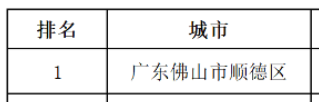
**大数据采集实验**

熟悉实践大数据实战的整体流程，包括python 数据爬取、数据清洗加数据预处理、python可视化图形展示、得出结论，指导预测，从实际的项目需求入手，完成整个大数据的流程处理，真正体会到大数据在实际生活中的应用。

项目需求是找出某市各个区的房屋出租数量与出租均价之间的关系。本次使用的数据爬取工具为python。

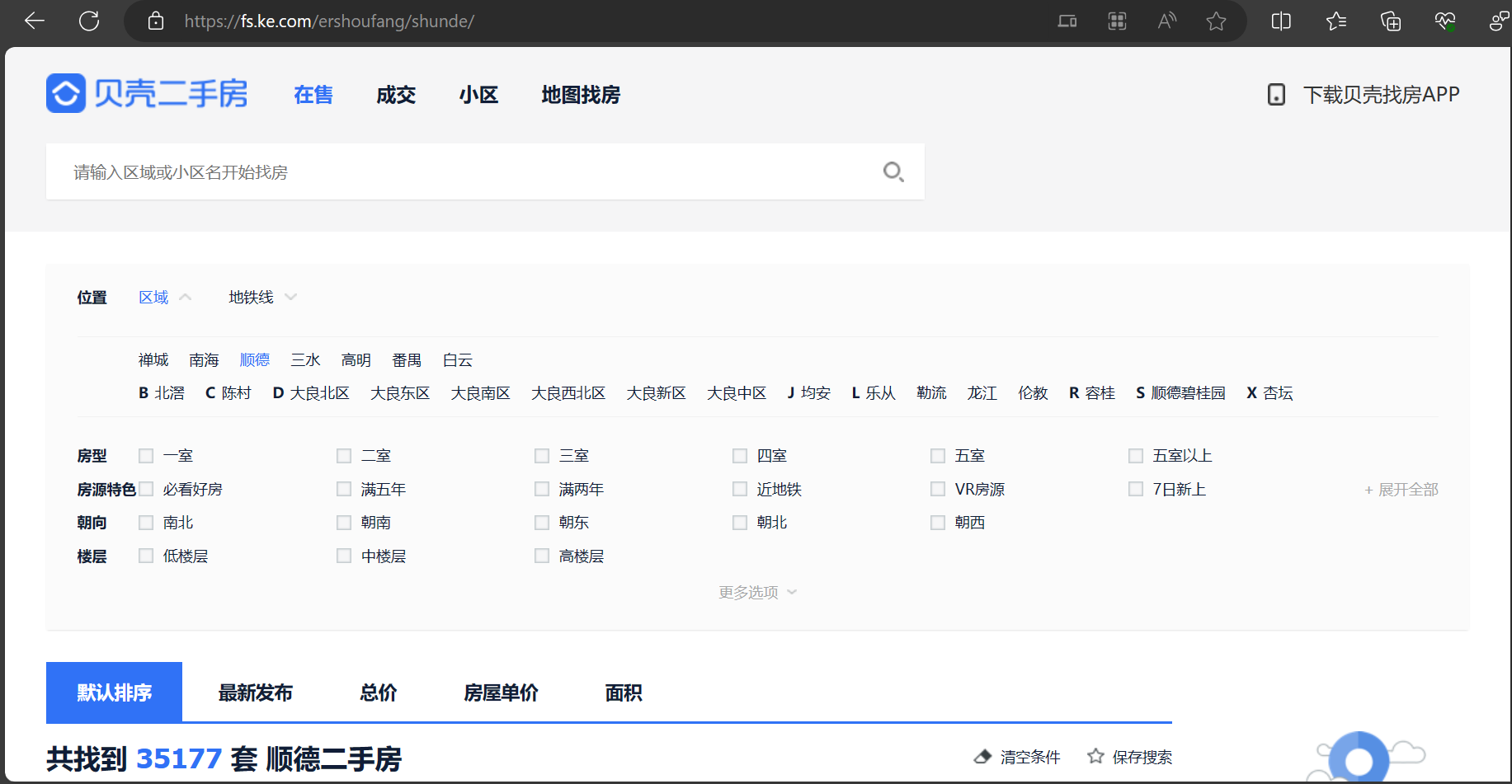
**·背景**

此次选择爬取的地区是广东省佛山市顺德区，也是因为顺德区成为全国百强区榜首，想要“蹭热度”，当然也同自己生长在顺德区有关，希望更加了解顺德区这一地方，算是为将来的一些规划提提提前做点研究工作。



**·数据爬取**

项目的数据来源于贝壳网佛山市顺德区(https://fs.ke.com/ershoufang/shunde/)的房价信息，如下图：

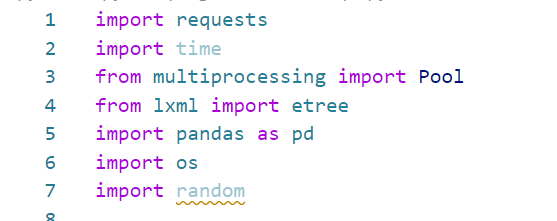




主要采用了python中的requests库来进行爬取网页租房信息。

这里主要涉及的是关于浏览器网页源代码的分析并书写XPath路径这一难点，或者说是关键点，网页源代码应该是用到Javascript来书写的(应该是吧)，网页Xpath路径主要分为两条，一条是绝对路径，从html源地址出发到相关信息存储地址，另一条则是相对路径，相对于某一页面出发寻找相关信息。

**·导入相应库**



—requests库主要用来爬取网页租房信息

—time库则是对爬取时间进行一定的设置，因为网站会对爬取进行一定的限制，设置时间会提高效率以及防止被封IP(实验中并没有用到time库也是可以进行爬取的，但是可能存在多次访问才能爬取成功的情况，虽然并没有显示，但网站确实有对频繁访问进行一定的时间约束，所以更加建议设置间隔时间，如0.5s访问一次)

—Pool库是实现多线程的，多线程爬取也能够提高爬取的效率

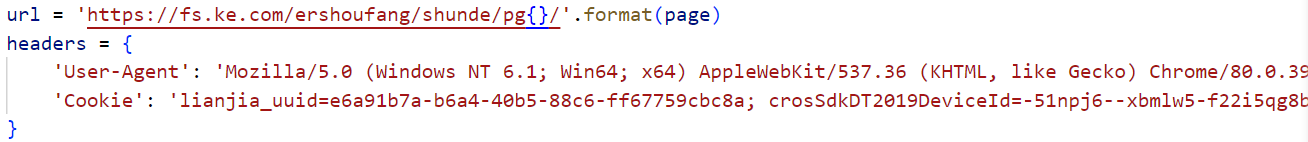
—etree库是同Beautiful Soup库类似的库，可以对爬取网站的源代码进行解析

—pandas库和os库都是用来存储爬取的信息，由于pandas存储的一定特点，加入os的辅助使得存储更加优化

—random库是用来实现随机访问IP的，防止IP被封(实验中也并没有用到，因为多个IP实现起来比较麻烦，单IP在少量少次爬取信息时风险还是比较低的)

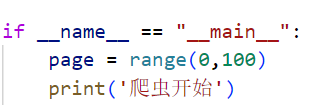
**·设置UserAgent，伪装成浏览器对网页进行访问**

爬取的网站是贝壳网顺德区的二手房网站，对爬虫进行伪装，避免服务器对程序的拒绝访问，也就不会发送相关的信息，但UserAgent的设置可以不唯一，此处仅设置一个UserAgent。

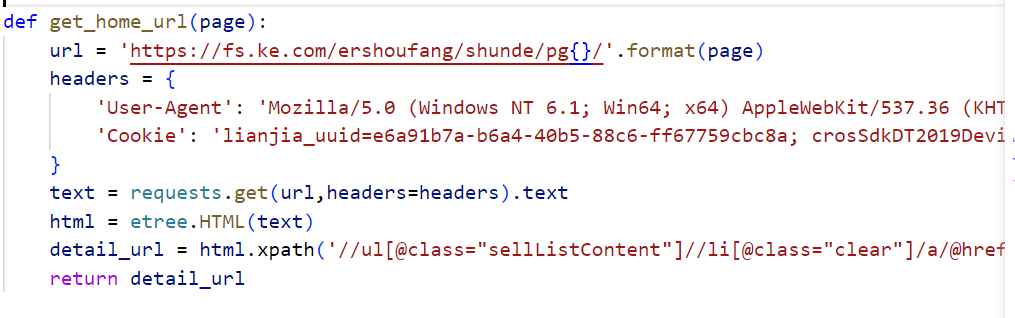


**·页码设置**

经过几次爬虫的尝试，在爬取到100页之后，数据就会出现重复，所以爬取页码直接设置从0到100页，相当于对数据进行了一定的去重处理。



**·发出请求(获取网站源代码信息)**



**·获取详细的房源信息**

秉着“宁可错杀一百，不能放过一千”的原则，选择将一房的所有信息都爬取下来，一共是包含23个属性(当然后面数据处理部分会进行数据的筛选)。其中也包含了XPath路径的书写，由于已经获取到每一页的网站，所以利用的是相对路径进行寻址。





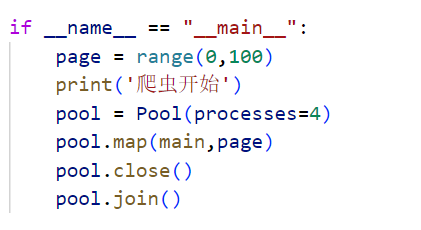
**·数据储存**

数据直接是利用pandas库保存在csv文件中，利用的是’UTF-8-SIG’的编码方式，利用’UTF-8’的编码方式存储会导致除了英文其余数据显示乱码的现象，所以采用此编码。



·**主程序**

main程序直接是对页码参数进行设置，然后调用已经编写好的函数。



**·总体代码**

import requests

from multiprocessing import Pool

from lxml import etree

import pandas as pd

import os

def get\_home\_url(page):

    url = 'https://fs.ke.com/ershoufang/shunde/pg{}/'.format(page)

    headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36',

        'Cookie': 'lianjia\_uuid=e6a91b7a-b6a4-40b5-88c6-ff67759cbc8a; crosSdkDT2019DeviceId=-51npj6--xbmlw5-f22i5qg8bh36ouv-yttqkmwdf; \_ga=GA1.2.121082359.1579583230; ke\_uuid=6de1afa21a5799c0874702af39248907; \_\_xsptplus788=788.1.1579583230.1579583347.4%234%7C%7C%7C%7C%7C%23%23Q6jl-k46IlXjCORdTOp6O3JyzHokoUrb%23; select\_city=110000; digv\_extends=%7B%22utmTrackId%22%3A%2280418605%22%7D; lianjia\_ssid=a4ab1bc0-cb04-492f-960c-342c66065da0; Hm\_lvt\_9152f8221cb6243a53c83b956842be8a=1583897013,1583932737; User-Realip=111.196.247.121; sensorsdata2015jssdkcross=%7B%22distinct\_id%22%3A%2216fc67f100b140-06f07f8f707639-33365a06-1049088-16fc67f100c603%22%2C%22%24device\_id%22%3A%2216fc67f100b140-06f07f8f707639-33365a06-1049088-16fc67f100c603%22%2C%22props%22%3A%7B%22%24latest\_traffic\_source\_type%22%3A%22%E7%9B%B4%E6%8E%A5%E6%B5%81%E9%87%8F%22%2C%22%24latest\_referrer%22%3A%22%22%2C%22%24latest\_referrer\_host%22%3A%22%22%2C%22%24latest\_search\_keyword%22%3A%22%E6%9C%AA%E5%8F%96%E5%88%B0%E5%80%BC\_%E7%9B%B4%E6%8E%A5%E6%89%93%E5%BC%80%22%2C%22%24latest\_utm\_source%22%3A%22baidu%22%2C%22%24latest\_utm\_medium%22%3A%22pinzhuan%22%2C%22%24latest\_utm\_campaign%22%3A%22wybeijing%22%2C%22%24latest\_utm\_content%22%3A%22biaotimiaoshu%22%2C%22%24latest\_utm\_term%22%3A%22biaoti%22%7D%7D; Hm\_lpvt\_9152f8221cb6243a53c83b956842be8a=1583933576; srcid='

    }

    text = requests.get(url,headers=headers).text

    html = etree.HTML(text)

    detail\_url = html.xpath('//ul[@class="sellListContent"]//li[@class="clear"]/a/@href')

    return detail\_url

# 获取房源详细数据信息

def get\_home\_detail\_infos(detail\_url):

    headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36',

        'Cookie': 'lianjia\_uuid=e6a91b7a-b6a4-40b5-88c6-ff67759cbc8a; crosSdkDT2019DeviceId=-51npj6--xbmlw5-f22i5qg8bh36ouv-yttqkmwdf; \_ga=GA1.2.121082359.1579583230; ke\_uuid=6de1afa21a5799c0874702af39248907; \_\_xsptplus788=788.1.1579583230.1579583347.4%234%7C%7C%7C%7C%7C%23%23Q6jl-k46IlXjCORdTOp6O3JyzHokoUrb%23; select\_city=110000; digv\_extends=%7B%22utmTrackId%22%3A%2280418605%22%7D; lianjia\_ssid=a4ab1bc0-cb04-492f-960c-342c66065da0; Hm\_lvt\_9152f8221cb6243a53c83b956842be8a=1583897013,1583932737; User-Realip=111.196.247.121; sensorsdata2015jssdkcross=%7B%22distinct\_id%22%3A%2216fc67f100b140-06f07f8f707639-33365a06-1049088-16fc67f100c603%22%2C%22%24device\_id%22%3A%2216fc67f100b140-06f07f8f707639-33365a06-1049088-16fc67f100c603%22%2C%22props%22%3A%7B%22%24latest\_traffic\_source\_type%22%3A%22%E7%9B%B4%E6%8E%A5%E6%B5%81%E9%87%8F%22%2C%22%24latest\_referrer%22%3A%22%22%2C%22%24latest\_referrer\_host%22%3A%22%22%2C%22%24latest\_search\_keyword%22%3A%22%E6%9C%AA%E5%8F%96%E5%88%B0%E5%80%BC\_%E7%9B%B4%E6%8E%A5%E6%89%93%E5%BC%80%22%2C%22%24latest\_utm\_source%22%3A%22baidu%22%2C%22%24latest\_utm\_medium%22%3A%22pinzhuan%22%2C%22%24latest\_utm\_campaign%22%3A%22wybeijing%22%2C%22%24latest\_utm\_content%22%3A%22biaotimiaoshu%22%2C%22%24latest\_utm\_term%22%3A%22biaoti%22%7D%7D; Hm\_lpvt\_9152f8221cb6243a53c83b956842be8a=1583933576; srcid='

    }

    detail\_text = requests.get(detail\_url,headers= headers).text

    html = etree.HTML(detail\_text)

    all\_data = []

    # 所在地址

    szdz = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@data-component="overviewIntro"]//\*[@class="overview"]//\*[@class="content"]//\*[@class="areaName"]//\*[@class="info"]/a/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(szdz)

    # 小区名称

    xqmc = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@data-component="overviewIntro"]//\*[@class="overview"]//\*[@class="content"]//\*[@class="communityName"]/a/text()')[0]

    all\_data.append(xqmc)

    # 总价格

    zjg = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@data-component="overviewIntro"]//\*[@class="overview"]//\*[@class="content"]//\*[@class="price "]/span/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(zjg)

    # 单价

    dj = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@data-component="overviewIntro"]//\*[@class="overview"]//\*[@class="content"]//\*[@class="unitPrice"]/span/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(dj)

    # 房屋户型

    fwhx = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[1]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(fwhx)

    # 建筑面积

    jzmj = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[2]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(jzmj)

    # 户型结构

    hxjg = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[3]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(hxjg)

    # 建筑类型

    jzlx = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[4]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(jzlx)

    # 所在楼层

    szlc = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[5]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(szlc)

    # 套内面积

    tnmj = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[6]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(tnmj)

    # 房屋朝向

    fwcx = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[7]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(fwcx)

    # 建筑结构

    jzjg = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[8]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(jzjg)

    # 装修情况

    zxqk = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[9]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(zxqk)

    # 梯户比例

    thbl = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[10]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(thbl)

    # 配备电梯

    pbdt = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="content"]/ul/li[10]/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(pbdt)

    # 挂牌时间

    gpsj = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[1]/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(gpsj)

    # 交易权属

    jyqs = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[2]/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(jyqs)

    # 上次交易

    scjy = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[3]/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(scjy)

    # 房屋用途

    fwyt = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[4]/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(fwyt)

    # 房屋年限

    fwnx = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[5]/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(fwnx)

    # 产权所属

    cqss = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[6]/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(cqss)

    # 抵押信息

    dyxx = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[7]/span/text()')[1].replace(" ","").replace("\n","")

    all\_data.append(dyxx)

    # 房本备件

    fbbj = html.xpath('//\*[@id="beike"]//\*[@class="sellDetailPage"]//\*[@class="m-content"]//\*[@class="introContent"]//\*[@class="transaction"]//\*[@class="content"]/ul/li[8]/text()')[0].replace(" ","").replace("\n","")

    all\_data.append(fbbj)

    return all\_data

# 数据保存至csv文件里（使用pandas中的to\_csv保存）

def save\_data(data):

    data\_frame = pd.DataFrame(data,columns = ['所在地址','小区名称','总价格','单价','房屋户型','建筑面积','户型结构','建筑类型','所在楼层','套内面积','房屋朝向','建筑结构','装修情况','梯户比例','配备电梯','挂牌时间','交易权属','上次交易','房屋用途','房屋年限','产权所属','抵押信息','房本备件'])

    print(data\_frame)

    if not os.path.exists('shunde.csv'):

        data\_frame.to\_csv('shunde.csv',encoding='utf\_8\_sig',mode='a',index=False,header=['所在地址','小区名称','总价格','单价','房屋户型','建筑面积','户型结构','建筑类型','所在楼层','套内面积','房屋朝向','建筑结构','装修情况','梯户比例','配备电梯','挂牌时间','交易权属','上次交易','房屋用途','房屋年限','产权所属','抵押信息','房本备件'])

    else:

        data\_frame.to\_csv('shunde.csv',encoding='utf\_8\_sig',mode='a',index=False,header=False)

def main(page):

    print('开始爬取第{}页的数据！'.format(page))

    urls = get\_home\_url(page)

    for url in urls:

        print('开始爬取详细网页为{}的房屋详细信息资料！'.format(url))

        all\_data = get\_home\_detail\_infos(detail\_url=url)

        data = []

        data.append(all\_data)

        save\_data(data)

if \_\_name\_\_ == "\_\_main\_\_":

    page = range(0,100)

    print('爬虫开始')

    pool = Pool(processes=4)

    pool.map(main,page)

    pool.close()

    pool.join()

**·数据清洗**

在读取到的数据中仅仅存在极少部分乱码或者空白的情况，对于这些数据，因为它们并不对后续数据分析造成影响，故直接进行删除操作。









**·导入相应库**

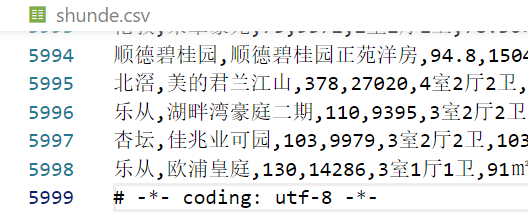


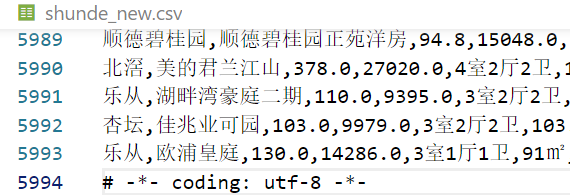
pandas库是用来read操作和保存csv操作

os库则是用来寻找操作系统文件

**·删除异常数据**







对于新旧两个csv文件的行数相可知，五行异常数据已经被删除。

**·总体代码**

import pandas

import os

all\_data = pandas.read\_csv('shunde.csv',encoding='utf-8')

data = all\_data.drop([348,349,2859,3787,5447])

def save\_data(data):

    data\_frame = pandas.DataFrame(data,columns = ['所在地址','小区名称','总价格','单价','房屋户型','建筑面积','户型结构','建筑类型','所在楼层','套内面积','房屋朝向','建筑结构','装修情况','梯户比例','配备电梯','挂牌时间','交易权属','上次交易','房屋用途','房屋年限','产权所属','抵押信息','房本备件'])

    print(data\_frame)

    if not os.path.exists('shunde\_new.csv'):

        data\_frame.to\_csv('shunde\_new.csv',encoding='utf\_8\_sig',mode='a',index=False,header=['所在地址','小区名称','总价格','单价','房屋户型','建筑面积','户型结构','建筑类型','所在楼层','套内面积','房屋朝向','建筑结构','装修情况','梯户比例','配备电梯','挂牌时间','交易权属','上次交易','房屋用途','房屋年限','产权所属','抵押信息','房本备件'])

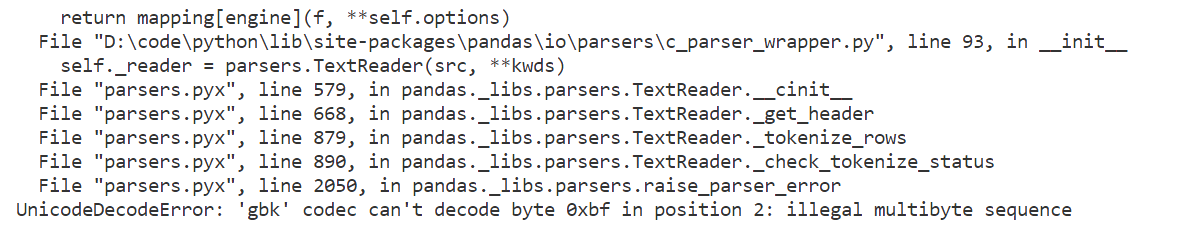
    else:

        data\_frame.to\_csv('shunde\_new.csv',encoding='utf\_8\_sig',mode='a',index=False,header=False)

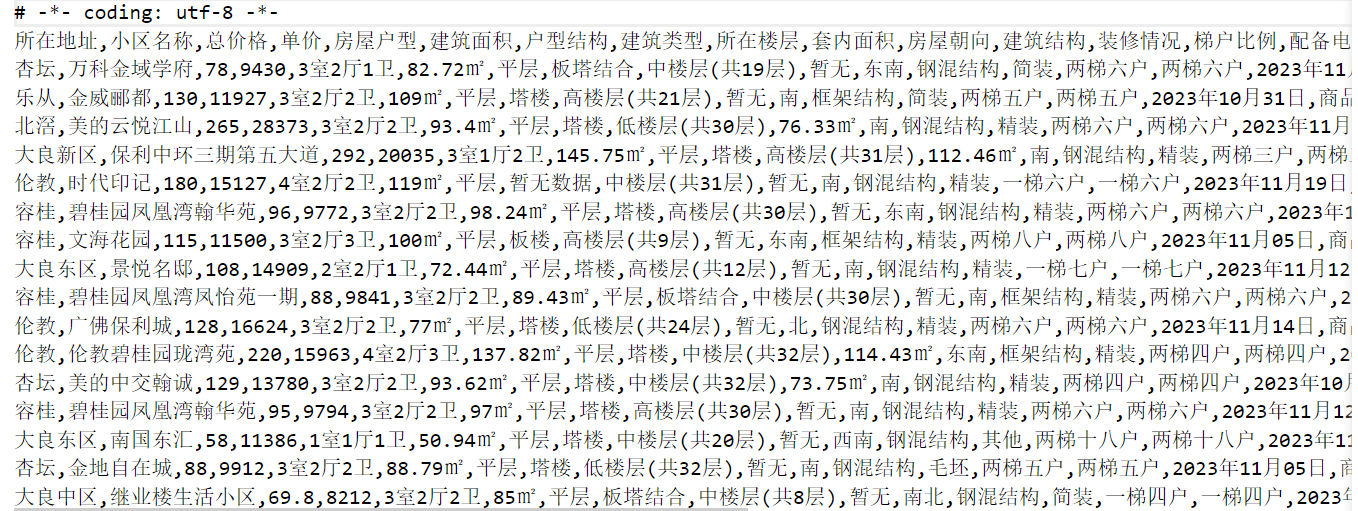
save\_data(data)

**·数据预处理与数据可视化**

**Tips：对于已经生成的csv文件，如果直接使用pandas库进行read操作，有可能会出现编码错误如下：**



**无论是使用UTF-8编码还是其他编码方式进行read操作都无法成功读取，此时在csv文件的开头加入 # -\*- coding: utf-8 -\*- 能够对文件编码方式进行改变**



**再利用UTF-8的编码方式进行读取则可成功读取**

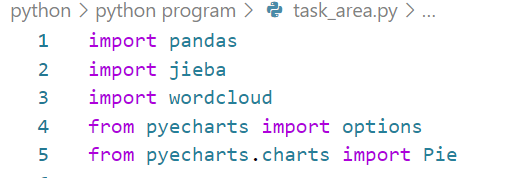
**但是如果放在表头会使得列名被更改，在数据读取时会造成影响，所以也可以将其放在表尾，在数据处理时手动忽略表尾的 # -\*- coding: utf-8 -\*-**

这里将数据预处理和数据可视化放在一起处理是因为并不能通过一次数据处理就能完成数据可视化，而是相对应的，所以一起处理。

**·各区域房源数量**

对于街道房源分析，思路是通过词云图获取大致的房源分布情况，然后再对数据进行分析，制作饼状图具体查看各街道的房源数量，其中由于顺德区大良街道是有老城区、新城区之分，所以在地理位置上大良街道比起其他街道会进行再次的细分。

**·导入相应库**

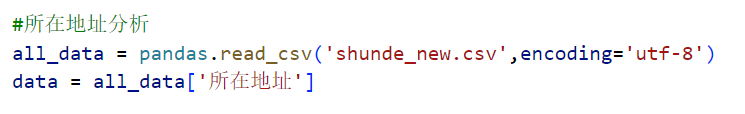


pandas库还是用来进行对数据的一些操作

jieba库是专门用来进行中文切分的库，wordcloud库则是来制作词云图，两者配合使用

pycharts库是可视化库，options与Pie两个函数结合使用进行饼状图的绘制

**·数据获取**



数据分析的第一步是数据获取，由于是对各街道的房源数进行统计，所以是直接将csv文件中对应的“所在位置”一列作为分析数据。

**·街道区域划分**

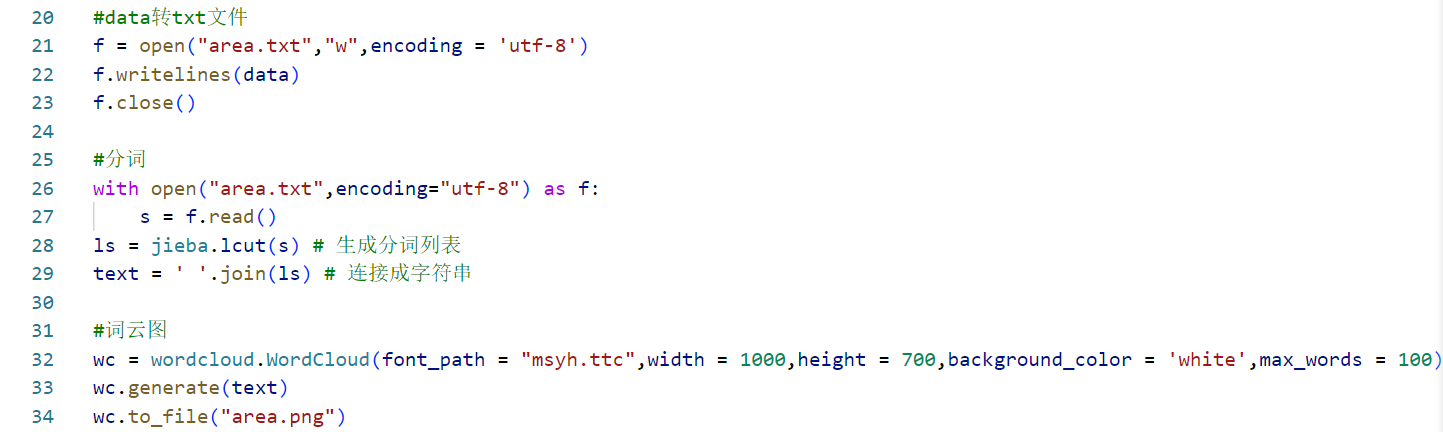


将“所在位置”的所有数据进行去重处理即可得到对应的街道区域划分，将最后的编码语句删去之后得到的结果如下：



**·词云图**

在得到街道区域划分后，可以通过词云图的方式来对大致的房源分布进行分析，这里的思路就是将“所在位置”进行分词处理，通过词云图的频率与大小关系来反映一定的情况。



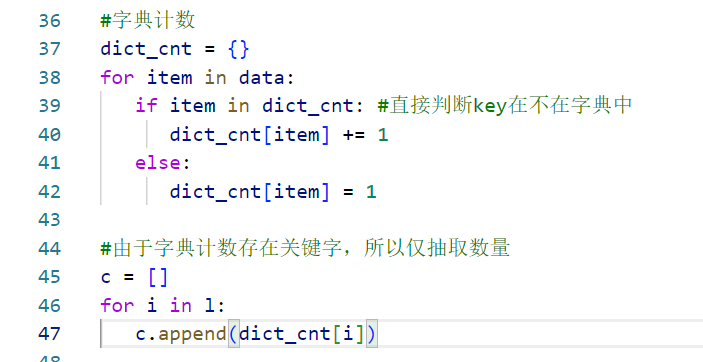
得到的词云图如下：



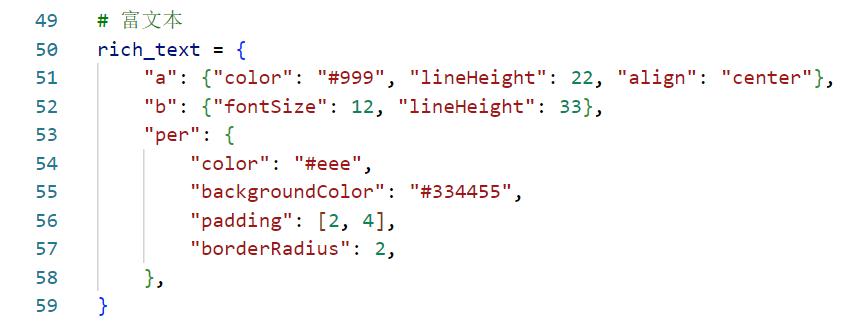
从词云图中可以看出，大良街道各区、龙江街道、北滘街道的房区是比较多的，但是否完全正确，客观来看是并不完全对的，因为分词时会出现不准确的情况，例如词云图中的容桂大良、新区之类的词，所以词云图只是用来进行一个大致的分析预测，如果需要更加准确，我认为可以通过字典的方式将上面得到的街道区域划分录入，然后再进行分词、制作词云图，这样会更加准确。

**·饼状图**

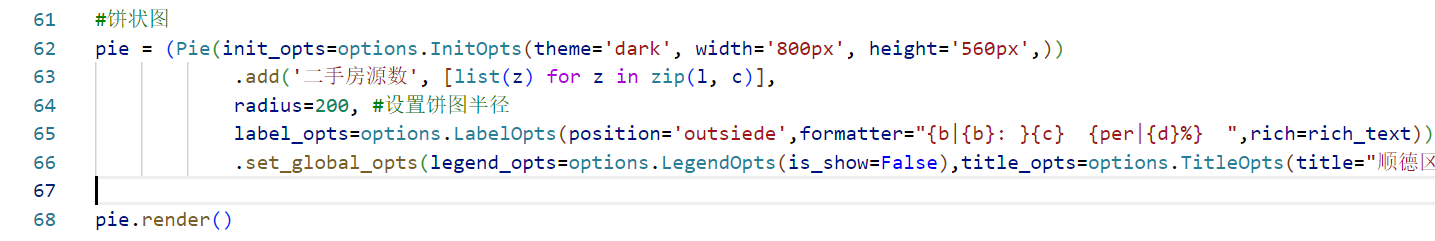
在通过词云图对各区域房源数量进行大致分析后，选择利用饼状图对各区域房源数量进行更加细致的分析。

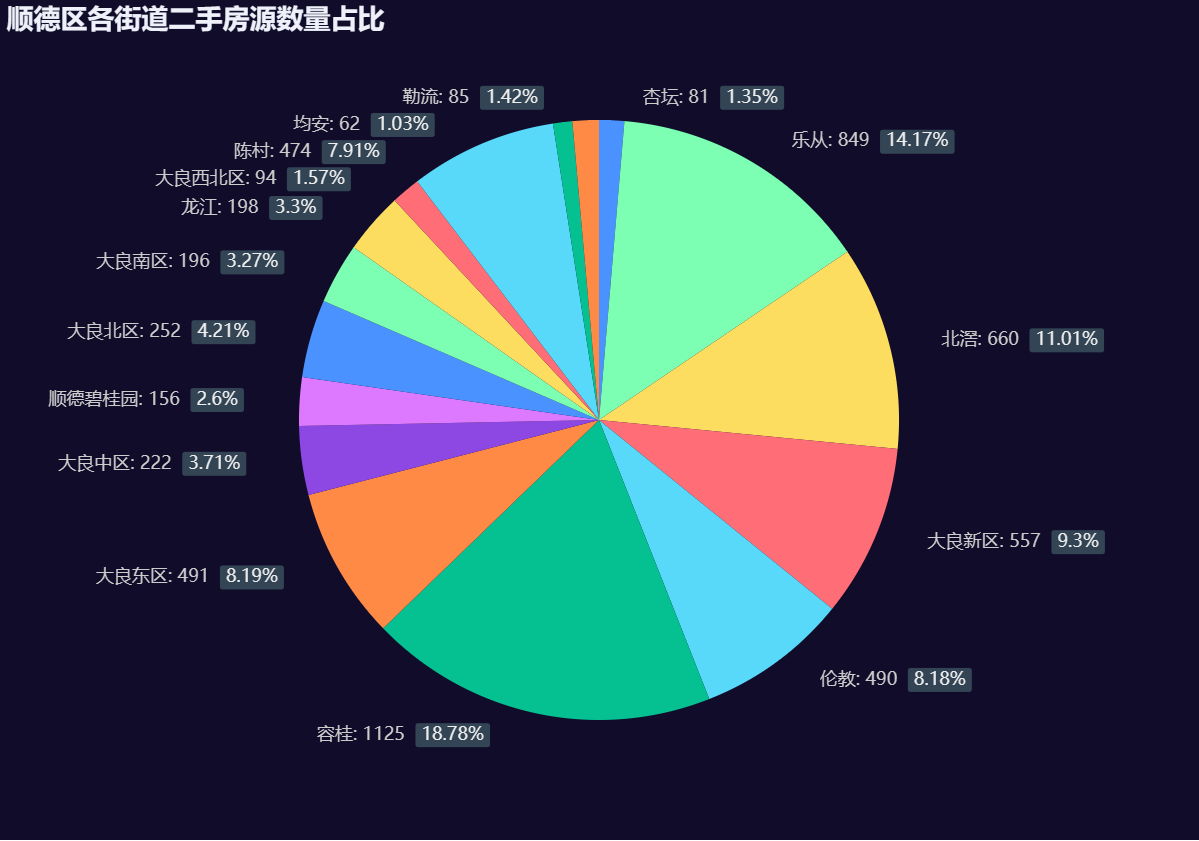


这里运用到上述所描述的“字典”思想，通过每次将关键字，即区域的信息比对或者录入进行后续操作，这里的操作即为计数，但是通过字典计数会生成一个带有关键字的列表，所以列表c就是将关键字删除，得到各区域对应的房源数量。



富文本就是用来对饼状图中字体、颜色等一些信息的设置。



最后是生成一个html文件，在浏览器中打开效果如下：  


通过饼状图可以看出确实词云图存在不正确的预测，但是也是有可取之处。

其中大良街道、容桂街道、北滘街道、伦教街道以及乐从街道的房源数是占比较多的。

**·总体代码**

import pandas

import jieba

import wordcloud

from pyecharts import options

from pyecharts.charts import Pie

#所在地址分析

all\_data = pandas.read\_csv('shunde\_new.csv',encoding='utf-8')

data = all\_data['所在地址']

#去重

l = []

for i in data:

    if i not in l:

        l.append(i)

#由于最后一行为UTF-8编码所以删除

l.pop()

print(l)

#data转txt文件

f = open("area.txt","w",encoding = 'utf-8')

f.writelines(data)

f.close()

#分词

with open("area.txt",encoding="utf-8") as f:

    s = f.read()

ls = jieba.lcut(s) # 生成分词列表

text = ' '.join(ls) # 连接成字符串

#词云图

wc = wordcloud.WordCloud(font\_path = "msyh.ttc",width = 1000,height = 700,background\_color = 'white',max\_words = 100)

wc.generate(text)

wc.to\_file("area.png")

#字典计数

dict\_cnt = {}

for item in data:

   if item in dict\_cnt: #直接判断key在不在字典中

      dict\_cnt[item] += 1

   else:

      dict\_cnt[item] = 1

#由于字典计数存在关键字，所以仅抽取数量

c = []

for i in l:

   c.append(dict\_cnt[i])

# 富文本

rich\_text = {

    "a": {"color": "#999", "lineHeight": 22, "align": "center"},

    "b": {"fontSize": 12, "lineHeight": 33},

    "per": {

        "color": "#eee",

        "backgroundColor": "#334455",

        "padding": [2, 4],

        "borderRadius": 2,

    },

}

#饼状图

pie = (Pie(init\_opts=options.InitOpts(theme='dark', width='800px', height='560px',))

            .add('二手房源数', [list(z) for z in zip(l, c)],

            radius=200, #设置饼图半径

            label\_opts=options.LabelOpts(position='outsiede',formatter="{b|{b}: }{c}  {per|{d}%}  ",rich=rich\_text))

            .set\_global\_opts(legend\_opts=options.LegendOpts(is\_show=False),title\_opts=options.TitleOpts(title="顺德区各街道二手房源数量占比"),))

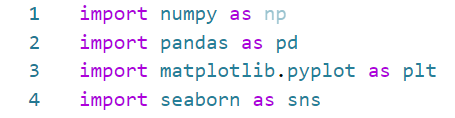
pie.render()

**·各区域房源单价与总价**

**·各区域房源单价（均值）**

对于各区域房源的单价分析，由于数据量相对来说还是比较大的，所以采取求平均值的方法来进行分析，当然，平均值并不能全面地反映出房源单价的特点信息，这里利用可视化进行分布的分析，从另一个角度反应房源单价的方差。

**·相应库导入**



numpy库和pandas库主要是对数据进行操作

matplotlib和seaborn库则是用于数据可视化

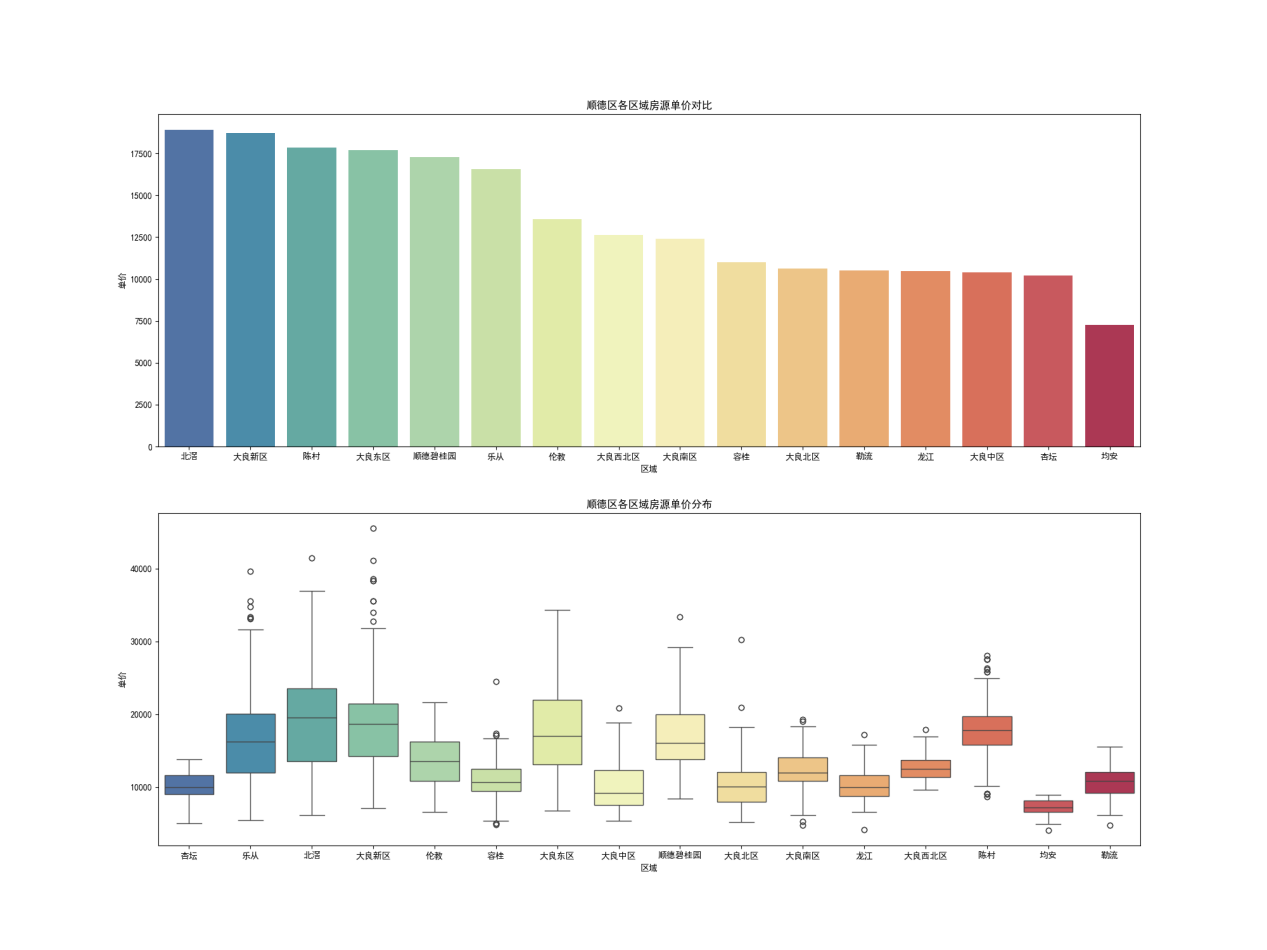
**·数据处理以及可视化**



由于是对各区域的单价进行分析，所以这里利用groupby函数对原始数据进行分组操作，再通过mean函数求均值可以得到某个区域的单价均值。

Tips：这里存在两个关键点，一是乱码问题，如果不引入plt.rcParams库对文字以及符号进行设置，在作图时会导致图片中文字生成变为方格，无法正常显示；二是reset\_index函数的引入，由于groupby会进行重新排序分组，所以如果不重设索引，在后续引用数据时可能会出现KeyError的错误。（由于本人是利用vscode进行python的编写运行，所以plt.show函数是不合法的操作，无法使用，而是选择将其保存为png文件）

得到的png文件效果如下：



通过普通直方图，可以明显地对比出各区域的房源单价均值的差异，可以看出顺德区各区域的房价还是比较平均的，可以说存在两个层级，一个是北滘、大良新区等组成的比较高水平的单价，一个是以容桂、勒流等组成的相比之下水平比较普通的单价。

通过箱型直方图，可以比较直观地看出各区域大部分房源的单价还是比较均匀分布，各区域的方差都比较接近。

**·各区域房源总价**

对于各区域房源的总价，由于总价与房源大小与单价有关，进行均值分析带来的意义不太明显，所以采取折线图的方式来分析，并且使用的是excel文件操作。

Tips：这里采用折线图是因为想制作房源总价与面积大小的关系，但是excel文件散点图无法设置横坐标标签，就无法利用面积大小做横坐标，所以采用折线图的方式，毕竟折线图也是可以看作是“加了连线”的散点图。

首先是通过筛选选择所在区域



接着筛选出各区域进行散点图的绘制

北滘

陈村

大良北区

大良东区

大良南区

大良西北区

大良新区

大良中区

均安

乐从

勒流

龙江

伦教

容桂

顺德碧桂园

杏坛

从各个区域的房源总价折线图还是能够看出其总价的分布是比较均匀的，少有较高以及较低的“异常”总价，说明顺德区各区域的房价还是比较稳定。当然，以上的分析并不完全正确，因为房源总价还与单价直接相联系，同时，电梯配备、周边设施、交通便利程度等其其它客观因素也对单价会造成影响，进而影响总价。

**·数据分析总结**

经过以上对顺德区各区域的房源数以及单价均价、总价的数据分析以及可视化，可以看出，顺德区对于房价的控制还是比较稳定的，二手房的房源数不能说“数不胜数”但也十分充足，可以供选择。但是，由于贝壳网对爬虫的限制，这里只能爬到“冰山一角”的数据，以及个人对数据的分析并不全面，像是房屋结构、房屋朝向等方面的数据并没有使用、分析，造成的数据浪费对此次数据分析也是起到负面作用。不过，仅从仅有的数据以及上述的分析来看，顺德区是比较吸引人才的，房源数充足，房价也在各个区间都有分布且大部分处于一个合理的范围，如果想要在比较发达的地方生活，可以选择北滘、大良比较旧的城区购买房屋，其周边设施、交通便利程度等各方面都比较的完善、良好，如果想要在发达但是压力不那么大的地方生活，容桂相比起北滘、大良旧城区来说，平均房价相对低些，但城市设施、发达程度也不落后太多，最后若是想要选择比较清闲、稍稍偏远的地方，可以去到勒流、龙江甚至是更加偏一点的杏坛生活，生活压力相对来说会小很多，但最近也都逐渐在发展建设，是比较适合长远发展的地方。