

06. 크롤러(Crawler) 만들기

김 은 경

데이터 수집에 필요한 대표적인 모듈

- ▶ `urllib.request` or `requests`

- ▶ 사이트에 연결해서 원하는 페이지를 요청 받아 주는 모듈

- ▶ `BeautifulSoup`

- ▶ 요청 받아 온 페이지를 파싱하고, 파싱 결과 나온 파이썬 객체를 대상으로 특정 정보를 추출할 수 있도록 하는 모듈

분석 및 시각화에 필요한 대표적인 모듈

▶ scipy & numpy

- ▶ 고성능 과학계산 컴퓨팅과 데이터 분석을 위한 모듈
- ▶ 배열 생성, 난수 생성, 산술 계산, 데이터 수치 분석 등을 지원

▶ pandas

- ▶ 고수준의 자료구조와 빠르고 쉬운 데이터 분석을 지원하는 모듈
- ▶ 데이터 가공을 위한 핵심 모듈로써 Series, DataFrame 자료구조 생성, 데이터 읽어오기, 연산 등을 지원

▶ matplotlib

- ▶ 데이터를 2D 및 3D로 시각화할 수 있도록 하는 모듈

개발 환경 구축

- ▶ 이미 설치한 모듈들 (*pip list 명령으로 확인 가능)
 - ▶ requests 과 bs4
- ▶ **아나콘다(Anaconda)**로 필요한 모듈 일괄 설치
 - ▶ Continuum Analytics 에서 만든 설치 소프트웨어
 - ▶ the popular Python data science platform
 - ▶ 450여 개의 파이썬 라이브러리 포함
 - ▶ 머신러닝이나 데이터 과학을 위한 패키지가 이미 포함되어 있으므로 tensorflow나 R 등을 사용할 때 더 편리
 - ▶ Window, Mac, Linux 모두에서 설치 가능
 - ▶ Anaconda를 설치하기 전이나 후에 별도로 Python을 설치할 필요가 없다

* PIP을 이용한 패키지 설치의 운영체제에 따라 에러가 나는 경우가 많다.

개발 환경 구축

▶ 훌륭한 개발 파트너 : Jupyter Notebook

- ▶ IPython(Interactive Python)의 업그레이드 버전
- ▶ 프로그램 코드, 코드의 실행 결과, 코드에 대한 설명을 한번에 작성하도록 하여 체계적인 기록이 남도록 하는 파이썬 패키지
- ▶ 아나콘다를 설치하면 바로 사용 가능하며,
pip install jupyter notebook 명령을 통해
독립적으로 설치 가능하다!

크롤러 만들기

크롤링(crawling)

- 페이지 요청
- 파싱 (파이썬 객체로의 변환)
- 정제 (필요한 정보만 추출)
- 파일로 저장

- 파일을 데이터 프레임으로 불러오기
- 목적을 위한 분석 및 시각화

웹 페이지 요청하기

- 파이썬에서의 웹 페이지 요청 방법
 - ① urllib 라이브러리 사용
 - ② requests 모듈 사용

요청 연습을 위한 대상

단순한 페이지

- ▶ <http://pythonscraping.com/pages/page1.html>



- ▶ urllib는 설치하지 않고 사용할 수 있다니
이것으로 시작~~

urllib로 요청하기

- 우선 첫 연결을 위해 다음과 같이 코딩 해 보자.

```
from urllib.request import urlopen  
  
htmlObj = urlopen("http://pythonscraping.com/pages/page1.html")
```

위 코드는

- ① urllib 라이브러리에서 request 모듈을 읽고,
그 속의 urlopen 함수만 импорт 한다.
- ② "http://pythonscraping.com/pages/page1.html" URL 을 이용하여
urlopen 함수가 서버에 자료를 요청한 후 응답 메시지를 받아 온다.

urllib로 요청하기

- ▶ 각 객체에 대한 간단한 소개
 - ▶ urllib ... 파이썬 표준 라이브러리
 - ▶ request ... urllib 에 속하는 하나의 모듈
 - ▶ urlopen ... request 모듈에 속하는 하나의 함수
 - ▶ htmlObj ... 요청으로 받은 응답 메시지 객체

urllib로 요청하기

■ urllib

*** 표준 라이브러리로 설치 없이 사용**

- <https://docs.python.org/3/library>
- a package that collects several modules for **working with URLs**
- <https://docs.python.org/3/library/urllib.html>
- 데이터를 요청하는 함수
- 쿠키를 처리하는 함수
- 헤더나 유저 에이전트 같은 메타 데이터를 바꾸는 함수....

urllib로 요청하기

■ urllib.request

- Extensible library for **opening** URLs
- <https://github.com/python/cpython/blob/3.6/Lib/urllib/request.py>

■ urlopen

- 네트워크를 통해 원격 객체를 읽는 함수
- HTML 파일이나 이미지 파일 기타 파일들을 쉽게 바이트 스트림으로 받아오는 표준 함수

request.py 소스 코드

The screenshot shows the GitHub repository page for `python / cpython`. The file `request.py` is selected, showing its code. A modal window titled "Users who have contributed to this file" is open, listing 14 contributors. The background shows the repository's star/fork counts (17,397 stars, 5,159 forks) and the file's commit history (03066a0 on Apr 10 2017).

Users who have contributed to this file

- orsenthil
- vadmiun
- benjaminp
- ronaldoussoren
- birkenfeld
- pitrou
- bitdancer
- asvetlov
- serhiy-storchaka
- rhettinger
- nvawda
- jeremyhilton
- giampaolo

요청 후 응답 상태 확인

```
from urllib.request import urlopen  
  
htmlObj = urlopen("http://pythonscraping.com/pages/page1.html")  
  
print(htmlObj.status)  
print(htmlObj)    # 응답 상태 간접적 확인
```

200

- ▶ 응답 코드
 - ▶ 100~199: 요청이 현재 처리되는 중
 - ▶ **200~299**: 요청에 의한 응답이 성공적으로 수행되었다는 것
 - ▶ 300~399: 리다이렉션(redirection)
 - ▶ 400~499: 클라이언트 쪽의 에러
 - ▶ 500~599: 서버 쪽의 에러

요청 후 응답 메시지

응답 메시지의 예

HTTP를 기반으로 한 통신의 4 단계

- ① 연결 설정
- ② 요청 메시지 전송
- ③ 응답 메시지
- ④ 연결 끊기

Server: Apache

MIME-version:1.0

Content-type:text/html

Content-length: 107

[공백 라인]

<html>

<HEAD>

<Title>

A Sample HTML file

</Title>

</HEAD>

<body>

The rest of the document goes here

</body>

</html>

응답 메시지의 헤더 정보 확인

```
>>> status = htmlObj.getheaders()
>>> for s in status :
    print(s)
```

<실행 결과>

```
('Date', 'Tue, 03 Jul 2018 05:55:30 GMT')
('Server', 'Apache')
('Last-Modified', 'Sat, 09 Jun 2018 19:15:58 GMT')
('ETag', '"4121bc8-234-56e3a58b39172"')
('Accept-Ranges', 'bytes')
('Content-Length', '564')
('Cache-Control', 'max-age=1209600')
('Expires', 'Tue, 17 Jul 2018 05:55:30 GMT')
('Connection', 'close')
('Content-Type', 'text/html')
```

getheaders()

응답 메시지에서
웹 서버에 대한
정보를 추출하여
리스트 형태로
돌려 준다.

운영체제, 날짜, 타입,
서버의 종류 등

크롤링 하려는
페이지가 어떤 형식으
로 만들어져 있는지 알
수 있게 해 준다.

urllib로 요청하기

- ▶ 요청해서 받은 페이지의 내용은 어떤 모습일까?

```
In [1]: from urllib.request import urlopen
```

```
In [2]: htmlObj = urlopen("http://pythonscraping.com/pages/page1.html")
```

```
In [3]: print(htmlObj.read( )) #바이트 스트림 형태를 문자로 변환
```

```
b'<html>\n<head>\n<title>A Useful Page</title>\n</head>\n<body>\n<h1>An  
Interesting Title</h1>\n<div>\nLorem ipsum dolor sit amet, consectetur a  
dipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore ma  
gna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco l  
aboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in  
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pari  
atur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui  
officia deserunt mollit anim id est laborum.\n</div>\n</body>\n</html>\n'
```

* 이것이 크롤링의 대상이 되는 내용

import urllib.request

#바이트 스트림을 전달 받음

```
htmlObj = urllib.request.urlopen("http://pythonscraping.com/pages/page1.html")
```

```
print(htmlObj)
```

#문자로 변환 후 출력

```
print(htmlObj.read())
```

만약 제대로 변환이 안되면 인코딩 했던 방법을 알아내어 디코딩 해 주어야 한다.

```
print(htmlObj.read().decode("utf-8"))
```

두 명령 중
선택적으로
사용할 것

```
import urllib.request as r
```

```
htmlObj = r.urlopen(" http://pythonscraping.com/pages/page1.html ")  
print(htmlObj)  
print(htmlObj.read())
```

```
import urllib.request
```

```
r = urllib.request
```

```
htmlObj = r.urlopen(" http://pythonscraping.com/pages/page1.html ")  
print(htmlObj)  
print(htmlObj.read())
```

```
from urllib.request import urlopen, Request
```

```
url = "http://pythonscraping.com/pages/page1.html"
```

```
req = Request(url)    #요청 객체 만들기
```

```
htmlObj = urlopen(req)    #요청 정보를 넘겨주고 페이지를 받아 옴
```

```
print(htmlObj)
```

```
print(htmlObj.read())
```

요청 정보로 url 외에 추가적인 정보를 같이 보내고 싶을 때
Request()를 명시적으로 호출하여
요청객체를 만든 후에 urlopen()을 호출한다.

```
from urllib.request import urlopen
```

```
htmlObj = urlopen("http://pythonscraping.com/pages/page1.html")
```

```
print(htmlObj)
```

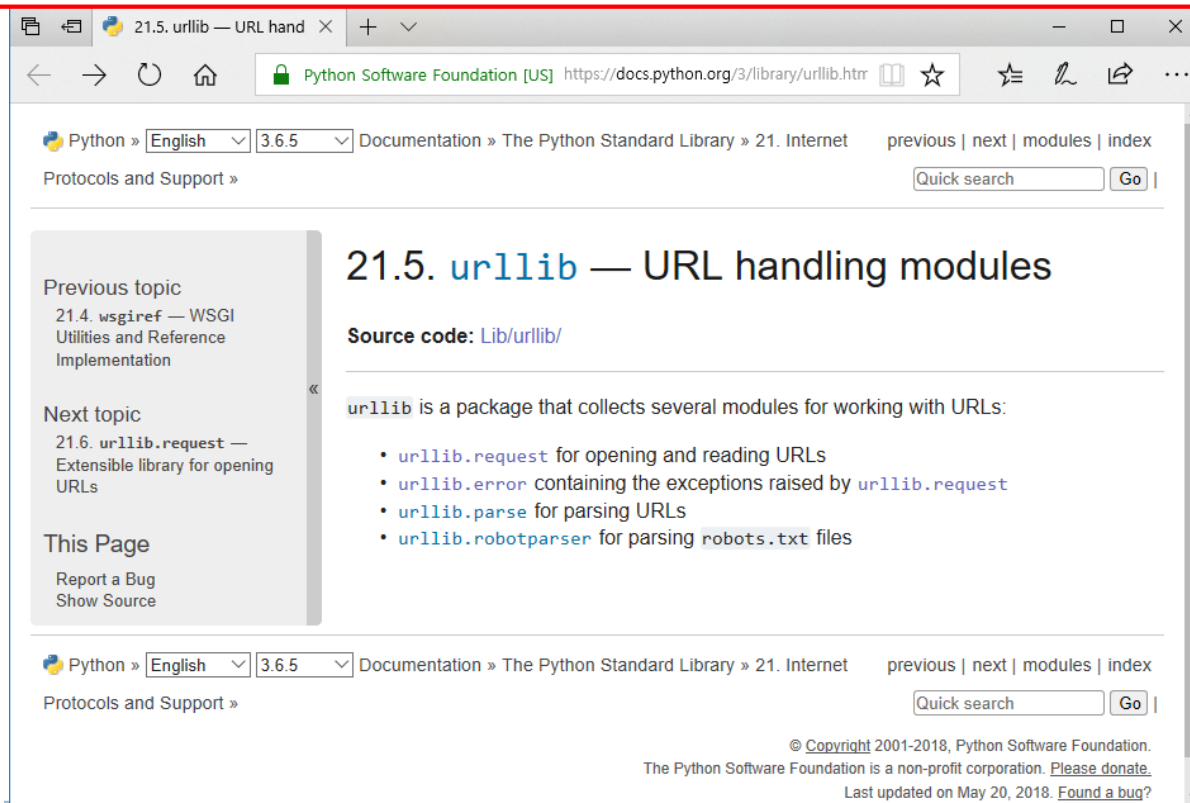
```
print(htmlObj.read())
```

* 시작하는 우리는 가장 간단한 이 요청 방법으로 적응합니다.

다른 사이트의 웹 페이지 요청하기

- ▶ 파이썬 표준라이브러리 중 urllib 페이지를 요청해 보자.

"https://docs.python.org/3/library/urllib.html"



다른 사이트에 페이지 요청하기

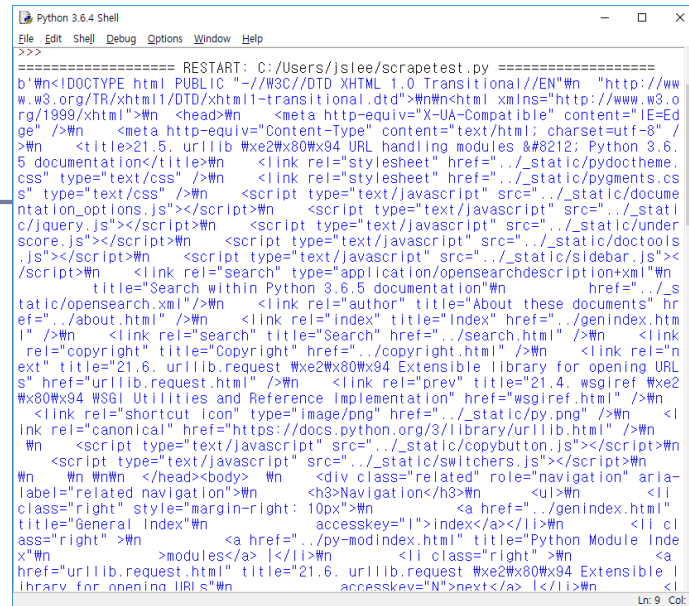
- ▶ 자~ 이렇게 url 만 바꾸어 요청하면 됩니다.

```
from urllib.request import urlopen
```

```
htmlObj = urlopen("https://docs.python.org/3/library/urllib.html")
```

```
print(htmlObj)
```

```
print(htmlObj.read())
```



```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:/Users/jslee/scrapetest.py =====
b'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml"><head><meta http-equiv="X-UA-Compatible" content="IE=Edge" /><meta http-equiv="Content-Type" content="text/html; charset=utf-8" /><title>21.6. urllib.request <code>urllib</code> Extensible library for opening URLs</title><link rel="stylesheet" href="._static/pydocthem.css" type="text/css" /><link rel="stylesheet" href="._static/pygments.css" type="text/css" /><script type="text/javascript" src="._static/documentation_options.js"></script><script type="text/javascript" src="._static/jquery.js"></script><script type="text/javascript" src="._static/underscore.js"></script><script type="text/javascript" src="._static/docketools.js"></script><script type="text/javascript" src="._static/search.js"></script><script type="text/javascript" src="._static/opensearchdescription.xml"></script><link rel="search" type="application/opensearchdescription+xml" title="Search within Python 3.6.5 documentation" href="._static/opensearch.xml" /><link rel="author" title="About these documents" href="._about.html" /><link rel="index" title="Index" href="._genindex.html" /><link rel="search" title="Search" href="._search.html" /><link rel="copyright" title="Copyright" href="._copyright.html" /><link rel="next" title="21.6. urllib.request <code>urllib</code> Extensible library for opening URLs" href="urllib.request.html" /><link rel="prev" title="21.4. urllib.request <code>urllib</code> Extensible library for opening URLs" href="urllib.request.html" /><link rel="canonical" href="https://docs.python.org/3/library/urllib.html" /><script type="text/javascript" src="._static/copybutton.js"></script><script type="text/javascript" src="._static/switchers.js"></script></head><body><div class="related" role="navigation" aria-label="related navigation"><h3>Navigation</h3><div class="right" style="margin-right: 10px"><a href="._genindex.html" title="General Index">Index</a><a href="._py-modindex.html" title="Python Module Index">Modules</a></div><div class="right"><a href="urllib.request.html" title="21.6. urllib.request <code>urllib</code> Extensible library for opening URLs">Next</a></div></div></body></html>'
Ln: 9 Col: 4
```


요청 시 예외 처리

신뢰할 수 있는 연결을 위한 예외처리

- ▶ 웹(www)은 깊이를 알 수 없는 바다와 같다!!!
 - ▶ 한 마디로 그 속은 엉망진창
 - ▶ 데이터 형식 다양하며
 - ▶ 서버는 수시로 다운되며
 - ▶ 태그는 종종 빠져 있기도 하다
- ▶ 위의 이유들로 언제든지 요청이 실패할 수도 있다.
- ▶ 따라서 신뢰할 수 있는 연결을 위해 **예외처리가 필요하다!!!**

실패할 수 있는 연결을 위한 예외처리

- ▶ 다음 코드 실행에 문제가 발생 할 수 있다.

```
htmlObj = urlopen("http://pythonscraping.com/pages/page.html")
```

- ▶ 서버나 페이지를 찾을 수 없는 경우
- ▶ URL 해석에서 에러가 생긴 경우

Traceback (most recent call last):

File "C:/Users/jslee/Desktop/a.py", line 5, in <module>

htmlObj = urlopen("http://pythonscraping.com/pages/page.html")

File "C:\Users\jslee\AppData\Local\Programs\Python\Python36-32\lib\urllib\request.py", line 223, in urlopen

return opener.open(url, data, timeout)

...

File "C:\Users\jslee\AppData\Local\Programs\Python\Python36-32\lib\urllib\request.py", line 570, in error

...

실패할 수 있는 연결을 위한 예외처리

- ▶ try ~ except~ else를 이용해서 비정상적 종료를 막자.

```
from urllib.request import urlopen
```

```
try :
```

```
    htmlObj = urlopen("http://pythonscraping.com/pages/page.html")
```

```
except :
```

```
    print("제대로 요청이 안됩니다")
```

```
else :
```

```
    print(htmlObj.read())
```

실패할 수 있는 연결을 위한 예외처리

- ▶ try ~ except~ else를 이용해서 비정상적 종료를 막자.

```
from urllib.request import urlopen
from urllib.error import HTTPError
```

```
try :
```

```
    htmlObj = urlopen("http://pythonscraping.com/pages/page.html")
```

```
except HTTPError as e :
```

```
    print(e)                # HTTP Error가 발생하면 404 Page Not Found
                             # 500 Internal Server Error 등의 오류
```

```
else :
```

```
    print(htmlObj.read())
```

신뢰할 수 있는 연결을 위한 예외처리

- ▶ try ~ except~ else를 이용해서 비정상적 종료를 막자.

```
from urllib.request import urlopen
from urllib.error import HTTPError
```

```
# 웹 페이지 요청 하는 함수
```

```
def reqPage(url) :
```

```
    try :
```

```
        htmlObj = urlopen(url)
```

```
    except HTTPError as e :
```

```
        print(e)
```

```
        return None
```

```
    else :
```

```
        return htmlObj
```

```
pageObj =
```

```
reqPage("http://pythonscraping.com/
pages/page.html")
```

```
if pageObj == None :
```

```
    print("페이지를 찾지 못했습니다")
```

```
else :
```

```
    print(pageObj.read())
```

더 큰 규모의 코딩을 계속할 사람은
독립적인 함수로 구현하는 습관을 길러나갑시다!!

requests 모듈로 요청하기

- ▶ requests
 - ▶ 간편한 HTTP 요청처리를 위해 사용하는 파이썬 라이브러리
 - ▶ 기본 내장 모듈이 아니므로 개발자가 따로 설치해야 함
 - ▶ 도스창에서 `pip install requests` 입력

urllib로 요청하는 것과 무엇이 다를까?

requests 모듈로 요청하기

urllib 요청과 비교하면서

```
>>> import requests as rq
```

```
>>> rq.get("http://pythonscraping.com/pages/page.html")
```

```
<Response [404]>
```

```
>>> rq.get("http://pythonscraping.com/pages/page1.html")
```

```
<Response [200]>
```

```
>>> resObj = rq.get("http://pythonscraping.com/pages/page1.html")
```

```
>>> print(resObj)
```

```
<Response [200]>
```


requests 모듈로 요청하기

urllib 요청과 비교하면서

```
>>> print(resObj.status_code)
```

```
200
```

```
>>> headDict = resObj.headers
```

```
>>> print(headDict)
```

```
{'Server': 'nginx', 'Date': 'Sun, 31 Jul 2022 09:03:23  
GMT', 'Content-Type': 'text/html', 'Content-Length':  
'361', 'Connection': 'keep-alive', 'ETag': '"234-56e3"  
'0.01', 'Last-Modified': 'Sun, 31 Jul 2022 09:03:23  
GMT', 'ETag': '"234-56e3"', 'Vary': 'Accept-Encoding',  
'Content-Encoding': 'gzip', 'PleskLin': ''}
```

```
>>> for h in headDict :  
    print(h, " => ", headDict[h])
```

```
Server => nginx  
Date => Sun, 31 Jul 2022 09:03:23 GMT  
Content-Type => text/html  
Content-Length => 361  
Connection => keep-alive  
... 뒷부분 생략 ...
```

requests 모듈로 요청하기

urllib 요청과 비교하면서

```
>>> print(resObj.text)
```

```
<html>
<head>
<title>A Useful Page</title>
</head>
<body>
<h1>An Interesting Title</h1>
<div>
Lorem ipsum dolor sit amet, consectetur
tempor incididunt ut labore et dolore
quis nostrud exercitation ullamco laboris
consequat. Duis aute irure dolor in
dolore eu fugiat nulla pariatur. Excepteur
sunt in culpa qui officia deserunt mollit
</div>
</body>
</html>
```

```
>>> print(resObj.content)
```

```
b'<html>\n<head>\n<title>A Useful
Page</title>\n</head>\n<body>\n<h1>An Interesting
Title</h1>\n<div>\nLorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod tempor
incidunt ut labore et dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.\n</div>\n</body>\n</html>\n'
```

urllib.request 와 requests 비교

	from urllib.request import urlopen	import requests as rq
요청	htmlObj = urlopen(url)	resObj = rq.get(url)
응답 코드	htmlObj.status	resObj.status_code
헤더 정보	htmlObj.getheaders()	resObj.headers
내용 보기	htmlObj.read() htmlObj.read().decode("utf-8")	resObj.text resObj.content

웹 데이터의 저작권 문제

- ▶ 동의 없이 자료를 가져오는 것은 저작권 문제를 일으킬 수 있다!!
- ▶ 웹 페이지에 대한 저작권법
 - 단순 링크(사이트의 대표 주소 링크) 허용
 - 직접 링크(특정 게시물 링크) 허용
 - 프레임 링크나 저작물 전체를 홈페이지에 포함(임베디드) 시키는 것은 저작권 위반으로 판단
 - 페이지의 맨 아래(footer)에 'All Rights Reserved'
- ▶ “빅 데이터는 싸지만 변호사 비는 비싸다”

웹 데이터의 저작권 문제

- ▶ 봇(크롤러)이 접근하는 것을 제한 하기 위한 규약
 - 웹 사이트의 루트(root)에 **robots.txt**을 지정 : 로봇 제외 표준,1994
- ▶ robots.txt 의 예

모두 허용	User-agent: * Allow: /
모두 차단	User-agent: * Disallow: /
다른 경우	User-agent: googlebot #googlebot 만 허용 Disallow: /bbs/ #bbs 디렉토리 접근 차단

파이썬 객체로 변환 : 파싱(parsing)

구슬이 서말이라도 꿰어야 보배!!



BeautifulSoup를 이용한 파싱

▶ 파싱(parsing)

- 요청 결과 html 문자열을 파이썬 객체로 변환하는 것
- 태그, 속성, 텍스트 정보 등을 분리하여 탐색이 가능하도록 함

▶ bs4 모듈

- ▶ 파이썬에서 제공하는 파싱 모듈
- ▶ 잘못된 HTML을 수정하여 쉽게 탐색할 수 있도록 XML(eXtensible Markup Language)형식의 파이썬 객체로 변환해 준다
- ▶ 표준 내장 모듈이 아니므로 설치해서 사용
- ▶ bs 3.x는 파이썬 2.x버전만 지원, **bs4**를 통해 파이썬 3.x를 지원

다음 코드부터 실행해 보자~

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

htmlObj = urlopen("http://pythonscraping.com/pages/page1.html")

bsObj = BeautifulSoup(htmlObj, "html.parser")
#bsObj = BeautifulSoup(htmlObj.read(), "html.parser")
```

- ▶ BeautifulSoup()의 첫 번째 인자 : 파싱 대상
- ▶ BeautifulSoup()의 두 번째 인자 : 파서의 종류

BeautifulSoup을 입력할 때의 주의 사항

- ▶ 설치 시, 대소문자 무관

- ▶ `pip install beautifulsoup4`
- ▶ `pip install bs4`

- ▶ импорт할 때, 첫 글자는 대문자

- ▶ `from bs4 import BeautifulSoup`

- ▶ 코딩 시, 첫 글자는 대문자

- ▶ `from bs4 import BeautifulSoup`
- ▶ `bsObj = BeautifulSoup(html, 'html.parser')`

파이썬이 지원하는 파서

- ▶ 파이썬은 **lxml**, **html5lib**, **html.parser** 세 가지를 제공!!

lxml	html5lib
c 언어로 구현되어 빠르다 XML, HTML 파싱 가능 html, body 태그가 없으면 추가 생성 파이썬 2.x, 3.x 모두 지원 파서를 지정하지 않으면 기본으로 인식	python 으로 구현 lxml에 비해 느리다 html, head, body 없으면 추가 생성

- html.parser 는 두 파서의 중간 속도이며, 파이썬 하위 버전까지 지원한다.
- 파서를 지정하지 않았을 때, lxml이 설치되어 있지 않으면 html.parser가 사용된다.

파싱 결과 출력하기 `print(bsObj)`

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

htmlObj = urlopen("http://pythonscraping.com/pages/page1.html")

bsObj = BeautifulSoup(htmlObj, "html.parser")

print(bsObj)
```

* 파싱하기 전과 무엇이 달라졌을까?

파싱 결과 출력 : print(bsObj)

```
<html>
<head>
<title>A Useful Page</title>
</head>
<body>
<h1>An Interesting Title</h1>
<div>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</div>
</body>
</html>
```

bsObj.

- findAllNext
- findAllPrevious
- findChild
- findChildren
- findNext
- findNextSibling
- findNextSiblings
- findParent
- findParents
- findPrevious

lxml 사용해서 파싱하기

```
from bs4 import BeautifulSoup
```

```
htmlObj = """<p>Python is strong!!!</p>"""
```

```
bsObj = BeautifulSoup(htmlObj, "lxml")
```

```
print(bsObj)
```

```
<html><body><p>Python is strong!!!</p></body></html>
```

* 우리는 지금부터 lxml 파서를 사용하겠습니다~

파싱한 결과 파이썬 객체의 구조

```
<html>  
  <head>  
    <title>A Useful Page</title>  
  </head>  
  <body>  
    <h1>An Interesting Title</h1>  
    <div> .....</div>  
  </body>  
</html>
```

파싱 후 원하는 태그 탐색하기

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

```
htmlObj =  
urlopen("http://pythonscraping.com/pages/page1.html")
```

```
bsObj = BeautifulSoup(htmlObj, "lxml")
```

```
print(bsObj.h1)
```

```
<h1>An Interesting Title</h1>
```

bsObj.h1 대신

bsObj.html.body.h1

bsObj.body.h1

bsObj.html.h1

모두 가능하다.
여기서 파이썬의 단순하면서도 강력한 면을 엿볼 수 있다!!!

파싱 후 원하는 태그 탐색하기 : **find()**

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

htmlObj =
urlopen("http://pythonscraping.com/pages/page1.html")

bsObj = BeautifulSoup(htmlObj, "lxml")

print(bsObj.find('h1'))
```

```
<h1>An Interesting Title</h1>
```


파싱 후 원하는 태그 탐색하기 : **find()**

<https://madelen2016.github.io/class/index.html>에서
<a> 태그를 탐색해 봅시다~

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

htmlObj =
urlopen("https://madelen2016.github.io/class/index.html ")

bsObj = BeautifulSoup(htmlObj, "lxml")

print( bsObj.find('a') )
```

```
<a href="01 intro.pdf"> - 빅데이터 개요 </a>
```

파싱 후 원하는 태그 탐색하기 : **find_all()**

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

htmlObj = urlopen("
https://madelen2016.github.io/class/index.html ")

bsObj = BeautifulSoup(htmlObj, "lxml")
print( bsObj.find_all('a') )
```

[- 빅데이터 개요 , - 빅데이터 처리를 위한 파이썬 기초 1 , - 빅데이터 처리를 위한 파이썬 기초 2 , - 빅데이터 처리를 위한 파이썬 기초 3 , - 크롤링 기초 1 , - 크롤링 기초 2 , - 크롤링에서 시각화까지 , - 파이썬을 이용한 데이터 수집과 분석 **]**

파싱 후 원하는 태그 탐색하기 : **find_all()**

or findAll()

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

```
htmlObj = urlopen("  
https://madelen2016.github.io/class/index.html")  
bsObj = BeautifulSoup(htmlObj, "lxml")
```

```
findList = bsObj.find_all('a')      #검색 결과 리스트 객체 반환
```

```
for i in findList :  
    print(i)
```

```
<a href="01 intro.pdf"> - 빅데이터 개요 </a>  
<a href="02 python basic1.pdf"> - 빅데이터 처리를 위한 파이썬 기초 1 </a>  
<a href="03 python basic2.pdf"> - 빅데이터 처리를 위한 파이썬 기초 2 </a>  
<a href="04 python basic3.pdf"> - 빅데이터 처리를 위한 파이썬 기초 3 </a>  
<a href="05 openAPI.pdf"> - 크롤링 기초 1 </a>  
<a href="06 crawling1.pdf"> - 크롤링 기초 2 </a>  
<a href="07 crawling2.pdf"> - 크롤링에서 시각화까지 </a>  
<a href="08 crawling_analysis.pdf"> - 파이썬을 이용한 데이터 수집과 분석  
</a>
```

잠깐~ 다음의 차이에 대해 정리 좀 할까요

- ▶ `print(bsObj.html.body.h1)`
- ▶ `print(bsObj.find("h1"))`
- ▶ `print(bsObj.find_all('h1'))`
- ▶ `print(bsObj.select("body > h1"))`

세것아함게가면
힘한길도즐겁다

서호남글 서지  