

# 빅데이터 수집시스템 개발



## HTML 파싱하기

## 학습목표

- **BeautifulSoup** 라이브러리를 설치하고 API 구성과 주요 방법을 설명할 수 있다.
- **BeautifulSoup** 라이브러리를 사용하여 CSS 선택자를 이용한 DOM 객체를 찾을 수 있다.

## 학습내용

- BeautifulSoup 라이브러리 알아보기
- BeautifulSoup 라이브러리 응용

# BeautifulSoup 라이브러리 알아보기

## 1 주요 모듈 활용

### 1 BeautifulSoup이란?

#### BeautifulSoup

- HTML 및 XML 파일에서 데이터를 추출하기 위한 파이썬 라이브러리

- 파이썬에서 기본적으로 제공하는 라이브러리가 아니므로 별도 설치 필요
  - ➡ Anaconda에는 BeautifulSoup 패키지가 Site-packages로 설치되어 있음

Anaconda3 > pkgs > beautifulsoup4-4.7.1-py37\_1 > Lib > site-packages



beautifulsoup4-4.7.1.dist-info



bs4

# BeautifulSoup 라이브러리 알아보기



## 1 주요 모듈 활용

### 1 BeautifulSoup이란?

- 설치를 해야 한다면 다음 명령으로 설치

```
pip install beautifulsoup4
```

- HTML 및 XML 파일의 내용을 읽을 때 다음 파서(Parser) 이용

html.parser

lxml

lxml-xml

html5lib

- 파이썬이 내장하고 있는 파서 사용 가능
- 좀 더 성능이 좋은 파서를 추가로 설치하여 사용해도 됨

# BeautifulSoup 라이브러리 알아보기



## 1 주요 모듈 활용

### 2 HTML 파싱

1 BeautifulSoup의 메인 패키지인 bs 패키지에서 BeautifulSoup() 함수 импорт

2 파싱할 HTML 문서와 파싱에 사용할 파서(구문 분석기)를 지정하여 호출

➡ `bs4.BeautifulSoup` 객체 리턴

3 HTML 문서에 대한 파싱이 끝나면 트리 구조 형식으로 DOM 객체 생성

➡ `bs4.BeautifulSoup` 객체를 통해 접근 가능

```
from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
bs = BeautifulSoup(html_doc, 'lxml')
bs = BeautifulSoup(html_doc, 'lxml-xml')
bs = BeautifulSoup(html_doc, 'html5lib')
```

```
pip install lxml
pip install html5lib
```

# BeautifulSoup 라이브러리 알아보기



## 1 주요 모듈 활용

### 3 파서 라이브러리(Parser Library) 비교

파서	구현 방법	특징
Python's html.parser	BeautifulSoup (markup, "html.parser")	<ul style="list-style-type: none"> <li>• 추가 설치 필요 없음</li> <li>• 적당한 속도</li> </ul>
lxml's HTML parser	BeautifulSoup (markup, "lxml")	<ul style="list-style-type: none"> <li>• 추가 설치 필요</li> <li>• 속도가 빠름</li> </ul>
lxml's XML parser	BeautifulSoup (markup, "lxml- xml") BeautifulSoup (markup, "xml")	<ul style="list-style-type: none"> <li>• 속도가 빠름</li> <li>• 추가 설치 필요</li> </ul>
html5lib	BeautifulSoup (markup, "html5lib")	<ul style="list-style-type: none"> <li>• 속도가 느림</li> <li>• 추가 설치 필요</li> <li>• 웹 브라우저와 동일한 방식으로 페이지의 파싱 지원</li> </ul>

# BeautifulSoup 라이브러리 알아보기



## 1 주요 모듈 활용

### 4 bs4.BeautifulSoup 객체의 태그 접근 방법

- HTML 문서를 파싱하고 bs4.BeautifulSoup 객체 생성

```
bs = BeautifulSoup(html_doc, 'html.parser')
```

- <html>, <head> 태그와 <body> 태그는 제외하고 접근하려는 태그에 **계층 구조**를 적용
- 태그명을 . 연산자와 함께 사용

bs.태그명

bs.태그명.태그명

bs.태그명.태그명.태그명

bs.태그명.태그명.태그명.태그명

# BeautifulSoup 라이브러리 알아보기

## 1 주요 모듈 활용

### 4 bs4.BeautifulSoup 객체의 태그 접근 방법

- HTML 문서의 내용을 파싱하여 BeautifulSoup 객체 생성

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Test BS</title>
  </head>
  <body>
    <h1>테스트</h1>
  </body>
</html>
```

```
bs =
BeautifulSoup(HTML문서,
'html.parser')
```

계층 구조를 적용한  
DOM 객체 생성

```
html
  head
    meta
    title
      Test BS
  body
    h1
      테스트
```

<h1> 태그 접근 방법

```
bs.html.body.h1
bs.html.h1
bs.h1
```



# BeautifulSoup 라이브러리 알아보기

## 1 주요 모듈 활용

### 5 태그의 정보 추출

- `bs4.element.Tag` 객체의 주요 속성과 메서드

#### 태그명 추출

`bs.태그명.name`

#### 속성 추출

`bs.태그명['속성명']`

`bs.태그명.attrs`

#### 콘텐츠 추출

`bs.태그명.string`  
`bs.태그명.text`  
`bs.태그명.contents`  
`bs.태그명.strings`  
`bs.태그명.get_text()`

`bs.태그명.string.strip()`  
`bs.태그명.text.strip()`  
`bs.태그명.stripped_strings`  
`bs.태그명.get_text(strip=True)`

# BeautifulSoup 라이브러리 알아보기

## 1 주요 모듈 활용

### 6 태그로부터 다른 태그로 이동

#### 부모 태그 추출

```
bs.태그명.parent
```

#### 자식 태그들 추출

```
bs.태그명.children
```

#### 형제 태그 추출

```
bs.태그명.next_sibling  
bs.태그명.previous_sibling  
bs.태그명.next_siblings  
bs.태그명.previous_siblings
```

#### 자손 태그들 추출

```
bs.태그명.descendants
```

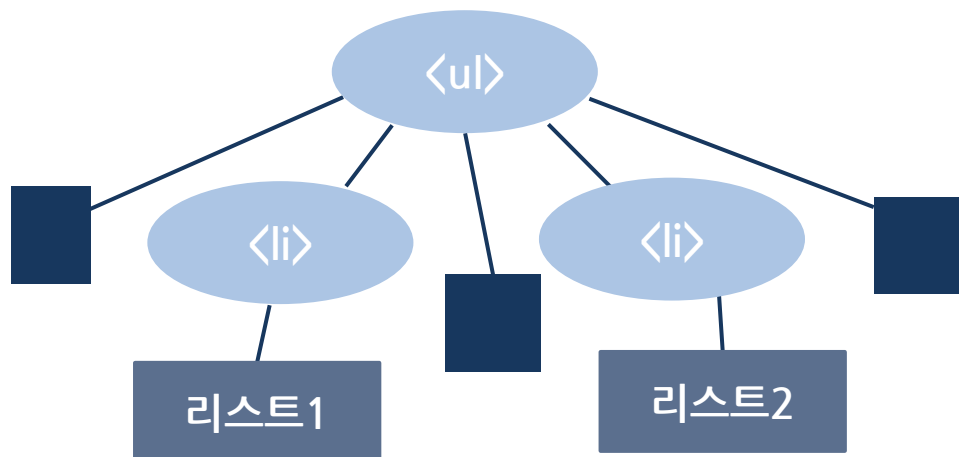
# BeautifulSoup 라이브러리 알아보기

## 1 주요 모듈 활용

### 6 태그로부터 다른 태그로 이동

- **bs4.element.Tag** 객체의 주요 속성과 메서드

```
<ul>  
  <li>리스트1</li>  
  <li>리스트2</li>  
</ul>
```



# BeautifulSoup 라이브러리 알아보기

## 2 소스 분석

### 1 소스 분석(1)

```
#파일명 : exam4_1.py
html_doc = """
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Test BeautifulSoup</title>
  </head>
  <body>
    <h1>테스트</h1>
  </body>
</html> """
from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
print(type(bs))
```

```
PS C:\example\myvscode> & C:/Users/UNICO/Anaconda3/python.exe
c:/example/myvscode/unit4/exam4_1.py
<class 'bs4.BeautifulSoup'>
```

# BeautifulSoup 라이브러리 알아보기

## 2 소스 분석

### 2 소스 분석(2)

```
#파일명 : exam4_2.py
html_doc = """
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <title>Test BeautifulSoup</title>
  </head>
  <body>
    <p align="center">P태그의 콘텐츠</p>
    
  </body>
</html> """
from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
print(bs.prettify())
```



# BeautifulSoup 라이브러리 알아보기



## 2 소스 분석

### 1 소스 분석(1)

```
PS C:\example\myvscode> & C:/Users/UNICO/Anaconda3/python.exe c:/example/myv
scode/unit4/exam4_2.py
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>
      Test BeautifulSoup
    </title>
  </head>
  <body>
    <p align="center">
      p태그의 콘텐츠
    </p>
    
  </body>
</html>
```

# BeautifulSoup 라이브러리 알아보기

## 2 소스 분석

### 3 소스 분석(3)

```
#파일명 : exam4_3.py
html_doc = """
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <title>Test BeautifulSoup</title>
  </head>
  <body>
    <p align="center">P태그의 콘텐츠</p>
    
    <ul>
      <li>테스트1<strong>강조</strong></li>
      <li>테스트2</li>
      <li>테스트3</li>
    </ul>
  </body>
</html> """

from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
print(type(bs.title), ':', bs.title)
print(type(bs.title.name), ':', bs.title.name)
print(type(bs.title.string), ':', bs.title.string)
print('-----')
print(type(bs.p['align']), ':', bs.p['align'])
print(type(bs.img['src']), ':', bs.img['src'])
print(type(bs.img.attrs), ':', bs.img.attrs)
```



# BeautifulSoup 라이브러리 알아보기



## 2 소스 분석

### 3 소스 분석(3)

```
PS C:\example\myvscode> & C:/Users/UNICO/Anaconda3/python.exe c:/example/myv  
scode/unit4/exam4_3.py  
<class 'bs4.element.Tag'> : <title>Test BeautifulSoup</title>  
<class 'str'> : title  
<class 'bs4.element.NavigableString'> : Test BeautifulSoup  
-----  
<class 'str'> : center  
<class 'str'> : http://unico2013.dothome.co.kr/image/flower.jpg  
<class 'dict'> : {'src': 'http://unico2013.dothome.co.kr/image/flower.jpg',  
'width': '300'}
```



# BeautifulSoup 라이브러리 알아보기

## 2 소스 분석

### 4 소스 분석(4)

```
#파일명 : exam4_4.py
html_doc = """
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <title>Test BeautifulSoup</title>
  </head>
  <body>
    <p align="center">P태그의 콘텐츠</p>
    
    <ul>
      <li>테스트1 <strong>강조</strong></li>
      <li>테스트2</li>
      <li>테스트3</li>
    </ul>
  </body>
</html> """

from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
print("[ string 속성 ]")
print(type(bs.p.string), ':', bs.p.string)
print(type(bs.ul.string), ':', bs.ul.string)
print(type(bs.ul.li.string), ':', bs.ul.li.string)
print(type(bs.ul.li.strong.string), ':', bs.ul.li.strong.string)
```



# BeautifulSoup 라이브러리 알아보기

## 2 소스 분석

### 4 소스 분석(4)

```
from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
print("[ string 속성 ]")
print(type(bs.p.string), ': ', bs.p.string)
print(type(bs.ul.string), ': ', bs.ul.string)
print(type(bs.ul.li.string), ': ', bs.ul.li.string)
print(type(bs.ul.li.strong.string), ': ', bs.ul.li.strong.string)
```

```
PS C:\example\myvscode> & C:/Users/UNICO/Anaconda3/python.exe c:/example/myvscode/
unit4/exam4_4.py
[ string 속성 ]
<class 'bs4.element.NavigableString'> : P태그의 콘텐츠
<class 'NoneType'> : None
<class 'NoneType'> : None
<class 'bs4.element.NavigableString'> : 강조
```

# BeautifulSoup 라이브러리 알아보기



## 2 소스 분석

### 4 소스 분석(4)

```
print("[ text 속성 ]")
print(type(bs.p.text), ': ', bs.p.text)
print(type(bs.ul.text), ': ', bs.ul.text)
print(type(bs.ul.li.text), ': ', bs.ul.li.text)
print(type(bs.ul.li.strong.text), ': ', bs.ul.li.strong.text)
```



```
[ text 속성 ]
<class 'str'> : p태그의 콘텐츠
<class 'str'> :
테스트1강조
테스트2
테스트3

<class 'str'> : 테스트1강조
<class 'str'> : 강조
```


# BeautifulSoup 라이브러리 알아보기



## 2 소스 분석

### 4 소스 분석(4)

```
print("[ contents 속성 ]")
print(type(bs.p.contents), ': ', bs.p.contents)
print(type(bs.ul.contents), ': ', bs.ul.contents)
print(type(bs.ul.li.contents), ': ', bs.ul.li.contents)
print(type(bs.ul.li.strong.contents), ': ', bs.ul.li.strong.contents)
```



```
[ contents 속성 ]
<class 'list'> : ['P태그의 콘텐츠']
<class 'list'> : ['\n', <li>테스트1<strong>강조</strong></li>, '\n', <li>테스트2</li>, '\n', <li>테스트3</li>, '\n']
<class 'list'> : ['테스트1', <strong>강조</strong>]
<class 'list'> : ['강조']
```

# BeautifulSoup 라이브러리 응용



## 1 메서드를 사용한 웹 페이지 파싱

### 1 bs4.BeautifulSoup 객체의 주요 메서드

- HTML 문서에 대한 파싱이 끝나고 생성된 트리 구조 형식의 DOM 객체
  - ➡ bs4.BeautifulSoup 객체의 속성으로 접근 가능
- 다음에 제시된 메서드로도 가능

#### 태그 찾기 기능의 주요 메서드

- `find_all()`
- `find()`
- `select()`

#### 태그 찾기 기능의 기타 메서드

- `find_parents()`와 `find_parent()`
- `find_next_siblings()`과 `find_next_sibling()`
- `find_previous_siblings()`와 `find_previous_sibling()`
- `find_all_next()`와 `find_next()`
- `first_all_previous()`와 `first_previous()`

# BeautifulSoup 라이브러리 응용



## 1 메서드를 사용한 웹 페이지 파싱

### 2 bs.find\_all()

주어진 기준에 맞는  
모든 태그들을 찾아옴



결과는  
**bs4.element.ResultSet**  
객체로 리턴

```
find_all(name=None, attrs={}, recursive=True,  
text=None, limit=None, **kwargs)
```

# BeautifulSoup 라이브러리 응용



## 1 메서드를 사용한 웹 페이지 파싱

### 2 bs.find\_all()

- 기준 설정

태그명

정규표현식을  
적용한  
태그명

태그명  
리스트

속성 정보

함수

논리값

# BeautifulSoup 라이브러리 응용



## 1 메서드를 사용한 웹 페이지 파싱

### 2 bs.find\_all()

- 호출 예제

```
find_all('div')
find_all(['p', 'img'])
find_all(True)
find_all(re.compile('^b'))
find_all(id='link2')
find_all(id=re.compile("para$"))
find_all(id=True)
find_all('a', class_="sister")
find_all(src=re.compile("png$"), id='link1')
find_all(attrs={'src':re.compile('png$'), 'id':'link1'})
find_all(text='example')
find_all(text=re.compile('example'))
find_all(text=re.compile('^test'))
find_all(text=['example', 'test'])
find_all('a', text='python')
find_all('a', limit=2)
find_all('p', recursive=False)
```



# BeautifulSoup 라이브러리 응용

## 1 메서드를 사용한 웹 페이지 파싱

### 3 bs.find()

주어진 기준에  
맞는 태그 한 개를  
찾아옴

결과는  
`bs4.element.Tag`  
객체로 리턴

결과값이 없으면  
None을 리턴

- `find()`는 `find_all()`에 `limit=1`로 설정한 것과 동일하게 수행
- `find_all()`에서 사용하는 아규먼트 값을 `find()`에서도 동일하게 사용 가능

```
find(name=None, attrs={}, recursive=True, text=None, **kwargs)
```

# BeautifulSoup 라이브러리 응용



## 1 메서드를 사용한 웹 페이지 파싱

### 3 bs.find()

- 기준 설정

태그명

정규표현식을  
적용한  
태그명

태그명  
리스트

속성 정보

함수

논리값

```
find('div') == find_all('div', limit=1)
find(re.compile('^b')) == find_all(re.compile('^b'),
limit=1)
```

# BeautifulSoup 라이브러리 응용

## 1 메서드를 사용한 웹 페이지 파싱

### 4 bs.select()

주어진 CSS 선택자에  
맞는 모든 태그를  
찾아옴

결과는 **list** 객체로 리턴

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

- CSS 선택자를 적용한 호출

```
select('태그명')  
select('.클래스명')  
select('#아이디명')  
select('태그명.클래스명')
```

# BeautifulSoup 라이브러리 응용



## 1 메서드를 사용한 웹 페이지 파싱

### 4 bs.select()

- 자식 선택자 및 자손 선택자를 사용하면 HTML문서의 트리 구조를 적용하여 태그를 찾을 수 있음

```
select('상위태그명 > 자식태그명 > 손자태그명')
select('상위태그명.클래스명 > 자식태그명.클래스명')
select('상위태그명.클래스명  자손태그명')
select('상위태그명 > 자식태그명  자손태그명')
select('#아이디명 > 태그명.클래스명')
select('태그명[속성]')
select('태그명[속성=값]')
select('태그명[속성$=값]')
select('태그명[속성^=값]')
select('태그명:nth-of-type(3)')
```

# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 1 소스 분석(1)

```
#파일명 : exam4_5.py
html_doc="""
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <title>Test BeautifulSoup</title>
  </head>
  <body>
    <p align="center"> text contents </p>
    <p align="right"> text contents 2 </p>
    <p align="left"> text contents 3 </p>
    
    <div>
      <p>text contents 4</p>
    </div>
  </body>
</html> """
```



# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 1 소스 분석(1)

```
from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
print(type(bs.find('p')))
print(type(bs.find_all('p')))
print("-----")
print(bs.find('title'))
print(bs.find('p'))
print(bs.find('img'))
print("-----")
ptags = bs.find_all('p')
print(ptags)
print("-----")
for tag in ptags :
    print(tag)
```



# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 1 소스 분석(1)

```
PS C:\example\myvscode> & C:/Users/UNICO/Anaconda3/python.exe c:/example/
myvscode/unit4/exam4_5.py
<class 'bs4.element.Tag'>
<class 'bs4.element.ResultSet'>
-----
<title>Test BeautifulSoup</title>
<p align="center"> text contents </p>

-----
[<p align="center"> text contents </p>, <p align="right"> text contents
2 </p>, <p align="left"> text contents 3 </p>, <p>text contents 4</p>]
-----
<p align="center"> text contents </p>
<p align="right"> text contents 2 </p>
<p align="left"> text contents 3 </p>
<p>text contents 4</p>
```

# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 2 소스 분석(2)

```
#파일명 : exam4_6.py
html="""
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <title>Test BeautifulSoup</title>
  </head>
  <body>
    <p align="center"> text contents </p>
    <p align="right" class="myp"> text contents 2 </p>
    <p align="left" a="b"> text contents 3 </p>
    
  </body>
</html> """
```



```
from bs4 import BeautifulSoup
bs = BeautifulSoup(html, 'html.parser')
print(bs.find('p', align="center"))
print(bs.find('p', class_="myp"))
print(bs.find('p', align="left"))
print("-----")
print(bs.find('p', attrs={"align":"center"}))
print(bs.find('p', attrs={"align":"right", "class":"myp"}))
print(bs.find('p', attrs={"align":"left"}))
```



# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 2 소스 분석(2)

```
PS C:\example\myvscode> & C:/Users/UNICO/Anaconda3/python.exe c:/example/
myvscode/unit4/exam4_6.py
<p align="center"> text contents </p>
<p align="right" class="myp"> text contents 2 </p>
<p a="b" align="left"> text contents 3 </p>
-----
<p align="center"> text contents </p>
<p align="right" class="myp"> text contents 2 </p>
<p a="b" align="left"> text contents 3 </p>
```

# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 3 소스 분석(3)



```
#파일명 : exam4_7.py
import requests
from bs4 import BeautifulSoup

req =
requests.get('http://unico2013.dothome.co.kr/crawling/exercise_css.html')
html = req.content
print(type(html))
html = html.decode('utf-8')
#print(html)
print("=====")
bs = BeautifulSoup(html, 'html.parser')
title = bs.select('h1')
title1 = bs.select('#f_subtitle')
title2 = bs.select('.subtitle')
title3 = bs.select('aside > h2')
img = bs.select('[src]')
print(type(title))
print(type(title[0]))
print("<h1>태그의 개수 : %d " %len(title))
print("f_subtitle이라는 id 속성을 갖는 태그 개수 : %d
"%len(title1))
print("subtitle이라는 class 속성을 갖는 태그 개수 : %d
"%len(title2))
print("aside 태그의 <h2> 자식 태그 개수 : %d " %len(title3))
print("src 속성을 갖는 태그 개수 : %d " %len(img))
```

# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 3 소스 분석(3)

```
for content in title:
    print(content.string)
print("-----")
for content in title1:
    print(content.text)
print("-----")
for content in title2:
    print(content.text)
print("-----")
for content in title3:
    print(content.text)
print("-----")
for content in img:
    print(content["src"])
```



# BeautifulSoup 라이브러리 응용



## 2 소스 분석

### 3 소스 분석(3)

```
PS C:\example\myvscode> & C:/Users/UNICO/Anaconda3/python.exe c:/example/
myvscode/unit4/exam4_7.py
<class 'bytes'>
=====
<class 'list'>
<class 'bs4.element.Tag'>
<h1>태그의 개수 : 1
f_subtitle이라는 id 속성을 갖는 태그 개수 : 1
subtitle이라는 class 속성을 갖는 태그 개수 : 2
aside 태그의 <h2> 자식 태그 개수 : 1
src 속성을 갖는 태그 개수 : 1
CSS 선택자 학습
-----
교육과정 소개
-----
웹 클라이언트 기술
학습 순서(수집)
-----
학습 순서(수집)
-----
https://www.python.org/static/img/python-logo.png
```

## 학습정리

## 1. BeautifulSoup 라이브러리 알아보기



- BeautifulSoup : HTML 및 XML 파일에서 데이터를 추출하기 위한 파이썬 라이브러리
- BeautifulSoup() 함수 호출 시 HTML 문서에 대한 파싱이 끝나면 트리 구조 형식으로 DOM 객체들이 생성되며 bs4.BeautifulSoup 객체를 통해 접근 가능
- <html>, <head> 태그와 <body> 태그는 제외하고 접근하려는 태그에 대하여 계층 구조를 적용하여 태그명을 . 연산자와 함께 사용
- 태그명은 bs4.element.Tag 객체의 name 속성, 태그의 속성은 bs4.element.Tag 객체에 ['속성명'] 식을 적용하고 태그의 내용은 bs4.element.Tag 객체의 text 속성 사용

## 학습정리

## 2. BeautifulSoup 라이브러리 응용



- `bs4.BeautifulSoup` 객체의 `find_all()`은 주어진 기준에 맞는 모든 태그들을 찾아오며 결과는 `bs4.element.ResultSet` 객체로 리턴
- `bs4.BeautifulSoup` 객체의 `bs.find()`는 주어진 기준에 맞는 태그 한 개를 찾아오며 결과는 `bs4.element.Tag` 객체로 리턴하며 결과값이 없으면 `None`을 리턴
- `bs4.BeautifulSoup` 객체의 `bs.select()` 주어진 CSS 선택자에 맞는 모든 태그들을 찾아오며 결과는 `list` 객체로 리턴