

파이썬

35강. 선택자 자료 수집

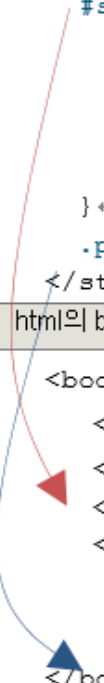


1. 선택자 자료 수집

- 선택자(selector)는 html의 <style>태그에서 디자인을 적용할 수 있는 tag, id, class 등을 말한다. 특히 id와 class는 <body> 태그에서는 특정 태그에서 속성으로 사용되고, <style> 태그에서는 #id와 .class로 표기된다. <body> 태그에서 특정 태그의 속성으로 id와 class를 지정하고, <style> 태그에서 id 선택자인 #summary에 적용할 스타일을 지정하고, class 선택자인 .plot-output에 적용할 스타일을 지정하고 있다.

1. 선택자 자료 수집

```
html의 style 태그 영역↵
<style>↵
  body{color : green}
  #summary{↵
    width : 400px; /* 폭 : 픽셀 크기 */↵
    border : 3px solid red; /* 테두리 : 크기 실선 빨강색 */↵
    padding : 4px; /* 안쪽 여백 : 픽셀 크기 */↵
    border-radius : 20px; /* 모서리가 둥근 사각형 - css3 추가 */↵
  }↵
  .plot-output{width: 100%; height: 400px} /* 폭과 높이 폭 크기 */↵
</style>↵
html의 body 태그 영역↵
<body>↵
  <h1>HTML UI</h1>↵
  <pre id="summary"></pre>↵
  <!-- 여기는 html 주석문↵
  <div id="plot" class="shiny-plot-output"
    style="width: 100%; height: 400px"></div> -->↵
  <div class="plot-output"></div>↵
</body>↵
```



1. 선택자 자료 수집

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

```
html파싱객체.select_one('선택자')  
html파싱객체.select('선택자') ↵
```

1. 선택자 자료 수집

- 다음은 예문에서 사용될 html 파일의 <style> 태그의 내용이다. 특히 #으로 시작하는 id 선택자의 스타일과 .으로 시작하는 class 선택자의 스타일을 지정하고 있다.

```
<!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>id/class 선택자, 표 꾸미기</title>
<style type="text/css">
  table{
    border-collapse : collapse; /* 중복 테두리 제거 */
    border : 3px solid black; /* 테두리 굵기 실선 색 */
    width: 80%; /* 화면 전체 폭 */
    text-align: center; /* 텍스트 정렬 방식 */
  }
  tr{
    height: 24px; /* 줄 간격 */
    font-size: 13px; /* 글자 크기 */
  }
```

1. 선택자 자료 수집

```
/* 행 위에 마우스 오버 시
*/tr:hover{↵
    color: #ffffff; /* 글자색 흰색 */
    font-weight: bold; /* 굵게 */↵
    background-color: #00ccff; /* 배경색 : 하늘색 계열
    */cursor: pointer; /* 커서 모양 변경 */↵
}↵
/* 표 제목 열 스타일 적용
*/th{↵
    height: 25px;
    font-size: 14px;↵
    background-color: #e0bbd2; /* RGB 색상표 - NAVER */↵
}↵
/* 형식 : #id이름{ 속성 : 값; } - 이름 중복 불가능 */
#id{width : 15%}↵
#name{width: 10%}↵
#major{width: 25%}↵
#email{width: 60%}↵
↵
/* 형식 : .class이름{ 속성 : 값; } - 이름 중복 가능 */↵
.odd{background-color: #F2FF9D;} /* 홀수행 배경색 적용 */↵
</style>↵
</head>↵
```

1. 선택자 자료 수집

- 다음은 예문에서 사용될 html 파일의 <body> 태그의 내용이다. 특히 <th> 태그에서 id 선택자를 속 성으로 사용했고, <tr> 태그에서는 class 선택자를 속성으로 사용하고 있다.

```
<body>␣
␣<table border = 1 id="tab">␣
␣␣<tr> ␣<!-- 1행 -->␣
␣␣␣<!-- 제목 열 : th -->␣
␣␣␣␣<th id="id"> 학번 </th>␣
␣␣␣␣<th id="name"> 이름 </th>␣
␣␣␣␣<th id="major"> 학과 </th>␣
␣␣␣␣<th id="email"> 이메일 </th>␣
␣␣</tr>␣
␣␣<tr> ␣<!-- 2행 -->␣
␣␣␣<td> 201601 </td>␣
␣␣␣<td> 홍길동 </td>␣
␣␣␣<td> 체육학과 </td>␣
␣␣␣<td> hong@naver.com </td>␣
␣␣</tr>␣
␣␣<tr class="odd"> ␣<!-- 3행 (홀수) -->␣
␣␣␣<td> 201602 </td>␣
␣␣␣<td> 이순신 </td>␣
␣␣␣<td> 해양학과 </td>␣
␣␣␣<td> lee@naver.com </td>␣
␣␣</tr>␣
```

1. 선택자 자료 수집

```
<tr class="odd"> <!-- 3행 (홀수) -->↵
  <td> 201602 </td>↵
  <td> 이순신 </td>↵
  <td> 해양학과 </td>↵
  <td> lee@naver.com </td>↵
</tr>↵
<tr> <!-- 4행 -->↵
  <td> 201603 </td>↵
  <td> 강감찬 </td>↵
  <td> 정치외교 </td>↵
  <td> kang@naver.com </td>↵
</tr>↵
<tr class="odd"> <!-- 5행 -->↵
  <td> 201604 </td>↵
  <td> 유관순 </td>↵
  <td> 유아교육 </td>↵
  <td> you@naver.com </td>↵
</tr>↵
</table>↵
</body>↵
</html>↵
```


1. 선택자 자료 수집

chapter09.lecture.step04_selector.py

Python Console

```
from bs4 import BeautifulSoup
```

```
# (1) 로컬 파일 읽기
```

```
file = open('chapter09/data/html03.html',  
mode='r',  
encoding='utf-8')  
source = file.read()
```

```
# (2) html 파싱
```

```
html = BeautifulSoup(source, 'html.parser')  
)
```

```
# (3) 선택자 이용 태그 내용 가져오기
```

```
# (3-1) id='tab' 선택자
```

```
print('>> table 선택자 <<')  
table = html.select_one('#tab') # <table i  
d='tab'>  
print(table) # table 태그 전체 출력
```

```
# (3-2) id 선택자와 계층
```

```
print('>> 선택자 & 계층 <<')  
ths = html.select('#tab > tr > th')  
print(ths) # list
```

```
↵  
↵  
↵  
↵  
↵  
↵
```

```
>> table 선택자 <<
```

```
<table border="1" id="tab">
```

```
<tr> <!-- 1행 -->
```

```
<!-- 제목 열 : th -->
```

```
<th id="id"> 학번 </th>
```

```
<th id="name"> 이름 </th>
```

```
생략 ...
```

```
<tr class="odd"> <!-- 5행 -->
```

```
<td> 201604 </td>
```

```
<td> 유관순 </td>
```

```
<td> 유아교육 </td>
```

```
<td> you@naver.com </td>
```

```
</tr>
```

```
</table>
```

```
↵
```

```
>> 선택자 & 계층 <<
```

```
[<th id="id"> 학번 </th>, <th  
id="name"> 이름 </th>, <th id=  
="major"> 학과 </th>, <th id=  
="email"> 이메일 </th>]
```

1. 선택자 자료 수집

```
# (3-3) class 선택자 : tr tag class='odd'
trs = html.select("#tab > .odd") # 홀수 행
print('>> class 선택자 <<')
print(trs)
```

```
# (4) 태그[속성=값] 찾기
trs = html.select("tr[class=odd]")
```

```
# (5) td 태그 내용
print('>> tr > td 출력 << ')
for tr in trs : # 행 : 2회 반복
    tds = tr.find_all('td')
    for td in tds : # 열
        print(td.string)
```

```
"email"> 이메일 </td>]

```

```
>> class 선택자 <<
[<tr class="odd"> <!-- 3행 (홀수) -->
<td> 201602 </td>
<td> 이순신 </td>
생략 ...
</tr>, <tr class="odd"> <!-- 5행 -->
<td> 201604 </td>
<td> 유관순 </td>
<td> 유아교육 </td>
<td> you@naver.com </td>
</tr>]
```

```
>> tr > td 출력 <<
201602
이순신
해양학과
lee@naver.com
201604
유관순
유아교육
you@naver.com
```

1. 선택자 자료 수집

- 선택자 자료 수집 예
- # (1) 로컬 파일 읽기
- 해당 경로에서 html03.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) id='tab' 선택자
- 선택자의 엘리먼트를 수집하기 위해서는 html 파싱 객체를 이용하여 `select_one('선택자')` 함수를 호출하면 해당 선택자의 태그를 찾아서 태그의 엘리먼트를 반환한다. 예문에서는 id 속성 값이 'tab' 인 `<table>` 태그를 찾아서 `<table>` 태그의 엘리먼트를 반환한다.

1. 선택자 자료 수집

- # (3-2) id 선택자와 계층
- html의 태그들은 상하관계의 계층적인 구조를 갖는다. 예를 들면 <table> 태그는 <tr> 태그를 포함하고 있고, <tr> 태그는 <th> 또는 <td> 태그를 포함하고 있다. 이러한 태그의 계층적 구조는 'table > tr > th 또는 td' 형식으로 태그의 상하관계를 나타낼 수 있다. 예문에서는 id 속성 값이 'tab'인 <table>태그의 최하위 <th>태그의 엘리먼트를 반환한다. 현재 html 파일에서는 복수 개의 <th> 태그를 사용하기 때문에 select('선택자') 함수를 호출한다.
- # (3-3) class 선택자
- <table> 태그의 하위 태그에서 class 속성 값이 'odd'인 태그의 엘리먼트를 반환한다. 즉 홀수 행으로 지정한 2개의 <tr> 태그가 여기에 해당되므로 2개의 엘리먼트를 리스트로 반환한다. 복수개의 태그를 사용하기 때문에 select('선택자') 함수를 호출한다.

1. 선택자 자료 수집

- # (4) 태그[속성=값] 찾기
- 태그의 계층적인 구조로 접근하지 않고, 직접 해당 태그의 선택자로 접근하기 위해서는 '태그[선택자=값]' 형식으로 나타낼 수 있다. 예문에서 `select("tr[class=odd]")` 명령문은 `<tr>` 태그 중에서 `class` 속성이 `odd`인 태그의 엘리먼트를 반환한다. 결과는 (3-3) class 선택자와 동일하다.
- # (5) td 태그 내용
- 예문에서 `trs`는 홀수 행으로 지정한 2개의 `<tr>` 태그의 엘리먼트를 리스트로 반환받은 객체이다. `for`문을 이용하여 `<tr>` 엘리먼트를 한 개씩 반환받아서 하위 태그인 `<td>` 태그를 모두 찾아서 태그의 내용을 출력한다.

2. news 자료 수집

- 실시간 news 기사를 제공하는 포털 사이트의 웹 문서를 대상으로 news 기사를 수집하고, 수집된 자료를 이진파일로 저장하는 과정에 대해서 알아본다.
- 다음 예문으로 사용되는 Daum 포털 사이트의 실시간 news를 제공하는 웹 페이지 (<http://media.daum.net19>)와 소스 보기의 결과이다.

2. news 자료 수집



2. news 자료 수집

- 해당 웹 페이지에서 news는 <a> 태그에 의해서 제공하고 있다. <a> 태그의 속성으로 class 선택자 를 이용했으며, 선택자의 값은 "link_txt" 이다.

chapter09.lecture.step05_newsCrawling.py ↵

Python Console ↵

```
# 모듈 import ↵
import urllib.request as req
from bs4 import BeautifulSoup ↵

# news 제공 포털 사이트 ↵
url = "http://media.daum.net" ↵

# (1) url 요청 ↵
res = req.urlopen(url)
source = res.read() ↵

# (2) source 디코딩 ↵
source = source.decode("utf-8") # charset=
"utf-8" ↵

# (3) html 파싱 ↵
html = BeautifulSoup(source, 'html.parser
') ↵
```


2. news 자료 수집

```

# (4) tab[속성=값] 요소 추출
atags = html.select('a[class=link_txt]')
print('a tag 수=', len(atags)) # a tag 수 = 115

# (5) a 태그 내용 수집
crawling_data = [] # 빈 list

cnt = 0
for atag in atags :
    cnt += 1
    atagStr = str(atag.string) # string 변환
    crawling_data.append(atagStr.strip())
    '''
    string.strip() : 문단 끝 불용어 (공백, tab,
    \n\r) 제거
    '''
    # 수집한 자료 확인
    print('수집한 자료 확인')
    print(crawling_data)

# (6) pickle save/load
import pickle # object -> file(binary) -> load(object)

```

수집한 자료 확인
 ['지하철 노사 막판 줄다리기..세 가지
 핵심 쟁점은', '내일 서울 지하철 파업하
 나? 노사 '마지막 협상' 돌입', '서울지
 하철 파업 '밤샘 협상'..교통혼란 예고
 ', '"일본 안 가고 홍콩 못 가고'..여
 행업계, 가을 성수기도 '시름'", '유니
 클로 야나이 회장 "이대로 가면 일본은 망
 할 것"', '"일본산 맥주' 외면..옛 명성
 회복 불능', '억울한 옥살이' 또
 ?.. "2년 복역했는데 이춘재 짓"', '"
 화성 사건'
 생략...']

pickle load
 ['지하철 노사 막판 줄다리기..세 가지

2. news 자료 수집

```
# save : binary file save↵
file = open('chapter09/data/data.pickle',
mode='wb')↵
pickle.dump(crawling_data, file)↵

# load : binary file load↵
file = open('chapter09/data/data.pickle',
mode='rb')↵
crawling_data = pickle.load(file)
print('pickle load')
print(crawling_data)↵
```

핵심 쟁점은', "내일 서울 지하철 파업하
나? 노사 '마지막 협상' 돌입", "서울지
하철 파업 '밤샘 협상'..교통혼란 예고
", "'일본 안 가고 홍콩 못 가고'..여↵
행업계, 가을 성수기도 '시름'", '유니
클로 야나이 회장 "이대로 가면 일본은 망
할 것"', "'일본산 맥주' 외면..옛 명성
회복 불능", '억울한 옥살이\'' 또↵
?... "2년 복역했는데 이춘재 짓"', "'↵
화성 사건'
생략...]↵

2. news 자료 수집

- 해설 news 자료 수집 예
- # (1) url 요청
- request 모듈의 urlopen(url) 함수를 이용하여 url에 해당하는 원격서버의 웹 문서 파일의 요청 객체를 만들고 객체에서 제공하는 read() 함수를 통해서 텍스트 자료를 읽어온다.
- # (2) source 디코딩
- 한글과 같은 유니코드를 대상으로 디코딩하여 한글 자료를 깨지지 않게 한다. 디코딩 방식은 해당 사이트의 소스 보기에서 확인할 수 있다.
- # (3) 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.

2. news 자료 수집

- # (4) 태그[속성=값] 찾기
- 태그의 계층적인 구조로 접근하지 않고, 직접 해당 태그의 선택자로 접근하기 위해서는 '태그[선택자=값]' 형식으로 나타낼 수 있다.
- 예문에서 `select("a[class=link_txt]")` 명령문은 `<a>` 태그 중에서 class 속성이 link_txt인
- `<a>` 태그의 모든 엘리먼트를 리스트로 반환한다. 수집한 엘리먼트의 수는 115개이다.
- # (5) a 태그 내용 수집
- 115개의 엘리먼트를 저장한 리스트 객체를 for문으로 하나씩 넘겨받아서 string 멤버를 이용하여 태그의 내용을 추출한다. 그리고 추출된 문자열 끝 부분에 붙어있는 불용어(공백, \t, \n \r)를 제거하기 위해서 제공하는 `strip()` 함수를 이용한다. 불용어가 제거된 news를 `crawling_data` 리스트에 추가한다.

2. news 자료 수집

- # (6) pickle save/load
- news를 수집한 리스트 객체(crawling_data)를 이진파일로 저장하기 위해서 pickle의 dump 함수를 이용한다. 또한 저장된 이진파일을 불러오기 위해서 pickle의 load() 함수를 이용한다. pickle 파일을 로드한 결과를 출력하면 수집한 자료의 원본과 동일한 것을 확인 할 수 있다. 결국 pickle 파일로 저장한다는 것은 현재 자료의 객체를 그대로 유지해서 파일로 저장하는 것이 목적이다.

THANK YOU