데이터 크롤링과 정제

3장. 크롤링 시작하기

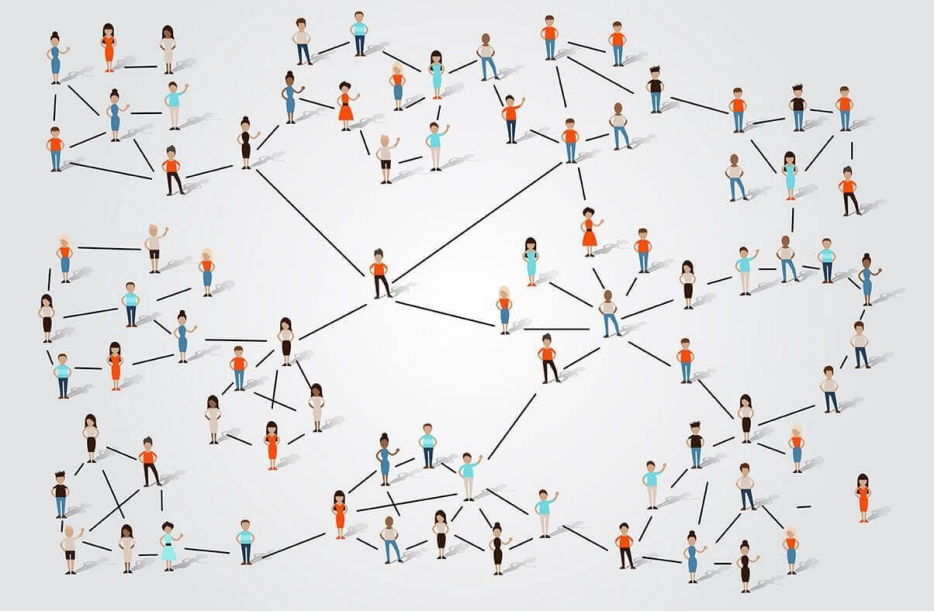
빅데이터 분석가 양성과정

목차

- 단일 도메인 내의 이동
- 전체 사이트 크롤링
  - 전체 사이트에서 데이터 수집
- 인터넷 크롤링

웹 크롤링 시작하기

- 위키 백과의 여섯 다리
  - 케빈 베이컨의 여섯 다리
    - https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon
    - http://ko.experiments.wikidok.net/wp-d/57b7cb0dccfd0ddc58659209/View
  - 6단계 법칙
    - 인간 관계는 6단계만 거치면 지구상 대부분의 사람들과 연결될 수 있다



Wikipedia 페이지 가져오기

- 위키백과의 여섯 다리
  - Eric Idle부터 Kevin Bacon 닿는 최소 클릭수 찾기
  - Eric Idle 위키피디아 URL
    - https://en.wikipedia.org/wiki/Eric_Idle
  - Kevin Bacon 위키피디아 URL
    - https://en.wikipedia.org/wiki/Kevin_Bacon


```
# 임의의 위키 페이지에서 모든 링크 목록 가져오기
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('https://en.wikipedia.org/wiki/Kevin_Bacon')
bs = BeautifulSoup(html, 'html.parser')
for link in bs.find_all('a'):
    if 'href' in link.attrs:
        print(link.attrs['href'])
```

```
/wiki/Wikipedia:Protection_policy#semi
#mw-head
#searchInput
/wiki/Kevin_Bacon_(disambiguation)
/wiki/File:Kevin_Bacon_SDCC_2014.jpg
/wiki/Philadelphia,_Pennsylvania
...
```

불필요한 내용이 많음

단일 도메인 내의 이동 #1

- 필요한 정보만 가져오기 위해, 위키백과 페이지를 분석할 필요
- 위키백과 페이지
  - 위키백과 내부 페이지 링크(internal page link)

```
<li id="n-contactpage" class="mw-list-item">
    <a href="//en.wikipedia.org/wiki/Wikipedia:Contact_us" title="How to contact Wikipedia">
        <span>Contact us</span>
    </a>
</li>
```

  - 항목 링크(article link)
    - 연관 기사 내용 링크
    - article link의 공통점
      - 태그의 id="bodyContent" 내부에 있음
      - URL에는 콜론이 포함되어 있지 않음
      - URL은 /wiki/로 시작

```
<i>
    <a href="/wiki/Footloose_(1984_film)" title="Footloose (1984 film)">Footloose</a>
</i>
 (1984), the controversial historical conspiracy legal thriller
<i>
    <a href="/wiki/JFK_(film)" title="JFK (film)">JFK</a>
</i>
```

항목 링크 찾기

- 항목 링크(연관 기사 링크)만 가져오기
  - 항목 링크의 3가지 특성을 이용
  - 정규식: `^(/wiki/)((?!:).)*$`

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

html = urlopen('https://en.wikipedia.org/wiki/Kevin_Bacon')
bs = BeautifulSoup(html, 'html.parser')
body_content = bs.find('div', {'id': 'bodyContent'})
#
#   ^(정규식 시작)... $(정규식 끝)
#   (/wiki/): '/wiki/를 포함
#   ((?!:).)*: ':' 콜론이 없는 문자열 및 임의의 문자가 0회 이상 반복되는 문자열 검색

pattern = '^(/wiki/)((?!:).)*$'
for link in body_content.find_all('a', href=re.compile(pattern)):
    if 'href' in link.attrs:
        print(link.attrs['href'])
```

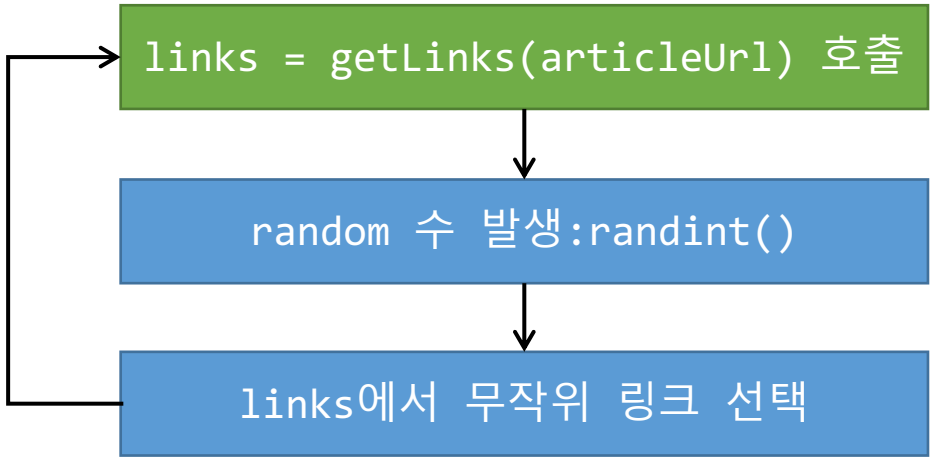
bodyContent 부분을 검색

```
/wiki/Kevin_Bacon_(disambiguation)
/wiki/Philadelphia,_Pennsylvania
/wiki/Kevin_Bacon_filmography
...
/wiki/SUDOC_(identifier)
/wiki/Trove_(identifier)
```

링크간 무작위 이동하기: 동작 과정

- getLinks(articleUrl) 함수 작성
  - 파라미터: 임의의 /wiki/<article_name> 형태를 받음
  - 리턴값: 해당 링크의 모든 URL 목록을 리턴(리스트 형태)

- 동작 과정
  - 시작 URL: https://en.wikipedia.org/wiki/Kevin_Bacon
  - 시작 URL 내부의 연관 기사 URL을 가져옴
  - 연관 기사 URL에서 랜덤하게 하나의 URL 선택
  - 선택된 URL로 이동해서 다시 연관 기사 URL을 가져오는 과정 반복
    - 무한 반복



random.ranint(a, b)
- 랜덤 숫자 N 리턴
- a <= N <= b

링크간 무작위 이동하기: 소스 코드

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

random.seed(datetime.datetime.now())
def getLinks(articleUrl):
    html = urlopen('https://en.wikipedia.org{}'.format(articleUrl))
    bs = BeautifulSoup(html, 'html.parser')
    return bs.find('div', {'id': 'bodyContent'}).find_all('a',
                            href=re.compile('^(/wiki/)((?!:).)*$'))

links = getLinks('/wiki/Kevin_Bacon')
while (len(links)) > 0:
    newArticle = links[random.randint(0, len(links)-1)].attrs['href']
    print(newArticle)
    links = getLinks(newArticle)
```

```
/wiki/Ellen_Barkin
/wiki/Harry_%26_Son
/wiki/Safe_Water_Network
/wiki/Sometimes_a_Great_Notion_(film)
/wiki/Siletz_River
/wiki/Lincoln_County,_Oregon
/wiki/2008_United_States_presidential_election_in_Oregon
/wiki/1864_United_States_presidential_election_in_Oregon
```

같은 페이지를 두 번 크롤링 하지 않기

- getLinks() 함수 수정: set 사용

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

pages = set() # 세트 선언
def getLinks(pageUrl):
    global pages
    html = urlopen('https://en.wikipedia.org{}'.format(pageUrl))
    bs = BeautifulSoup(html, 'html.parser')
    for link in bs.find_all('a', href = re.compile('^(/wiki/)')):
        if 'href' in link.attrs:
            if link.attrs['href'] not in pages:
                newPage = link.attrs['href']
                print(newPage)
                pages.add(newPage)
                getLinks(newPage)
getLinks('')
```

재귀 호출
- Python에서는 1000회로 제한

set 내부에 해당 link가 없는지 확인(not in)

set에 새로운 link 추가: add()함수 사용

```
/wiki/Wikipedia
/wiki/Wikipedia:Protection_policy#semi
/wiki/Wikipedia:Requests_for_page_protection
/wiki/Wikipedia:Requests_for_permissions
```

전체 사이트에서 데이터 수집

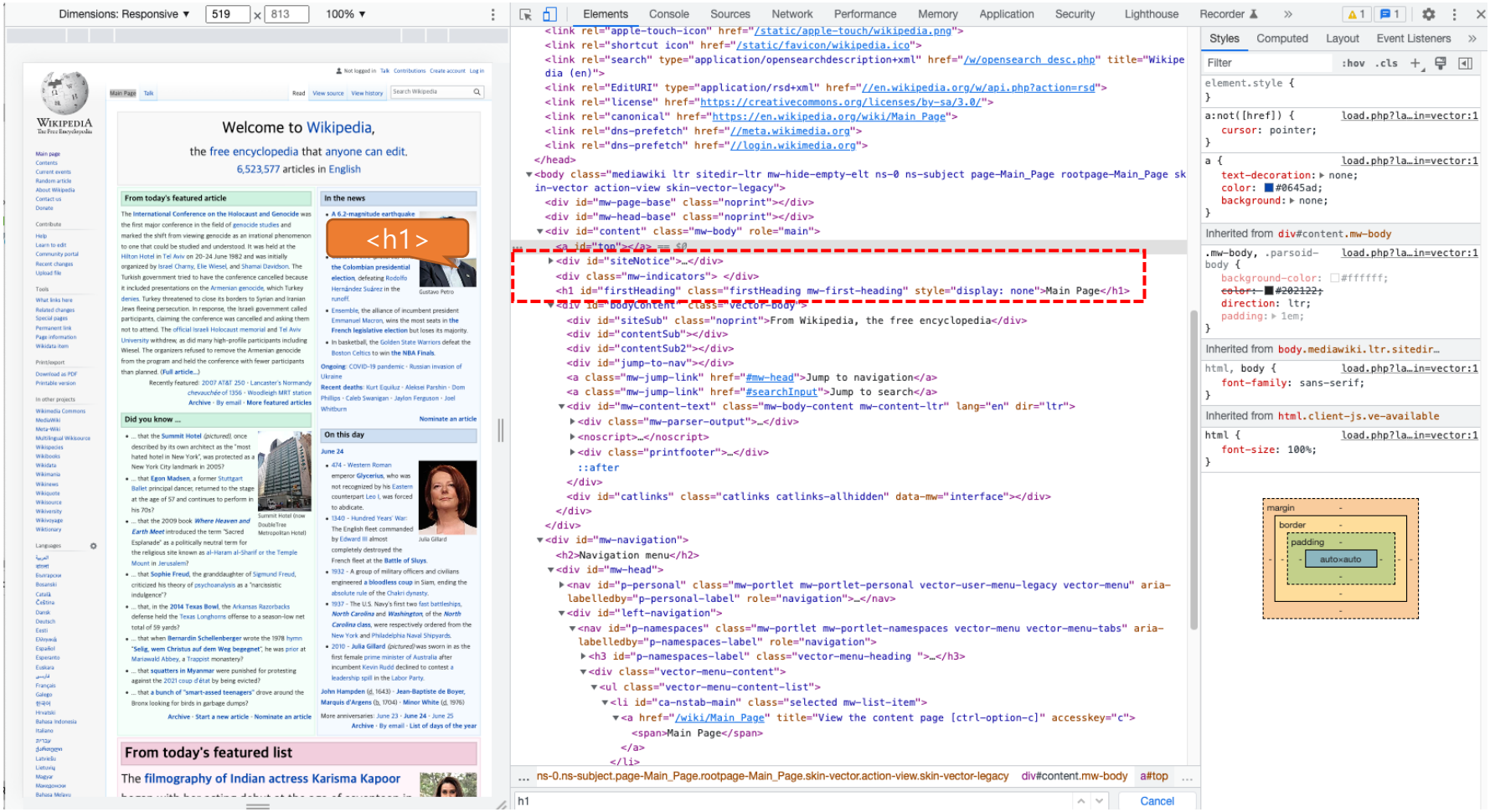
- 페이지 방문 과정에서 필요한 정보를 추출
  - 수집 정보
    - 페이지 제목
    - 첫 번째 문단
    - 편집 페이지 링크 등

- 웹 페이지의 패턴 분석
  - 제목: `<h1>` 태그 사용 (하나만 사용)
  - body 텍스트: div#bodyContent 태그에 있음
  - 첫 번째 문단의 텍스트만 선택
    - `<div id="mw-content-text">` => `<p></p>` 태그 사용
      - `<div>` 태그: division (웹 페이지의 내용을 구분하는데 사용)
      - `<p>`태그: paragraph (하나의 문단을 만들 때 사용)
  - 편집 링크는 항목 페이지에만 존재
    - `li#ca-edit` => `span` => `a`
      - `<li>`태그: list의 약자 (목록을 만드는 태그)
      - `<span>`태그: 인라인 컨테이너, 스타일을 나타내기 위해 사용

전체 사이트 데이터 수집 소스

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

pages = set()
def getLinks(pageUrl):
    global pages
    html = urlopen('https://en.wikipedia.org{}'.format(pageUrl))
    bs = BeautifulSoup(html, 'html.parser')
    try:
        print(bs.h1.get_text())
        print(bs.find(id='mw-content-text').find_all('p')[0])
        print(bs.find(id='ca-edit').find('span').find('a').attrs['href'])
    except AttributeError:
        print('this page is missing something! Continuing.')

    for link in bs.find_all('a', href=re.compile('^(/wiki/)')):
        if 'href' in link.attrs:
            if link.attrs['href'] not in pages:
                newPage = link.attrs['href']
                print('-'*20)
                print(newPage)
                pages.add(newPage)
                getLinks(newPage)

getLinks('')
```

wikipedia 초기 화면 구성



인터넷 크롤링

- 웹 크롤러를 만들기 전에 고려할 사항
  - 수집하려는 데이터는 무엇인가?
  - 특정 웹사이트에 도달하면, 새 웹사이트 링크를 따라가야 할까?
  - 특정 사이트를 제외할 것인가?
    - 다른 언어를 사용하는 웹사이트 정보 수집 여부
  - 저작권 침해 관련 문제는 없을까?

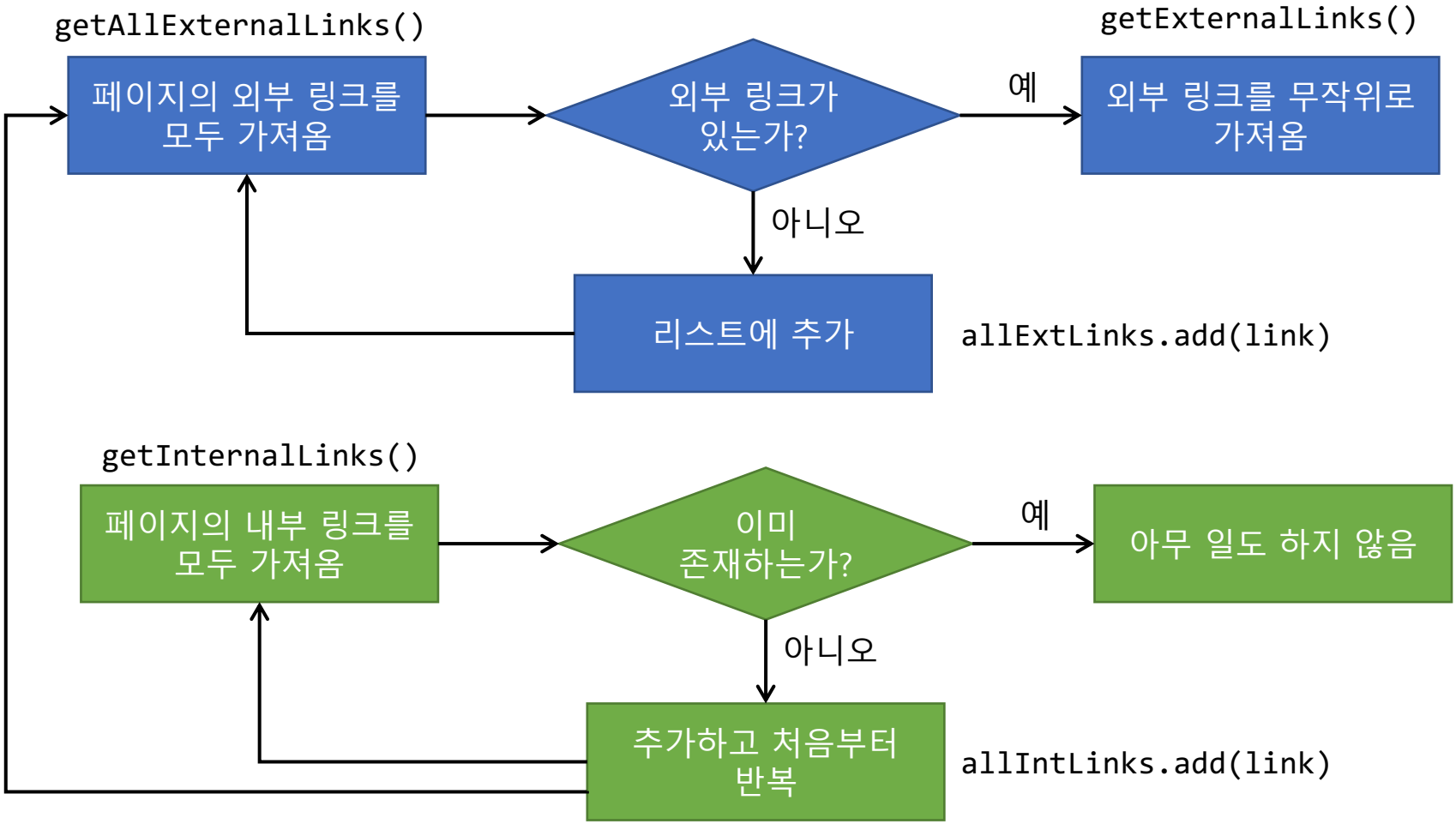
- 예제
  - 시작 URL: http://oreilly.com

인터넷 크롤링 예제 소스 분석 내용 #1

- 정규식
  - `href=re.compile('^(/|.*' + includeUrl + ')')`
    - '/'로 시작하는 링크를 찾음
    - ^: 문자열 시작
    - ( ): 그룹
    - /.*: '/' 문자 또는 임의의 한문자가 없거나 여러 개 존재 (zero or more)
  - `href=re.compile('^(http|www)((?!' + excludeUrl + ').)*$')`
    - (http|www): http 또는 www로 시작하는 문자열
    - (?! excludeUrl): excludeUrl 문자열이 존재하지 않는 링크
      - ➢ 전방 부정 탐색

인터넷 크롤링: 인터넷 사이트 탐색 순서도

- 외부 링크 수집 과정



인터넷 크롤링: URL 구조

- URL 구조

```
scheme://netloc/path;parameters?query#fragment
```

  - scheme: 'http' 또는 'https'
    - ftp, file, gopher, mms, news, nntp, sftp, telnet 등
  - netloc: 인터넷 주소

- urllib.urlparse
  - 파이썬 표준 라이브러리
  - HTTP요청, 파싱과 관련된 패키지

```
from urllib.parse import urlparse

urlString1 = 'https://shopping.naver.com/home/p/index.naver'

url = urlparse(urlString1)
print(url.scheme)
print(url.netloc)
print(url.path)
```

```
https
shopping.naver.com
/home/p/index.naver
```

인터넷 크롤링: 예제 코드 #1

```
from urllib.request import urlopen
from urllib.parse import urlparse
from bs4 import BeautifulSoup
import re
import datetime
import random

pages = set()
random.seed(datetime.datetime.now())

# Retrieves a list of all Internal links found on a page
def getInternalLinks(bs, includeUrl):
    includeUrl = '{}://{}'.format(urlparse(includeUrl).scheme,
                                  urlparse(includeUrl).netloc)
    internalLinks = []
    # Finds all links that begin with a "/"
    for link in bs.find_all('a', href=re.compile('^(/|.*' + includeUrl + ')')):
        if link.attrs['href'] is not None:
            if link.attrs['href'] not in internalLinks:
                if (link.attrs['href'].startswith('/')):
                    internalLinks.append(includeUrl + link.attrs['href'])
                else:
                    internalLinks.append(link.attrs['href'])
    return internalLinks
```

인터넷 크롤링: 예제 코드 #2

```
# Retrieves a list of all external links found on a page
def getExternalLinks(bs, excludeUrl):
    externalLinks = []
    # Finds all links that start with "http" that do
    # not contain the current URL
    for link in bs.find_all('a', href=re.compile('^(http|www)((?!' +
                                                 excludeUrl + ').)*$')):
        if link.attrs['href'] is not None:
            if link.attrs['href'] not in externalLinks:
                externalLinks.append(link.attrs['href'])
    return externalLinks

def getRandomExternalLink(startingPage):
    html = urlopen(startingPage)
    bs = BeautifulSoup(html, 'html.parser')
    externalLinks = getExternalLinks(bs, urlparse(startingPage).netloc)

    if len(externalLinks) == 0: # 외부 링크가 없으면 내부 링크 검색
        print('No external links, looking around the site for one')
        domain = '{}://{}'.format(urlparse(startingPage).scheme,
                                  urlparse(startingPage).netloc)
        internalLinks = getInternalLinks(bs, domain)
        return getRandomExternalLink(internalLinks[random.randint(0,
                                                    len(internalLinks) - 1)])
    else: # 랜덤하게 외부 링크 선택
        return externalLinks[random.randint(0, len(externalLinks) - 1)]
```

인터넷 크롤링: 예제 코드 #3

```
def followExternalOnly(startingSite):
    externalLink = getRandomExternalLink(startingSite)
    print('Random external link is: {}'.format(externalLink))
    followExternalOnly(externalLink)


followExternalOnly('http://oreilly.com')
```

```
Random external link is: https://play.google.com/store/apps/details?id=com.safariflow.queue
Random external link is: https://support.google.com/googleplay/?p=about_play
Random external link is: https://www.google.co.kr/intl/en/about/products?tab=uh
Random external link is: https://www.google.com/analytics/?utm_medium=referral-internal&utm_source=google-products&utm_campaign=product-cross-promo&utm_content=analytics-icon
Random external link is: https://developers.google.com/analytics/?utm_source=marketingplatform.google.com&utm_medium=et&utm_campaign=marketingplatform.google.com%2Fabout%2Fanalytics%2F
Random external link is: https://www.youtube.com/user/googleanalytics
Random external link is: https://developers.google.com/youtube
Random external link is: https://www.youtube.com/user/YouTubeDev
```

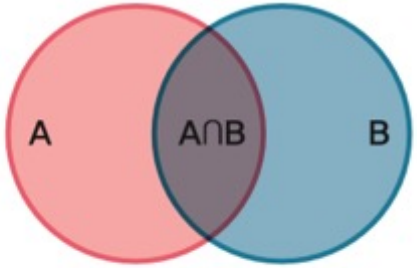
인터넷 크롤링: 외부, 내부 링크 모두 저장

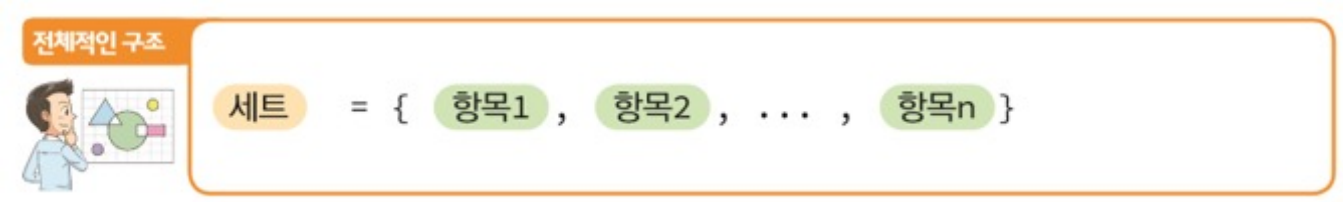
```
# Collects a list of all external URLs found on the site
allExtLinks = set()
allIntLinks = set()

def getAllExternalLinks(siteUrl):
    html = urlopen(siteUrl)
    domain = '{}://{}'.format(urlparse(siteUrl).scheme,
                              urlparse(siteUrl).netloc)
    bs = BeautifulSoup(html, 'html.parser')
    internalLinks = getInternalLinks(bs, domain)
    externalLinks = getExternalLinks(bs, domain)

    for link in externalLinks:
        if link not in allExtLinks:
            allExtLinks.add(link)
            print(link)
    for link in internalLinks:
        if link not in allIntLinks:
            allIntLinks.add(link)
            getAllExternalLinks(link)

allIntLinks.add('http://oreilly.com')
getAllExternalLinks('http://oreilly.com')
```

참고 자료

세트(Set)

- 세트(set)는 우리가 **수학에서 배웠던 집합**이다.
- 세트는 중복되지 않은 항목들이 모인 것
- 세트의 항목 간에는 순서가 없다.
- 파이썬에서 세트를 생성하려면 중괄호 기호{ }를 사용





세트: 중복 요소 자동 제거

- 세트는 집합이기 때문에
  - 요소가 중복되면 자동으로 중복된 요소를 제거함
- 중복된 원소를 포함하는 리스트의 경우, set으로 변경

```
cities = {“Paris”, “Seoul”, “London”, “Berlin”, “Paris”, “Seoul”}
print(cities)
mySet = {1, 2, 3, 2, 5, 4, 5, 3}
print(mySet)

myList = [1, 2, 3, 2, 5, 4, 5, 3]
print(myList)
mySet = set(myList) # 리스트를 세트로 변경
print(mySet)
myList = list(mySet) # 세트를 리스트로 변경
print(myList)
-----------------
실행 결과
{'Paris', 'Seoul', 'Berlin', 'London’}
{1, 2, 3, 4, 5}
[1, 2, 3, 2, 5, 4, 5, 3]
{1, 2, 3, 4, 5}
[1, 2, 3, 4, 5]
```

in 연산자

```
numbers = {2, 1, 3}
if 1 in numbers:
    print("집합 안에 1이 있습니다.")

---------------
실행 결과

집합 안에 1이 있습니다.
```

if 문장에서 사용된 **in**은 **존재 여부**를 확인

```
numbers = {2, 1, 3}
for x in numbers:
    print(x, end=" ")

---------------
실행 결과

1 2 3
```

for 문장에서 사용된 **in**은 **요소를 하나씩 가져옴**

출력 순서는 입력 순서와 다를 수 있음

세트에 요소 추가하기

```
numbers = { 2, 1, 3 }
numbers[0]
...
TypeError: 'set' object does not support indexing
```

set는 순서가 없기 때문에 index를 사용하여 세트 항목에 접근할 수 없음

- add(항목): 요소 추가

```
numbers = { 2, 1, 3 }
numbers.add(4)    # Set에 요소 추가
print(numbers)
numbers.discard(4) # discard(): Set에서 요소 삭제
print(numbers)     # discard()는 없는 요소를 삭제해도 예외 발생하지 않음
numbers.remove(3)  # remove(): Set에 없는 요소를 삭제하면 예외를 발생시킴
print(numbers)
numbers.clear()    # clear(): 모든 요소 삭제
print(numbers)
---------------
실행 결과
{1, 2, 3, 4}
{1, 2, 3}
{1, 2}
set()
```

Questions?