

데이터 크롤링과 정제

2장. BeautifulSoup 활용

목차

- BeautifulSoup 라이브러리 기초
 - Tag를 활용한 요소 접근
- BeautifulSoup의 함수 활용
 - find() 함수
 - find_all() 함수
 - select() 함수
 - select_one() 함수

BeautifulSoup 라이브러리

■ BeautifulSoup 라이브러리

- HTML 문서나 XML 문서를 탐색해서 원하는 부분만 쉽게 추출
- BeautifulSoup 객체 생성

```
soup = BeautifulSoup(markup, "html.parser")
```

- 첫 번째 파라미터: HTML, XML 문서
- 두 번째 파라미터: 구문 분석기(다양한 해석기 지원)
 - html.parser, lxml, html5lib

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('https://www.daangn.com/hot_articles')
bs = BeautifulSoup(html.read(), 'html.parser')
print(bs.h1)
```

```
<h1 class="head-title" id="hot-articles-head-title">
```

```
    중고거래 인기매물
</h1>
```

bs.h1: 첫 번째 h1
태그를 반환

BeautifulSoup 기초 #1: 샘플 HTML 구성

```
html_example = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BeautifulSoup 활용</title>
</head>
<body>
    <h1 id="heading">Heading 1</h1>
    <p>Paragraph</p>
    <span class="red">BeautifulSoup Library Examples!</span>
    <div id="link">
        <a class="external_link" href="www.google.com">google</a>
        <div id="class1">
            <p id="first">class1's first paragraph</p>
            <a class="external_link" href="www.naver.com">naver</a>
            <p id="second">class1's second paragraph</p>
            <a class="internal_link" href="/pages/page1.html">Page1</a>
            <p id="third">class1's third paragraph</p>
        </div>
    </div>
    <div id="text_id2">
        Example page
        <p>g</p>
    </div>
    <h1 id="footer">Footer</h1>
</body>
</html>
'''
```

BeautifulSoup 기초 #2

- 태그를 사용하여 요소에 직접 접근하기
 - <title> 태그에 접근(`soup.태그명`)

```
soup = BeautifulSoup(html_example, 'html.parser')

print(soup.title) # <title> 태그 전체를 가져옴
print(soup.title.text) # <title> 태그의 텍스트만 리턴
print(soup.title.get_text()) # .text와 동일한 기능
```

```
<title>BeautifulSoup 활용</title>
BeautifulSoup 활용
BeautifulSoup 활용
```

- `태그명.parent`: 해당 태그를 포함하고 있는 부모

```
print(soup.title.parent)
```

```
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
<title>BeautifulSoup 활용</title>
</head>
```

BeautifulSoup 기초

- 태그를 사용하여 요소에 직접 접근하기
 - <body>태그에 접근

```
print(soup.body)
```

```
<body>
<h1 id="heading">Heading 1</h1>
<p>Paragraph</p>
<span class="red">BeautifulSoup Library Examples!</span>
<div id="link">
<a class="external_link" href="www.google.com">google</a>
<div id="class1">
<p id="first">class1's first paragraph</p>
<a class="external_link" href="www.naver.com">naver</a>
<p id="second">class1's second paragraph</p>
<a class="internal_link" href="/pages/page1.html">Page1</a>
<p id="third">class1's third paragraph</p>
</div>
</div>
<div id="text_id2">
    Example page
    <p>g</p>
</div>
<h1 id="footer">Footer</h1>
</body>
```

BeautifulSoup 기초

- 태그를 사용하여 요소에 직접 접근하기

- <h1> 태그 접근

- 동일한 태그가 여러 개 있는 경우, 첫 번째 요소를 추출

```
print(soup.h1)
print(soup.h1.get_text())
```

```
<h1 id="heading">Heading 1</h1>
Heading 1
```

- <a> 태그 접근

- 첫 번째 <a> 태그 요소 추출

```
print(soup.a)
```

```
<a class="external_link" href="www.google.com">google</a>
```

BeautifulSoup 기초: find() 함수

▪ find() 함수 파라미터

```
find(tag, attrs, recursive, text, keywords)
```

- 해당 조건에 맞는 맨 처음 검색 결과만 추출
- 이름, 속성, 속성값을 이용하여 원하는 태그를 찾을 수 있음

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

```
<div id="link">
<a class="external_link" href="www.google.com">google</a>
<div id="class1">
<p id="first">class1's first paragraph</p>
<a class="exteranl_link" href="www.naver.com">naver</a>
<p id="second">class1's second paragraph</p>
<a class="internal_link" href="/pages/page1.html">Page1</a>
<p id="third">class1's third paragraph</p>
</div>
</div>
```


BeautifulSoup 기초: find()

▪ find() 함수

- 여러 <div> 태그 중 특정 속성을 가지는 항목 추출
 - **딕셔너리 형태로 입력** (id속성의 값이 'text_id2'인 항목 검색)

```
print(soup.find('div', {'id': 'text_id2'}))
```

```
<div id="text_id2">
  Example page
  <p>g</p>
</div>
```

- **.text** 또는 **get_text()**
 - 추출된 요소에서 텍스트만 가져옴

```
div_text = soup.find('div', {'id': 'text_id2'})
print(div_text.get_text())
```

```
Example page
g
```

BeautifulSoup 기초: find() 함수

- <a> 태그 및 <a> 태그의 href 속성 추출

```
<a class="internal_link" href="/pages/page1.html">Page1</a>
```

```
href_link = soup.find('a', {'class': 'internal_link'}) # 딕셔너리 형태
href_link = soup.find('a', class_='internal_link') # class는 파이썬 예약어

print(href_link)
print(href_link['href']) # <a> 태그 내부 href 속성의 값(url)을 추출
print(href_link.get('href')) # ['href']와 동일 기능
print(href_link.text) # <a> Page1 </a> 태그 내부의 텍스트(Page1) 추출
```

```
<a class="internal_link" href="/pages/page1.html">Page1</a>
/pages/page1.html
/pages/page1.html
Page1
```

- <a> 태그 내부의 모든 속성의 값 가져오기: dict의 values() 호출

```
print(href_link.attrs.values()) # 모든 속성값 추출
values = list(href_link.attrs.values()) # dictionary의 값들을 리스트로 변경
print(values[0], values[1])
```

```
dict_values(['internal_link', '/pages/page1.html'])
['internal_link', '/pages/page1.html']
```

BeautifulSoup 기초: find() 함수

- href 속성의 값이 'www.google.com'인 항목 검색

```
href_value = soup.find(attrs={'href' : 'www.google.com'})  
print(href_value)  
print(href_value.text)
```

```
<a class="external_link" href="www.google.com">google</a>  
google
```

- span 태그의 속성 가져오기

```
# <span class="red">BeautifulSoup Library Examples!</span>  
span_tag = soup.find('span')
```

attrs: 딕셔너리 형태로 리턴

```
print('span tag:', span_tag)  
print('attrs:', span_tag.attrs) # attribute 속성 추출  
print('value:', span_tag.attrs['class']) # class 속성의 값 추출  
print('text:', span_tag.text)
```

```
span tag: <span class="red">BeautifulSoup Library Examples!</span>  
attrs: {'class': ['red']}  
value: ['red']  
text: BeautifulSoup Library Examples!
```

BeautifulSoup 기초: find_all() 함수

▪ find_all()

```
find_all(tag, attrs, recursive, text, limit, keywords)
```

• 검색된 모든 태그를 리턴(리스트 형태)

- **tag**: HTML 태그, 태그 이름으로 이루어진 리스트 전달
- **attrs**: 속성, 파이썬 딕셔너리를 받음 (or 속성)

```
bs.find_all('span', {'class': {'green', 'red'}}) # 'green' or 'red'
```

- **recursive**: 재귀 검색
 - True: 모든 문서에서 태그를 검색(자식, 자식의 자식을 검색)
 - False: 최상의 태그만 검색
- **text**: 텍스트 콘텐츠와 일치하는 문장 검색

```
princeList = bs.find_all(text='the prince')  
print('the prince count: ', len(princeList))
```

- **limit**: 일치하는 검색어를 몇 개까지 찾을 것인지 설정
 - limit = None: 제한 없음 (모두 검색)
 - limit = 1: find() 메소드와 동일
- **keyword**: BeautifulSoup 자체 기능과 중복

BeautifulSoup 기초: find_all() 함수

- 모든 div 태그 검색

```
div_tags = soup.find_all('div')
print(div_tags) # 전체 div 태그를 모두 검색 (리스트 형태로 반환)
```

```
[<div id="link">
<a class="external_link" href="www.google.com">google</a>
<div id="class1">
<p id="first">class1's first paragraph</p>
<a class="external_link" href="www.naver.com">naver</a>
<p id="second">class1's second paragraph</p>
<a class="internal_link" href="/pages/page1.html">Page1</a>
<p id="third">class1's third paragraph</p>
</div>
</div>, <div id="class1">
<p id="first">class1's first paragraph</p>
<a class="external_link" href="www.naver.com">naver</a>
<p id="second">class1's second paragraph</p>
<a class="internal_link" href="/pages/page1.html">Page1</a>
<p id="third">class1's third paragraph</p>
</div>, <div id="text_id2">
  Example page
  <p>g</p>
</div>]
```

```
print(len(div_tags))
print(div_tags[2])
```



```
3
<div id="text_id2">
  Example page
  <p>g</p>
</div>
```

BeautifulSoup 기초: find_all() 함수

- 모든 <a>태그 검색 및 속성 보기

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

for alink in links:
    print(alink)
    print('url:{0}, text:{1}'.format(alink['href'], alink.get_text()))
    print()
```

```
<a class="external_link" href="www.google.com">google</a>
url:www.google.com, text:google
```

```
<a class="exteranal_link" href="www.naver.com">naver</a>
url:www.naver.com, text:naver
```

```
<a class="internal_link" href="/pages/page1.html">Page1</a>
url:/pages/page1.html, text:Page1
```

BeautifulSoup 기초: find_all() 함수

- 특정 태그 중 여러 속성값을 한 번에 검색

- 여러 <a>태그에서 2개의 class 속성값 검색

- 'external_link', 'internal_link'만 검색

- 검색할 속성값을 리스트 형태로 추가

- {'class':['external_link', 'internal_link']}

```
link_tags = soup.find_all('a', {'class':['external_link', 'internal_link']})  
print(link_tags)
```

```
[<a class="external_link" href="www.google.com">google</a>,  
<a class="external_link" href="www.naver.com">naver</a>,  
<a class="internal_link" href="/pages/page1.html">Page1</a>]
```

- <p>태그의 id값이 'first', 'third'인 항목 검색

```
p_tags = soup.find_all('p', {'id': ['first', 'third']})  
  
for p in p_tags:  
    print(p)
```

```
<p id="first">class1's first paragraph</p>  
<p id="third">class1's third paragraph</p>
```

BeautifulSoup 기초: select() 함수

▪ select() 함수

- CSS selector로 tag객체를 찾아 리턴
- 조건에 맞는 모든 태그를 리턴
 - find_all()과 유사

▪ select_one() 함수

- 조건에 맞는 첫 번째 태그만 리턴
 - find()와 유사

▪ select_one()과 find() 차이점

- find()
 - 하위 태그를 찾을 때, 반복적으로 코드를 작성
 - `soup.find('div').find('p')`
- select()
 - 하위 태그를 찾을 때, 직접 하위 경로 지정
 - `soup.select_one('div > p')`

BeautifulSoup 기초: select() 함수

■ 사용법

```
select(selector, namespaces=None, limit=None, **kwargs):
```

- select('태그')
 - 해당 태그를 포함하는 모든 요소 리턴
- select('태그#id이름') 또는 select('#id이름')
 - 태그 내부의 id이름을 이용하여 검색: #id
- select('태그.클래스이름') 또는 select('.클래스이름')
 - 특정 태그에 포함된 클래스명 검색: 태그 이름은 생략 가능
 - .클래스이름
- select('상위태그 > 하위태그1 > 하위태그2')
 - 계층적으로 하위 태그 접근
- select('태그[속성1=값1]')
 - 특정 태그의 속성과 속성값을 이용한 검색

BeautifulSoup 기초: select_one() 함수

▪ select_one() 예제

- <head> 태그 검색

```
head = soup.select_one('head')  
print(head)
```

```
<head>  
<meta charset="utf-8"/>  
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>  
<title>BeautifulSoup 활용</title>  
</head>
```

- 첫 번째 <h1> 태그 검색

```
h1 = soup.select_one('h1') # 첫 번째 <h1> 태그 선택  
print(h1)
```

```
<h1 id="heading">Heading 1</h1>
```

BeautifulSoup 기초: select_one() 함수

▪ select_one() 함수 예제

- <h1>태그의 id 검색: #id

```
# <h1>태그의 id가 "footer" 인 항목 추출
heading = soup.select_one('h1#footer')
print(heading)
```

```
<h1 id="footer">Footer</h1>
```

- 클래스 이름 검색: 태그.클래스이름
 - 검색

```
class_link = soup.select_one('a.internal_link')
print(class_link)
```

```
<a class="internal_link" href="/pages/page1.html">Page1</a>
```

```
print(class_link.text)
print(class_link['href'])
```

```
Page1
/pages/page1.html
```

BeautifulSoup 기초: select_one() 함수

- 계층적 하위 태그 접근 #1
 - (상위태그 > 하위태그) 형식으로 접근

계층적 접근

```
link1 = soup.select_one('div#link > a.external_link')  
print(link1)
```

```
<a class="external_link" href="www.google.com">google</a>
```

```
1 <!DOCTYPE html>  
2 <html lang="en">  
3 <head>  
4   <meta charset="UTF-8">  
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6   <title>BeautifulSoup 활용</title>  
7 </head>  
8 <body>  
9   <h1 id="heading">Heading 1</h1>  
10  <p>Paragraph</p>  
11  <span class="red">BeautifulSoup Library Examples!</span>  
12  <div id="link">  
13    <a class="external_link" href="www.google.com">google</a>  
14  </div>  
15  <div id="class1">  
16    <p id="first">class1's first paragraph</p>  
17    <a class="external_link" href="www.naver.com">naver</a>  
18  </div>  
19  <p id="second">class1's second paragraph</p>  
20  <a class="internal_link" href="/pages/page1.html">Page1</a>  
21  <p id="third">class1's third paragraph</p>  
22 </div>  
23 <div id="text_id2">  
24   Example page  
25   <p>g</p>  
26 </div>  
27 <h1 id="footer">Footer</h1>  
28 </body>  
29 </html>
```

BeautifulSoup 기초: select_one() 함수

- 계층적 하위 태그 접근 #2
 - (상위태그 하위태그) 형식으로 접근

```
link2 = soup.select_one('div#class1 p#second')
print(link2)
print(link2.text)
```

```
<p id="second">class1's second paragraph</p>
class1's second paragraph
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>BeautifulSoup 활용</title>
7 </head>
8 <body>
9   <h1 id="heading">Heading 1</h1>
10  <p>Paragraph</p>
11  <span class="red">BeautifulSoup Library Examples!</span>
12  <div id="link">
13    <a class="external_link" href="www.google.com">google</a>
14
15    <div id="class1">
16      <p id="first">class1's first paragraph</p>
17      <a class="external_link" href="www.naver.com">naver</a>
18
19      <p id="second">class1's second paragraph</p>
20      <a class="internal_link" href="/pages/page1.html">Page1</a>
21      <p id="third">class1's third paragraph</p>
22    </div>
23  </div>
24  <div id="text_id2">
25    Example page
26    <p>g</p>
27  </div>
28  <h1 id="footer">Footer</h1>
29 </body>
30 </html>
```

BeautifulSoup 기초: select() 함수

▪ select() 함수

- 모든 <h1> 태그 검색

```
h1_all = soup.select('h1')  
print(h1_all)
```

```
[<h1 id="heading">Heading 1</h1>, <h1 id="footer">Footer</h1>]
```

- 모든 url 링크 검색

```
# html 문서의 모든 <a> 태그의 href 값 추출  
url_links = soup.select('a')  
for link in url_links:  
    print(link['href'])
```

```
www.google.com  
www.naver.com  
/pages/page1.html
```

BeautifulSoup 기초: select() 함수

- `<div id="class1">` 내부의 모든 url 검색

```
div_urls = soup.select('div#class1 > a')
print(div_urls)

print(div_urls[0]['href'])
```

```
[<a class="external_link" href="www.naver.com">naver</a>,
 <a class="internal_link" href="/pages/page1.html">Page1</a>]
```

www.naver.com

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>BeautifulSoup 활용</title>
7 </head>
8 <body>
9   <h1 id="heading">Heading 1</h1>
10  <p>Paragraph</p>
11  <span class="red">BeautifulSoup Library Examples!</span>
12  <div id="link">
13    <a class="external_link" href="www.google.com">google</a>
14
15    <div id="class1">
16      <p id="first">class1's first paragraph</p>
17      <a class="external_link" href="www.naver.com">naver</a>
18
19      <p id="second">class1's second paragraph</p>
20      <a class="internal_link" href="/pages/page1.html">Page1</a>
21      <p id="third">class1's third paragraph</p>
22    </div>
23  </div>
24  <div id="text_id2">
25    Example page
26    <p>g</p>
27  </div>
28  <h1 id="footer">Footer</h1>
29 </body>
30 </html>
```

BeautifulSoup 기초: select() 함수

■ 여러 항목 검색하기

- <h1>태그의 id가 "heading"과 "footer"를 모두 검색
 - `선택표(.)`로 나열함

```
# <h1 id="heading">과 <h1 id="footer"> 항목 가져오기
h1 = soup.select('#heading, #footer')
print(h1)
```

```
[<h1 id="heading">Heading 1</h1>, <h1 id="footer">Footer</h1>]
```

- <a>태그의 class이름이 "external_link"와 "internal_link" 모두 검색

```
url_links = soup.select('a.external_link, a.internal_link')
print(url_links)
```

```
[<a class="external_link" href="www.google.com">google</a>,
<a class="external_link" href="www.naver.com">naver</a>,
<a class="internal_link" href="/pages/page1.html">Page1 </a>]
```


BeautifulSoup 기초: select() 함수

```
national_anthem = '''
<!DOCTYPE html>
<html lang="en">
<head>
  <title>애국가</title>
</head>
<body>
  <div>
    <p id="title">애국가</p>
    <p class="content">
      동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세.<br>
      무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.<br>
    </p>
    <p class="content">
      남산 위에 저 소나무 철갑을 두른 듯 바람서리 불변함은 우리 기상일세.<br>
      무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.<br>
    </p>
    <p class="content">
      가을 하늘 공활한데 높고 구름 없이 밝은 달은 우리 가슴 일편단심일세.<br>
      무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.<br>
    </p>
    <p class="content">
      이 기상과 이 맘으로 충성을 다하여 괴로우나 즐거우나 나라 사랑하세.<br>
      무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.<br>
    </p>
  </div>
</body>
</html>
'''
```

BeautifulSoup 기초: select() 함수

■ 제목과 가사 내용 추출

```
bs4 = BeautifulSoup(national_anthem, 'html.parser')  
print(bs4.select_one('p#title').text)
```

애국가

```
contents = bs4.select('p.content')  
for content in contents:  
    print(content.text)
```

동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세.
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.

남산 위에 저 소나무 철갑을 두른 듯 바람서리 불변함은 우리 기상일세.
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.

가을 하늘 공활한데 높고 구름 없이 밝은 달은 우리 가슴 일편단심일세.
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.

이 기상과 이 맘으로 충성을 다하여 괴로우나 즐거우나 나라 사랑하세.
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세.

select() vs find()

검색 내용	select(), select_one() 사용	find_all(), find() 사용
class 이름이 <code>item_unit</code> 인 항목 모두 검색	<code>item_units = soup.select('.item_unit')</code>	<code>find_item_units = soup.find_all('span', {'class': 'item_unit'})</code>
div id가 <code>stats1</code> 인 항목 검색	<code>stats_one = soup.select_one('#stats1')</code>	<code>find_stats1 = soup.find('div', attrs={'id': 'stats1'})</code>
span 태그의 class 이름이 <code>item_number</code> 인 첫 번째 항목 검색	<code>item_number = soup.select_one('.item_number')</code>	<code>find_item_number = soup.find('span', {'class': 'item_number'})</code>
첫 번째 <code>item_number</code> 의 <code>text</code> 검색	<code>item_number_text = soup.select_one('.item_number').text</code>	<code>find_item_number_text = soup.find('span', {'class': 'item_number'}).text</code>
div id가 <code>stats3</code> 인 항목의 <code>item_unit</code> 값 검색	<code>stat3_item_number = soup.select_one('div#stats3 > span.item_unit').text</code>	<code>find_stats3 = soup.find('div', {'id': 'stats3'})</code> <code>find_item_unit = find_stats3.find('span', {'class': 'item_unit'}).text</code>

```
<div class="question">
  <div id="stats1">
    <span class="item_number">0</span>
    <span class="item_unit">votes</span>
  </div>
  <div id="stats2">
    <span class="item_number">10</span>
    <span class="item_unit">answer</span>
  </div>
  <div id="stats3">
    <span class="item_number">15</span>
    <span class="item_unit">views</span>
  </div>
</div>
```

참고 자료


BeautifulSoup Documentation Link

- <https://beautiful-soup-4.readthedocs.io/en/latest/#>

Beautiful Soup
latest

Search docs

- Beautiful Soup Documentation
 - Quick Start
- Installing BeautifulSoup
 - Making the soup
- Kinds of objects
- Navigating the tree
- Searching the tree
- Modifying the tree
- Output
- Specifying the parser to use
- Encodings
 - Line numbers
- Comparing objects for equality
- Copying BeautifulSoup objects
- Parsing only part of a document
- Troubleshooting
- Translating this documentation
- Beautiful Soup 3



Digital Ocean: Create your world-changing apps on the cloud developers love **Try now with a \$100 Credit**

Ad by EthicalAds · Monetize your site

Read the Docs v: latest

Docs » BeautifulSoup Documentation

[View page source](#)

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of BeautifulSoup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers BeautifulSoup version 4.8.1. The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

You might be looking for the documentation for **Beautiful Soup 3**. If so, you should know that BeautifulSoup 3 is no longer being developed and that support for it will be dropped on or after December 31, 2020. If you want to learn about the differences between BeautifulSoup 3 and BeautifulSoup 4, see [Porting code to BS4](#).

This documentation has been translated into other languages by BeautifulSoup users:

- [这篇文档当然还有中文版](#).
- [このページは日本語で利用できます\(外部リンク\)](#)
- [이 문서는 한국어 번역도 가능합니다.](#)
- [Este documento também está disponível em Português do Brasil.](#)

Getting help

If you have questions about BeautifulSoup, or run into problems, [send mail to the discussion group](#). If your problem involves parsing an HTML document, be sure to mention [what the `diagnose\(\)` function says](#) about that document.



BeautifulSoup4 메소드 이름 변경 #1

- BeautifulSoup4에서는 Python 스타일의 메소드 이름 사용
 - Java 스타일에서 Python 스타일(소문자)로 변경
 - 수업 교재는 bs3 스타일의 메소드 이름 사용

BeautifulSoup3 (bs3)	BeautifulSoup4 (bs4)
renderContents	encode_contents
replaceWith	replace_with
replaceWithChildren	unwrap
findAll	find_all
findAllNext	find_all_next
findAllPrevious	find_all_previous
findNext	find_next
findNextSibling	find_next_sibling
findNextSiblings	find_next_siblings
findParent	find_parent
findParents	find_parents

BeautifulSoup4 메소드 이름 변경 #2

BeautifulSoup3 (bs3)	BeautifulSoup4 (bs4)
<code>findPrevious</code>	<code>find_previous</code>
<code>findPreviousSibling</code>	<code>find_previous_sibling</code>
<code>findPreviousSiblings</code>	<code>find_previous_siblings</code>
<code>getText</code>	<code>get_text</code>
<code>nextSibling</code>	<code>next_sibling</code>
<code>previousSibling</code>	<code>previous_sibling</code>
<code>BeautifulSoup(parseOnlyThese=...)</code>	<code>BeautifulSoup(parse_only=...)</code>
<code>BeautifulSoup(fromEncoding=...)</code>	<code>BeautifulSoup(from_encoding=...)</code>
<code>Tag.has_key()</code>	<code>Tag.has_attr()</code>
<code>Tag.isSelfClosing</code>	<code>Tag.is_empty_element</code>
<code>UnicodeDammit.unicode</code>	<code>UnicodeDammit.unicode_markup</code>
<code>Tag.next</code>	<code>Tag.next_element</code>
<code>Tag.previous</code>	<code>Tag.previous_element</code>

BeautifulSoup4 메소드 이름 변경 #3

- Method()가 attribute로 변경

BeautifulSoup3 (bs3)	BeautifulSoup4 (bs4)
childGenerator()	children
nextGenerator()	next_elements
nextSiblingGenerator()	next_siblings
previousGenerator()	previous_elements
previousSiblingGenerator()	previous_siblings
recursiveChildGenerator()	descendants
parentGenerator()	parents

참고 사이트

- BeautifulSoup documentation
 - <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
- 참고 사이트
 - <https://goodthings4me.tistory.com/180>
 - <https://brownbears.tistory.com/414>
 - https://www.skytowner.com/explore/finding_elements_by_class_in_beautiful_soup
 - <https://yeo0.github.io/data/2018/09/20/2.-BeautifulSoup-활용하기/>
 - <https://velog.io/@jisu0807/웹크롤링-BeautifulSoup에서-find와-select-사용하기>



Questions?