How to find a cozy room?

A data mining project in rental market

Warm-up

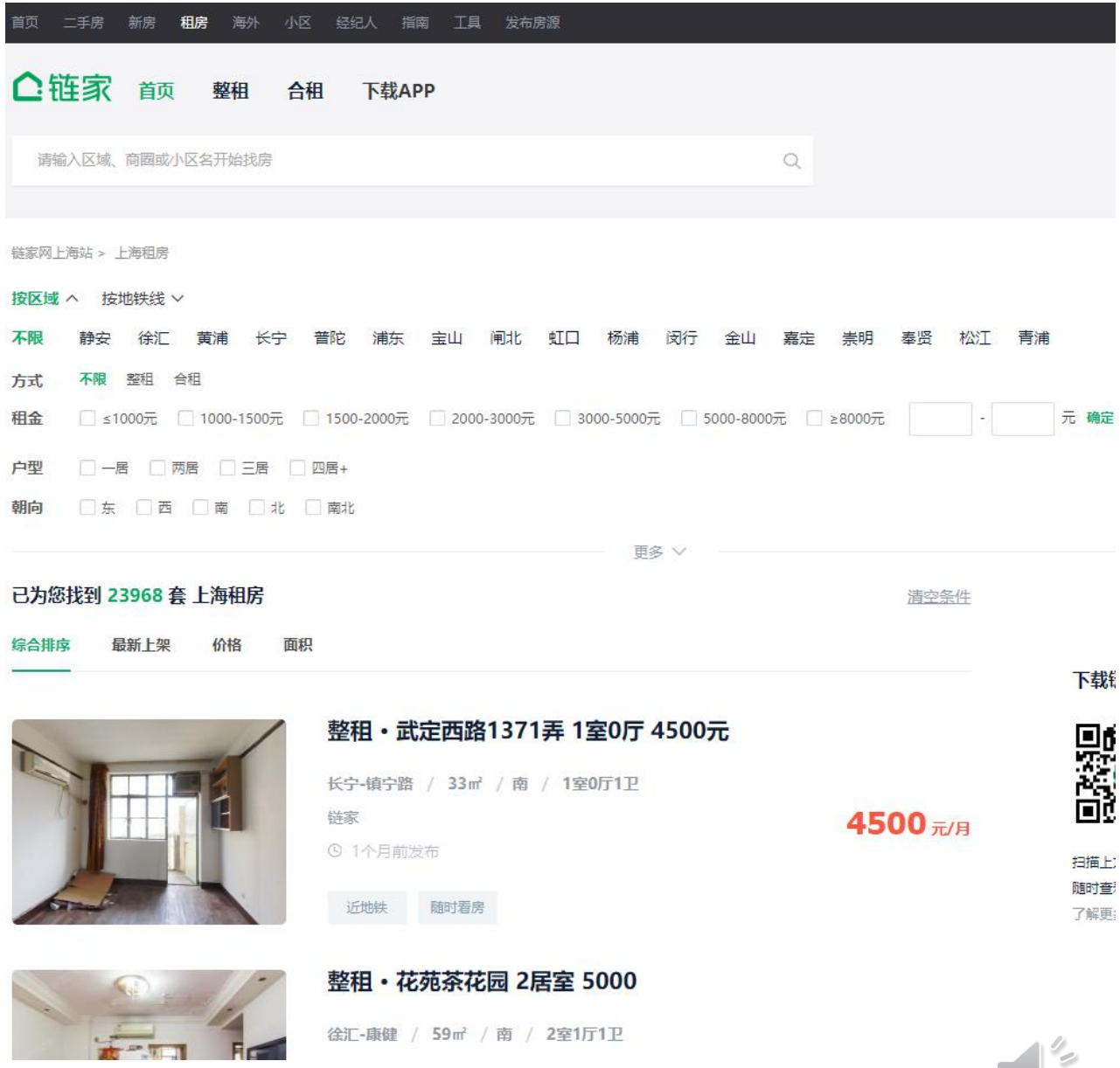
- Have you ever rented a room?
- What factors will you consider?
- Will you take the room shown here?

Aim

- In this project, we want to study the rental market in Shanghai in a quantitative way by investigating the relationship between unit price and some other related variables.

Plan

- Crawl the data from http://sh.lianjia.com/zufang
- Clean the data and extract factors, unit price and more
- Visualize the factors
- Explore the relationship between unit price and variables



Tools

- Chrome: learn to press F12
- Python with packages
- Web crawler: urllib, requests, re, beatifulsoup
- Data analysis: NumPy, pandas, scikit-learn (sklearn)
- Data visualization: pyecharts, matplotlib

Web crawler

- A Web crawler starts with **a list of URLs to visit**, called the *seeds*. As the crawler visits these URLs, it identifies all the hyperlinks in the page and **adds them to the list of URLs to visit**, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites it saves the information as it goes. The archives are usually stored as ‘snapshots'.

https://en.wikipedia.org/wiki/Web_crawler

Web crawler

- Situations to face
  - The archive is known as the *repository* and is designed to store and manage the collection of web pages. The repository only stores HTML pages and these pages are stored as distinct files.
  - The large volume implies the crawler can only download a limited number of the Web pages within a given time.
  - The number of possible URLs crawled being generated by server-side software has also made it difficult for web crawlers to avoid retrieving duplicate content.

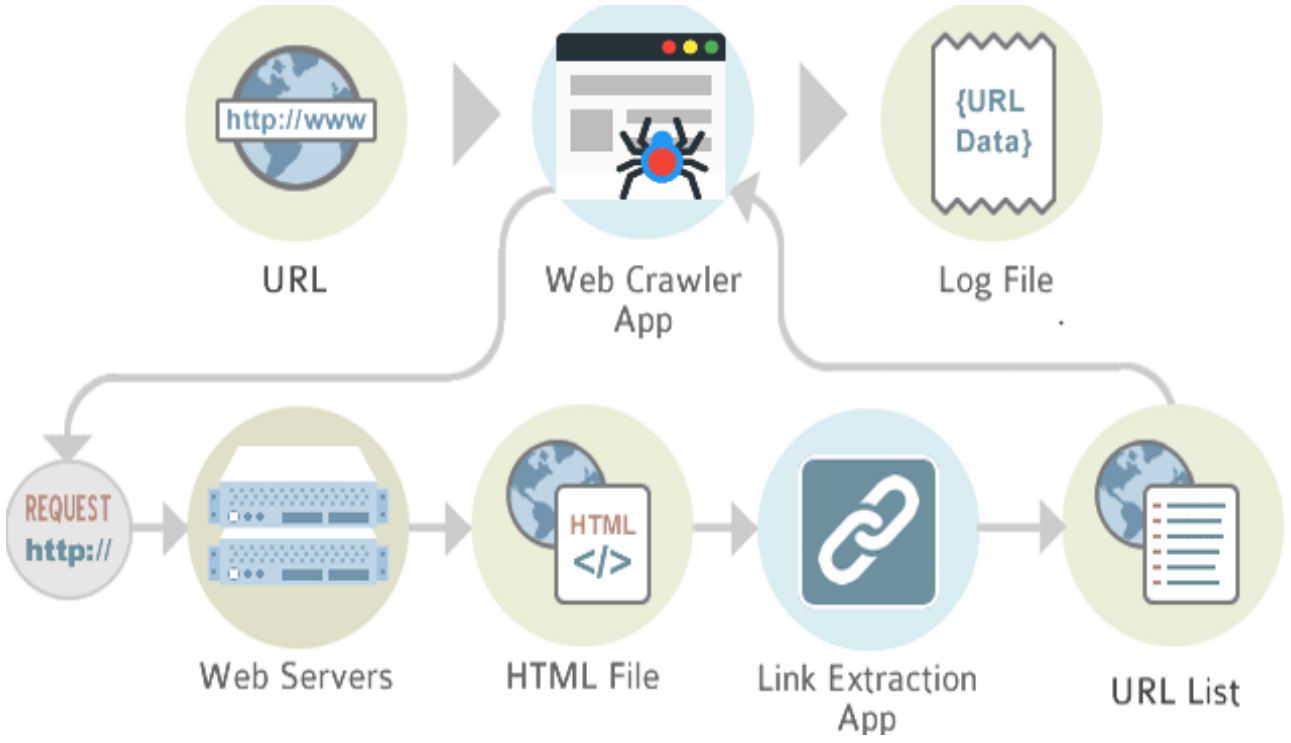
https://en.wikipedia.org/wiki/Web_crawler

Web crawler

- Crawling policy
  - a *selection policy* which states the pages to download,
  - a *re-visit policy* which states when to check for changes to the pages,
  - a *politeness policy* that states how to avoid overloading Web sites.
  - a *parallelization policy* that states how to coordinate distributed web crawlers.

https://en.wikipedia.org/wiki/Web_crawler

Web crawler



https://fiverr-res.cloudinary.com/images/

Web crawler: html

- **Hypertext Markup Language** (**HTML**)
  - is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.
  - Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

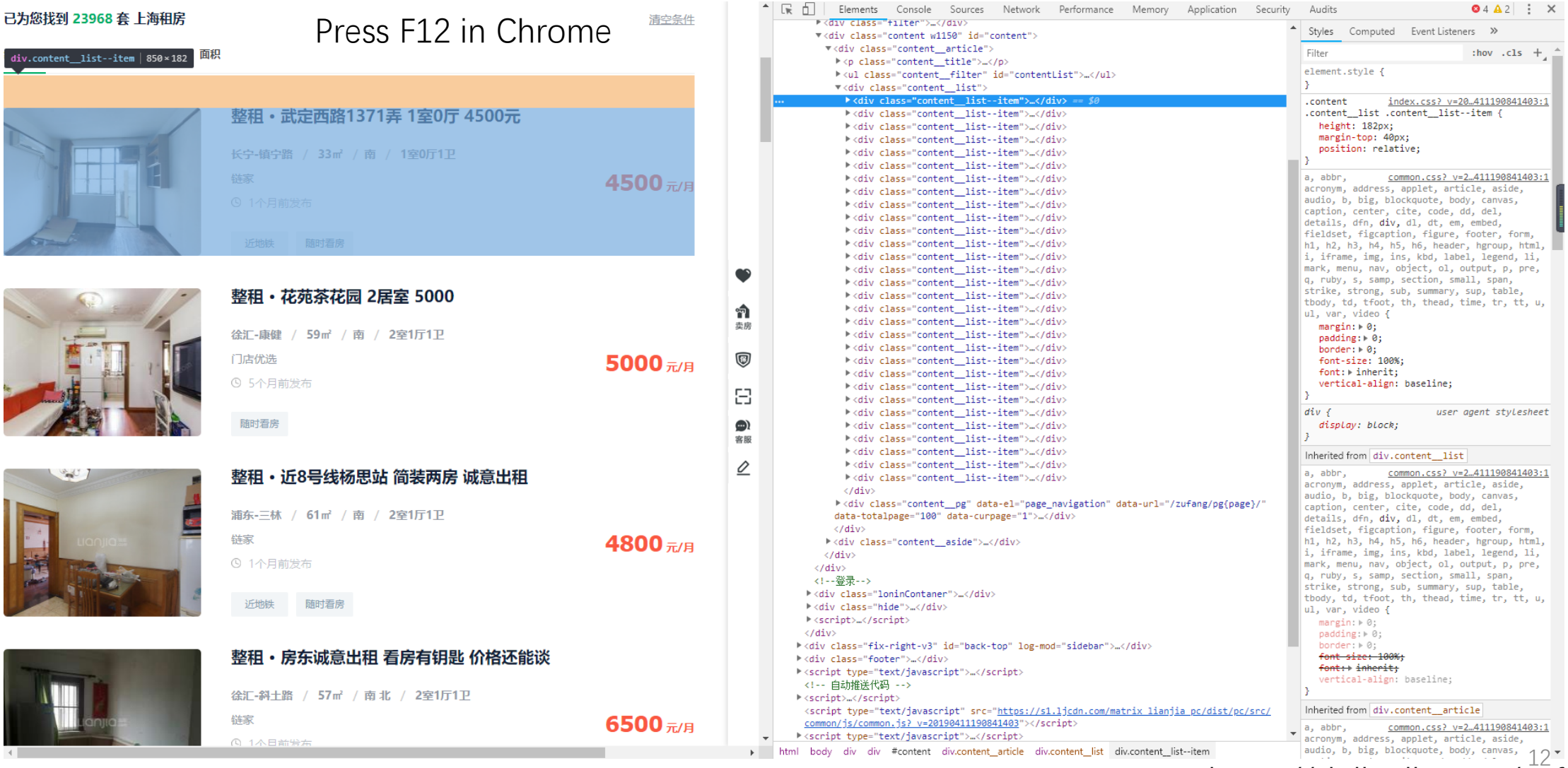
https://en.wikipedia.org/wiki/HTML

Web crawler: html

- HTML elements
  - building blocks of HTML pages. Images and objects such as interactive forms may be embedded into the page.
  - text can have headings, paragraphs, lists, links, quotes and other items.
  - HTML elements are denoted by *tags*, written using angle brackets. Tags such as <**img** /> and <**input** /> directly introduce content into the page.

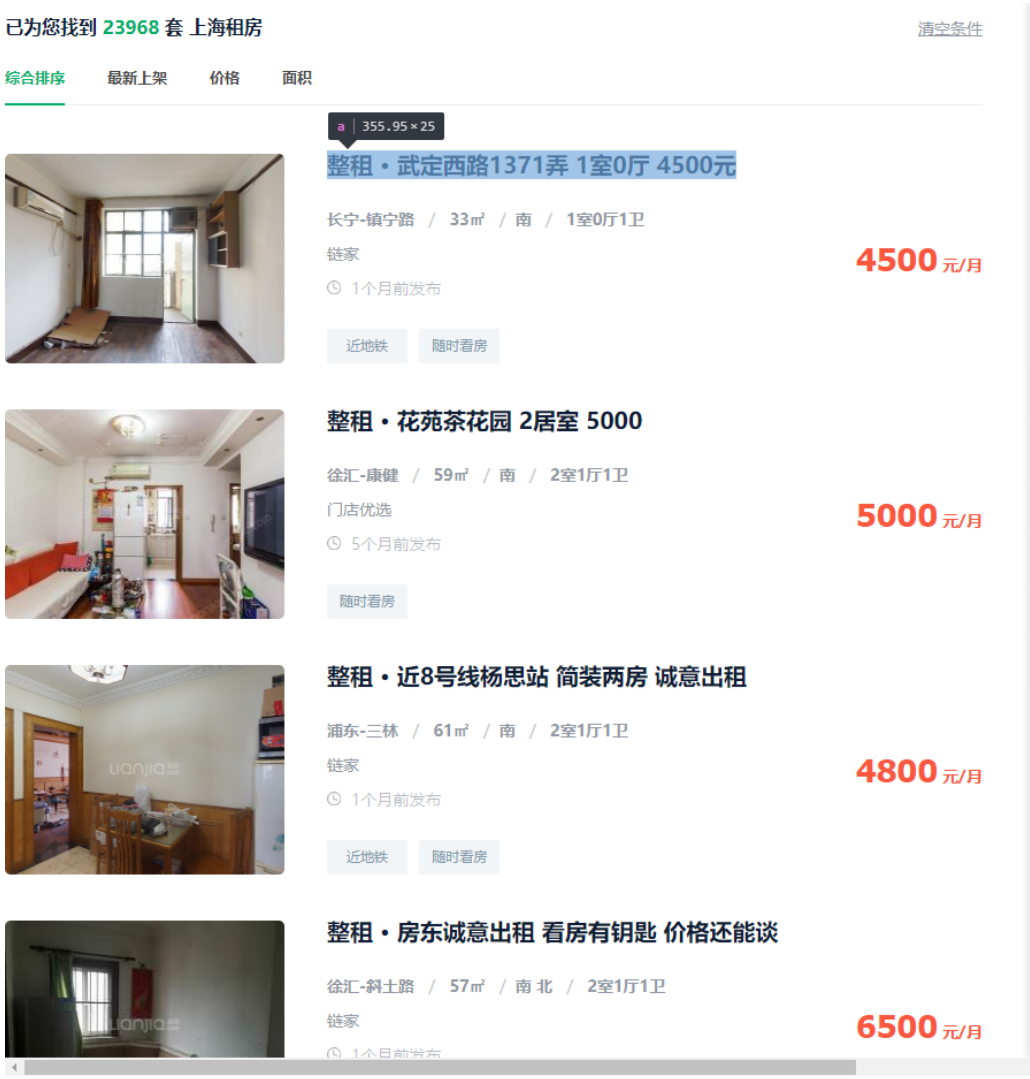
https://en.wikipedia.org/wiki/HTML

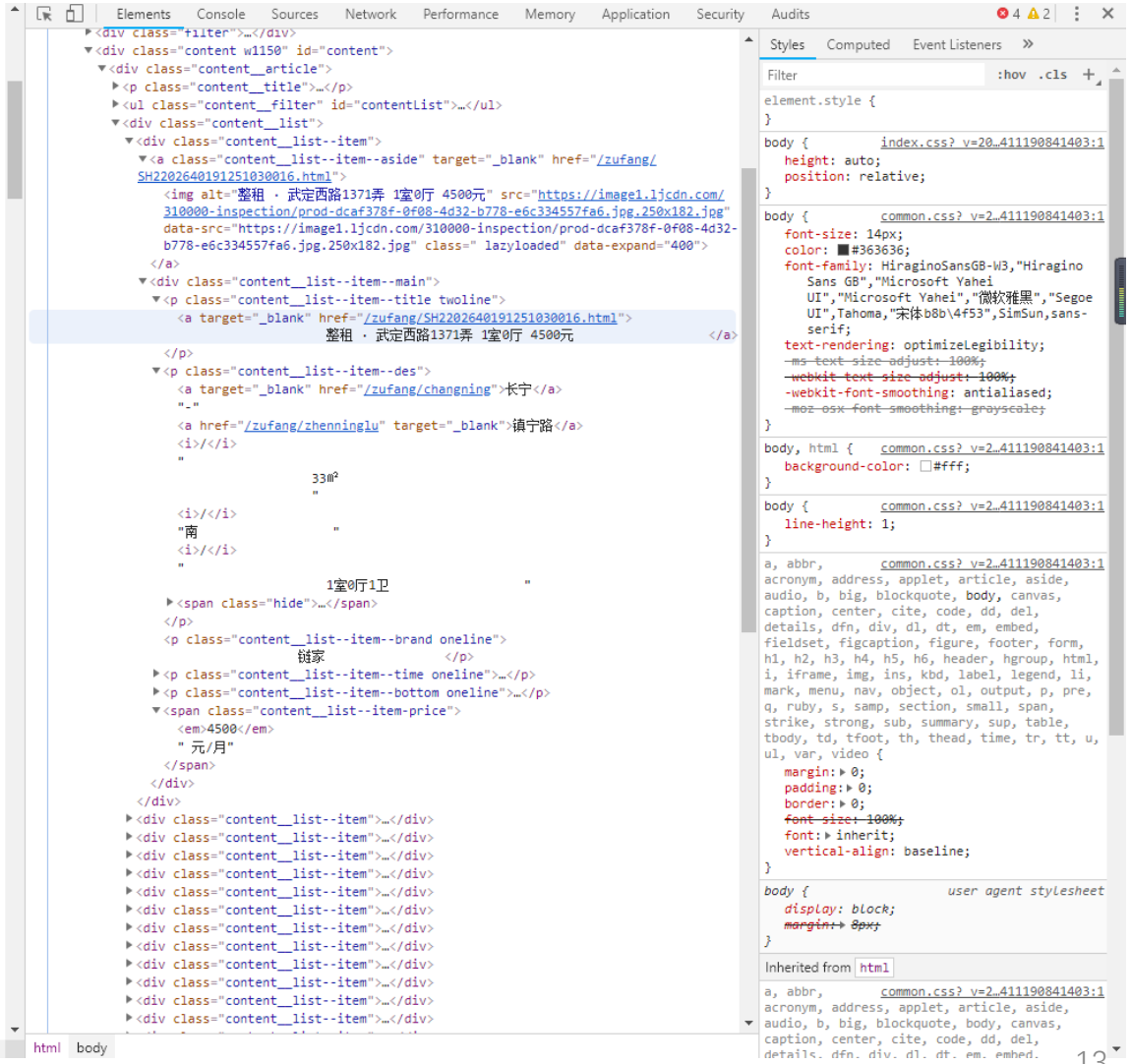
Web crawler: web pages



https://sh.lianjia.com/zufang/d1

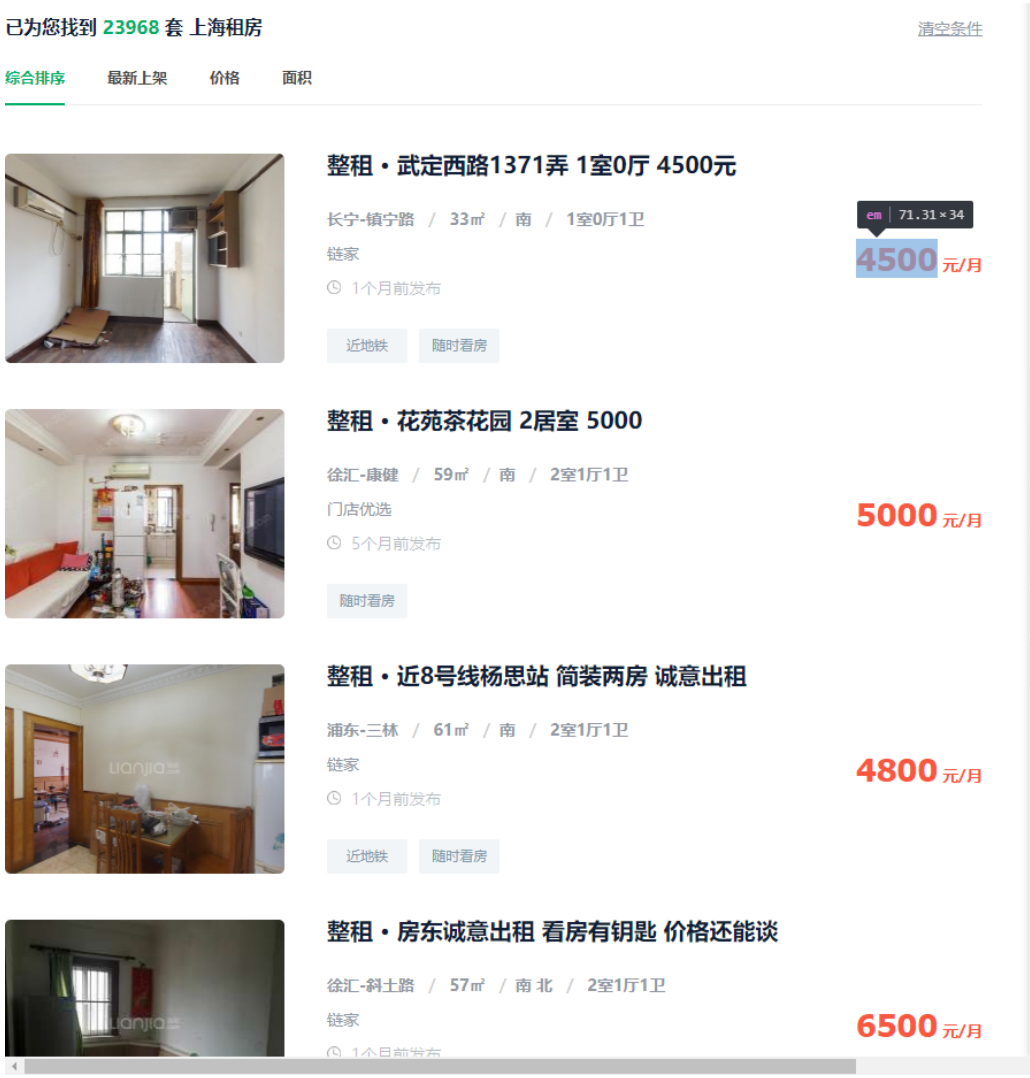
Web crawler: web pages





https://sh.lianjia.com/zufang/d1

Web crawler: web pages



https://sh.lianjia.com/zufang/d1

Web crawler: web pages

- Download the web page.
  - import urllib.request
  - i=1
  - html = urllib.request.urlopen('http://sh.lianjia.com/zufang/d'+str(i)).read().decode('utf-8')
- Try:
  - Go to https://sh.lianjia.com/zufang/d1
  - Download https://sh.lianjia.com/zufang/d2, …, up to https://sh.lianjia.com/zufang/d100 by a for-loop

Web crawler: web pages

- Print the page and see what is in the html file.

What is this?

```
In [4]: i=1
html = urllib.request.urlopen('http://sh.lianjia.com/zufang/d'+str(i)).read().decode('utf-8')

In [5]: print(html)
```

```
                    <p class="content__list--item--title twoline">
                      <a target="_blank" href="/zufang/SH2202640191251030016.html">
                        整租 · 武定西路1371弄 1室0厅 4500元
                      </a>
                    </p>
                    <p class="content__list--item--des">
                      <a target="_blank" href="/zufang/changning">长宁</a>-<a href="/zufang/zhenninglu" target="_
blank">镇宁路</a>
                      <i>/</i>
                      33㎡
                      <i>/</i>南
                      <i>/</i>
                      1室0厅1卫
                      <span class="hide">
                      <i>/</i>
                      中楼层
                      (6层)
                    </span>
                    </p>
                    <p class="content__list--item--brand oneline">
                      链家
                    </p>
                    <p class="content__list--item--time oneline">1个月前发布</p>
                    <p class="content__list--item--bottom oneline">
                    <i class="content__item__tag--is_subway_house">近地铁</i>
```

Web crawler: web pages

- A detailed page is shown.

Web crawler: web pages

- Pieces of detailed information are listed in the URL with a string started with ‘SH’.
- Next step is to obtain all the URLs such as https://sh.lianjia.com/zufang/SH2202640191251030016.html and add them to a list.
- Knowledge of **regular expression** is needed.

Web crawler: regular expression

- A **regular expression**, **regex** or **regexp** is a sequence of characters that define a *search pattern*.
- Used to specify a set of strings required for a particular purpose. Often called a **pattern**.
- In 1950s mathematician Stephen Cole Kleene formalized the description of a *regular language*. It came into common use with Unix text-processing utilities in 80s.

https://en.wikipedia.org/wiki/Regular_expression

Web crawler: regular expression

- **Boolean** "or“:
  - A vertical bar separates alternatives. gray|grey can match "gray" or "grey".
- **Grouping**:
  - Parentheses are used to define the scope and precedence of the operators. gray|grey and gr(a|e)y are equivalent patterns, both describe the set of "gray" or "grey".

https://en.wikipedia.org/wiki/Regular_expression

Web crawler: regular expression

- **Quantification**:
  - A quantifier after a token (e.g a character) or group specifies how often that a preceding element is allowed to occur. Common quantifiers are the question mark ?, the asterisk *, and the plus sign +.
- ?
  - Indicates zero or one occurrence of the preceding element. colou?r matches both "color" and "colour".
- *
  - Indicates zero or more occurrences of the preceding element. ab*c matches "ac", "abc", "abbc", "abbbc", and so on.

https://en.wikipedia.org/wiki/Regular_expression

Web crawler: regular expression

- +
  - Indicates **one or more** occurrences of the preceding element. For example, `ab+c` matches "abc", "abbc", "abbbc", and so on, but not "ac".
- {n}
  - The preceding item is matched exactly n times.
- {min,}
  - The preceding item is matched min or more times.
- {min,max}
  - The preceding item is matched at least min times, but not more than max times.

https://en.wikipedia.org/wiki/Regular_expression

Web crawler: regular expression

- **Wildcard**
  - The wildcard . matches any character. For example, a.b matches any string that contains an "a", then any other character and then a "b", a.*b matches any string that contains an "a" and a "b" at some later point.
- These constructions can be combined to form **arbitrarily** complex expressions.

https://en.wikipedia.org/wiki/Regular_expression

Web crawler: regular expression

- Package *re* in Python:
- *re.compile(pattern, flags=0)*
  - Compile a regular expression pattern into a regular expression object, to be used for matching using its match(), search() and other methods.
- *re.findall(pattern, string, flags=0)*
  - Return all non-overlapping matches of pattern in string, as a list of strings. The string is scanned left-to-right, and matches are returned in the order found. If one or more groups are present in the pattern, return a list of groups; this will be a list of tuples if the pattern has more than one group. Empty matches are included in the result.

https://docs.python.org/3/library/re.html#regular-expression-syntax

Web crawler: regular expression

- Find URLs like /zufang/SH22026401912510300l6.html in HTML

```
<div class="content__list--item--main">
  <p class="content__list--item--title twoline">
    <a target="_blank" href="/zufang/SH22026401912510300l6.html">
      整租 · 武定西路1371弄 1室0厅 4500元                    </a>
  </p>
```


Web crawler: regular expression

```
import re
pattern_house_url = '<a target="_blank" href="(/zufang/SH\d*\.html)">'
house_url = re.compile(pattern_house_url).findall(html)
```

- The highlight part is expected.

```
<div class="content__list--item--main">
  <p class="content__list--item--title twoline">
    <a target="_blank" href="/zufang/SH2202640191251030016.html">
      整租 · 武定西路1371弄 1室0厅 4500元                    </a>
  </p>
```


Web crawler: regular expression

- Print and check obtained URLs in the html.

```
In [7]: print(house_url)
```

```
['/zufang/SH2202640191251030016.html', '/zufang/SH2118065364868284416.html', '/zufang/SH2202735136326287360.html', '/zufang/SH2202827081300590592.html', '/zufang/SH2203600761156812800.html', '/zufang/SH2204255418761887744.html', '/zufang/SH2125351985594245120.html', '/zufang/SH2204278043474657280.html', '/zufang/SH2204400164872404992.html', '/zufang/SH2207071044735877120.html', '/zufang/SH2207104618712793088.html', '/zufang/SH2166375793389731840.html', '/zufang/SH2207731073776369664.html', '/zufang/SH2207837728912187392.html', '/zufang/SH2207887402079895552.html', '/zufang/SH2208692475068432384.html', '/zufang/SH2189736028540518400.html', '/zufang/SH2209392328781594624.html', '/zufang/SH2209407044606230528.html', '/zufang/SH2210088101043511296.html', '/zufang/SH2210185310271176704.html', '/zufang/SH2197570070966697984.html', '/zufang/SH2210826089419513856.html', '/zufang/SH2212173277810663424.html', '/zufang/SH2212203789283229696.html', '/zufang/SH2213061614511595520.html', '/zufang/SH2137670128563920896.html', '/zufang/SH2213082905108807680.html', '/zufang/SH2213685431558742016.html', '/zufang/SH2214222747201454080.html']
```

- These URLs are not complete.

Web crawler: regular expression

- Complete obtained URLs
  - url_list = []
  - for j in house_url:
  - url_list.append('http://sh.lianjia.com'+str(j))

```
In [9]: print(url_list)
```

```
['http://sh.lianjia.com/zufang/SH2202640191251030016.html', 'http://sh.lianjia.com/zufang/SH2118065364868284416.
html', 'http://sh.lianjia.com/zufang/SH2202735136326287360.html', 'http://sh.lianjia.com/zufang/SH22028270813005
90592.html', 'http://sh.lianjia.com/zufang/SH2203600761156812800.html', 'http://sh.lianjia.com/zufang/SH22042554
18761887744.html', 'http://sh.lianjia.com/zufang/SH2125351985594245120.html', 'http://sh.lianjia.com/zufang/SH22
04278043474657280.html', 'http://sh.lianjia.com/zufang/SH2204400164872404992.html', 'http://sh.lianjia.com/zufan
g/SH2207071044735877120.html', 'http://sh.lianjia.com/zufang/SH2207104618712793088.html', 'http://sh.lianjia.co
m/zufang/SH2166375793389731840.html', 'http://sh.lianjia.com/zufang/SH2207731073776369664.html', 'http://sh.lian
jia.com/zufang/SH2207837728912187392.html', 'http://sh.lianjia.com/zufang/SH2207887402079895552.html', 'http://s
h.lianjia.com/zufang/SH2208692475068432384.html', 'http://sh.lianjia.com/zufang/SH2189736028540518400.html', 'ht
tp://sh.lianjia.com/zufang/SH2209392328781594624.html', 'http://sh.lianjia.com/zufang/SH2209407044606230528.htm
l', 'http://sh.lianjia.com/zufang/SH2210088101043511296.html', 'http://sh.lianjia.com/zufang/SH22101853102711767
04.html', 'http://sh.lianjia.com/zufang/SH2197570070966697984.html', 'http://sh.lianjia.com/zufang/SH22108260894
19513856.html', 'http://sh.lianjia.com/zufang/SH2212173277810663424.html', 'http://sh.lianjia.com/zufang/SH22122
03789283229696.html', 'http://sh.lianjia.com/zufang/SH2213061614511595520.html', 'http://sh.lianjia.com/zufang/S
H2137670128563920896.html', 'http://sh.lianjia.com/zufang/SH2213082905108807680.html', 'http://sh.lianjia.com/zu
fang/SH2213685431558742016.html', 'http://sh.lianjia.com/zufang/SH2214222747201454080.html']
```

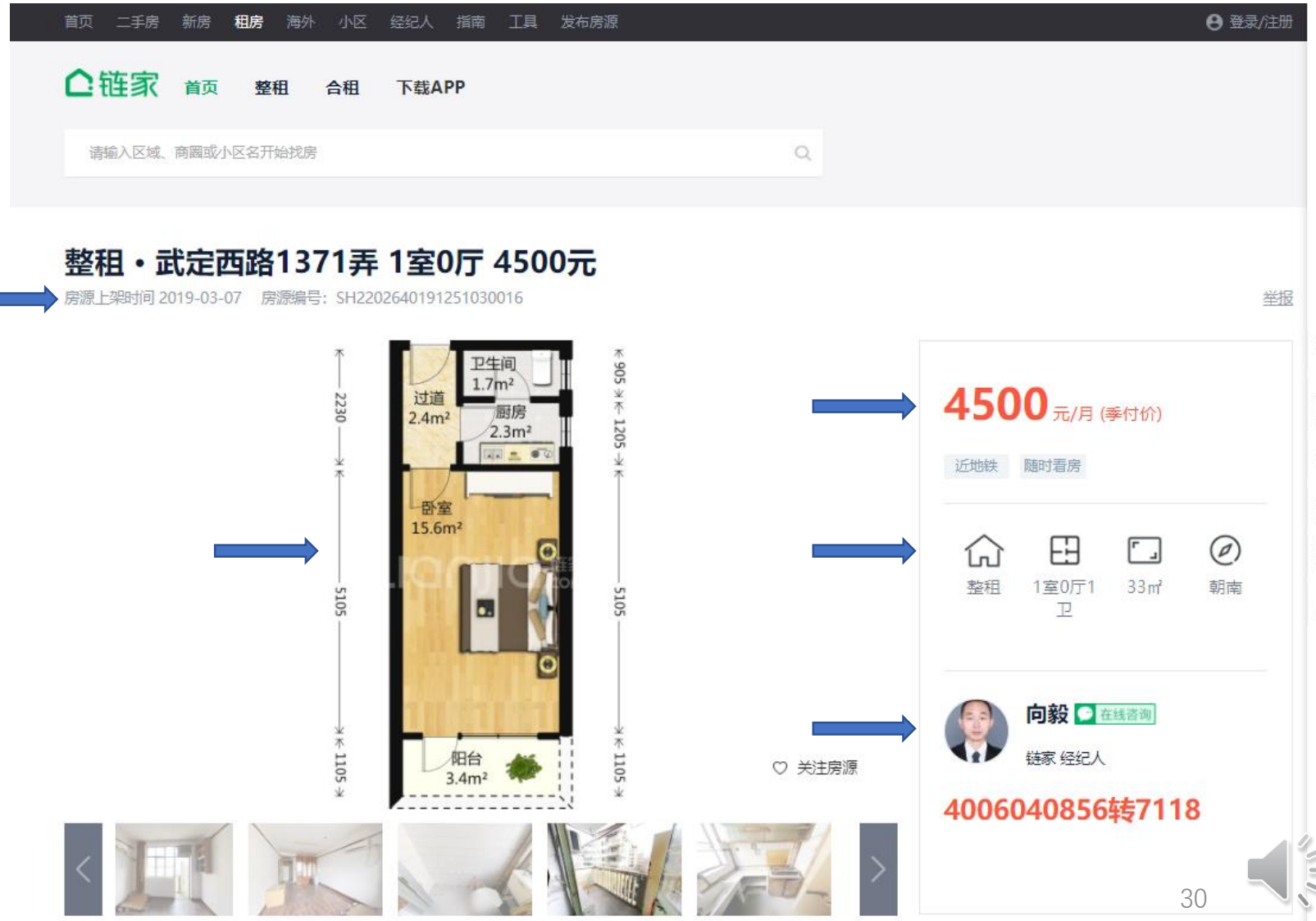
Web crawler: regular expression

- Visit the first URL in the url_list and get the html
- Use the pattern to get the content title from the html

```
In [10]: i=0
         html = urllib.request.urlopen(url_list[i]).read().decode('utf-8')

In [11]: print(url_list[i])

         http://sh.lianjia.com/zufang/SH2202640191251030016.html

In [12]: pattern_introduction = '<p class="content__title">(.*)</p>'
         introduction = re.compile(pattern_introduction).findall(html)
         print(introduction)

         ['整租 · 武定西路1371弄 1室0厅 4500元']
```


Web crawler: regular expression

- More details to explore:
  - Room plan
  - Price
  - Period of payment
  - Agent
  - ……
- All can be found by **regular expression**!



Web crawler: beautiful soup

- Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly **saves programmers hours or days of work**.
- Let's try this new tool to grab something from https://sh.lianjia.com/zufang/SH22026401912510300 16.html

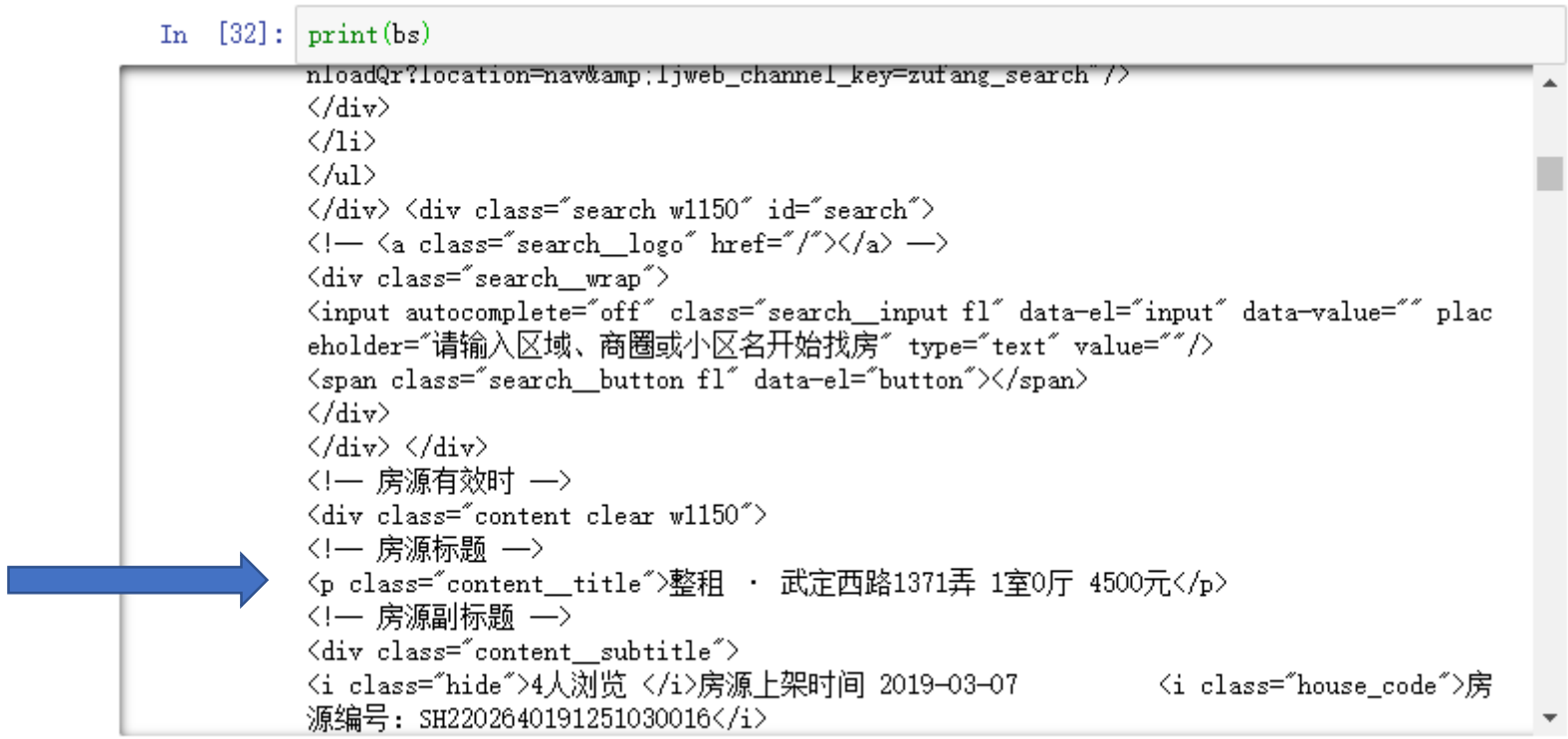
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Web crawler: beautiful soup

- Import packages.
  - from bs4 import BeautifulSoup
  - import requests
- To “fool” sh.lianjia.com, we need some mask.
  - header={'User-Agent':'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.108 Safari/537.36’,}
- Download the page as usual.
  - data=requests.get('https://sh.lianjia.com/zufang/SH2202640191251030016.html',headers=header)
  - bs=BeautifulSoup(data.text)

Web crawler: beautiful soup

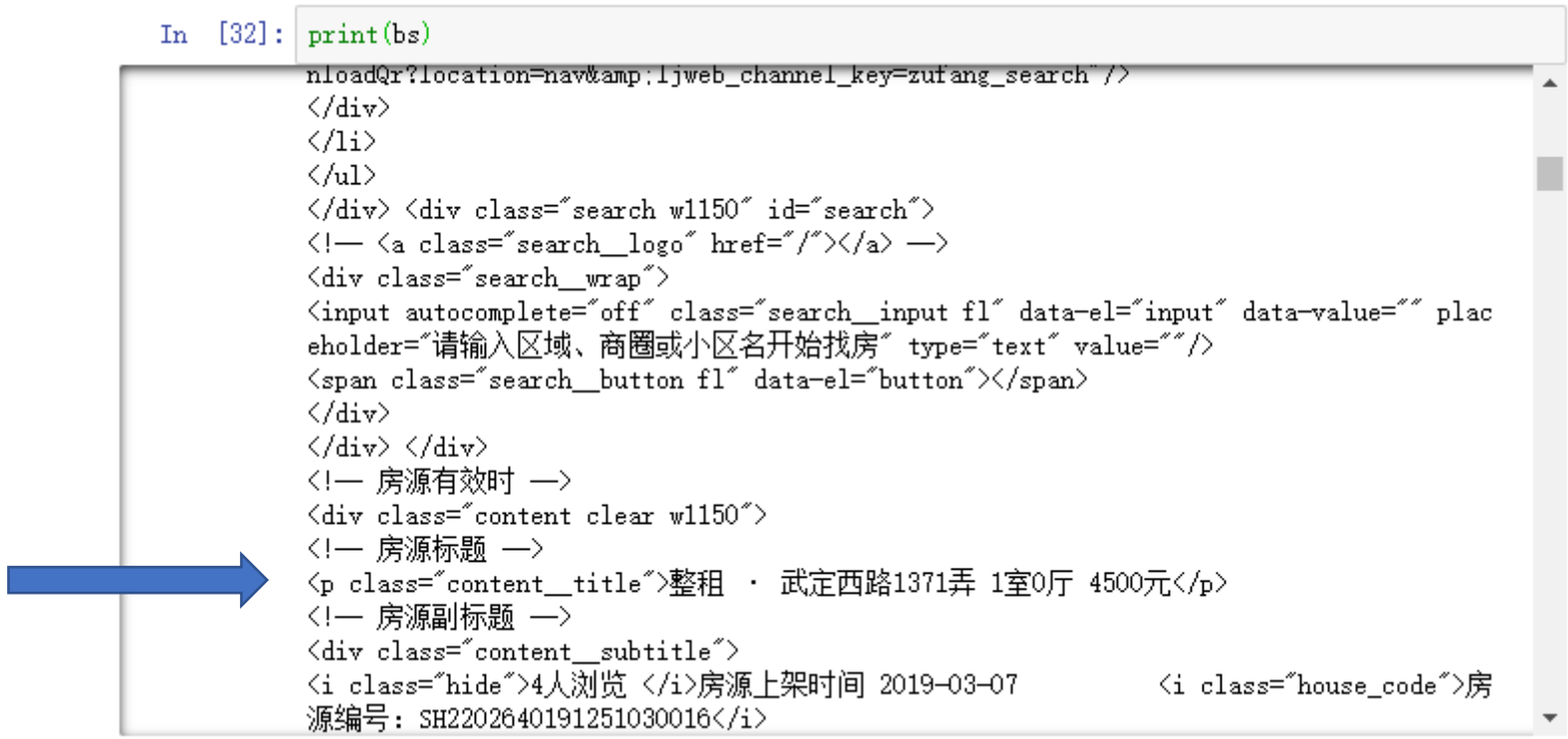
- We want to obtain the content title as we did previously.



```
In [32]: print(bs)
nloadQr?location=nav&amp;ljweb_channel_key=zufang_search"/>
</div>
</li>
</ul>
</div> <div class="search w1150" id="search">
<!— <a class="search__logo" href="/"></a> —>
<div class="search__wrap">
<input autocomplete="off" class="search__input fl" data-el="input" data-value="" plac
eholder="请输入区域、商圈或小区名开始找房" type="text" value=""/>
<span class="search__button fl" data-el="button"></span>
</div>
</div> </div>
<!— 房源有效时 —>
<div class="content clear w1150">
<!— 房源标题 —>
<p class="content__title">整租 · 武定西路1371弄 1室0厅 4500元</p>
<!— 房源副标题 —>
<div class="content__subtitle">
<i class="hide">4人浏览 </i>房源上架时间 2019-03-07          <i class="house_code">房
源编号：SH22026401912510300l6</i>
```

Web crawler: beautiful soup

- Recall the pattern we wrote
  - pattern_introduction = ' (.*) '



Web crawler: beautiful soup

- We only need use *find* by specifying the tag <p> and class 'content_title'
  - bs.find('p',class_='content__title').text

```
<p class="content__title">整租 · 武定西路1371弄 1室0厅 4500元</p>
<!— 房源副标题 —>
<div class="content__subtitle">
<i class="hide">4人浏览 </i>房源上架时间 2019-03-07          <i class="house_code">房
源编号：SH2202640191251030016</i>
```

```
In [33]: #<p class="content__title">整租 · 武定西路1371弄 1室0厅 4500元</p>
         bs.find('p',class_='content__title').text

Out[33]: '整租 · 武定西路1371弄 1室0厅 4500元'
```

We get the same result as the previously obtained.

Data mining starts from here!

- With obtained data, we start data mining.
- This part is organized as follows.
  - Data description
  - Visualization
  - Correlation

Data description

Variables	Description
url	url
title	Title
price	Price per month
housing_estate	Building
halls	Number of living rooms
rooms	Number of rooms
area	Total area
address	Address
price_per_area	Unit price (square meter)
district	District
block	Block
floor_type	Floor types (high, medium, low)
floor	Floor number

Data description

Variable	Description
face_direction	Room facing direction
nearby_metro_line	Metro line nearby
nearby_metro_station	Metro station nearby
distance_to_metro	Nearest Metro station
rent_type	Rental type (1 for 合租, 0 for 整租)
has_own_washroom	Private washroom availability (yes, no)
has_own_balcony	Private balcony availability (yes, no)
is_on_sale	On sale status (yes, no)
update_time	Update time
bringin_records	Bring-in records
bringin_7_days	Bring-in records within last 7 days
bringin_count	Number of bring-in records
images	Pictures of rooms for rent
layout_image	Floor plan
time_to_pplsquare	Hours travelling to people's square

Data description

- **pandas** is suited for different kinds of data:
  - **Tabular data** with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
  - Ordered and unordered (not necessarily fixed-frequency) **time series data**.
  - Arbitrary **matrix data** (homogeneously typed or heterogeneous) with row and column labels
  - Any other form of **observational / statistical data sets**. The data need not be labeled.

https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html

Data description

- Here are just a few of the things that pandas does well:
  - Handling missing data
  - Inserting and deleting columns from DataFrame
  - Aggregating and transforming data
  - Slicing, fancy indexing, and subsetting of large datasets
  - Reshaping of datasets
  - ……

https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html
https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Data description

Column/ variable

```
In [14]: import pandas as pd

In [19]: df_lj = pd.read_csv('house_data.csv')                                    ıse_data.csv')

In [21]: df_lj.head(3)
```

Out[21]:

	address	area	block	bringin_7_days	bringin_count	distance_to_metro	district	face_direction	fitment_type	floor	...	nearby_metro_station	price	pı	price	price_per_area	rent_type	rooms	title	update_time	url	time_to_pplsquare	Unnamed: 26
0	佳林路115弄（一期）佳林路82弄（二期）佳京路99弄（三期）金高路1617弄（四期）	171	金桥	0	2	0	浦东	(进门) 南	0	0	...	NaN	14500.0		14500.0	84.795322	0	4	阳光欧洲城，链家好房，大气四室	2017.01.13	http://sh.lianjia.com/zufang/shz3723680.html	-1	NaN
1	宋园路69弄	110	古北	0	0	535	长宁	朝南北	0	16	...	宋园路	19000.0		19000.0	172.727273	0	2	圣美邸，有钥匙方便看，南北户型，2室2厅1卫	2017.02.02	http://sh.lianjia.com/zufang/shz3739422.html	20	NaN
2	嘉涌路99弄	43	江桥	2	2	0	嘉定	朝西	0	9	...	NaN	4000.0		4000.0	93.023256	0	2	MAX未来，好楼层，放心好房，高清实拍	2017.02.02	http://sh.lianjia.com/zufang/shz3739516.html	-1	NaN

Row/Observation

3 rows × 27 columns

Data description

- To simplify the case, we select variables such as block, district, area, and price_per_area to study.
- Select desired columns.

```
In [36]: df_use = df_lj[['block','district','area','price_per_area','rent_type']]

In [37]: df_use.head(5)
```

Out[37]:

	block	district	area	price_per_area	rent_type
0	金桥	浦东	171	84.795322	0
1	古北	长宁	110	172.727273	0
2	江桥	嘉定	43	93.023256	0
3	万里	普陀	77	77.922078	0
4	曹路	浦东	89	65.168539	0

Data description

- First of all, we want to see some descriptive statistics of numerical variables.
- Use *df.describe* to obtain count, mean, standard deviation, and quantiles.

```
In [46]: df_use.describe()
```

Out[46]:

	area	price_per_area	rent_type
count	45182.000000	45182.000000	45182.000000
mean	83.676619	88.035169	0.074631
std	62.806153	174.941158	0.262799
min	1.000000	1.052632	0.000000
25%	50.000000	51.562500	0.000000
50%	76.000000	76.923077	0.000000
75%	102.000000	109.375000	0.000000
max	3800.000000	35000.000000	1.000000

Data description

- We now have 45182 candidates totally.
- If we want to know numbers of rental rooms **in each district**, we need to use *df.groupby*.
- We call this action **aggregation**.
- See details in
  - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html

```
In  [42]: df_use.groupby(by='district').size()

Out[42]: district
         上海周边       12
         嘉定         2900
         奉贤         1099
         宝山         3348
         崇明            2
         徐汇         3155
         普陀         2660
         杨浦         2081
         松江         2763
         浦东        12283
         虹口         1375
         金山           20
         长宁         2077
         闵行         6271
         闸北         1549
         青浦         1342
         静安          890
         黄浦         1355
         dtype: int64
```

Data description

- We obtain three more statistics for each district by applying the mean aggregation function for each numerical variables.
- What conclusion can we get for each of the means?
- Recall that to rent_type, we define 1 for 合租 and 0 for 整租.

```
In [67]: df_use.groupby(by='district').mean().reset_index()
```

Out[67]:

	district	area	price_per_area	rent_type
0	上海周边	111.333333	47.671525	0.166667
1	嘉定	77.673103	62.892753	0.100345
2	奉贤	87.767061	31.523732	0.006369
3	宝山	74.882616	68.835521	0.116189
4	崇明	103.500000	24.175610	0.000000
5	徐汇	75.036767	114.273836	0.043740
6	普陀	74.146617	94.756379	0.057143
7	杨浦	71.981259	99.239025	0.058626
8	松江	98.988780	64.142876	0.132103
9	浦东	86.412196	86.204175	0.081169
10	虹口	72.643636	106.209820	0.085091
11	金山	72.550000	161.646349	0.000000
12	长宁	84.701974	124.959281	0.028406
13	闵行	86.266784	75.251716	0.083400
14	闸北	68.352485	103.241463	0.078115
15	青浦	124.092399	77.410788	0.000000
16	静安	89.450562	152.795272	0.021348
17	黄浦	86.926937	143.029715	0.051661

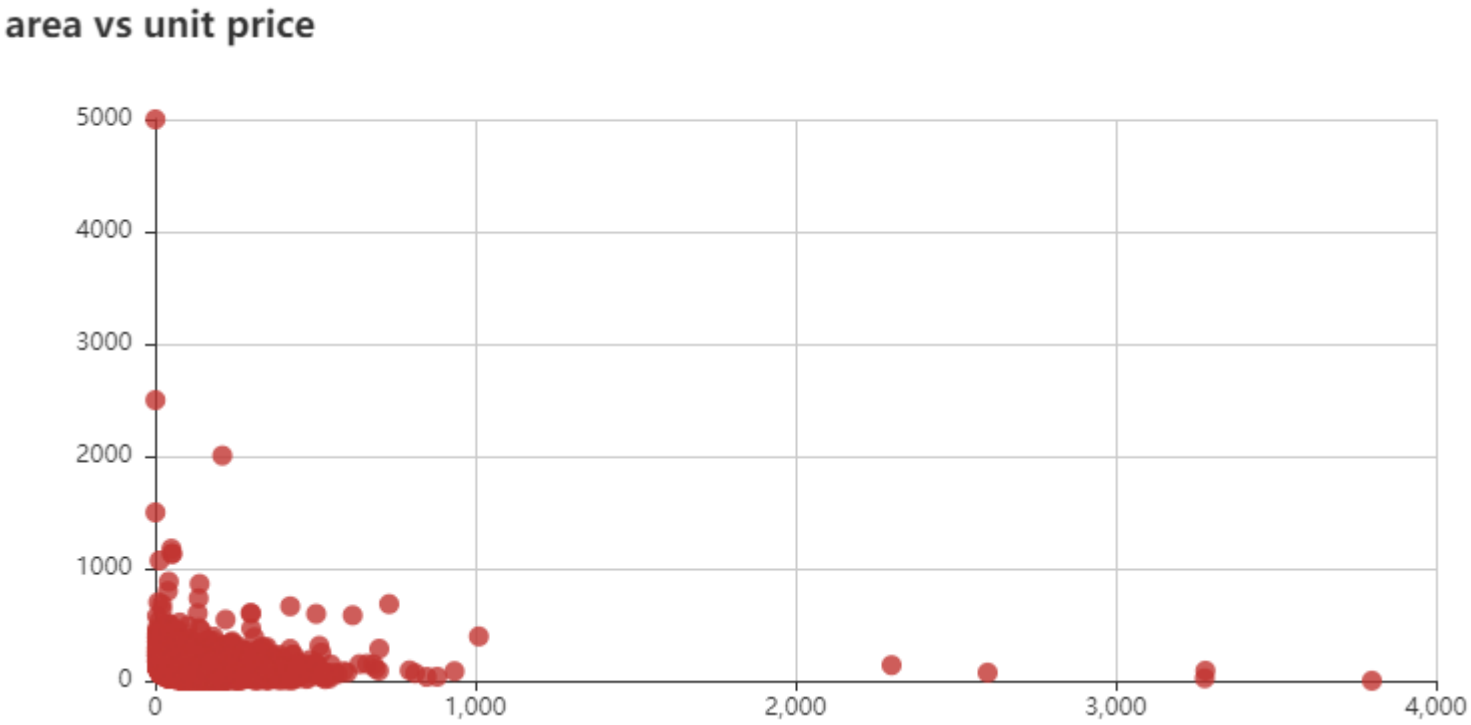
Visualization

- Pyecharts: 30+ kinds of charts and maps of hundreds of regions supported.
- See the following link for more details.
  - https://github.com/pyecharts/pyecharts

https://pyecharts.org/#/?id=pyecharts

Visualization

- scatter plot: area vs unit price.
  - from pyecharts import Scatter
  - v1 = df_use['area']
  - v2 = df_use['price_per_area']
  - scatter = Scatter("area vs unit price")
  - scatter.add("", v1, v2)
  - scatter.render()



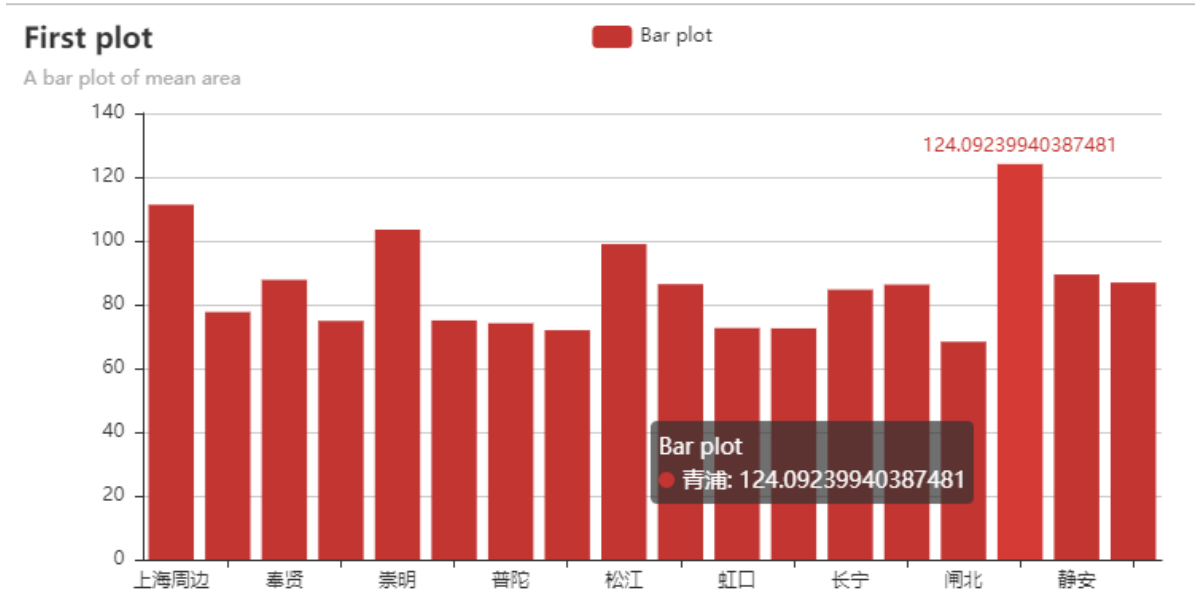
Visualization

- Scatter plot is a usual way to plot raw data.
- What else can you plot?
- Visualize some aggregation results in various ways.

Visualization

- District vs mean area.

- df_mean = df_use.groupby(by='district').mean().reset_index()
- from pyecharts import Bar
- bar = Bar('First plot','A bar plot of mean area')
- kwargs = dict(
- name = 'Bar plot',
- x_axis = df_mean['district'],
- y_axis = df_mean['area']
-)
- bar.add(**kwargs)
- bar.render()
- With similar code, you can generate bar plots for the other two variables.



https://pyecharts.org/#/?id=pyecharts

Visualization

- Word cloud is popular and size of words is controlled by area in this case.
- from pyecharts import WordCloud
- name_list = df_mean['district']
- value_list = df_mean['area']
- wordcloud = WordCloud(width=800, height=500)
- wordcloud.add("", name_list, value_list, word_size_range=[20, 100])
- wordcloud.render()

Visualization

- What is more?
- You may show data with Shanghai map involved.
- Go to https://pyecharts.org/#/zh-cn/intro and give a try.

Correlation

- Measure relationship between area and unit price
- **Correlation** refers to the degree of relationship (or dependency) between two variables.
- Linear correlation refers to straight-line relationships between two variables.
- A correlation can range between -1 (perfect negative relationship) and +1 (perfect positive relationship), with 0 indicating no straight-line relationship.

https://en.wikiversity.org/wiki/Correlation

Pearson correlation

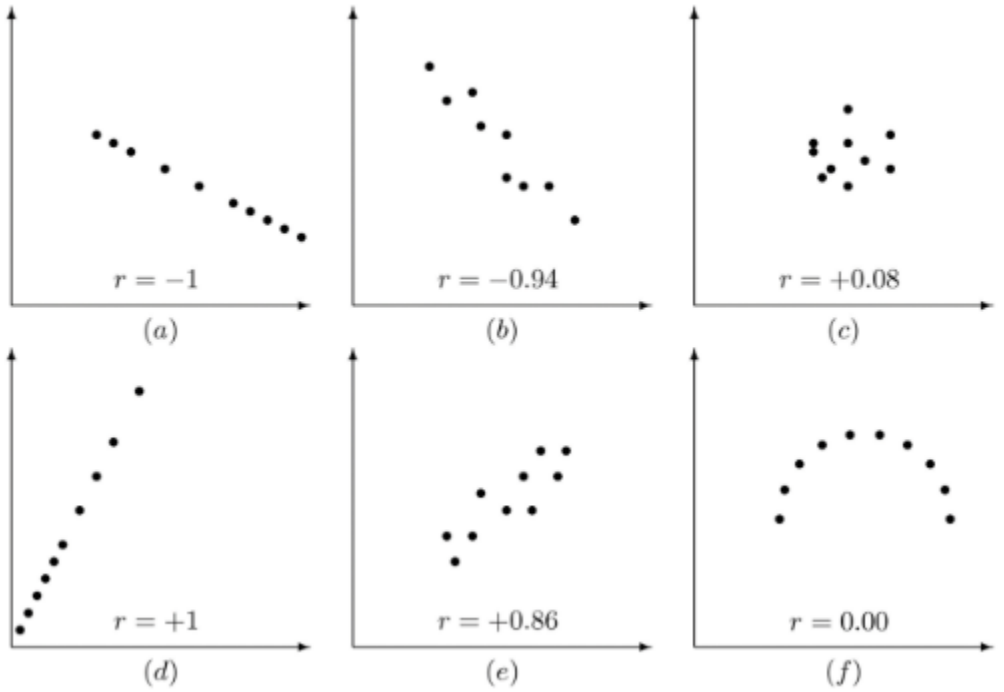
$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where:

- n is sample size
- x_i, y_i are the individual sample points indexed with *i*
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

Correlation

- Sub-figure (a) and (d) show perfect linear correlation.
- Sub-figure (f) shows zero correlation even though the pattern is obvious.



https://saylordotorg.github.io/text_introductory-statistics/s14-02-the-linear-correlation-coeffic.html

Correlation

- **NumPy**
  - N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

http://www.numpy.org/

Correlation

- With **NumPy** function *corrcoef*, we can obtain the correlation between area and price_per_area.
- import numpy as np
- x = df_use['area'].values
- y = df_use['price_per_area'].values
- np.corrcoef(x,y)
- The result is called

Correlation matrix.

```
In  [87]: import numpy as np

In  [99]: x = df_use['area'].values
          y =  df_use['price_per_area'].values
          np.corrcoef(x,y)

Out[99]: array([[ 1.        , -0.16117171],
                [-0.16117171,  1.        ]])
```

Linear regression

- In statistics, **linear regression** is a linear approach to modelling the *relationship* between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).
- In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data.
- used extensively in practical applications.

https://en.wikipedia.org/wiki/Linear_regression

Linear regression

- **scikit-learn -** *Machine Learning in Python*
  - Simple and efficient tools for data mining and data analysis
  - Accessible to everybody, and reusable in various contexts
  - Built on NumPy, SciPy, and matplotlib
  - Open source, commercially usable - BSD license

https://scikit-learn.org/stable/

Linear regression

https://scikit-learn.org/stable/

Linear regression

- analyze the relationship between area and price_per_area using *linear_model* in **sklearn**.
    - import matplotlib.pyplot as plt
    - import numpy as np
    - from sklearn import linear_model
    - regr = linear_model.LinearRegression()
    - x = df_use['area'].values.reshape(-1, 1)
    - y = df_use['price_per_area'].values_model
    - regr.fit(x, y)

Linear regression

- The linear model price_per_area vs area is built.
- Price_per_area = 100.19-0.15*area

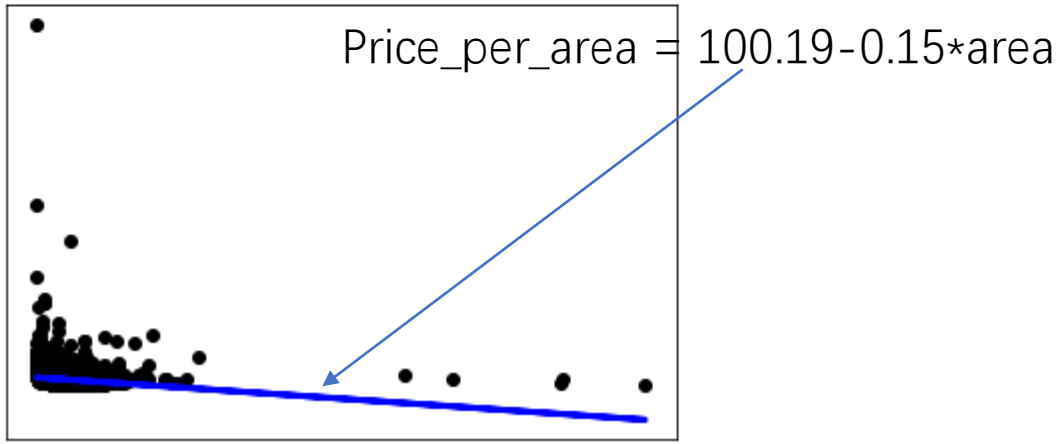
```
In [119]: print('Coefficients: \n', regr.coef_,'\n Intercept: \n',regr.intercept_)

Coefficients:
 [-0.15454709]
 Intercept:
 100.19471649716381
```

- This equation tells us that price_per_area gets 0.15 Yuan cheaper as area gets 1 square meter larger.

Linear regression

- We can plot the model.
- We should mention that analysis here is not rigorous at all. The statistical knowledge needed is out of the scope of the course.

```
In [120]: y_pred = regr.predict(x)

In [121]: # Plot outputs
          plt.scatter(x, y,  color='black')
          plt.plot(x, y_pred, color='blue', linewidth=3)

          plt.xticks(())
          plt.yticks(())

          plt.show()
```



Thanks!