class="p3"> 강아지 사진과 네이버 링크 </p>
<p id="p4"> 간단한 나의 홈페이지를 만들다.</p>
<p class="p3"> 강아지 사진과 네이버 링크222 </p>
</div>
</body>
```

In [9]:

```
## soup 정보에서 body 부분안의 p태그 부분 가져오기
soup.body.p
```

Out[9]:

```
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
```

In [10]:

```
## soup 정보에서 body 부분안의 p태그 부분 텍스트 정보 가져오기
soup.body.p.text
```

Out[10]:

```
' 내가 가장 좋아하는 동물은 강아지입니다.'
```

In [11]:

```
print(soup.prettify())
```

```
<html>
 <head>
  <title>
   나의 홈페이지
  </title>
 </head>
 <body>
  <div>
   <a href="https://www.naver.com/">
    naver
   </a>
   <a href="https://www.google.com">
    google
   </a>
   <img height="300" src="dog_01.jpg" width="500"/>
   <p>
    내가 가장 좋아하는 동물은 강아지입니다.
   </p>
   <p>
    나는 그리고 네이버 홈페이지에 자주 갑니다.
   </p>
   <p class="p3">
    강아지 사진과 네이버 링크
   </p>
   <p id="p4">
    간단한 나의 홈페이지를 만들다.
   </p>
   <p class="p3">
    강아지 사진과 네이버 링크222
   </p>
  </div>
 </body>
</html>
```

2-2 children 정보를 활용하는 방법

- children에 대해 알아보기
- 지정된 태그보다 한 레벨 아래 위치한 태그

In [12]:

```
soup.body.children
```

Out[12]:

```
<list_iterator at 0x1a54b763580>
```

body 태그 아래 children 내용 확인

In [13]:

```
list(soup.body.children)
```

Out[13]:

```
['₩n',
 <div>
 <a href="https://www.naver.com/">naver</a>
 <a href="https://www.google.com">google</a>
 <img height="300" src="dog_01.jpg" width="500"/>
 <p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
 <p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>
 <p class="p3"> 강아지 사진과 네이버 링크 </p>
 <p id="p4"> 간단한 나의 홈페이지를 만들다.</p>
 <p class="p3"> 강아지 사진과 네이버 링크222 </p>
 </div>,
 '₩n']
```

In [14]:

```
tmp = list(soup.body.children)[1]
tmp
```

Out[14]:

```
<div>
<a href="https://www.naver.com/">naver</a>
<a href="https://www.google.com">google</a>
<img height="300" src="dog_01.jpg" width="500"/>
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
<p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>
<p class="p3"> 강아지 사진과 네이버 링크 </p>
<p id="p4"> 간단한 나의 홈페이지를 만들다.</p>
<p class="p3"> 강아지 사진과 네이버 링크222 </p>
</div>
```

In [15]:

```
list(tmp.children)[3]
```

Out[15]:

```
<a href="https://www.google.com">google</a>
```

2-3 find_all(), find()을 이용한 정보 얻기

- find_all() : 조건이 맞는 전체 정보를 리스트로 가져오기
- find() : 조건이 맞는 정보 하나만 가져오기

In [16]:

```
soup.find_all('p')
```

Out[16]:

```
[<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>,
 <p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>,
 <p class="p3"> 강아지 사진과 네이버 링크 </p>,
 <p id="p4"> 간단한 나의 홈페이지를 만들다.</p>,
 <p class="p3"> 강아지 사진과 네이버 링크222 </p>]
```

In [17]:

```
soup.find('p')
```

Out[17]:

```
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
```

In [18]:

```
soup.find('div')
```

Out[18]:

```
<div>
<a href="https://www.naver.com/">naver</a>
<a href="https://www.google.com">google</a>
<img height="300" src="dog_01.jpg" width="500"/>
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
<p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>
<p class="p3"> 강아지 사진과 네이버 링크 </p>
<p id="p4"> 간단한 나의 홈페이지를 만들다.</p>
<p class="p3"> 강아지 사진과 네이버 링크222 </p>
</div>
```

- HTML 속성 정보인 class_, id로 활용하여 정보를 가져올 수 있음.

In [19]:

```
soup.find_all('p', class_='p3')
```

Out[19]:

```
[<p class="p3"> 강아지 사진과 네이버 링크 </p>, <p class="p3"> 강아지 사진과 네이버
링크222 </p>]
```

In [20]:

```
soup.find_all(id='p4')
```

Out[20]:

```
[<p id="p4"> 간단한 나의 홈페이지를 만들다.</p>]
```

2-4 p 태그의 문자들만 가져오기

In [21]:

```
for ptag in soup.find_all('p'):
    print(ptag)
```

```
<p> 내가 가장 좋아하는 동물은 강아지입니다.</p>
<p> 나는 그리고 네이버 홈페이지에 자주 갑니다.</p>
<p class="p3"> 강아지 사진과 네이버 링크 </p>
<p id="p4"> 간단한 나의 홈페이지를 만들다.</p>
<p class="p3"> 강아지 사진과 네이버 링크222 </p>
```

In [22]:

```
for ptag in soup.find_all('p'):
    print(ptag.get_text())
```

```
 내가 가장 좋아하는 동물은 강아지입니다.
 나는 그리고 네이버 홈페이지에 자주 갑니다.
 강아지 사진과 네이버 링크
 간단한 나의 홈페이지를 만들다.
 강아지 사진과 네이버 링크222
```

2-5 링크 가져오기(link)

In [23]:

```
soup.find_all('a')
```

Out[23]:

```
[<a href="https://www.naver.com/">naver</a>,
 <a href="https://www.google.com">google</a>]
```

In [24]:

```
links = soup.find_all('a')
print(links[1]['href'])
print(links[1].string)
```

```
https://www.google.com (https://www.google.com)
google
```

In [25]:

```
for each in links:
    href = each['href']
    text = each.string
    print(text + ' -> ' + href)
```

```
naver -> https://www.naver.com/ (https://www.naver.com/)
google -> https://www.google.com (https://www.google.com)
```