

## Chap 2. BeautifulSoup로 원하는 값 추출 후 저장하기

지난 챕터에서 우리는 웹사이트에서 특정 키워드로 자동 검색하는 방법을 배웠습니다.

이번 시간에는 검색된 결과에서 원하는 데이터만 추출하여 txt형식으로 저장하는 방법을 살펴보겠습니다.

HTML 코드에서 특정 태그 값이나 데이터를 추출하기 위해서는 BeautifulSoup라는 파이썬 라이브러리를 사용하면 아주 편하게 작업할 수 있습니다.

그래서 이번 챕터에서 공부할 학습목표는 아래와 같습니다.

### \*\* 학습 목표 \*\*

1. BeautifulSoup의 역할과 설정을 할 수 있다.
2. BeautifulSoup의 find(), find\_all(), select() 함수로 원하는 데이터만 추출할 수 있다.
3. 수집된 데이터를 txt 형식의 파일로 저장할 수 있다.

위 내용을 한가지씩 자세하게 살펴보겠습니다.

### 1. BeautifulSoup (뷰티풀 수프) 역할과 설치하기

BeautifulSoup는 아주 복잡한 HTML 코드에서 지정된 특정 태그나 값을 추출할 때 사용하는 라이브러리입니다. 물론 반드시 BeautifulSoup 라이브러리를 사용하지 않아도 웹 크롤러를 만드는 것은 충분히 가능하지만 웹에서 우리가 원하는 데이터를 가져오기 위해서 필요한 복잡하고 번거로운 작업들이 BeautifulSoup를 이용하면 아주 간단하게 해결이 됩니다.

예를 들어 웹에서 우리가 원하는 이미지를 가져온다고 했을 때 BeautifulSoup를 사용하지 않고 코딩을 하게 되면 HTML 코드 전체를 대상으로 정규식 등을 사용하여 우리가 원하는 이미지가 있는 태그를 찾아 내야 합니다.

하지만 BeautifulSoup를 이용하면 단 한 줄로 이 작업을 대신할 수 있다는 것이죠.

마치 땅을 팔 때 삽으로 100 번 파야 할 분량을 포크레인을 쓰면 1 번에 끝나는 것과 비슷한 경우입니다. BeautifulSoup는 파이썬에서 Web 관련 작업을 하시려면 꼭 배워야 할 라이브러리입니다.

BeautifulSoup에 대한 자세한 정보는 아래의 링크를 참조해 주세요.

한글 링크 : <https://www.crummy.com/software/BeautifulSoup/bs4/doc.ko/>

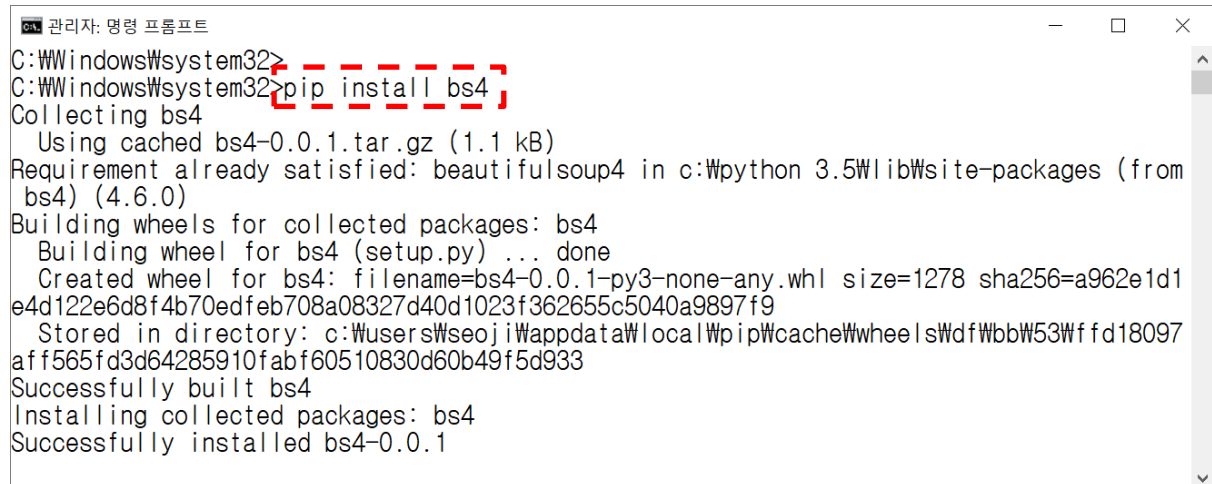
영어 링크 : <https://www.crummy.com/software/BeautifulSoup/bs4/doc/index.html>

그리고 웹 크롤러를 만들려면 파이썬의 기초 문법은 물론이고 HTML 언어의 내용까지 일부 알고 있어야 합니다. 혹시라도 이후 내용을 보시다가 잘 생각 안 나면 이 책의 Part 3에 있는 기본 문법의 내용을 먼저 살펴보세요.

BeautifulSoup를 사용하기 위해서는 먼저 설치를 해야 합니다.

[ 파이썬 능력자 너도 될 수 있어~! - 서진수 저 - ]

설치하는 방법은 여러가지가 있는데 가장 간편하게 사용할 수 있는 방법은 cmd 창에서 pip 명령을 이용하는 것입니다. 아래와 같이 cmd (윈도의 명령 프롬프트) 창을 열고 아래의 명령을 실행하세요.



```
관리자: 명령 프롬프트
C:\Windows\system32>
C:\Windows\system32>pip install bs4
Collecting bs4
  Using cached bs4-0.0.1.tar.gz (1.1 kB)
Requirement already satisfied: beautifulsoup4 in c:\python 3.5\lib\site-packages (from bs4) (4.6.0)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1278 sha256=a962e1d1e4d122e6d8f4b70edfeb708a08327d40d1023f362655c5040a9897f9
  Stored in directory: c:\users\seoji\appdata\local\pip\cache\wheels\df\bb\53\ff\d18097aff565fd3d64285910fabf60510830d60b49f5d933
Successfully built bs4
Installing collected packages: bs4
Successfully installed bs4-0.0.1
```

위 명령으로 설치가 되었다면 BeautifulSoup를 사용할 준비는 모두 마쳤습니다.

## 2. BeautifulSoup를 사용하여 데이터 추출하기

Beautiful Soup는 아주 많은 훌륭한 기능들을 가지고 있습니다.

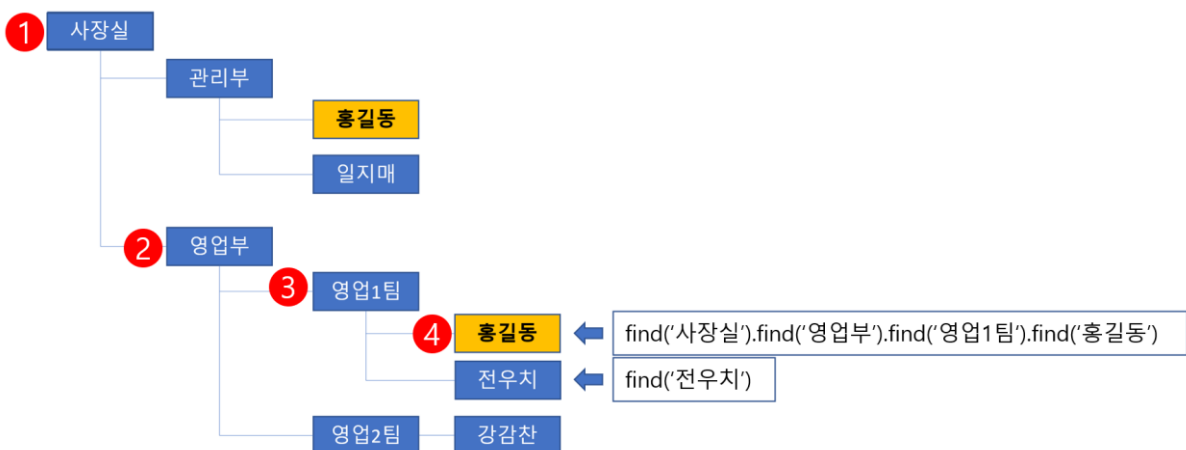
하지만 우리는 이 책에서 그 모든 기능들을 다 볼 수는 없고 웹 크롤러를 만드는 데 반드시 필요한 부분을 위주로 살펴보겠습니다.

Beautiful Soup로 “웹 페이지의 데이터를 가져온다”는 것은 “웹 페이지의 HTML 태그를 가져온다”라는 말과 같습니다.

그럼 BeautifulSoup를 이용해서 웹 페이지의 특정 태그를 가져오기 위해서는 어떻게 해야 할까요?? 특정한 태그를 추출하는 여러 가지 방법이 있지만 이 책에서는 `find()` / `find_all()` / `select()` 함수를 사용하는 방법을 살펴보겠습니다.

함수를 사용하기 전에 찾아올 태그를 지정하는 원리를 먼저 살펴보겠습니다.

아래 그림을 보세요.

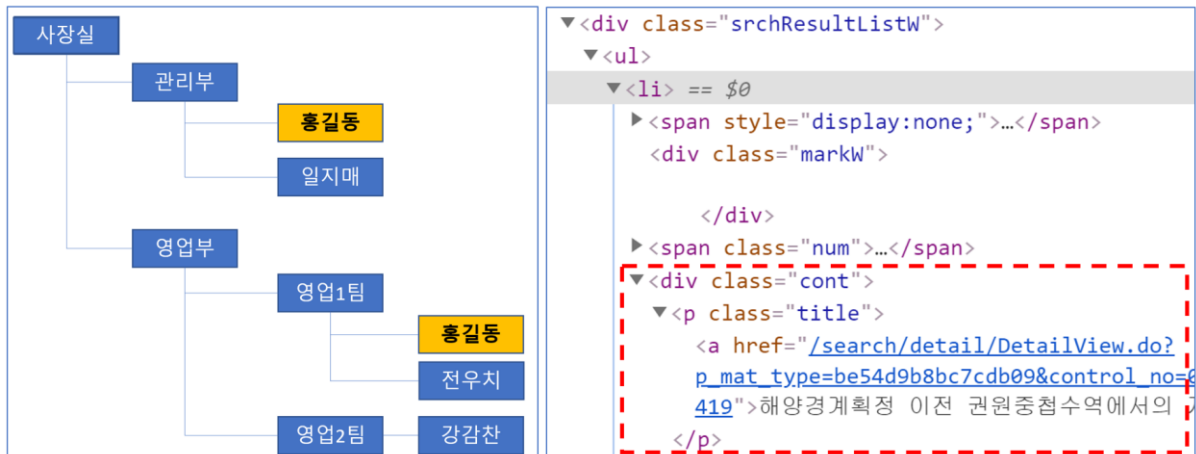


위 그림에서 “홍길동” 직원을 찾고 싶다고 할 때 관리부에도 “홍길동” 이 있고 영업1팀에도 “홍길동” 이 있습니다. 이 때 영업1팀의 “홍길동”을 찾고 싶다면 위 그림에 있는 것처럼 맨 처음에 `find('사장실').find('영업부').find('영업1팀').find('홍길동')` 과 같이 최 상위 레벨부터 차례대로 적어줘야 합니다. 이렇게 해야 하는 이유는 “홍길동”이 2명 이상이기 때문에 정확한 부서를 다 적어줘야 원하는 “홍길동”을 정확하게 찾을 수가 있습니다.

그런데 위 그림에서 “전우치”를 찾고 싶다면 그냥 `find('전우치')` 를 해도 찾을 수 있습니다. 왜냐면 회사 전체에 ‘전우치’가 1명밖에 없기 때문에 쉽게 찾을 수가 있기 때문입니다.

HTML 코드의 구조도 위의 회사 조직도와 비슷하게 계층으로 이루어져 있습니다.

아래 그림으로 비교해 볼까요?



위 그림의 왼쪽 회사 조직도에서 만약에 “관리부 직원 다 데려와” 라고 하면 자동으로 홍길동과 일지매가 한꺼번에 오겠죠?

즉 상위 태그의 값을 가져오면 자동적으로 해당 태그 아래에 있는 하위태그의 값은 다 가져오게 됩니다. 즉 위 그림의 오른쪽 HTML 태그에서 <div class="cont"> 값을 찾아 오라고 하면 자동적으로 그 아래 있는 <p class="title"> 값을 가져오게 된다는 뜻입니다.

이제 각 함수들의 사용방법을 보겠습니다.

## 1) find( ) 함수 – 주어진 조건을 만족하는 첫 번째 태그 값만 가져오기

find() 함수를 이용하면 HTML 코드 안에서 원하는 태그를 가져올 수 있습니다.

아래 그림을 보세요.

```

1 #Beautiful Soup 예제 1
2
3 from bs4 import BeautifulSoup
4 ex1 = '''
5 <html>
6   <head>
7     <title> HTML 연습 </title>
8   </head>
9   <body>
10    <p align="center"> text 1 </p>
11    
12  </body>
13 </html> '''
14
15 soup = BeautifulSoup(ex1 , 'html.parser')
16 soup.find('title')

```

<title> HTML 연습 </title>

위의 그림은 title 태그를 가져오는 모습입니다.

Beautiful Soup를 활용하여 특정 태그를 찾을 때 작업 순서가 아주 중요합니다.

위 그림에서 3번 행에서 BeautifulSoup를 사용하도록 불러온 후 4번 행부터 13번 행까지 예제로 사용할 HTML 코드를 만들었습니다. 실제 크롤러를 만들 때는 이 부분에 특정 웹 페이지의 HTML

소스 코드가 들어가겠죠?

그리고 15번 행에서 HTML 코드를 BeautifulSoup 에게 넘겨서 파싱(분석) 하고 그 결과를 soup 이라는 변수에 저장하였습니다. 그리고 16번 행에서 분석 결과가 저장된 soup 변수에서 find함수 에게 'title' 이라는 태그를 찾도록 시켰습니다.

HTML 코드 안에서 어떤 특정태그를 가져오고 싶다면 위와 같이 find함수의 인수에 태그의 이름을 전달해 주는 겁니다. 참 간단하죠?

그럼 이번에는 p태그를 찾아볼까요?

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

```
<p align="center"> text 1 </p>
```

만약에 찾고 싶은 태그가 없다면 아무 내용이 나오지 않습니다.

find( )함수는 동일한 태그가 여러 개 있을 경우에 첫번째 태그 1개만 가져옵니다.

아래 그림을 보세요.

```
1 #Beautiful Soup 예제 2
2
3 from bs4 import BeautifulSoup
4 ex1 = '''
5 <html>
6     <head>
7         <title> HTML 연습 </title>
8     </head>
9     <body>
10        <p align="center"> text 1 </p>
11        <p align="righth"> text 2 </p>
12        <p align="left"> text 3 </p>
13        
14    </body>
15 </html> '''
16
17 soup = BeautifulSoup(ex1, 'html.parser')
```

위 그림을 보면 10번행부터 12번 행까지 <p> 태그가 총 3개가 있습니다.

이제 p 태그를 가져오겠습니다. 아래 그림을 보세요.

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

```
<p align="center"> text 1 </p>
```

위 그림을 보니까 총 3개의 p 태그 중에서 첫번째 1건만 가져오는 거 보이죠?

이번에는 태그에다가 속성을 추가로 지정하여 사용하는 방법입니다.

먼저 html코드 중 가운데 부분을 다음과 같이 수정하고 진행하겠습니다.

```

1  #Beautiful Soup 예제 3
2
3  from bs4 import BeautifulSoup
4  ex1 = '''
5  <html>
6      <head>
7          <title> HTML 연습 </title>
8      </head>
9      <body>
10         <p align="center"> text 1 </p>
11         <p align="righth"> text 2 </p>
12         <p align="left"> text 3 </p>
13         
14     </body>
15 </html> '''
16
17 soup = BeautifulSoup(ex1 , 'html.parser')
```

위의 html 코드를 보면 align속성이 각각 center, right, left인 p태그들이 있습니다.

각각의 p태그를 속성을 이용해서 조회해보도록 할까요??

```
1 soup.find('p',align='center')
```

```
<p align="center"> text 1 </p>
```

```
1 soup.find('p', align="righth")
```

```
<p align="righth"> text 2 </p>
```

```
1 soup.find('p', align="left")
```

```
<p align="left"> text 3 </p>
```

위 그림처럼 속성값을 이용하여 원하는 태그를 추출할 수도 있습니다.

지금까지 살펴본 방법 외에도 find( ) 함수에 여러가지 옵션들이 있지만 가장 많이 사용하는 방법들을 소개해 드렸으니 열심히 연습해 주세요.

## 2) find\_all() 함수 - 해당 태그가 여러 개 있을 경우 한꺼번에 모두 가져오기

find\_all() 함수는 find() 함수와는 다르게 원하는 태그가 몇 개가 있던지 모두 가져옵니다.

find\_all() 함수를 사용하기 위해서 HTML코드를 아래 그림처럼 바꾸고 진행하겠습니다.

```

1  #Beautiful Soup 예제 4
2
3  from bs4 import BeautifulSoup
4  ex1 = '''
5  <html>
6      <head>
7          <title> HTML 연습 </title>
8      </head>
9      <body>
10         <p align="center"> text 1 </p>
11         <p align="center"> text 2 </p>
12         <p align="center"> text 3 </p>
13         
14     </body>
15 </html> '''
16
17 soup = BeautifulSoup(ex1, 'html.parser')
```

위 코드의 17번 행에서 HTML코드가 변경되어 다시 BeautifulSoup 객체를 생성했습니다.

그리고 아래 그림처럼 find\_all() 함수를 사용하여 모든 <p> 태그를 추출하면 됩니다.

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

```
[<p align="center"> text 1 </p>,
<p align="center"> text 2 </p>,
<p align="center"> text 3 </p>]
```

일반적인 웹 페이지는 동일한 태그가 아주 많이 있기 때문에 find\_all() 함수를 사용하는 경우가 많이 발생합니다.

지금까지는 BeautifulSoup를 사용해서 한 가지 태그를 찾아오는 방법들을 살펴보았습니다.

그런데 코딩을 하다 보면 지금까지 했던 것과는 다르게 한가지의 태그만 찾는 것이 아니라 여러 가지의 태그를 찾아야 하는 상황이 자주 있습니다.

예를 들어 p태그와 img태그를 같이 찾고 싶을 때처럼 말이죠.

이런 경우는 다음 그림처럼 사용하면 됩니다.

```
1 soup.find_all(['p', 'img'])
```

```
[<p align="center"> text 1 </p>,
<p align="center"> text 2 </p>,
<p align="center"> text 3 </p>,
]
```

### 3) select( ) 함수 사용하기

앞에서 find( ) 함수와 find\_all( ) 함수를 이용하여 원하는 태그를 찾는 방법을 소개했습니다. 그런데 이 함수들 외에도 select( ) 함수로 css\_selector를 활용해서 원하는 태그를 찾는 방법도 많이 사용합니다. 실제 크롤링 할 때는 세 가지 방법을 함께 사용하는 경우가 많기에 이 방법도 자세하게 설명하겠습니다. 예제로 아래와 같은 html을 사용하겠습니다.

```

1  ex2 = '''
2  <html>
3      <head>
4          <h1> 사야할 과일
5      </head>
6      <body>
7          <h1> 시장가서 사야할 과일 목록
8              <div> <p id='fruits1' class='name1' title='바나나'> 바나나
9                  <span class = 'price'> 3000원 </span>
10                 <span class = 'count'> 10개 </span>
11                 <span class = 'store'> 바나나가게 </span>
12                 <a href = 'https://www.banana.com'> banana.com </a>
13             </p>
14         </div>
15         <div> <p id='fruits2' class='name2' title='체리'> 체리
16             <span class = 'price'> 100원 </span>
17             <span class = 'count'> 50개 </span>
18             <span class = 'store'> 체리가게 </span>
19             <a href = 'https://www.cherry.com'> cherry.com </a>
20         </p>
21     </div>
22     <div> <p id='fruits3' class='name3' title='오렌지'> 오렌지
23         <span class = 'price'> 500원 </span>
24         <span class = 'count'> 20개 </span>
25         <span class = 'store'> 오렌지가게 </span>
26         <a href = 'https://www.orange.com'> orange.com </a>
27     </p>
28 </div>
29 </body>
30 </html> '''
31
32 soup2 = BeautifulSoup(ex2 , 'html.parser')
```

위 내용을 입력하시 힘든 분들은 제가 제공하는 “bs테스트용 html예제.txt” 파일을 열어서 복사해서 파이썬 콘솔에 붙여넣기 하시면 됩니다.

이제 select( ) 함수를 사용해서 다양한 태그들을 찾아보겠습니다.

select( ) 함수를 이용한 방법의 장점은 다양한 옵션들을 사용할 수 있다는 것입니다.

아래의 실습을 통해 여러가지 옵션들을 사용하여 원하는 태그를 추출하는 방법들을 살펴보겠습니다.



**(1) select('태그이름')**

```
1 soup2.select('p')
```

```
[<p class="name1" id="fruits1" title="바나나"> 바나나
    <span class="price"> 3000원 </span>
  <span class="count"> 10개 </span>
  <span class="store"> 바나나가게 </span>
  <a href="https://www.banana.com"> banana.com </a>
</p>, <p class="name2" id="fruits2" title="체리"> 체리
    <span class="price"> 100원 </span>
  <span class="count"> 50개 </span>
  <span class="store"> 체리가게</span>
  <a href="https://www.cherry.com"> cherry.com </a>
</p>, <p class="name3" id="fruits3" title="오렌지"> 오렌지
    <span class="price"> 500원 </span>
  <span class="count"> 20개 </span>
  <span class="store"> 오렌지가게</span>
  <a href="https://www.orange.com"> orange.com </a>
</p>]
```

위 그림은 <p> 태그의 내용을 모두 추출한 그림입니다.

**(2)select('.클래스명')**

```
1 soup2.select('.name1')
```

```
[<p class="name1" id="fruits1" title="바나나"> 바나나
    <span class="price"> 3000원 </span>
  <span class="count"> 10개 </span>
  <span class="store"> 바나나가게 </span>
  <a href="https://www.banana.com"> banana.com </a>
</p>]
```

위 그림은 class 이름이 name1 인 태그만 추출했습니다. 위 명령에서 태그 이름 앞에 . 이 있다는 거 주의하세요.

**(3)select(' 상위태그 > 하위태그 > 하위태그 ' )**

```
1 soup2.select('div > p > span')
```

```
[<span class="price"> 3000원 </span>,
  <span class="count"> 10개 </span>,
  <span class="store"> 바나나가게 </span>,
  <span class="price"> 100원 </span>,
  <span class="count"> 50개 </span>,
  <span class="store"> 체리가게</span>,
  <span class="price"> 500원 </span>,
  <span class="count"> 20개 </span>,
  <span class="store"> 오렌지가게</span>]
```

위 화면은 여러 태그가 단계적으로 있을 때 아주 요긴하게 사용하는 방법입니다.  
부등호 앞 뒤로 공백이 반드시 들어가야 한다는 점 주의하세요.  
그런데 위와 같이 특정 태그가 여러 개 있는 경우 한꺼번에 모두 나와서 보기가 번거롭죠?  
이때는 앞의 과정 중 리스트에 배웠던 인덱싱 관련 기능을 함께 사용하면 됩니다.  
아래 그림을 보세요.

```
1 soup2.select('div > p > span')[0]
<span class="price"> 3000원 </span>

1 soup2.select('div > p > span')[1]
<span class="count"> 10개 </span>

1 soup2.select('div > p > span')[2]
<span class="store"> 바나나가게 </span>
```

동일한 태그가 여러 개가 있을 경우 특정 위치 값을 추출할 때 아주 요긴하게 사용할 수 있는 방법입니다.

#### (4) select('상위태그.클래스이름 > 하위태그.클래스이름')

```
1 soup2.select('p.name1 > span.store')
[<span class="store"> 바나나가게 </span>]
```

#### (5)select('#아이디명')

```
1 soup2.select('#fruits1')
[<p class="name1" id="fruits1" title="바나나"> 바나나
    <span class="price"> 3000원 </span>
    <span class="count"> 10개 </span>
    <span class="store"> 바나나가게 </span>
    <a href="https://www.banana.com"> banana.com </a>
</p>]
```

#### (6)select('#아이디명 > 태그명.클래스명')

```
1 soup2.select('#fruits1 > span.store')
[<span class="store"> 바나나가게 </span>]
```

### (7)select('태그명[속성1=값1]')

```
1 soup2.select('a[href]')
```

```
[<a href="https://www.banana.com"> banana.com </a>,
 <a href="https://www.cherry.com"> cherry.com </a>,
 <a href="https://www.orange.com"> orange.com </a>]
```

```
1 soup2.select('a[href]')[0]
```

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

## 4) 태그 뒤의 텍스트만 추출하기

앞의 실습으로 find() 함수와 find\_all() 함수와 select() 함수를 이용해서 태그를 찾아냈습니다.

그런데 정작 우리에게 필요한 것은 화면에 보여지는 텍스트 내용이 가장 중요합니다.

태그를 찾았지만 태그의 콘텐츠(문장이나 이미지 등)를 가져오지 못하면 무용지물이겠죠?

그래서 지금부터 태그내의 문장을 가져오는 방법을 알아보도록 하겠습니다.

```
1 txt = soup.find('p')
2 txt.string
```

```
'text 1 '
```

위 그림을 보면 p태그가 실제로는 3개가 존재하지만 find() 함수를 사용하면 제일 먼저 나오는 p태그를 가져온다는 사실 기억하시죠?

그래서 1개만 가져와서 txt 변수에 담았습니다. 그리고 p태그를 찾은 후에 객체에서 string을 가져왔습니다. string은 태그의 문장을 가지고 있습니다.

태그에 포함된 문장만을 가지고 올 때 자주 사용되니까 잘 기억해 두세요.

그런데 위의 string을 사용하면 한 번에 한 문장 밖에 가져오지 못합니다.

그렇다면 태그 안에 존재하는 여러 개의 문장을 한꺼번에 가져오려면 어떻게 해야 할까요??

바로 아래와 같은 방법을 사용하시면 됩니다.

```
1 txt2 = soup.find_all('p')
2 for i in txt2 :
3     print(i.string)
```

```
text 1
text 2
text 3
```

head태그를 제외하고 body태그의 모든 문장을 가져오기 위해서 body태그의 strings를 사용했습니다. strings에는 태그 안에 있는 모든 문장들이 저장되어 있습니다. 그래서 위의 그림과 같이 출력을 해주니까 body태그의 모든 문장들이 출력 되었습니다.

이번에는 get\_text() 함수를 사용하는 방법을 알려 드립니다.

이 함수도 태그 내의 텍스트 데이터를 추출할 때 아주 많이 사용합니다.

```
1 txt3 = soup.find_all('p')
2 for i in txt3 :
3     print(i.get_text())
```

```
text 1
text 2
text 3
```

지금까지 기본적인 BeautifulSoup의 사용방법을 꼭 살펴보았습니다.

내용도 생소하고 많이 어려웠죠?

하지만 크롤러를 만들기 위해서는 꼭 알아야 하는 부분이므로 계속 반복해서 연습해 주세요~

### 3. 수집된 내용을 txt 형식의 파일로 저장하기

Beautiful Soup을 이용하여 원하는 데이터를 추출하는 방법을 배웠습니다.

그래서 이번에는 RISS 사이트에서 특정 검색어로 검색한 후 결과를 txt 파일에 저장하는 방법을 살펴보겠습니다.

이번에 살펴볼 내용은 Chap 1에서 진행했던 RISS 사이트에서 자동 검색하는 코드에 내용을 추가하겠습니다.

\*\*\*\*\*

#### 1 #Chap 13. riss.kr 사이트에서 특정 키워드로 자동 검색한 후 내용 추출하여 저장하기

#### 2 #Step 1. 필요한 모듈을 로딩합니다

```
3 from selenium import webdriver
4 from selenium.webdriver.common.by import By
5 from selenium.webdriver.common.keys import Keys
6 from selenium.webdriver.chrome.service import Service
7 import time
8
```

#### 9 #Step 2. 사용자에게 검색 관련 정보들을 입력 받습니다.

```
10 print("=" * 100)
11 print(" 이 크롤러는 riss 사이트의 논문 자료 수집용 웹크롤러입니다.")
12 print("=" * 100)
13 query_txt = input("1.수집할 자료의 키워드는 무엇입니까?(예: 해양자원): ")
14 print("\n")
15
```

#### 16 #Step 3. 크롬 드라이버 설정 및 웹 페이지 열기

```
17 s = Service("c:/py_temp/chromedriver.exe")
```

```

18 driver = webdriver.Chrome(service=s)
19
20 url = 'https://www.riss.kr/'
21 driver.get(url)
22 time.sleep(5)
23 driver.maximize_window()
24
25 #Step 4. 자동으로 검색어 입력 후 조회하기
26 element = driver.find_element(By.ID,'query')
27 driver.find_element(By.ID,'query').click( )
28 element.send_keys(query_txt)
29 element.send_keys("\n")
30
31 #Step 5. 학위 논문 선택하기
32 driver.find_element(By.LINK_TEXT,'학위논문').click()
33 time.sleep(2)
34
35 #Step 6.Beautiful Soup 로 본문 내용만 추출하기
36 from bs4 import BeautifulSoup
37 html_1 = driver.page_source #현재 페이지의 전체 소스코드를 다 가져오기
38 soup_1 = BeautifulSoup(html_1, 'html.parser')
39
40 content_1 = soup_1.find('div','srchResultListW').find_all('li')
41 for i in content_1 :
42     print(i.get_text().replace("\n"," ").strip())
43     print("\n")
44
45 #Step 7. 표준 출력 방향을 바꾸어 txt 파일에 저장하기
46 import sys
47 f_name = input('결과를 저장할 파일명을 쓰세요(예: c:\wwtemp\wwriss.txt): ')
48
49 orig_stdout = sys.stdout
50 file = open(f_name , 'a' , encoding='UTF-8')
51 sys.stdout = file #모니터에 출력하지 말고 file 에 출력해라
52
53 for i in content_1 :
54     print(i.get_text().replace("\n",""))
55
56 file.close()

```

[ 파이썬 능력자 너도 될 수 있어~! - 서진수 저 - ]

```

57 sys.stdout = orig_stdout #원래대로 변경 - 다시 화면에 출력시켜라
58
59 print('요청하신 데이터 수집 작업이 정상적으로 완료되었습니다')
60 print('수집된 결과는 %s 에 저장되었습니다' %f_name)

```

\*\*\*\*\*

위 코드에서 Step 1 ~ Step 4 까지는 앞에서 설명했던 부분이라서 설명을 생략합니다.  
Step 5부분부터 볼까요?

### 31 #Step 5.학위 논문 선택하기

```

32 driver.find_element(By.LINK_TEXT,'학위논문').click()
33 time.sleep(2)

```

위 코드에서 32번 행은 RISS 사이트에서 학위논문 카테고리로 가기 위해 해당 메뉴를 클릭하라는 의미입니다. 위 코드를 실행하면 아래 그림에서 학위논문 메뉴를 클릭합니다.



이처럼 특정 메뉴 이름이나 페이지 번호 등을 클릭할 때는 이 방법을 많이 사용합니다.

조회를 하고 메뉴까지 클릭했으니 이번에는 BeautifulSoup 을 활용하여 데이터를 추출하는 부분을 살펴보겠습니다. 아래 코드를 보세요.

### 35 #Step 6.Beautiful Soup 로 본문 내용만 추출하기

```

36 from bs4 import BeautifulSoup
37 html_1 = driver.page_source #현재 페이지의 전체 소스코드를 다 가져오기
38 soup_1 = BeautifulSoup(html_1, 'html.parser')
39
40 content_1 = soup_1.find('div','srchResultListW').find_all('li')
41 for i in content_1 :
42     print(i.get_text().replace("\n", " ").strip())
43     print("\n")

```

위 코드의 36번 행에서 BeautifulSoup 을 사용하기 위해 import 했습니다.

그 후 37번 행에서 현재 크롬 드라이버로 열려 있는 웹 페이지의 전체 소스코드를 다 가져 온 후 38번 행에서 파싱을 한 후 soup\_1 이라는 변수에 저장을 했습니다.

이제 중요한 부분이 나옵니다.

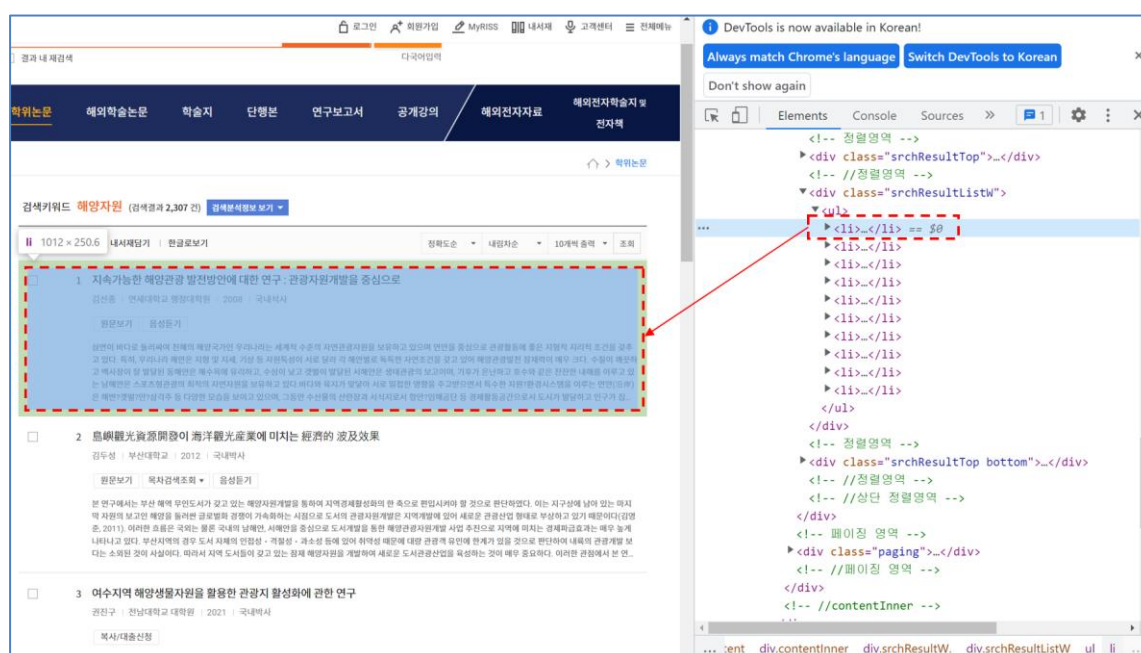
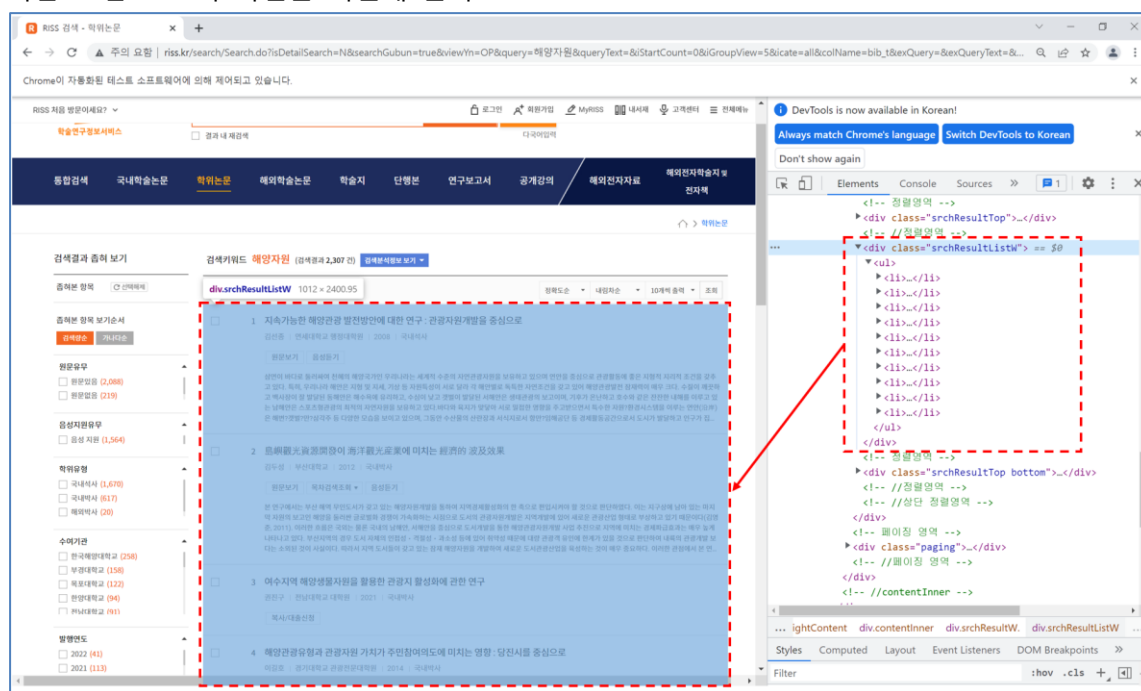
40번 행에서 전체 웹 페이지에서 검색된 결과만 추출하기 위해서

content\_1 = soup\_1.find('div','srchResultListW').find\_all('li') 라는 코드를 작성했습니다.

이 부분의 코드를 이해하는 것이 아주 중요합니다.

HTML 소스 코드를 보면 <div class= "srchResultListW">를 선택하면 웹 페이지에서 검색된 결과 부분만 하이라이트 되면서 선택됨을 알 수 있습니다. 그리고 그 아래에 있는 li 태그 1개가 검색된 결과 1 건에 해당된다는 것을 알 수 있습니다.

다음 그림으로 이 사실을 확인해 볼까요?



[ 파이썬 능력자 너도 될 수 있어~! - 서진수 저 - ]

위 그림을 보니까 `content_1 = soup_1.find('div','srchResultListW').find_all('li')` 라는 코드가 가져오는 데이터는 검색된 전체 결과라는 것을 이해하겠죠?

위 작업을 진행하면 `content_1` 변수에는 10개의 게시물 관련 내용이 들어가게 되겠죠?

li태그가 총 10개니까요.

그래서 41번 행과 같이 for 반복문으로 1개씩 텍스트 결과를 출력시켰습니다.

이렇게 하면 아래와 같이 텍스트가 출력 되겠죠?

1 지속가능한 해양관광 발전방안에 대한 연구 : 관광자원개발을 중심으로 김선중 연세대학교 행정대학원 2008 국내석사 RANK : 13880191  
원문보기 음성듣기  
삼면이 바다로 둘러싸여 천혜의 해양국가인 우리나라는 세계적 수준의 자연관광자원을 보유하고 있으며 연안을 중심으로 관광활동에 좋은 지형적 지리적 조건을 갖추고 있다. 특히, 우리나라 해안은 지형 및 지세, 기상 등 자원특성이 서로 달라 각 해안별로 독특한 자연조건을 갖고 있어 해양관광발전 잠재력이 매우 크다. 수질이 깨끗하고 백사장이 잘 발달된 동해안은 해수욕에 유리하고, 수심이 낮고 갯벌이 발달된 서해안은 생태관광의 보고이며, 기후가 온난하고 호수와 같은 잔잔한 내해를 이루고 있는 남해안은 스포츠형관광의 최적의 자연자원을 보유하고 있다. 바다와 육지가 맞닿아 서로 밀접한 영향을 주고받으면서 특수한 자연·환경시스템을 이루는 연안(沿岸)은 해변?갯벌?만?삼각주 등 다양한 모습을 보이고 있으며, 그동안 수산물의 산란장과 서식지로서 항만?임해공간 등 경제활동공간으로서 도시가 발달하고 인구가 집중되어있는 중요한 활동공간이었으나, 이제는 레저스포츠 등 관광활동공간으로서 그 중요성이 더해가고 있다. 우리나라는 우수한 자연관광자원을 보유하고 있으나 그동안 육상위주의 레저 및 관광정책추진으로 해양관광은 크게 발전하지 못하고 있다. 그러나 국민들의 소득수준 증가와 더불어 주말 휴무제 정착에 따른 여가시간의 증대 등 라이프스타일 변화로 직접체험을 즐길 수 있는 해양레저스포츠 등 관광수요는 크게 증가할 것으로 보인다. 이러한 좋은 여건에도 불구하고 자치단체에서는 사전 타당성 검토가 부족한 대단위 관광단지 위주의 개발계획을 만들 경쟁적으로 추진하고 있어 증가하고 있는 해양관광수요에는 효과적으로 대응하지 못하는 문제점을 노출하고 있다. 따라서 우수한 자연자원을 효과적으로 활용하고, 새로운 해양관광수요를 수용하고 발전시킬 수 있는 활성화대책 마련이 긴요한 실정이다. 육지관광과는 달리 해양에서의 관광활동은 안전에 대한 관심뿐만 아니라 환경에 대한 검토도 충분히 이루어져야만 좋은 관광환경이 조성될 수 있다. 특히, 자치단체별로 관광자원개발계획 추진시 이에 대한 SWOT분석을 이행하고 있으나 대부분 최적의 조건만을 생각하고 이를 검토하여 결국은 사업추진 원동력이 떨어지고 있다. 해양관광도 육상과 동일하게 취급되어져 특색을 갖추지 않고 차별화되지 못한 자원개발이 되고 있어 최근 많은 수요발생에도 효과적으로 대응하지 못하고 있는 실정이다. 따라서 우리나라 연안·해양에서의 해양관광발전을 위해 추진하고 있는 관광자원개발 사업에 대한 정책적 문제점을 살펴보고 해양관광자원개발 사업에 대한 SWOT를 제시하여 해양관광 활성화를 위한 바람직한 정책적 방안을 제시하였다.

( 위와 같은 결과가 총 10건이 출력되는데 지면 관계상 아래의 내용은 생략합니다)

이제 수집한 내용들을 파일로 저장하는 코드를 보겠습니다.

```
45 #Step 7. 표준 출력 방향을 바꾸어 txt 파일에 저장하기
46 import sys
47 f_name = input('결과를 저장할 파일명을 쓰세요(예: c:\WWtemp\WWriss.txt): ')
48
49 orig_stdout = sys.stdout
50 file = open(f_name , 'a' , encoding='UTF-8')
51 sys.stdout = file      #모니터에 출력하지 말고 file 에 출력해라
52
53 for i in content_1 :
54     print(i.get_text().replace("\n",""))
55
56 file.close()
57 sys.stdout = orig_stdout #원래대로 변경 - 다시 화면에 출력시켜라
58
59 print('요청하신 데이터 수집 작업이 정상적으로 완료되었습니다')
60 print('수집된 결과는 %s 에 저장되었습니다' %f_name)
```

위 코드의 47번 행에서 저장할 파일명을 입력 받았습니다.

그리고 49번 행부터 51번 행 까지가 중요합니다.

[ 파이썬 능력자 너도 될 수 있어~! - 서진수 저 - ]



txt 형식으로 저장하는 방법 중 open -> write -> close 방법은 이 책의 파일과 디렉토리 관리 부분에 자세하게 설명이 되어 있으니 참고하시고 여기서는 다른 방법을 한가지 더 소개하겠습니다.

다른 방법은 바로 표준 출력 방향을 바꾸어 저장하는 방법입니다.

원래 컴퓨터의 기본 출력 장치는 화면인 모니터이고 기본 입력 장치는 키보드입니다.

즉 사람이 결과를 출력(print)하라고 하면 자동으로 모니터에 결과를 출력시킨다는 의미입니다.

이것을 표준 출력이라고 부릅니다.

그런데 표준 출력 방향을 모니터에서 특정 파일로 바꾼다면 모니터에 출력될 내용들이 전부 파일에 저장되게 만들 수 있습니다.

먼저 위 코드의 46번 행에서 표준 입/출력을 관리하는 sys 모듈을 import 시켰습니다.

그리고 47번 행에서 결과를 저장할 파일의 경로와 이름을 입력 받아서 f\_name 변수에 저장했습니다. 그 후 49번 행에서 표준 출력 방향을 바꾼다고 알리고 50번 행에서 파일을 지정합니다.

그리고 51번 행에서 결과를 출력할 표준 출력을 파일로 지정합니다.

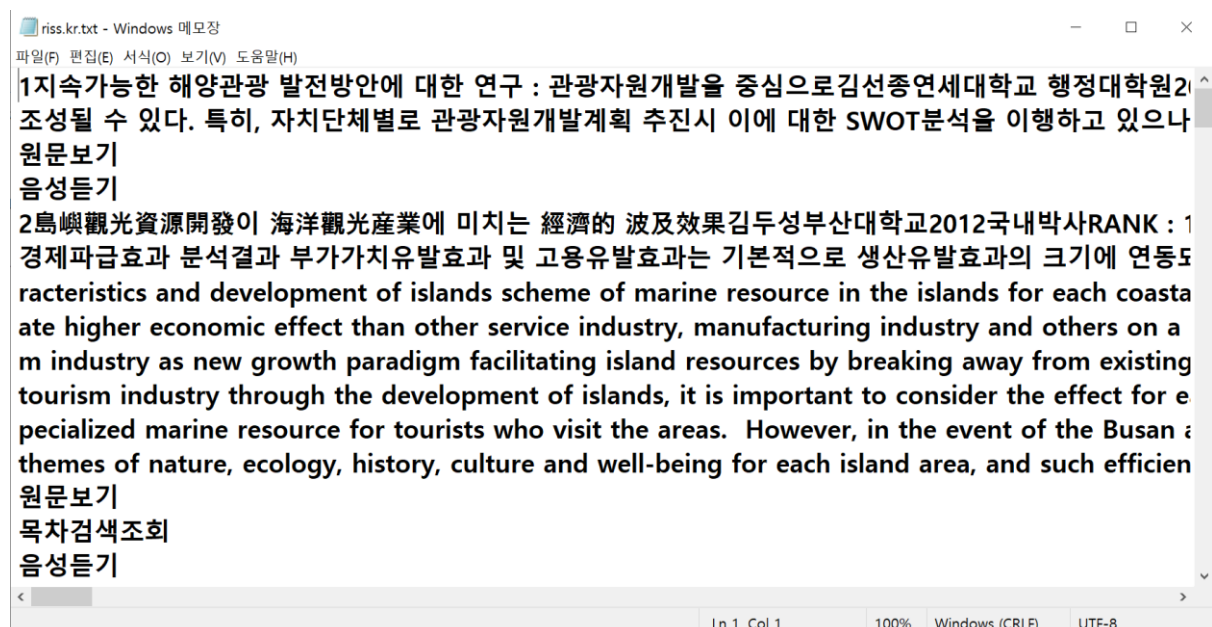
이렇게 하면 이제 이후로 화면에 출력되는 모든 내용들은 전부 파일에 다 출력이 됩니다.

그래서 53번행과 54번행을 수행하면 결과가 모니터 화면에 안보이고 파일에 저장되는 것입니다.

그리고 정말 중요한 것은 57번 행으로 모든 작업이 끝나면 표준 출력 장치를 원래대로 모니터로 바꾸어야 합니다.

그 후 59번 행과 60번 행에서 작업 결과를 안내하고 마무리합니다.

결과가 저장된 파일을 열어보면 아래 그림과 같이 잘 저장되어 있습니다.



지금까지 RISS 사이트에서 특정 키워드로 검색하여 특정 카테고리를 선택한 후 조회된 내용만 추출하여 txt 형식의 파일로 저장하는 것을 살펴보았습니다.

[ 파이썬 능력자 너도 될 수 있어~! - 서진수 저 - ]

#### 4. 연습 문제로 실력 굳히기

1. 네이버 사이트에서 “서진수 빅데이터” 로 검색 한 후 “뉴스” 카테고리 선택하여 조회된 기사들을 수집하여 txt 형식으로 저장하세요(단 파일경로와 이름은 자유롭게 선택하세요)
2. 다음 사이트에서 “서진수 빅데이터” 로 검색 한 후 “뉴스” 카테고리 선택하여 조회된 기사들을 수집하여 txt 형식으로 저장하세요(단 파일경로와 이름은 자유롭게 선택하세요)
3. 한국관광공사의 대한민국 구석구석 사이트(<https://korean.visitkorea.or.kr>) 에서 “제주도” 키워드로 검색 한 후 화면 오른쪽의 “어제의 인기 검색어” 목록을 수집하여 txt 형식으로 저장하세요.  
(단 파일경로와 이름은 자유롭게 선택하세요)
4. 한국관광공사의 대한민국 구석구석 사이트(<https://korean.visitkorea.or.kr>) 에서 “제주도” 키워드로 검색 한 후 검색 결과 목록을 수집하여 txt 형식으로 저장하세요.  
(단 파일경로와 이름은 자유롭게 선택하세요)

이번 챕터에서도 중요한 내용을 많이 배웠습니다.  
열심히 연습해서 꼭 독자님의 중요한 실력으로 만드세요~~!



데이터쟁이 서진수가 여러분을 힘차게 응원합니다~!

[ 파이썬 능력자 너도 될 수 있어~! - 서진수 저 - ]