Python

담당 : 이도연

교육기간 : 2019.5.18~19.6.8

>> Web Crawling (웹 크롤링)

- 정제되지 않은 웹페이지에서 필요한 데이터만 추출해내는 행위
- 웹 크롤링(Web crawling)은 웹 페이지들을 긁어와서 데이터를 추출하는 것을 말합니다.
- 웹 크롤러는 자동화된 방식으로 웹 페이지들을 탐색하는 컴퓨터 프로그램입니다.
- 스크레이핑은 웹에서 HTML 파일을 다운로드하는 과정과 다운로드한 HTML에서 원하는 데이 터를 파싱하는 두 단계로 진행합니다. 파이썬에서는 requests 모듈을 이용해 HTML 코드를 다운로드하 고 BeautifulSoup 모듈로 원하는 데이터를 파싱합니다
- . 주의: 정보를 읽어올 수 있다고 해서, 마음대로 활용할 수 있다는 것은 아닙니다. 저작권에 유의

- **beautifulsoup, selenium 활용**

- 스크랩핑(scraping) : 데이터를 수집하는 행위
- 크롤링(Crawling) : 조직적 자동화된 방법으로 월드와이드웹을 탐색 하는 것
- 파싱(parsing) : 문장 혹은 문서를 구성 성분으로 분해하고 위계 관계를 분석하여 문장의 구조를 결정하는 것

- Requests 설치
- BeautifulSoup 설치
- html 소스 가져오기
- 원하는 데이터 추출하기

```
import bs4

htmlStr = """
    <html>
    <table border=1>
        <tr>
            <td> 항목 </td>
            <td> 2013 </td>
            <td> 2014 </td>
            <td> 2015 </td>
        </tr>
        <tr>
            <td> 매출액 </td>
            <td> 100 </td>
            <td> 200 </td>
            <td> 300 </td>
        </tr>
    </table>
</html>
"""

print(htmlStr)
```

```
bsObj =bs4.BeautifulSoup(htmlStr,
  "html.parser")     # 공백이랑 제거
  # print(bsObj)
  # print(type(bsObj))


  # 첫번째 태그만 찾아옴
  # print(bsObj.find("td"))

   # td 관련 다 찾아옴
  print(bsObj.find_all("td"))
```

```
import  requests
from bs4 import BeautifulSoup

html = requests.get(http://www.ikosmo.co.kr/').text
soup = BeautifulSoup(html, 'html.parser’)
For tag in soup.select(‘#course_list > .course > a’):
Print(tag.text, tag[‘href’])
```

>> 검색 시 (단일/다중 element 검색)
> BeautifulSoup

```
bsObj.find({'id' : 'id명')

bsObj.find( {'class' : 'class명')

bsObj.find('태그 이름', {'class' : 'class명'})

bsObj.find_all('태그 이름')

bsObj.find_all('a')[0]   # a tag 여러 개 중에서 첫번째 것 찾아옴

bsObj.find_all('a')[0].get_text()  # a tag 내부의 텍스트를 얻어옴
```

>> 검색 시 (단일/다중 element 검색)
> Selenium

find_element_by_id('id명')

find_element_by_class_name('class명')

find_element_by_xpath('xpath')

find_element_by_tag_name('tag명')

driver.text

> Selenium sample

/Users/Administrator/Desktop/chromedriver_win32/chromedriver

```
from selenium import webdriver
import time

browser = webdriver.Chrome("/Users/HappyDoYeon/Desktop/chromedriver_win32/chromedriver")
browser.get("http://python.org")

menus = browser.find_elements_by_css_selector('#top ul.menu li')

pypi = None
for m in menus:
    if m.text == "PyPI":
        pypi = m
    print(m.text)

pypi.click()  # 클릭

time.sleep(5)  # 5초 대기
browser.quit()  # 브라우저 종료
```

>> 정규표현식(Regular Expressions)

메타 문자

```
. ^ $ + ? { } [ ] \ | ( )
```

11가지를 안보고 기능까지 정확하게 읊을 수 있도록 노력하려 한다.

메타 문자	설명
[ ]	문자 클래스
.	₩n을 제외한 모든 문자와 매치 (점 하나는 글자 하나를 의미)
*	0회 이상 반복 (없어도 상관 없음)
+	1회 이상 반복 (무조건 한번 이상 등장해야 함)
{m, n}	m회 이상 n회 이하
|	or 조건식을 의미
^	문자열의 시작 의미
$	문자열의 끝을 의미
?	0회 이상 1회 이하
₩	이스케이프, 또는 메타 문자를 일반 문자로 인식하게 한다
( )	그룹핑, 추출할 패턴을 지정한다.

1. [ ] : 대괄호 안에 아무거나 집어넣을 수 있다.

https://whatisthenext.tistory.com/116

> 정규표현식(Regular Expressions)

문자 클래스의 특징은 세 가지만 알고 있으면 된다.

1. 대괄호 안에는 어떤 것이든 들어갈 수 있다.
    - 단, 엄밀하게 구분된다. 즉, a와 A가 다르고, z와 Z가 다르다.
2. 안에 들어간 것끼리는 or 로 연결된다.
    - 하이픈(-)으로 연결해줄 수 있다.
    - [a-z]는 a부터 z까지를 의미한다. [0-9]는 0부터 9까지를 의미한다.
3. 만약 [abc]라면, 'a, b, c' 중에서 한 개의 문자 와 매칭된다.

> 정규표현식(Regular Expressions)

1. 정규표현식 [abc]가 있다. 다음중 매칭되는 것은?

   (1) a (2) before (3) dude

2. 정규표현식 [a-c]가 있다. 다음중 같은 의미는?

   (1) a 또는 c (2) a 또는 b 또는 c (3) [abc] (4) a 그리고 b 그리고 c

3. 정규 표현식 a[a-z0-9]z가 있다. 다음중 매칭(Y) 되는 것은? (응용)
   (1) a!012z (2) aBz (3) a999z (4) azX09z (5) a9z

---

1. (1), (2) 풀이 : a 또는 b 또는 c가 들어있으면 된다.

2. (2) 풀이 : 하이픈(-)은 범위를 나타낸다. [0-9]는 0~9를 의미한다.

3. (5) 풀이 : 대괄호 안에 `한 글자` 만을 의미한다. `+` 나 `*` 를 써준다면 (3)도 출력된다.

https://www.w3schools.com/python/python_regex.asp

```
import re

source = "ct cat caat caaat caaaat"
m1 = re.findall("ca{2}t", source)
m2 = re.findall("ca{2,5}t", source)
m3 = re.findall("ca{0,}t", source)
m4 = re.findall("ca{0,1}t", source)
m5 = re.findall("ca{,3}t", source)
print("m1 : ", m1)
print("m2 : ", m2)
print("m3 : ", m3)
print("m4 : ", m4)
print("m5 : ", m5)

>>>출력결과
m1 :  ['caat']
m2 :  ['caat', 'caaat', 'caaaat']
m3 :  ['ct', 'cat', 'caat', 'caaat', 'caaaat']
m4 :  ['ct', 'cat']
m5 :  ['ct', 'cat', 'caat', 'caaat']
```

```
import re

source = "ct cat caat caaat caaaat"
m1 = re.findall("ca?", source)
print("m1 : ", m1)

>>> 출력결과
m1 :  ['c', 'ca', 'ca', 'ca', 'ca']
```