학습 정리

팀	이진조	구성원	이규진

일정	발제자	주제
5/30 (목)	이규진	BeautifulSoup 사용법 및 간단 웹 파싱 기초 및 파싱 실습(네이버, 다음, 인프런)

주요 내용 요약

Section 2 - 파이썬 기초 스크랩핑

파이썬으로 youtube 동영상 다운받고 mp3 변환하기

● 오늘 배울 내용 정리

1. Pytube로 원하는 youtube 동영상 저장하기
2. **동영상 -> mp3 변환**
3. 동영상 다운 및 mp3 변환 한 번에 자동화 하기

* 실습(과제): 동영상 url 입력 받아 다운 & 변환 해보기 +@

https://github.com/nficano/pytube

https://docs.python.org/3.6/library/subprocess.html

윈도우: https://www.filehorse.com/download-ffmpeg-64 => ffmpeg 코덱 다운받기

Mac(linux) : https://www.ffmpeg.org/download.html

안녕하세요. 현재 2018.12.24일 기준으로 pytube가 버전업이 되었습니다. 기존 강의에서 youtube 동영상이 다운이 안되는 에러는 아래와 같이 pytube를 설치하시면 실행 가능합니다. github에는 새로운 버전의 pytube가 업데이트 되어 있습니다.

* **pytube가 설치 되었을 경우** -> pytube 삭제 후 재설치 -> 아나콘다 재실행 -> atom 재실행
  * **pip uninstall pytube**
  * **pip install git+git://github.com/nficano/pytube**

* **pytube가 설치 안되었을 경우** -> pytube 설치 -> 아나콘다 재 실행 -> atom 재 실행
  - **pip install git+git://github.com/nficano/pytube**

위와 같이 github에 있는 버전으로 설치하시면 정상 실행 됩니다. **섹션 6에서 최종적으로 유투브 다운로더 어플 제작 관련 강의**에도 위와같이 설치 부탁드립니다. 감사합니다.

● 유튜브 동영상 다운 및 mp3 다운

```python
import pytube
import os
import subprocess # 파이썬을 실행하면서 별도의 프로세스를 생성해서
# 터미널이나 커맨드에 명령어를 실행 할 수 있고 그 반환값을 가져 올 수 있다.


yt = pytube.YouTube("https://youtu.be/junvn7qKikc") # "" 안에 다운 받을 URL
videos = yt.streams.all() # 동영상 화질별로 리스트 형태로 변수에 저장됨

for i in range(len(videos)):
    print(i, ',', videos[i])

cNum = int(input("다운 받을 화질은?(0~21 입력)"))

down_dir = "C:/youtube" # 다운받을 URL 경로

videos[cNum].download(down_dir) # 동적으로 원하는 화질을 선택 후 저장

newFileName = input("변환 할 mp3 파일명은?")
oriFileName = videos[cNum].default_filename

subprocess.call(['ffmpeg', '-i',
    os.path.join(down_dir, oriFileName),
    os.path.join(down_dir,newFileName),
# ffmpeg를 환경변수로 등록하면 어디에서나 코드를 실행해도 관계없지만
# 그렇지 않은 경우에는 파이썬 코드를 영상을 받을 위치에 옮겨줘야한다.
])

print("동영상 다운로드 및 mp3 변환 완료")
```

BeautifulSoup 사용법 및 간단 웹 파싱 기초(1)

● 오늘 내용 정리

1. BeautifulSoup 간단 파싱 학ㄱ습
2. **`urljoin`, `find_all`, `select_one`, `next_sibling`, `previous_sibling`**
3. 선택자(Selector)

BeautifulSoup : https://www.crummy.com/software/BeautifulSoup/bs4/doc/
Selector: https://www.w3schools.com/cssref/css_selectors.asp

온라인(추천): https://www.w3schools.com/cssref/trysel.asp

BeautifulSoup 설치

콘솔창에 `pip install beautifulsoup4`

`conda list`로 설치된 프로그램 확인

1. urljoin

```python
from urllib.parse import urljoin

baseUrl = "https://test.com/html/a.html"
print(">>", urljoin(baseUrl, "b.html"))
print(">>", urljoin(baseUrl, "sub/b.html"))
print(">>", urljoin(baseUrl, "../index.html"))
print(">>", urljoin(baseUrl, "../img/img.jpg"))
print(">>", urljoin(baseUrl, "../css/img.css"))

# urljoin을 사용하게 되면 절대 경로는 묶어놓고 나머지 파일의 위치만 조정 가능
# ..을 사용하게 되면 한칸 윗단계로 올라감, 위에 올라 갔을때 절대 경로위치면 그 위치부터 경로 설정
```

2. next_sibling, previous_sibling

```python
html = """
<html>
<body>
<h1>파이썬 BeautifulSoup 공부 </h1>
<p>태그 선택자</p>
<p>CSS 선택자</p>
</body>
</html>
"""
soup = BeautifulSoup(html, 'html.parser') # 첫번째 인자로는 URL, html등을 받음, 두번째 인자로 parser를 지정
# print('soup', type(soup))
# print('prettify',soup.prettify()) # prettify 함수를 쓰면 들여쓰기를 해줌
<li>

h1 = soup.html.body.h1
print('h1',h1)
print('h1', type(h1))
print('h1', h1.string) # 스트링만 출력

p1 = soup.html.body.p
```

```
print('p1', p1)

p2 = p1.next_sibling.next_sibling
# 같은 형제위치에 있는 selector가 여러개면 next_sibling으로 '다음" 위치태그 접근 가능
# 줄바꿈했을 경우 next_sibling을 한번 더 입력 해야함
print('p2', p2)

p3 = p2.previous_sibling.previous_sibling
# 같은 형제위치에 있는 selector가 여러개면 next_sibling으로 '이전' 위치태그 접근 가능
# 더 이상 형제가 없다면 그 위에 상위 부모 selector로 이동
print('p3', p3)

print("h1 >> ", h1.string)
print("p >> ", p1.string)
print("p >> ", p2.string)

# 사실상 현업에서 중간에 코드를 삽입하고 수정하는 경우가 많기때문에 이렇게 수동으로 접근하는 방법은 잘 쓰이지 않음
```

3. find_all()

```python
from bs4 import BeautifulSoup
import sys
import io

sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding = 'utf-8')
sys.stderr = io.TextIOWrapper(sys.stderr.detach(), encoding = 'utf-8')

html = """
<html><body>
   <ul>
      <li><a href="http://www.naver.com">naver</a></li>
      <li><a href="http://www.daum.net">daum</a></li>
      <li><a href="https://www.google.com">google</a></li>
      <li><a href="https://www.tistory.com">tistory</a></li>
    </ul>
</body></html>
"""

soup = BeautifulSoup(html, 'html.parser')

links = soup.find_all("a")
# print('links', type(links))
a = soup.find_all("a",string='daum') # string이 daum인 요소만 갖고옴
print('a', a)
# => a [<a href="http://www.daum.net">daum</a>]
```

```
b = soup.find("a") # find는 하나만 가져옴(가장 첫번째 것)
print('b', b)
# => b <a href="http://www.naver.com">naver</a>


c = soup.find_all("a", limit=2) # limit까지의 수만큼 가져옴
print('c', c)
# => [<a href="http://www.naver.com">naver</a>, <a
href="http://www.daum.net">daum</a>]

d = soup.find_all(string=["naver", "google"]) # 해당 스트링을 가지고옴
print('d', d)
# => ['naver', 'google']

for a in links:
    print('a', type(a), a)
    href = a.attrs['href'] # attrs를 사용하면 key=value형태로 반환 (href가 키인 value(링크)를
반환)
    txt = a.string
    print('txt >>', txt, 'href >>', href)
```

● select, select_one

=> selector 선택자 접근으로 파싱한다면 중간에 코드가 추가되거나 삭제되더라도 크게 수정 할 필요가 없다.

```python
html = """
<html><body>
<div id="main">
  <h1>강의목록</h1>
  <ul class="lecs">
    <li>Java 초고수 되기</li>
    <li>파이썬 기초 프로그래밍</li>
    <li>파이썬 머신러닝 프로그래밍</li>
    <li>안드로이드 블루투스 프로그래밍</li>
  </ul>
</div>
</body></html>
"""

soup = BeautifulSoup(html, 'html.parser')
h1 = soup.select("div#main > h1") # select를 사용하게 되면 리스트 형태로 저장됨
print("h1", h1)
print("h1", type(h1))
print(*h1)
for i in h1: # 리스트형태는 바로 스트링으로 출력을 못하기 때문에 for으로 요소를
꺼내줘야함
    print(i.string)
```

```
h2 = soup.select_one("div#main > h1") # 가지고올 Selector가 하나라면 select_one으로 접근해서 바로 스트링출력 가능
print('h1', h2.string)

list_li = soup.select("div#main > ul.lecs > li")
for li in list_li:
    print("li >>>", li.string)

```

BeautifulSoup 사용법 및 간단 웹 파싱 기초(2)

● 오늘 내용 정리

1. BeautifulSoup HTML 파일 파싱 실습
2. 더욱 다양하게 CSS 선택자 사용해보기
3. find, select 실습 예제

```python
from bs4 import BeautifulSoup
import sys
import io
import re # regex


sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding = 'utf-8')
sys.stderr = io.TextIOWrapper(sys.stderr.detach(), encoding = 'utf-8')

html = """
<html><body>
    <ul>
        <li><a id="naver" href="http://www.naver.com">naver</a></li>
        <li><a href="http://www.daum.net">daum</a></li>
        <li><a href="https://www.google.com">google</a></li>
        <li><a href="https://www.tistory.com">tistory</a></li>
      </ul>
</body></html>
"""

soup = BeautifulSoup(html, 'html.parser')
print(soup.find(id="naver").string)
li = soup.find_all(href=re.compile(r"^https://")) # 정규식표현 ^은 https://로 시작하는 문자열

print('li', li)
for e in li:
    print(e.attrs['href'])
```

id값을 통한 접근

```python
from bs4 import BeautifulSoup
import sys
import io

sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding = 'utf-8')
sys.stderr = io.TextIOWrapper(sys.stderr.detach(), encoding = 'utf-8')

fp = open("food-list.html",encoding="utf-8") # food-list 파일 열
soup = BeautifulSoup(fp, "html.parser")

print(soup)
print("1", soup.select("li:nth-of-type(4)")[1].string) #각 li 태그 그룹의 4번째 요소 선택
print("2", soup.select_one("#ac-list > li:nth-of-type(4)").string)
print("3", soup.select("#ac-list > li[data-lo='cn']")[0].string)
print("4", soup.select("#ac-list > li.alcohol.high")[0].string)

param = {"data-lo": "cn", "class": "alcohol"}
print("5", soup.find("li", param).string) # dictionary형태도 두번째 인작밧에 들어 갈 수 있다.
print("6", soup.find(id="ac-list").find("li",param).string)

for ac in soup.find_all("li"):
    if ac['data-lo'] == 'us':
        print('data-lo == us', ac.string)
```

람다식을 활용한 selector 접근

```python
from bs4 import BeautifulSoup
import sys
import io

sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding = 'utf-8')
sys.stderr = io.TextIOWrapper(sys.stderr.detach(), encoding = 'utf-8')

fp = open("cars.html",encoding="utf-8") # food-list 파일 열
soup = BeautifulSoup(fp, "html.parser")

def car_func(selector):
    print("car_func", soup.select_one(selector).string) # select_one은 스트링으로 접근가능

# 람다식
car_lambda = lambda q : print("car_labda", soup.select_one(q).string)

car_func("#gr")
car_func("li#gr") # li태그의 id가 gr
car_func("ul > li#gr")
car_func("#cars #gr")
```

```
car_func("#cars > #gr")
car_func("li[id='gr']")

car_lambda("#gr")
car_lambda("li#gr") # li태그의 id가 gr
car_lambda("ul > li#gr")
car_lambda("#cars #gr")
car_lambda("#cars > #gr")
car_lambda("li[id='gr']")

print("car_func",soup.select("li")[3].string)
print("car_func",soup.find_all("li")[3].string)
```