2. 웹 크롤링 & 스크래핑

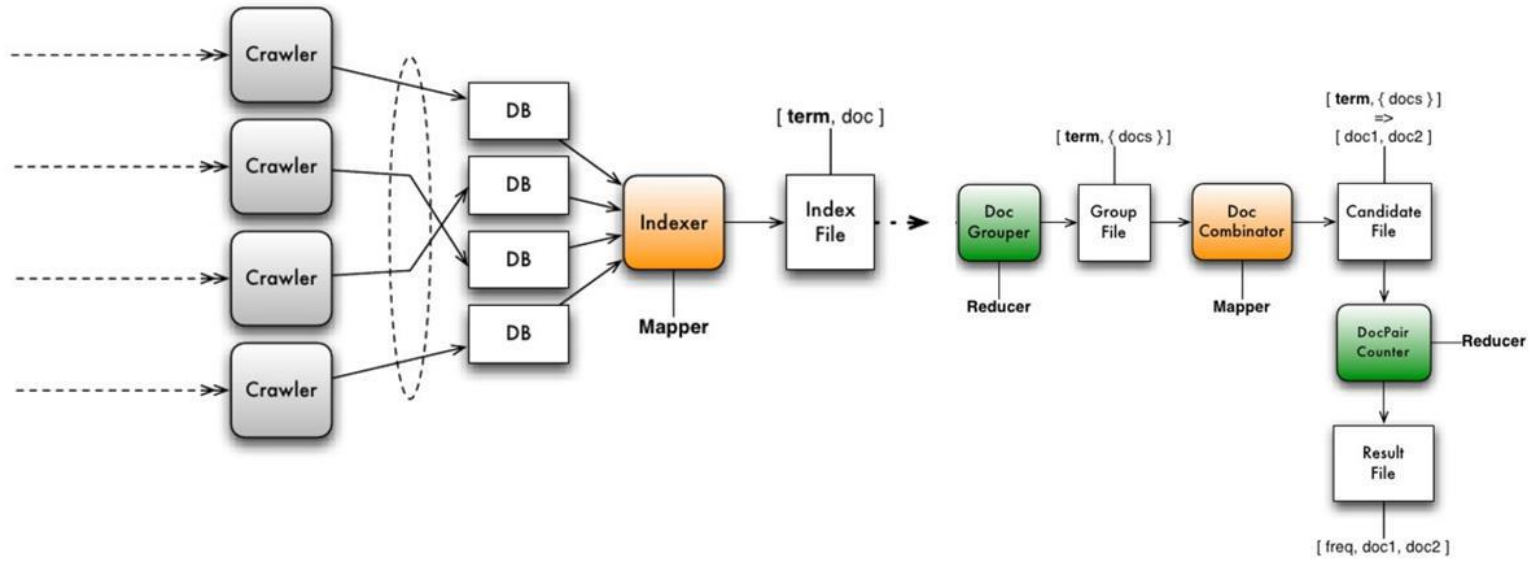
1. 웹 크로링 & 스크래핑 개요
2. 웹 페이지를 구성하는 기술
3. 웹 페이지 콘텐츠 요청
4. HTML 파싱하기
5. 웹 사이트 웹크롤링 실습

1. 웹 크롤링 & 스크래핑 개요

- **웹 스크래핑(Web scraping)**
  - 웹 페이지 상에서 원하는 콘텐츠 정보를 컴퓨터로 하여금 자동으로 추출하여 수집하도록 하는 기술
  - 웹 페이지를 구성하고 있는 HTML 태그의 콘텐츠나 속성의 값을 읽어오는 작업
- **웹 크롤링(web crawling)**
  - 자동화 봇(bot)인 웹 크롤러(web crawler)가 정해진 규칙에 따라 복수 개의 웹 페이지를 브라우징하는 작업
- **Python의 웹 스크래핑 라이브러리**
  - BeautifulSoup
  - scrapy

1. 웹 크롤링 & 스크래핑 개요

빅데이터의 수집, 분석, 시각화 과정



- Fetching : 인터넷 상에 산재해있는 데이터를 수집하기 위해 크롤러(Crawler) 구성
- Storing : 수집한 정보는 데이터 베이스 또는 HDFS(Hadoop File System)에 저장
- 인덱싱(Indexind) : 저장된 데이터는 검색의 효율성을 높이기 위하여 인덱싱(Indexing) 과정을 거침
- Data Mart : 데이터를 가공하여 축약된 정보 DB(Mart DB)를 생성.
- 시각화(Visualization) : 정보를 사용자에게 효율적으로 보여주기 위해 시각화(Visualizatino) 과정을 거쳐 인포그래픽(Infographic-정보를 포함한 그래픽)으로 최종적으로 생성해 내는 것

1. 웹 크롤링 & 스크래핑 개요

Crawling 저작권

“ 허용 ”

단순 링크 - 사이트 대표 주소를 링크
직접 링크 – 특정 게시물을 링크

“ 위반 ”

프레임 링크 - 저작물의 일부를 홈페이지에 표시
임베드 링크 – 저작물 전체를 홈페이지에 표시

“ 크롤링 배제 ”

웹 사이트에 로봇(크롤러)이 접근하는 것을 방지하기 위한 규약

Site	Status
모두 허용	User-agent:* Allow: /
모두 차단	User-agent:* Disallow: /
다른 예	User-agent: googlebot #googlebot 크롤러만 적용 Disallow: /bbs/ #/bbs 디렉터리 접근 차단

1. 웹 크롤링 & 스크래핑 개요

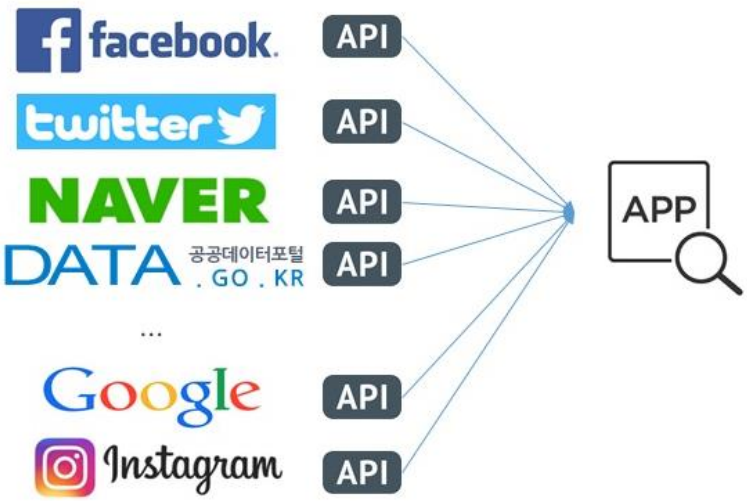
□ Portal/SNS 별 Crawling 방법 및 한계

- "robots.txt" 파일 : 무분별한 인터넷 데이터의 사용을 금지하기 위하여 웹 규약에서는 해당 사이트의 크롤링 범위를 정의하는 "robots.txt" 파일을 서비스 웹 서비스의 root에 저장하도록 되어 있다.
- 예 : http://www.naver.com/robots.txt 에 웹 브리우저를 이용하여 접근해 보면 네이버의 크롤러 접근 규칙에 대해 확인이 가능

Site	Status
NAVER	• SCRAPY 등의 Crawler를 이용한 데이터 수집 불가 (http://www.naver.com/robots.txt → User-agent: * Disallow: /) • API: 최대 100건의 데이터 검색 가능 (25,000/일)
DAUM	• SCRAPY 등의 Crawler를 이용한 데이터 수집 불가 (http://search.daum.net/robots.txt → User-agent: * Disallow: /) • API : 기존에 최대 500page까지 검색이 가능하였으나 현재 최대 3페이지까지 검색 가능
KAKAO	• API : 별도의 검색 API 제공하지 않음
FACEBOOK	• API : Graph API를 통하여 특정 페이지 수집 가능

1. 웹 크롤링 & 스크래핑 개요

- SNS API(Application Programming Interface)
  - SaaS(Software as a Service)
    - 소프트웨어 및 관련 데이터가 중앙에 위치하고 사용자는 웹 브라우저등을 이용하여 접속하여 소프트웨어를 사용하는 서비스 모델
    - 페이스북이나 트위터는 자신들이 서비스하는 여러 가지 기능을 SaaS의 개념으로 다른 사용자들이 활용할 수 있는 API(Application Programming Interface)를 제공



SaaS : 크롤러(Crawler)를 만들고 URL에 접근하여 데이터를 가져오는 수고를 들어줌

2. 웹페이지 구성 기술

HTML(HyperText Markup Language)

- 웹 페이지를 만들 때 사용하는 마크업 언어
  ➡ 태그를 사용하여 내용 작성

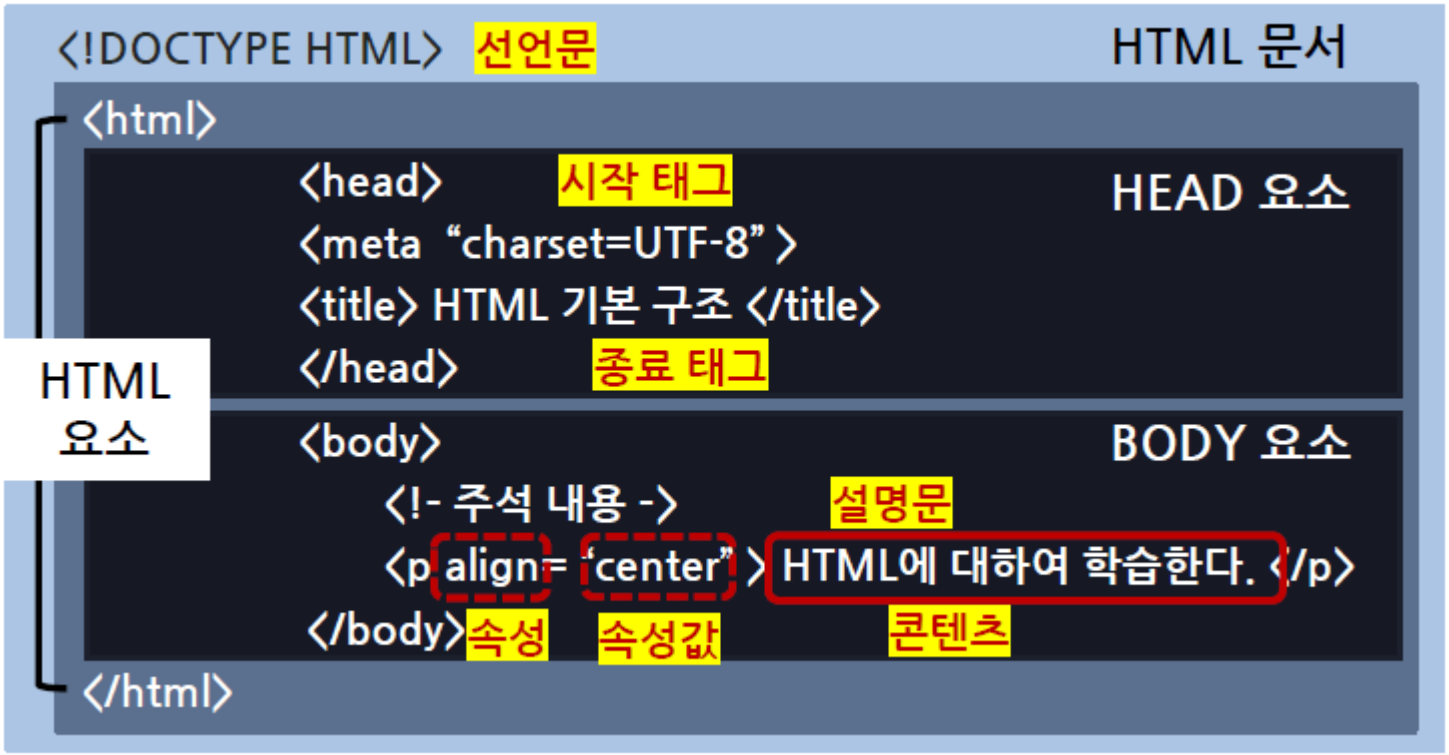
- 전체적으로 <html> 태그로 감싸 짐
- 문서의 정보를 제공하는 <head> 태그와 브라우저에 렌더링되는 내용을 작성하는 <body> 태그로 구성

웹 페이지 데이터 추출

- 추출하려는 콘텐츠의 태그를 찾아서 속성의 값이나 콘텐츠 부분을 추출하는 것

HTML(HyperText Markup Language)

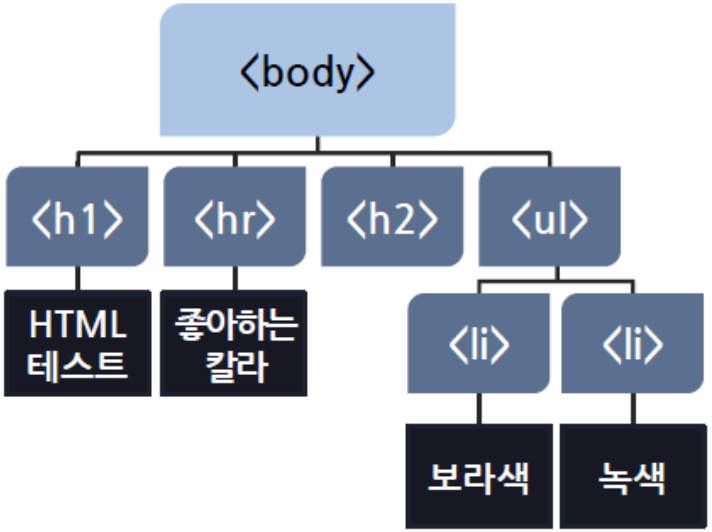
HTML 구성 요소



HTML(HyperText Markup Language)

- 브라우저가 HTML 문서를 파싱하여 브라우저의 도큐먼트 영역에 렌더링할 때 HTML 문서를 구성하는 모든 태그와 속성, 콘텐츠들을 DOM(Document Object Model)이라는 규격을 적용하여 JavaScript 객체 생성
- HTML 문서의 내용으로 구성되는 DOM 객체들은 HTML 문서 그대로 계층구조를 이룸

```
<head>
  <meta charset="utf-8">
</head>
<body>
  <h1>HTML 테스트</h1>
  <hr>
  <h2>좋아하는 칼라</h2>
  <ul>
   <li>보라색</il>
   <li>보라색</il>
  </ul>
 <body>
</html>
```



CSS(Cascade Style Sheet)

- HTML과 같은 마크업 언어가 브라우저에 표시되는 방법을 기술하는 언어
- HTML과 XHTML에 주로 사용되는 W3C의 표준

CSS 사용의이점

- 웹 표준에 기반한 웹 사이트 개발 가능(페이지의 내용과 디자인 분리)
- 클라이언트 기기에 알맞는 반응형 웹 페이지 개발 가능
- 이미지의 사용을 최소화시켜 가벼운 웹 페이지 개발 가능

CSS(Cascade Style Sheet)

```
선택자 {
    CSS속성명 : CSS속성값;
    CSS속성명 : CSS속성값;
        :
}
```

```
<style>
  h1{
    color : red:
    background-color : yellow;
    width : 200px;
    border : 3px solid magenta;
    border-radius : 10px;
    padding : 3px
    text-align : center;
}
h2{
    color : blue;
    text-shadow : 2px 2px 2px skyblue;
}
</style>
```

HTML 문서를 CSS 없이 렌더링한 경우

HTML 문서를 CSS를 적용하여 렌더링한 경우

CSS 선택자

CSS 선택자 (Selector)	▪ 스타일을 적용하기 위해 대상 태그를 선택하는 방법

태그 선택자

- 태그명으로태그를선택하려는경우로태그명을그대로사용

```
h2 { color : blue; }
```

```
<h2> CSS(Cascade Style Sheet) </h2>
```

CSS 선택자

클래스선택자

- 태그에정의된class 속성의값으로태그를선택하려는경우로. 과함께작성

```
.redtext { color : red; }
```

```
<h2  class="redtext"> CSS(Cascade Style Sheet) </h2>
```

id 선택자

- 태그에정의된id 속성의값으로태그를선택하려는경우로#과함께작성

```
#t1 { color : green; }
```

```
<h2 id="t1"> CSS(Cascade Style Sheet)  </h2>
```

CSS 선택자

자식 선택자

- 지정된 부모 태그의 자식 태그에만 스타일이 적용

```
section > p { color : blue; }
```

```
<section>
  <p>
   선택됨
  </p>
</section>
```

```
<nav>
  <p>
   선택되지 않음
  </p>
</nav>
```

CSS 선택자

자손 선택자

- 지정된 부모 태그의 자식 태그에만 스타일이 적용

```
div  p { color : yellow; }
```

```
<div>
  <p>
   선택됨
  </p>
  <section>
    <p>
     선택됨
    <p>
</section>
</div>
```

CSS 선택자

- 속성선택자

```
img[src=duke.png] { radius : 0.5; }
```

```
<img src="duke.png">
```

JavaScript

JavaScript	▪ 스크립트 방식으로 구현되는 OOP 프로그래밍 언어 ▪ 최근에는 다양한 기능의 프로그래밍에 사용 가능해짐 ▪ 주로 웹 페이지 개발 시 동적인 처리를 구현하기 위해 사용
JavaScript 코드	▪ <script> 태그와 함께 HTML 문서 내에 작성 ▪ xxx.js 라는 독립된 파일로 만들어 <script> 태그를 이용하여 HTML 문서에서 호출하는 방식으로 작성

JavaScript

정적 콘텐츠로 구성된 웹 페이지

- HTML 태그와 CSS만으로 구성되는 웹 페이지는 간단하게 콘텐츠 추출 가능

동적 컨텐츠로 구성된 웹 페이지

- JavaScript를 이용하여 웹 페이지의 콘텐츠가 동적으로 구성되는 경우, Selenium 같은 기술을 추가로 사용해야 함

- **웹 크롤링을 할때는 크롤링하려는 콘텐츠 부분이 정적으로 만들어진 것인지 JavaScript에 의해서 동적으로 만들어지는 것인지부터 파악**

Ajax

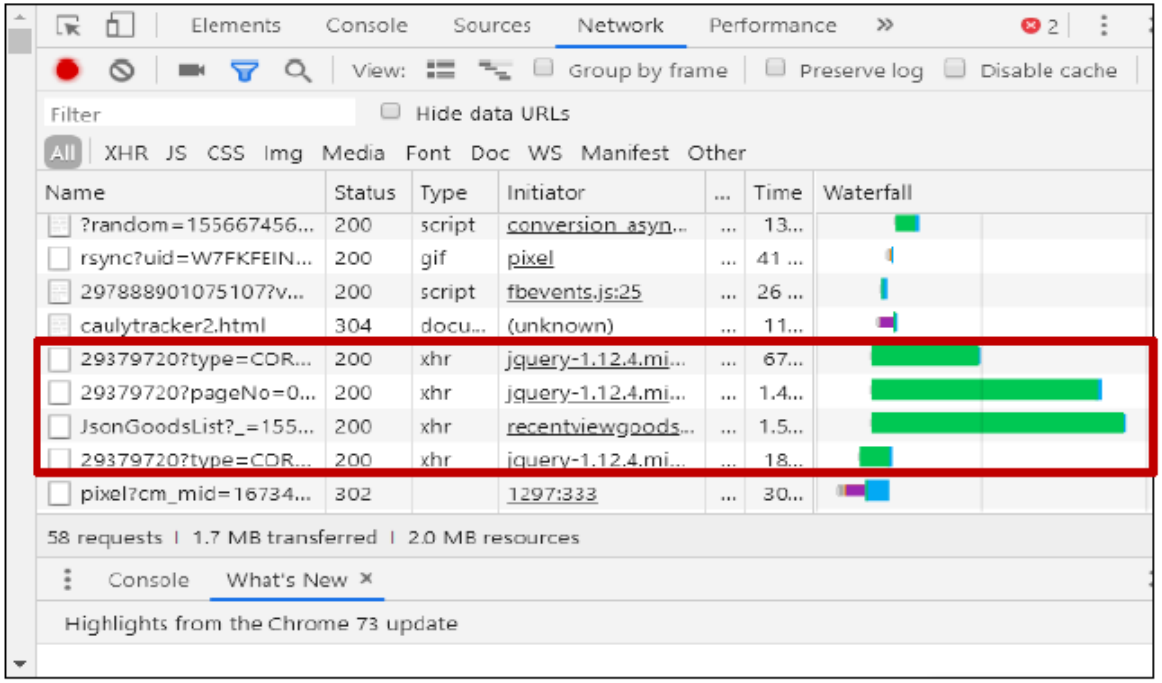
- Asynchronous JavaScript and XML의약어
- JavaScript 코드로 서버와 통신하는 기술
- 통신방식을 비동기적으로 처리하여 요청하고나서 대기하지 않고 다른작업을 처리할 수 있음
- 전체페이지가 아닌 필요한 일부분만 요청하여 받아올 수 있는 통신

1. 전체 페이지를 리로드(Refresh)하지 않고 보여지고 있는 현재페이지 내에 서버로부터 받아온 내용을 자연스럽게 추가할 수 있음
2. 필요한 만큼의 일부 데이터만 요청하여 받아 빠르게 동적 웹페이지 생성
3. 데이터 전송량이 중요한 모바일웹과 최근 많이 활용되는 SPA(Single Page Application)에서 더욱 중요해진 통신 기술

Ajax

Ajax 기술을사용한페이지확인방법

- 크롬브라우저의개발자도구를열고네트워크탭선택
- 네트워크탭에서 웹브라우저와 웹 서버간 통신상태 정보 출력
- 확인하고자하는 웹 페이지 요청
- 웹 페이지의 렌더링이 끝난 후 개발자도구의 통신상태 정보 리스트에서 통신 Type열을 체크했을때 xhr로 출력되는 통신 Ajax를 이용한 통신



3. 웹 콘텐츠 요청

- **주요 모듈**

urllib	▪ URL 작업을 위한 여러 모듈을 모은 패키지 ▪ 파이썬의 표준 라이브러리

URL 문자열과 웹 요청에 관련된 모듈 5개 제공

- urllib.request—URL 문자열을 가지고 요청 기능 제공
- urllib.response—urllib모듈에 의해 사용되는 응답 클래스들 제공
- urllib.parse—URL 문자열을 파싱하여 해석하는 기능 제공
- urllib.error—urllib.request에 의해 발생하는 예외 클래스들 제공
- urllib.robotparser—robots.txt 파일을 구문 분석하는 기능 제공

URL 문자열을 가지고 HTTP 요청을 수행하는 urllib.request 모듈

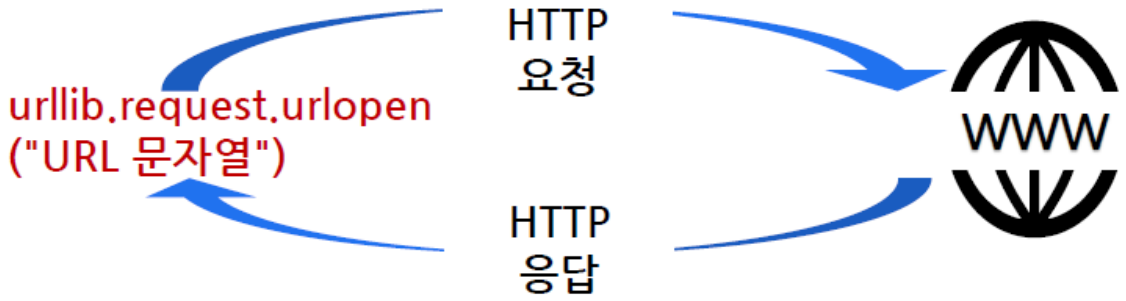
URL 문자열(주소)을 해석하는 urllib.parse 모듈

주요 모듈 활용

urllib.request모듈

- URL 문자열을 가지고 HTTP 요청을 수행
- urlopen() 함수를 사용하여 웹 서버에 페이지를 요청하고, 서버로부터 받은 응답을 저장하여 응답 객체(http.client.HTTPResponse)를 반환

```
res = urllib.request.urlopen
("요청하려는 페이지의 URL 문자열")
```



주요 모듈 활용

urllib.request모듈

- 웹 서버로부터 받은 응답을 래핑하는 객체
- 응답 헤더나 응답 바디의 내용을 추출하는 메서드 제공

- HTTPResponse.read([amt])
- HTTPResponse.readinto(b)
- HTTPResponse.getheader(name, default=None)
- HTTPResponse.getheaders()
- HTTPResponse.msg
- HTTPResponse.version
- HTTPResponse.status
- HTTPResponse.reason
- HTTPResponse.closed

주요 모듈 활용

http.client.HTTPResponse 객체의read() 메서드

- read() 메서드를 실행하면 웹 서버가 전달한 데이터(응답 바디)를 바이트열로 읽어 들임

바이트열	
	16진수로 이루어진 수열이기 때문에 읽기 어려우므로 웹 서버가 보낸 한글을 포함한 텍스트 형식의 HTML 문서의 내용을 읽을 때는 텍스트 형식으로 변환함
	바이트열(bytes)의 decode('문자 셋') 메서드를 실행하여 응답된 문자 셋에 알맞은 문자로 변환함

주요 모듈 활동

http.client.HTTPResponse 객체의read() 메서드

res.read()

<body>₩r₩n<h1>₩xea₩
xb0₩x80₩xeb₩x82₩x98
₩xeb₩x8b₩xa4ABC</h1>
₩r₩n</body>

res.read().decode('utf-8')

<body>
<h1>가나다ABC</h1>
</body>

실습

```
#html 소스 읽기
from urllib.request import urlopen
html=urlopen("http://google.com")
print(html.read())
```

```
#예외 처리 방법
from urllib.error import HTTPError
from urllib.error import URLError

try:
    html=urlopen("https://java.com")
except HTTPError as e:
    print("HTTP 에러입니다")
except URLError as e:
    print("존재하지 않는 사이트입니다")
else:
    print(html.read())
```

이미지 다운로드

```
# 이미지 다운로드 방법1
import urllib.request
#이미지 주소 복사
url="http://image.iacstatic.co.kr/allkill/item/2019/06/20190603102925421r.jpg"
savename="d:/data/images/test2.jpg"
urllib.request.urlretrieve(url,savename)
print("저장되었습니다....")
```

```
# 이미지 다운로드 방법2
import urllib.request
#이미지 주소 복사
url="http://image.iacstatic.co.kr/allkill/item/2019/06/20190603102925421r.jpg"
savename="d:/data/images/test3.jpg"
image=urllib.request.urlopen(url).read()
with open(savename,mode="wb") as f:
    f.write(image)
    print("저장되었습니다")
```

```
#매개변수를 추가하여 인터넷 리소스를 요청하는 방법
import urllib.request
import urllib.parse
API="http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
# 지역번호 : 전국 108, 서울/경기 109, 강원 105, 충북 131 충남 133,
# 전북 146, 전남 156, 경북 143, 경남 159, 제주 184
values={'stnId':'108'}
params=urllib.parse.urlencode(values)
#요청 전용 url 생성
url=API+"?"+params

print("url=",url)
#다운로드
data=urllib.request.urlopen(url).read()
text=data.decode('utf-8')
print(text)
```

```
# 스크래핑(scraping): 웹상의 정보를 추출하는 것
# beautitflSoup :HTML 파싱 라이브러리
# 사이트 : https://www.crummy.com/software/BeautifulSoup/
# pip install beautifulsoup4
from urllib.request import urlopen
from bs4 import BeautifulSoup
html=urlopen("http://stackoverflow.com")
bs=BeautifulSoup(html.read(), 'html.parser')
print(bs.title)
print(bs.title.text)
print(bs.h1)
print(bs.h1.text)
print(bs.span)
```

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html='''
<html><body>
<h1>Hello python</h1>
<p>웹 페이지 분석</p>
<p>웹 스크래핑</p>
</body></html>
'''
soup=BeautifulSoup(html,'html.parser')
h1=soup.html.body.h1
p1=soup.html.body.p

p2=p1.next_sibling.next_sibling
print("h1=",h1)
print("h1=",h1.string)
print("p=",p1)
print("p=",p1.string)
print("p=",p2)
print("p=",p2.string)
```

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html='''
<html><body>
<h1 id="title">Hello python</h1>
<p id="body">웹 페이지 분석</p>
<p>웹 스크래핑</p>
<span>데이터 수집1</span>
<span>데이터 수집2</span>
</body></html>
'''
soup=BeautifulSoup(html,'html.parser')
title=soup.find(id='title')
body=soup.find(id='body')
span=soup.find('span')

print('#title=',title)
print('#title='+title.string)
print('#body=',body)
print('#body=',body.string)
print('span=',span.string)
```

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html='''
<html><body>
<ul>
    <li><a href="http://naver.com">naver</a></li>
    <li><a href="http://daum.net">daum</a></li>
    <li><a href="http://nate.com">nate</a></li>
    <li><a href="http://google.com">google</a></li>
    <li><a href="http://yahoo.com">yahoo</a></li>
</ul>
</body></html>
'''
soup=BeautifulSoup(html,'html.parser')
links=soup.find_all("a")
print(links)
print(type(links))
for a in links:
    href=a.attrs['href']
    text=a.string
    print(text, ":", href)
```

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html=urlopen("http://www.auction.co.kr")
soup=BeautifulSoup(html.read(),'html.parser')
links=soup.find_all('a')
for a in links:
    try:
        href=a.attrs['href']
        text=a.string
        print(text, ":", href)
    except:
        pass
```

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import urllib.request as req
url="http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
res=req.urlopen(url)
soup=BeautifulSoup(res,'html.parser')
title=soup.find("title").string
wf=soup.find("wf").string
print(title)
print(wf)
```

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html='''
<html><body>
<h1>테스트</h1>
<div>div1</div>
<div>div2</div>
<div id="main">
<h1 i>도서 목록</h1>
<ul class="items">
   <li>자바 프로그래밍 입문</li>
   <li>HTML5</li>
   <li>Python</li>
</ul>
</div>
</body></html>'''
soup=BeautifulSoup(html,'html.parser')
h1=soup.select_one("div#main > h1").string
print("h1=",h1)
li_list=soup.select("div#main > ul.items >li")
for li in li_list:
    print("li=",li.string)
```

```
# 네이버 금융에서 환율정보 가져오기
from urllib.request import urlopen
from bs4 import BeautifulSoup
import urllib.request as req
url="https://finance.naver.com/marketindex/?tabSel=exchange" # 네이버 금융으로 검색
res=req.urlopen(url)
soup=BeautifulSoup(res, "html.parser")
price=soup.select_one("div.head_info > span.value").string
print("usd/krw=",price)
```

웹스크랩 1

```
# 웹브라우즈의 F12 개발자 도구, Elements에서 원하는 부분을 선택한 후 우클릭
# Copy->copy Selector
# 윤동주의 작품 목록 스크랩

from bs4 import BeautifulSoup
import urllib.request as req

# 뒤의 인코딩 부분은 "저자 : 윤동주"라는 의미입니다.
# 위키 문헌 홈페이지에 들어간 뒤에서 주소를 복사해서 사용

url="https://ko.wikisource.org/wiki/%EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F
%99%EC%A3%BC"
res=req.urlopen(url)
soup=BeautifulSoup(res,"html.parser")
li_list=soup.select(".mw-parser-output>ul>li")
#print(li_list)
for title in li_list:
    print(title.a.string)
    li=title.find_all('li')
    for i in li:
        print(" -",i.a.string)
```

war_andd Peace

```
from bs4 import BeautifulSoup
from urllib.request import urlopen
html=urlopen("http://www.pythonscraping.com/pages/warandpeace.html")
bs=BeautifulSoup(html,'html.parser')

#nameList=bs.select('span.green')
nameList=bs.findAll('span',{'class':'green'})

for name in nameList:
    print(name.get_text())
```

```
titles=bs.find_all(['h1','h2','h3','h4','h5','h6'])
print([title for title in titles])
```

table 데이터 스크랩

```
from bs4 import BeautifulSoup
from urllib.request import urlopen
html=urlopen("http://www.pythonscraping.com/pages/page3.html")
bs=BeautifulSoup(html,'html.parser')

for child in bs.find('table',{'id':'giftList'}).children: # 하위 태그들으 모두 가져옴
    print(child)
```

```
# 시블링(sibling, 형제자매)
# next_sibling : 다음 형제 노드
# previous_sibling: 이전 형제 노드
# 제목행은 제외하고 탐색하게 됨
for sibling in bs.find('table',{'id':'giftList'}).tr.next_siblings:
    print(sibling)
```

```
img_urls=bs.find_all('img')
print(img_urls)
print(bs.find('img',{'src':'../img/gifts/img1.jpg'}) # 이미지가 있는 td 구함
      .parent.previous_sibling.get_text())
```

pdf 파일 스크랩-1

```
# pip install pdfminer3k
# pdf 문서 읽기

from pdfminer.pdfinterp import PDFResourceManager,process_pdf
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from io import StringIO
from io import open
from urllib.request import urlopen
```

pdf 파일 스크랩-2

```
def readPDF(pdfFile):
    rsrcmgr=PDFResourceManager() # pdf 리소스 관리
    retstr=StringIO() # pdf 내부의 텍스트 입출력을 위한 객
    laparams=LAParams() # 파라미터 객체
    # pdf 내용을 텍스트로 반환하기 위한 객
    device=TextConverter(rsrcmgr,retstr,laparams=laparams)
    process_pdf(rsrcmgr,device,pdfFile) #pdf 내용을 텍스트로 변환하는 작
    device.close()
    content=retstr.getvalue() # 리턴되는 스트링
    retstr.close()
    return content
pdfFile=urlopen("http://www.pythonscraping.com/pages/warandpeace/chapter1.pdf")
outputString=readPDF(pdfFile)
#print(outputString)
pdfFile.close()

with open("d:/data/pdf/result.txt",'w') as f:
    f.write(outputString)
    print("저장되었습니다")
```

스크랩 내용 csv로 저장-1

```
# 웹페이지의 내용을 분석하여 CSV 파일로 저장
# <table> 내부의 텍스트 저장
import csv
from urllib.request import urlopen
from bs4 import BeautifulSoup

html=urlopen("https://en.wikipedia.org/wiki/Comparison_of_text_editors")
bsObj=BeautifulSoup(html,'html.parser')

# class가 wiktable인 테이블들 중 첫 번째 태그 선택
table=bsObj.findAll("table",{"class":"wikitable"})[0]
rows=table.findAll("tr")
#print(rows)
```

스크랩 내용 csv로 저장-2

```
# wt: 텍스트 쓰기 모
csvFile=open("d:/data/csv/editors.csv","wt",newline='₩n',encoding='utf-8')
# csv 파일 저장 객체
writer=csv.writer(csvFile)
try:
    for row in rows:
        csvRow=[]
        # td, th 태그의 내용을 리스트에 추가
        for cell in row.findAll(['td','th']):
            # td, th 태그내의 문자열 추출
            csvRow.append(cell.get_text()) # cell.get_text().strip("₩n")
        #print(csvRow)
        writer.writerow(csvRow)
finally:
    print("csv로 저장되었습니다")
    csvFile.close()
```

웹 스크랩 내용 mysql에 저장-3

```
'''mysql Table 만들기
create table pages(
id int not null auto_increment primary key,
title varchar(200),
content text,
reg_date datatime default now()
)
'''
```

웹 스크랩 내용 mysql에 저장-2

```
from urllib.request import urlopen
from bs4 import BeautifulSoup as bs
import re
import datetime
import random
import MySQLdb
```

웹 스크랩 내용 mysql에 저장-3

```
#conn = pymysql.connect(host='localhost', user='pgm',
password='1234',db='pyboard', charset='utf8')
conn=MySQLdb.connect("localhost","pgm","1234","pyweb",charset='utf8')
cursor = conn.cursor()

random.seed(datetime.datetime.now())

def store(title,content):
    # 따옴표 처리
    #print(title)
    #print(content)
    title =title.replace("'","''") #작은 따옴표 1개를 2개
    title =title.replace('"','₩"') # 큰 따옴표를 작은 따옴표로
    content=content.replace("'","''")
    content=content.replace('"','₩"')
    sql="insert into pages(title,content) values('%s','%s')"%(title,content)
    cursor.execute(sql)
    conn.commit()
```

웹 스크랩 내용 mysql에 저장-4

```
# 위키피디아의 url 수집
def getLinks(url):
    html=urlopen("http://en.wikipedia.org"+url)
    obj=bs(html,'html.parser')
    #h1 태그의 내
    title=obj.find('h1').get_text()
    # id가 mw-content-text인 div 태그 부분의 p태그의 텍스
    content=obj.find("div",{"id":"mw-content-text"}).find("p").get_text()
    #테이블에 저
    store(title,content)
    #링크를 리턴함(정규표현식 사용, ^시작, $ 끝, *0회 이상 반)
    return obj.find("div",{"id":"bodyContent"})
            .findAll("a",href=re.compile("^(/wiki/)((?!:.).)*$"))
```

웹 스크랩 내용 mysql에 저장-5

```
links=getLinks("/wiki/Kevin_Bacon")
#print(links)
try:
    count=0
    while len(links)> 0 :
        newArticle=links[random.randint(0, len(links)-1)].attrs['href']
        print(newArticle)
        links=getLinks(newArticle)
        count+=1
        if count >=5:
            break
finally:
    cursor.close()
    conn.close()
    print("완료되었습니다")
```

스크린 샷

```
#웹사이트 캡쳐
# phantomjs 다운로드
# pip install selenium
# 화면이 없고 커맨드라인상에서 웹브라우저를 제어할 수 있는 브라우저
# 웹개발 테스트 및 성능 측정, 스크린샷 등에 활용되고 있음
# http://phantomjs.org/download.html
# phantomjs-2.1.1-windows.zip (17.4 MB) download
# d:/phantomjs 디렉토리에 압축 해제

# 사이트 화면을 이미지로 캡쳐하기
from selenium import webdriver
from selenium.webdriver.remote.webelement import WebElement
from selenium.webdriver import ActionChains
# 드라이버 로등
driver=webdriver.PhantomJS(executable_path="d:/phantomjs/bin/phantomjs")
# 컨텐츠를 불러올 때가지 최대 5초까지 기다림
driver.implicitly_wait(5)
driver.get("http://daum.net")
# 스크린샷을 이미지 파일로 저
driver.get_screenshot_as_file('d:/data/images/screen.png')
driver.close()
print("저장되었습니다")
```

실습 – 교보문고

```
import urllib.parse
from urllib.request import urlopen
from bs4 import BeautifulSoup
html=urlopen("http://www.kyobobook.co.kr/search/SearchKorbookMain.jsp?v
PstrCategory=KOR&vPstrKeyWord=%26%2351088%3B%26%2348148%3B&v
Pplace=top")

bsObj=BeautifulSoup(html,"html.parser")
titles=bsObj.select("td.detail > div.title > a > strong")
print(len(titles),"권이 검색되었습니다")
print("="*20)
for title in titles:
    print(title.string.strip())
```

실습하기

- 네이버 영화페이지
- 상영작.예정작 영화정보 스크랩하기