IPA 주관 인공지능센터 기본(fundamental) 과정: 특강

- GitHub link: here
- E-Mail: windkyle7@gmail.com

특강 주제(오후): Web Crawling & Scraping

목표: 수집을 하기 위해서는 먼저 Web Server에 request를 보내고, 그에 대한 response를 받게 되면 데이터를 스크래핑하여 가져온다.

Requests

`requests`는 `urllib` 기반으로 만들어진 request를 위한 High-level 패키지이다.

In [1]:

```
import requests
```

Unicode => byte => Persent Encoding

url에 요청을 보낼 때, 유니코드 데이터를 바이트로 변환한 후 퍼센트 인코딩을 수행한다.

- ex) 박보영 => %EB%B0%95%EB%B3%B4%EC%98%81

예를 들어, 구글에 '박보영'이라는 키워드로 검색한다면, 아래의 링크처럼 `GET` 방식으로 url에 파라미터가 붙어 보내질 것이다.

링크: https://www.google.com/search?q=%EB%B0%95%EB%B3%B4%EC%98%81

즉 `GET` 방식으로 요청을 보낼 때, url은 `프로토콜://도메인명/서비스명?파라미터` 형식으로 구성되어있다. 파라미터는 각각 `key=value` 쌍으로 이루어진다.

In [2]:

```
url = 'https://www.google.com/search'
params = {
    'q': '박보영',
    'ie': 'UTF-8',
}
```

요청 시 실제 브라우저에서 요청하는 것처럼 `User-Agent`를 지정해준다. 이는 어떤 브라우저를 사용하느냐에 따라 다르다.

In [3]:

```
headers = {
    'User-Agent':
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.157 Safari/537.36'
}
```

`requests.get`으로 서버에 GET 방식으로 request를 보내면 그에 대한 `response` 객체를 반환해준다.

In [4]:

```
resp = requests.get(url, params=params, headers=headers)
```

상태 코드를 확인해보면 200 코드가 받아졌다. 200 코드는 요청이 잘 수행되었다는 의미이다.

HTTP Status Code

HTTP 상태 코드가 의미하는 바는 아래와 같다.

- 1xx (조건부 응답)
- 2xx (성공)
- 3xx (리다이렉션 완료)
- 4xx (요청 오류, 클라이언트 측 잘못)
- 5xx (서버 오류, 서버 측 잘못)

In [5]:

```
resp.status_code
```

Out[5]:

```
200
```

이제 헤더를 확인해보자. 요청 헤더와 응답 헤더를 확인해보면 다음과 같다.

In [6]:

```
resp.request.headers
```

Out[6]:

```
{'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/74.0.3729.157 Safari/537.36', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*',
'Connection': 'keep-alive'}
```

위의 결과값을 확인해보면 요청하기 전에 `User-Agent`를 지정해주었는데, 지정해준 `User-Agent`가 잘 적용되었음을 확인할 수 있다.

In [7]:

```
resp.headers
```

Out[7]:

```
{'Content-Type': 'text/html; charset=UTF-8', 'Date': 'Wed, 22 May 2019 12:11:35 GMT', 'Expires': '
-1', 'Cache-Control': 'private, max-age=0', 'Strict-Transport-Security': 'max-age=31536000',
'P3P': 'CP="This is not a P3P policy! See g.co/p3phelp for more info."', 'Content-Encoding': 'gzip
', 'Server': 'gws', 'X-XSS-Protection': '0', 'X-Frame-Options': 'SAMEORIGIN', 'Set-Cookie': '1P_JA
R=2019-05-22-12; expires=Fri, 21-Jun-2019 12:11:35 GMT; path=/; domain=.google.com, CGIC=IgMqLyo;
expires=Mon, 18-Nov-2019 12:11:35 GMT; path=/complete/search; domain=.google.com; HttpOnly, CGIC=I
gMqLyo; expires=Mon, 18-Nov-2019 12:11:35 GMT; path=/search; domain=.google.com; HttpOnly, NID=184
=IzrIRWE9Uq089N9e9YgFyeaRC7OfnmHYD-
KyfyVXSFqzM_h4ELT_N3MfjQ31i03ESPOTg3gYB_nHFafLM43uYFB83TQVe18HIQYFiWaa2urv57_ixC5_lirbxqiFjnY22ZD5I
po0_IXBBuM47UHBxRbzLyQnUUNX4dJh0SI; expires=Thu, 21-Nov-2019 12:11:35 GMT; path=/;
domain=.google.com; HttpOnly', 'Alt-Svc': 'quic=":443"; ma=2592000; v="46,44,43,39"', 'Transfer-
Encoding': 'chunked'}
```

위의 결과값을 확인해보면 응답 헤더는 콘텐츠 타입(Content-Type)을 `text/html` 즉 html 문서의 콘텐츠로, `Date` 부분에는 응답 받은 날짜가 나와있다.

실제 요청을 보낸 URL을 확인해보면 다음과 같다.

In [8]:

```
resp.request.url
```

Out[8]:

```
'https://www.google.com/search?q=%EB%B0%95%EB%B3%B4%EC%98%81&ie=UTF-8'
```

`http`는 `tcp-ip` 방식으로 통신하기에 데이터를 `byte`로 변환시켜야한다. 따라서 원래대로라면 `resp.content.decode()`를 출력한 결과처럼 나와야하지만, 요청을 보낼 때 UTF-8 형식으로 요청을 보냈기 때문에 다음과 같은 출력값을 얻을 수 있다.

In [9]:

```
# resp.encoding = 'utf-8'
```

응답받은 html 문서를 300자만 출력해본다.

In [10]:

```
resp.text[:300]
```

Out[10]:

```
'<!doctype html><html itemscope="" itemtype="http://schema.org/SearchResultsPage" lang="ko"><head>
<meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"><met
a content="origin" name="referrer"><title>박보영 - Google 검색</title><script
nonce="GsawFzxlZXIMg8ivdm8KRg==">'
```

DOM

`DOM`이란, Document Object Model의 약어이며 HTML, XML 문서의 프로그래밍 interface이다. DOM은 문서의 구조화된 표현을 제공하며 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공하여 그들이 문서 구조, 스타일, 내용 등을 변경할 수 있게 돕는다.

DOM 트리 구조를 만들기 위해서 `bs4` 라이브러리에 있는 `BeautifulSoup`을 사용한다. `bs4` 모듈이 없을 경우, `pip`를 통해 설치한다.

In [11]:

```
# !pip install bs4
```

In [12]:

```
from bs4 import BeautifulSoup
```

Python에서 XML parser로서 주로 이용되는 parser는 `lxml`이다.

지원하는 Parser 종류

- html.parser: 빠르지만 유연하지 않기 때문에 단순한 HTML문서에 사용한다.
- lxml: 매우 빠르고 유연하다.
- xml: XML 파일에만 사용한다.
- html5lib: 복잡한 구조의 HTML에 대해서 사용한다.

In [13]:

```
dom = BeautifulSoup(resp.text, 'lxml')
```

CSS Selector

css에는 `id`, `class`, `attribute`가 있다.

- id: `#`을 붙여 표현
- class: `.`을 붙여 표현, 다중 상속 지원

- `태그명[id='아이디']`
- `[class^='g']`

ex)

아래와 같은 html 코드가 있다고 가정할 때,

```
<div id='cst' class='b g r'>
    ...
</div>
```

클래스에서 상속 관계를 표현할 때 `b g r`과 같이 스페이스로 구분짓는다. 위의 html 코드에서 `div` 태그를 가져오고 싶을 때, `css selector`를 사용하여 다음과 같이 가져올 수 있다.

```
dom.select('div#cst') # div 태그의 cst 아이디를 가진 태그
dom.select('#cst') # 태그명을 생략할 경우, cst 아이디를 가지는 모든 태그를 가져온다.
dom.select('div.b.g.r') # div 태그의 b g r 클래스를 가져온다.
dom.select('.b.g.r') # 태그명을 생략할 경우, b g r 클래스를 가지는 모든 태그를 가져온다.
```

In [14]:

```
titles = dom.select('.LC20lb')
```

In [15]:

```
len(titles)
```

Out[15]:

```
8
```

In [16]:

```
titles
```

Out[16]:

```
[<h3 class="LC20lb">박보영 - 위키백과, 우리 모두의 백과사전</h3>,
 <h3 class="LC20lb">박보영의 작품 목록 - 위키백과, 우리 모두의 백과사전</h3>,
 <h3 class="LC20lb">박보영 - 나무위키</h3>,
 <h3 class="LC20lb">김사랑이 죽었는데 박보영의 몸으로 부활하는 드라마 - YouTube</h3>,
 <h3 class="LC20lb">박보영, tvN 드라마 '어비스' 여주인공 - MSN.com</h3>,
 <h3 class="LC20lb">#박보영 hashtag on Twitter</h3>,
 <h3 class="LC20lb">박보영은 오래 지켜본다. 연애도, 연기 변신도 - 중앙일보 - 조인스</h3>,
 <h3 class="LC20lb">[종합] '어비스' 박보영X안효섭, 살인 사건 공범 존재 알았다 '권수현 ...</h3>]
```

In [17]:

```
dom.select('.kpshf.qs-css-a.line.gsr.bilit')
```

Out[17]:

```
[<span class="kpshf qs-css-a line gsr bilit" style="display:none"></span>]
```

아래는 css selector를 사용하여 링크를 포함하는 `a 태그`의 링크부분과 그에 대한 내용(제목)을 뽑아 출력하는 코드이다.

In [18]:

```
for tag in dom.select('.r > a'):
    print(tag['href'], tag.select_one('h3').text)
```

```
https://ko.wikipedia.org/wiki/%EB%B0%95%EB%B3%B4%EC%98%81 박보영 - 위키백과, 우리 모두의 백과사전
https://ko.wikipedia.org/wiki/%EB%B0%95%EB%B3%B4%EC%98%81%EC%9D%98_%EC%9E%91%ED%92%88_%EB%AA%A9%EB%D 박보영의 작품 목록 - 위키백과, 우리 모두의 백과사전
https://namu.wiki/w/%EB%B0%95%EB%B3%B4%EC%98%81 박보영 - 나무위키
```

```
https://www.youtube.com/watch?v=KK0dddmzhHA 김사랑이 죽었는데 박보영의 몸으로 부활하는 드라마 - YouTube
https://www.msn.com/ko-kr/entertainment/news/%EB%B0%95%EB%B3%B4%EC%98%81-tvn-%EB%93%9C%EB%9D%BC%EB%A7%88-%EC%96%B4%EB%B9%84%EC%8A%A4-%EC%97%AC%EC%A3%BC%EC%9D%B8%EA%B3%B5/ar-BBLZzVh 박보영, tvN 드라마 '어비스' 여주인공 - MSN.com
https://twitter.com/hashtag/%EB%B0%95%EB%B3%B4%EC%98%81 #박보영 hashtag on Twitter
https://news.joins.com/article/22895953 박보영은 오래 지켜본다. 연애도, 연기 변신도 - 중앙일보 - 조인스
https://www.mk.co.kr/star/broadcasting-service/view/2019/05/335761/ [종합] '어비스' 박보영X안효섭, 살인사건 공범 존재 알았다 '권수현 ...
```

이미지 스크래핑

이제 위의 개념을 토대로 이미지를 스크래핑 해보고자 한다.

In [19]:

```
url = 'https://www.google.com/search'
```

In [20]:

```
params = {
    'q':'박보영',
    'source':'lnms',
    'tbm':'isch',
    'sa':'X',
    'ved':'0ahUKEwjCq6SrwK7iAhXLMt4KHSyICIoQ_AUIDigB',
    'biw':'1119',
    'bih':'969'
}
```

In [21]:

```
resp = requests.get(url, params=params, headers=headers)
```

In [22]:

```
dom = BeautifulSoup(resp.text, 'lxml')
```

In [23]:

```
img_link = [tag['data-src'] for tag in dom.select('img') if tag.has_attr('data-src')]
```

여기서는 `image`라는 폴더에 이미지를 다운받으려고 한다. 따라서 `image` 폴더를 생성하기 위한 코드를 작성했다.

In [24]:

```
import os
```

In [25]:

```
if not os.path.exists('image'):
    os.mkdir('image')
```

이제 수집한 이미지를 내려받는다.

In [26]:

```
for i, link in enumerate(img_link):
    resp = requests.get(link)
    ext = '.' + resp.headers['Content-Type'].split('/')[1]

    with open('./image/' + str(i) + ext, 'wb') as img:
        img.write(resp.content)
```

위의 코드를 설명하면 다음과 같다.

위의 코드를 설명하면 다음과 같다.

1. 먼저, css selector를 사용해서 `data-src` 속성을 가지는 `img` 태그에서 이미지 링크를 리스트로 `img_link`라는 식별자에 가져온다.
2. 그 다음 `img_link`를 순회하며 그 링크에 다시 GET 방식으로 요청을 보내게 되면 서버로부터 이미지 형식 콘텐츠를 바이트 코드 형태로 전달받을텐데, `resp.content`가 바로 그것이다.
3. 식별자 `ext`는 파일 확장자를 의미하며, 콘텐츠 타입에 `/`로 구분되어있기 때문에 `split`하여 마지막 부분을 확장자로써 사용하겠다는 부분이다.
4. 최종적으로, 파일 입출력을 통해서 이미지를 다운받는다.

실제로 잘 받아왔는지 이미지를 불러와서 확인해본다. 여기서는 `PIL` 라이브러리를 사용하여 불러왔다.

In [27]:

```
from PIL import Image
```

In [28]:

```
Image.open('./image/2.jpeg')
```

Out[28]:

Base64

Tip. 아래와 같이 `base64`로 인코딩 되어있는 부분은 `decoding` 해주면 된다.

In [29]:

```
dom.select('img')[2]['src']
```

Out[29]:

```
''
```

In [30]:

```
import base64
```

In [31]:

```
for tag in dom.select('img'):
    if tag.has_attr('src'):
        if 'base64' in tag['src']:
            code = tag['src'].split('base64,')[-1]
            code = base64.decodebytes(bytes(code, 'utf8'))
            print('디코딩 결과:', code)
```

```
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
```

```
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
디코딩 결과:
b'GIF89a\x01\x00\x01\x00\x80\x00\x00\xff\xff\xff\xff\xff\xff!\xf9\x04\x01\n\x00\x01\x00,\x00\x00\x0
0\x01\x00\x01\x00\x00\x02\x02L\x01\x00;'
```

Selenium

Web Driver를 통해 요청을 수행한다.

In [32]:

```
# !pip install --upgrade selenium
```

Web Driver를 사용하기 전에, 먼저 사용중인 브라우저의 드라이버를 다운받는다. 여기서는 크롬 브라우저를 사용하고 있으므로 크롬 드라이버를 다운받았다.

크롬 드라이버 다운로드: http://chromedriver.chromium.org/downloads

In [33]:

```
from selenium import webdriver
```

드라이버 클래스의 첫번째 인자값으로 다운받은 드라이버가 있는 경로를 넣어준다.

In [34]:

```
driver = webdriver.Chrome('driver/chromedriver')
```

코드를 실행하면 드라이버를 로딩하여 새로운 브라우저가 열리게 된다. 그럼 아래와 같이 GET 방식으로 네이버에 요청을 보낸다.

In [35]:

```
driver.get('https://www.naver.com')
```

아래의 코드를 실행하면 네이버 로그인 페이지로 이동한다. 아래의 코드는 앞서 설명한대로 css selector로 버튼 태그를 찾은 후 그것을 click 이벤트를 수행하는 코드이다.

In [36]:

```
driver.find_element_by_class_name('ico_local_login').click()
```

Cookie

로그인 페이지에서 로그인을 시도한 후 쿠키를 확인해본다.

In [37]:

```
driver.get_cookies()
```

Out[37]:

```
[{'domain': '.naver.com',
  'expiry': 2524640400.578402,
  'httpOnly': False,
  'name': 'NNB',
  'path': '/',
  'secure': False,
  'value': 'CU3DUOMNHTSVY'},
 {'domain': '.nid.naver.com',
  'expiry': 1590063117,
  'httpOnly': False,
  'name': 'nid_slevel',
  'path': '/',
  'secure': False,
  'value': '1'},
 {'domain': '.nid.naver.com',
  'expiry': 1873887117,
  'httpOnly': False,
  'name': 'nid_buk',
  'path': '/',
  'secure': True,
  'value': 'CU3DUOMNHTSVY'}]
```

현재 쿠키는 name과 value 쌍으로 구성되어있다.

In [38]:

```
for cookie in driver.get_cookies():
    print(cookie['name'])
    print(cookie['value'])
```

```
NNB
CU3DUOMNHTSVY
```

```
nid_slevel
1
nid_buk
CU3DUOMNHTSVY
```

Session

로그인 정보를 세션으로 등록하여 GET 방식으로 보내보고자 한다.

In [39]:

```
from requests import Session
```

In [40]:

```
session = Session()
```

In [41]:

```
for cookie in driver.get_cookies():
    print(cookie['name'])
    print(cookie['value'])
    session.cookies.set(cookie['name'], cookie['value'])
```

```
NNB
CU3DUOMNHTSVY
nid_slevel
1
nid_buk
CU3DUOMNHTSVY
```

세션으로 등록된 쿠키들의 목록들을 확인해보면 아래와 같다.

In [42]:

```
session.cookies
```

Out[42]:

```
<RequestsCookieJar[Cookie(version=0, name='NNB', value='CU3DUOMNHTSVY', port=None,
port_specified=False, domain='', domain_specified=False, domain_initial_dot=False, path='/',
path_specified=True, secure=False, expires=None, discard=True, comment=None, comment_url=None,
rest={'HttpOnly': None}, rfc2109=False), Cookie(version=0, name='nid_buk', value='CU3DUOMNHTSVY',
port=None, port_specified=False, domain='', domain_specified=False, domain_initial_dot=False,
path='/', path_specified=True, secure=False, expires=None, discard=True, comment=None,
comment_url=None, rest={'HttpOnly': None}, rfc2109=False), Cookie(version=0, name='nid_slevel', va
lue='1', port=None, port_specified=False, domain='', domain_specified=False,
domain_initial_dot=False, path='/', path_specified=True, secure=False, expires=None, discard=True,
comment=None, comment_url=None, rest={'HttpOnly': None}, rfc2109=False)]>
```

GET 방식으로 요청하여 세션을 등록한다. POST 방식으로 요청을 보낼 때는 `session.post` 를 사용해주면 된다.

In [43]:

```
resp = session.get('https://www.naver.com')
```

네이버 메일 페이지로 이동하게 되면 현재 로그인 되어있는 계정의 이메일 페이지로 이동하는 것을 확인할 수 있다.

In [44]:

```
driver.get("https://mail.naver.com/")
```