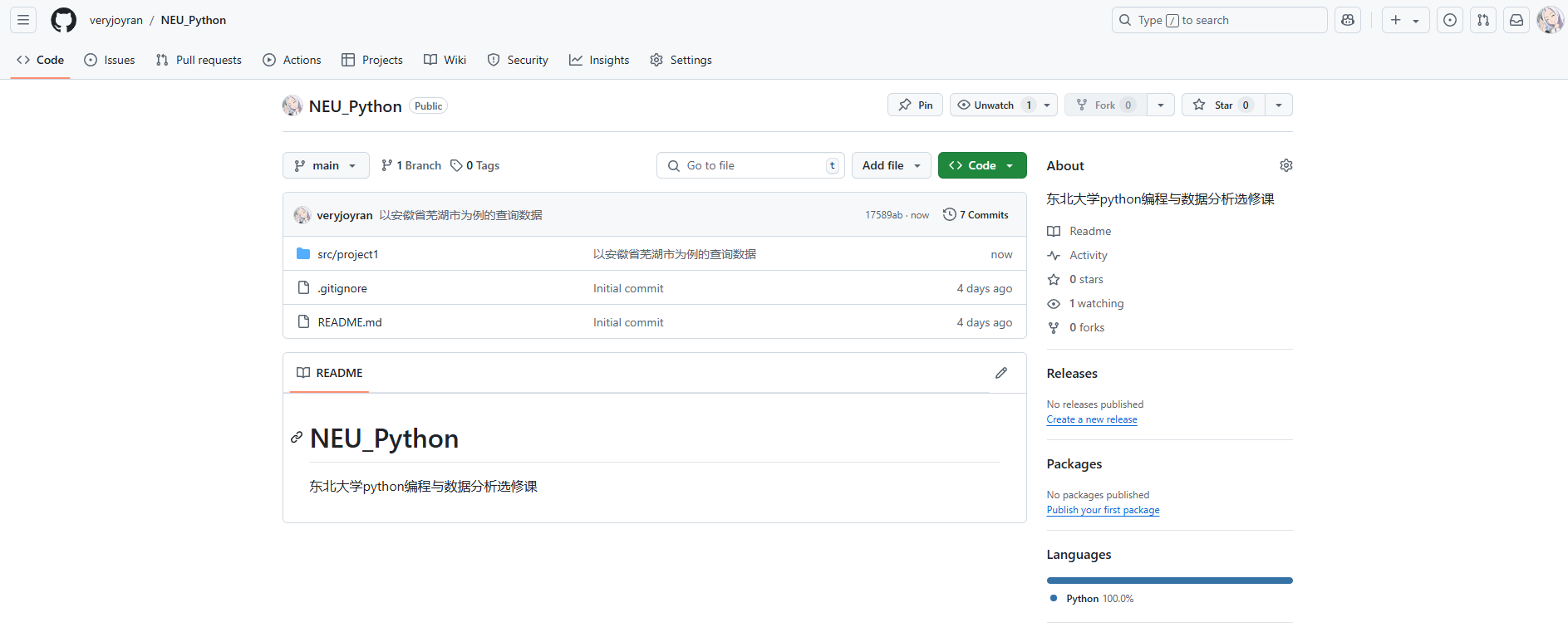
Project3：二手房数据爬取与展示系统 - 客户端-服务器 (Client-Server) 版本

人工智能2202班-刘庆然-20226494

所有代码以及README均已上传至github:

https://github.com/veryjoyran/NEU\_Python/tree/main/src/project3





目录

[一、 问题描述 3](#_Toc186775032)

[二、说明文档 4](#_Toc186775033)

[三、实验结果 7](#_Toc186775034)

[1.启动服务端 7](#_Toc186775035)

[2.启动客户端 7](#_Toc186775036)

[3.客户端操作 7](#_Toc186775037)

[（a）数据爬取 7](#_Toc186775038)

[(b)柱状图显示 9](#_Toc186775039)

[4.自动化测试用例 9](#_Toc186775040)

[5.定时爬取 11](#_Toc186775041)

[四、源码解读 13](#_Toc186775042)

# 问题描述

Project3 需求：

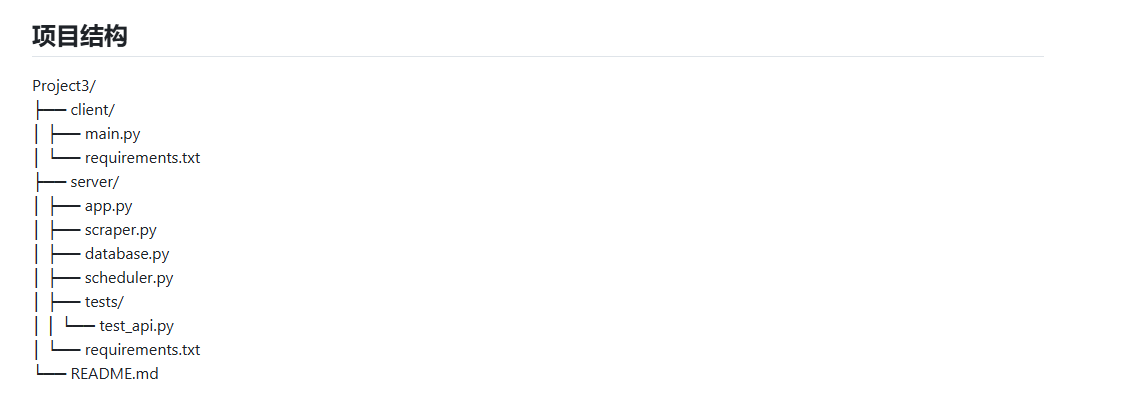
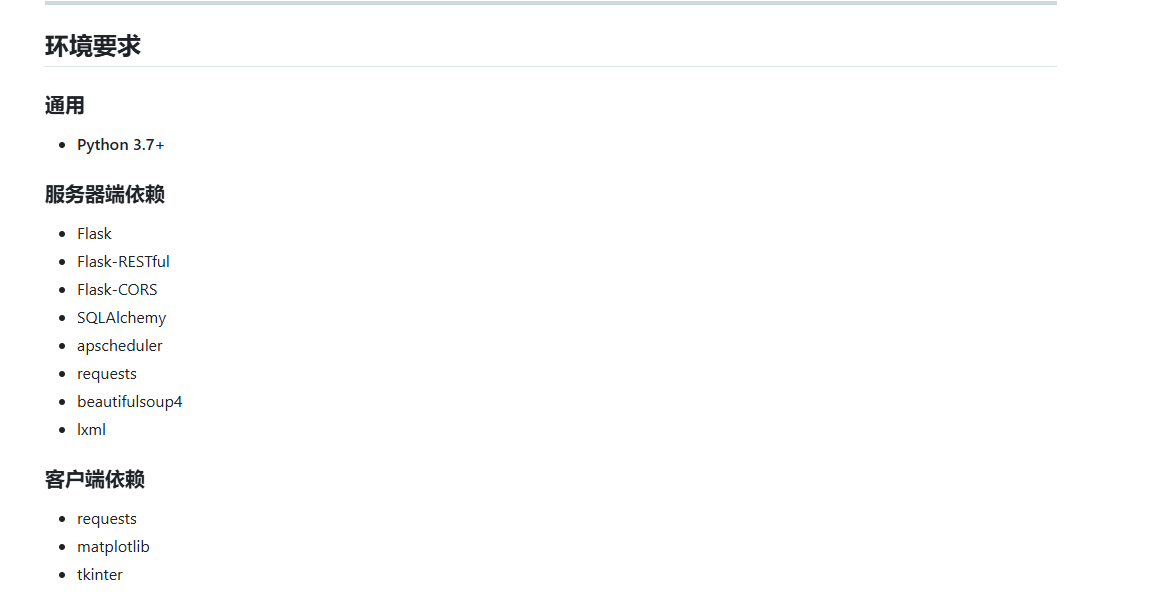
1）开发服务器，重构为CS架构，S端（数据库及信息传输，使用python的web框架），C端可以复用原Tkinter界面（采用restful服务形式调用），也可以自己重构为Web页面；

2）书写自动化测试用例；

3）统计二手房数量，并画出柱状图显示到界面；

4）后台增加更新数据功能，如定时一周（只保存增加新房屋，售出的不删除）抓取数据更新一次。

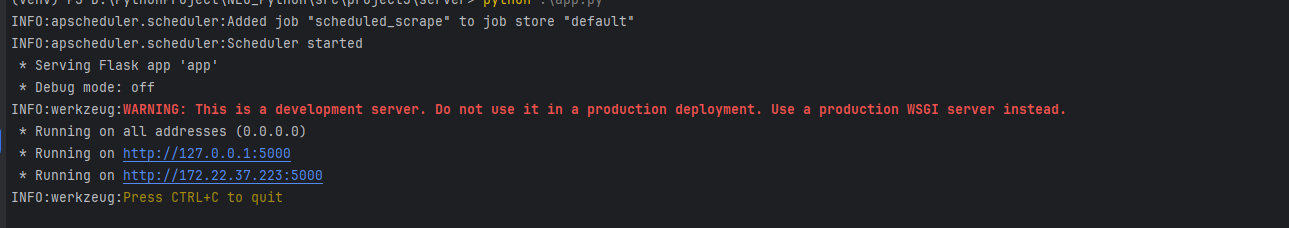
# 二、说明文档



# 三、实验结果

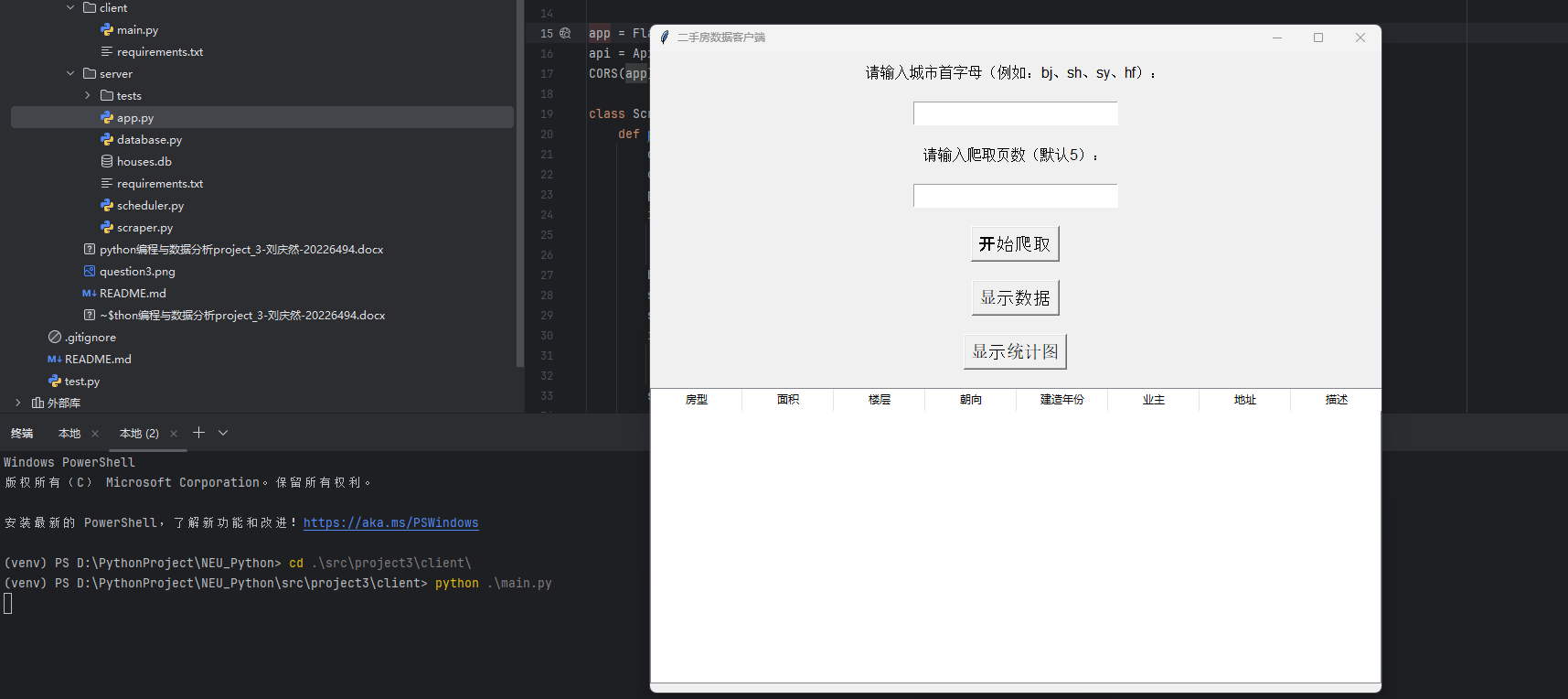
## 1.启动服务端

在安装玩相应依赖之后，在如下图所示的路径启动 python app.py，服务端口运行在 <http://127.0.0.1:5000> 上



## 2.启动客户端

在相应路径启动 python main.py，其中页面复用了project2中的tkinter界面。



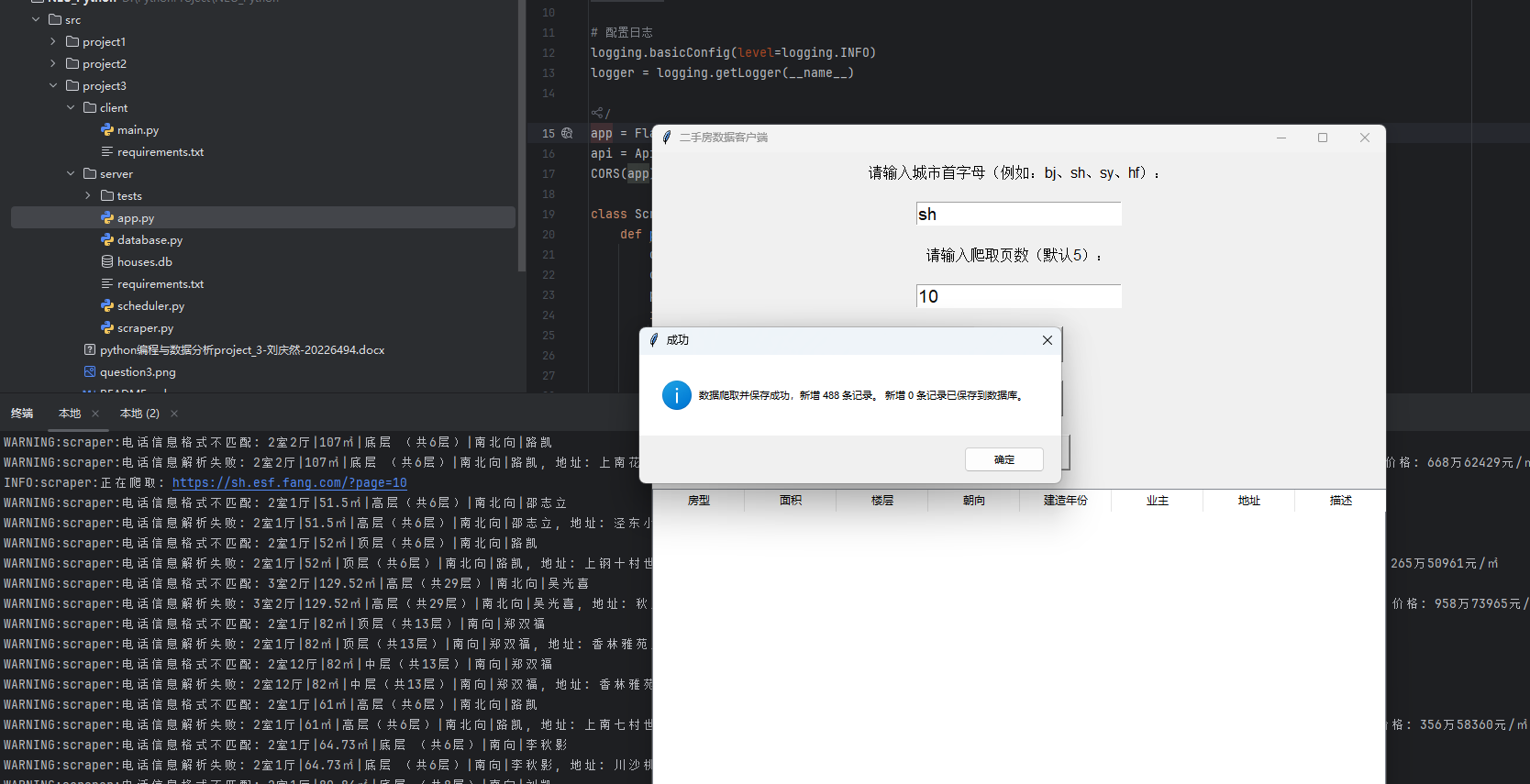
## 3.客户端操作

与project2中相似，但是在这里将爬取数据和将数据保存到数据库的按钮合二为一，在爬取数据之后会自动将数据保存到相应数据库，并新增了显示统计图的按钮。

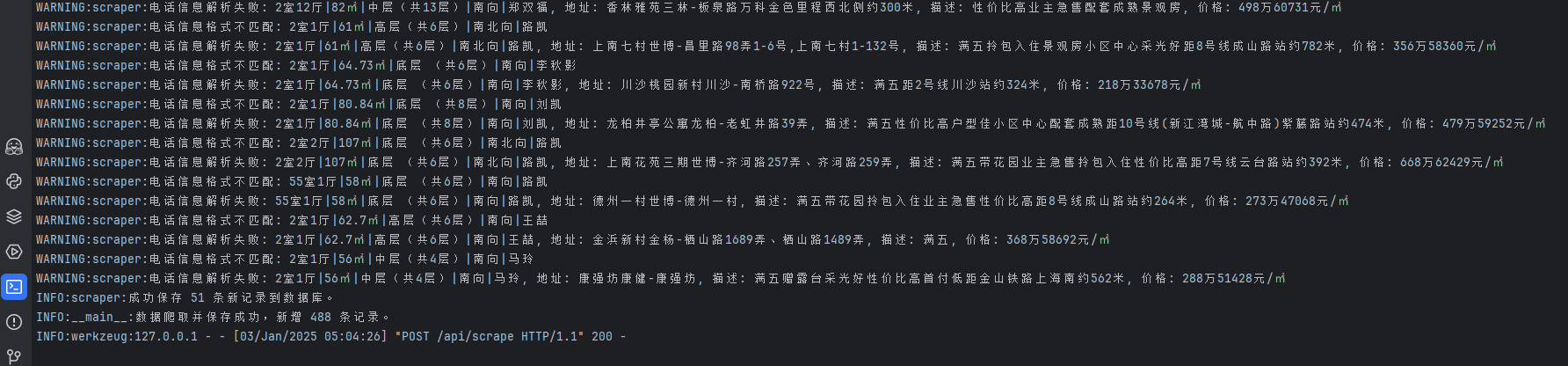
### （a）数据爬取

在输入城市名称之后，按下开始爬取按钮，服务端开始响应：、



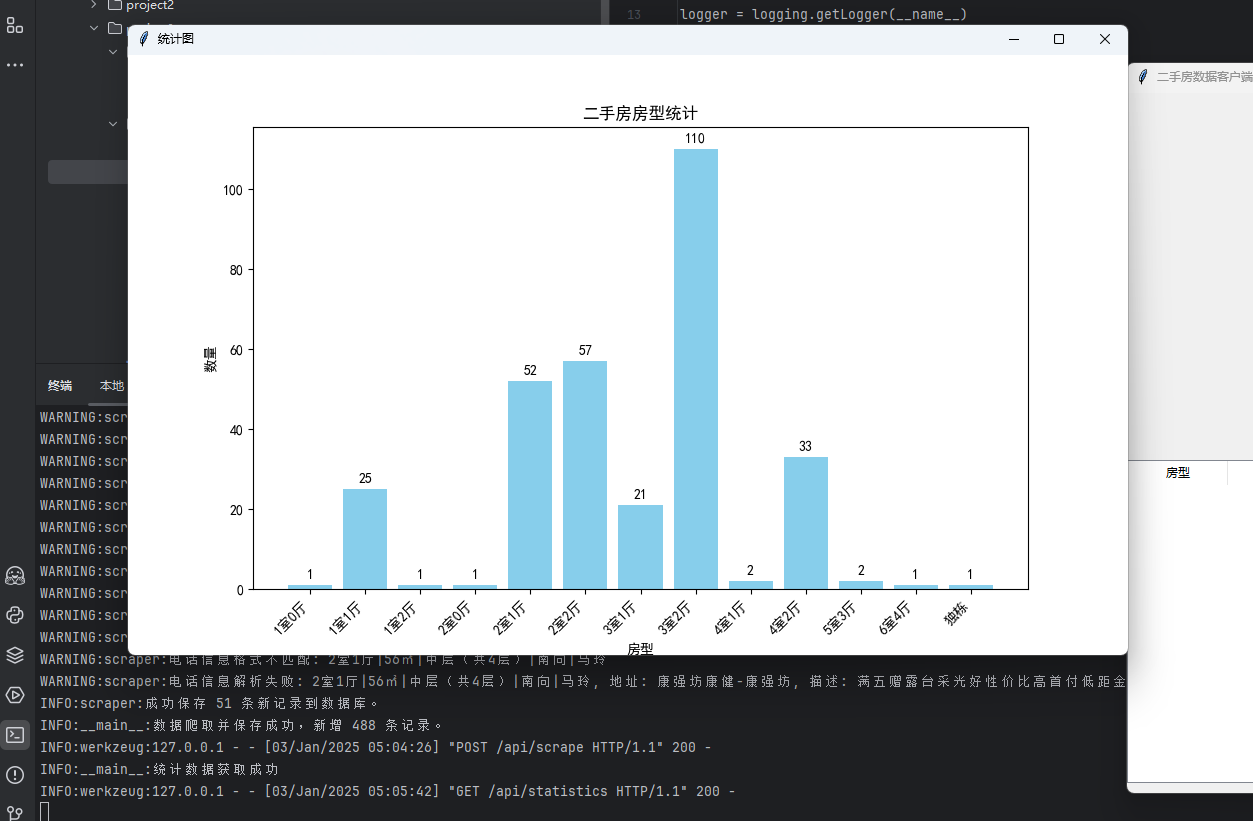


爬取成功弹出相应信息。



### (b)柱状图显示

在此处以房型进行分类显示各个房型的房子有多少个。



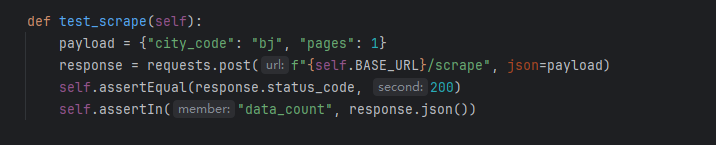
## 4.自动化测试用例

在tests文件下有一个test\_api.py，主要测试三个，分别是测试 /api/scrape 端点，确保数据爬取功能正常；测试 /api/houses 端点，确保能够正确获取存储的房源数据；测试 /api/statistics 端点，确保能够正确返回统计数据。

如下(需确保服务端在运行)：



其中 test\_scrape具体内容如下：



 其中，响应状态码应为 200，表示请求成功。响应 JSON 中应包含 data\_count 字段，表示新增的数据条数。

比如我们现在在app.py中删除 Scrape的data\_count字段





然后重新运行app.py 和 test\_api.py可以看到如下信息：



显示 data\_count 无法没有在返回信息中获取到，与预期不符，弹出错误。

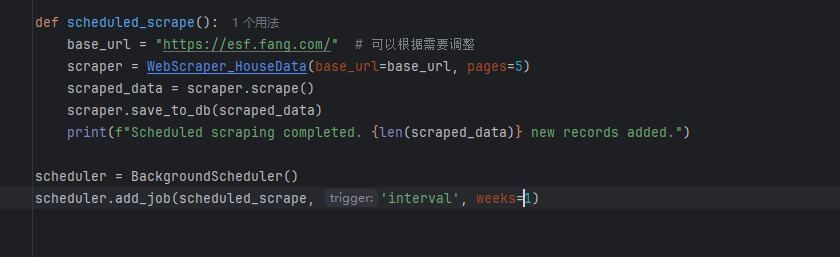
## 5.定时爬取

使用：APScheduler：用于设置和管理定时任务。

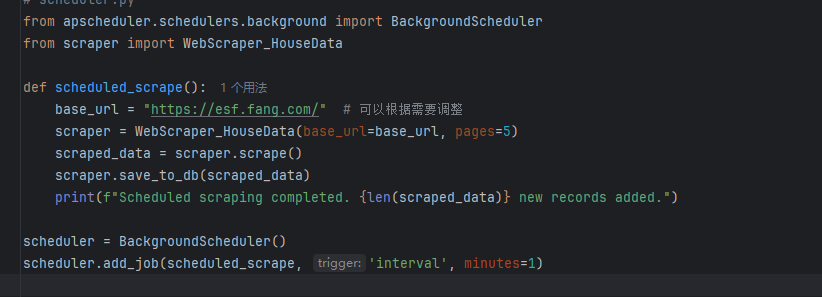
定时爬取：通过 BackgroundScheduler 创建一个后台调度器，每周自动执行一次 scheduled\_scrape 函数，爬取指定网站的二手房数据（默认5页），并将爬取到的数据保存到数据库中。

日志输出：每次定时任务完成后，输出新增记录的数量，便于监控数据更新情况。

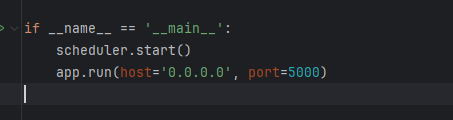
如下：



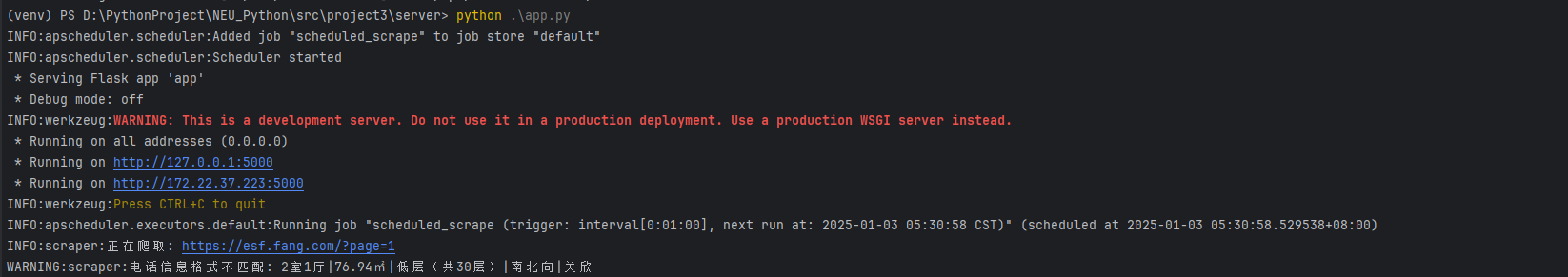
为了测试，我们更改下时间，看看相应输出信息：



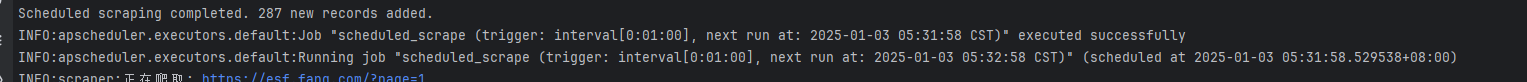
重新运行app.py如下（scheduler在app.py启动时就启动了，具体为 app.py中的主程序入口，先启动定时器，再启动服务：



有如下输出：

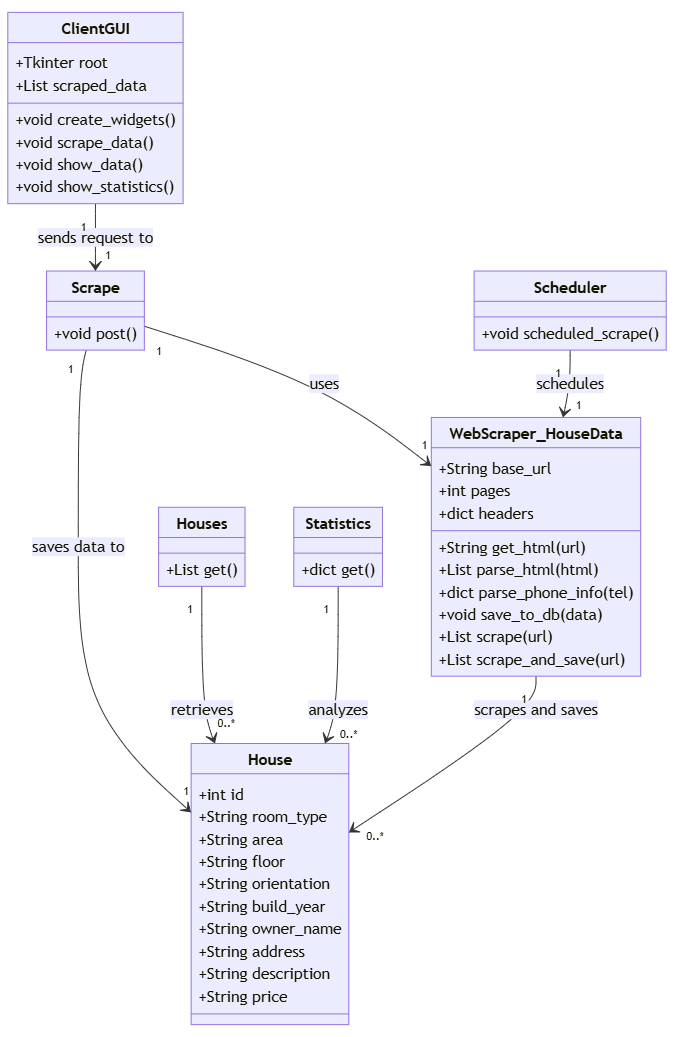


过了一分钟后可以看到：



执行时间差了一分钟，说明定时任务正常启动。

# 四、源码解读



**说明：**

* **ClientGUI**：客户端的图形用户界面，负责与用户交互，提供数据爬取、查看和统计功能。
* **Scrape**：服务器端的爬取功能，处理来自客户端的爬取请求，执行数据爬取并保存到数据库中。
* **Houses**：服务器端的房源数据接口，提供从数据库读取房源数据的功能。
* **Statistics**：服务器端的统计数据接口，提供房源数据的统计分析功能。
* **House**：数据库模型，表示房源数据的各个属性，如房型、面积、楼层等。
* **Database**：数据库管理类，负责与SQLite数据库的连接和会话管理，提供数据的读写操作。
* **Scheduler**：服务器端的定时任务模块，使用 APScheduler 定期调用爬取功能以更新数据库中的数据。
* **TestAPI**：自动化测试类，包含对服务器API端点的测试用例，确保各项功能正常运行。

**关系说明：**

* **ClientGUI** 通过发送请求与 **Scrape**、**Houses**、**Statistics** 进行交互，分别对应数据爬取、数据展示和统计分析功能。
* **Scrape** 通过 **House** 类将爬取到的房源数据保存到数据库中。
* **Scheduler** 定期调用 **Scrape**，确保数据库中的数据保持最新。
* **TestAPI** 测试 **Scrape**、**Houses**、**Statistics** 端点的功能，确保数据爬取、获取和统计分析的正确性。
* **Scrape**、**Houses**、**Statistics** 使用 **Database** 类与数据库进行交互，执行数据的存储和读取操作。

源码及README.md均在压缩包中，并且均包含了详细注释，在此就不再cv一下，感觉没有什么意义，还显得文档冗余，故文档只保留了一些必要说明和功能展示。