

2. 웹크롤링



차례

1. 웹 크로링 & 스크래핑 개요
2. 웹 페이지를 구성하는 기술
3. 웹 페이지 콘텐츠 요청- urllib 패키지 활용
4. 웹 페이지 콘텐츠 요청- requests 패키지 활용

1. 웹 크롤링 & 스크래핑 개요

■ 크롤링이란

■ 크롤링

- 웹에서 데이터를 수집하는 작업, 크롤러 또는 스파이더라는 프로그램으로 웹 사이트에서 데이터를 추출

■ 웹 API

- 웹 API는 일반적으로 HTTP 통신을 사용하는데 사용
- 지도, 검색, 주가, 환율 등 다양한 정보를 가지고 있는 웹 사이트의 기능을 외부에서 쉽게 사용할 수 있도록 사용 절차와 규약을 정의한 것



웹 API 제공자

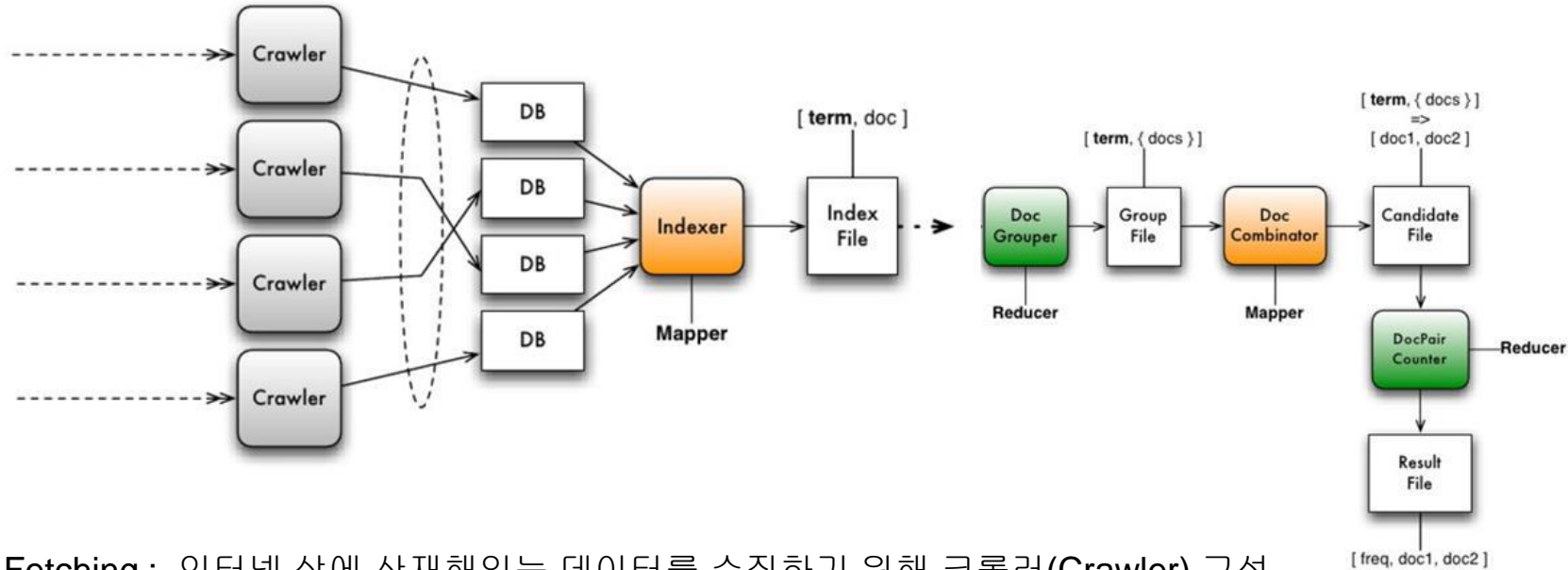
종류	주소
네이버 개발자 센터	https://developers.naver.com
카카오 앱 개발 플랫폼 서비스	https://developers.kakao.com
페이스북 개발자 센터	https://developers.facebook.com
트위터 개발자 센터	https://developer.twitter.com
공공데이터포털	https://www.data.go.kr
세계 날씨	http://openweathermap.org
유료/무료 API 스토어	http://mashup.or.kr http://www.apistore.co.kr/api/apiList.do

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
 - selenium

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(Indexing)** : 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(Visualizatio) 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

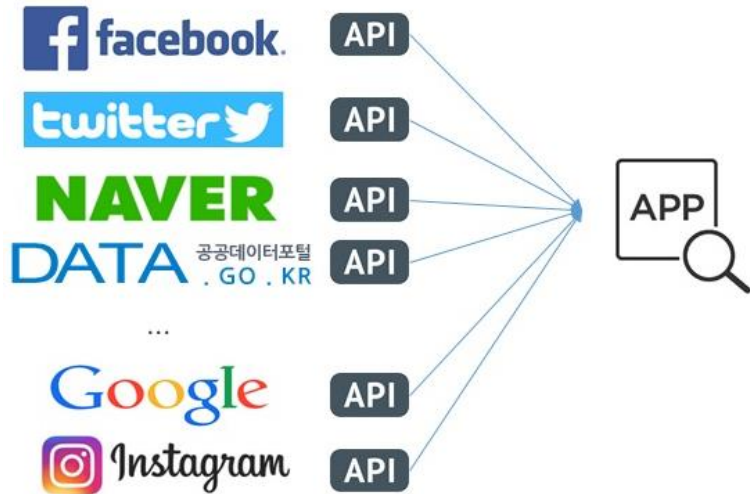
■ Portal/SNS 별 Crawling 방법 및 한계

- “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 브라우저를 이용하여 접근해 보면 네이버의 크롤러 접근 규칙에 대해 확인이 가능

Site	Status
NAVER	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• API: 별도의 검색 API 제공하지 않음
FACEBOOK	<ul style="list-style-type: none">• 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

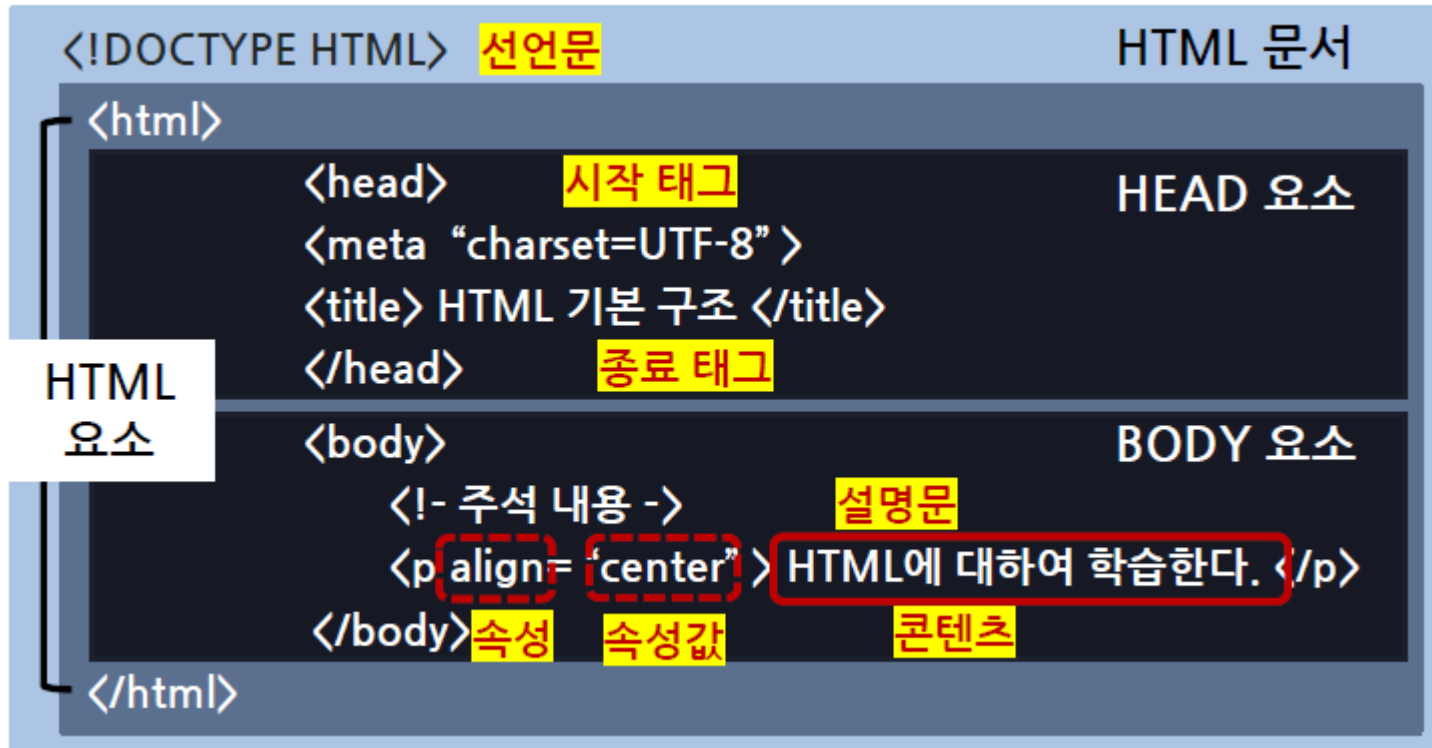
- 전체적으로 <html> 태그로 감싸 짐
- 문서의 정보를 제공하는 <head> 태그와 브라우저에 렌더링되는 내용을 작성하는 <body> 태그로 구성

웹 페이지
데이터
추출

- 추출하려는 콘텐츠의 태그를 찾아서
속성의 값이나 콘텐츠 부분을 추출하는 것

2. 웹페이지 구성 기술

- HTML(HyperText Markup Language)
 - HTML 구성 요소

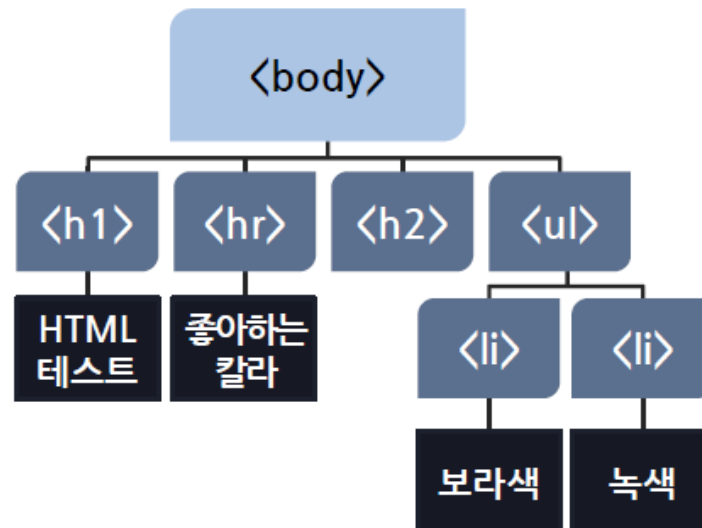


2. 웹페이지 구성 기술

■ 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</li>
    <li>보라색</li>
  </ul>
</body>
</html>
```



2. 웹페이지 구성 기술

- 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

2. 웹페이지 구성 기술

■ 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 테스트

좋아하는 칼라

- 보라색
- 녹색

HTML 문서를 CSS를
적용하여 렌더링한 경우

HTML 테스트

좋아하는 칼라

- 보라색
- 녹색

2. 웹페이지 구성 기술

- CSS 선택자

CSS
선택자
(Selector)

- 스타일을 적용하기 위해 대상 태그를 선택하는 방법

- 태그 선택자 : 태그명으로 선택하려는 경우로 태그명을 그대로 사용

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

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

2. 웹페이지 구성 기술

- 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>
```

2. 웹페이지 구성 기술

- 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>
```


2. 웹페이지 구성 기술

- CSS 선택자

- 자손 선택자 : 지정된 부모 태그의 자식 태그에만 스타일이 적용

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

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

2. 웹페이지 구성 기술

- CSS 선택자
 - 속성 선택자

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

```

```

2. 웹페이지 구성 기술

■ 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

2. 웹페이지 구성 기술

- JavaScript

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

정적 콘텐츠로 구성된 웹 페이지

- HTML 태그와 CSS만으로 구성되는 웹 페이지는 간단하게 콘텐츠 추출 가능

동적 콘텐츠로 구성된 웹 페이지

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

3. 웹콘텐츠 요청

- 웹 콘텐츠 요청 방법
 - urllib 패키지를 활용한 웹페이지요청
 - requests 패키지를 활용한 웹페이지요청

3. 웹 콘텐츠 요청 – Urllib 패키지 활용

- 주요 모듈 활용

urllib

- URL 작업을 위한 여러 모듈을 모은 패키지
- 파이썬의 표준 라이브러리

URL 문자열과 웹 요청에 관련된 모듈 5개 제공

URL 문자열을 가지고
HTTP 요청을 수행하는
`urllib.request` 모듈

URL 문자열(주소)을
해석하는
`urllib.parse` 모듈

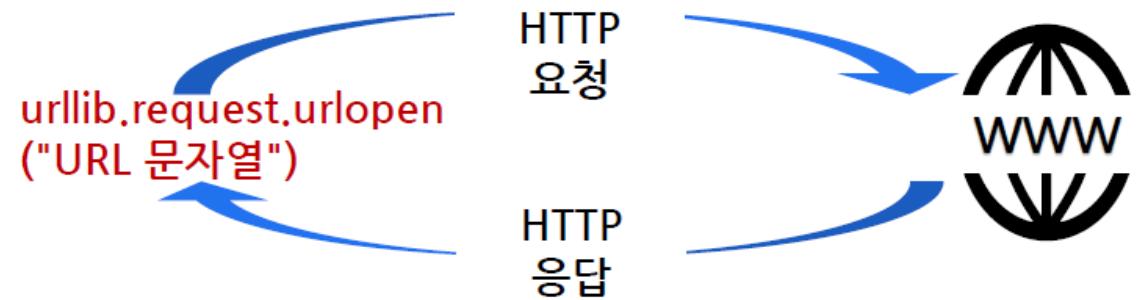
- `urllib.request`—URL 문자열을 가지고 요청 기능 제공
- `urllib.response`—`urllib` 모듈에 의해 사용되는 응답 클래스들 제공
- `urllib.parse`—URL 문자열을 파싱하여 해석하는 기능 제공
- `urllib.error`—`urllib.request`에 의해 발생하는 예외 클래스들 제공
- `urllib.robotparser`—`robots.txt` 파일을 구문 분석하는 기능 제공

3. 웹 콘텐츠 요청 - Urllib 패키지 활용

- 주요 모듈 활용
 - urllib.request 모듈

- 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")
```



3. 웹 콘텐츠 요청- Urllib 패키지 활용

- 주요 모듈 활용

- urllib.request 모듈

- 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`

3. 웹 콘텐츠 요청- Urllib 패키지 활용.

```
import urllib.request
res = urllib.request.urlopen("http://www.naver.com/")
print(type(res))
print(res.status)
print(res.version)
print(res.msg)
res_header= res.getheaders()
print("[ header 정보]-----")
for s in res_header:
    print(s)
```

```
<class 'http.client.HTTPResponse'>
200
11
OK
[ header 정보 ]-----
('Server', 'NWS')
('Date', 'Fri, 26 Feb 2021 08:26:05 GMT')
('Content-Type', 'text/html; charset=UTF-8')
('Transfer-Encoding', 'chunked')
('Connection', 'close')
('Set-Cookie', 'PM_CK_loc=6a99d2fc1b5471d9d14cea36ec9cb8ccc63cdb8691969debad2127e01ed5cabf; Expires=Sat, 27 Feb 2021 08:26:05 GMT; Path=/; HttpOnly')
('Cache-Control', 'no-cache, no-store, must-revalidate')
('Pragma', 'no-cache')
('P3P', 'CP="CAO DSP CURa ADMa TA1a PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"')
('X-Frame-Options', 'DENY')
('X-XSS-Protection', '1; mode=block')
('Strict-Transport-Security', 'max-age=63072000; includeSubdomains')
('Referrer-Policy', 'unsafe-url')
```

3. 웹 콘텐츠 요청 : Urllib 패키지 활용

#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())
```

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- 주요 모듈 활용

- `http.client.HTTPResponse` 객체의 `read()` 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

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- 주요 모듈 활동
 - `http.client.HTTPResponse` 객체의 `read()` 메서드

`res.read()`

```
<body>\r\n<h1>\xea\x80\xeb\x82\x98\xeb\x8b\xa4ABC</h1>\r\n</body>
```

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

```
<body>\n<h1>가나다ABC</h1>\n</body>
```

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- decode() 함수 실습

```
import urllib.request
url="http://unico2013.dothome.co.kr/crawling/tagstyle.html"
res = urllib.request.urlopen()
print(res)
print("[ header 정보 ]-----")
res_header = res.getheaders()
for s in res_header :
    print(s)
print("[ body 내용 ]-----")
print(res.read())
#print(res.read().decode('utf-8'))
```

3. 웹 콘텐츠 요청- Urllib 패키지 활용

■ 웹페이지 인코딩 체크(1)

- 웹 크롤링하려는 웹 페이지가 어떠한 문자 셋으로 작성되었는지 파악하는 것이 필수

페이지의
문자 셋 정보

- 페이지의 소스 내용에서 **<meta>** 태그의 charset 정보를 체크하면 파악 가능

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>블러스타의 태그</h1>
```

```
<!doctype html>
<!--[if lt IE 7]> <html class="n
<!--[if IE 7]> <html class="n
<!--[if IE 8]> <html class="n
<!--[if gt IE 8]><!--><html class=
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compati
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<html>
<head>
<title id="Title">알라딘</title>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<meta content="Microsoft Visual Studio .NET 7.1" name="GENERATOR">
<meta content="C#" name="CODE_LANGUAGE">
<meta content="JavaScript" name="vs_defaultClientScript">
```

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- 웹 페이지 인코딩 체크(2)

- 웹 페이지의 문자 셋 정보를 파이썬 프로그램으로도 파악할 수 있음

사용되는
API

- http.client.HTTPMessage 객체의 `get_content_charset()` 메서드

urllib.request.urlopen() 함수의 리턴 값인
http.client.HTTPResponse 객체의 info() 메서드 호출

http.client.HTTPMessage 객체가 리턴 됨

get_content_charset() 메서드 호출

문자 셋 정보를 문자열로 리턴 받음

웹 서버로부터 응답될 때 전달되는
Content-Type이라는 응답 헤더 정보를 읽고
해당 페이지의 문자 셋 정보를 추출해 줌

```
url = 'http://www.python.org/'  
f = urllib.request.urlopen(url)  
encoding = f.info().get_content_charset()
```

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- get_content_charset()

```
import urllib.request
print("=====")
url = 'https://www.python.org/'
f = urllib.request.urlopen(url)
print(type(f))
print(type(f.info()))
encoding = f.info().get_content_charset()
print(url, ' 페이지의 인코딩 정보 :', encoding)
text = f.read(500).decode(encoding)
print(text)
print("=====")

url = 'https://www.daum.net/'
f = urllib.request.urlopen(url)
encoding = f.info().get_content_charset()
print(url, ' 페이지의 인코딩 정보 :', encoding)
text = f.read(500).decode(encoding)
print(text)
print("=====")
```


3. 웹 콘텐츠 요청- Urllib 패키지 활용

■ urllib.parse 모듈

- 웹 서버에 페이지 또는 정보를 요청할 때 함께 전달하는 데이터
 - GET 방식 요청 : Query 문자열
 - POST 방식 요청 : 요청 파라미터

`name=value&name=value&name=value&.....`

- 영문과 숫자는 그대로 전달되지만 **한글은 %기호와 함께 16진수 코드 값**으로 전달되어야 함
- 웹 크롤링을 할 때 요구되는 Query 문자열을 함께 전달해야 하는 경우, 직접 Query 문자열을 구성해서 전달해야 함


urllib.parse 모듈 사용

- `urllib.parse.urlparse()`
- `urllib.parse.urlencode()`

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- urllib.parse 모듈-urlparse() 함수
 - urllib.parse.urlparse("URL문자열")
 - 아규먼트에 지정된 URL 문자열의 정보를 파싱하고 각각의 정보를 정해진 속성으로 저장하여 urllib.parse.ParseResult 객체를 리턴함
 - 각 속성들을 이용하여 필요한 정보만 추출할 수 있음

```
url1 =urlparse  
(‘https://movie.daum.net/moviedb/main?movieId=93252’)
```



```
ParseResult(scheme='https',  
netloc='movie.daum.net',  
path='/moviedb/main', params="",  
query='movieId=93252', fragment="")
```

```
url1.netloc, url1.path, url1.query, url1.scheme,  
url1.port, url1.fragment, url1.geturl()
```

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- urllib.parse 모듈-urlencode() 함수

- urllib.parse.urlencode()

- 메서드의 아규먼트로 지정된 name과 value로 구성된 딕셔너리정보를 정해진 규격의 Query 문자열 또는 요청 파라미터 문자열로 리턴 함

```
urlencode({'number': 12524, 'type': 'issue', 'action':  
'show'})
```

number=12524&type=issue&action=show

```
urlencode({'addr': '서울시 강남구 역삼동'})
```

addr=%EC%84%9C%EC%9A%B8%EC%8B%
9C+%EA%B0%95%EB%82%A8%EA%B5%A
C+%EC%97%AD%EC%82%BC%EB%8F%99

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- Query 문자열을 포함하여 요청

- Query 문자열을 포함하여 요청하는 것
➡ GET 방식 요청

- urllib.parse.urlencode 함수로 name과 value로 구성되는 Query 문자열을 만듦
- URL 문자열의 뒤에 '?' 기호를 추가하여 요청 URL로 사용

```
params = urllib.parse.urlencode({'name': '유니코',  
                                'age': 10})
```

```
url = "http://unico2013.dothome.co.kr/  
crawling/get.php?%s" % params
```



```
http://unico2013.dothome.co.kr/crawling/get.php?n  
ame=%EC%9C%A0%EB%8B%88%EC%BD%94&  
age=10
```

```
urllib.request.urlopen(url)
```

3. 웹 콘텐츠 요청- Urllib 패키지 활용

- 요청 파라미터를 포함하여 요청(2)

- URL 문자열과 요청 파라미터 문자열을 지정한 `urllib.request.Request` 객체 생성
- `urllib.request.urlopen()` 함수 호출 시 URL 문자열 대신 `urllib.request.Request` 객체 지정

```
data = urllib.parse.urlencode({'name': '유니코', 'age': 10})  
postdata = data.encode('ascii')  
req = urllib.request.Request(url='http://unico2013.  
dothome.co.kr/crawling/post.php', data=postdata)  
urllib.request.urlopen(req)
```

3. 웹 콘텐츠 요청 : 실습

- 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")
```

3. 웹 콘텐츠 요청 : 실습

- RSS 요청

```
#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)
```

4. 웹 콘텐츠 요청- requests 패키지 활용

- requests 패키지란?

requests
패키지

- Kenneth Reitz에 의해 개발된 파이썬 라이브러리
- HTTP 프로토콜과 관련된 기능 지원

requests 패키지의 공식 홈페이지 소개

Requests is an elegant and simple HTTP library for Python, built for human beings. You are currently looking at the documentation of the development release.

< <https://2.python-requests.org/en/master/> >

4. 웹 콘텐츠 요청- requests 패키지 활용

- requests 패키지란?
 - 아나콘다에는 requests 패키지가 site-packages로 설치되어 있음
 - pipenv명령으로 설치 가능

`pipenv install requests`

```
C:\Users\Samsung>pip show requests
Name: requests
Version: 2.21.0
Summary: Python HTTP for Humans.
Home-page: http://python-requests.org
Author: Kenneth Reitz
Author-email: me@kennethreitz.org
License: Apache 2.0
Location: c:\users\samsung\anaconda3\lib\site-packages
Requires: urllib3, certifi, chardet, idna
Required-by: Sphinx, conda, conda-build, anaconda-project, anaconda-client
```

urllib 패키지	requests 패키지
인코딩하여 바이너리 형태로 데이터 전송	딕셔너리 형태로 데이터 전송
데이터 전달 방식에 따라 GET 요청, POST 요청을 구분	요청 메서드(GET, POST)를 명시하여 요청

4. 웹 콘텐츠 요청- requests 패키지 활용

- requests 패키지 소개- requests.request() 함수
 - requests 패키지의 대표 함수
 - HTTP 요청을 서버에 보내고 응답을 받아오는 기능 지원

`requests.request(method, url, **kwargs)`

- `method` : 요청 방식 지정(GET, POST, HEAD, PUT, DELETE, OPTIONS)
- `url` : 요청할 대상 URL 문자열 지정
- `params` : [선택적] 요청 시 전달할 Query 문자열 지정
(딕셔너리, 튜플리스트, 바이트열 가능)
- `data` : [선택적] 요청 시 바디에 담아서 전달할
요청 파라미터 지정
(딕셔너리, 튜플리스트, 바이트열 가능)
- `json` : [선택적] 요청 시 바디에 담아서 전달할
JSON 타입의 객체 지정
- `auth` : [선택적] 인증처리(로그인)에 사용할 튜플 지정

4. 웹 콘텐츠 요청- requests 패키지 활용

- HTTP 요청 방식을 지원하는 함수
 - requests.request() 함수 외에 각각의 요청방식에 따른 메서드들도 제공
 - requests.request() 함수에 요청방식을 지정하여 호출하는 것과 동일

- requests.get(url, params=None, **kwargs)
- requests.post(url, data=None, json=None, **kwargs)
- requests.head(url, **kwargs)
- requests.put(url, data=None, **kwargs)
- requests.patch(url, data=None, **kwargs)
- requests.delete(url, **kwargs)

4. 웹 콘텐츠 요청- requests 패키지 활용

- HTTP 요청방식을 지원하는 함수

- HTTP 프로토콜에서 정의한 GET, POST, HEAD, PUT, PATCH, DELETE 등의 요청방식을 처리하는 메서드들을 모두 지원
- GET, HEAD, POST만 학습

GET 방식	<ul style="list-style-type: none">▪ 요청한 페이지의 헤더, 바디를 모두 받아오는 요청▪ Query 문자열을 추가하여 요청할 수도 있음
HEAD 방식	<ul style="list-style-type: none">▪ 콘텐츠 없이 요청 헤더만을 받아오는 방식▪ 요청 바디에 요청 파라미터 데이터를 추가하여 요청▪ 헤더와 바디를 모두 받아옴

4. 웹 콘텐츠 요청- requests 패키지 활용

- GET 방식요청

- GET 방식 요청은 다음 두 가지 함수 중 하나를 호출하여 처리 가능

```
requests.request('GET', url, **kwargs)
```

```
requests.get(url, **kwargs)
```

```
[ kwargs ]
```

params - (선택적) 요청 시 전달할 Query 문자열을 지정

- Query 문자열을 포함하여 요청: params 매개변수에 딕셔너리, 튜플리스트, 바이트열(bytes) 형식으로 전달
- Query 문자열을 포함하지 않는 요청: params 매개변수의 설정 생략

4. 웹 콘텐츠 요청- requests 패키지 활용

- POST 방식 요청
 - POST 방식 요청은 다음 두 가지 함수 중 하나를 호출하여 처리 가능

```
requests.request('POST', url, **kwargs)
```

```
requests.post(url, **kwargs)
```

[kwargs]

data - (선택적) 요청 시 바디에 담아서 전달할 요청 +
파라미터를 지정

json - (선택적) 요청 시 바디에 담아서 전달할 JSON 타입의
객체를 지정
JSON 형식

4. 웹 콘텐츠 요청- requests 패키지 활용

■ 응답 처리

- requests.request(), requests.get(), requests.head(), requests.post() 함수 모두 리턴 값은 **requests.models.Response** 객체임

text

- 문자열 형식으로 응답 콘텐츠 추출
- 추출 시 사용되는 문자 셋은 'ISO-8859-1' 이므로 'utf-8'이나 'euc-kr' 문자 셋으로 작성된 콘텐츠 추출 시 한글이 깨지는 현상발생
- 추출 전 응답되는 콘텐츠의 문자 셋 정보를 파악하여 **Response** 객체의 **encoding** 속성에 문자 셋 정보를 설정한 후 추출

content

- 바이트열 형식으로 응답 콘텐츠 추출
- 응답 콘텐츠가 이미지와 같은 바이너리 형식인 경우 사용
- 한글이 들어간 문자열 형식인 경우 **r.content.decode('utf-8')**를 사용해서 디코드 해야 함

4. 웹 콘텐츠 요청- requests 패키지 활용- 실습

```
import requests

r = requests.request('get', 'http://unico2013.dothome.co.kr/crawling/exam.html')
r.encoding = 'utf-8'
print(type(r))
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
print('-----')
r = requests.request('head', 'http://unico2013.dothome.co.kr/crawling/exam.html')
r.encoding = 'utf-8'
print(type(r))
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
print('-----')
r = requests.request('post', 'http://unico2013.dothome.co.kr/crawling/post.php', data= {'name':'백도', 'age' : 12})
r.encoding = 'utf-8'
print(type(r))
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
```


4. 웹 콘텐츠 요청- requests 패키지 활용- 실습

```
import requests
r = requests.get('http://unico2013.dothome.co.kr/crawling/exam.html')
r.encoding = 'utf-8'
print(type(r))
print(r.headers)
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
```

```
import requests

r = requests.head('http://unico2013.dothome.co.kr/crawling/exam.html')
print(type(r))
print(r.headers)
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
```

4. 웹 콘텐츠 요청- requests 패키지 활용- 실습

```
import urllib.request
import urllib.parse
params = urllib.parse.urlencode({'category': '역사', 'page': 25})
url = "http://unico2013.dothome.co.kr/crawling/exercise.php?%s" % params
with urllib.request.urlopen(url) as f:
    print(f.read().decode('utf-8'))
```

```
import requests
dicdata = {'category': '여행', 'page': 100}
urlstr = 'http://unico2013.dothome.co.kr/crawling/exercise.php'
r = requests.get(urlstr, params=dicdata)
r.encoding = "utf-8"
print(r.text)
```