

2023 Python程序设计大作业：租房数据分析

1. 数据抓取

要求抓取链家官网北上广深4个一线城市，再加上一个离你家乡最近的一个非一线城市/或者你最感兴趣的一个城市的租房数据。

页面分析

通过谷歌浏览器提供的开发者工具，能找到租房信息的网页html中有我们需要爬取的信息,并且都位于一个div元素下 `<div class="content__list"></div>`

所以我们可以通过这种XPath路径进行爬取，`response.xpath("//*[@id='content']/div[1]/div[1]/*")`

然后通过静态和动态调试，能够发现我们正确获取的所需要的信息，其中一个模块如下

```
<div
class="content__list--item"
data-el="listItem"
data-house_code="BJ1816009503133401088"
data-brand_code="200301001000"
data-ad_code="0"
data-bid_version=""
data-c_type="1"
data-position="0"
data-total="66377"
data-fb_expo_id="786707837408985088"
data-t="default"
data-strategy_id=""
data-click_position="0"
data-ad_type="0"
data-distribution_type="203500000001"
data-event_id="21333"
data-event_action="click_position=0"
data-event_position="click_position"
data-event_send="no"
data-rank_score="-1"
data-operation_score="0"
>
<a
class="content__list--item--aside" target="_blank"
href="/zufang/BJ1816009503133401088.html"
title="整租·乐想汇 1房间 南">
  
```

```

<!-- 是否展示vr图片 -->
    <!-- 是否展示省心租图片 -->
    <!-- 广告标签 -->
    </a>
<div class="content__list--item--main">
<p class="content__list--item--title">
    <a class="twoline" target="_blank" href="/zufang/BJ1816009503133401088.html">
        整租·乐想汇 1房间 南    </a>
    </p>
<p class="content__list--item--des">
    <a target="_blank" href="/zufang/chaoyang/">朝阳</a>-<a
href="/zufang/beiyuan2/" target="_blank">北苑</a>-<a title="乐想汇"
href="/zufang/c1111046126379/" target="_blank">乐想汇</a>
    <i>/</i>
    68.00m2
    <i>/</i>南    <i>/</i>
    1房间1卫    <span class="hide">
    <i>/</i>
    低楼层    (18层 )
    </span>
</p>
<p class="content__list--item--bottom oneline">
    <i class="content__item_tag--gov_certification">官方核验</i>
    <i class="content__item_tag--is_subway_house">近地铁</i>
    <i class="content__item_tag--central_heating">集中供暖</i>
    <i class="content__item_tag--is_key">随时看房</i>
    </p>
<p class="content__list--item--brand oneline">
    <span class="brand">
        链家    </span>
    <span class="content__list--item--time oneline">2天前维护</span>
</p>
    <span class="content__list--item-price"><em>5100</em> 元/月</span>
</div>
</div>

```

通过谷歌浏览器开发者工具提供的XPath复制和查找功能，我们可以找到我们所需要的信息的XPath路径。

这样创建几个对应的item，就可以将数据存储在对应的item中，然后再通过pipelines.py中的代码，将数据存储在对应的csv文件中。

代码编写

1. 爬虫代码

使用之前的python爬虫框架，重新编写了5个爬虫（分别对应北京，上海，广州，深圳，以及郑州5个城市），并且将爬虫的代码进行了封装，使得代码更加简洁，易于阅读。

下面以bj代码为例，进行说明：

```
import scrapy

from test1.items import Test1Item # 从items.py中引入MyItem对象

class LianjiaSpider(scrapy.Spider):
    name = 'bj'
    allowed_domains = ['bj.lianjia.com']
    # https://bj.lianjia.com/zufang/pg2/
    base_url = 'https://bj.lianjia.com/zufang/'

    maxpage = 200
    # 最大爬取页数

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.download_delay = 1
        with open("bj.txt", "r") as f:
            # 读取上次爬取的页数
            self.begin_index = int(f.read())
            self.page_index = self.begin_index
            self.start_urls = [self.base_url + "pg" + str(self.begin_index)]
            print(self.begin_index)
            f.close()
        with open("bj.txt", "w") as f:
            # 更新爬取的页数
            f.write(str(self.begin_index + self.maxpage))
            f.close()

    def parse(self, response, **kwargs):
        for each in response.xpath("//*[@id='content']/div[1]/div[1]/*"):
            item = Test1Item()
            item['price'] = each.xpath("div/span/em/text()").extract_first()
            if '-' in item['price']:
                temp = item['price'].split("-")
                item['price'] = (float(temp[0]) + float(temp[1])) / 2
            temp = each.xpath("div/p[1]/a/text()").extract_first()
            # 从temp中提取数据
            temp = temp.split(" ")
            item['name'] = temp[0]
            item["house_type"] = ""
            item["area"] = ""
            item["direct"] = ""
            item["name_chinese"] = "北京"
            for i in temp:
                if '室' in i or '厅' in i or '卫' in i:
                    item['house_type'] = i
                if '东' in i or '南' in i or '西' in i or '北' in i:
                    if len(i) < 5:
                        item['direct'] = i
            temp = each.xpath("div/p[2]/text()").extract()
            for i in temp:
```

```

        if 'm' in i:
            i = i.replace("m", "").replace(" ", "").replace("\n", "")
            if "-" in i:
                temp = i.split("-")
                item['area'] = (float(temp[0]) + float(temp[1])) / 2
            else:
                item['area'] = i
            item["block"] = each.xpath("div/p[2]/a[2]/text()").extract_first()
            print(item["block"])
            if item['price'] and item['name'] and item['house_type'] and
            item['area'] and item['direct'] and item["block"]:
                yield item
            self.page_index += 1
            if self.page_index > self.maxpage+self.begin_index-1:
                return
            url=self.base_url+"pg"+str(self.page_index)

            print(url)

            yield scrapy.Request(url, callback=self.parse)
            # 递归爬取下一页

```

通过一个bj.txt文件，记录上次爬取的页数，然后通过递归爬取下一页的方式，实现了爬取多页的功能。这样在多次爬取时只需要修改爬虫的名字和pipe中输出的文件名就能实现多个城市的爬取。

2. 数据处理代码

在pipelines.py中，我们对数据进行了处理，将数据存储到对应的csv文件中。

```

import json
import csv
class Test1Pipeline:

    def open_spider(self, spider):
        try:
            self.file = open('zz.csv', "a", encoding="utf-8-sig", newline='')
            self.writer = csv.writer(self.file)
        except Exception as err:
            print(err)

    def process_item(self, item, spider):
        list_item = [item['name_chinese'], item['block'], item['house_type'],
            item['direct'], item['area'], item['price']]
        self.writer.writerow(list_item)
        return item

    def close_spider(self, spider):
        self.file.close() #关闭文件

```

3. 运行代码

在main.py中，我们通过调用execute函数，运行爬虫代码。

```
from scrapy import cmdline
cmdline.execute("scrapy crawl zz".split())
```

2. 数据获取

应获取每个城市的全部租房数据（一线城市的数据量应该在万的数量级）。

运行代码

为防止爬取大量数据时，被网站封禁，我们将爬取的数据分为多次爬取，每次爬取300-500页，然后将爬取的数据存储到对应的csv文件中。

发现问题

本来以为爬取的数据量不会很大，但是在爬取的过程中，发现了一些问题：

开始通过chrome浏览器发现<https://bj.lianjia.com/zufang/pg1233/>这个网页是存在的，但是通过爬虫爬取时，发现这个网页中的数据很不友好，实际上是重复的，这就很让人头疼。

想到的解决方法是重写一下parse函数，对每个区进行遍历，如果数据再不够的话只能进行进一步细分了

同样的，以bj.py为例，进行说明：

```
import scrapy

from test1.items import Test1Item # 从items.py中引入MyItem对象

class LianjiaSpider(scrapy.Spider):
    name = 'bj'
    allowed_domains = ['bj.lianjia.com']
    zone_list_chinese=["东城","西城","朝阳","海淀","丰台","石景山","通州","昌平","大兴",
"亦庄开发区","顺义","房山","门头沟","平谷","怀柔","密云","延庆"]

    zone_list=
["dongcheng","xicheng","chaoyang","haidian","fengtai","shijingshan","tongzhou","ch
angping","daxing","yizhuangkaifaq","shunyi","fangshan","mentougou","pinggu","huai
rou","miyun","yanqing"]
    # https://bj.lianjia.com/zufang/pg2/
    zone_list_number=
[100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100]
    base_url = 'https://bj.lianjia.com/zufang/'
    maxpage = 500
    def __init__(self, **kwargs):
        // 读取上次爬取的页数
        super().__init__(**kwargs)
        self.download_delay = 1
        with open("bj.txt", "r") as f:
```

```

temp=f.read().split("\n")
self.zone_index=int(temp[0])
self.page_index =int (temp[1])

self.start_urls = [self.base_url
+self.zone_list[self.zone_index]+"/pg" + str(self.page_index)]
print(self.zone_index,self.page_index)
f.close()
def parse(self, response, **kwargs):

for each in response.xpath("//*[@id='content']/div[1]/div[1]/*"):
    item = TestItem()
    item['price'] = each.xpath("div/span/em/text()").extract_first()
    if '-' in item['price']:
        temp = item['price'].split("-")
        item['price'] = (float(temp[0]) + float(temp[1])) / 2
    temp = each.xpath("div/p[1]/a/text()").extract_first()
    # 从temp中提取数据
    temp = temp.split(" ")
    item['name'] = temp[0]
    item["house_type"] = ""
    item["area"] = ""
    item["direct"] = ""
    item["name_chinese"] = "北京"
    for i in temp:
        if '室' in i or '厅' in i or '卫' in i:
            item['house_type'] = i
        if '东' in i or '南' in i or '西' in i or '北' in i:
            if len(i) < 5:
                item['direct'] = i
    temp = each.xpath("div/p[2]/text()").extract()
    for i in temp:
        if '㎡' in i:
            i = i.replace("㎡", "").replace(" ", "").replace("\n", "")
            if "-" in i:
                temp = i.split("-")
                item['area'] = (float(temp[0]) + float(temp[1])) / 2
            else:
                item['area'] = i
    item["block"]=each.xpath("div/p[2]/a[2]/text()").extract_first()
    print(item["block"])
    item["zone_name"] = self.zone_list_chinese[self.zone_index]
    if item['price'] and item['name'] and item['house_type'] and
item['area'] and item['direct'] and item["block"]:
        yield item
if self.maxpage <= 0:
    # 爬取完毕，这样写很丑陋，但是我写在析构函数里面不会执行不知道为什么，也许是
    框架本身就不甚严谨
    with open("bj.txt", "w") as f:
        f.write(str(self.zone_index))
        f.write("\n")
        f.write(str(self.page_index))
        f.close()
    return

```

```

// 通过判断是否有下一页，来决定是否进行下一页的爬取
if response.xpath("//*[@id='content']/div[1]/div[2]/@data-
totalpage").extract_first() :
    self.zone_list_number[self.zone_index]=int(response.xpath("//*
[@id='content']/div[1]/div[2]/@data-totalpage").extract_first())

print("self.zone_list_number[self.zone_index]",self.zone_list_number[self.zone_ind
ex])

self.page_index += 1
if self.zone_index < len(self.zone_list):
    if self.page_index <= self.zone_list_number[self.zone_index]:

url=self.base_url+self.zone_list[self.zone_index]+"/pg"+str(self.page_index)
    else:
        self.page_index = 1
        url = self.base_url + self.zone_list[self.zone_index]
        self.zone_index += 1
    else:
        with open("bj.txt", "w") as f:
            f.write(str(self.zone_index))
            f.write("\n")
            f.write(str(self.page_index))
            f.close()
        return
self.maxpage -= 1
yield scrapy.Request(url, callback=self.parse)

```

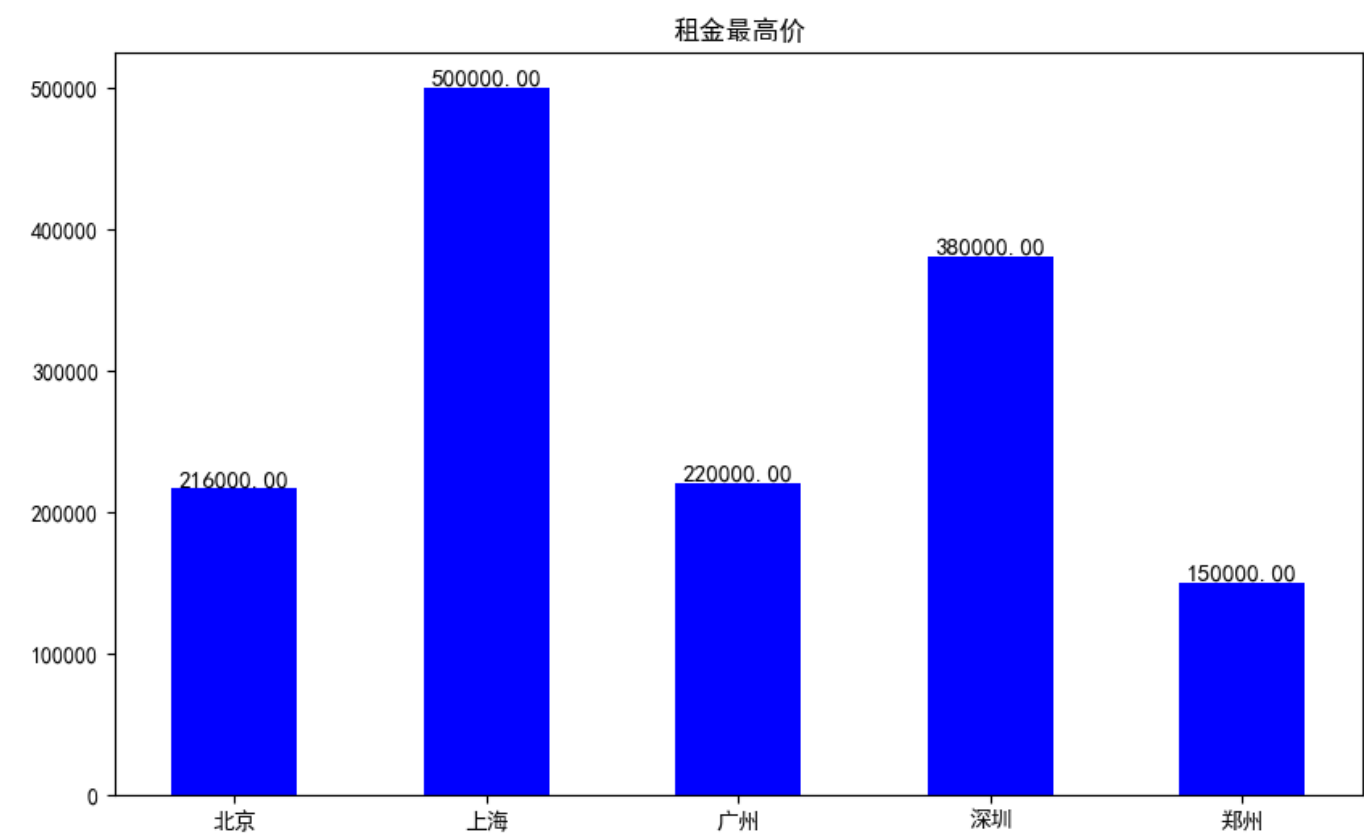
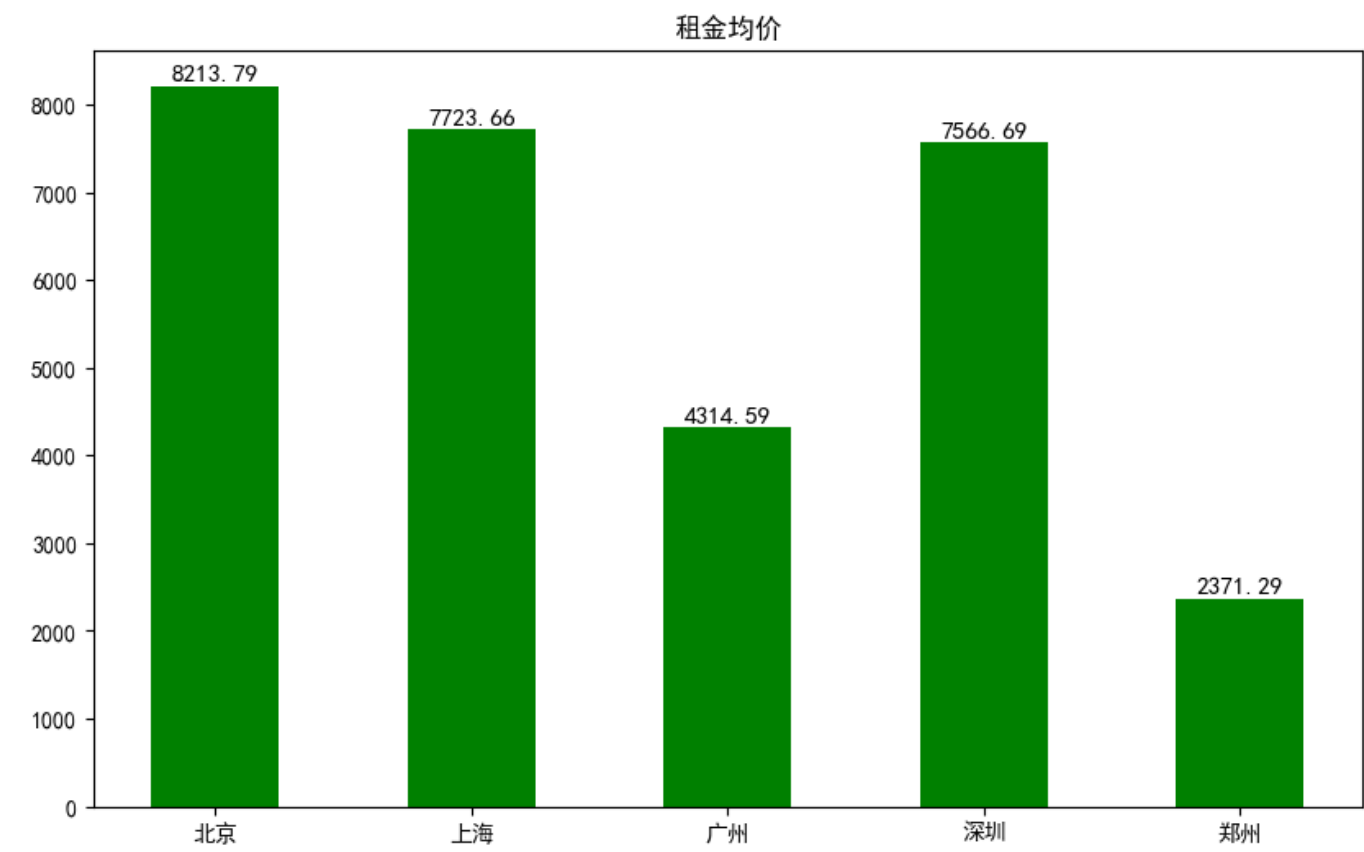
3. 总体房租情况比较

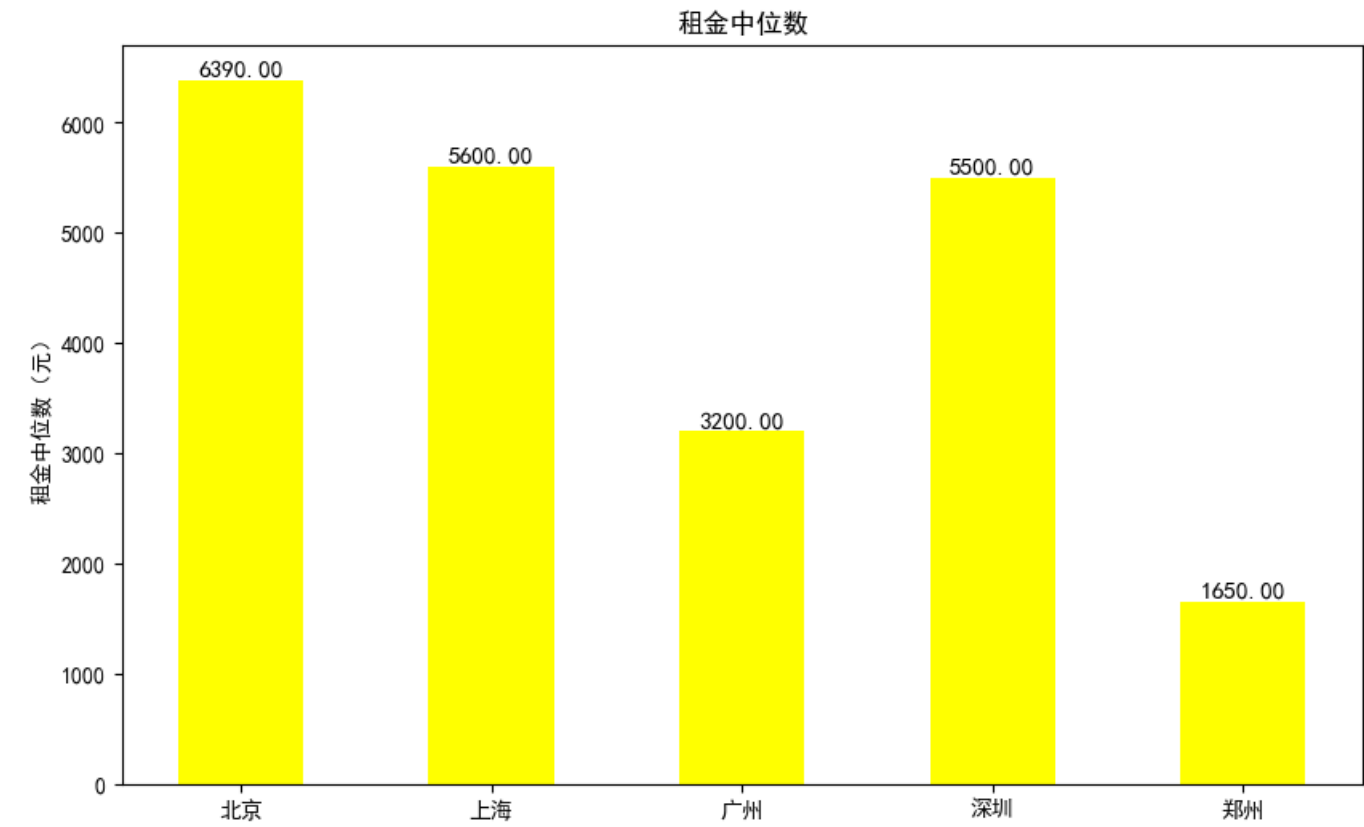
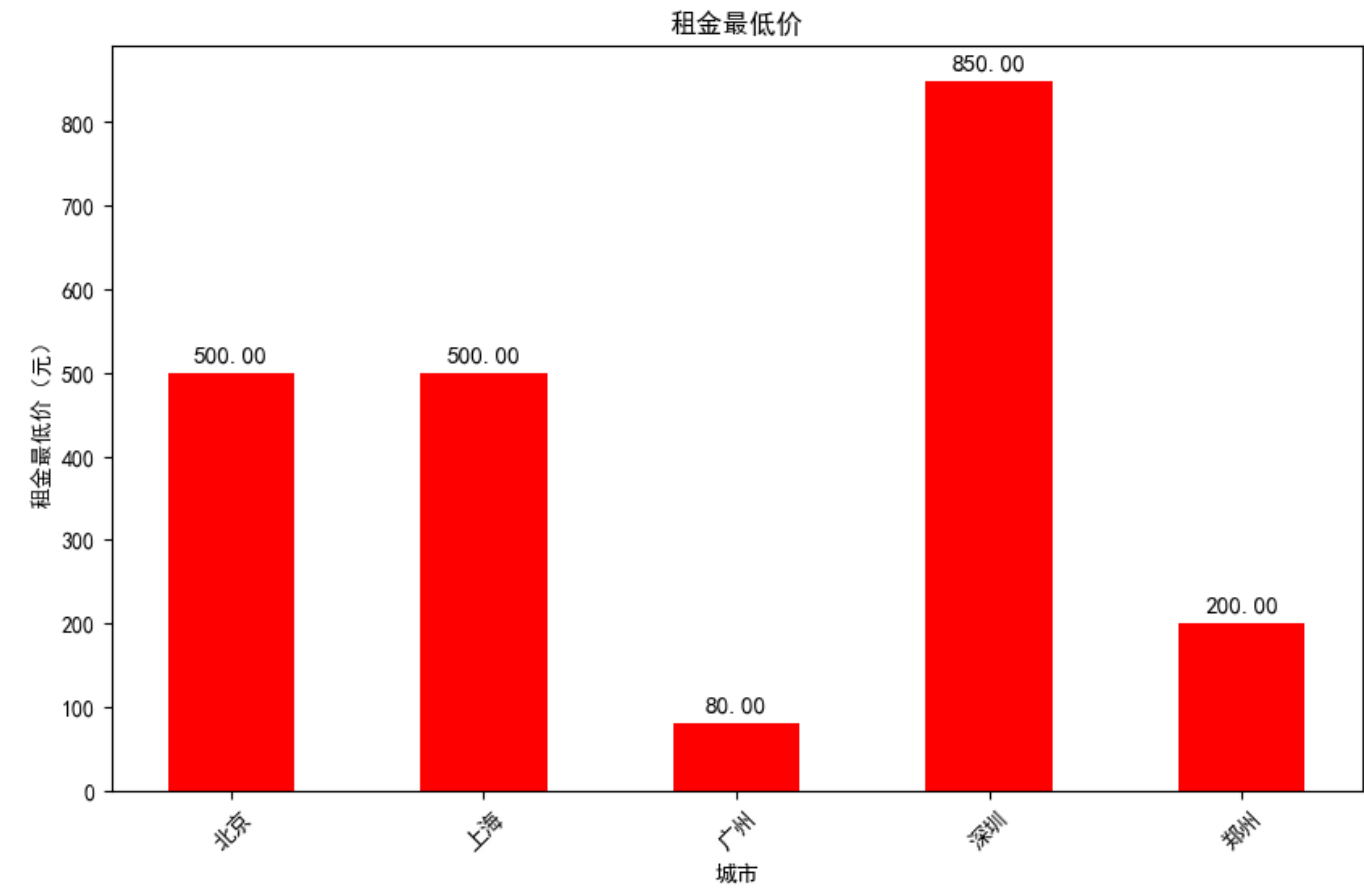
比较5个城市的总体房租情况，包含租金的均价、最高价、最低价、中位数等信息，单位面积租金（元/平米）的均价、最高价、最低价、中位数等信息。

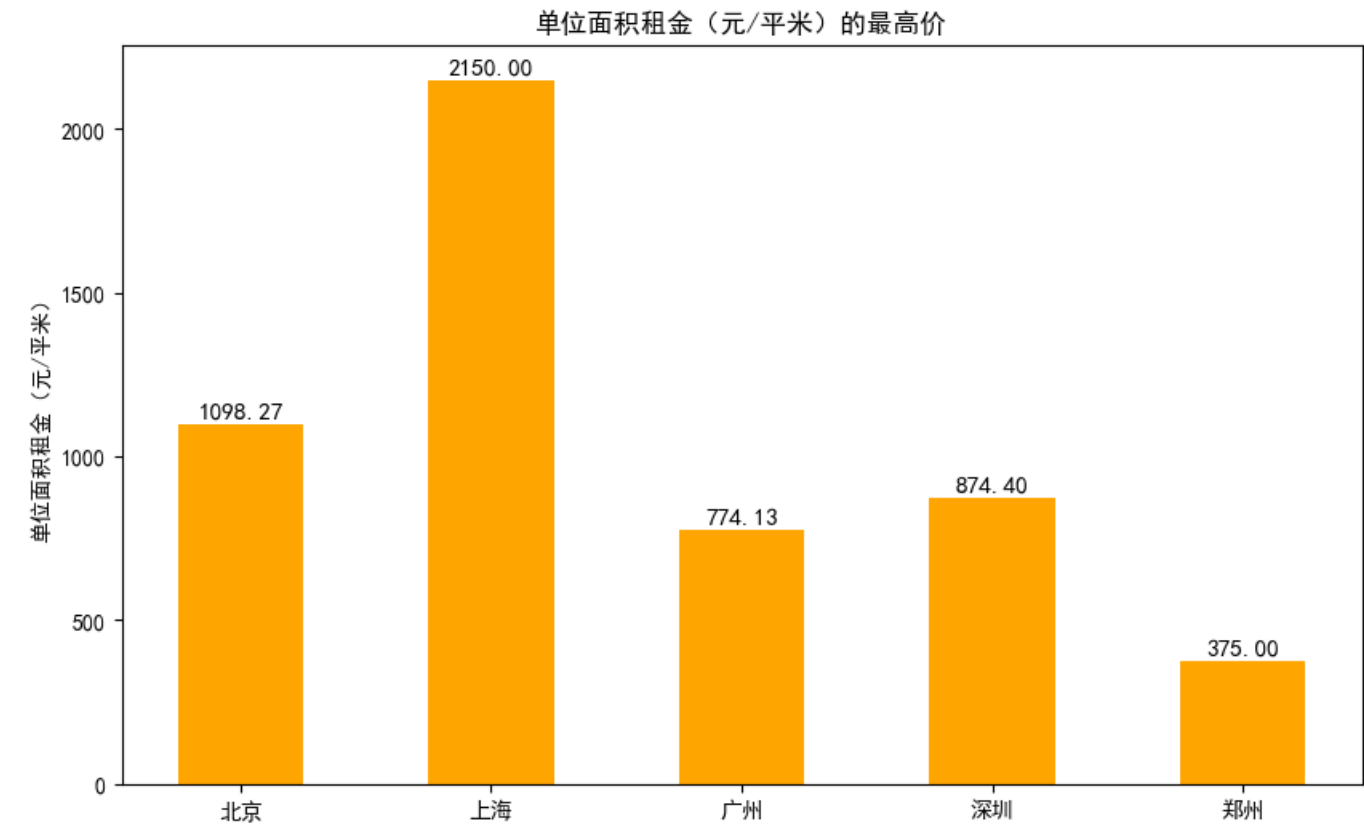
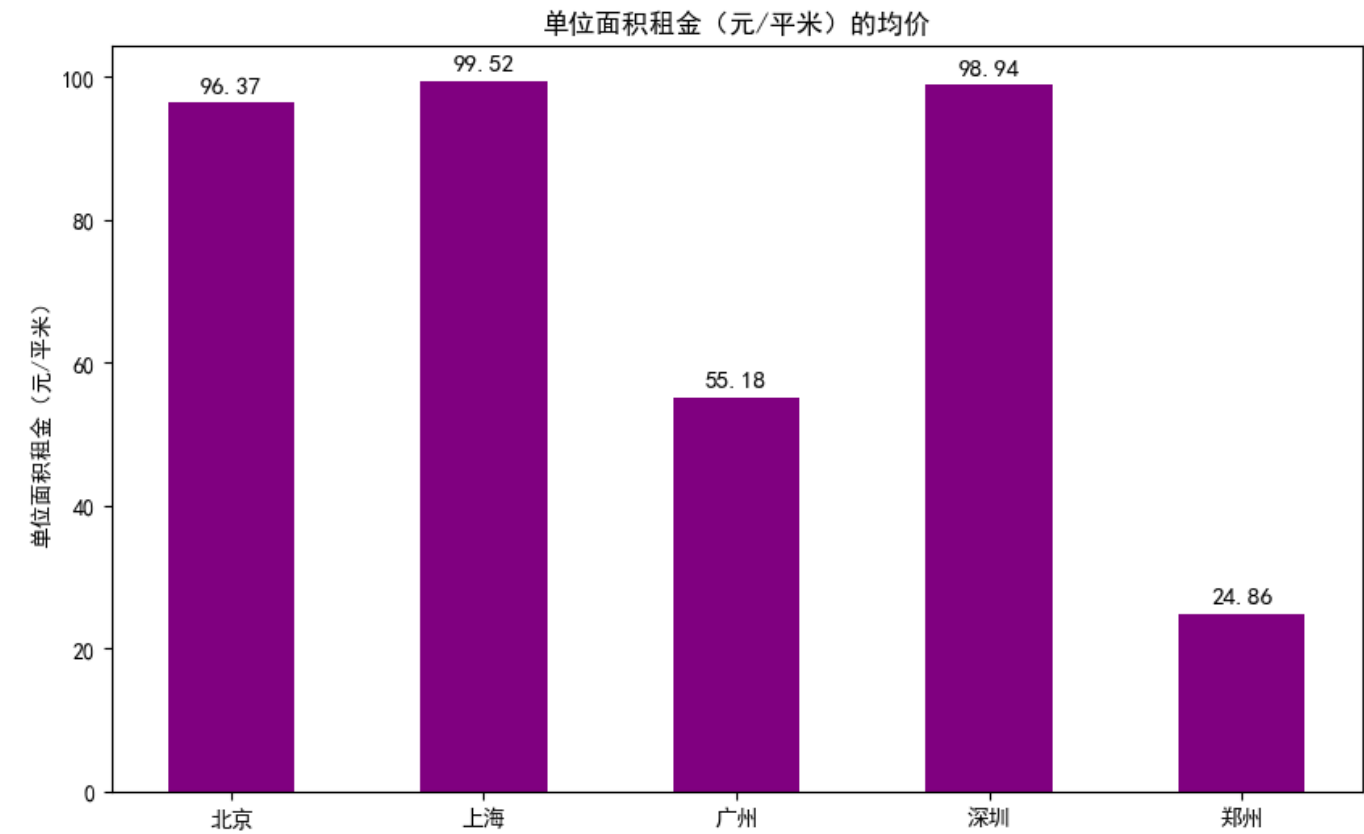
代码编写

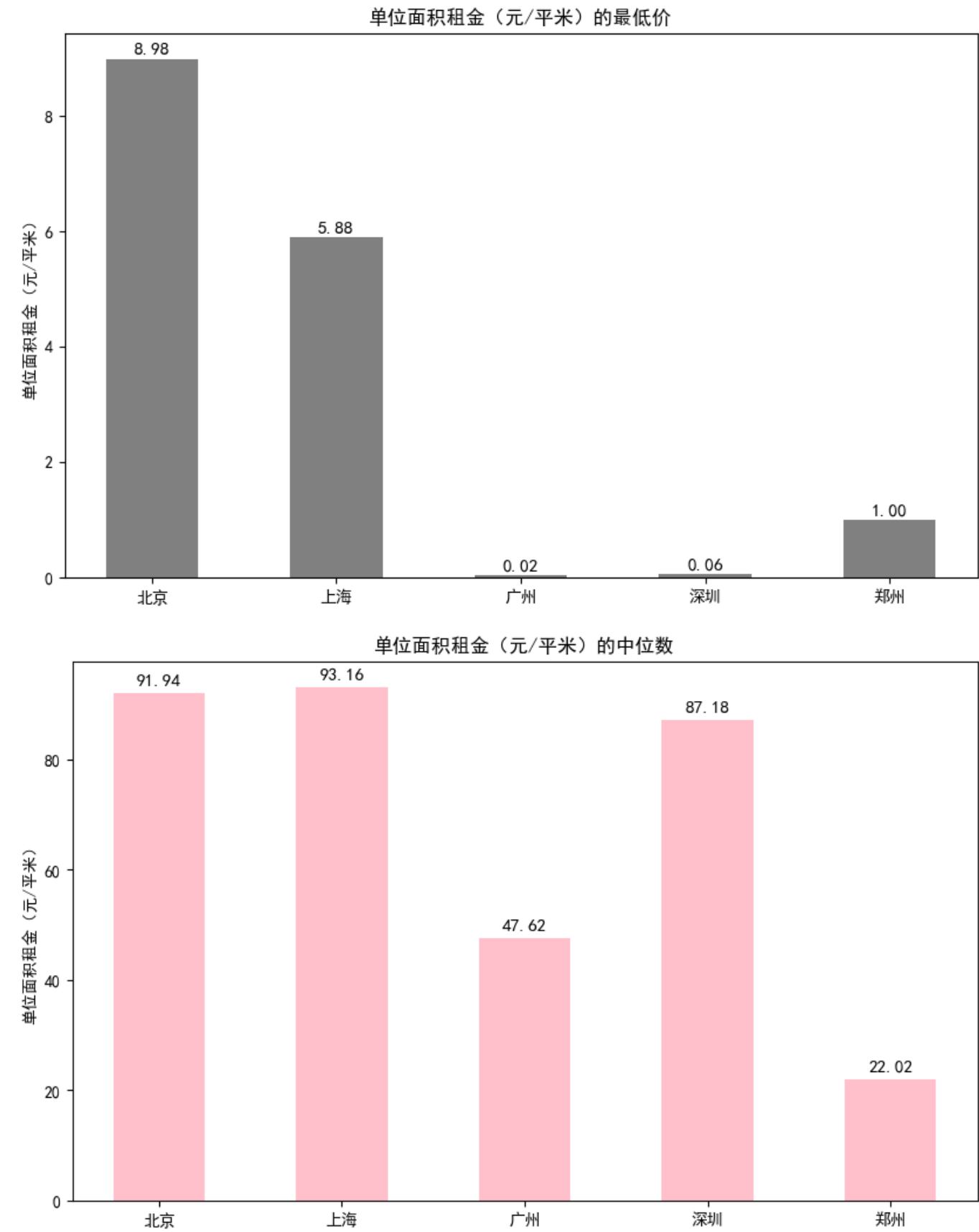
这里使用jupyter进行编写，因为能方便的进行输出，以及进行数据的可视化。看起来会舒服许许多多 不过缺点是有点长了，所以这里仅仅加入一个引用的代码[总体和户型](#)

结果展示









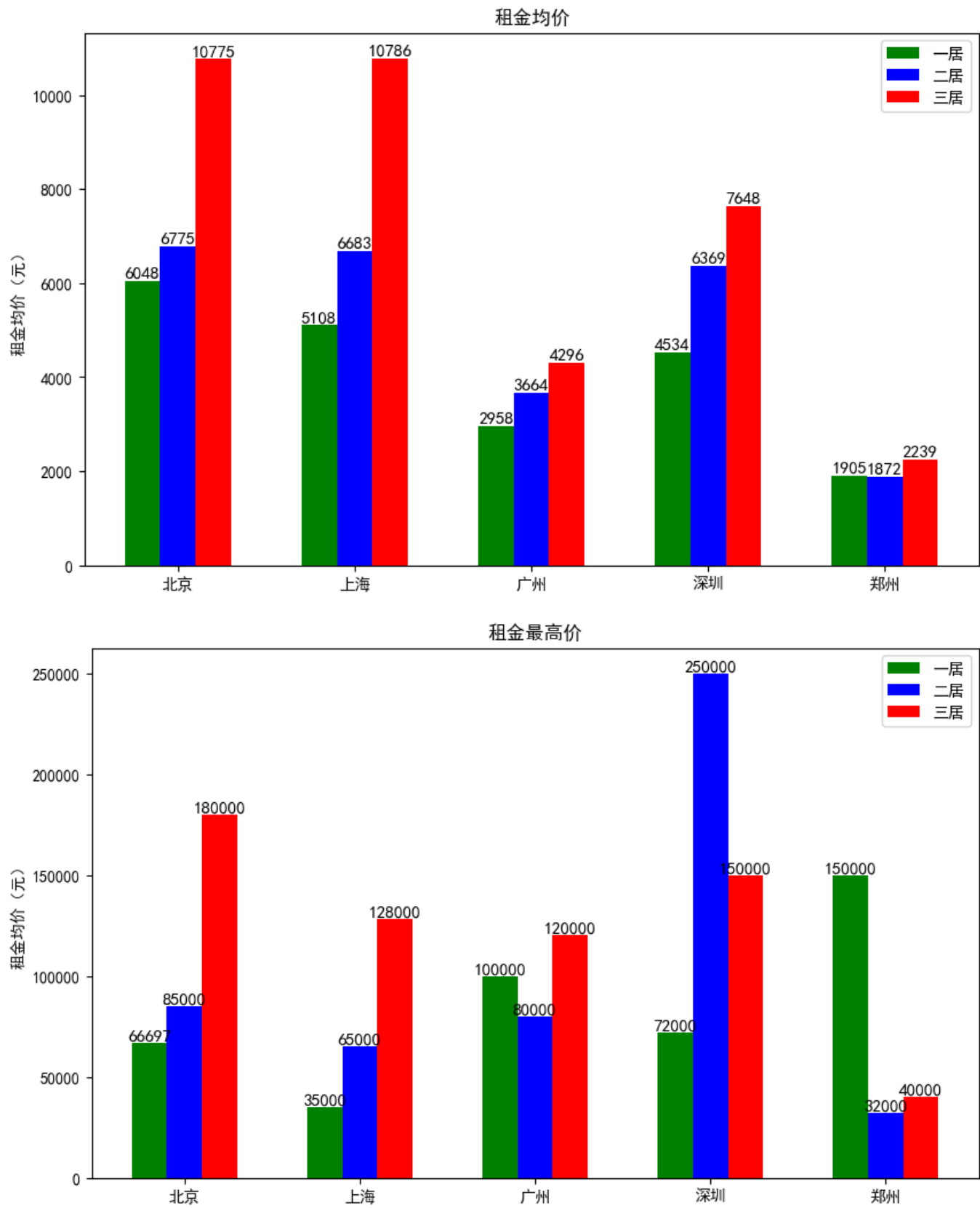
4. 不同户型比较

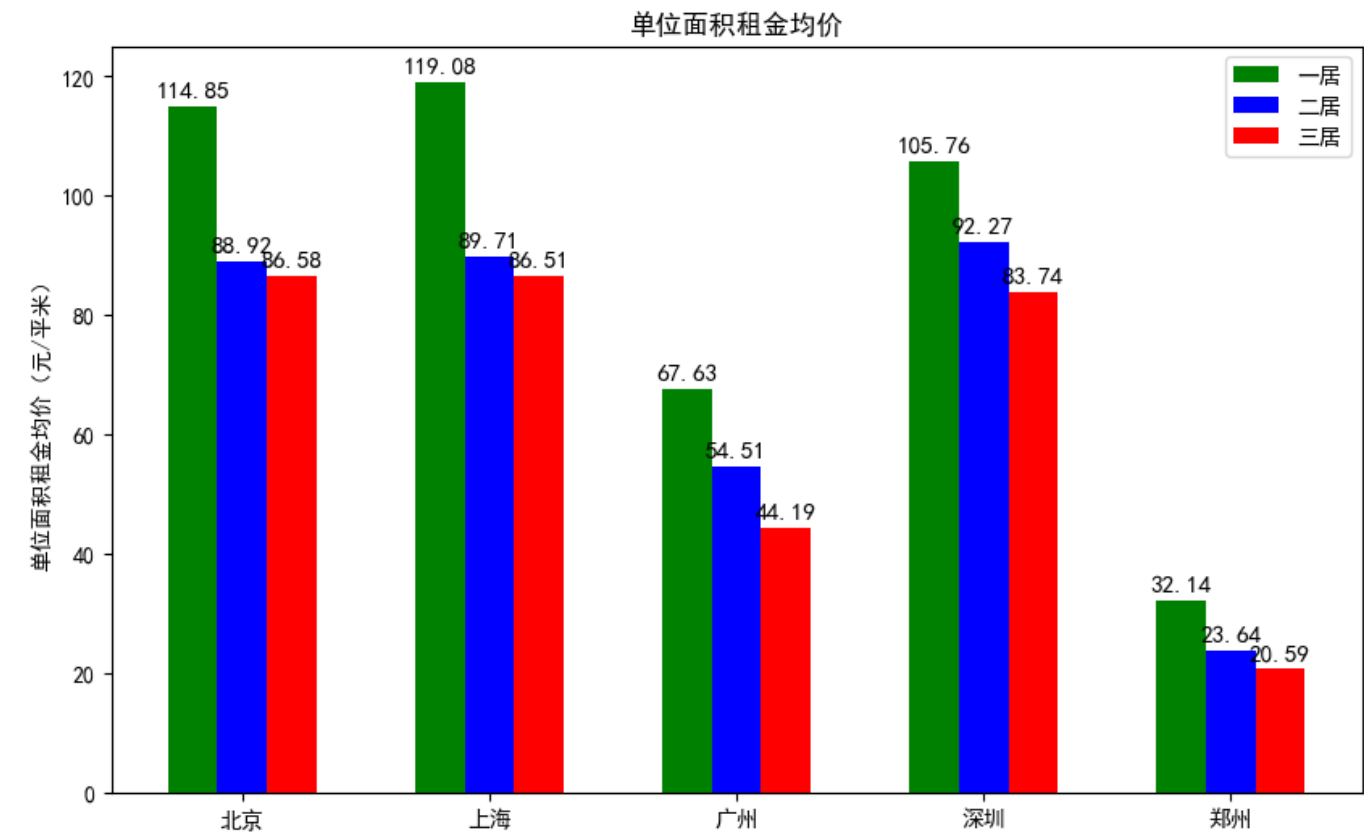
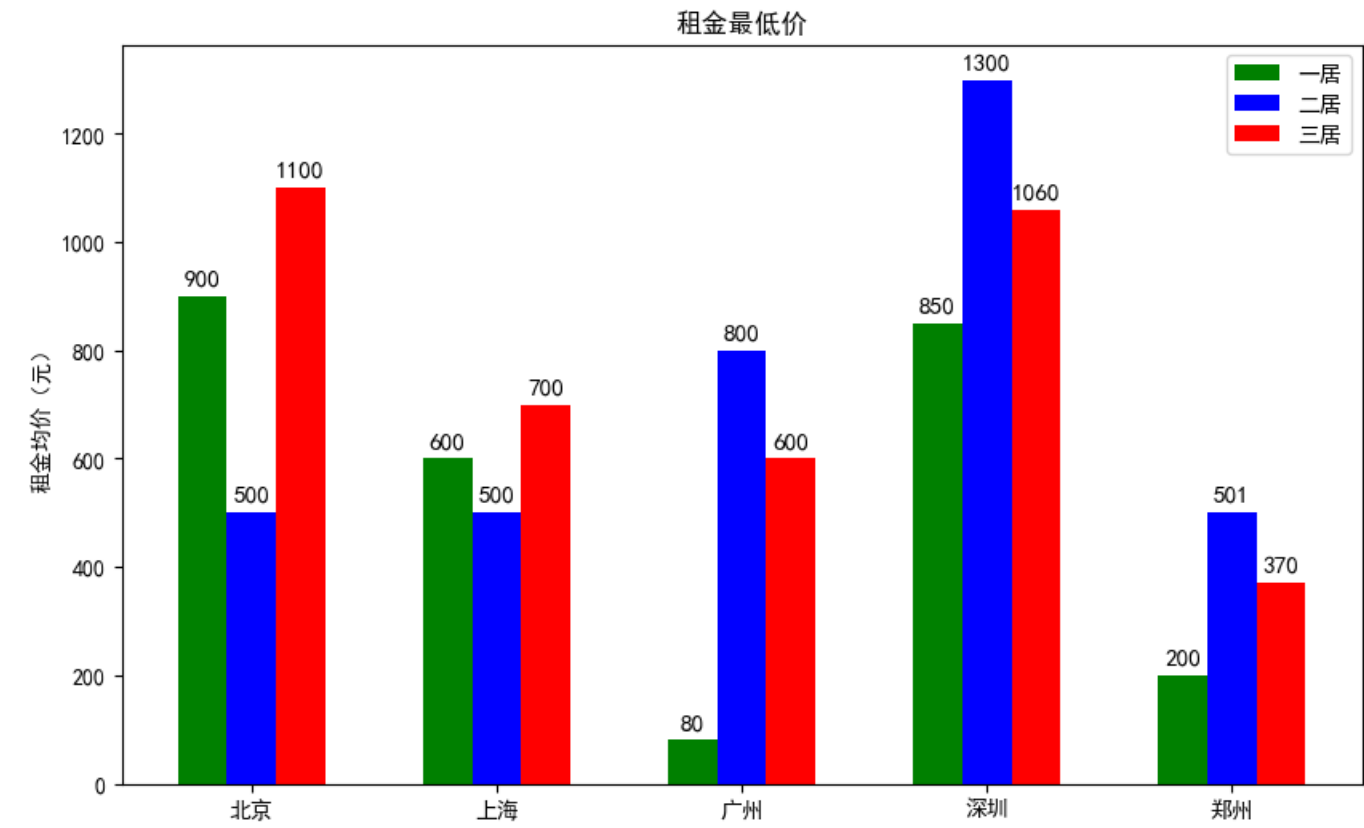
比较5个城市一居、二居、三居的情况，包含均价、最高价、最低价、中位数等信息，采用合适的图或表形式进行展示。

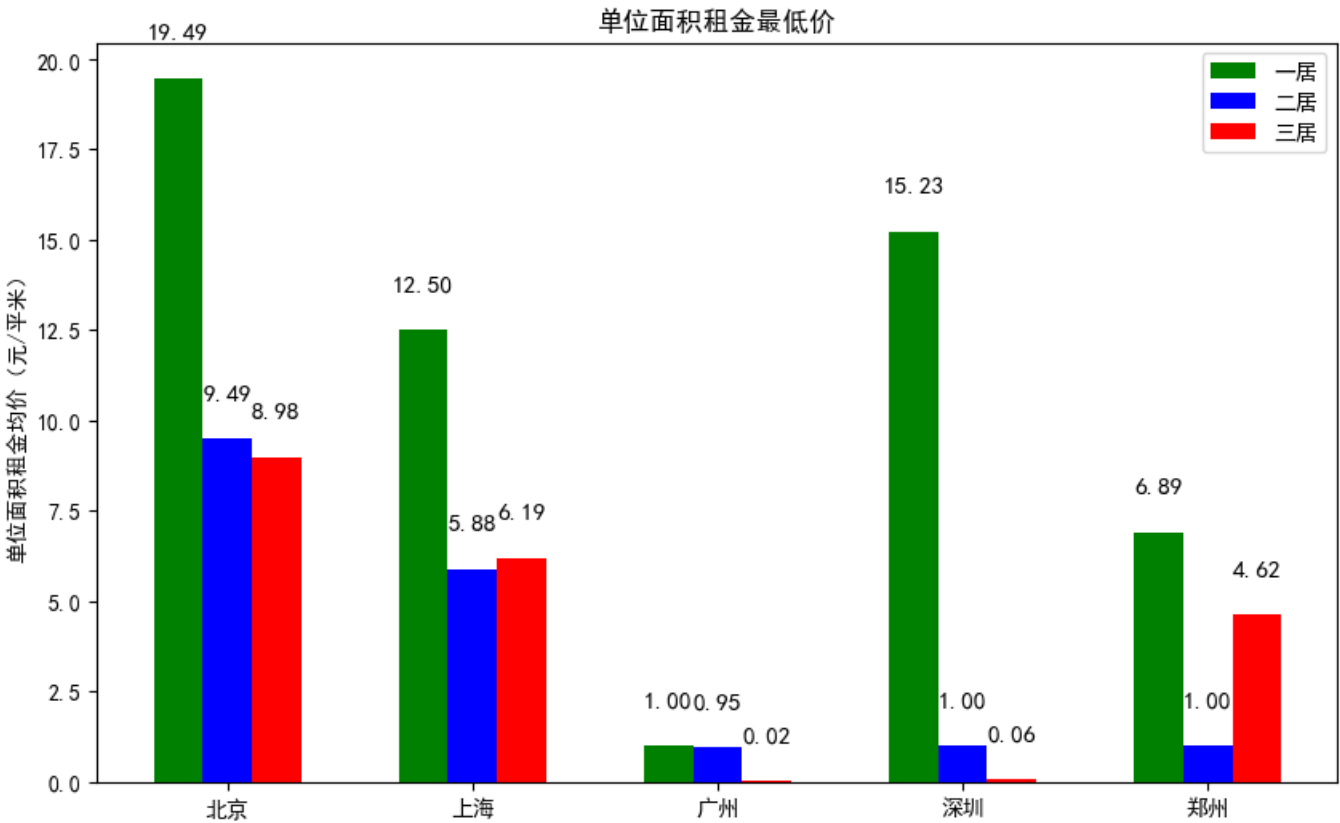
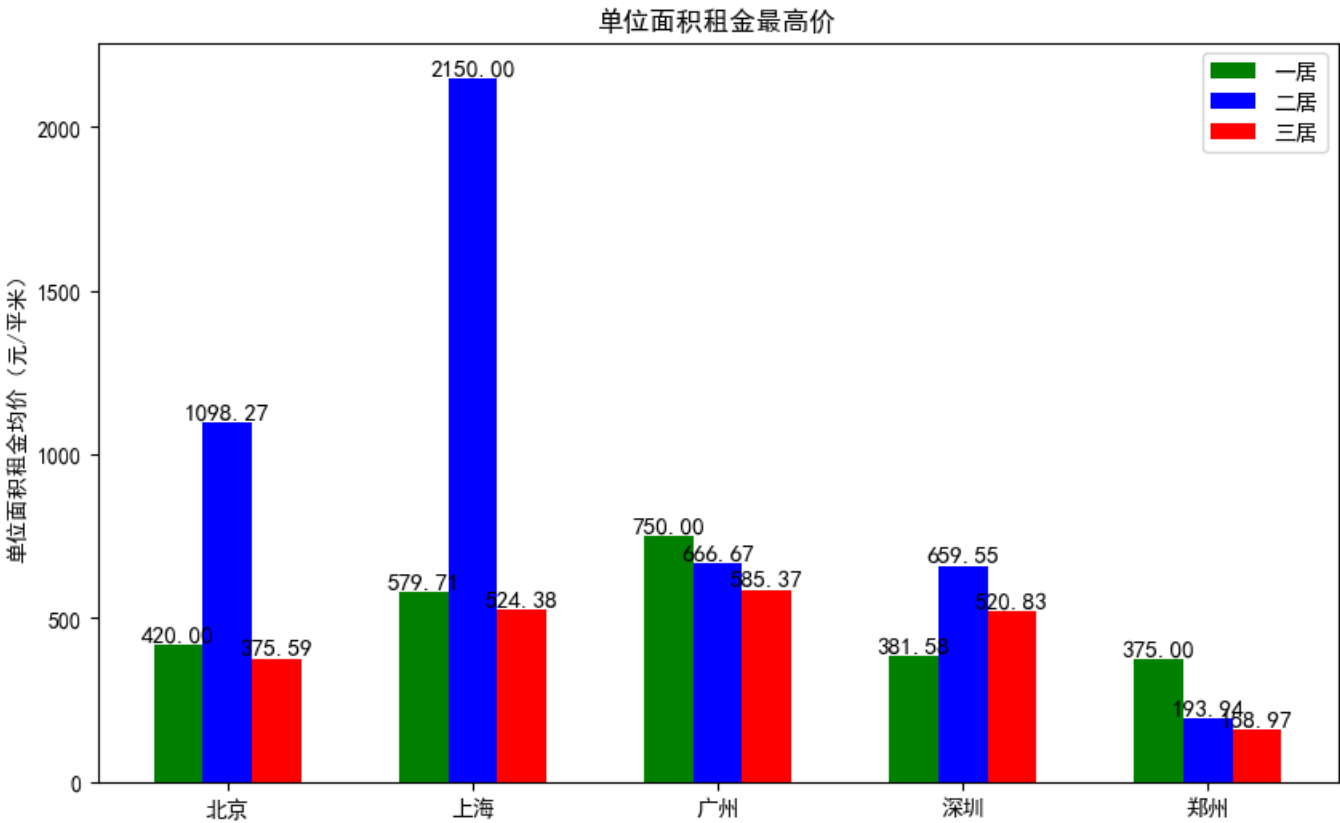
代码编写

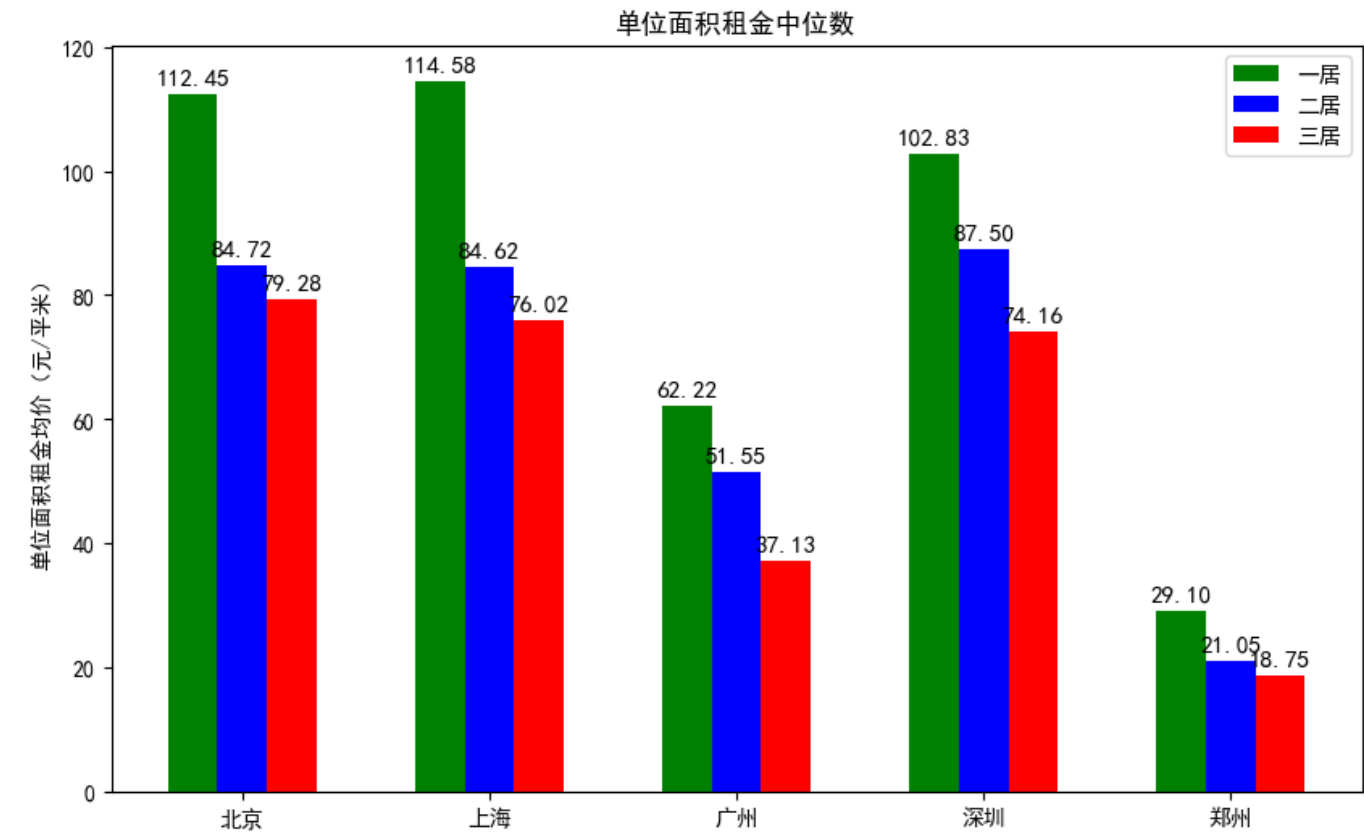
与上面的代码类似，这里仅仅加入一个引用的代码[总体和户型](#)

结果展示









5. 板块均价比较

计算和分析每个城市不同板块的均价情况，并采用合适的图或表形式进行展示。例如上图中的“海淀-四季青-五福玲珑居北区”，“四季青”即为板块名称。

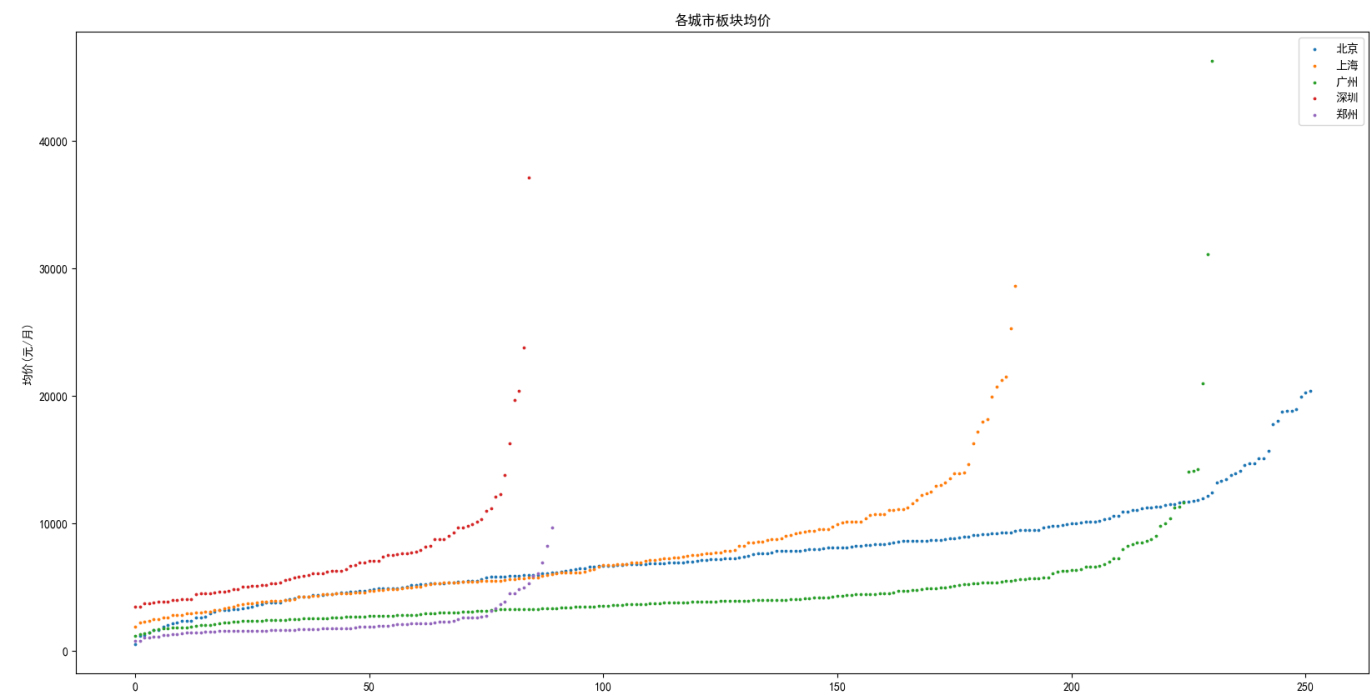
代码编写

与上面的代码类似, [板块均价](#)

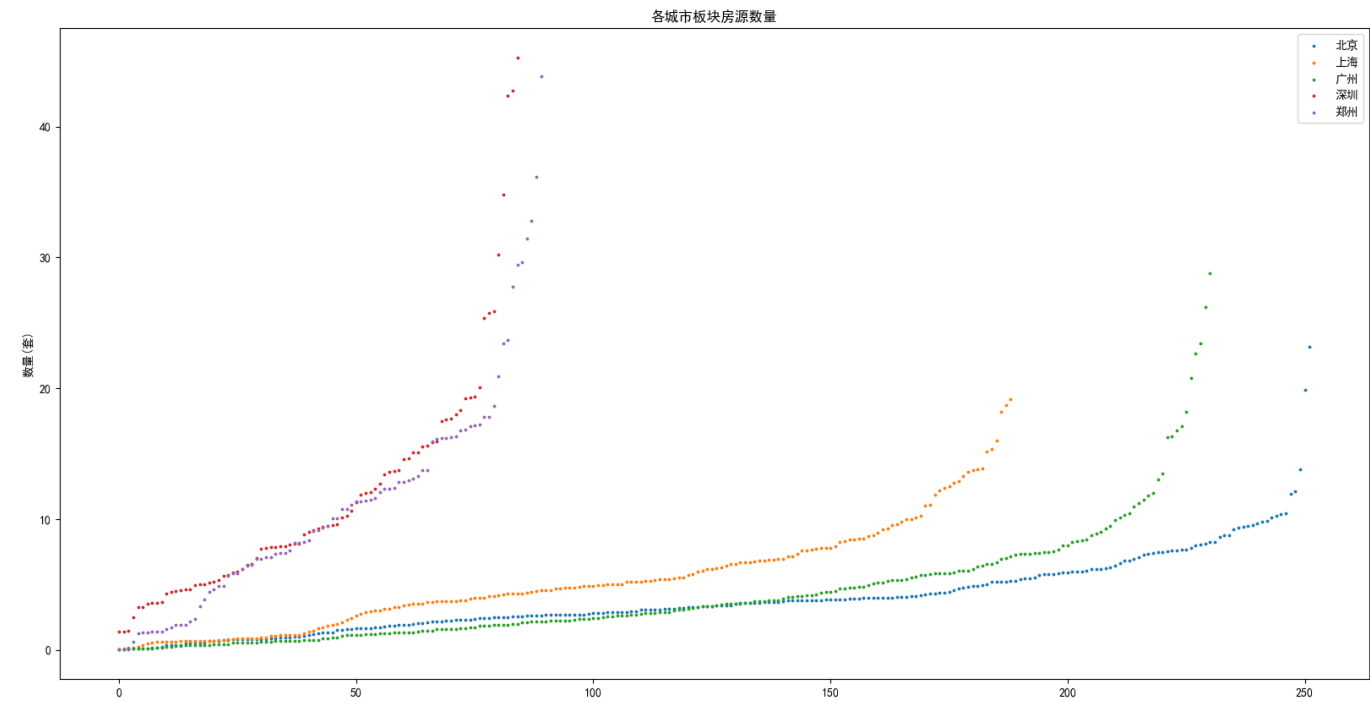
结果展示

[北京板块均价](#) [上海板块均价](#) [广州板块均价](#) [深圳板块均价](#) [郑州板块均价](#)

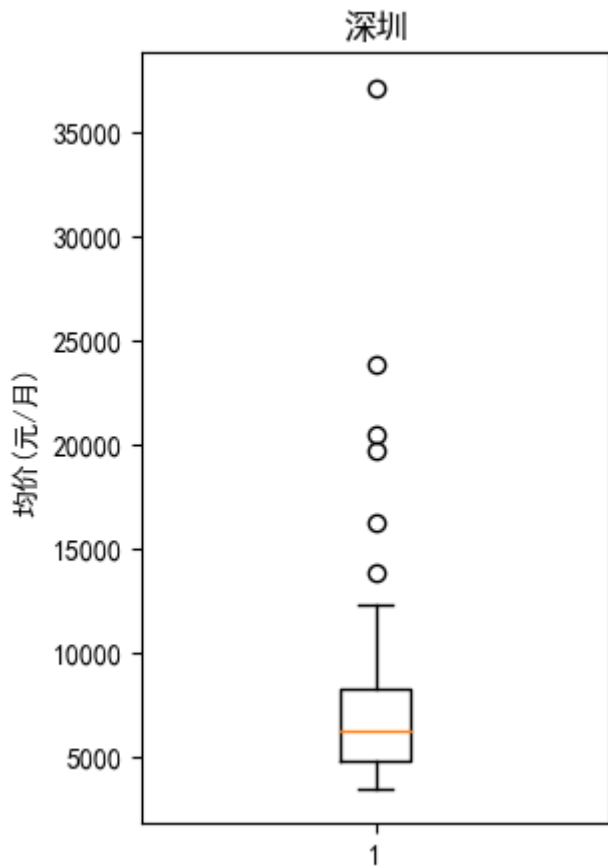
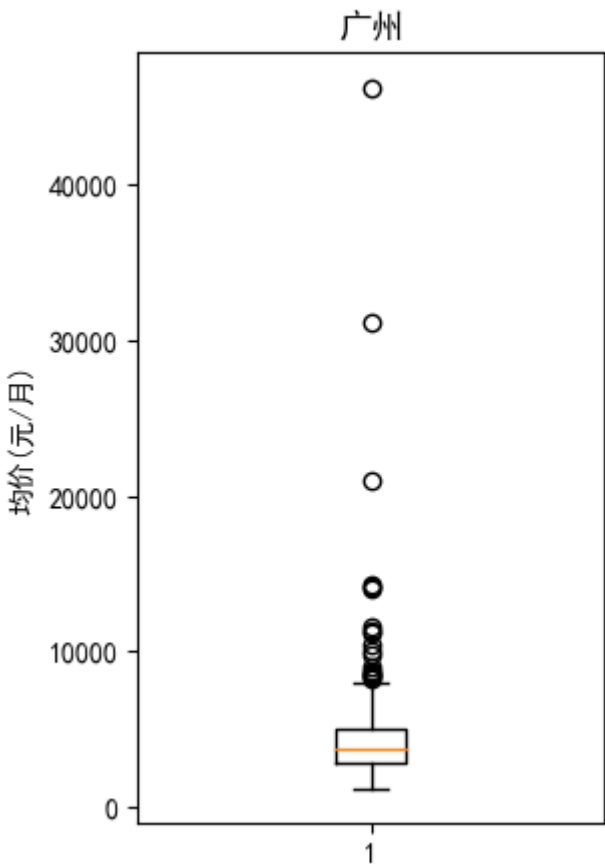
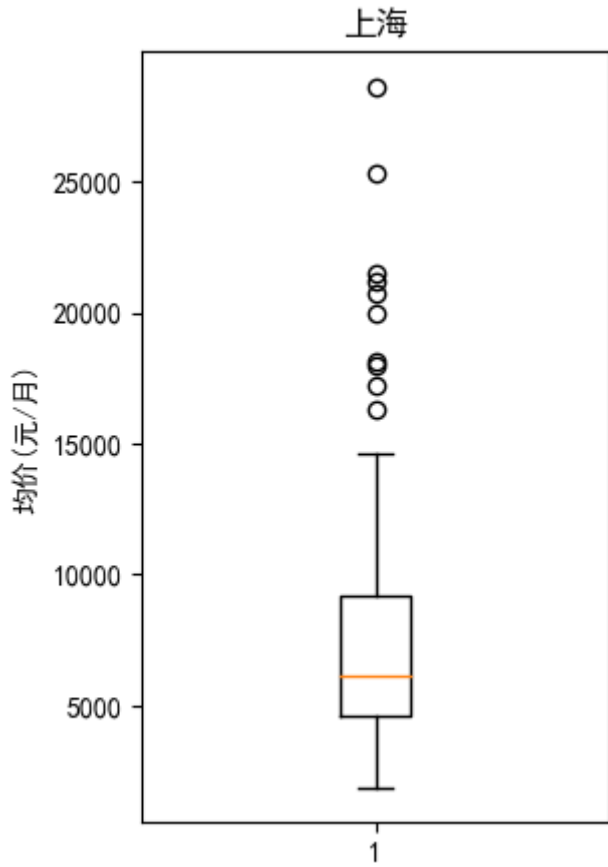
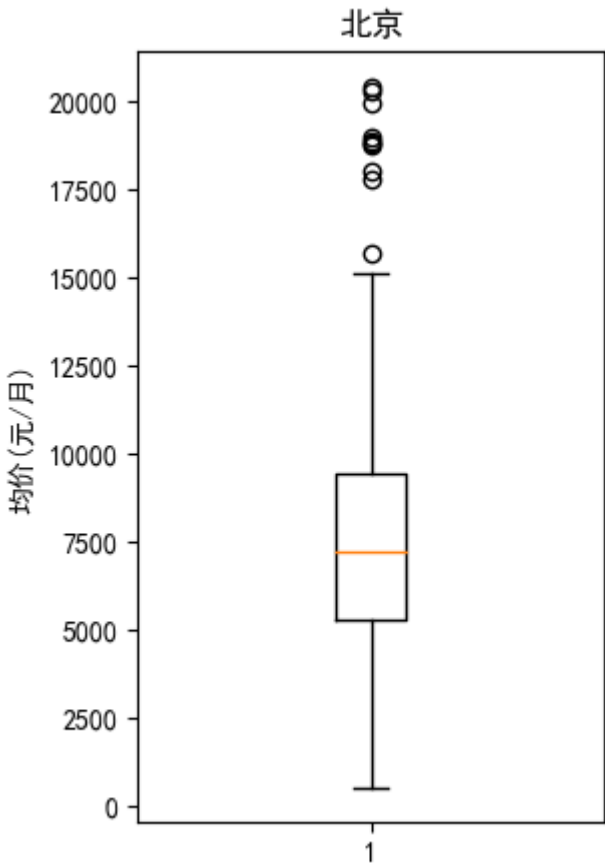
这里采用散点图来表示，因为散点图能够很好的表示数据的分布情况，而且能够很好的表示数据的异常情况，比如上海的某个板块的均价就很高，这样的数据就很容易被发现。

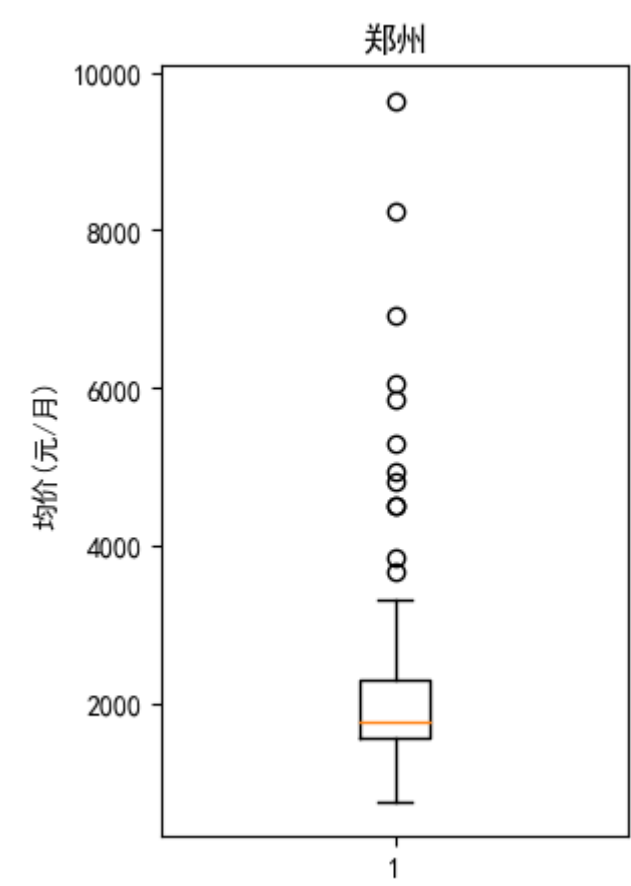


下面是各个城市的板块数量的散点图，为了能从中看出城市里不同板块中房子的数量，所以对数据进行了一些基本的归一化处理，然后再进行绘图。这里能从图上看出来，北京的板块数据相对分散，而上海的板块数据相对分散，这样的数据也能够很好的反映出城市的特点。



下面对五个城市的板块价格进行了箱线图的绘制，从图中能够看出每个城市板块价格的分布情况，以及每个城市板块价格的异常情况，比如上海的某个板块的均价就很高，这样的数据就很容易被发现同时也能直观地观察到方差等数据。





6. 朝向租金分布比较

比较各个城市不同朝向的单位面积租金分布情况，采用合适的图或表形式进行展示。哪个方向最高，哪个方向最低？各个城市是否一致？如果不一致，你认为原因是什么？

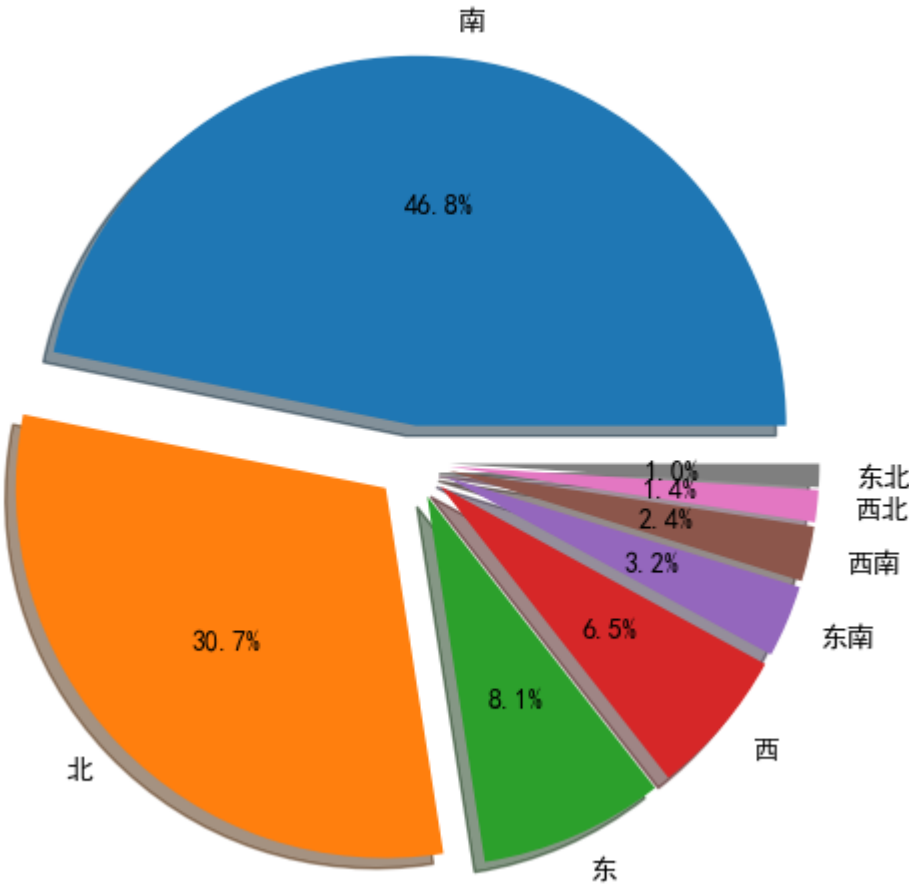
代码编写

与上面的代码类似, [朝向租金分布](#)

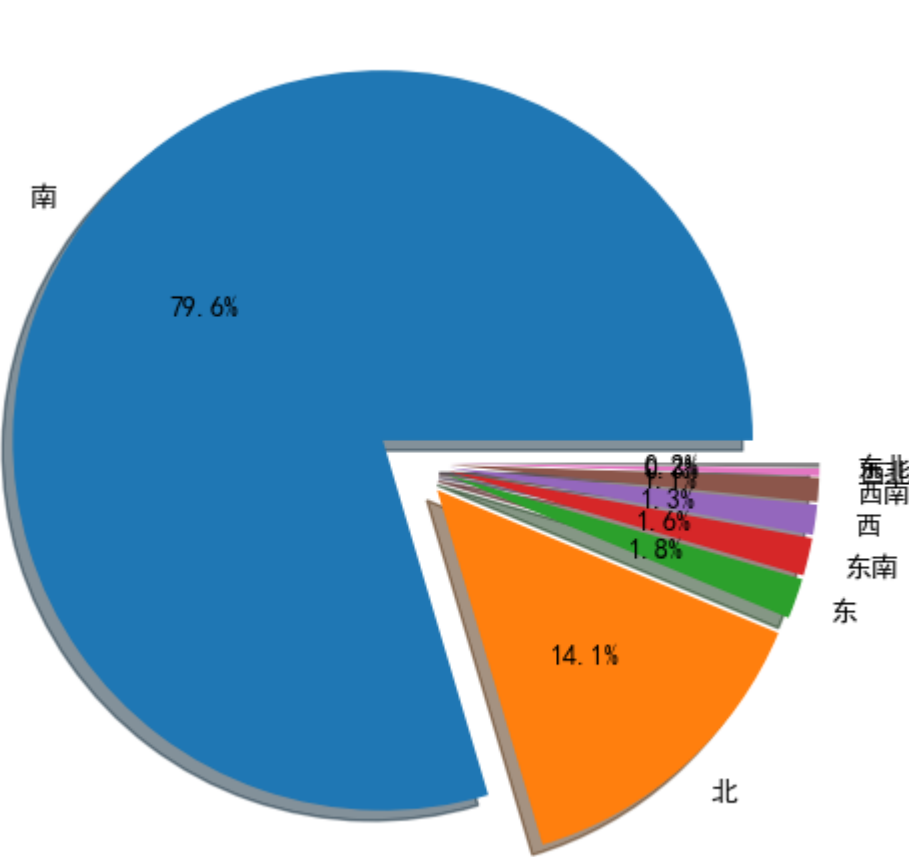
结果展示

下面五张图表示了各个城市中不同朝向的比例，能看到不同城市朝向分布有很大差别，也许是受气候风俗等影响

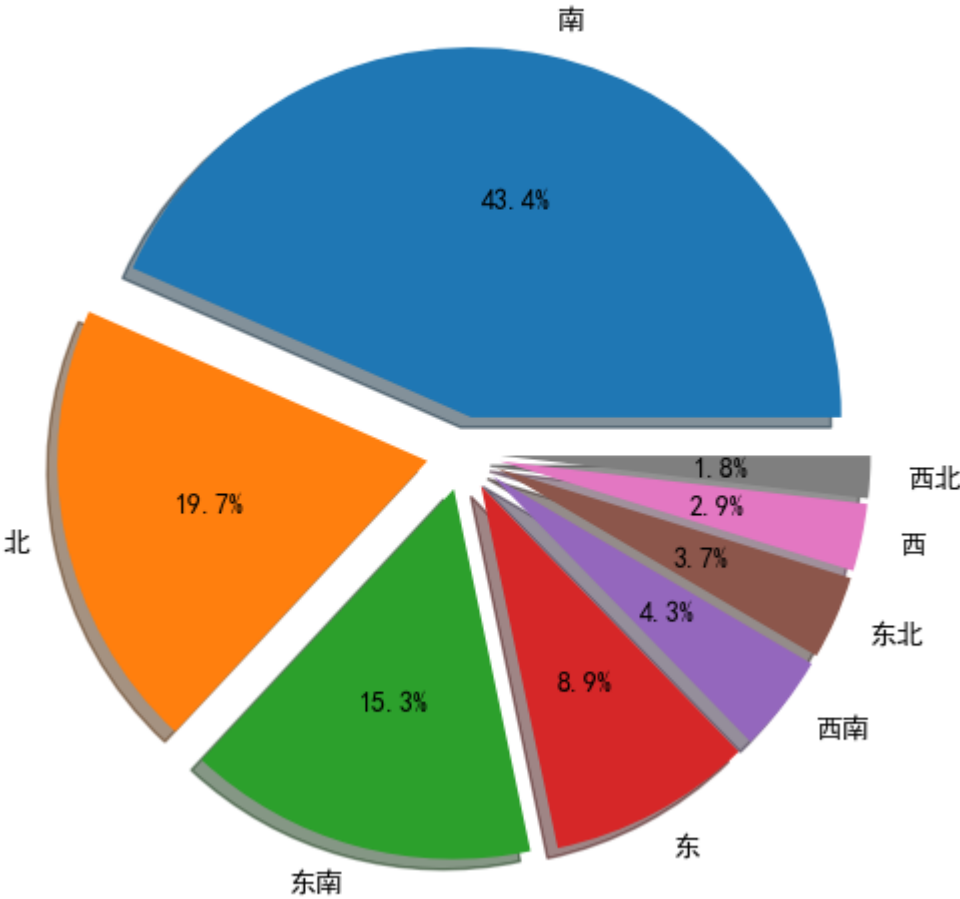
北京的房源朝向分布



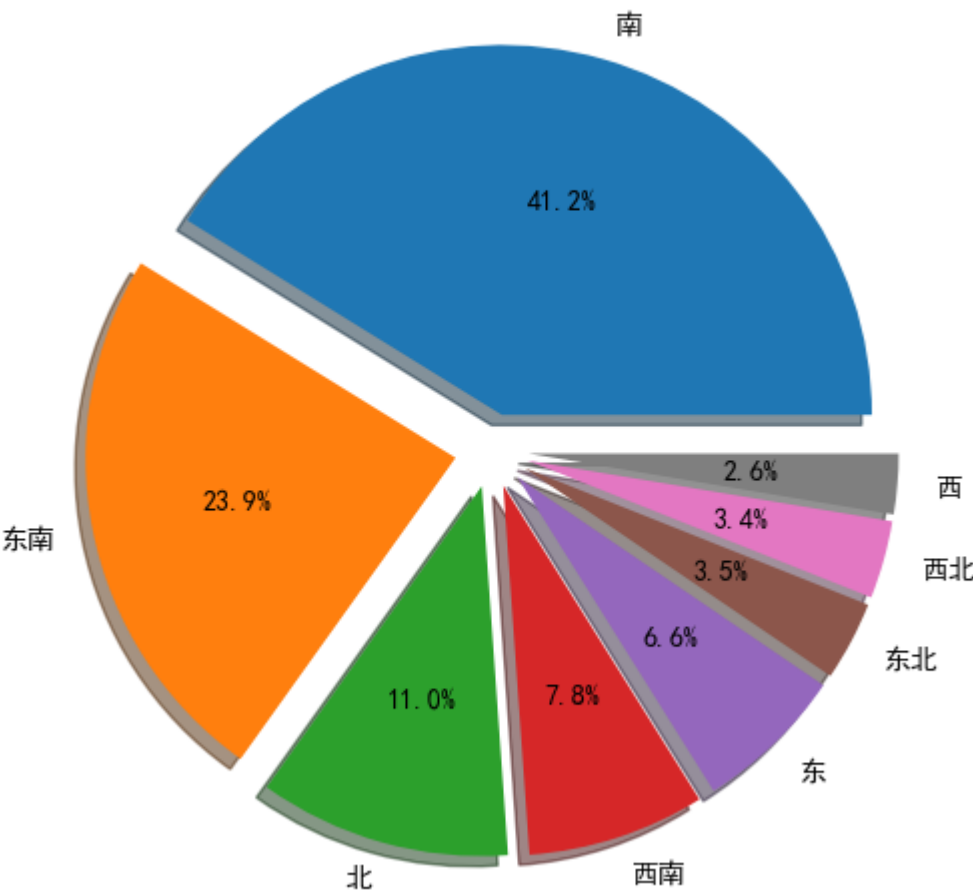
上海的房源朝向分布

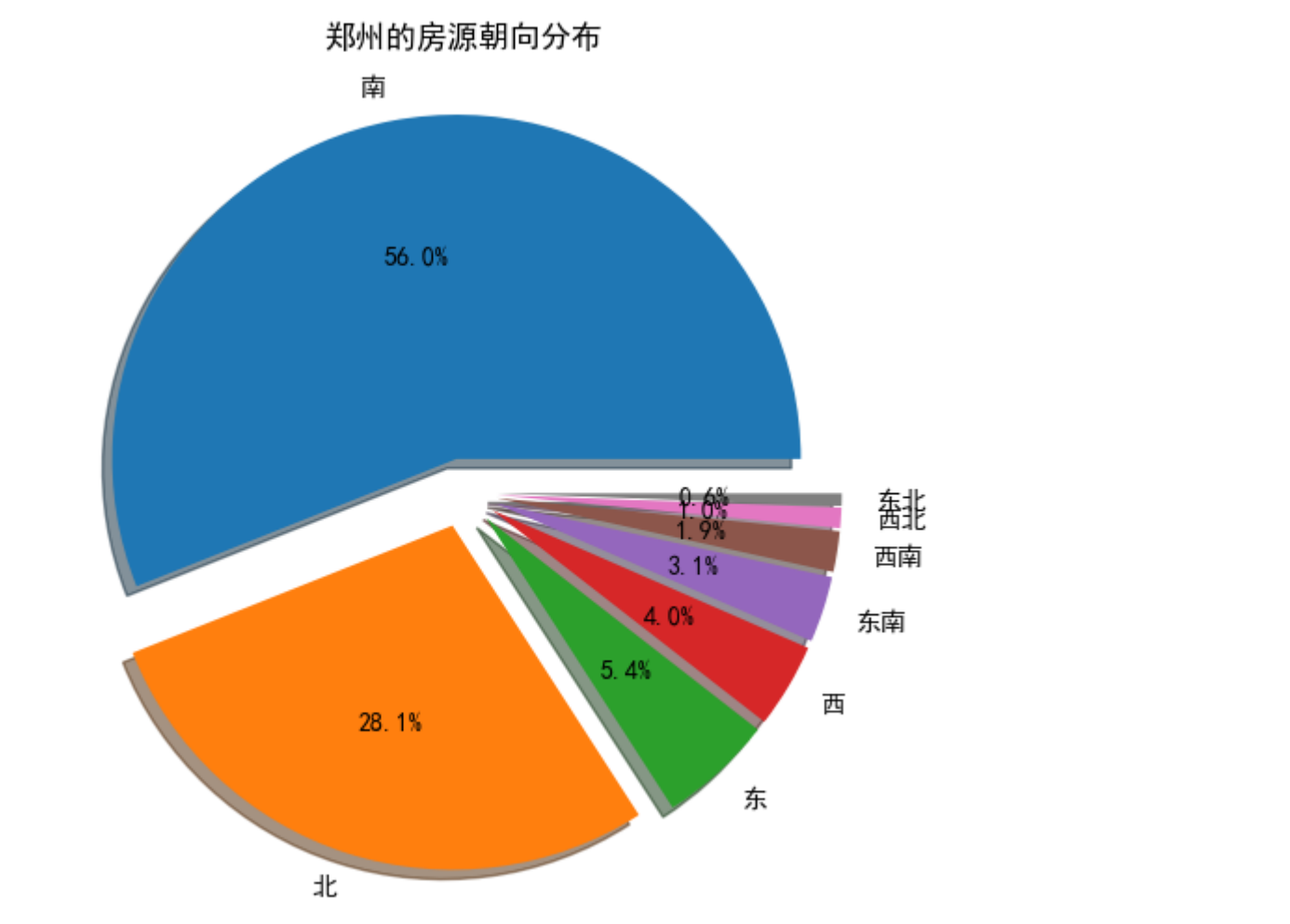


广州的房源朝向分布

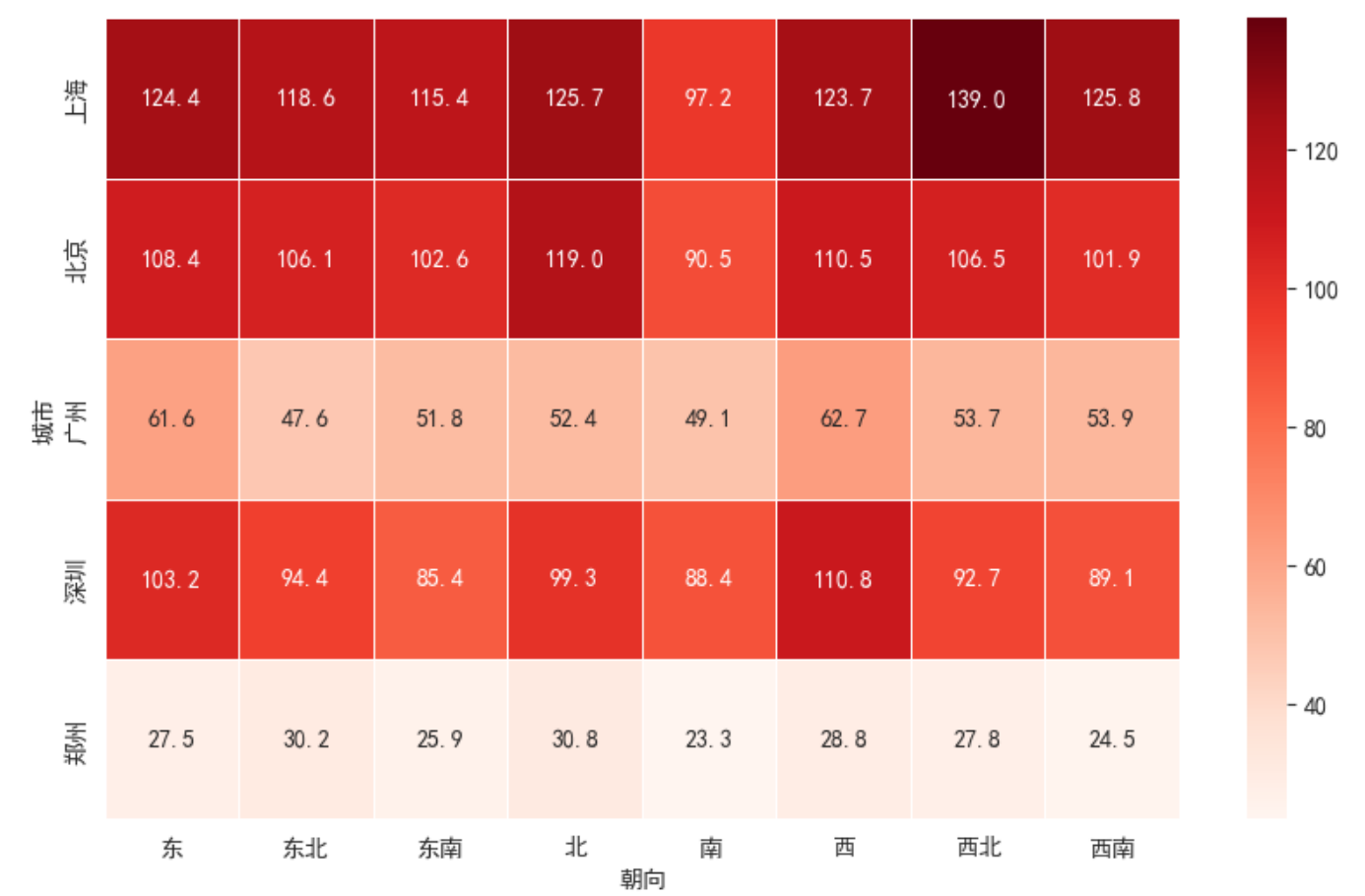


深圳的房源朝向分布





下面是城市和朝向的单位面积价格的热力图，颜色越深，价格越低



7. 城市平均工资与租金分布关系

查询各个城市的平均工资，分析并展示其和单位面积租金分布的关系。比较一下在哪个城市租房的负担最重？

数据获取

根据各地区人力资源和社会保障局所提供的数据

- 北京,135567 11297 [数据来源](#)
- 上海 136757 11396 [数据来源](#)
- 广州 130596 10883 [数据来源](#)
- 深圳 164754 13729 [数据来源](#)
- 郑州 93191 7766 [数据来源](#)

中国人均居住面积 41.76平方米 [数据来源](#) 中国人均住房支出占比 24% [数据来源](#)

数据分析

代码编写

8. 与2022年数据对比

与2022年的租房数据进行对比（只比较北上广深4个城市，原始数据会给出），总结你观察到的变化情况，并用图、表、文字等支撑你得到的结论。

代码编写

因为2022年的租房数据格式是json，所以需要有一个程序进行转化

```
import json
import csv
import os
file_list_chinese=['北京','上海','广州','深圳']
filename_list=
["BeijingHouseInfo.json","ShanghaiHouseInfo.json","GuangzhouHouseInfo.json","Shenz
henHouseInfo.json"]
file_list=["bj","sh","gz","sz"]
for i in range(4):
    with open(filename_list[i],'r',encoding='utf-8') as f:
        with open(file_list[i]+'.csv','w',encoding='utf-8',newline='') as f1:
            writer=csv.writer(f1)

writer.writerow(['name_chinese','block','house_type','direct','area','price'])
    for line in f.readlines():
        dic=json.loads(line)

writer.writerow([file_list_chinese[i],dic['district'],dic['layout'],dic['direction
'].split(" ")[0],dic['area'],dic['total_price']])
```

10. 提交要求

以pdf格式提交到教学云平台上，文件名为学号，总页数不超过30页。

11. 截止时间

截止时间为2024年1月6日23:59。