HTML, XML 파싱
BeautifulSoup

BeautifulSoup

- HTML 과 XML 파일을 파싱하는 라이브러리
- 설치하기

  python –m pip install beaurifulsoup4

- Import 하기

  from **bs4** import BeautifulSoup

- 참고 사이트

http://www.crummy.com/software/BeautifulSoup/bs4/doc/

BeautifulSoup

```
html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their
names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""
```

soup = BeautifulSoup(html_doc, "html.parser")

print(**soup.prettify()**)
print(**soup.a.prettify()**)

prettyify() : Parse Tree 형태로 출력

BeautifulSoup

- 태그 이름을 변수이름으로 사용할 수 있음

soup.html.head.title

- 계층구조의 중간단계를 생략할 수 있음

soup.title

- 태그안에 다른 태그가 없을 경우 string 속성으로 태그 내용을 얻을 수 있음

soup.title.string

- 같은 이름의 태그가 여러 개 있다면 제일 먼저 나오는 태그를 알려줌

soup.p

soup.p['class']

BeautifulSoup

- 모든 태그 목록을 반환함

soup(‘a’)
soup(‘a’)[0]

soup('a',{'class' : 'sister'})
Soup(‘img’,{‘name’:’main’})
Soup(‘p’,{‘class’:’layout’})

- 계층 구조 속성
  - Parent : 부모
  - Contens : 한칸 아래
  - NextSibling : 같은 위치의 바로 앞
  - previousSbling : 같은 위치의 바로 뒤
  - Next/ previous : 계층구조와 무관하게 바로 뒤, 앞에 있는 태그

soup(‘a’)[0].parent
soup(‘a’)[0].contents

soup(‘a’)[0].parent.name

BeautifulSoup

- find(str)
  - 처음으로 나오는 해당 태그 객체를 반환

```
soup.find('p')
soup.find(‘a’)[‘href’]
soup.find_all(id='link3')
```

- find_all(str) / findAll(str)
  - 전체 문서에서 태그를 검색하여 반환

```
soup.find_all('p')
soup.find_all(class_='sister')
```

BeautifulSoup

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

url = "http://comic.naver.com/webtoon/list.nhn?titleId=20853"

data = urlopen(url)
soup = BeautifulSoup(data, 'html.parser')

cartoons = soup.find_all('td', {'class' : 'title'})

for i in range(len(cartoons)):
    title = cartoons[i].find('a').string
    ref = cartoons[i].find('a')['href']
    tempurl = urljoin(url, ref)
    print(title, " " , tempurl)
```

webbrowser.open_new(tempurl)

WebCrawler

```
class crawler:
    def crawl(self, pages, depth=2):
        for i in range(depth):
            newpage = set()
            for page in pages:
                try:
                    c = urllib.request.urlopen(page)
                except:
                    print("Could not open %s" % page)
                    continue
                soup = BeautifulSoup(c.read(), from_encoding="utf-8")
                print('Found %s' % page)
            links = soup('a')
```

WebCrawler

```
        for link in links:
            if('href' in dict(link.attrs)):
                url = urllib.parse.urljoin(page, link['href'])
                if url.find("'")!=-1 : continue
                url = url.split("#")[0]
                if url[0:4]=='http':
                    newpage.add(url)
        pages = newpage

pagelist=['http://www.naver.com']
crawler=crawler()
crawler.crawl(pagelist)
```

XML 파싱

```
<nodes>
  <node attr1='a'> Node1 </node>
  <node attr1='b'> Node2 </node>
</nodes>
```

test.xml

```
from bs4 import BeautifulSoup

f = open('test.xml')
xml = f.read()
soup = BeautifulSoup(xml)
for node in soup.findAll('node'):
    print("Node : "+node.string)
    print("Attr1 : "+node['attr1'])
```

XML 파싱

```
<?xml version="1.0" encoding="utf-8"?>
<test>
  <song>
    <title>제발</title>
    <length>4:54</length>
  </song>
  <song>
    <title>청혼</title>
    <length>3:21</length>
  </song>
</test>
```

song.xml

```
f = open('song.xml', encoding='utf-8')
xml = f.read()
soup = BeautifulSoup(xml)
for nodes in soup.test('song'):
    for node in nodes:
        print(node.string)
```

XML 파싱

alcohol.xml

```
<?xml version='1.0' encoding='utf-8'?>
<alcohol>
  <cate1 tt="술">
    <cate2 tt="소주">
      <item>참이슬</item>
      <item>처음처럼</item>
      <item>잎새주</item>
    </cate2>
    <cate2 tt='맥주'>
      <item>카스</item>
      <item>라거</item>
      <item>하이트</item>
    </cate2>
  </cate1>
  <cate1 tt="안주">
    <cate2 tt="고가">
      <item>회</item>
      <item>등심</item>
      <item>양곱창</item>
    </cate2>
    <cate2 tt="저가">
      <item>참치캔</item>
      <item>날계란</item>
      <item>새우깡</item>
    </cate2>
  </cate1>
</alcohol>
```

```
f = open('alcohol.xml', encoding='utf-8')
xml = f.read()
soup = BeautifulSoup(xml,'html.parser')
for nodes in soup.alcohol('cate1'):
    print('Cate1 :' + nodes['tt'])
    for node in nodes('cate2'):
        print('₩tCate2 :' + node['tt'])
        for item in node('item'):
            print('₩t₩t' + item.string)
```

XML 파싱

- lxml 파서
  - 공식적인 3.4버전 window용 파서는 없음
  - http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml
    - 해당 파일을 다운로드 받음

python –m pip install **lxml-3.4.4-cp34-none-win_amd64.whl**

```
* 파서 지정할 수 있음
soup = BeautifulSoup(xml,'lxml')
```

Json 파싱

- 데이터를 교환할 때 사용할 수 있는 자료 표현방법
- dumps()
  - 파이썬 데이터를 Json 데이터 형식으로 변환
- loads()
  - Json 데이터를 python 데이터로 변환

```
import json
data = {1:'a',2:'b'}
data2 = json.dumps(data)
data3 = json.loads(data2)
print(type(data2))
print(type(data3))
```

```
data = {1:'우리',2:'나라'}
data2 = json.dumps(data, ensure_ascii=False)
print(data2)
```

Json 파싱

```
import json
s = """
 {
 "name": "cybaek",
 "detail" : { "last": "baek" },
 "emails": [ "cybaek@xxx.com", "cybaek@yyy.com" ]
 }
 """
data = json.loads(s)

print(data['name'])
print(data['detail'])
print(data['detail']['last'])
```

Json 파싱

```
import json
s = """
 {
 "name": "cybaek",
 "detail" : { "last": "baek" },
 "emails": [ "cybaek@xxx.com", "cybaek@yyy.com" ]
 }
 """
```

```
class JsonObject:
  def __init__(self, d):
    self.__dict__ = d
data = json.loads(s, object_hook=JsonObject)
print(data.name)
print(data.detail)
print(data.detail.last)
for email in data.emails:
  print(email)
```