Python 프로그래밍
#5. 웹 크롤링과 데이터 분석

목표

- 사이트 구조를 파악할 수 있다
- 사이트에서 크롤링을 하여 **원하는 데이터를 가져올 수 있다**
  - 동적인 사이트
  - 정적인 사이트

웹의 구조

- 브라우저 <-> 인터넷 <-> 서버
- request를 url으로부터 시작하고,
- response로 html, css, js등등의 값을 받는다
- HTTP(Hypertext Transfer Protocol)라는 규약으로 통신한다

데이터 수집의 유형

1. 정적인 사이트 (네이버)
2. 동적인 사이트 (직방, 왓챠, ...)
3. 한국형 사이트: IFrame
4. 한국형 사이트: javascript

지피지기 백전백승(知彼知己 百戰百勝)! 크롤링할 사이트와 내가 가져올 데이터에 대한 이해가 있다면 어떠한 사이트의 정보도 가져올 수 있다

HTML (HyperText Markup Language)

- 웹페이지를 작성하는데 쓰는 마크업 언어
- 팀 버너스리가 유럽 입자 물리 연구소 (CERN)에서 문서 공유를 위해 만듦
- 여러 태그들로 구성되어 있고, 각 태그를 이용해서 문서를 만들어 간다
- Hypertext: 한 문서에서 다른 문서로의 이동할 수 있는 링크

HTML (HyperText Markup Language)

```
<html>                                  -- html의 시작
  <body>                                -- body의 시작
    <a> 안녕하세요 </a>                  -- a 태그
    <div> ... </div>                    -- div 태그
  </body>                               -- body의 끝
</html>                                 -- html의 끝
```

HTML 태그

- html: 문서의 시작과 끝을 알리는 태그
- body: 본문의 시작과 끝을 알리는 태그
- h: header 제목을 표시할 때 사용하는 태그 h1 ~ h6
- p: paragraph 새 문단을 표시할 때 사용하는 태그
- ul: unordered list 순서 없는 리스트
- li: list item 순서 있는 리스트
- a: 하이퍼링크를 생성하는 태그 href 속성에 링크가 담긴다

HTML Attribute

- 태그마다 속성(attribute)를 다음과 같이 넣을 수 있다.

```
<div id="my_id"></div>
<div class="my_class"></div>
<a href="http://naver.com"> 네이버 링크 </a>
<img src="http://img.url"> 이미지 표시 </img>
```

CSS (Cascading Style Sheets)

- html이 웹페이지 문서의 구조라면, css는 html로 만들어진 문서의 텍스트 크기나 색상, 이미지 위치, 표 색상, 배치방법등 문서의 디자인 요소를 담당하는 언어
- 태그안의 속성에 clsss, id 같은 것들로 꾸밀 수 있다
  -

HTML with CSS

```
<p class="good_class"> good_class 이름의 css 클래스 </p>
<a id="good_id"> good_id 이름의 css ID </a>
```

- css문서

```
.good_class { ... }
#good_id { ... }
```

CSS Selector

- CSS 선택자. 스타일을 꾸미고 싶은 특정 요소들을 선택할때 사용한다.
- CSS Selector로 우리가 원하는 값들을 뽑아 낼 때 사용할 예정.
- 종류:
  - class: .으로 표현 예) div.class
  - id: #으로 표현 예) div#id
  - 요소 안의 하위 요소: 띄어쓰기로 표현 예) div div
  - 하위 요소: >로 표현 예) div > p

A. 정적인 사이트 크롤링

- 서버에서 렌더링(rendering)을 해서 주는 웹 사이트

예) 네이버, 다음, 네이버 모바일 페이지 등등..

A. 정적인 사이트 크롤링

- requests 모듈 (HTTP for Human)
- 웹 데이터를 가져올 수 있다
- requests.get(url)로 요청
- .status_code
- .text

requests

- HTTP 요청을 쉽게 할 수 있는 인간친화적 라이브러리 (HTTP for Humans)

```
import requests

# 네이버로 GET request
response = requests.get('http://www.naver.com')

# HTTP response code 확인하기
print(response.status_code)

# 요청으로 받은 response 내용 보기
print(response.text)
```

HTTP Response code

- 200: 성공 (ok)
- 400: 잘못된 요청 (Bad Request)
- 403: 금지됨 (Forbidden)
- 404: 찾을 수 없음 (Not Found)
- 500: 서버 오류 (Internal Server Error)
- 503: 서비스를 사용할 수 없음 (Service Unavailable)

BeautifulSoup

- BeautifulSoup 받아온 html문서를 파싱할 때 사용한다.

```
from bs4 import BeautifulSoup

# 데이터 파싱 준비. 첫번째 인자에는 데이터, 두번째 인자는 'html.parser'가 들어간다
dom = BeautifulSoup("data", "html.parser")
```

BeautifulSoup

- css selector를 통해서 하위 요소들을 찾아감
- dom.select_one("{css_selector}")
  - 제일 첫번째 매칭되는 것만 반환
- dom.select('{css_selector}')
  - css selector에 매칭되는 것들을 모두 반환
- element에 대해서는 다음과 같이 적용 가능
  - element.attr.get(""): html의 속성 접근하기
  - element.text: 텍스트 값 출력하기

네이버 블로그 크롤링하기

- 네이버 검색 주소
  - https://search.naver.com/search.naver?where=post&query=파이썬
- url에서 query부분이 검색어를 뜻하고, where부분이 블로그 검색을 할 것인지 나타낸다

네이버 블로그 크롤링하기

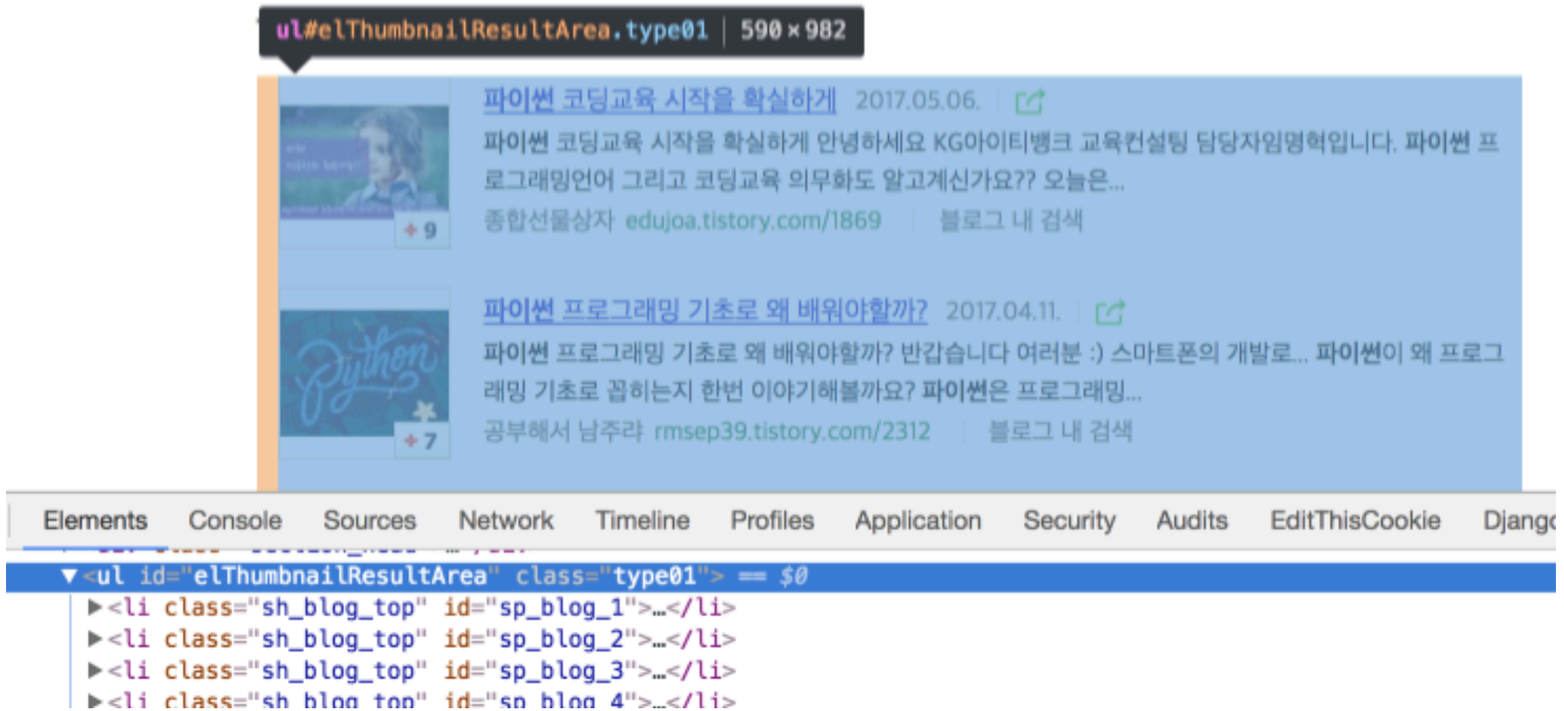
```
import requests
url = "https://search.naver.com/search.naver?where=post&query=파이썬"

# url로 웹 데이터 가져오기
response = requests.get(url)

# 가져온 데이터 출력
print(response.text)
```

네이버 블로그 파싱하기

- 엘리먼트들이 들어있는 `ul`의 `id`가 `elThumbnailResultArea`



네이버 블로그 파싱하기

```
from bs4 import BeautifulSoup

dom = BeautifulSoup(response.text, 'html.parser')
elements = dom.select("ul#elThumbnailResultArea li")

# 첫번째 검색 결과 아이템의 title
elements[0].select_one("a.sh_blog_title").attrs.get("title")

# 첫번째 검색 결과 아이템의 url
elements[0].select_one("a.sh_blog_title").attrs.get("href")
```

실습

- 네이버 실시간 검색어를 크롤링 해보세요.
- 크롤링한 결과를 CSV파일로 만들어 보세요

JSON

- Javascript Objects Notation
- 서로다른 언어들과 데이터 통신을 위해서 사용하는 형식
- 딕셔너리랑 비슷하게 key와 value값이 들어가게 된다
- {"name":"임원균","email":"corikachu@gmail.com"}

JSON 예시

```
{
  "id": "123",
  "student": [
    {
      "Name": "임원균",
      "Phonenumber": "01012341234"
    },
    {
      "Name": "홍길동",
      "Phonenumber": "01012341234"
    },
  ],
}
```

JSON

```
import json
data = {"name":"임원균","email":"corikachu@gmail.com"}

# dict -> str로 변환
json_string = json.dumps(data)

# json str -> dict로 변환
json.loads(json_string)
```

실습

- 네이버 중고나라 매물 데이터를 크롤링 해보세요

B. 동적인 사이트 크롤링

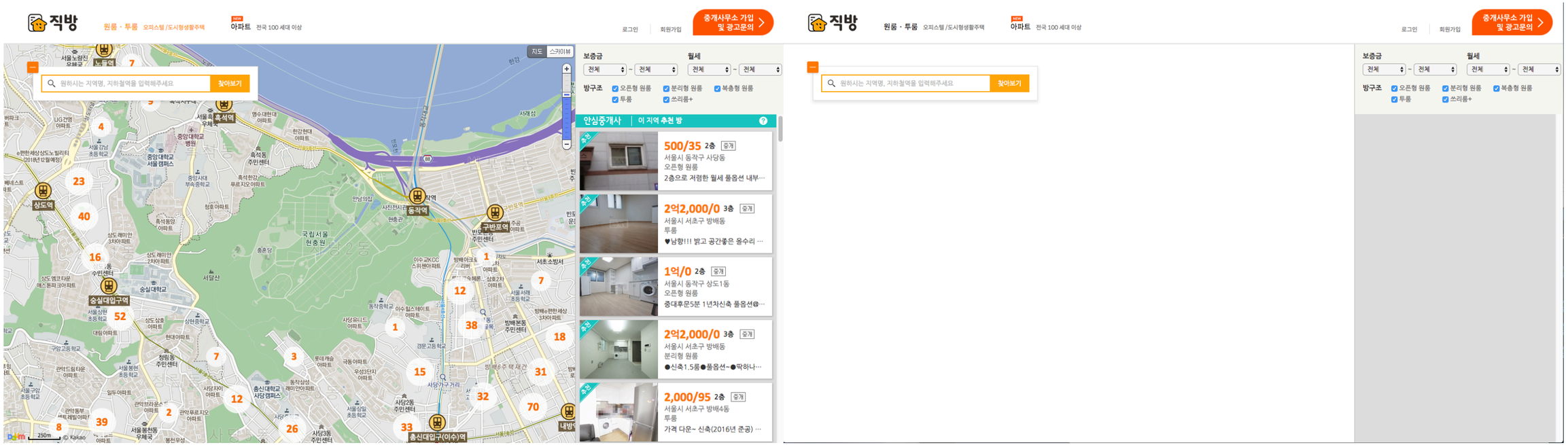
- 서버에서 랜더링 해주는 것이 아닌 브라우저에서 자바스크립트 등을 통해서 데이터를 추가적으로 가져온 사이트들은 크롤링 방식이 조금 다르다
- 서버에서 처음 html을 받을때에는 데이터가 채워지지 않은채로 온다
- 때문에 사이트에서 자바스크립트가 호출하는 API를 통해 데이터를 어디서 받아오는지를 알아내고,
- 그 주소로부터 데이터를 가져와서 파싱해야한다.

B. 동적인 사이트 크롤링

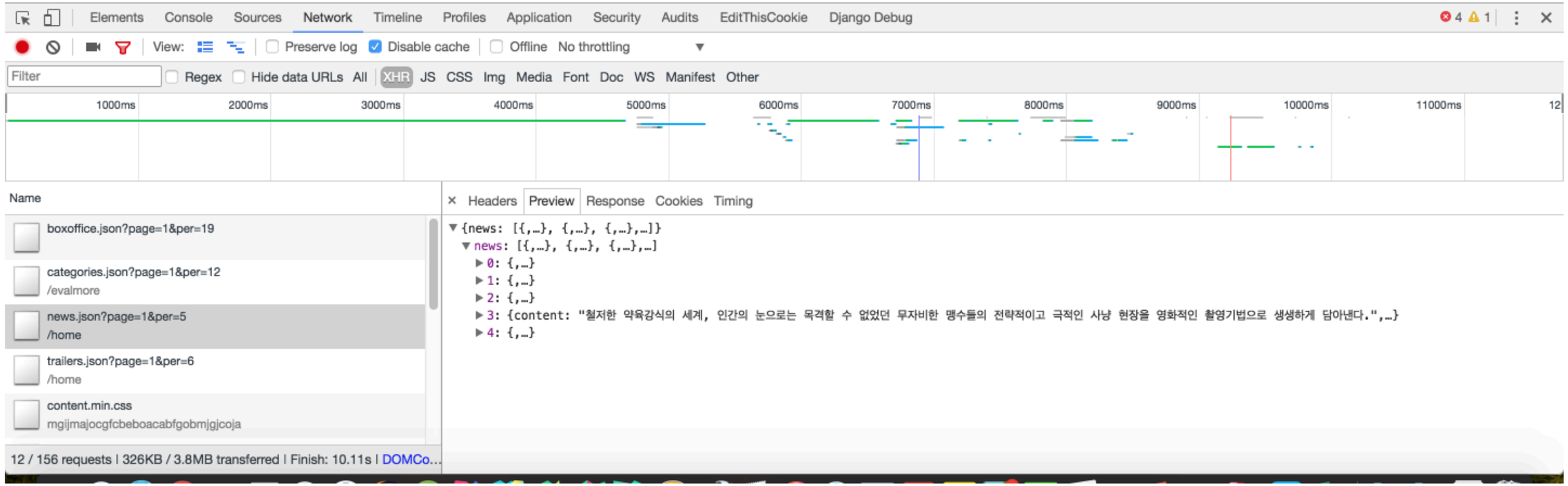
- 사이트에서 자바스크립트가 부르는 API를 통해 데이터를 어디서 받아오는지 알아낸다
- 그 주소로부터 데이터를 가져와서 파싱한다

직방사이트

- 왼쪽은 자바스크립트를 허용했을때, 왼쪽은 자바스크립트를 사용금지 했을때
- 직방은 데이터를 자바스크립트를 통해서 추가적으로 로딩한다



브라우저 개발자 도구의 network탭 활용



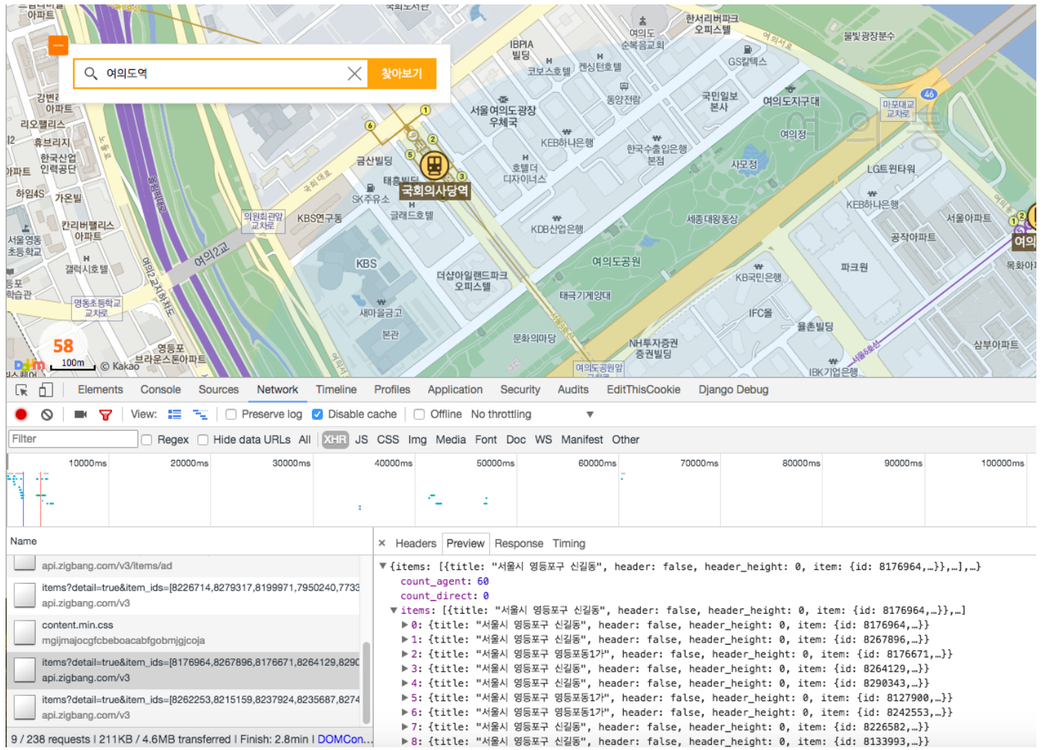
- 홈페이지에서 요청하는 데이터를 확인 할 수 있음

B. 동적인 사이트 크롤링

- 주소를 통해서 requests 라이브러리로 가져오고,
- BeautifulSoup 라이브러리를 통해서 필요한 데이터를 파싱한다
- **주소만 가져올 수 있다면 정적인 사이트 파싱과 같다**
- 대신 주소를 잘 찾아내야 한다

직방 크롤링하기

데이터를 어디서 요청하는지 개발자도구를 통해서 살펴본다



직방 크롤링하기

```
url = 'https://api.zigbang.com/v3/items?detail=true&item_ids=[8176964,8267896]'
# url을 통해서 데이터를 웹에서 가져오고
response = requests.get(url)

# json 형태의 정보를 dict으로 바꾼다
# data = json.loads(response.text)
data = response.json()
```

직방 파싱하기

```
# items의 리스트로부터 각각의 아이템을 파싱한다
for item in data.get("items"):
    detail_item = item.get("item")
    rent = detail_item.get("rent")
    deposit = deatil_item.get("deposit")
    address = detail_item.get("address")
    print((rent, deposit, address))
```

실습

- 왓챠플레이의 새로나온 신작 뉴스 크롤링하기