학습 내용

3부. 데이터 분석 라이브러리 활용

10장. 데이터프레임과 시리즈

11장. 데이터 시각화

12장. 웹 데이터 수집

- 1. 퓨티풀솝과 파서
- 2. requests를 이용한 웹 데이터 수집

1절. 뷰티풀솝과 파서

[공식 사이트](https://www.crummy.com/software/BeautifulSoup)
[Documentation](https://www.crummy.com/software/BeautifulSoup/bs4/doc)

1.1. Beautiful Soup

1절. 뷰티풀솝과 파서

- 뷰티풀솝(Beautiful Soup)은 스크린 스크래핑(screen-scraping) 프로젝트를 위해 설계된 파이썬 라이브러리
- 구문 분석, 트리 탐색, 검색 및 수정을 위한 몇 가지 간단한 방법과 파이썬 관용구를 제공하며 문서를 분석하고 필요한 것을 추출하는 도구
- 들어오는 문서를 유니코드로 보내고 문서를 UTF-8로 자동 변환
- 뷰티풀솝은 lxml 및 html5lib과 같은 파이썬 파서 라이브러리를 사용할 수 있음
  - 이를 이용하면 속도를 개선시키거나 호환성이 높은 응용프로그램을 만들 수 있음
- 공식 사이트
  - https://www.crummy.com/software/BeautifulSoup/
- Documentation
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/

1.2. 파서 라이브러리

파서	사용법	장점	단점
파이썬의 html.parser	BeautifulSoup(markup, "html.parser")	• 보통 속도 • 파이썬 2.7.3, 3.2.2 이상 버전에서 호환됨	• 파이썬 2.7.3 또는 3.2.2 이전버전에서 호환되지 않음
lxml's HTML parser	BeautifulSoup(markup, "lxml")	• 매우 빠름 • 호환성	• 외부 C에 의존함
lxml's XML parser	BeautifulSoup(markup, "lxml-xml") BeautifulSoup(markup, "xml")	• 매우 빠름	• XML 파서만 지원 • 외부 C에 의존함

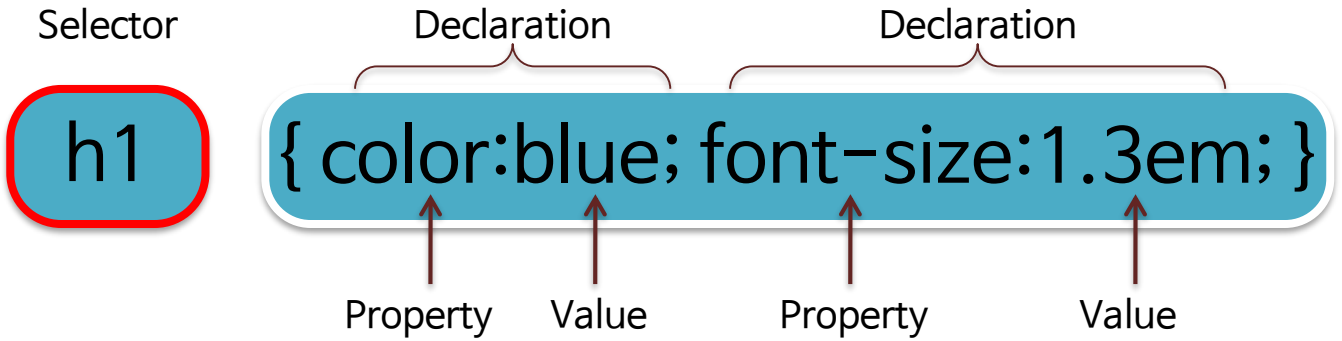
1.3. Selector API

- BeautifulSoup은 가장 일반적으로 사용되는 CSS 선택자를 지원
- BeautifulSoup의 Selector API는 select()와 select_one(), find() 등
- select()와 select_one() 메서드만 알아도 원하는 요소를 찾기에 충분
- soup.select("CSS 선택자") : CSS 선택자에 해당하는 모든 요소를 반환
- soup.select_one("CSS 선택자") : CSS 선택자에 맞는 오직 첫 번째 태그 요소만 반환

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/

1.4. CSS 선택자

- CSS(Cascading Style Sheet)
  - CSS는 문서의 콘텐츠와 레이아웃, 글꼴 및 시각적 요소들로 표현되는 문서의 외관(디자인)을 분리하기 위한 목적으로 만들어졌음
- CSS 선택자
  - CSS를 이용해서 HTML 문서에 시각적 요소를 부여할 때 문서 내의 어느 요소에 부여할지를 결정하기 위해 사용하는 것
  - CSS 선택자(Selector)는 HTML 문서의 태그 이름, class 속성, id 속성 등을 이용해서 작성할 수 있음



실습을 위한 준비


```
import requests
from requests_file import FileAdapter

s = requests.Session()
s.mount('file://', FileAdapter())
```

```
res = s.get('file:///sample.html')
```

```
res
```

<Response [200]>

```
from bs4 import BeautifulSoup
```

```
soup = BeautifulSoup(res.content, "html.parser")
```

```
el = soup.select_one("h1")
print(el)
```

<h1>Hello CSS</h1>

```
<html>
<head><title>HTML Sample</title>
</head>
<body>
    <h1>Hello CSS</h1>
    <div id="subject">선택자</div>
    <div class="contents">선택자를 어떻게 작성하느냐에 따라 <span>다른 <b>요소가 반환</b></span> 됩니다.</div>
    <div>CSS 선택자는 다양한 곳에서 <b>활용</b>됩니다.</div>
</body>
</html>
```

선택자 종류(1/3)

- 태그 선택자 ("element")
  - 태그 선택자는 일반적으로 스타일 정의하고 싶은 html 태그 이름을 사용
  - 요소 안의 텍스트는 text, 태그이름은 name 그리고 태그의 속성들은 attrs를 이용해 조회
  - soup.select("h1")

- 다중(그룹) 선택자 ("selector1, selector2, selectorN")
  - 선택자를 ","(comma)로 분리하여 선언하면 여러 개 선택자 적용.
  - 해당하는 모든 선택자의 요소를 찾기 때문에 select_one()아닌 select() 메서드를 이용
  - soup.select("h1, p")

선택자 종류(2/3)

- 내포 선택자 ("ancestor descendant")
  - 요소가 내포 관계가 있을 때 적용시키기 위한 선택자.
  - 선택자와 선택자 사이를 공백으로 띄우고 나열
  - soup.select("div b")

- 자식 선택자 ("parent > child")
  - 선택자와 선택자 사이에 >를 입력하며 반드시 부모자식간의 관계에만 스타일이 적용되도록 함.
  - 두 단계 이상 건너뛴 관계에서는 자식 선택자가 작동하지 않음
  - soup.select("div > b")

선택자 종류(3/3)

- 클래스(class) 선택자 (".class")
  - HTML 문서에서 class 속성의 값과 일치하는 요소를 선택
  - 선택자 이름 앞에 "."을 이용하여 선언.
  - soup.select("div.contents")

- 아이디(id) 선택자 (" #id ")
  - HTML 문서에서 id 속성의 값과 일치하는 요소를 선택
  - id 선택자는 #으로 정의합니다.
  - soup.select_one("#subject")

- 속성 선택자 [name="value"]
  - 특정한 속성을 갖는 요소만 선택.
  - 속성 선택자는 [와]사이에 속성의 이름과 값을 지정
  - soup.select_one("[id=subject]")

2절. requests를 이용한 웹 데이터 수집

requests

2절. requests를 이용한 웹 데이터 수집

- 파이썬에서 HTTP 요청을 만들기 위한 사실상의 표준

- 2절의 내용
  - 가장 일반적인 HTTP 메소드를 사용하여 요청하기
  - 쿼리 문자열 및 메시지 본문을 사용하여 사용자의 요청과 데이터를 변경하기
  - 요청 및 응답에서 데이터 검사하기
  - 인증 된 요청 만들기
  - 응용 프로그램이 백업 또는 속도 저하를 막을 수 있도록 요청을 구성하기

2.1. requests 모듈

2절. requests를 이용한 웹 데이터 수집

- pip 명령으로 쉽게 설치
  - pip install requests

- requests가 설치되면 응용 프로그램에서 사용
  - import requests

2.2. GET 요청

2절. requests를 이용한 웹 데이터 수집

- GET은 HTTP 요청(Request) 방식 중 하나
- requests.get() : 지정된 리소스에서 데이터를 가져옴

```
1  import requests
```

```
1  requests.get('https://api.github.com')
```

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

2.3. 응답 객체

2절. requests를 이용한 웹 데이터 수집

- 응답(Response)은 요청의 결과를 저장한 개체

```
1  response = requests.get('https://api.github.com')
```

```
1  response.
```

apparent_encoding
close
connection
content
cookies
elapsed
encoding
headers
history
is_permanent_redirect

1) 상태 코드

- HTTP 응답 메시지의 상태 라인에는 상태 코드와 상태 메시지를 포함하고 있음

상태 코드	의미	내 용
100번 영역	정보전송	임시적인 응답을 나타내는 것은 Status-Line과 선택적인 헤더들로 구성되어 있고, 빈 줄로 끝을 맺는다.
200번 영역	성공	클라이언트의 요구가 성공적으로 수신되어 처리되었음을 의미 200(성공), 201(POST요청 처리), 204(전송할 데이터 없음)
300번 영역	리다이렉션	해당 요구사항을 처리하기 위해서는 사용자 에이전트에 의해 수행되어야 할 추가적인 동작이 있음을 의미
400번 영역	클라이언트 측 오류	클라이언트가 서버에게 보내는 요구 메시지를 완전히 처리하지 못한 경우와 같이 클라이언트에서 오류가 발생한 경우에 사용 401(사용자인증), 403(접근권한 없음), 404(요청한 URL 없음)
500번 영역	서버측 오류	서버 자체에서 발생된 오류상황이나 요구사항을 제대로 처리할 수 없을 때 사용.

상태 코드


```
1  response.status_code
```

200

```
1  if response.status_code == 200:
2      print('Success!')
3  elif response.status_code == 404:
4      print('Not Found.')
```

상태코드 정보를 사용하여 코드에서 의사 결정을 내릴 수 있음

Success!

Response 인스턴스를 사용하면 상태 코드가 200에서 400 사이 이면 True로 평가되고 그렇지 않으면 False로 평가

```
1  if response:
2      print('Success!')
3  else:
4      print('An error has occurred.')
```

Success!

1) 환율 정보 가져오기


```
1 import requests
2 from bs4 import BeautifulSoup
```

```
1 url = 'https://finance.naver.com/marketindex/' # 환율 정보
2 market_index = requests.get(url)
```

```
1 market_index
```

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

```
1 soup = BeautifulSoup(market_index.content, "html.parser")
```

```
1 price = soup.select_one("div.head_info > span.value")
2 print(price)
```

```
<span class="value">1,189.70</span>
```

```
1 print("usd/krw=", price.text)
```

```
usd/krw= 1,189.70
```