python™

스크래핑

웹 스크래핑

- 웹 스크래핑
  - 모든 웹은 HTML로 구성되어 있으므로, HTML의 규칙을 파악한다면 얼마든지 HTML에서 필요한 정보를 가져올 수 있다. 이러한 과정을 일반적으로 웹 스크래핑(web scrapping) 이라고 한다.

웹 스크래핑

- 웹에서 데이터 다운로드하기
  - https://www.google.com/googlebooks/uspto-patents-grants-text.html
  - http://storage.googleapis.com/patents/grant_full_text/2014/ipg140107.zip

```
import urllib.request   # urllib 모듈 호출
# 다운로드 URL 주소
url = "http://storage.googleapis.com/patents/grant_full_text/2014/ipg140107.zip"

print("Start Download")
# urlretrieve() 함수 호출(URL 주소, 다운로드할 파일명),
# 결과값으로 다운로드한 파일명과 Header 정보를 언패킹
fname, header = urllib.request.urlretrieve(url, 'ipg140107.zip')

print("End Download")
```

웹 스크래핑

- HTML 파싱
  - 웹 페이지의 HTML을 분석하여 필요한 필요한 정보를 추출하는 과정
  - https://sports.news.naver.com/kbaseball/record/index.nhn?category=kbo
  - 아래 웹 페이지에서 팀 순위별 팀 이름만 축출해야 한다면 어떻게 해야 할까?

KBO리그 기록 및 순위

팀순위 2020. 현재

순위	팀	경기수	승	패	무	승률	게임차	연속	출루율	장타율	최근 10경기
1	NC	94	57	35	2	**0.620**	0.0	1패	0.363	0.470	6승-4패-0무
2	키움	101	60	41	0	**0.594**	1.5	1승	0.358	0.423	6승-4패-0무
3	LG	98	56	40	2	**0.583**	3.0	5승	0.354	0.444	6승-3패-1무
4	두산	97	53	41	3	**0.564**	5.0	1승	0.367	0.438	6승-3패-1무
5	KT	94	50	43	1	**0.538**	7.5	1승	0.355	0.440	5승-5패-0무
6	KIA	94	49	45	0	**0.521**	9.0	2승	0.350	0.406	5승-5패-0무
7	롯데	92	47	44	1	**0.516**	9.5	1패	0.348	0.403	5승-5패-0무
8	삼성	96	43	52	1	**0.453**	15.5	2패	0.337	0.395	3승-7패-0무
9	SK	97	32	64	1	**0.333**	27.0	4패	0.333	0.384	3승-7패-0무
10	한화	95	26	68	1	**0.277**	32.0	3패	0.312	0.326	4승-6패-0무

웹 스크래핑

- HTML 파싱
  - HTML 의 소스 코드에서 원하는 데이터가 어느 Tag에 위치하는지, 어떤 패턴으로 되어 있는지를 찾는다면 원하는 팀 이름을 쉽게 축출

```
                <td class="tm">
                    <div>


                                <img src="https://dthumb-phinf.pstatic.net/?src=https:/
alt="구단로고" onerror="javaScript:noTeamEmblemImg(this, 'team', 'NC');">


                        <span id="team_NC">NC</span>
                    </div>
                </td>
                <td><span>94</span></td>
                <td><span>57</span></td>
                <td><span>35</span></td>
                <td><span>2</span></td>
                <td><strong>0.620</strong></td>
                <td><span>0.0</span></td>
                <td><span>1패</span></td>
                <td><span>0.363</span></td>
                <td><span>0.470</span></td>
                <td><span>6승-4패-0무</span></td>
            </tr>

            <tr>
                <th><strong>2</strong></th>
                <td class="tm">
                    <div>


                                <img src="https://dthumb-phinf.pstatic.net/?src=https:/
alt="구단로고" onerror="javaScript:noTeamEmblemImg(this, 'team', 'WO');">


                        <span id="team_WO">키움</span>
                    </div>
                </td>
```

찾은 패턴

```
<span id="team_NC">NC </span>
<span id="team_WO">키움</span>
<span id="team_LG">LG </span>
…
```

웹 스크래핑

- 정규 표현식 문법
  - 정규 표현식을 배우기 위해 기본으로 알아야 하는 개념은 메타문자(meta-characters) 이다. 메타문자는 문자를 설명하기 위한 문자로, 문자의 구성을 설명하기 위해 원래의 의미가 아니라 다른 의미로 쓰이는 문자를 뜻한다.

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

  - 기본 메타문자 []
    - 먼저 대괄호 [ ]는 [ ] 블록 사이의 문자와 매칭. [ ]에는 or의 의미.
      예를 들어, [abc]는 어떤 텍스트에 a 또는 b 또는 c라는 텍스트가 있는지 확인
  - 반복 관련 메타문자 -, +, *, ?, { }
    - + : 해당 글자가 1개 이상 출현
    - {} : 출현 횟수를 조정해야 할 때 사용하는 메타문자는 중괄호
    - * : 해당 글자가 0번부터 무한대까지 반복
    - () : 묶음을 표시
      [.]는 일반적인 마침표를 뜻하고 (.)는 줄 바꿈 기호를 제외한 전체 문자를 뜻한다.
    - 메타문자 |나 ^은 or와 not의 의미, 정규 표현식의 처음과 끝에는 메타문자 ^과 $를 붙인다.

웹 스크래핑

- 정규 표현식 문법
  - 전화번호 찾기

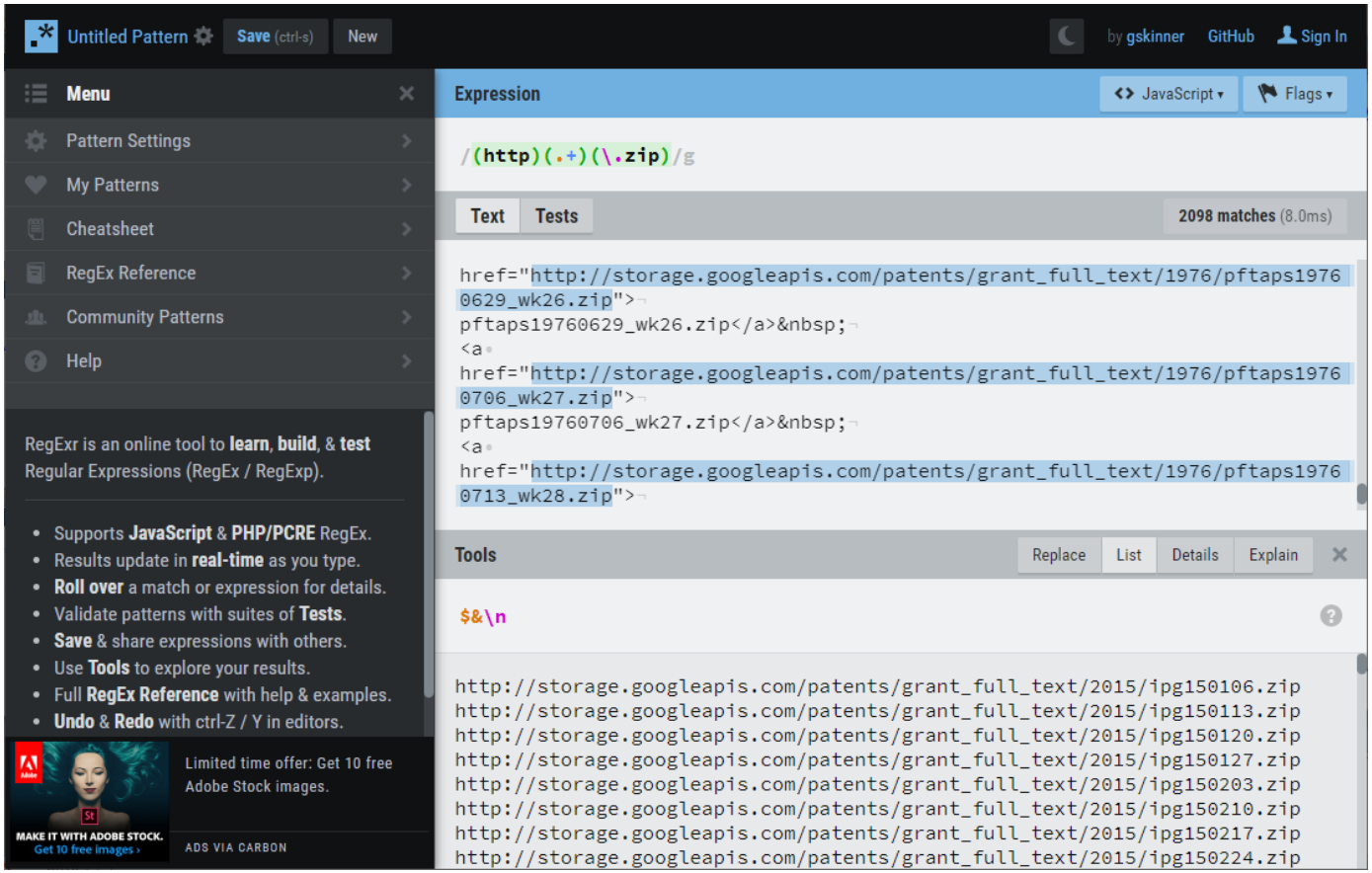
```
[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]
```

```
[0-9]+-[0-9]+-[0-9]+
```

```
[0-9]{3}-[0-9]{3,4}-[0-9]{4}
```

웹 스크래핑

- 정규 표현식 문법
  - 정규식 예제
    - https://www.google.com/googlebooks/uspto-patents-grants-text.html
    - 위 페이지의 소스코드를 정규 표현식 연습장 웹 사이트(http://www.regexr.com) 에 붙여 넣기



웹 스크래핑

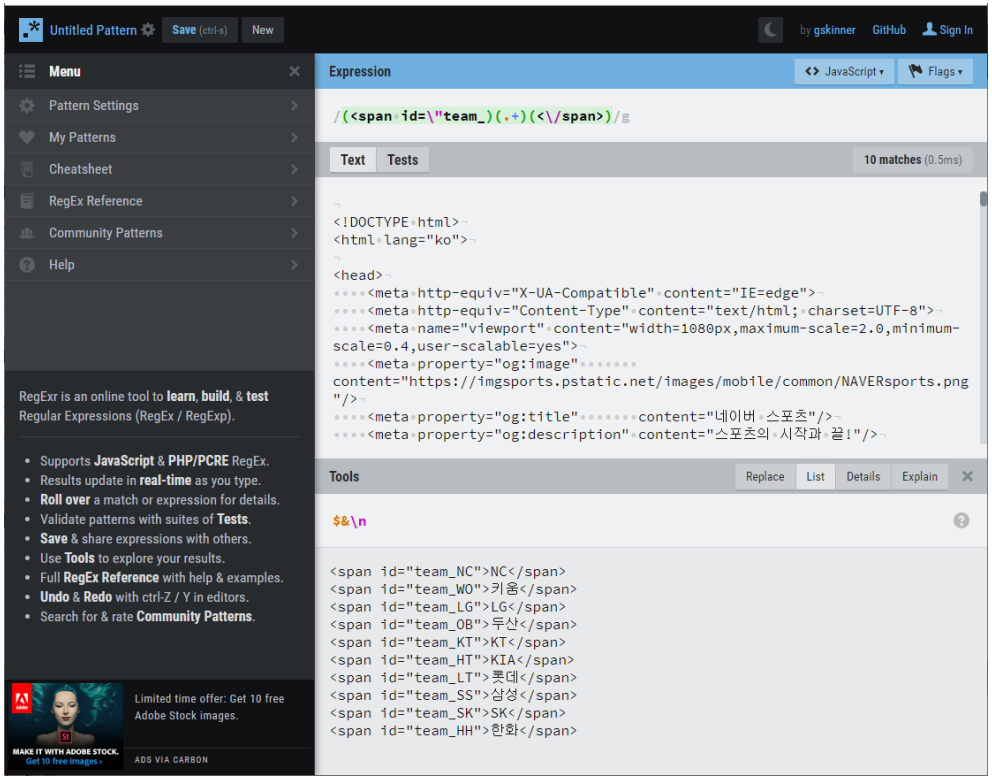
- 파일 자동 다운로드 프로그램

```
# urllib 모듈 호출
import urllib.request
# 정규싱을 사용
import re

# url 값 입력
url = "http://www.google.com/googlebooks/uspto-patents-grants-text.html"
# url 열기
html = urllib.request.urlopen(url)
# html 파일 읽고, 문자열로 변환
html_contents = str(html.read().decode("utf8"))

# 정규식으로 특정 데이터 찾기
url_list = re.findall(r"(http)(.+)(zip)", html_contents)

for url in url_list:
    print(url)
    # 출력된 튜플 형태 데이터 문자열로 join
    full_url = "".join(url)
    print(full_url)
    # file_name에 다운로드할 파일명 입력한 후, 파일 다운로드
    file_name = full_url.split("/")[-1]
    fname, header = urllib.request.urlretrieve(full_url, file_name)
    print("End Download")
```

웹 스크래핑

- 정규식으로 야구 순위 찾기
  - 정규 표현식(regular expression) 은 일종의 문자를 표현하는 공식으로, 특정 규칙이 있는 문자열 집합을 추출할 때 자주 사용하는 기법
  - 정규 표현식 연습장 웹 사이트(http://www.regexr.com)

```
(<span id=₩"team_)(.+)(<₩/span>)
```



웹 스크래핑

- 정규식으로 야구 순위 찾기

```
import urllib.request
import re

#URL 설정
url = "https://sports.news.naver.com/kbaseball/record/index.nhn?category=kbo"

# url 열기
html = urllib.request.urlopen(url)
# html 파일 읽고, 문자열로 변환
html_contents = str(html.read().decode("utf-8"))

# 정규식으로 특정 데이터 찾기
tag_list = re.findall(r"(<span id=\"team_)(.+)(</span>)", html_contents)

# 순위 는 index로 표현하기 위해 enumerate 사용
for index, result in enumerate(tag_list, start=1):
    print(result);
    print(index, '위 ', result[1].split('>')[1])
```

python™

웹 크롤링

웹 크롤링

- 웹 크롤링 (Web crawling)
  - 웹 스크래핑(Web Scraping)이라고도 하며, 컴퓨터소프트웨어기술로 각종 웹사이트들에 서원하는정보를추출하는것
  - 웹 크롤러

    : 인터넷에 있는 웹 페이지를 방문해서 자료를 수집하는 일을 하는 프로그램
  - 크롤링을 위한 선행학습
    1. 웹(Web)의 개념
    2. HTML, CSS, Javascript구조 및 태그
    3. 파이썬 기초

웹 크롤링

- 웹 크롤링의 기법
  - HTML 페이지를 가져와서, HTML/CSS등을 파싱 하고, 필요한 데이터만 추출하는 기법
  - Open API(Rest API)를 제공하는 서비스에 Open API를 호출해서, 받은 데이터 중 필요한 데이터만 추출하는 기법
  - 브라우저를 프로그래밍으로 조작해서, 필요한 데이터만 추출하는 기법
- 웹 크롤링 예시
  - 구글 검색엔진
    - 구글에서는 크롤링 봇을 통해 수집한 결과를 검색 결과로 보여줌
  - 쇼핑몰
    - 경쟁사의 제품 정보 등을 크롤링 하여 사용
  - API 외의 정보
    - API에서 제공하는 정보 외에 필요한 정보들을 크롤링 하여 사용

웹 크롤링

- BeautifulSoup 모듈
  - 홈페이지 내 데이터를 쉽게 추출할 수 있도록 도와주는 파이썬 외부 라이브러리
  - HTML 태그들을 파서를 활용해 사용하기 편한 파이썬 객체로 만들어 제공
  - html 외에 xml 파서도제공
  - 웹 문서 구조를 알고 있다면, 아주 편하게 원하는 데이터를 뽑아 활용 할 수 있음

- 기존 방식과의 차이
  - 기존 방식

    정규 표현식, 문자열 함수 등을 활용하여 홈페이지 텍스트 내 패턴을 분석하여 하나 씩 원하는 데이터를 찾아가는 형식
  - BeautifulSoup 이용 방식

    HTML 문서를 태그를 기반으로 구조화하여 태그로 원하는 데이터를 찾아가는 형식

웹 크롤링

- BeautifulSoup 모듈 설치
  - pip을 활용해 설치해 주어야만 사용 가능
    - 콘솔창에서 pip install beautifulsoup4 명령어로 설치
    - jupyter notebook에서 !pip install beautifulsoup4 명령어로 설치
    - Anaconda Navigator에서 검색 후 설치

```
PS C:\Users\jin\Documents\Github\aiajr202003\python> pip install beautifulsoup4
                                                                    Collecting beautifulsoup4
  Using cached beautifulsoup4-4.9.1-py3-none-any.whl (115 kB)
Requirement already satisfied: soupsieve>1.2 in c:\users\jin\appdata\local\programs\python\python38-32\lib\site-packages (from bea
utifulsoup4) (2.0.1)
Installing collected packages: beautifulsoup4
Successfully installed beautifulsoup4-4.9.1
PS C:\Users\jin\Documents\Github\aiajr202003\python> pip show beautifulsoup4
                                                                    Name: beautifulsoup4
Version: 4.9.1
Summary: Screen-scraping library
Home-page: http://www.crummy.com/software/BeautifulSoup/bs4/
Author: Leonard Richardson
Author-email: leonardr@segfault.org
License: MIT
Location: c:\users\jin\appdata\local\programs\python\python38-32\lib\site-packages
Requires: soupsieve
Required-by:
PS C:\Users\jin\Documents\Github\aiajr202003\python>
```

웹 크롤링

- BeautifulSoup 모듈 사용법

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

  – from bs4 import BeautifulSoup로 모듈을 호출하여 사용

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(target, "html.parser")
```

  – BeautifulSoup은 HTML을 파싱하여 구조화하는 모듈로 urllib, requests 모듈등과 함께 사용

```
from urllib import request
from bs4 import BeautifulSoup

url = "https://sports.news.naver.com/kbaseball/record/index.nhn?category=kbo"
# url 열기 -> 응답
target = request.urlopen(url)

soup = BeautifulSoup(target, "html.parser")
```

웹 크롤링

- BeautifulSoup 모듈 사용법(검색)
  - 태그
    - HTML의 해당 태그에 대한 첫 번째 정보를 가져옴
  - 태그['속성'] : HTML 해당 태그의 속성에 대한 첫 번째 정보를 가져옴

```
from urllib import request
from bs4 import BeautifulSoup

url = "https://sports.news.naver.com/kbaseball/record/index.nhn?category=kbo"
# url 열기 -> 응답
target = request.urlopen(url)

soup = BeautifulSoup(target, "html.parser")

print(soup.title)
print(soup.title.name)
print(soup.title.string)

print(soup.img)
print(soup.img['alt'])
print(soup.img['height'])
print(soup.img['width'])
```

웹 크롤링

- BeautifulSoup 모듈 사용법(검색)
  - find()
    - HTML의 해당 태그에 대한 첫 번째 정보를 가져옴
    - find(속성='값') : HTML 해당 속성과 일치하는 값에 대한 첫 번째 정보를 가져옴
  - find_all()
    - 태그를 모두 얻거나, 특정이름으로 첫 번째 태그 말고 좀 더 복잡한 태그를 축출할 때 사용
    - re 모듈을 이용해서 정규식을 이용해서 축출

웹 크롤링

- BeautifulSoup 모듈 사용법(검색)

```
# 하나의 태그를 축출
print(soup.find('a', class_='logo'))
print(soup.find('a', 'logo'))

print(soup.find(id='team_LG'))
print('_____________________________________________________________________')

for tag in soup.find_all(text="순위"):
    print('>>>>> ', tag)
print('_____________________________________________________________________')

for tag in soup.find_all(text=re.compile('순위')):
    print('>>>>> ', tag)
print('_____________________________________________________________________')

# 열러개 태그를 축출
for tag in soup.find_all('a'):
    print(tag)
print('_____________________________________________________________________')

for tag in soup.find_all('a', attrs={'class', 'logo'}):
    print(tag)
print('_____________________________________________________________________')

for tag in soup.find_all(['a','img']):
    print(tag)
print('_____________________________________________________________________')
```

웹 크롤링

- BeautifulSoup 모듈 사용법(검색)
  - select()
    - CSS 의 표준 선택자를 이용해서 태그를 축출

```
# select() 메서들르 이용해서 span 태그를 찾습니다.
for index, team in enumerate(soup.select('td>div>span[id]'), start=1):
    print(index, '위 ', team.string)
```