web_scraping

December 5, 2018

1 Python for Data Science

Getting Web Data: Web Scraping and Rest APIs

- Reading large file collections
- Getting tabular data from web pages with `pandas`
- Scraping data with
- BeautifulSoup
- Scrapy
- Data from REST APIs
- Building your own REST API with flask

2 Reading Large File Collections

- Data from large files (or collections thereof) often does not fit in memory
- But computations on such data can be performed in a streaming fashion
- For complex computations libraries that support streaming are useful (e.g. `pandas.read_csv` with chunksize arg)
- Plain python also supports lazy evaluations and computations with a small memory footprint
- The general principle is the same, for plain python or libraries

```
In [9]: import glob
        import os
        import itertools

        # finds all csv files in data/random_numbers
        files = glob.glob(os.path.join("data",'random_numbers','*.csv'))

        # return an iterator of lines over all files
        lines = itertools.chain(*map(open,files))

        def process_line(line):
            return [int(c) * 2 for c in line.split(",")]

        # this could be done without a list comprehension (which loads the data in memory),
```

```
        # e.g. to store computations immediately in another file
        numbers = [process_line(l) for l in lines]
        numbers[:5]
```

```
Out[9]: [[0], [98], [172], [22], [162]]
```

3 Getting tabular data from web pages with `pandas`

- Birth statistics from Berlin data portal
- https://en.wikipedia.org/wiki/Berlin_population_statistics

3.1 Birth statistics of Berlin

Birth statistics can be obtained through the Berlin data portal
E.g. https://www.berlin.de/daten/liste-der-vornamen-2014/charlottenburg-wilmersdorf.csv

```
vorname;anzahl;geschlecht
Marie;118;w
Sophie;92;w
Charlotte;76;w
Maria;73;w
Maximilian;66;m
Alexander;53;m
Emilia;52;w
```

```
In [3]: import pandas as pd
        import urllib
        import os
        %matplotlib inline
        import matplotlib.pyplot as plt
        import warnings
        warnings.filterwarnings("ignore", message="numpy.dtype size changed")
```

```
In [4]: basedir = os.path.join("data","vornamen")
        os.makedirs(basedir, exist_ok=True)

        base_url = "https://www.berlin.de/daten/liste-der-vornamen-{}/{}.csv"

        boroughs = [
        "charlottenburg-wilmersdorf",
        "friedrichshain-kreuzberg",
        "lichtenberg",
        "marzahn-hellersdorf",
        "mitte",
        "neukoelln",
        "pankow",
        "reinickendorf",
```

```
        "spandau",
        "steglitz-zehlendorf",
        "tempelhof-schoeneberg",
        "treptow-koepenick"
        ]

        years = range(2013,2018)
```

```
In [6]: import pandas as pd
        # download all name files from Berlin open data portal
        all_names = []

        for borough in boroughs:
            for year in years:
                try:
                    url = base_url.format(year, borough)
                    filename = os.path.join(basedir, "{}-{}.csv".format(year,borough))
                    urllib.request.urlretrieve(url, filename)
                    df_vornamen_stadtteil = pd.read_csv(filename,sep=';',error_bad_lines=False)
                    df_vornamen_stadtteil['borough'] = borough
                    df_vornamen_stadtteil['year'] = year
                    all_names.append(df_vornamen_stadtteil)
                except:
                    print("File {} not found".format(url))

        # concatenate DataFrames
        all_names_df = pd.concat(all_names, sort=True)
```

```
In [8]: all_names_df.sample(n=10)
```

Out[8]:

	anzahl	borough	geschlecht	position	vorname	year
3723	1	charlottenburg-wilmersdorf	m	1.0	Loi-Loan	2017
1062	1	spandau	m	NaN	Curtis	2014
1718	1	spandau	w	2.0	Paola	2017
281	6	pankow	m	NaN	Luka	2016
3439	1	mitte	w	NaN	Shahed	2016
3187	1	friedrichshain-kreuzberg	m	NaN	Émile	2015
2459	1	mitte	w	NaN	Medine	2013
561	3	friedrichshain-kreuzberg	m	NaN	Pierre	2014
449	1	marzahn-hellersdorf	w	NaN	Cleo	2015
2429	1	friedrichshain-kreuzberg	w	NaN	Madita	2016

4 Getting tabular data from Wikipedia

Let's look at some population data from Wikipedia

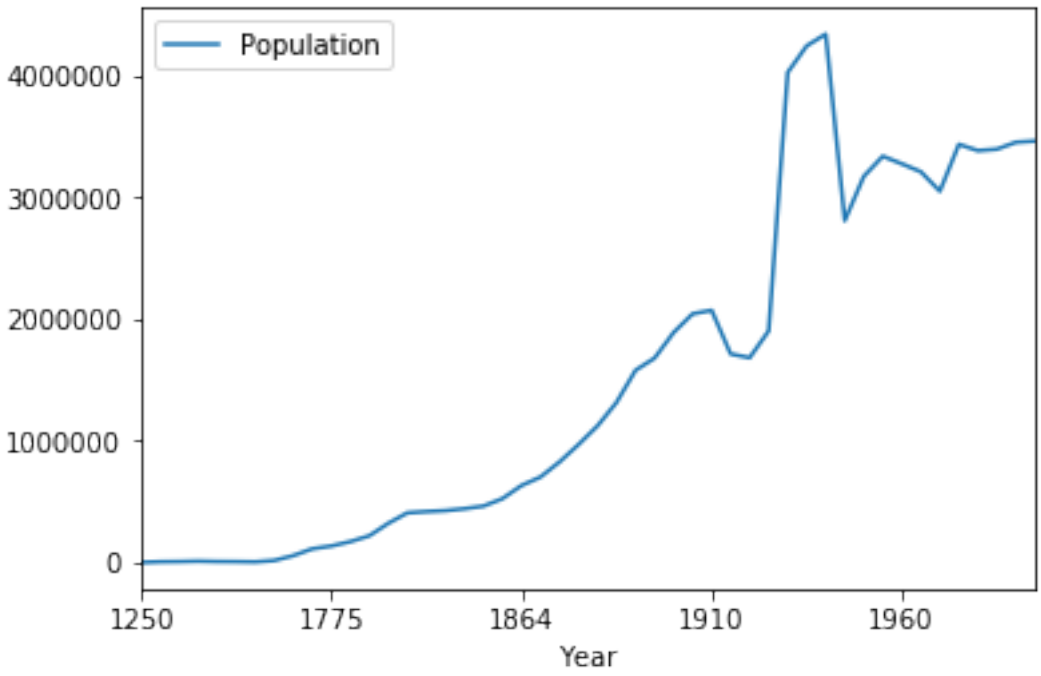
```
In [26]: berlin_population = pd.read_html(
             "https://en.wikipedia.org/wiki/Berlin_population_statistics", header=0)
```

```
df_berlin_population = berlin_population[0][:11].set_index('Borough')
df_berlin_population
```

Out[26]:

Borough	Population 30 September 2010	Area in kmš	Largest Non-German ethnic groups
Mitte	332100	39.47	Turks, Arabs, Kurds, many Asians, Africans and...
Friedrichshain-Kreuzberg	268831	20.16	Turks, Arabs, African, Kurds, Chinese
Pankow	368956	103.01	Poles, Italians, French, Americans, Vietnamese...
Charlottenburg-Wilmersdorf	320014	64.72	Turks, Africans, Russians, Arabs, others.
Spandau	225420	91.91	Turks, Africans, Russians, Arabs, others.
Steglitz-Zehlendorf	293989	102.50	Poles, Turks, Croats, Serbs, Koreans
Tempelhof-Schöneberg	335060	53.09	Turks, Croats, Serbs, Koreans, Africans
Neukölln	310283	44.93	Arabs, Turks, Kurds, Russians, Africans, Poles
Treptow-Köpenick	241335	168.42	Russians, Poles, Ukrainians, Vietnamese
Marzahn-Hellersdorf	248264	61.74	Russians, Vietnamese, several other Eastern Eu...
Lichtenberg	259881	52.29	Vietnamese, Russians, Ukrainians, Poles, Chinese

In [40]:
```
%matplotlib inline
# concatenate all tables on population statistics
overall_population = pd.concat(berlin_population[2:5])
# extract the years
overall_population.Year = overall_population.Year.str.extract('(\d{4})', expand=False)
# set the index to the year column, so plotting is nicer
overall_population = overall_population.set_index("Year")
overall_population.plot();
```



5 Beautiful Soup for Web Scraping

- HTML pages are often not well structured
- Beautiful Soup
  - tidies up dirty HTML
  - allows for convenient parsing of HTML

5.1 A Simple Webpage

```
In [79]: a_simple_webpage = \
         """<html>
             <head>
             </head>
             <body>
                 <p>
                     A paragraph
                 </p>
                 <p id="second_paragraph">
                    Another paragraph with a <a href="https://de.wikipedia.org/wiki/Beuth_(Lokom
                 </p>
             </body>
         </html>"""
```

```
In [83]: from bs4 import BeautifulSoup
         soup = BeautifulSoup(a_simple_webpage, 'html.parser')

In [84]: list(soup.children)

Out[84]: [<html>
          <head>
          </head>
          <body>
          <p>
                      A paragraph
                  </p>
          <p id="second_paragraph">
                     Another paragraph with a <a href="https://de.wikipedia.org/wiki/Beuth_(Loko
          </p>
          </body>
          </html>]
```

5.2 DOM Elements

5.2.1 Extracting Texts

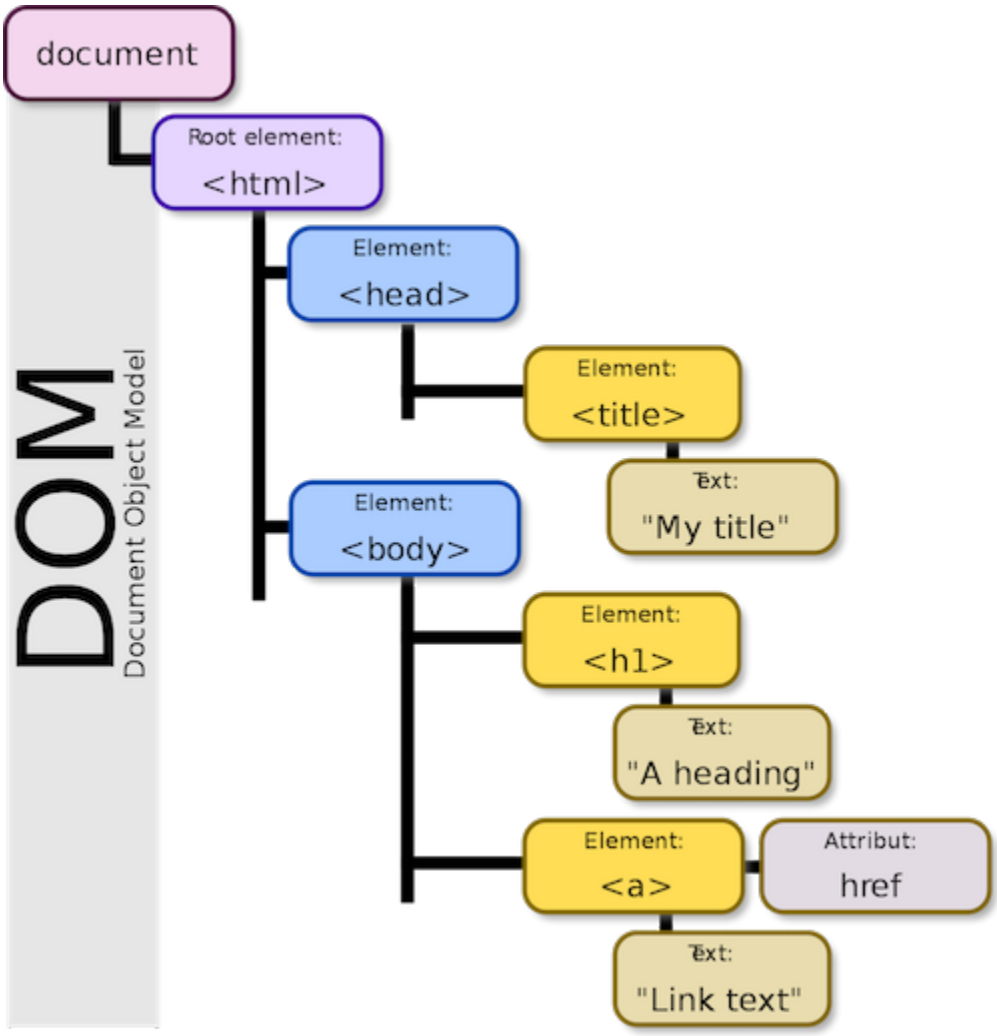
```
In [85]: paragraphs = soup.find_all('p')
         paragraphs

Out[85]: [<p>
                      A paragraph
                  </p>, <p id="second_paragraph">
                     Another paragraph with a <a href="https://de.wikipedia.org/wiki/Beuth_(Loko
          </p>]

In [86]: [p.get_text().strip() for p in paragraphs]

Out[86]: ['A paragraph', 'Another paragraph with a link']
```

5.2.2 Extracting Links

```
In [87]: links = soup.find_all('a')
         links

Out[87]: [<a href="https://de.wikipedia.org/wiki/Beuth_(Lokomotive)">link</a>]

In [88]: links[0].get('href')

Out[88]: 'https://de.wikipedia.org/wiki/Beuth_(Lokomotive)'
```

5.2.3 Extracting Arbitrary Elements

```
In [94]: [x.get_text().strip() for x in soup.find_all(id="second_paragraph")]

Out[94]: ['Another paragraph with a link']
```



HTML DOM Model

5.3 Example: Downloading Some Data from Wikipedia

Let's find out what was going on with Christian Peter Beuth and August Borsig, or: Why is a steam locomotive and your university named after Beuth?

```
In [95]: import requests # for downloading web pages

         url = "https://de.wikipedia.org/wiki/Beuth_(Lokomotive)"

         page = requests.get(url)
```

5.3.1 Excursion: HTML Status Codes

Code	Type	Meaning
1xx	Information	The request was received, continuing process
2xx	Successful	The request was successfully received, understood, and accepted
3xx	Redirection	Further action needs to be taken in order to complete the request
4xx	Client Error	The request contains bad syntax or cannot be fulfilled
5xx	Server Error	The server failed to fulfill an apparently valid request

```
In [97]: page.content[:500]
```

```
Out[97]: b'<!DOCTYPE html>\n<html class="client-nojs" lang="de" dir="ltr">\n<head>\n<meta charse
```

```
In [102]: # let beautiful soup parse the html
          soup = BeautifulSoup(page.content, 'html.parser')
          # find a paragraph in which both beuth (the persion, not the steam locomotive) and bor
          for p in soup.find_all('p'):
              if 'christian peter wilhelm beuth' in p.get_text().lower() \
                  and 'borsig' in p.get_text().lower():
                  print(p.get_text())
```

```
Die von August Borsig 1844 konstruierte Lokomotive BEUTH mit Werknummer 24 gilt als die erste ei
```

6 Crawling Data from Web Pages with `scrapy`

- Beautiful Soup is great for extracting information from *single webpages*
- Often web sites have multiple pages
- Writing a custom 'spider' to crawl those websites can be tedious
- Dedicated libraries like `scrapy` help to scrape data from larger websites efficiently

6.1 Crawling Data First Steps

Find relevant Document Object Model (DOM) elements of website:

- Browse to website and right-click `Inspect` (Chrome) or `Inspect Element` (Firefox)
- Remember the class / id of elements you are interested in

6.2 Example: Finding a Flat in Berlin

Let's scrape flat data from immoscout

- Find and remember `class` of DOM element of each listing
- For each listing, find relevant flat attributes
  - price
  - size
  - location

Then we'll use that information to build our own spider for scrapy

6.3 Example: Finding a Flat in Berlin

Let's scrape flat data from immoscout

6.3.1 Some Imports and Relevant DOM Elements

```
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors.lxmlhtml import LxmlLinkExtractor
from scrapy.selector import Selector
from scrapy.item import Item, Field
```

```
from scrapy.http.request import Request

listings_class = "result-list-entry__brand-title-container"
allowed_domain = "immobilienscout24.de"
start_url = "https://www.immobilienscout24.de/Suche/S-T/Wohnung-Miete/Berlin/Berlin/-/-/-/-/true

attributes = [
    'is24qa-etage',
    'is24qa-flaeche',
    'is24qa-zi',
    'is24qa-bezugsfrei-ab',
    'is24qa-kaltmiete',
    'is24qa-nebenkosten'
]
```

6.3.2 A Simple Data Model

```
class ISItem(Item):
    url = Field()

    # allgemein
    is24qa_etage = Field()
    is24qa_flaeche = Field()
    is24qa_zi = Field()
    is24qa_bezugsfrei_ab = Field()

    # kosten
    is24qa_kaltmiete = Field()
    is24qa_nebenkosten = Field()

    lat = Field()
    lng = Field()
    address = Field()
    zip_region_country = Field()
```

6.3.3 The `scrapy` Spider File

```
class DetailsPageSpider(CrawlSpider):
    name = "is24"
    allowed_domains = [allowed_domain]
    start_urls = [ start_url ]
    rules = (
        # Extract links for next pages
        Rule(LxmlLinkExtractor(
            allow=(),
            restrict_xpaths=(".//*[@id='pager']//div//a")),
            callback='parse_listings',
```

```
            follow=True
        ),
    )

    def parse_start_url(self, response):
        return self.parse_listings(response)

    def parse_listings(self, response):
        sel = Selector(response)
        print("listing page", response.url)
        links = sel.xpath(".//*[@class='{}']".format(listing_class))
        for link in links:
            print("="*100)
            link = "https://www.immobilienscout24.de" + link.xpath("@href").extract()[0]
            print(link)
            yield Request(link, callback=self.parse_details)
...
```

6.3.4 Parsing Single Listings

```
class DetailsPageSpider(CrawlSpider):
...
    def parse_details(self, response):

        sel = Selector(response)
        print("details", response.url)
        item = ISItem()
        item['url'] = response.url

        for attribute in attributes:
            try:
                item[attribute.replace("-", "_")] = sel.css('.{}::text'.format(attribute)).extra
            except Exception as e:
                item[attribute.replace("-", "_")] = None
```

6.3.5 Running the spider:

Type

```
scrapy runspider is24_spider.py -o mietwohnungen.csv -t csv -L WARN
```

in your command line

```
In [10]: import pandas as pd
         flats_df = pd.read_csv('data/mietwohnungen.csv')
         flats_df['is24qa_kaltmiete'] = flats_df['is24qa_kaltmiete'].str.replace('[.]','').str.r
         flats_df['is24qa_flaeche'] = flats_df['is24qa_flaeche'].str.replace(',','.').str.extrac
```

```
flats_df['rent_per_qm'] = flats_df['is24qa_kaltmiete'].astype(float) / flats_df['is24qa
flats_df.sort_values(by='rent_per_qm')[:10]
```

Out[10]:

	address	is24qa_bezugsfrei_ab	is24qa_etage
3321	Chemnitzer Straße 11,	ab sofort	4
694	Sandstraße 64a,	sofort	11
736	Sandstr. 64b,	sofort	8
2539	Wiclefstraße 42,	sofort	NaN
1285	Marzahner Chaussee 194,	sofort	4
2141	Maulbeerallee 49,	sofort	1 von 6
1491	Ribnitzer Str 19,	Nach Vereinbarung	10 von 11
2008	Erich-Kästner-Straße 19,	sofort	4 von 5
2675	Altenhofer Str. 40,	sofort	8 von 18
1254	Märkische Allee 280,	01.01.2019	18 von 21

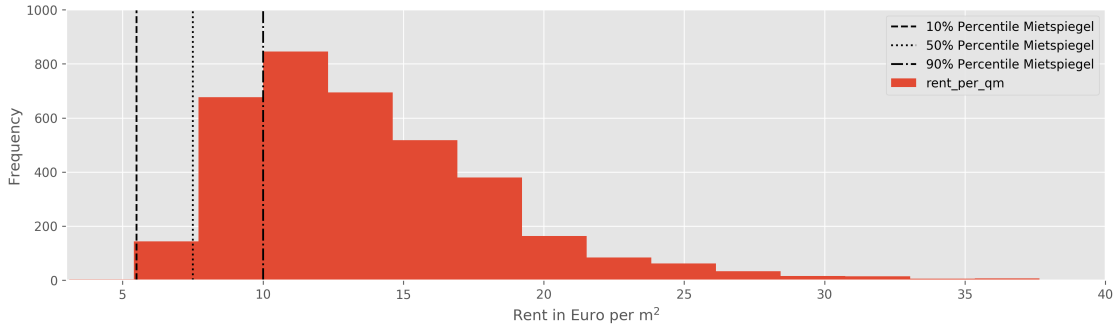
	is24qa_flaeche	is24qa_kaltmiete	is24qa_nebenkosten	is24qa_zi
3321	482.0	1493.70	NaN	3
694	60.0	312.94	NaN	2
736	58.0	303.39	NaN	2
2539	130.0	700.00	NaN	1
1285	58.0	318.42	NaN	2
2141	89.0	490.00	NaN	3
1491	70.0	389.78	NaN	3
2008	70.0	391.15	NaN	3
2675	61.0	345.00	NaN	2
1254	62.0	357.73	NaN	3

	lat	lng
3321	52.50431814189261,	13.581778
694	52.51939542525069,	13.171378
736	52.51958769930993,	13.171705
2539	52.530806644275785,	13.330809
1285	52.52430857597931,	13.534668
2141	52.523074993438286,	13.163599
1491	52.56937485558645,	13.496459
2008	52.527856075499464,	13.592260
2675	52.53585643100882,	13.484577
1254	52.55813472218006,	13.556239

	url
3321	https://www.immobilienscout24.de/expose/103502657
694	https://www.immobilienscout24.de/expose/108497983
736	https://www.immobilienscout24.de/expose/108602455
2539	https://www.immobilienscout24.de/expose/108097130
1285	https://www.immobilienscout24.de/expose/108665411
2141	https://www.immobilienscout24.de/expose/108379014
1491	https://www.immobilienscout24.de/expose/108624980
2008	https://www.immobilienscout24.de/expose/108441291

```
2675  https://www.immobilienscout24.de/expose/107934956
1254  https://www.immobilienscout24.de/expose/108669708

                                    zip_region_country  rent_per_qm
3321              12621 Berlin, Kaulsdorf (Hellersdorf)     3.098963
694                     13593 Berlin, Staaken (Spandau)     5.215667
736                     13593 Berlin, Staaken (Spandau)     5.230862
2539              10551 Berlin, Tiergarten (Tiergarten)     5.384615
1285                    12681 Berlin, Marzahn (Marzahn)     5.490000
2141                    13593 Berlin, Staaken (Spandau)     5.505618
1491  13051 Berlin, Neu-Hohenschönhausen (Hohenschön...     5.568286
2008              12619 Berlin, Kaulsdorf (Hellersdorf)     5.587857
2675  13055 Berlin, Alt-Hohenschönhausen (Hohenschön...     5.655738
1254                    12687 Berlin, Marzahn (Marzahn)     5.769839
```

In [76]:
```
plt.figure(figsize=[15,4],dpi=300)
plt.style.use('ggplot')
flats_df.rent_per_qm[flats_df.rent_per_qm<50].plot.hist(20);
plt.plot([5.5,5.5],[0,2000],'k--',label='10% Percentile Mietspiegel')
plt.plot([7.5,7.5],[0,2000],'k:',label='50% Percentile Mietspiegel')
plt.plot([10,10],[0,2000],'k-.',label='90% Percentile Mietspiegel')
plt.ylim([0,1000]);
plt.xlim([3,40]);
plt.legend()
plt.xlabel("Rent in Euro per m$^2$");
```



7 Getting Data From REST APIs

Representational State Transfer (REST) is a software architectural style for creating web services

RESTful web services allow to access and manipulate web resources through a uniform and predefined set of stateless operations

7.1 Example: Air Quality Data from AQICN

- Air quality is an important topic

- You can query recent worldwide air quality data through an API at http://aqicn.org/
- For example:
  - Air Quality Data Berlin
  - For accessing the API, get an access token
  - then you can query the data for Berlin by requesting (in a browser or in a programmatic fashion) `http://api.waqi.info/feed/berlin/?token=[accessToken]`
  - e.g. http://api.waqi.info/feed/berlin/?token=my_access_token

In [106]:
```
import requests

def get_air_quality_data(city, accesstoken='7672327f1d6675ef5d2d554b63b6175afec9fe77'):
    # build request
    url = 'http://api.waqi.info/feed/'+city+'/?token=' + accesstoken
    # get data
    r = requests.get(url, auth=('user', 'pass'))
    # check status code and return data
    if r.status_code == 200:
        data = r.json()
        return data['data']

get_air_quality_data('berlin')
```

Out[106]:
```
{'aqi': 40,
 'attributions': [{'name': 'Berlin Air Quality - (Luftqualität in Berlin)',
   'url': 'http://www.stadtentwicklung.berlin.de/umwelt/luftqualitaet/'},
  {'name': 'World Air Quality Index Project', 'url': 'https://waqi.info/'}],
 'city': {'geo': [52.5200066, 13.404954],
  'name': 'Berlin, Germany',
  'url': 'https://aqicn.org/city/germany/berlin'},
 'debug': {'sync': '2018-12-02T01:34:46+09:00'},
 'dominentpol': 'pm10',
 'iaqi': {'no2': {'v': 37.1},
  'o3': {'v': 2.1},
  'p': {'v': 996.9},
  'pm10': {'v': 40},
  't': {'v': 6.1}},
 'idx': 6132,
 'time': {'s': '2018-12-01 17:00:00', 'tz': '+01:00', 'v': 1543683600}}
```

In [108]:
```
get_air_quality_data('newyork')
```

Out[108]:
```
{'aqi': 23,
 'attributions': [{'name': 'New York State Department of Environmental Conservation (N
   'url': 'http://www.dec.ny.gov/'},
  {'name': 'Air Now - US EPA', 'url': 'http://www.airnow.gov/'},
  {'name': 'World Air Quality Index Project', 'url': 'https://waqi.info/'}],
 'city': {'geo': [40.7127837, -74.0059413],
  'name': 'New York',
```

```
 'url': 'https://aqicn.org/city/newyork'},
'debug': {'sync': '2018-12-02T01:21:43+09:00'},
'dominentpol': 'pm25',
'iaqi': {'co': {'v': 2.4},
 'h': {'v': 70.2},
 'no2': {'v': 10.7},
 'o3': {'v': 12.8},
 'p': {'v': 1023.3},
 'pm25': {'v': 23},
 't': {'v': 7.5}},
'idx': 3309,
'time': {'s': '2018-12-01 11:00:00', 'tz': '-05:00', 'v': 1543662000}}
```

```
In [110]: get_air_quality_data('seoul')
```

```
Out[110]: {'aqi': 112,
           'attributions': [{'name': 'South Air Korea Environment Corporation ()',
             'url': 'http://www.airkorea.or.kr/'},
            {'name': 'Seoul Clean Air Pollution Information ( )',
             'url': 'http://cleanair.seoul.go.kr/'},
            {'name': 'World Air Quality Index Project', 'url': 'https://waqi.info/'}],
           'city': {'geo': [37.566535, 126.9779692],
            'name': 'Seoul ()',
            'url': 'https://aqicn.org/city/seoul'},
           'debug': {'sync': '2018-12-02T01:25:10+09:00'},
           'dominentpol': 'pm25',
           'iaqi': {'co': {'v': 11.2},
            'h': {'v': 44},
            'no2': {'v': 64.2},
            'o3': {'v': 1.6},
            'p': {'v': 1028.2},
            'pm10': {'v': 61},
            'pm25': {'v': 112},
            'r': {'v': 9.6},
            'so2': {'v': 8.6},
            'w': {'v': 0.2},
            'wd': {'v': 112.5}},
           'idx': 5508,
           'time': {'s': '2018-12-02 01:00:00', 'tz': '+09:00', 'v': 1543712400}}
```

8 Building your own REST API with `flask`

- `flask` is a microframework for web development in python
- We will build a simple REST API with flask

```
In [ ]: !pip install newspaper3k
        !pip install Flask
```

```
In [2]: import newspaper, json
        newspapers = {
            'zeit': 'http://zeit.de',
            'tagesspiegel': 'https://www.tagesspiegel.de/'
        }

        def process_article(article):
            try:
                article.download()
                article.parse()
                return {
                        'title': article.title,
                        'url': article.url,
                }
            except:
                pass

        def process_newspaper(newspaper_url):
            articles = newspaper.build(newspaper_url).articles
            return [process_article(a) for a in articles]

        def download_and_save_news(save_path='news.json'):
            news = {n:process_newspaper(url) for n,url in newspapers.items()}
            json.dump(news, open(save_path, 'wt'))

        # download_and_save_news()

In [3]: news = json.load(open('news.json'))
        news

Out[3]: {'tagesspiegel': [{'title': 'IT-Systemadministrator (m/w)',
           'url': 'https://karriere.tagesspiegel.de/stellenangebot/2563936/IT_Systemadministrato
          {'title': 'Büromitarbeiter/-in',
           'url': 'https://karriere.tagesspiegel.de/stellenangebot/2539229/B%C3%BCromitarbeiter_
          {'title': 'Sekretärin/Sekretär',
           'url': 'https://karriere.tagesspiegel.de/stellenangebot/2539233/Sekret%C3%A4rin_Sekre
          {'title': 'Mitarbeiterin für Privatsekretariat',
           'url': 'https://karriere.tagesspiegel.de/stellenangebot/2572190/Mitarbeiterin_f%C3%BC
          {'title': 'Lehrkraft (w/m/d)',
           'url': 'https://karriere.tagesspiegel.de/stellenangebot/2558134/Lehrkraft_(w_m_d)_Sch
         'zeit': [{'title': 'Nachrichten, Hintergründe und Debatten',
           'url': 'http://zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-wirtscha
          {'title': 'Nachrichten, Hintergründe und Debatten',
           'url': 'http://zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-wirtscha
          {'title': 'Nachrichten, Hintergründe und Debatten',
           'url': 'http://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-wirt
          {'title': 'Nachrichten, Hintergründe und Debatten',
           'url': 'http://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-wirt
```

```
          {'title': 'Nachrichten, Hintergründe und Debatten',
           'url': 'https://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-wir
          {'title': 'Nachrichten, Hintergründe und Debatten',
           'url': 'https://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-wir
```

8.1 A Minimal Flask API

in `newsapi.py`:

```
from flask import Flask, jsonify
import json

app = Flask(__name__)

### API

@app.route('/newsapi/<newspaper_id>')
def get_news_by_newspaper(newspaper_id):
    # return the news of a newspaper
    return jsonify(news.get(newspaper_id, {}))

if __name__ == "__main__":
    port = 5001
    # load some previously downloaded news
    news = json.load(open('news.json'))
    app.run(host='0.0.0.0', port = port)
```

Now start the server by typing in the commandline

```
python newsapi.py
```

Opening a browser and navigating to `http://0.0.0.0:5001/newsapi/zeit` will yield

```
[{"title":"Nachrichten, Hintergr\u00fcnde und Debatten","url":"http://zeit.de/\n
https:/www.brandeins.de/magazine/brand-eins-wirtschaftsmagazin/2018/lebensmittel/was-waere-wenn-
"},{"title":"Nachrichten, Hintergr\u00fcnde und Debatten","url":"http://zeit.de/\n
https:/www.brandeins.de/magazine/brand-eins-wirtschaftsmagazin/2018/lebensmittel/food-start-ups-
"},{"title":"Nachrichten, Hintergr\u00fcnde und Debatten","url":"http://www.zeit.de/\n
https:/www.brandeins.de/magazine/brand-eins-wirtschaftsmagazin/2018/lebensmittel/was-waere-wenn-
"},{"title":"Nachrichten, Hintergr\u00fcnde und Debatten","url":"http://www.zeit.de/\n
https:/www.brandeins.de/magazine/brand-eins-wirtschaftsmagazin/2018/lebensmittel/food-start-ups-
"},{"title":"Nachrichten, Hintergr\u00fcnde und Debatten","url":"https://www.zeit.de/\n
https:/www.brandeins.de/magazine/brand-eins-wirtschaftsmagazin/2018/lebensmittel/was-waere-wenn-
"},{"title":"Nachrichten, Hintergr\u00fcnde und Debatten","url":"https://www.zeit.de/\n
https:/www.brandeins.de/magazine/brand-eins-wirtschaftsmagazin/2018/lebensmittel/food-start-ups-
"}]
```

9 Exercises

9.1 Assignment 01

Write a function `assignment_06_01` that reads the random numbers in the files with `csv` extension under `data/random_numbers`, sums up all values and returns the result. Try to avoid reading the entire file in memory and avoid using a library like pandas or numpy.

```
import glob
import os
import itertools


def assignment_06_01():
    # finds all csv files in data/random_numbers
    files = glob.glob(os.path.join("data",'random_numbers','*.csv'))
    #...
    return sum_of_values
```

9.2 Assignment 02

Write a function `assignment_06_02` that reads Wikipedia html pages and extracts the infobox key-value pairs as strings. The infobox is the blue table in the top right of wikipedia pages.

```
import bs4 as BeautifulSoup
import requests

def assignment_06_02(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.content, 'html.parser')
    infobox = {}

    #...

    return infobox
```

9.3 Assignment 03

Write a function `assignment_06_03` that reads the information about all Christmas markets in Berlin and returns the name of the district that has most registered Christmas markets.

```
import json
import requests

def assignment_06_03():
    christmas_market_url = "https://www.berlin.de/sen/web/service/" + \
    "maerkte-feste/weihnachtsmaerkte/index.php/index/all.json?q="
    data = json.loads(requests.get(christmas_market_url).content)
```

```
    #...

    return district_with_most_christmas_markets
```