UNISTUDY

파이썬

34강. 웹문서 수집과 처리

1. 관련 용어

- 웹 문서는 여러 개의 약속된 태그(Tag)를 이용하여 자료를 만들고, 웹 브라우저를 통해서 볼 수 있는 문서들을 의미한다. 따라서 해당 웹 문서를 수집하기 위해서는 URL을 통해서 웹 문서를 소스(source) 형태로 수집하고, html 문서로 바꾸는 과정이 필요하다. 이절에서는 웹 문서 수집과 관련된 용어들을 살펴본다.

2. 크롤링

- 크롤링(Crawling)이란 웹을 탐색하는 컴퓨터 프로그램(크롤러)을 이용하여 여러 인터넷 사이트의 웹 페이지 자료를 수집해서 분류하는 과정을 말한다. 크롤러(crawler)란 자동화된 방법으로 월드와이드 웹(www)을 탐색하는 컴퓨터 프로그램을 의미한다.

3. 스크래핑

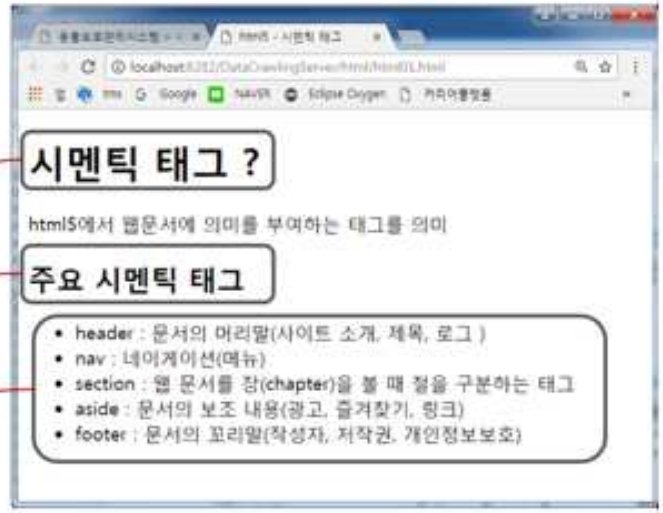
- 스크래핑(Scraping)이란 웹 사이트의 내용을 긁어다 원하는 형태로 가공하는 기술을 의미한다. 즉 웹 사이트의 데이터를 수집하는 모든 작업을 의미한다. 결국 크롤링도 스크래핑 기술의 일종이라고 할 수 있다.

4. 파싱

- 파싱(parsing)이란 어떤 페이지(문서, html 등)에서 내가 원하는 데이터를 특정 패턴이나 순서로 추출 하여 정보를 가공하는 것을 말한다. 예를 들면 html 소스를 문자열로 수집한 후 실제 html 태그로 인 식할 수 있도록 문자열을 의미 있는 단위로 분해하고, 계층적인 트리 구조를 만드는 과정을 말한다.

4. 파싱

```
import urllib.request
from bs4 import BeautifulSoup

url = 'http://localhost:8282/DataCrawlingServer/html/html01.html'

# 1. html source 가져오기
res = urllib.request.urlopen(url) # web 문서 get
data = res.read() # binary 형태로 읽음

# 2. html 파싱
html = data.decode("utf-8") # 디코딩
soup = BeautifulSoup(html, 'html.parser') # html source 파싱

# 3. 태그 내용 가져오기

# 1) 태그 <h1> 가져오기
h1 = soup.html.body.h1
print('h1 :', h1.string) # h1 : 시맨틱 태그 ?

# 2) find()함수로 찾기
h2 = soup.find("h2")
print("h2 :", h2.string) # h2 : 주요 시맨틱 태그

li = soup.find("li")
print(li.string) # header : 문서의 머리말(사이트 소개, 제목, 로고 )

# 2) find_all() 함수로 여러 개 찾기 : list 반환
li2 = soup.find_all("li")
print(li2) # [<li> header : 문서의 머리말(사이트 소개, 제목, 로고 )</li>, ""]
# print(li2.string) # error 발생

for li in li2 :
    print(li.string)
```



5. 태그 자료 수집

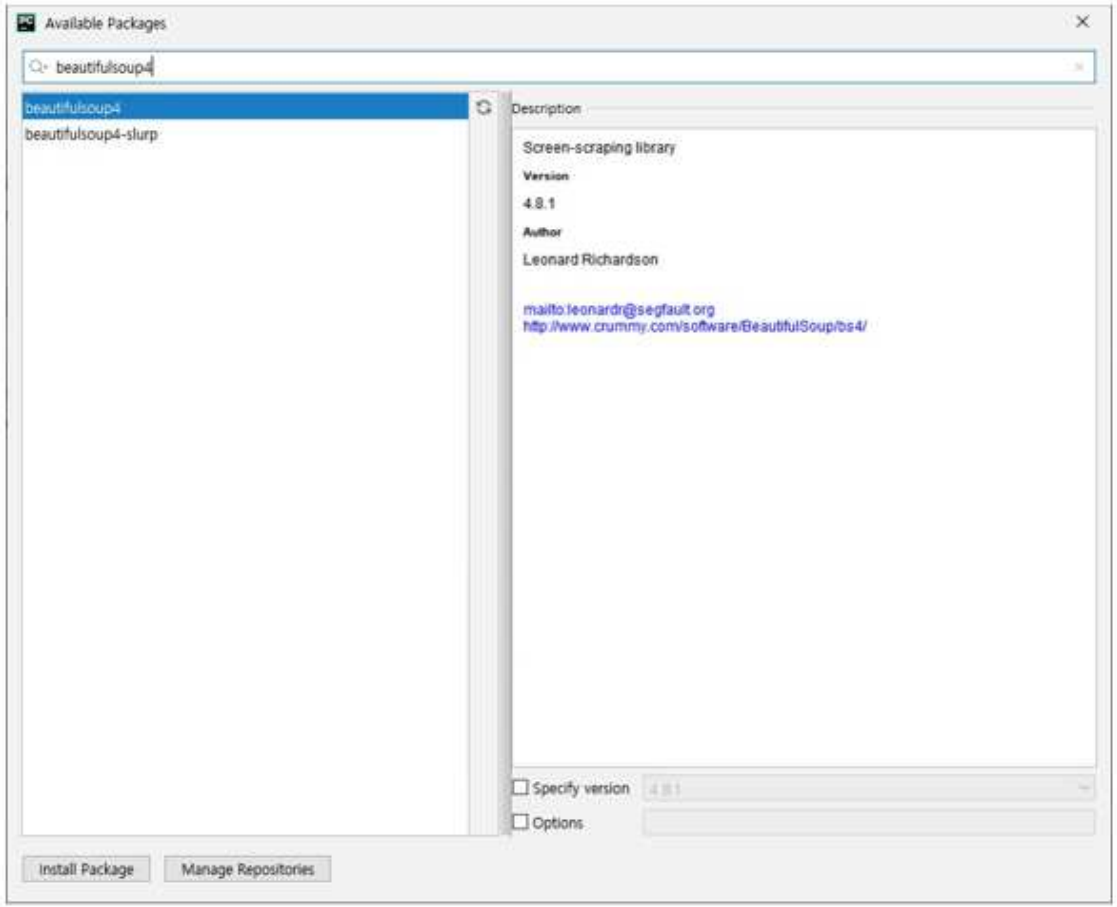
- 웹 문서를 수집할 url를 대상으로 원격 서버의 파일 자료를 요청하여 텍스트 자료를 수집하고, 수집한 자료를 html 양식으로 파싱한 후 태그가 감싸고 있는 자료를 수집하는 절차에 대해서 알아본다. html 은 웹 문서를 만들기 위해서 태그(Tag)를 이용한다. 다음은 웹 문서를 꾸미는데 사용되는 태그의 형식 이다. <시작태그>는 태그 시작을 나타내고, 속성을 이용하여 태그의 특징을 지정할 수 있다. 내용은 웹 브라우저에서 보이는 내용이다. 그리고 </종료태그>는 해당 태그가 끝남을 알린다.

```
    <시작태그  속성='값'> 내용 </종료태그>
예)
    <a src="http://www.test.com/index.html"> 자료 </a>
```

5. 태그 자료 수집

- 태그 형식을 <a> 태그로 나타낸 예문에서 <a> 태그의 시작태그, 속성과 내용 그리고 종료태그를 나타 내고 있다. 이렇게 <시작태그>와 </종료태그>로 구성된 명령어를 엘리먼트(Element)라고 한다.
- 결국 html 파일은 여러 개의 엘리먼트들이 모여서 만들어진다.
- 파이참에서 url을 통해서 수집한 html 문서를 파싱하기 위해서는 beautifulsoup4 패키지를 설치해야한다

5. 태그 자료 수집


Available Packages
beautifulsoup4
beautifulsoup4
beautifulsoup4-slurp
Description
Screen-scraping library
Version
4.8.1
Author
Leonard Richardson
mailto:leonardr@segfault.org
http://www.crummy.com/software/BeautifulSoup/bs4/
Specify version
Options
Install Package
Manage Repositories

6. 원격 서버에서 자료 수집

- BeautifulSoup 클래스에 의해서 파싱된 객체를 이용하여 html 문서에서 특정 태그를 찾는 형식은 다음과 같다. html 문서에서 한 개의 태그를 찾는 경우 find() 함수를 이용하고, 여러 개의 태그를 모두 찾을 경우에는 find_all()함수를 이용한다.

```
html파싱객체.find('태그명')
html파싱객체.find_all('태그명')
```

6. 원격 서버에서 자료 수집

- 다음 예문은 포털 사이트인 Naver의 시작페이지에서 가장 최초로 발견된 <a> 태그 한 개를 수집하여 내용을 출력하는 과정이다.

6. 원격 서버에서 자료 수집

chapter09.lecture.step01_url_request.py

```
# (1) request, BeautifulSoup 모듈 import
fimport urllib.request # 원격 서버 파일 요청
from bs4 import BeautifulSoup # html 파싱

# 요청할 url
url = 'http://www.naver.com/index.html'

# (2) 원격 서버 파일 요청
res = urllib.request.urlopen(url) # web 문서 요청
data = res.read() # text 형태로 읽음

# (3) source 디코딩
src = data.decode("utf-8") # 디코딩
print(src)


# (4) html 파싱
html = BeautifulSoup(src, 'html.parser') # html 파싱
print(html)


# (5) 태그 내용
a = html.find('a')
print('a tag : ', a)
print('a tag 내용 : ', a.string)
```

Python Console

```
a tag :  <a href="#news_cast" onclick="document.getElementById('news_cast2').tabIndex = -1;document.getElementById('news_cast2').focus();return false;"><span>연합뉴스 바로가기</span></a>
a tag 내용 :  연합뉴스 바로가기
```

6. 원격 서버에서 자료 수집

- 해설 원격 서버에서 수집 예
- # (1) request, BeautifulSoup 모듈 import
- 원격 서버의 파일을 요청하는 request와 html 파싱을 위해서 BeautifulSoup 모듈을 import해야 한다.
- # (2) 원격 서버 파일 요청
- request 모듈의 urlopen(url) 함수를 이용하여 url에 해당하는 원격서버의 웹 문서 파일의 요청 객체를 만들고 객체에서 제공하는 read() 함수를 통해서 텍스트 자료를 읽어온다.
- # (3) source 디코딩
- 한글과 같은 유니코드를 대상으로 디코딩하여 한글 자료를 깨지지 않게 한다. 해당 사이트의 소스 보기에서 디코딩 방식은 확인하는 화면이다.

6. 원격 서버에서 자료 수집

6. 원격 서버에서 자료 수집

- # (4) html 파싱
- html 태그로 인식할 수 있도록 문자열을 의미 있는 단위로 분해하고, 계층적인 트리 구조를 만드는 과정이다. 파싱 과정을 거쳐야 태그(tag)를 인식할 수 있다.
- # (5) 태그 내용
- html 파싱 객체는 find() 함수를 이용하여 특정 태그를 찾을 수 있다. 예문에서 find('a')은 현재 html 문서에서 최초로 발견된 <a> 태그를 찾아서 태그 엘리먼트를 반환한다. 그리고 반환 결과에 string 멤버를 붙여서 태그가 감싸고 있는 내용을 추출한다. a 태그가 감싸고 있는 내용은 '연합뉴스 바로가기' 이다.

7. 로컬에서 자료 수집

- 내 컴퓨터에 있는 html파일로 부터 자료를 수집하는 방법에 대해서 알아본다. html 파일도 텍스트 파 일처럼 줄 단위로 읽은 후 html 파싱을 통해서 원하는 태그를 찾아서 자료를 수집할 수 있다.
- 다음은 예문에서 사용될 html 파일의 내용이다. 자료를 수집할 태그는 <body> 태그에 포함된 <h1> 태그, <h2> 태그, <li> 태그이다. 여기서 <h1>과 <h2> 태그는 현재 문서에서 유일하게 한 개의 태 그만 사용되었고, <li> 태그는 현재 문서에서 5개의 태그를 사용하였다. 따라서 <li> 태그를 모두 수집하기 위해서는 find_all()함수를 이용해야한다.

7. 로컬에서 자료 수집

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> html5 - 시멘틱 태그 </title>
</head>
<body>
   <h1> 시멘틱 태그 ?</h1>
   <p> html5에서 웹문서에 의미를 부여하는 태그를 의미 </p>
   <h2> 주요 시멘틱 태그 </h2>
   <ul>
      <li> header : 문서의 머리말(사이트 소개, 제목, 로그 )</li>
      <li> nav : 네비게이션(메뉴) </li>
      <li> section : 웹 문서를 장(chapter)으로 볼 때 절을 구분하는 태그</li>
      <li> aside : 문서의 보조 내용(광고, 즐겨찾기, 링크) </li>
      <li> footer : 문서의 꼬리말(작성자, 저작권, 개인정보보호) </li>
   </ul>
</body>
</html>
```

7. 로컬에서 자료 수집

chapter09.lecture.step02_tag_name.py

```
# (1) 로컬 서버 파일 읽기
file = open('chapter09/data/html01.html',
mode='r', encoding='utf-8')
text = file.read()

# (2) html 파싱
html = BeautifulSoup(text, 'html.parser')

# (3) 태그 내용 가져오기

# (3-1) tag 이용
h1 = html.html.body.h1 # 계층 접근
print('h1 : ', h1.string) # h1 :  시멘틱 태그
?
```

Python Console

```
h1 :  시멘틱 태그 ?
```

7. 로컬에서 자료 수집

```
# (3-2) find('tag')함수
h2 = html.find('h2')
print('h2 : ', h2.string) # h2 :  주요 시멘틱 태그

# (3-3) find_all('tag') 함수
lis = html.find_all('li') # list 반환
print(lis)


# (4) li 태그 내용
for li in lis :
    print(li.string)
```

```
h2 :  주요 시멘틱 태그


[<li> header : 문서의 머리말(사이트 소개, 제목, 로그 )</li>, <li> nav : 네이게생략...
<li> footer : 문서의 꼬리말(작성자, 저작권, 개인정보보호) </li>]

 header : 문서의 머리말(사이트 소개, 제목, 로그 )
 nav : 네이게이션(메뉴)
 생략 ...
 footer : 문서의 꼬리말(작성자, 저작권, 개인정보보호)
```

7. 로컬에서 자료 수집

- 로컬에서 자료 수집 예
- # (1) 로컬 파일 읽기
- 해당 경로에서 html01.html 파일을 텍스트 방식으로 읽어온다.
- # (2) html 파싱
- html 태그로 인식할 수 있도록 문자열을 의미 있는 단위로 분해하고, 계층적인 트리 구조를 만드는 과정이다.
- # (3-1) tag 이용
- html 파싱 객체를 이용하여 <h1> 태그를 계층적으로 접근하여 <h1> 태그의 엘리먼트를 가져온다. 엘리먼트 객체의 string 멤버를 호출하면 <h1> 태그가 감싸고 있는 내용을 출력한다.

7. 로컬에서 자료 수집

- # (3-2) find('tag') 함수
- html 파싱 객체를 이용하여 find('tag')함수를 호출하면 해당 태그를 찾아서 태그의 엘리먼트를 반환한다. 예문에서는 <h2> 태그를 찾아서 태그가 감싸고 있는 내용을 출력한다. find() 함수는 최초로 발견된 태그를 찾아서 해당 엘리먼트 하나만 반환한다.
- # (3-3) find_all('tag') 함수
- html 파싱 객체를 이용하여 find_all('tag')함수를 호출하면 해당 태그들을 찾아서 태그들의 엘리 먼트를 반환한다. 예문에서는 <li> 태그를 모두 찾아서 리스트 자료구조로 엘리먼트들이 반환된다. find_all() 함수는 여러 개의 태그를 찾기 때문에 해당 엘리먼트들은 리스트에 저장된다.
- # (4) li 태그 내용
- 엘리먼트들이 저장된 리스트 객체(lis)를 대상으로 for문을 이용하여 엘리먼트 원소를 하나씩 넘겨받 아서 엘리먼트 객체의 string 멤버를 호출하여 태그를 감싸고 있는 목록을 출력한다.

8. 태그 속성 수집

- 엘리먼트의 구성요소는 시작태그, 속성과 값, 내용, 종료태그 로 구성된다. 지금까지는 특정 태그를 찾 아서 태그의 내용을 가져오는 방법에 대해서 알아보았는데, 이 절에서는 특정 태그의 속성을 가져오는 방법에 대해서 알아본다.
- 다음은 예문에서 사용될 html 파일의 내용이다. 속성을 수집할 태그는 <a> 태그이다. <a> 태그는 두 개의 속성을 가질 수 있는데, 하나는 'href' 속성으로 링크될 웹페이지의 주소를 나타내고, 또 하나는 'target' 속성으로 링크된 페이지를 새로운 창으로 열어주는 속성이다. html 파일에서 <a> 태그는 5 개를 사용하였다. 그 중에서 'target' 속성은 3번째 태그이다.

8. 태그 속성 수집

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>선(hr), 줄바꿈(br), 링크(a)</title>
</head>
<body>
  <hr/>
  <h1> 링크 태그 </h1>
  <hr/>

  <!-- 링크 태그 -->
  <!-- 형식) <태그명 속성="값"> 내용 </태그명> -->
  <a href="www.naver.com">네이버</a>
  <a href="http://www.naver.com">네이버</a>
  <br><br> <!-- 줄바꿈 -->
  <a href="http://www.naver.com" target="_blank">네이버 새창으로</a>
  <br><br>
  <a href="www.duam.net">다음</a>
  <a href="http://www.duam.net">다음</a>

</body>
</html>
```

8. 태그 속성 수집

chapter09.lecture.step03_tag_attr.py

```
from bs4 import BeautifulSoup # html 파싱

# (1) 로컬 파일 읽기
file = open("chapter09/data/html02.html",
  mode='r', encoding='utf-8')
source = file.read()

# (2) html 파싱
html = BeautifulSoup(source, 'html.parser
')

# (3) a 태그 찾기
links = html.find_all('a') # list 반환
print('links size=', len(links))
```

Python Console

```
links size= 5
```

8. 태그 속성 수집

```
# (3) a 태그 찾기
links = html.find_all('a') # list 반환
print('links size=', len(links))

# (4) a 태그에서 속성 찾기
for link in links :
    try :
        print(link.attrs['href']) # 5개
        print(link.attrs['target']) # erro
r(1,2,4,5)
    except Exception as e :
        print('예외발생 : ', e)


# (5) 정규표현식으로 속성 찾기
import re
print('패턴 객체 이용 속성 찾기')
patt=re.compile('http://') # pattern objec
t 생성
links = html.find_all(href = patt) # 패턴 찾
기
print(links)
```

```
links size= 5

www.naver.com
예외발생 : 'target'
http://www.naver.com
예외발생 : 'target'
http://www.naver.com
_blank
www.duam.net
예외발생 : 'target'
http://www.duam.net
예외발생 : 'target'


패턴 객체 이용 속성 찾기
[<a href="http://www.naver.
com">네이버</a>, <a href="htt
p://www.naver.com" target="
_blank">네이버 새창으로</a>, <a
href="http://www.duam.net">
다음</a>]
```

8. 태그 속성 수집

- 태그 속성 찾기 예
- # (1) 로컬 파일 읽기
- 해당 경로에서 html02.html 파일을 텍스트 방식으로 읽어온다.
- # (2) html 파싱
- html 태그로 인식할 수 있도록 문자열을 의미 있는 단위로 분해하고, 계층적인 트리 구조를 만드는 과정이다.
- # (3) a 태그 찾기
- html 파싱 객체를 이용하여 find_all('tag')함수를 호출하면 해당 태그들을 찾아서 태그들의 엘리 먼트를 반환한다. 예문에서는 <a> 태그를 모두 찾아서 리스트 자료구조로 엘리먼트들이 반환된다. 따라서 리스트의 원소는 5개가 된다.

8. 태그 속성 수집

- # (4) a 태그 속성 찾기
- <a> 태그의 속성을 찾기 위해서 엘리먼트 객체를 이용하여 attrs 멤버를 호출할 수 있다. attrs 멤버는 '키:값' 형식의 딕트 자료구조이다. '키'는 해당 태그의 속성이고, 값은 속성에 해당하는 값이 된다. 따라서 'href' 속성 값을 가져오기 위해서는 link.attrs['href'] 명령어를 이용하고, target 속성은 link.attrs['target'] 명령문을 이용한다. 하지만 <a> 태그 5개 중에서 target 속성이 없는 4개의 엘리먼트에서는 해당 키를 찾을 수 없기 때문에 예외가 발생되므로 try ~ except 블록으로 예외를 처리해주어야 한다. 그렇지 않으면 target 속성이 없는 엘리먼트에서 오류가 발생되 고 프로그램이 중단된다. 웹 문서의 수집과정에 이와 같은 예외는 빈번하게 발생되기 때문에 예외처리 를 반드시 적용할 필요가 있다.

8. 태그 속성 수집

- # (5) 정규표현식으로 속성 찾기
- re 모듈에서 제공하는 compile(pattern) 함수를 이용하여 정규표현식의 패턴 객체를 생성하고, find_all() 함수를 이용하여 href 속성 중에서 해당 패턴과 일치하는 엘리먼트를 반환 받을 수 있다.
- 예문에서는 <a> 태그 5개 중에서 url 주소가 'http://~'로 시작하는 올바른 url 주소를 갖는 엘리먼 트를 수집하는 것이 목적이다.

UNISTUDY

THANK YOU