Fastcampus Sprint - Programming

Day 1. 일단 웹스크래퍼를 만들어보자!

Introduce

최우영

- Co-founder, Developer at Disceptio
- Solution Architect, Web Developer, Instructor
- Skills: Python, Golang, Julia, Node.js, Google tag manager ...

blog: https://ulgoon.github.io/

github: https://github.com/ulgoon/

email: me@ulgoon.com

Goal

- 파이썬을 잘 몰라도 작업의 흐름을 이해한 뒤, 웹 스크래퍼를 만들 수 있다.
- HTML/CSS의 문법을 이해하고, 이를 웹 스크래핑에 활용하기 위한 개발자도구를 능숙하게 사용할 수 있다.
- 코드를 본격적으로 생산할 수 있고, 웹 스크래퍼의 효율을 높여줄 Python 기본 문법을 이해할 수 있다.
- 다양한 형태의 웹페이지의 요소 혹은 일반적인 구조를 갖지 않은 형태의 데이터에 접근하여 데이터를 가져올 수 있다.

Index

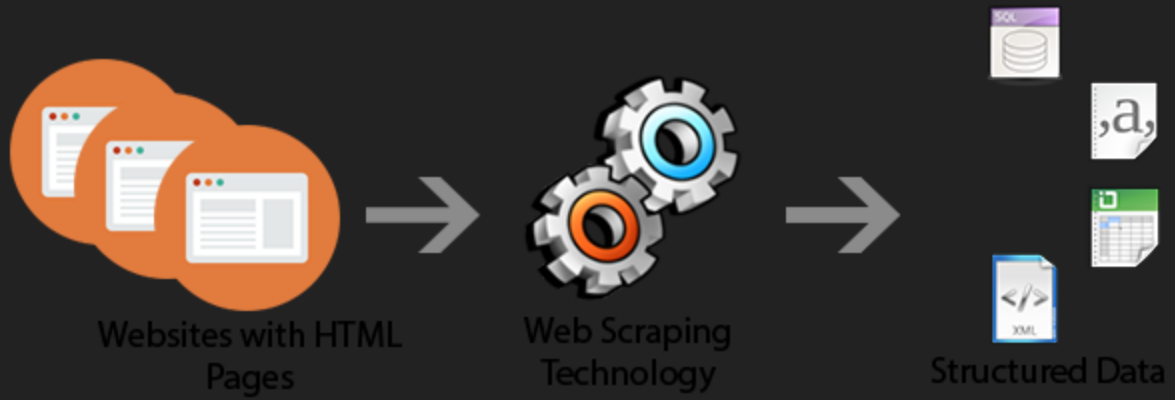
- 웹페이지 소스 가져오기
- 원하는 요소 선택하기
- 선택한 요소 정리하기
- 정리한 요소 저장하기

python 이란?

- 1991년 Guido Van Rossum이 발표한 고급 프로그래밍 언어
- 인터프리터 언어
- 객체지향적
- 동적타이핑

Web Scraping 이란?

Scraping: 데이터를 수집하는 행위



웹페이지 소스 가져오기

파이썬과 라이브러리 설치하기(Python)

- for windows: https://www.python.org/ftp/python/3.7.4/python-3.7.4-amd64.exe
- for ubuntu: `$ sudo apt-get install python3`
- for macOS

```
$ xcode-select --install
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
$ brew install python
```

`$ python3 --version` 으로 성공유무를 확인합니다

파이썬과 라이브러리 설치하기(Library)

```
$ pip install requests
$ pip install beautifulsoup4
$ pip install lxml
$ pip install openpyxl
$ pip install jupyter
```

`$ pip list` 로 설치 내역을 확인합니다

웹페이지 소스 가져오기

```
$ jupyter notebook
>>> import requests
>>> response = requests.get("https://www.google.com/")
>>> response
<    ..    >
>>> response.headers
{
..
}
>>> response.text
<!doctype html>..
```

사용할 라이브러리 불러오기

```
import requests
from bs4 import BeautifulSoup
import lxml
from openpyxl import Workbook, load_workbook
import json
import time
```

원하는 요소 선택하기

1. 앙대포털 검색어 가져오기(리스트형 데이터)

N사

```
response = requests.get("https://www.naver.com/")
executed_time = time.ctime()
response.status_code
page_text = response.text
soup = BeautifulSoup(page_text, "lxml")
ul_text = soup.find("ul", attrs={"class":"ah_l"})
li_list = ul_text.find_all("li")
nv_keywords = []

for li in li_list:
    rank = int(li.find("span", attrs={"class":"ah_r"}).text)
    keyword = li.find("span", attrs={"class":"ah_k"}).text
    nv_keywords.append((executed_time, rank, keyword))
nv_keywords
```

D사

```
response = requests.get("https://www.daum.net")
executed_time = time.ctime()
page_text = response.text
soup = BeautifulSoup(page_text, "lxml")

ol_text = soup.find("ol", attrs={"class":"list_hotissue"})
li_list = ol_text.find_all("li")
du_keywords = []
for li in li_list:
    item = li.select("div.rank_cont:nth-of-type(1)")[0]
    rank = int(item.find("span", attrs={"class": "ir_wa"}).text[:-1])
    keyword = item.find("a", attrs={"class":"link_issue"}).text
    du_keywords.append((executed_time, rank, keyword))
du_keywords
```

2. 박스오피스 데이터 가져오기(테이블형 데이터)

```
base_uri = "https://rottentomatoes.com"
executed_time = time.ctime()
response = requests.get(base_uri)
page_text = response.text
soup = BeautifulSoup(page_text, "lxml")

table_data = soup.find("table", attrs={"id":"Top-Box-Office"})
tr_list = table_data.find_all("tr")
```

선택한 요소 정리하기

추출한 데이터를 사용하기 편리한 형태로 가공합니다.

ex) "10" -> 10
"$1.0M" -> 1000000

```
box_office_list = []
for tr in tr_list:
    a_list = tr.find_all("a")
    score = int(a_list[0].find("span", attrs={"class":"tMeterScore"}).text[:-1])
    url = base_uri + a_list[0]["href"]
    movie_name = a_list[1].text
    revenue = a_list[2].text[1:]

    # convert scale to zeros
    if revenue[-1] == 'M':
        digits = 6
    elif revenue[-1] == 'B':
        digits = 9
    elif revenue[-1] == 'K':
        digits = 3

    if revenue.find('.') == -1:
        revenue = int(revenue[:-1] + '0'*digits)
    elif revenue[0] == '0':
        revenue = int(revenue[1:-1].replace(".","") + '0'*(digits-1))
    else:
        revenue = int(revenue[:-1].replace(".","") + '0'*(digits-1))

    box_office_list.append((executed_time, url, score, movie_name, revenue))
box_office_list
```

정리한 요소 저장하기

정리한 요소를 저장하기 위해 저장할 포맷을 먼저 정합니다.

다양한 파일 포맷

- **TXT(TeXT)**
- **CSV(Comma Spread Values)**
- TSV(Tab Spread Values)
- XML(eXtensible Markup Language)
- **XLS, XLSX(eXceL Spreadsheet (XML based))**
- **json(javascript object notation)**

TXT

```
with open('nvquery.txt', 'a') as f:
    for kw in nv_keywords:
        f.write(kw[0] + "의" +
        str(kw[1]) + "위는 " + kw[2] + "입니다.\n")
```

파일 읽기

```
with open('nvquery.txt', 'r') as f:
    text = f.readlines()
    for item in text:
        print(item)
```

CSV

```
with open('nv_query.csv', 'a') as f:
    for kw in nv_keywords:
        f.write(kw[0]+","+str(kw[1])+","+kw[2]+"\n")
```

XLSX

```
workbook_name = 'nv_query.xlsx'
try:
    workbook = load_workbook(workbook_name)
except FileNotFoundError as e:
    workbook = Workbook()
worksheet = workbook.active
for keyword in nv_keywords:
    worksheet.append(keyword)

workbook.save('nv_query.xlsx')
```

write and update json

```
try:
    with open('nv_query.json', 'r+') as f:
        data = json.load(f)

        data["data"].append(nv_object)
        f.seek(0)
        json.dump(data,f)
        f.truncate()
except FileNotFoundError:
    with open('nv_query.json', 'w') as f:
        json.dump({"data":[nv_object]}, f)
```

read json

```
with open('nv_query.json', 'r') as f:
    data = json.load(f)
data
```