web_scraping

November 11, 2020

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)
- Some legal aspects of scraping
- 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

[1]:
```
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]
```

[1]:
```
[[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
```

[2]:
```
import pandas as pd
import urllib
import os
%matplotlib inline
import matplotlib.pyplot as plt
```

[3]:
```
basedir = os.path.join("data","vornamen")
os.makedirs(basedir, exist_ok=True)

base_url = "https://raw.githubusercontent.com/berlinonline/
 ↪haeufige-vornamen-berlin/master/data/cleaned/{}/{}.csv"

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

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

years = range(2012,2019)
```

[4]:
```
# download all name files from Berlin open data portal
all_names = []

for borough in boroughs:
    for year in years:
        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)

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

[5]:
```
all_names_df.sample(n=10)
```

[5]:

	anzahl	borough	geschlecht	position	\
3392	1	mitte	m	NaN	
890	2	pankow	m	1.0	
247	4	neukoelln	w	NaN	
1298	1	lichtenberg	m	NaN	
2416	1	mitte	m	NaN	
2512	1	neukoelln	m	NaN	
746	2	neukoelln	m	NaN	
491	1	reinickendorf	w	NaN	
2416	1	charlottenburg-wilmersdorf	m	NaN	
3249	1	mitte	w	1.0	

	vorname	year
3392	Éric	2013
890	Gunnar	2017
247	Annabelle	2013
1298	Jannick	2015
2416	Laurenz	2014

```
2512                 Torsten  2013
746                 Nicholas  2016
491                   Elizan  2016
2416  Maximilian-Zhichengrui  2012
3249                 Faizher  2017
```

4 Getting tabular data from Wikipedia

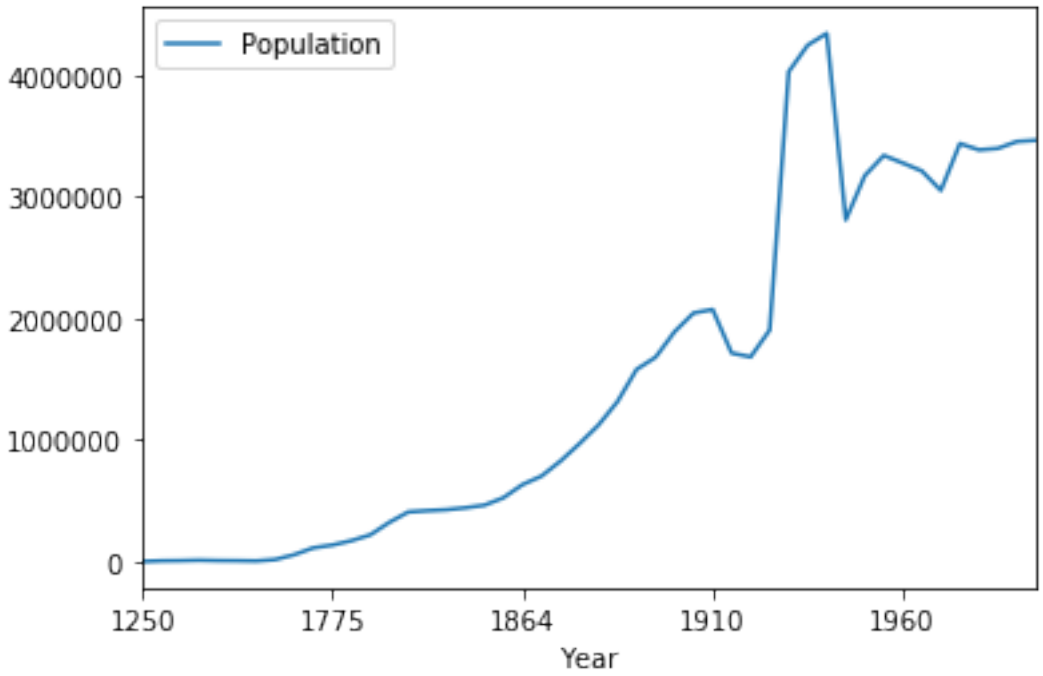
Let's look at some population data from Wikipedia

[6]:
```
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
```

[6]:
```
                          Population 30 September 2010  Area in km²  \
Borough
Mitte                                           332100        39.47
Friedrichshain-Kreuzberg                        268831        20.16
Pankow                                          368956       103.01
Charlottenburg-Wilmersdorf                      320014        64.72
Spandau                                         225420        91.91
Steglitz-Zehlendorf                             293989       102.50
Tempelhof-Schöneberg                            335060        53.09
Neukölln                                        310283        44.93
Treptow-Köpenick                                241335       168.42
Marzahn-Hellersdorf                             248264        61.74
Lichtenberg                                     259881        52.29

                                                 Largest Non-German ethnic groups
Borough
Mitte                          Turks, Arabs, Kurds, many Asians, Africans and…
Friedrichshain-Kreuzberg                 Turks, Arabs, African, Kurds, Chinese
Pankow                         Poles, Italians, French, Americans, Vietnamese…
Charlottenburg-Wilmersdorf            Turks, Africans, Russians, Arabs, others.
Spandau                               Turks, Africans, Russians, Arabs, others.
Steglitz-Zehlendorf                        Poles, Turks, Croats, Serbs, Koreans
Tempelhof-Schöneberg                  Turks, Croats, Serbs, Koreans, Africans
Neukölln                        Arabs, Turks, Kurds, Russians, Africans, Poles
Treptow-Köpenick                      Russians, Poles, Ukrainians, Vietnamese
Marzahn-Hellersdorf            Russians, Vietnamese, several other Eastern Eu…
Lichtenberg                     Vietnamese, Russians, Ukrainians, Poles, Chinese
```

[7]:
```
# 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

[8]:
```
a_simple_webpage = \
"""<html>
    <head>
    </head>
    <body>
        <p>
            A paragraph
```

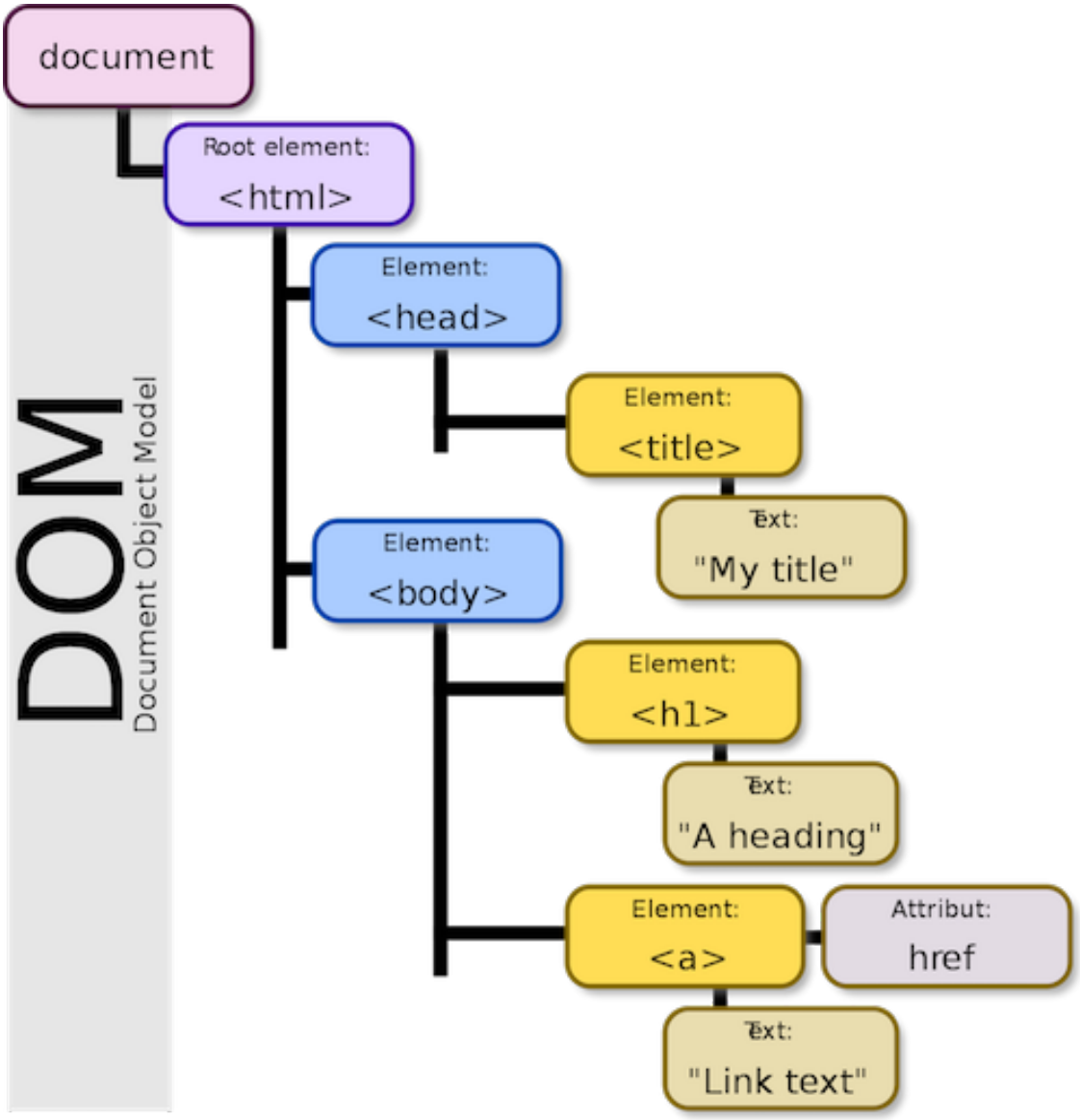
```
        </p>
        <p id="second_paragraph">
           Another paragraph with a <a href="https://de.wikipedia.org/wiki/
↪Beuth_(Lokomotive)">link</a>
        </p>
    </body>
</html>"""
```

[9]:
```
from bs4 import BeautifulSoup
soup = BeautifulSoup(a_simple_webpage, 'html.parser')
```

[10]:
```
list(soup.children)
```

[10]:
```
[<html>
 <head>
 </head>
 <body>
 <p>
            A paragraph
         </p>
 <p id="second_paragraph">
            Another paragraph with a <a
href="https://de.wikipedia.org/wiki/Beuth_(Lokomotive)">link</a>
 </p>
 </body>
 </html>]
```

5.2 DOM Elements



5.2.1 Extracting Texts

[11]:
```
paragraphs = soup.find_all('p')
paragraphs
```

[11]:
```
[<p>
                A paragraph
            </p>, <p id="second_paragraph">
               Another paragraph with a <a
href="https://de.wikipedia.org/wiki/Beuth_(Lokomotive)">link</a>
 </p>]
```

[12]:
```
[p.get_text().strip() for p in paragraphs]
```

[12]:
```
['A paragraph', 'Another paragraph with a link']
```

5.2.2 Extracting Links

[13]:
```
links = soup.find_all('a')
links
```

[13]:
```
[<a href="https://de.wikipedia.org/wiki/Beuth_(Lokomotive)">link</a>]
```

[14]:
```
links[0].get('href')
```

[14]:
```
'https://de.wikipedia.org/wiki/Beuth_(Lokomotive)'
```

5.2.3 Extracting Arbitrary Elements

[15]:
```
[x.get_text().strip() for x in soup.find_all(id="second_paragraph")]
```

[15]:
```
['Another paragraph with a link']
```

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?

[16]:
```
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	Informational	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

Code	Type	Meaning
4xx	Client Error	The request contains bad syntax or cannot be fulfilled
5xx	Server Error	The server failed to fulfill an apparently valid request

[17]:
```
page.content[:500]
```

[17]:
```
b'<!DOCTYPE html>\n<html class="client-nojs" lang="de" dir="ltr">\n<head>\n<meta
charset="UTF-8"/>\n<title>Beuth (Lokomotive) \xe2\x80\x93
Wikipedia</title>\n<script>document.documentElement.className="client-js";RLCONF
={"wgBreakFrames":!1,"wgSeparatorTransformTable":["",\\t.",".\\t,"],"wgDigitTrans
formTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","Januar","Febr
uar","M\xc3\xa4rz","April","Mai","Juni","Juli","August","September","Oktober","N
ovember","Dezember"],"wgRequestId":"44f862b0-2f12-4e08-a8a0-4d629'
```

[18]:
```
# let beautiful soup parse the html
soup = BeautifulSoup(page.content, 'html.parser')
# find a paragraph in which both beuth (the person, not the steam locomotive)␣
 ↪and borsig are mentioned
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 eigenständig in Deutschland entwickelte Dampflokomotive. Vorher
baute Borsig Lokomotiven nach amerikanischen Vorbildern nach. Die Lok gewann ein
Wettrennen gegen ein Modell von Stephenson mit etwa zehn Minuten Vorsprung und
galt für die folgenden zehn Jahre als Prototyp schneller deutscher
Lokomotivkonstruktionen. Eine angetriebene Achse und zwei Laufachsen sowie ein
Stehkessel sorgten für vergleichsweise hohe Geschwindigkeiten. Sie bekam ihren
Namen nach dem Leiter der preußischen Gewerbeakademie Christian Peter Wilhelm
Beuth, der August Borsig prophezeit hatte, dass aus ihm nie etwas werden würde.
Ein Nachbau der Lok ist heute im Deutschen Technikmuseum Berlin ausgestellt.
```

6 Crawling Data from Web Pages with scrapy

DISLAIMER: Doesn't work with most recent immoscout website

- 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/-/-/-/-/tr

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)).ext
            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

[19]:
```
import pandas as pd
flats_df = pd.read_csv('data/mietwohnungen.csv')
flats_df['is24qa_kaltmiete'] = flats_df['is24qa_kaltmiete'].str.replace('[.
 ↪€]','').str.replace(',','.').astype(float)
flats_df['is24qa_flaeche'] = flats_df['is24qa_flaeche'].str.replace(',','.').
 ↪str.extract('(\d+[.\d+])').astype(float)
flats_df['rent_per_qm'] = flats_df['is24qa_kaltmiete'].astype(float) /␣
 ↪flats_df['is24qa_flaeche']
flats_df.sort_values(by='rent_per_qm')[:10]
```

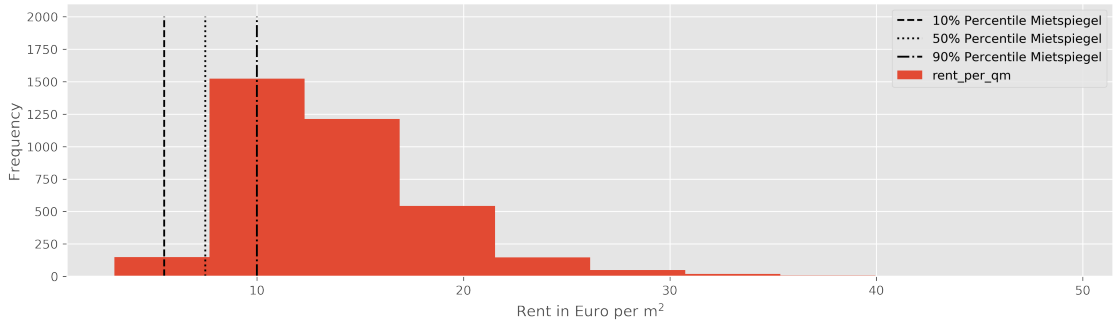
[19]:

	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	

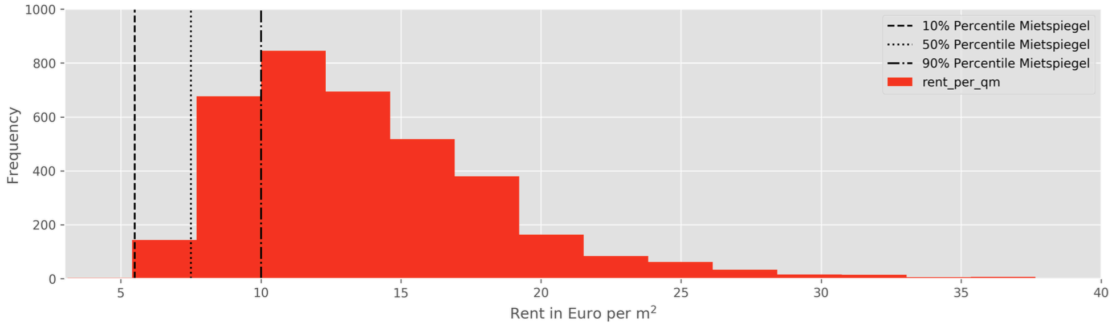
```
1491         Ribnitzer Str 19,    Nach Vereinbarung    10 von 11
2008  Erich-Kästner-Straße 19,               sofort      4 von 5
2675       Altenhofer Str. 40,               sofort     8 von 18
1254      Märkische Allee 280,           01.01.2019    18 von 21

      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
```

[20]:
```
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$^2$");
```



6.4 Rent Distribution 2018



7 Legal Aspects of Scraping

Short version:

If you're doing it for research and if you're not circumventing technical barriers, it's ok.

See https://www.forschung-und-lehre.de/recht/grenzen-des-web-scrapings-2421

7.1 National vs International Legislation

- The internet is global / international
- Legislation / courts are national
- I will focus on **German Legislation** here

7.2 Database Producer Right

The compilation of data served by website providers is protected by *Database Producer Right*

Database Producers (shopping portals, rating portals, …) have exclusive right to: - Reproduce - Distribute - Publicly display

their data

7.3 Implications

Usually web scraping is legal: (Even if website usage conditions state that it's not)

- If only minor parts of a DB are downloaded
- BGH (highest German court) decided 10% of a DB are minor
- If technical barriers are not circumvented

7.4 Implications for Researchers

For non-commercial research: * 'Copyright science barrier': * 75% of a DB can be downloaded * But not distributed!

Since 2018 'Barrier for Text und Data Mining (TDM)' * Researchers can download entire DB to build corpus * Must not be distributed * Must be deleted after research project * Source must be published

8 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

8.1 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

[22]:
```
import requests

# PLEASE USE YOUR OWN ACCESS TOKEN!
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')
```

[22]:
```
{'aqi': 37,
 'idx': 6132,
 'attributions': [{'url':
'http://www.stadtentwicklung.berlin.de/umwelt/luftqualitaet/',
   'name': 'Berlin Air Quality - (Luftqualität in Berlin)',
   'logo': 'Germany-Berlin.png'},
  {'url': 'https://waqi.info/', 'name': 'World Air Quality Index Project'}],
 'city': {'geo': [52.5200066, 13.404954],
  'name': 'Berlin, Germany',
  'url': 'https://aqicn.org/city/germany/berlin'},
 'dominentpol': 'pm10',
 'iaqi': {'h': {'v': 84},
  'no2': {'v': 17.9},
  'o3': {'v': 4.9},
  'p': {'v': 1026.3},
  'pm10': {'v': 37},
  't': {'v': 6.6},
  'w': {'v': 6},
  'wg': {'v': 12}},
 'time': {'s': '2020-11-11 13:00:00',
```

```
 'tz': '+01:00',
 'v': 1605099600,
 'iso': '2020-11-11T13:00:00+01:00'},
'forecast': {'daily': {'o3': [{'avg': 9,
    'day': '2020-11-09',
    'max': 13,
    'min': 6},
   {'avg': 13, 'day': '2020-11-10', 'max': 17, 'min': 9},
   {'avg': 7, 'day': '2020-11-11', 'max': 14, 'min': 2},
   {'avg': 11, 'day': '2020-11-12', 'max': 23, 'min': 3},
   {'avg': 10, 'day': '2020-11-13', 'max': 18, 'min': 6},
   {'avg': 12, 'day': '2020-11-14', 'max': 15, 'min': 7},
   {'avg': 16, 'day': '2020-11-15', 'max': 16, 'min': 16}],
  'pm10': [{'avg': 28, 'day': '2020-11-09', 'max': 30, 'min': 24},
   {'avg': 30, 'day': '2020-11-10', 'max': 32, 'min': 25},
   {'avg': 31, 'day': '2020-11-11', 'max': 34, 'min': 26},
   {'avg': 23, 'day': '2020-11-12', 'max': 32, 'min': 17},
   {'avg': 18, 'day': '2020-11-13', 'max': 21, 'min': 14},
   {'avg': 14, 'day': '2020-11-14', 'max': 18, 'min': 9},
   {'avg': 9, 'day': '2020-11-15', 'max': 9, 'min': 9}],
  'pm25': [{'avg': 79, 'day': '2020-11-09', 'max': 85, 'min': 69},
   {'avg': 81, 'day': '2020-11-10', 'max': 86, 'min': 71},
   {'avg': 81, 'day': '2020-11-11', 'max': 86, 'min': 77},
   {'avg': 67, 'day': '2020-11-12', 'max': 78, 'min': 56},
   {'avg': 53, 'day': '2020-11-13', 'max': 60, 'min': 46},
   {'avg': 46, 'day': '2020-11-14', 'max': 55, 'min': 35},
   {'avg': 36, 'day': '2020-11-15', 'max': 37, 'min': 36}],
  'uvi': [{'avg': 0, 'day': '2020-11-09', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-10', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-11', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-12', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-13', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-14', 'max': 0, 'min': 0},
   {'avg': 0, 'day': '2020-11-15', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-16', 'max': 0, 'min': 0}]}},
'debug': {'sync': '2020-11-11T21:27:34+09:00'}}
```

[23]:
```
get_air_quality_data('newyork')
```

[23]:
```
{'aqi': 16,
 'idx': 3309,
 'attributions': [{'url': 'http://www.dec.ny.gov/',
   'name': 'New York State Department of Environmental Conservation (NYSDEC)',
   'logo': 'US-NYDEC.png'},
  {'url': 'http://www.airnow.gov/', 'name': 'Air Now - US EPA'},
  {'url': 'https://waqi.info/', 'name': 'World Air Quality Index Project'}],
 'city': {'geo': [40.7127837, -74.0059413],
```

```
 'name': 'New York',
 'url': 'https://aqicn.org/city/newyork'},
'dominentpol': 'pm25',
'iaqi': {'h': {'v': 79},
 'p': {'v': 1017.9},
 'pm25': {'v': 16},
 't': {'v': 16.6},
 'w': {'v': 0.3}},
'time': {'s': '2020-11-11 07:00:00',
 'tz': '-05:00',
 'v': 1605078000,
 'iso': '2020-11-11T07:00:00-05:00'},
'forecast': {'daily': {'o3': [{'avg': 3,
    'day': '2020-11-09',
    'max': 19,
    'min': 1},
   {'avg': 5, 'day': '2020-11-10', 'max': 15, 'min': 1},
   {'avg': 11, 'day': '2020-11-11', 'max': 17, 'min': 6},
   {'avg': 4, 'day': '2020-11-12', 'max': 19, 'min': 1},
   {'avg': 2, 'day': '2020-11-13', 'max': 10, 'min': 1},
   {'avg': 14, 'day': '2020-11-14', 'max': 29, 'min': 3}],
  'pm10': [{'avg': 102, 'day': '2020-11-09', 'max': 147, 'min': 77},
   {'avg': 40, 'day': '2020-11-10', 'max': 67, 'min': 22},
   {'avg': 14, 'day': '2020-11-11', 'max': 17, 'min': 8},
   {'avg': 18, 'day': '2020-11-12', 'max': 42, 'min': 5},
   {'avg': 18, 'day': '2020-11-13', 'max': 44, 'min': 6},
   {'avg': 14, 'day': '2020-11-14', 'max': 38, 'min': 6}],
  'pm25': [{'avg': 193, 'day': '2020-11-09', 'max': 251, 'min': 168},
   {'avg': 93, 'day': '2020-11-10', 'max': 162, 'min': 53},
   {'avg': 33, 'day': '2020-11-11', 'max': 42, 'min': 22},
   {'avg': 46, 'day': '2020-11-12', 'max': 99, 'min': 13},
   {'avg': 45, 'day': '2020-11-13', 'max': 96, 'min': 12},
   {'avg': 37, 'day': '2020-11-14', 'max': 78, 'min': 18}],
  'uvi': [{'avg': 0, 'day': '2020-11-10', 'max': 0, 'min': 0},
   {'avg': 0, 'day': '2020-11-11', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-12', 'max': 1, 'min': 0},
   {'avg': 0, 'day': '2020-11-13', 'max': 1, 'min': 0},
   {'avg': 1, 'day': '2020-11-14', 'max': 2, 'min': 0},
   {'avg': 0, 'day': '2020-11-15', 'max': 1, 'min': 0}]}},
'debug': {'sync': '2020-11-11T21:32:40+09:00'}}
```

[24]:
```
get_air_quality_data('seoul')
```

[24]:
```
{'aqi': 104,
 'idx': 5508,
 'attributions': [{'url': 'https://www.airkorea.or.kr/',
   'name': 'South Air Korea Environment Corporation (      )',
```

```
  'logo': 'SouthKorea-AirKorea.png'},
 {'url': 'http://cleanair.seoul.go.kr/',
  'name': 'Seoul Clean Air Pollution Information (        )',
  'logo': 'SouthKorea-Seoul.png'},
 {'url': 'http://www.airkorea.or.kr/',
  'name': 'South Air Korea Environment Corporation (      )',
  'logo': 'SouthKorea-AirKorea.png'},
 {'url': 'https://waqi.info/', 'name': 'World Air Quality Index Project'}],
'city': {'geo': [37.566535, 126.9779692],
 'name': 'Seoul ( )',
 'url': 'https://aqicn.org/city/seoul'},
'dominentpol': 'pm25',
'iaqi': {'co': {'v': 8.9},
 'h': {'v': 62},
 'no2': {'v': 71.8},
 'o3': {'v': 1.6},
 'p': {'v': 1032.1},
 'pm10': {'v': 59},
 'pm25': {'v': 104},
 'r': {'v': 0.2},
 'so2': {'v': 7.2},
 't': {'v': 9.6},
 'w': {'v': 1.2},
 'wd': {'v': 22.5}},
'time': {'s': '2020-11-11 21:00:00',
 'tz': '+09:00',
 'v': 1605128400,
 'iso': '2020-11-11T21:00:00+09:00'},
'forecast': {'daily': {'o3': [{'avg': 7,
    'day': '2020-11-09',
    'max': 23,
    'min': 1},
   {'avg': 4, 'day': '2020-11-10', 'max': 23, 'min': 1},
   {'avg': 3, 'day': '2020-11-11', 'max': 16, 'min': 1},
   {'avg': 3, 'day': '2020-11-12', 'max': 13, 'min': 1},
   {'avg': 2, 'day': '2020-11-13', 'max': 14, 'min': 1},
   {'avg': 3, 'day': '2020-11-14', 'max': 22, 'min': 1},
   {'avg': 2, 'day': '2020-11-15', 'max': 4, 'min': 1}],
  'pm10': [{'avg': 13, 'day': '2020-11-10', 'max': 18, 'min': 0},
   {'avg': 21, 'day': '2020-11-11', 'max': 27, 'min': 18},
   {'avg': 35, 'day': '2020-11-12', 'max': 53, 'min': 18},
   {'avg': 49, 'day': '2020-11-13', 'max': 72, 'min': 18},
   {'avg': 36, 'day': '2020-11-14', 'max': 45, 'min': 27},
   {'avg': 24, 'day': '2020-11-15', 'max': 27, 'min': 18},
   {'avg': 39, 'day': '2020-11-16', 'max': 45, 'min': 27},
   {'avg': 50, 'day': '2020-11-17', 'max': 57, 'min': 45}],
  'pm25': [{'avg': 53, 'day': '2020-11-10', 'max': 67, 'min': 20},
```

```
    {'avg': 67, 'day': '2020-11-11', 'max': 70, 'min': 66},
    {'avg': 106, 'day': '2020-11-12', 'max': 152, 'min': 67},
    {'avg': 136, 'day': '2020-11-13', 'max': 173, 'min': 67},
    {'avg': 111, 'day': '2020-11-14', 'max': 137, 'min': 88},
    {'avg': 75, 'day': '2020-11-15', 'max': 83, 'min': 67},
    {'avg': 109, 'day': '2020-11-16', 'max': 137, 'min': 69},
    {'avg': 147, 'day': '2020-11-17', 'max': 158, 'min': 137}],
   'uvi': [{'avg': 1, 'day': '2020-11-11', 'max': 3, 'min': 0},
    {'avg': 1, 'day': '2020-11-12', 'max': 3, 'min': 0},
    {'avg': 0, 'day': '2020-11-13', 'max': 2, 'min': 0},
    {'avg': 1, 'day': '2020-11-14', 'max': 3, 'min': 0},
    {'avg': 1, 'day': '2020-11-15', 'max': 3, 'min': 0},
    {'avg': 0, 'day': '2020-11-16', 'max': 0, 'min': 0}]}},
 'debug': {'sync': '2020-11-11T21:55:51+09:00'}}
```

9 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

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

[25]:
```
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()
```

[26]:
```
news = json.load(open('news.json'))
news
```

[26]:
```
{'zeit': [{'title': 'Nachrichten, Hintergründe und Debatten',
   'url': 'http://zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-
wirtschaftsmagazin/2018/lebensmittel/was-waere-wenn-die-europaeische-union-sich-
aufloeste?utm_source=zeit&utm_medium=parkett\n      '},
  {'title': 'Nachrichten, Hintergründe und Debatten',
   'url': 'http://zeit.de/\n        https:/www.brandeins.de/magazine/brand-eins-
wirtschaftsmagazin/2018/lebensmittel/food-start-ups-was-mit-
essen?utm_source=zeit&utm_medium=parkett\n      '},
  {'title': 'Nachrichten, Hintergründe und Debatten',
   'url': 'http://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-
eins-wirtschaftsmagazin/2018/lebensmittel/was-waere-wenn-die-europaeische-union-
sich-aufloeste?utm_source=zeit&utm_medium=parkett\n      '},
  {'title': 'Nachrichten, Hintergründe und Debatten',
   'url': 'http://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-
eins-wirtschaftsmagazin/2018/lebensmittel/food-start-ups-was-mit-
essen?utm_source=zeit&utm_medium=parkett\n      '},
  {'title': 'Nachrichten, Hintergründe und Debatten',
   'url': 'https://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-
eins-wirtschaftsmagazin/2018/lebensmittel/was-waere-wenn-die-europaeische-union-
sich-aufloeste?utm_source=zeit&utm_medium=parkett\n      '},
  {'title': 'Nachrichten, Hintergründe und Debatten',
   'url': 'https://www.zeit.de/\n        https:/www.brandeins.de/magazine/brand-
eins-wirtschaftsmagazin/2018/lebensmittel/food-start-ups-was-mit-
essen?utm_source=zeit&utm_medium=parkett\n      '}],
 'tagesspiegel': [{'title': 'IT-Systemadministrator (m/w)',
   'url': 'https://karriere.tagesspiegel.de/stellenangebot/2563936/IT_Systemadmi
nistrator_(m_w)_Neubrandenburg_Hochs'},
  {'title': 'Büromitarbeiter/-in',
   'url': 'https://karriere.tagesspiegel.de/stellenangebot/2539229/B%C3%BCromita
rbeiter__in_Berlin_Sensible_Osteopathie'},
  {'title': 'Sekretärin/Sekretär',
   'url': 'https://karriere.tagesspiegel.de/stellenangebot/2539233/Sekret%C3%A4r
in_Sekret%C3%A4r_Berlin_Kampmann_Architekten_G'},
  {'title': 'Mitarbeiterin für Privatsekretariat',
   'url': 'https://karriere.tagesspiegel.de/stellenangebot/2572190/Mitarbeiterin
_f%C3%BCr_Privatsekretariat_Berlin_Dr__We'},
  {'title': 'Lehrkraft (w/m/d)',
   'url': 'https://karriere.tagesspiegel.de/stellenangebot/2558134/Lehrkraft_(w_
m_d)_Schwielowsee_Ev__Jugendhilfe_Ge'}]}
```

9.1 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
"}]
```


10 Exercises

10.1 Assignment 01

Write a function `assignment_07_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_07_01():
    # finds all csv files in data/random_numbers
    files = glob.glob(os.path.join("data",'random_numbers','*.csv'))
    # ...
    return sum_of_values
```

[ ]:
```
assignment_07_01() == 203455
```

10.2 Assignment 02

Write a function `assignment_07_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

beuth_url = "https://de.wikipedia.org/wiki/
 ↪Beuth_Hochschule_f%C3%BCr_Technik_Berlin"

def assignment_07_02(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.content, 'html.parser')
    infobox = {}

    #...

    return infobox
```

[ ]:
```
infobox = assignment_07_02(beuth_url)
assert infobox['Ort'] == 'Berlin-Wedding'
```

10.3 Assignment 03

Write a function `assignment_07_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_07_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
```

[ ]:
```
assignment_07_03() == 'Steglitz-Zehlendorf'
```