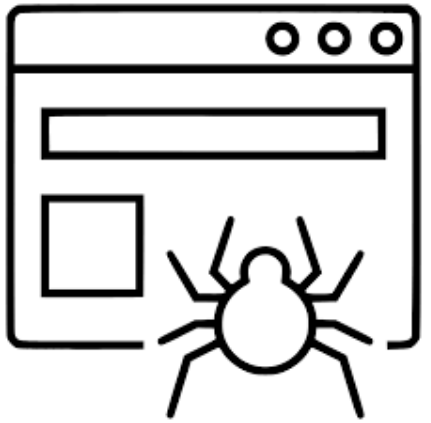
BeautifulSoup

2019.10

1. Web Crawling

■ 개요

- 사전적으로는 기어다닌다는 뜻
- Web상을 돌아다니면서 정보를 수집하는 행위
- 크롤링, 스크래핑(Scraping) 또는 데이터 긁기 등 다양한 단어로 불리움



■ 대상

- 웹 상의 자원
- 정적인 문서 또는 API와 같은 서비스
- 구글과 같은 검색엔진에서는 웹 사이트의 정적인 데이터를 긁어다가 필요한 정보를 추출해서 검색 인덱스를 생성
- 가격 정보 비교 사이트는 상품과 가격정보등을 긁어다가 서로 다른 쇼핑몰의 가격을 비교해줌

■ 툴, 라이브러리

- 라이브러리
  - Beautiful Soup: 일반적, 파이썬
  - Jsoup: 자바 버전
  - Selenium: 브라우저를 이용
- 사전 지식: HTML, CSS, JavaScript 기초
- Chrome Web Browser 개발자 도구

■ 과정

- 크롤링 대상 선정 (API 또는 웹 문서)
- 데이터 로드
- 데이터 분석
- 수집

- 예제 소스코드

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Web Crawling Example</title>
</head>

<body>
  <div>
    <p>a</p>
    <p>b</p>
    <p>c</p>
  </div>
  <div class="ex_class">
    <p>1</p>
    <p>2</p>
    <p>3</p>
  </div>
  <div id="ex_id">
    <p>X</p>
    <p>Y</p>
    <p>Z</p>
  </div>

  <h1>This is a heading.</h1>
  <p>This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <a href="www.naver.com" class="a">
     Naver</a>
</body>

</html>
```

2. Beautiful Soup

- HTML 파일 열기

```
from bs4 import BeautifulSoup
with open("00_Example.html") as fp:
    soup = BeautifulSoup(fp, 'html.parser')
```

- urllib를 통해서 웹에 있는 소스 가져오기

```
import urllib.request
import urllib.parse

# web_url에 원하는 웹의 URL을 넣어주면 됨
with urllib.request.urlopen(web_url) as response:
    html = response.read()
    soup = BeautifulSoup(html, 'html.parser')
```

2. Beautiful Soup

- 태그를 이용해서 가져오기
- find - 해당 조건에 맞는 하나의 태그를 가져온다. 중복이면 가장 첫 번째 태그를 가져온다.

```
from bs4 import BeautifulSoup
fp = open("00_Example.html")
soup = BeautifulSoup(fp, 'html.parser')

first_div = soup.find("div")
print(first_div)

# 출력 결과
<div>
<p>a</p>
<p>b</p>
<p>c</p>
</div>
```

- find_all - 해당 조건에 맞는 모든 태그들을 가져온다.

```
all_divs = soup.find_all("div")
print(all_divs)

# 출력 결과
[<div>
<p>a</p>
<p>b</p>
<p>c</p>
</div>, <div class="ex_class">
<p>1</p>
<p>2</p>
<p>3</p>
</div>, <div id="ex_id">
<p>X</p>
<p>Y</p>
<p>Z</p>
</div>]
```

- find_all - 해당 조건에 맞는 모든 태그들을 가져온다.

```
all_ps = soup.find_all("p")
print(all_ps)

# 출력 결과
[<p>a</p>, <p>b</p>, <p>c</p>, <p>1</p>, <p>2</p>, <p>3</p>, <p>X</p>, <p>Y</p>,
<p>Z</p>, <p>This is a paragraph.</p>, <p>This is another paragraph.</p>]
```

2. Beautiful Soup

- 태그와 속성을 이용해서 가져오기

```
# 함수의 인자로 원하는 태그를 첫번째 인자로
# 그 다음에 속성:값의 형태(dictionary)로 만들어서 넣어주면 됨
# - find_all('태그명', {'속성명' : '값' ...})
# - find('태그명', {'속성명' : '값' ...})

ex_id_divs = soup.find('div', {'id' : 'ex_id'})
print(ex_id_divs)

# 출력 결과
<div id="ex_id">
<p>X</p>
<p>Y</p>
<p>Z</p>
</div>

# 동일한 결과
result = soup.find('div', id = 'ex_id')
print(result)
```

- class 이름으로 찾기

```
result = soup.find('div', {'class' : 'ex_class'})
print(result)

# 출력 결과
<div class="ex_class">
<p>1</p>
<p>2</p>
<p>3</p>
</div>


# 동일한 결과를 얻는 방법
# class가 아니라 class_ (under bar가 있음)
result = soup.find('div', class_ = 'ex_class')
print(result)
```

2. Beautiful Soup

- HTML 구조를 이용해 부분부분 가져오기

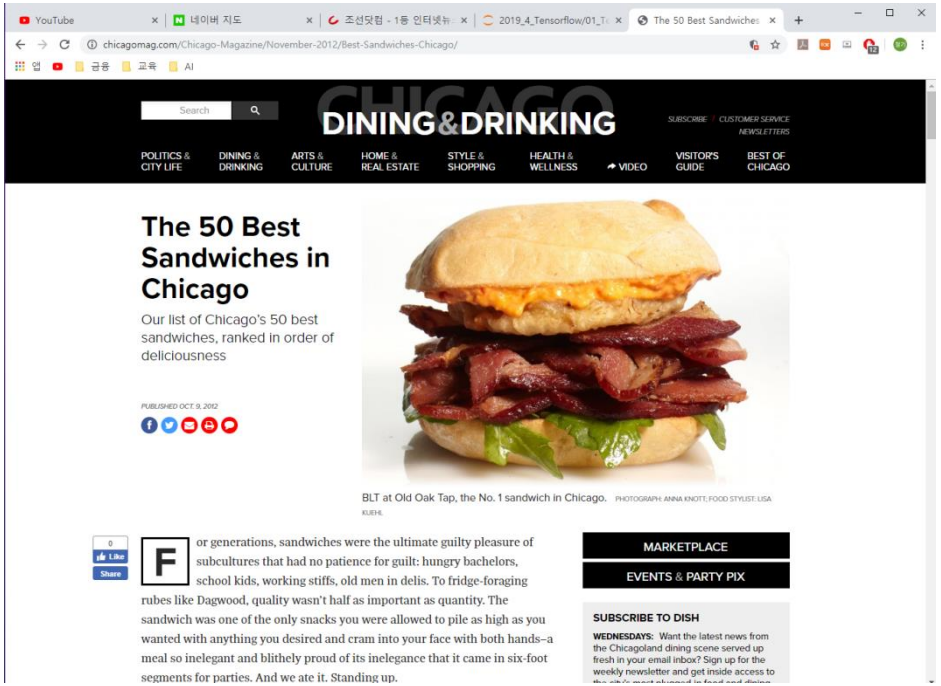
```
ex_id_divs = soup.find("div", {"id":"ex_id"})
# 찾은 태그들 안에서 p 태그를 가져온다.
all_ps_in_ex_id_divs = ex_id_divs.find_all("p")
print(all_ps_in_ex_id_divs)

# 출력 결과
[<p>X</p>, <p>Y</p>, <p>Z</p>]


all_divs = soup.find_all('div')
for each_div in all_divs:
    print(each_div)
    print()
```

■ 시카고 샌드위치 맛집 분석

- 접속 주소: https://goo.gl/wAtv1s
- 50개 맛집 정보: 가게명, 메뉴, 가격, 주소
- 주소의 위치 정보를 알아서 지도에 표시하기



1. **BLT**
   Old Oak Tap
   Read more
2. **Fried Bologna**
   Au Cheval
   Read more
3. **Woodland Mushroom**
   Xoco
   Read more
4. **Roast Beef**
   Al's Deli
   Read more
5. **PB&L**
   Publican Quality Meats
   Read more
6. **Belgian Chicken Curry Salad**
   Hendrickx Belgian Bread Crafter
   Read more
7. **Lobster Roll**
   Acadia
   Read more
8. **Smoked Salmon Salad**
   Birchwood Kitchen
   Read more
9. **Atomica Cemitas**
   Cemitas Puebla
   Read more