**广州城市理工学院**

**2021-2022学年度第1学期**

**课程名称： Python语言课程设计（网络爬虫）**

**题　　目： 链家新房数据爬取**

**专业班级： 软件工程4班**

**年　　级： 2019级**

**姓　　名： 孙誉瑛**

**学　　号：** 201910098039

**班级序号：** **07**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 一级指标 | 二级指标 | 三级指标及分值 | | 分值 | 评分及成绩 |
| 得分 |
| 代码  （占60%） | 1.达到基本要求，功能点完整（355%） | 网页抓取 | 10 | 55 |  |
| 页面解析 | 10 |
| 数据意义 | 15 |
| 数据存储 | 20 |
| 4.运行测试（5%） | | | 5 |  |
| 项目报告（20%） | 1.内容与结构完整（10%） | | | 10 |  |
| 2.规格满足要求，文字描述规范，思路描述清晰（10%） | | | 10 |  |
| 答辩成绩 （20%） | 1.对实训设计技术和结构熟悉，能进行合理说明（10%） | | | 10 |  |
| 2.能较好的回答答辩所提问题，解释合理清晰（10%） | | | 10 |  |
| 合计（百分制） | | | | |  |

目录

[一、 爬取主题说明 3](#_Toc88212859)

[1. 背景 3](#_Toc88212860)

[2. 数据来源 3](#_Toc88212861)

[3. 数据项 3](#_Toc88212862)

[二、 技术可行性分析 3](#_Toc88212863)

[三、 详细设计及实现 4](#_Toc88212864)

[1. 抓取 4](#_Toc88212865)

[2. 解析 6](#_Toc88212866)

[3. 存储 7](#_Toc88212867)

[4. 图片保存到本地 8](#_Toc88212868)

[5. 数据可视化 10](#_Toc88212869)

[6. 主函数 11](#_Toc88212870)

[四、 运行测试 11](#_Toc88212871)

[五、 总结 14](#_Toc88212872)

# 爬取主题说明

## 背景

**需求来源于生活**，大数据时代来临，数据就是核心，数据就是生产力，越来越多的企业开始注重收集用户数据,而爬虫技术是收集数据的一种重要手段。如今距离毕业不远了，生活压力也在增大，压力来源之一就是买房问题。以此为基准，我决定做一个爬取房价数据的爬虫作业。

## 数据来源

链家新房，首页：https://gz.fang.lianjia.com/

## 数据项

所有广州房源。按页获取。具体数据如图：

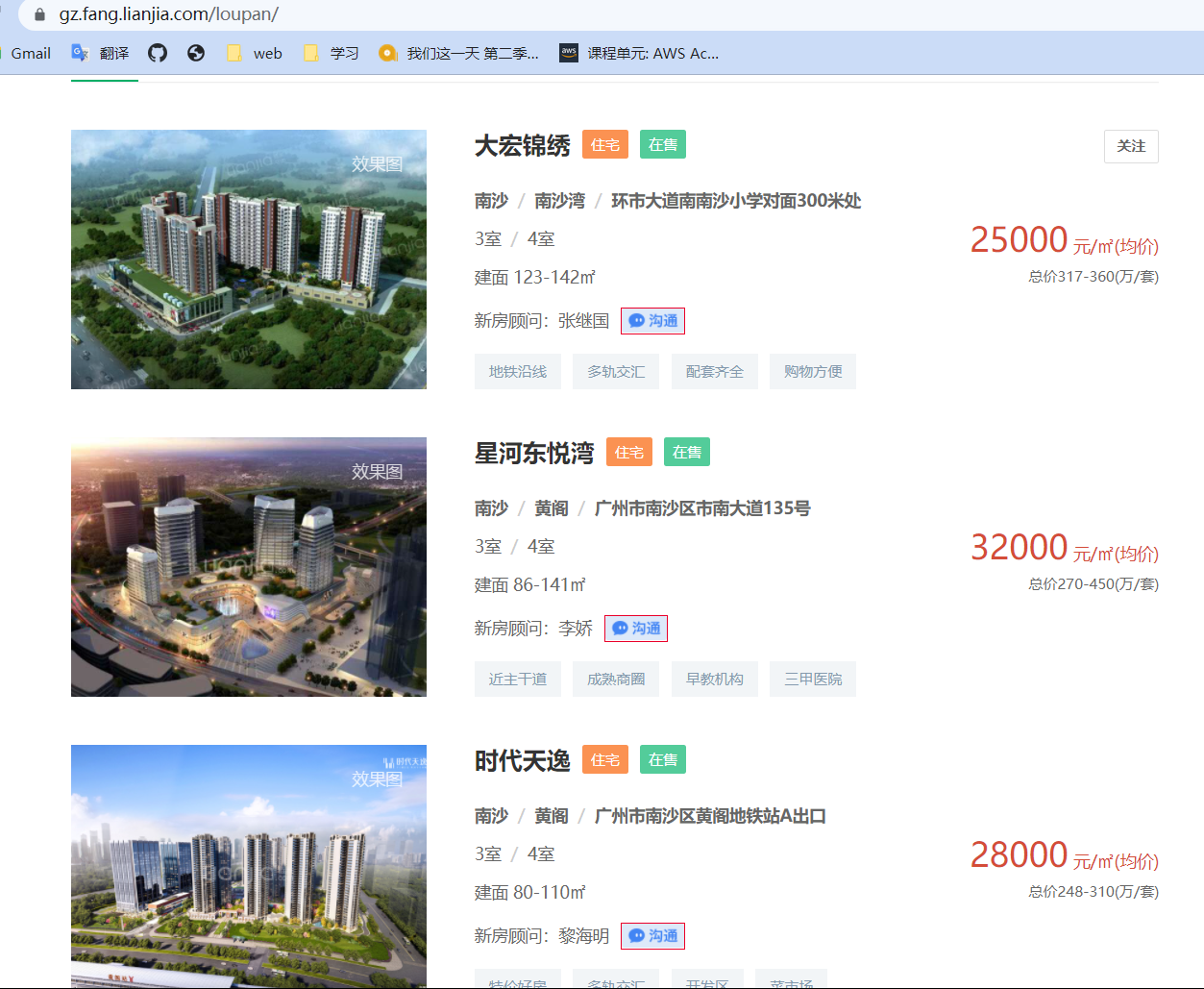


图 1 要爬取的数据项

# 技术可行性分析

1. 技术方案

这次的语言主要用到Python 3.9, 具体用requests库对链接进行数据抓取，用BeautifulSoup4辅助解析，用mysql存储。

Mysql 8.0存储数据。

1. 相关依赖包说明

Pymysql，链接mysql数据库插件

BeautifulSoup，数据抓取解析插件

Time，设置线程休眠

Requests，http库，是连接池。

matplotlib.pyplot，可视化绘图插件。

# 详细设计及实现

## 抓取



图 2链家数据图



图 3 链家静态页面图

我去这个链家的广州房源下大致看了一下这个网页的结构和我需要爬取的资料位置，锁定位置之后就开始数据获取。

首先分析他的结构。这个开头的’gz’就是我们需要获取的市的名字拼音缩写，fang.lianjia.com/loupan/这个是固定的。然后开始思考多页获取数据问题。第一页是这样的：<https://gz.fang.lianjia.com/loupan/>，到了第二页及以后url就变成了这样：<https://gz.fang.lianjia.com/loupan/pg2/>，多了个’pgn’的标记。在爬取的时候就要分情况讨论，如果想获取多页数据，那就要对这个链接进行一些特殊的处理，如果获取的数据只有第一页，那么这一页的数据就是requests.get(url=a, headers=headers)的数据，如果非第一页，那就要做一个数据拼接，把两页的数据拼起来。由于链家有一个反爬机制，在这个反反爬的过程我思考了很多种办法，也有想过用代理池，但由于数据量不算大且代理池的成本较高（要学docker和redis），所以还是选择了最简单的办法：sleep。爬一页过一会再爬，虽然效率直线下降但是确实也做到了反反爬。下面是拼接两页数据和休眠的代码：

def generate\_cityurl(user\_in\_city): # 生成url

cityurl = 'https://' + user\_in\_city + '.lianjia.com/loupan/'

return cityurl

def areainfo(url):

page = ('pg')

for i in range(1, 4): # 获取1-n页的数据

if i == 1:

i = str(i)

a = (url + page + i + '/')

r = requests.get(url=a, headers=headers)

print(a)

htmlinfo = r.content

else:

i = str(i)

a = (url + page + i + '/')

print(a)

r = requests.get(url=a, headers=headers)

html2 = r.content

htmlinfo = htmlinfo + html2

time.sleep(0.2)

return htmlinfo

## 解析

关于数据的解析，我想的是用列表嵌套字典的形式。这里只贴了一部分代码，对于这里面也是有遇到一点困难的，比如对拿到的数据处理这一块，比如“总价”这个数据，有些楼盘并没有给出总价，在拿到的时候就会是空，但是空数据存入数据库无论是观感还是语法都不太好，所以做了一些处理，如果碰到空就把这个数据的值设成“暂无”。剩下的数据处理就是一些字符串切割之类的操作，这里的图片我用了一个i作为变量加入字典，i在for外面定义，每进行一次加入到h字典操作i就+1，直到图片路径全部存完为止。

hlist = []

def listinfo(listhtml):

areasoup = BeautifulSoup(listhtml, 'html.parser')；

ljhouse = areasoup.find\_all('div', attrs={'class': 'resblock-desc-wrapper'})

loupanimg = areasoup.find\_all("img", attrs={"class": "lj-lazy"})

i=0

for house in ljhouse:

loupantitle = house.find("div", attrs={"class": "resblock-name"})

location = house.find("div", attrs={"class": "resblock-location"}).get\_text()

price = jiage.find("div", attrs={"class": "main-price"}).get\_text().split()[0]

totalprice = "暂无"

if total is not None:

totalprice = total.get\_text()

h = {'title': loupanname, 'wuye': wuye, 'states': xiaoshouzhuangtai, 'location': location.replace("\n", ""),

'jishi': jishi.replace("\n", ""), 'area': area.replace('\n', ''), 'tag': tag.replace('\n', ''), 'price': price.replace('\n', ''),

'totalprice': totalprice,'loupanimg':loupanimg[i].get('data-original')};

i = i+1

hlist.append(h)

声明了一个hlist列表，用bs4解析拿到的url，在for循环里重复获取数据，然后根据网页的结构写出对应的抓取式子并拿到数据添加进h字典。再把h字典添加进hlist列表里。关于价格我这里做了一个处理，如果获取的值是空，那么这个值就回被添加为“暂无”，如果不是空，就会被添加为对应的价格。

## 存储

这里的数据使用mysql数据库存储，图片存储到本地。代码如下：

图片：

# 保存图片

for i in range(len(hlist)):

imgUrl = requests.get(hlist[i]['loupanimg']).content

f = open('C:\\Users\\Lenovo\\Desktop\\study\\py\\catchLJ\\imgFile\\'+hlist[i]['title']+'.jpg', 'wb')

f.write(imgUrl)

print(hlist[i]['title'], "图片正在下载")

print("图片下载完成")

f.close()

先用requests下的get()方法下载图片，content方法对图片进行转码保存到imgUrl这个变量。然后用python自带的open()函数新建并用write()函数写入图片，本地路径是C:\\那一串，文件名是hlist[1][‘title’]也就是楼盘的名字，格式是jpg。然后关闭文件流。

数据库存储：

# 连接数据库

conn = pymysql.connect(

host='localhost',

user='root',

password='123456',

database='catchdata',

charset='utf8'

)

# 使用 cursor() 方法创建一个游标对象 cursor

cursor = conn.cursor()

for i in range(len(hlist)):

try:

insert\_sql = "insert into gzdata(title, wuye, states, location, jishi, area, tag, price, totalprice, imgurl) values ("

for key in hlist[i]:

insert\_sql = insert\_sql + "'" + hlist[i][key] + "',"

insert\_sql = insert\_sql[:-1] + ")"

print(insert\_sql)

cursor.execute(insert\_sql)

conn.commit()

except:pass

# 关闭数据库连接

conn.close()

利用pymysql配置连接数据库到conn变量里，创建游标对象cursor，用一个for来遍历hlist数组，在插入语句里遍历hlist里每个字典的key，把每个key的值存入数据库，再把结果提交到数据库中，然后关闭数据库连接。

## 图片保存到本地

先去看看图片是存在哪里的



图 4 链家图片存储图

<a href="/loupan/p\_zjbswbjgvr/" class="resblock-img-wrapper " title="珠江铂世湾" >

<img alt="珠江铂世湾楼盘图片" class="lj-lazy" data-original="https://image1.ljcdn.com/newhouse-user-image/a086284c435e97a5f35f1ac3cb0cad0a.jpeg.592x432.jpg" src="https://s1.ljcdn.com/feroot/pc/asset/img/blank.gif">

经过观察，发现真正的图片链接其实是在'data-original'这里，那我们只需要获取这串url并下载，编码，存储就可以了。

解决了数据来源问题，之后就是怎么存储了。Python里有自带的io流open()函数，用于打开一个文件，如果文件不存在则会被创建，这里我设置了他的文件名是楼盘名称，格式是jpg，以wb的形式创建。然后把刚才处理好的数据通过write()函数写入文件，图片就成功保存了。

def downloadPic(hlist):

for i in range(len(hlist)):

imgUrl = requests.get(hlist[i]['loupanimg']).content

f = open('C:\\Users\\Lenovo\\Desktop\\study\\py\\catchLJ\\imgFile\\'+hlist[i]['title']+'.jpg', 'wb')

f.write(imgUrl)

print(hlist[i]['title'], "图片正在下载")

print("图片下载完成")

f.close()

## 数据可视化

可视化我用的是matplotlib这个依赖包来辅助开发，我先是定义了三个整形变量lowNum, midNum, HeighNum来记录爬取的数据中单价在不同区间段的数量，又定义了三个变量p\_low, p\_mid, p\_height来记录不同价位区间段数据所占的比例，然后把它们都加进nums数组里，再定义一个labels数组存放需要显示的代表这个区间段的名字，用pie()函数画饼状图。其中数据来源是nums数组，autopct="%.1f%%"是把计算出来的小数转换成百分数显示，shadow是阴影显示与否。再用title()函数规定了这个饼状图的题目和字体大小，最后用show()函数显示图像。这就实现了一个简单的数据可视化功能。

def printPic(list):

plt.rcParams["font.family"] = "kaiti"

priceData = []

lowNum = 0

midNum = 0

heighNum = 0

# 数据准备

for i in range(len(list)):

priceData.append(int(list[i]['price']))

# print(priceData)

for num in priceData:

if num <= 20000:

lowNum = lowNum + 1

elif …

p\_low = lowNum / len(priceData)

p\_mid = midNum / len(priceData)

p\_height = heighNum / len(priceData)

nums = [p\_low, p\_mid, p\_height]

labels = ['0-20000', '20001-45000', '>45000']

# 用Matplotlib画饼图

plt.pie(x=nums, labels=labels, autopct="%.1f%%", shadow=True)

plt.title('楼价区间比例(每平)', size=20)

plt.show()

## 主函数

这属于是对整个代码的分段和整理了。因为个人觉得代码如果一整个文件百来行没有一点注释和分块是很难受的，想实现版本控制和变量更改都是会变得很麻烦，也不符合开发规范。因此，我对这个代码进行了简单的分块，也就是有主函数和各类方法，由主函数控制参数传递和方法调用，这样事情就会变得科学可控。这里贴上我的主函数代码：

if \_\_name\_\_ == '\_\_main\_\_':

user\_in\_city = 'gz'

url = generate\_cityurl(user\_in\_city)

print(url)

areahtml = areainfo(url)

listinfo(areahtml)

downloadPic(hlist)

printPic(hlist)

# 运行测试

为了方便演示，我把一些必要的语句进行了输出，有：需要爬取的url获取打印，下载图片的下载语句打印，获取数据的字典打印，数据库入库完成的打印。

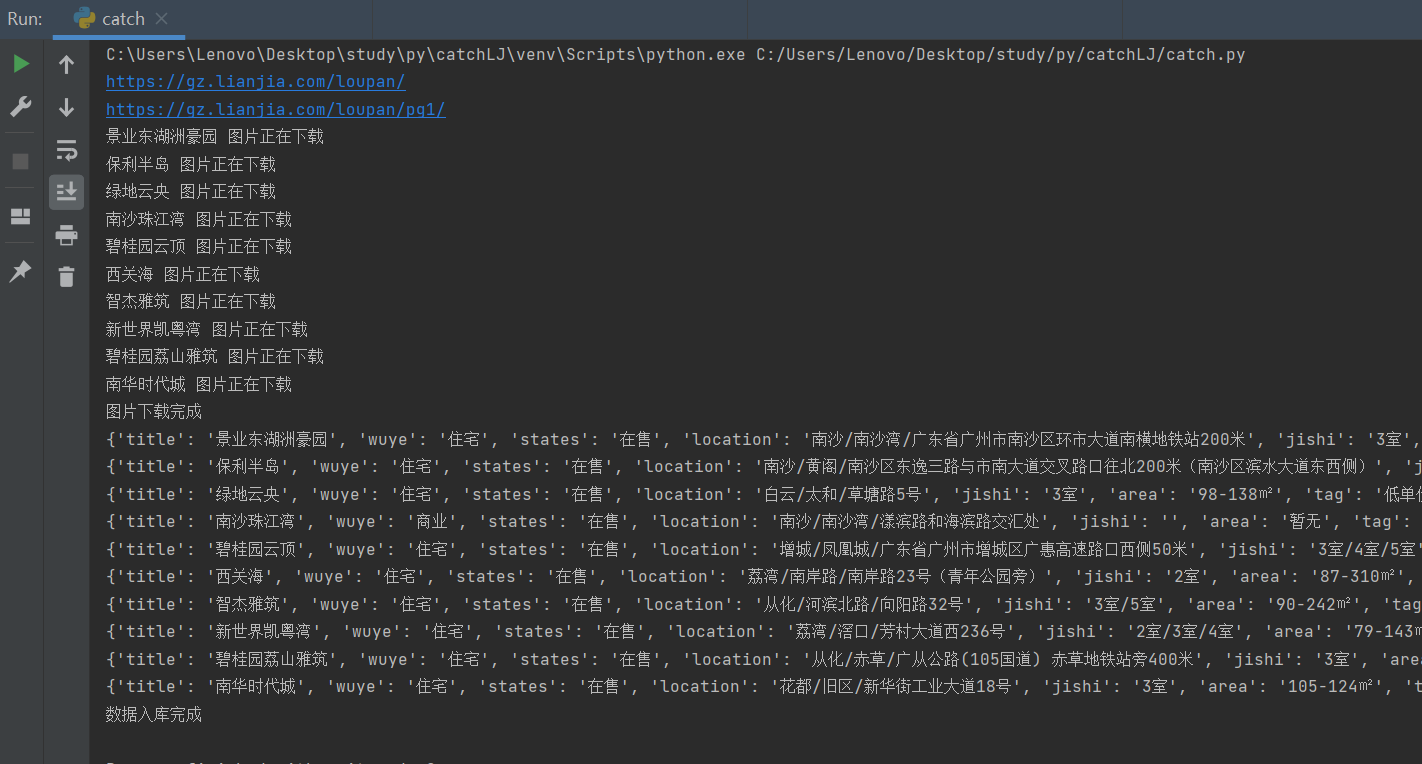


图 5 运行控制台

再就是数据库入库的情况，这里是数据库截图，我用的是mysql。



图 6 数据入库

然后是下载图片保存到本地，我这里选择的是保存到项目根目录下的imgFile文件夹。目录如下：

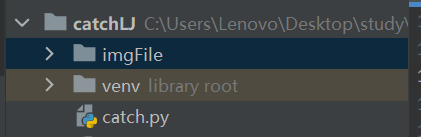


图 7 文件目录图

浏览效果如下：

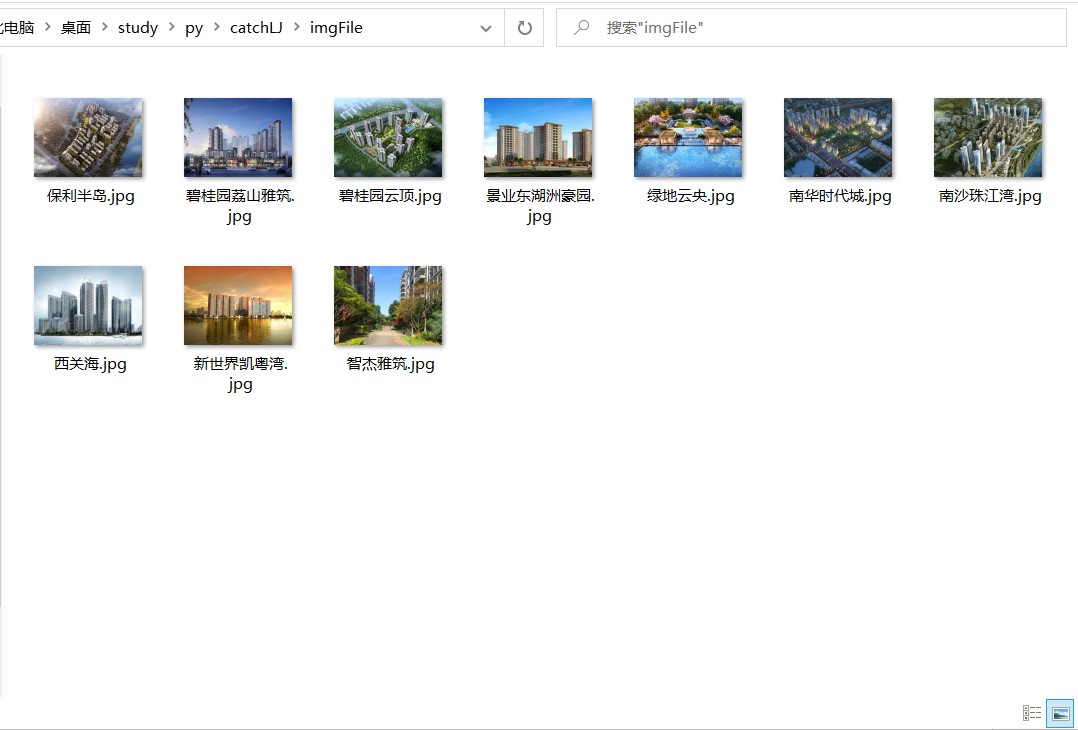


图 8 保存图片浏览

最后就是数据可视化的界面。饼图画出来的效果是这样的：

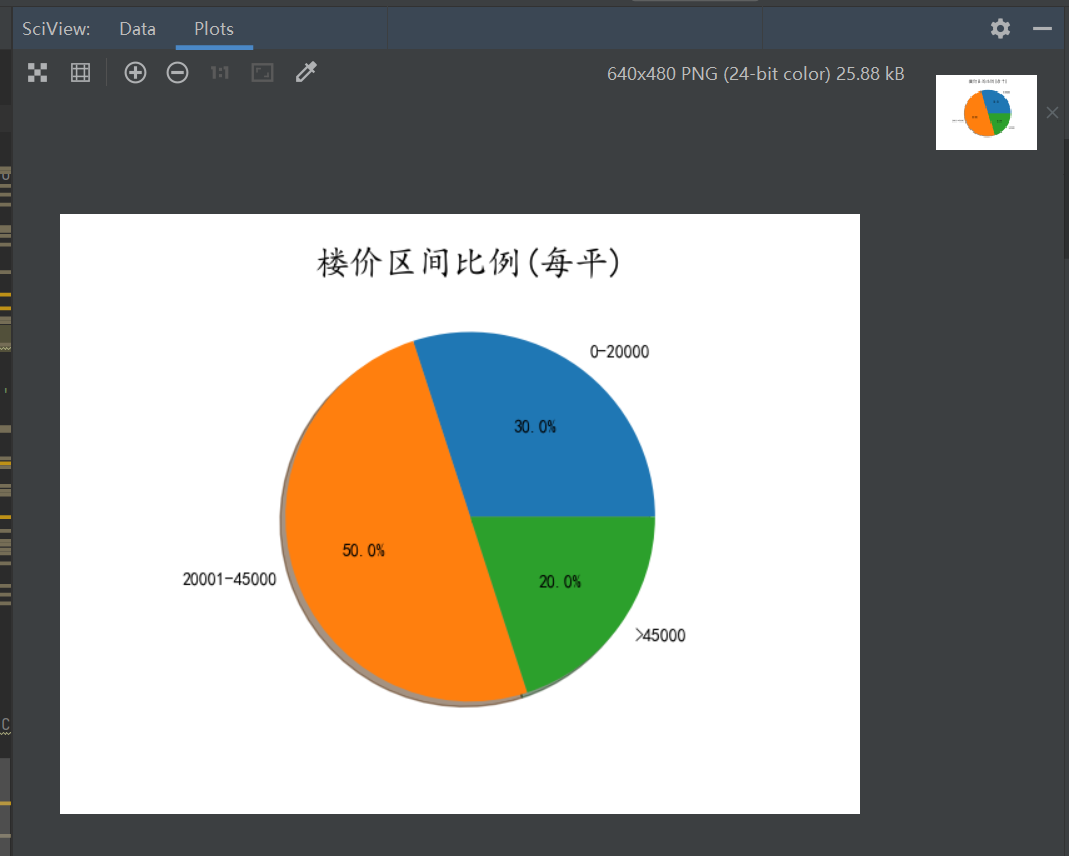


图 9 数据可视化

# 总结

这次的大作业其实对我来说并不轻松，但也不会特别难。自己完成了还挺有成就感的，python确实很有趣，能爬取自己想获得的数据，通过这个大作业，我不止对python，还对html，对http和https协议，传输头等知识都有了更深入的了解和学习。其实一开始选择用BeautifulSoup来进行数据解析是因为当时刚好上课讲完这个我就开始写了，后面的xpath和正则还没开，但是我大作业已经开了，那头都起一大半了就顺势用这个做完了。除了找标签和测试数据麻烦点之外，用着也还可以。后面根据老师讲的可视化加上我网上冲浪翻阅了一部分资料，决定用matplotlib这个插件进行可视化的开发工作，因为我已经有线程的数据，想要一个“开箱即用”的效果，事实证明我选择这个插件是正确的，在可视化这个功能上开发得还算是轻松，简单阅读一下api就可以上手。但这个也引起了我的思考，现在的“快速开发”和“快速上手”都是用别人造的轮子，用别人开发的便利产品，似乎对我本身的编程能力并没有一个很大的锻炼和提高，更像是站在巨人的肩膀上开发。

最后关于加分项这个问题，思考了一下关于这个数据源的必要性，感觉没太大必要每天定时爬取，所以就没有做定时爬取的功能，不太符合实际。可视化可以做一下，所以选了可视化来做。