3. 정적 크롤링

차례

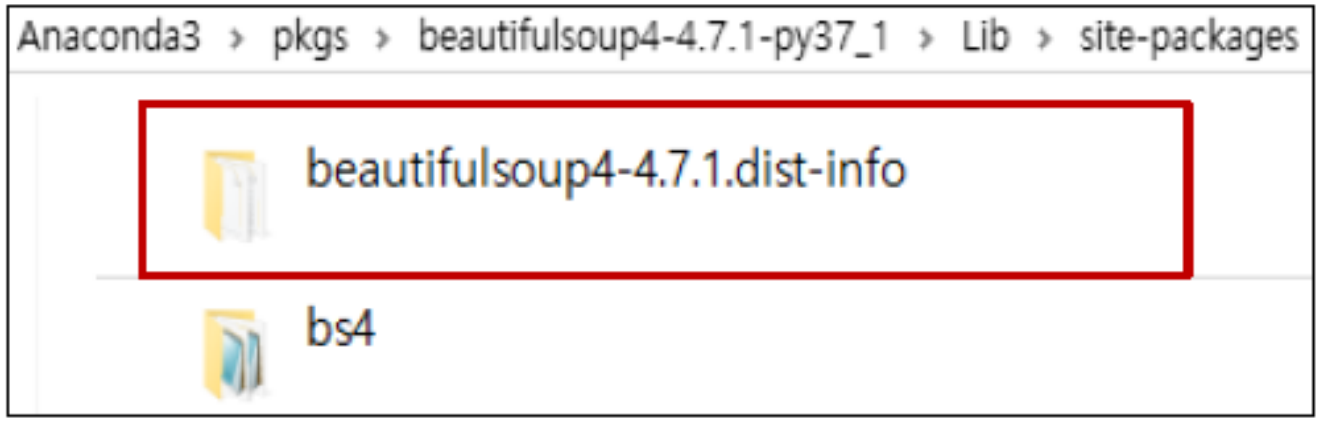
- HTML 파싱
  - BeautifulSoup 라이브러리
  - BeautifulSoup 활용
- 다양한 웹 사이트 웹 크롤링 실습
  - 영화 및 서점 웹페이지 크롤링과 스크레핑
  - 출판 네트워크, 기상청 및 클라이언트 정보 변경 크롤링과 스크래핑

1. BeautifulSoup 라이브러리

- BeautifulSoup

BeautifulSoup	▪ HTML 및 XML 파일에서 데이터를 추출하기 위한 파이썬 라이브러리

  - 파이썬 기본적으로 제공하는 라이브러리가 아니므로 별도 설치 필요
  - Anaconda는 BeautifulSoup 패키지가 Site-packages로 설치되어 있음



1. BeautifulSoup 라이브러리

- BeautifulSoup
  - 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

1. BeautifulSoup 라이브러리

- HTML 파싱

1. BeautifulSoup의 메인 패키지인 bs 패키지에서 BeautifulSoup() 함수 임포트

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

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

```
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')
```

1. BeautifulSoup 라이브러리

- 파서 라이브러리(Parser Library)비교

파서	구현 방법	특징
Python's html.parser	BeautifulSoup (markup, "html.parser")	• 추가 설치 필요 없음 • 적당한 속도
lxml's HTML parser	BeautifulSoup (markup, "lxml")	• 추가 설치 필요 • 속도가 빠름
lxml's XML parser	BeautifulSoup (markup, "lxml-xml") BeautifulSoup (markup, "xml")	• 속도가 빠름 • 추가 설치 필요
html5lib	BeautifulSoup (markup, "html5lib")	• 속도가 느림 • 추가 설치 필요 • 웹 브라우저와 동일한 방식으로 페이지의 파싱 지원

1. BeautifulSoup 라이브러리

- 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 계층구조를 적용
  - 태그명을. 연산자와 함께 사용

```
bs.태그명
bs.태그명.태그명
bs.태그명.태그명.태그명
bs.태그명.태그명.태그명.태그명
```

1. BeautifulSoup 라이브러리

- 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
```

1. BeautifulSoup 라이브러리

- 태그의정보추출
  - bs4.element.Tag객체의 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)

1. BeautifulSoup 라이브러리

- 태그로부터 다른 태그로 이동

부모 태그 추출

bs.태그명.parent

자식 태그들 추출

bs.태그명.children

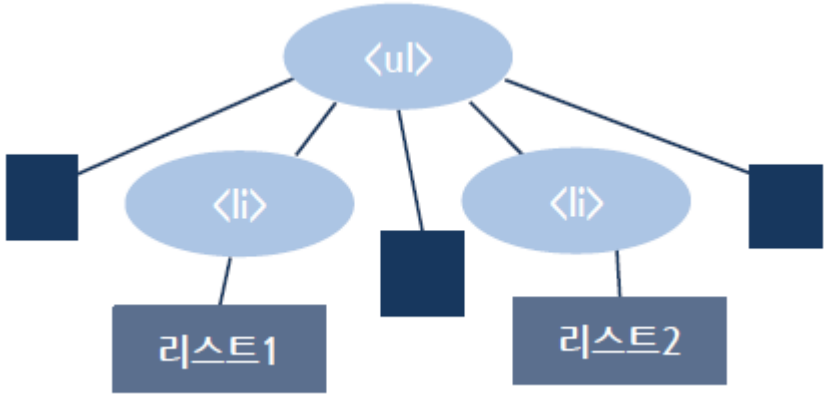
자손 태그들 추출

bs.태그명.descendants

형제 태그 추출

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

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



1. BeautifulSoup 라이브러리 – 실습1

```
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')
bs1=BeautifulSoup(html_doc, 'lxml')
bs2=BeautifulSoup(html_doc, 'lxml-xml')
bs3=BeautifulSoup(html_doc, 'html5lib')
print(type(bs))
print(bs)
print(bs1)
print(bs2)
print(bs3)
```

1. BeautifulSoup 라이브러리 – 실습2

```
html_doc = """
<!DOCTYPE html>
<html>
    <head>
        <meta charset='utf-8'>
        <title>Test BeautifulSoup</title>
    </head>
    <body>
        <p align="center">P태그의 컨텐트</p>
        <img src="http://unico2013.dothome.co.kr/image/flower.jpg" width="300">
    </body>
</html> """
from bs4 import BeautifulSoup
bs = BeautifulSoup(html_doc, 'html.parser')
print(bs.prettify())
```

1. BeautifulSoup 라이브러리 – 실습3

```
html_doc = """
<!DOCTYPE html>
<html>
    <head>
        <meta charset='utf-8'>
        <title>Test BeautifulSoup</title>
    </head>
    <body>
        <p align="center">P태그의 컨텐트</p>
        <img src="http://unico2013.dothome.co.kr/image/flower.jpg" width="300">
        <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)
```

1. BeautifulSoup 라이브러리 – 실습4

```
html_doc = """
<!DOCTYPE html>
<html>
    <head>
        <meta charset='utf-8'>
        <title>Test BeautifulSoup</title>
    </head>
    <body>
       <p align="center">P태그의 컨텐트</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)

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)

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

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

1. BeautifulSoup 라이브러리 – 실습5

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html=urlopen("http://stackoverflow.com")
bs=BeautifulSoup(html.read(), 'html.parser')
print(bs.title)
print(bs.title.text)
print(bs.h1)
print(bs.h1.text)
print(bs.span)
```

1. BeautifulSoup 라이브러리 – 실습6

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html="'
<html><body>
<h1>Hello python</h1>
<p>웹 페이지 분석</p>
<p>웹 스크래핑</p>
</body></html>
"'
```

```
soup=BeautifulSoup(html,'html.parser')
h1=soup.html.body.h1
p1=soup.html.body.p

p2=p1.next_sibling.next_sibling
print("h1=",h1)
print("h1=",h1.string)
print("p=",p1)
print("p=",p1.string)
print("p=",p2)
print("p=",p2.string)
```

2. BeautifulSoup 라이브러리 응용

- bs4.BeautifulSoup 객체의 주요 메서드
  - HTML 문서에 대한 파싱이 끝나고 생성된 트리구조 형식의 DOM 객체
    - bs4.BeautifulSoup 객체의속성으로접근가능
  - 다음에 제시된 메서드로도 가능

태그 찾기 기능의 주요 메서드

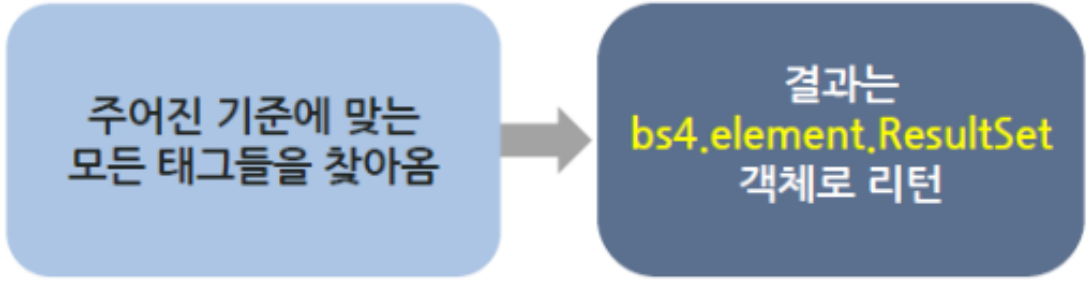
- find_all()
- find()
- select()

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()

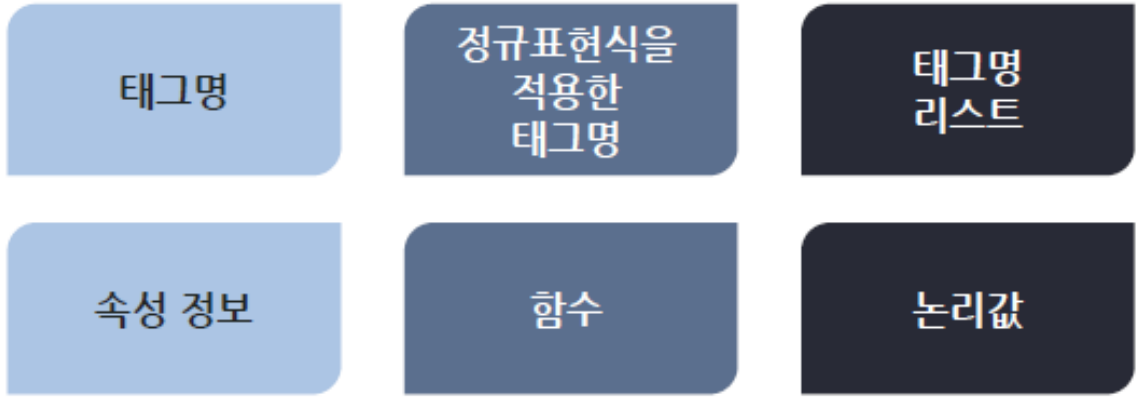
2. BeautifulSoup 라이브러리 응용

- 메서드를 사용한 웹페이지 파싱 : bs.find_all()




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

- 기준 설정



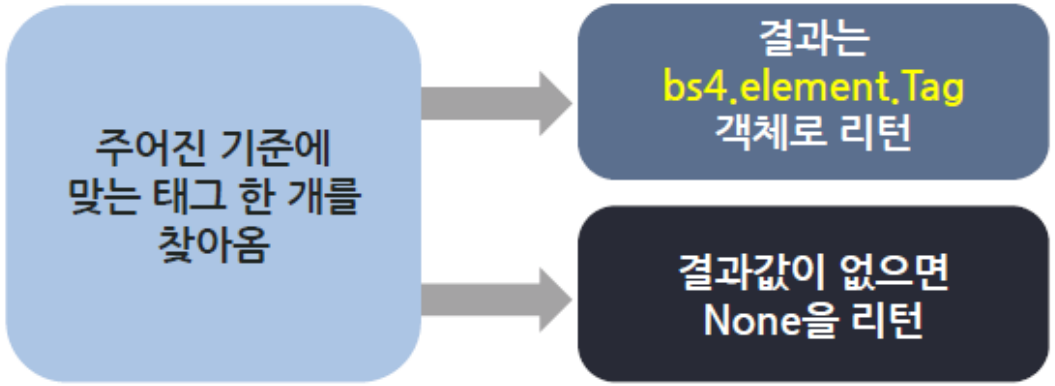
2. BeautifulSoup 라이브러리 응용

- 메서드를 사용한 웹페이지 파싱 : bs.find_all()

```
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)
```

2. BeautifulSoup 라이브러리 응용

- 메서드를 사용한 웹페이지 파싱 : bs.find()

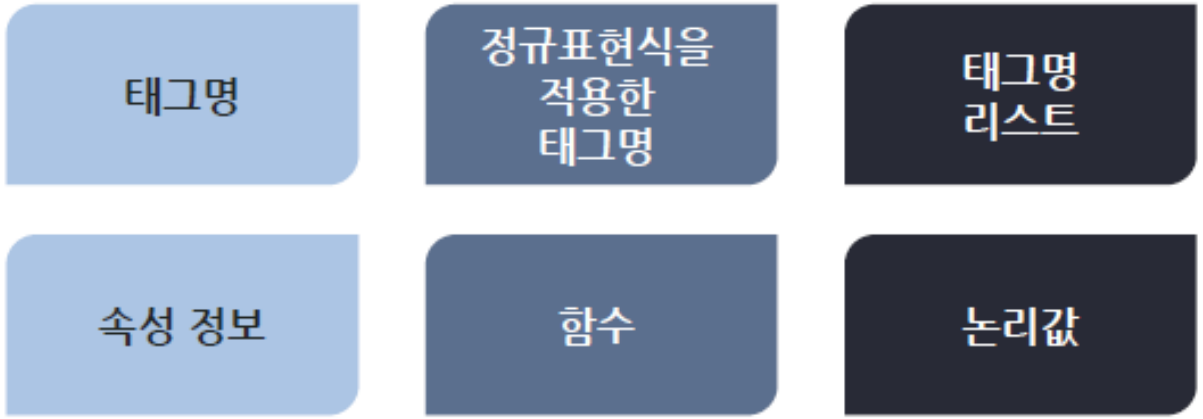


find(name=None, attrs={}, recursive=True, text=None, **kwargs)

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

2. BeautifulSoup 라이브러리 응용

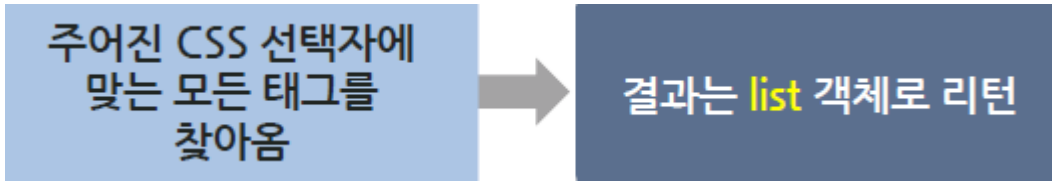
- 메서드를 사용한 웹페이지 파싱 : bs.find()
  - 기준 설정




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

2. BeautifulSoup 라이브러리 응용

- 메서드를 사용한 웹페이지 파싱 : bs.select()



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

- CSS 선택자를 적용한 호출

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

▪ 메서드를 사용한 웹페이지 파싱 : 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)')
```

bs.find(), bs.find_all() 실습1

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html="'
<html><body>
<h1 id="title">Hello python</h1>
<p id="body">웹 페이지 분석</p>
<p>웹 스크래핑</p>
<span>데이터 수집1</span>
<span>데이터 수집2</span>
</body></html>
"'
```

```
soup=BeautifulSoup(html,'html.parser')
title=soup.find(id='title')
body=soup.find(id='body')
span=soup.find('span')

print('#title=',title)
print('#title='+title.string)
print('#body=',body)
print('#body=',body.string)
print('span=',span.string)
```

bs.find(), bs.find_all() 실습2

```
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>
    <img src="http://unico2013.dothome.co.kr/image/flower.jpg" width="500">
    <div>
      <p>text contents 4</p>
    </div>
  </body>
</html> """
```

```
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)
```

bs.find(), bs.find_all() 실습3

```
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>
    <img src="http://unico2013.dothome.co.kr/image/flower.jpg"
width="500">
  </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"}))
```

bs.find_all() 실습4

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html='''
<html><body>
<ul>
    <li><a href="http://naver.com">naver</a></li>
    <li><a href="http://daum.net">daum</a></li>
    <li><a href="http://nate.com">nate</a></li>
    <li><a href="http://google.com">google</a></li>
    <li><a href="http://yahoo.com">yahoo</a></li>
</ul>
</body></html>
'''
```

```
soup=BeautifulSoup(html,'html.parser')
links=soup.find_all("a")
print(links)
print(type(links))
for a in links:
    href=a.attrs['href']
    text=a.string
    print(text, ":", href)
```

bs.find_all() 실습4

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import urllib.request as req
url="http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
res=req.urlopen(url)
soup=BeautifulSoup(res,'html.parser')
title=soup.find("title").string
wf=soup.find("wf").string
print(title)
print(wf)
```

```
서울
['A02', '2021-08-06', '00:00', '구름많음', '26', '33', '40']
['A02', '2021-08-06', '12:00', '흐리고', '비', '26', '33', '70']
['A02', '2021-08-07', '00:00', '흐리고', '비', '26', '31', '60']
['A02', '2021-08-07', '12:00', '흐리고', '비', '26', '31', '80']
['A02', '2021-08-08', '00:00', '흐림', '25', '31', '40']
['A02', '2021-08-08', '12:00', '구름많음', '25', '31', '40']
['A02', '2021-08-09', '00:00', '구름많음', '25', '32', '40']
['A02', '2021-08-09', '12:00', '구름많음', '25', '32', '40']
['A02', '2021-08-10', '00:00', '구름많음', '25', '32', '40']
['A02', '2021-08-10', '12:00', '구름많음', '25', '32', '40']
['A01', '2021-08-11', '00:00', '흐림', '25', '31', '40']
['A01', '2021-08-12', '00:00', '흐림', '25', '31', '40']
['A01', '2021-08-13', '00:00', '흐림', '26', '31', '40']
인천
['A02', '2021-08-06', '00:00', '구름많음', '27', '32', '40']
['A02', '2021-08-06', '12:00', '흐리고', '비', '27', '32', '70']
['A02', '2021-08-07', '00:00', '흐리고', '비', '26', '30', '60']
['A02', '2021-08-07', '12:00', '흐리고', '비', '26', '30', '80']
['A02', '2021-08-08', '00:00', '흐림', '25', '31', '40']
```

bs.select() 실습 1

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html="'
<html><body>
<h1>테스트</h1>
<div>div1</div>
<div>div2</div>
<div id="main">
<h1 i>도서 목록</h1>
<ul class="items">
  <li>자바 프로그래밍 입문</li>
  <li>HTML5</li>
  <li>Python</li>
</ul>
</div>
</body></html>"'
```

```
soup=BeautifulSoup(html,'html.parser')
h1=soup.select_one("div#main > h1").string
print("h1=",h1)
li_list=soup.select("div#main > ul.items > li")
for li in li_list:
    print("li=",li.string)
```

bs.select() 실습 2

```
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 갯수 : %d " %len(title))
print("f_subtitle이라는 id 속성을 갖는 태그 갯수 : %d " %len(title1))
print("subtitle이라는 class 속성을 갖는 태그 갯수 : %d " %len(title2))
print("aside 태그의 <h2> 자식 태그 갯수 : %d " %len(title3))
print("src 속성을 갖는 태그 갯수 : %d " %len(img))
```

```
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"])
```

bs.select() 실습 2

```
import urllib.request
from bs4 import BeautifulSoup
url = "http://unico2013.dothome.co.kr/crawling/exercise_bs.html"
html = urllib.request.urlopen(url)
bs = BeautifulSoup(html, "html.parser")
print('[<h1> 태그의 콘텐츠] ', bs.select('h1')[0].text)
print('[텍스트 형식으로 내용을 가지고 있는 <a> 태그의 콘텐츠와 href 속성값] ',)
aTag = bs.select('a')
for tag in aTag:
    if(tag.text.strip()):
        print(tag.text, ' : ', tag['href'])
print('[<img> 태그의 src 속성값] ',bs.select('img')[0]['src'])
print('[첫 번째 <h2> 태그의 콘텐츠] ',bs.select('h2:nth-of-type(1)')[0].text)
print('[<ul> 태그의 자식 태그들 중 style 속성의 값이 green으로 끝나는 태그의 콘텐츠] ',
    bs.select('ul > li[style$=green]')[0].text)
print('[두 번째 <h2> 태그의 콘텐츠] ',bs.select('h2:nth-of-type(2)')[0].text)
print('[<ul> 태그의 모든 자식 태그들의 콘텐츠 ]')
olliTag = bs.select('ol > li')
for tag in olliTag:
    print(tag.text)
print('[<table> 태그의 모든 자손 태그들의 콘텐츠 ]')
print(bs.select('table')[0].text.strip())
print('[name이라는 클래스 속성을 갖는 <tr> 태그의 콘텐츠] ',bs.select('tr.name')[0].text)
print('[target이라는 아이디 속성을 갖는 <td> 태그의 콘텐츠] ',bs.select('td#target')[0].text)
```

웹스크랩 실습-1

```
# 윤동주의 작품 목록 스크랩

from bs4 import BeautifulSoup
import urllib.request as req

# 뒤의 인코딩 부분은 "저자 : 윤동주"라는 의미입니다.
# 위키 문헌 홈페이지에 들어간 뒤에서 주소를 복사해서 사용

url="https://ko.wikisource.org/wiki/%EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC"
res=req.urlopen(url)
soup=BeautifulSoup(res,"html.parser")
li_list=soup.select(".mw-parser-output > ul > li")
#print(li_list)
for title in li_list:
    print(title.a.string)
    li=title.find_all('li')
    for i in li:
        print(" -",i.a.string)
```

웹스크랩 실습-2

```
#war_and Peace
from bs4 import BeautifulSoup
from urllib.request import urlopen
html=urlopen("http://www.pythonscraping.com/pages/warandpeace.html")
bs=BeautifulSoup(html,'html.parser')

#nameList=bs.select('span.green')
nameList=bs.findAll('span',{'class':'green'})

for name in nameList:
    print(name.get_text())
```

```
titles=bs.find_all(['h1','h2','h3','h4','h5','h6'])
print([title for title in titles])
```

table 데이터 스크랩

```
from bs4 import BeautifulSoup
from urllib.request import urlopen
html=urlopen("http://www.pythonscraping.com/pages/page3.html")
bs=BeautifulSoup(html,'html.parser')

for child in bs.find('table',{'id':'giftList'}).children: # 하위 태그들으 모두 가져옴
    print(child)
```

```
# 시블링(sibling, 형제자매)
# next_sibling : 다음 형제 노드
# previous_sibling: 이전 형제 노드
# 제목행은 제외하고 탐색하게 됨
for sibling in bs.find('table',{'id':'giftList'}).tr.next_siblings:
    print(sibling)
```

```
img_urls=bs.find_all('img')
print(img_urls)
print(bs.find('img',{'src':'../img/gifts/img1.jpg'}) # 이미지가 있는 td 구함
      .parent.previous_sibling.get_text())
```

pdf 파일 스크랩-1

```
# pip install pdfminer3k
# pdf 문서 읽기

from pdfminer.pdfinterp import PDFResourceManager,process_pdf
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from io import StringIO  #pdf 내부의 텍스트 입출력을 위한 객체
from io import open
from urllib.request import urlopen
```

pdf 파일 스크랩-2

```
def readPDF(pdfFile):
    rsrcmgr=PDFResourceManager() # pdf 리소스 관리
    retstr=StringIO() # pdf 내부의 텍스트 입출력을 위한 객체
    laparams=LAParams() # 파라미터 객체
    # pdf 내용을 텍스트로 반환하기 위한 객체
    device=TextConverter(rsrcmgr,retstr,laparams=laparams)
    process_pdf(rsrcmgr,device,pdfFile) #pdf 내용을 텍스트로 변환하는 작
    device.close()
    content=retstr.getvalue() # 리턴되는 스트링
    retstr.close()
    return content

pdfFile=urlopen("http://www.pythonscraping.com/pages/warandpeace/chapter1.pdf")
outputString=readPDF(pdfFile)
#print(outputString)
pdfFile.close()

with open("d:/data/pdf/result.txt",'w') as f:
    f.write(outputString)
    print("저장되었습니다")
```

스크랩 내용 csv로 저장-1

```
# 웹페이지의 내용을 분석하여 CSV 파일로 저장
# <table> 내부의 텍스트 저장
import csv
from urllib.request import urlopen
from bs4 import BeautifulSoup

html=urlopen("https://en.wikipedia.org/wiki/Comparison_of_text_editors")
bsObj=BeautifulSoup(html,'html.parser')

# class가 wiktable인 테이블들 중 첫 번째 태그 선택
table=bsObj.findAll("table",{"class":"wikitable"})[0]
rows=table.findAll("tr")
#print(rows)
```

스크랩 내용 csv로 저장-2

```
# wt: 텍스트 쓰기 모
csvFile=open("d:/data/csv/editors.csv","wt",newline='\n',encoding='utf-8')
# csv 파일 저장 객체
writer=csv.writer(csvFile)
try:
    for row in rows:
        csvRow=[]
        # td, th 태그의 내용을 리스트에 추가
        for cell in row.findAll(['td','th']):
            # td, th 태그내의 문자열 추출
            csvRow.append(cell.get_text()) # cell.get_text().strip("\n")
        #print(csvRow)
        writer.writerow(csvRow)
finally:
    print("csv로 저장되었습니다")
    csvFile.close()
```

웹 스크랩 내용 mysql에 저장-1

```
# mysql Table 만들기

create table movie_review(
id int auto_increment primary key,
title varchar(200),
point int,
review text);
```

웹 스크랩 내용 mysql에 저장-2

```
# 데이터 수집
from urllib.request import urlopen
from bs4 import BeautifulSoup as bs
# 네이버 영화 평점 수집
def getLinks(page):
    html=urlopen("https://movie.naver.com/movie/point/af/list.naver?&page="+str(page))
    print(html)
    obj=bs(html,'html.parser')
    all_text=obj.select('.title')
    data_list=[]
    for s1 in all_text:
        text=s1.get_text()
        textarr=text.replace('신고','').replace('\t','').replace('\n\n\n\n','').split("\n\n")
        textarr[0]=textarr[0].strip('\n')
        textarr[1]=int(textarr[1][textarr[1].find('중')+1:])
        if len(textarr)<3:
            textarr.append('-')
        data_list.append(tuple(textarr))

    return data_list
```

웹 스크랩 내용 mysql에 저장-3

```
# csv 파일로 저장
import numpy as np
import pandas as pd

def csv_save(data_total):
    df=pd.DataFrame(data_total, columns=['titll', 'point','review'])
    df.to_csv("d:/data/movie_revie.csv", index=False)
```

웹 스크랩 내용 mysql에 저장-4

```
# mysql db에 저장
import pymysql
def mysql_save(data_total):
    conn = pymysql.connect(host='localhost',
                           user='pgm',
                           password='1234',
                           db='pydb',
                           charset='utf8')
    cursor = conn.cursor()
    sql='insert into movie_review(title,point,review) values(%s, %s, %s)'
    cursor.executemany(sql, data_total)
    conn.commit()
    conn.close()
```

웹 스크랩 내용 mysql에 저장-5

```
data_total=[]
for i in range(1,11):
    data_total.extend(getLinks(i))

print(data_total)
mysql_save(data_total)
csv_save(data_total)
```

연습문제 1

- 기상청 웹페이지의 기상청 육상 중기 예보

https://www.weather.go.kr/weather/forecast/mid-term-rss3.jsp

- **도시명**
- **날짜:시간(tmEf),**
- **최저온도(tmn)**
- **최고 온도(tmx)**
- **강수확률(rnSt)**
  - **크롤링하여 데이터베이스에 저장 및 csv파일로 저장하기**

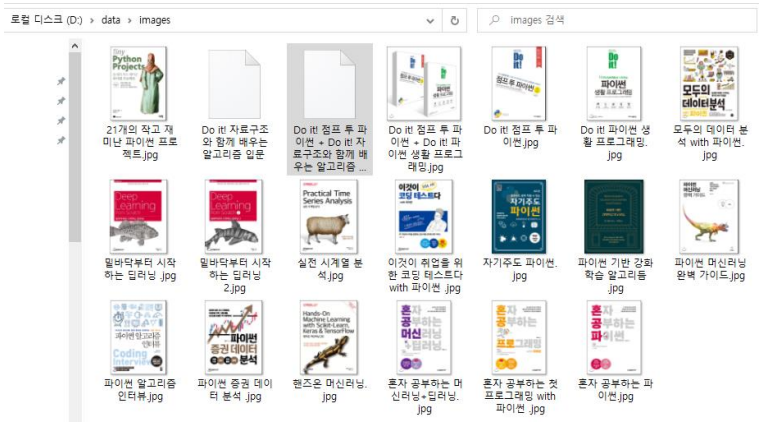
연습문제 2

- yes24에서 파이썬 관련 서적 정보 스크래핑
  - 크롤링 URL

http://www.y***4.com/SearchCorner/Search?domain=BOOK&query=python

- 스크래핑 내용
  - 책표지 이미지는 ~/data/images폴더에 저장하고
  - 제목, 저자, 출판사, 출판연도, 가격 => Pandas의 DataFrame을 작성하여 출력

	title	author	publisher	year	price
0	Do it! 점프 투 파이썬	박응용 저	이지스퍼블리싱	2019년 06월	16920
1	혼자 공부하는 파이썬	윤인성 저	한빛미디어	2019년 06월	16200
2	혼자 공부하는 머신러닝+딥러닝	박해선 저	한빛미디어	2020년 12월	23400
3	이것이 취업을 위한 코딩 테스트다 with 파이썬	나동빈 저	한빛미디어	2020년 08월	30600
4	모두의 데이터 분석 with 파이썬	송석리,이현아 저	길벗	2019년 04월	16200
5	파이썬 알고리즘 인터뷰	박상길 저/정진호 그림	책만	2020년 07월	34200
6	밑바닥부터 시작하는 딥러닝	사이토 고키 저/개앞맵시 역	한빛미디어	2017년 01월	21600
7	핸즈온 머신러닝	오렐리앙 제롱 저/박해선 역	한빛미디어	2020년 05월	49500
8	파이썬 머신러닝 완벽 가이드	권철민 저	위키북스	2020년 02월	34200
9	Do it! 자료구조와 함께 배우는 알고리즘 입문 : 파이썬 편	시바타 보요 저/강민 역	이지스퍼블리싱	2020년 07월	19800
10	Do it! 파이썬 생활 프로그래밍	김창현 저	이지스퍼블리싱	2020년 07월	18000
11	혼자 공부하는 첫 프로그래밍 with 파이썬	문현일 저	한빛미디어	2020년 06월	15300
12	Do it! 점프 투 파이썬 + Do it! 자료구조와 함께 배우는 알고리즘 입문 ...	박응용,김창현,시바타 보요 저/강민 역	이지스퍼블리싱	0001년 01월	54720
13	Do it! 점프 투 파이썬 + Do it! 파이썬 생활 프로그래밍	박응용,김창현 저	이지스퍼블리싱	0001년 01월	34920
14	파이썬 증권 데이터 분석	김황후 저	한빛미디어	2020년 07월	28800
15	파이썬 기반 강화학습 알고리듬	안드레아 론자 저/정사범 역	에이콘출판사	2021년 08월	27000
16	21개의 작고 재미난 파이썬 프로젝트	켄 유엔스-클락 저/김완섭 역	제이펍	2021년 08월	27000



과제- 네이버 영화페이지

- 상영작.예정작 영화정보 스크랩하기
  - 영화제목
  - 영화관람 등급
  - 네티즌평점,
  - 장르
  - 러닝타임
  - 개봉일
  - 감독
  - 배우
  - 줄거리
  - 성별. 나이별 관람 추이
  - 영화포스트 이미지
- Mysql DB에 크롤링 내용 저장하기