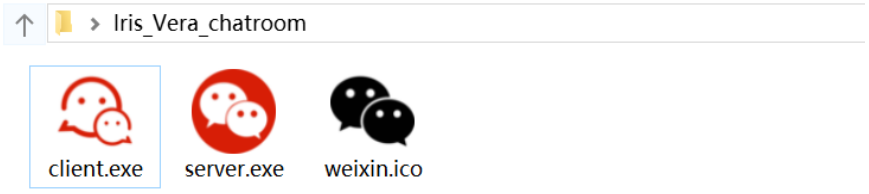
本实验实现了一个多人在线聊天室

功能

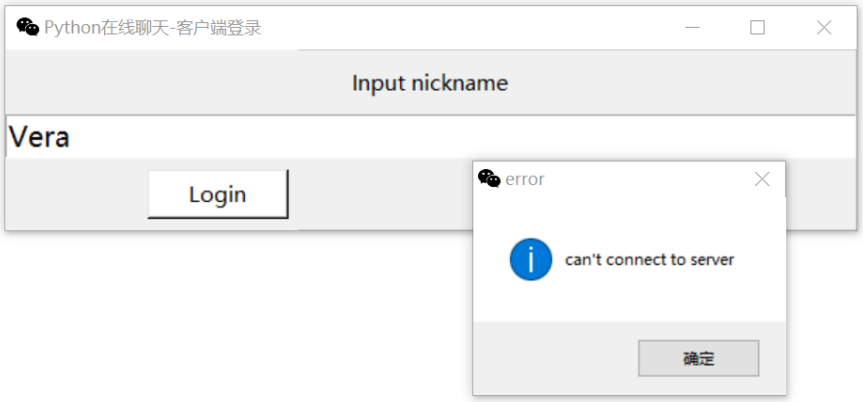
1. 登录及退出控制
   - 登录时进行昵称检查：防止空用户名、重复昵称或 robot
   - 点击按钮或 Enter 登录后跳转至聊天主界面
   - 某个用户登录时显示欢迎信息并通知其他用户
   - 某个用户退出时通知其他用户已离开房间
2. 聊天功能
   - 多人在线聊天，实时可见所有用户发送的消息
   - 通过@robot 可唤醒聊天机器人与其对话
   - 有新消息时自动滚动到底部
   - 支持多行内容（Enter 换行，Ctrl+Enter 或点击按钮发送）
   - 不允许发送空消息（指全部由空白字符组成）
3. 界面设计
   - Tkinter GUI
   - 窗口放大缩小
   - 聊天机器人的消息以蓝色标识
   - 点击按钮有动态显示
   - 用户窗口标题栏个性化设计
   - 标题栏及可执行程序图标设计
4. 异常处理
   - 弹窗提示异常信息：用户名不合要求/发送消息为空
   - 开启多个服务器，提示错误
   - 客户端试图连接到还未开启的服务器，提示错误

运行方法

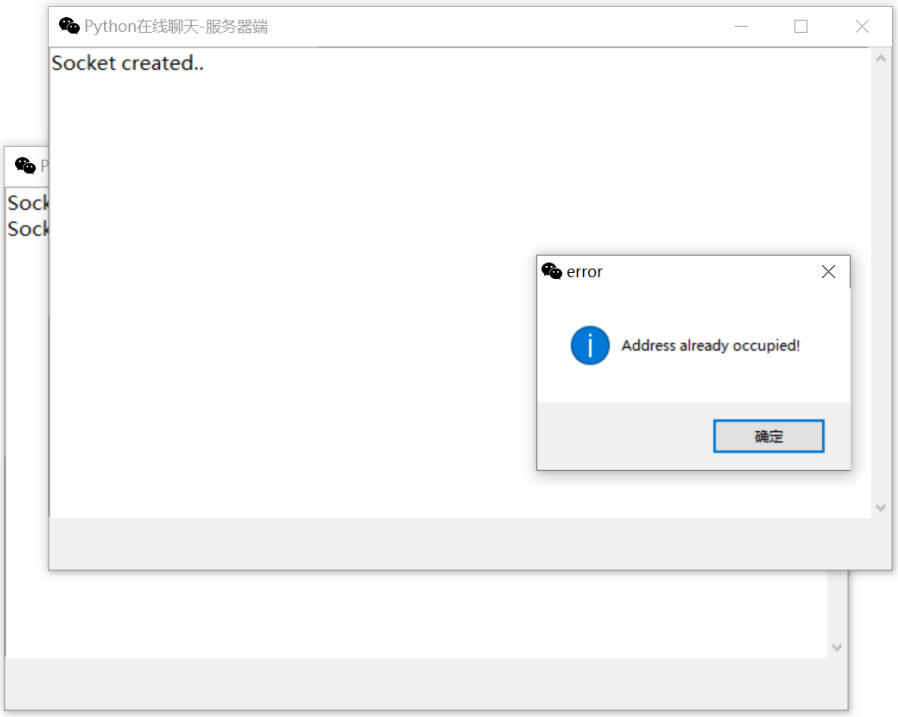
已打包为可执行文件，直接运行 server.exe 与 client.exe 即可。
注意 ico 图标须放在同一目录下，否则可能导致找不到图标文件。



必须先运行 server.exe，显示 socket now listening..之后再运行 client.exe。否则程序弹窗提示 can't connect to server.

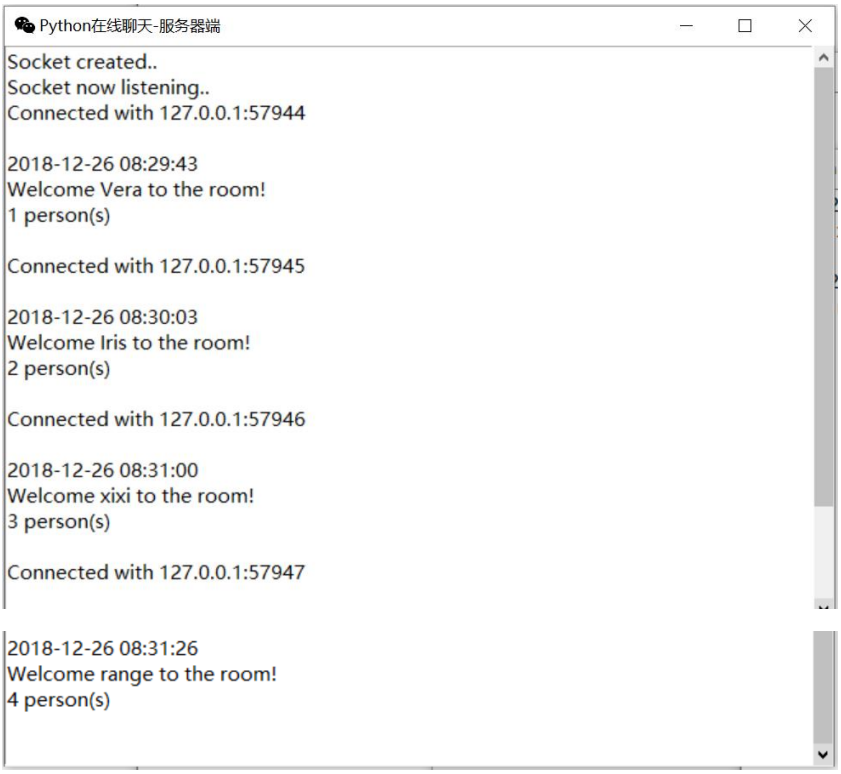


不能运行多个 server，否则程序弹窗提示 Address already occupied.



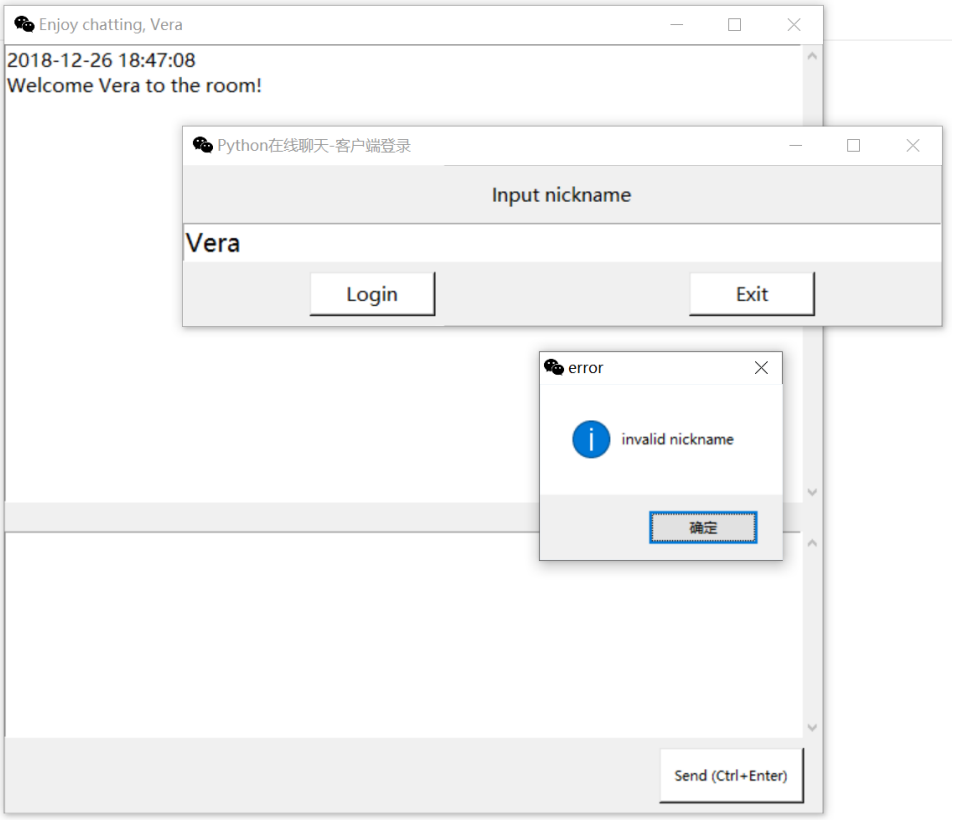
一次只能运行一个 server，但可以运行 N 个 client。

server

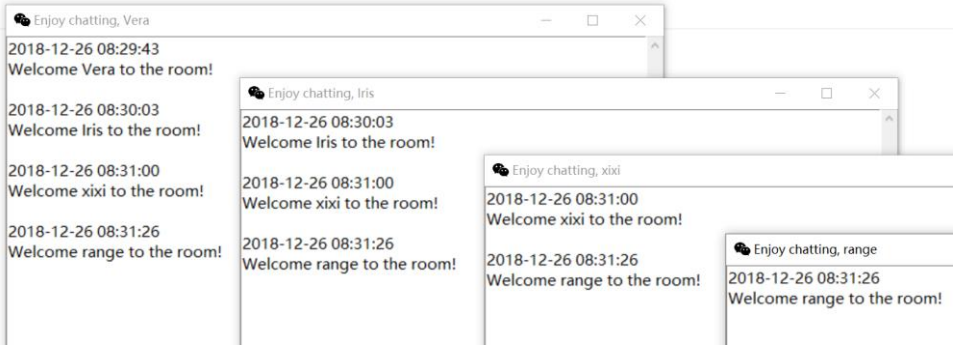


client

不可重复登录

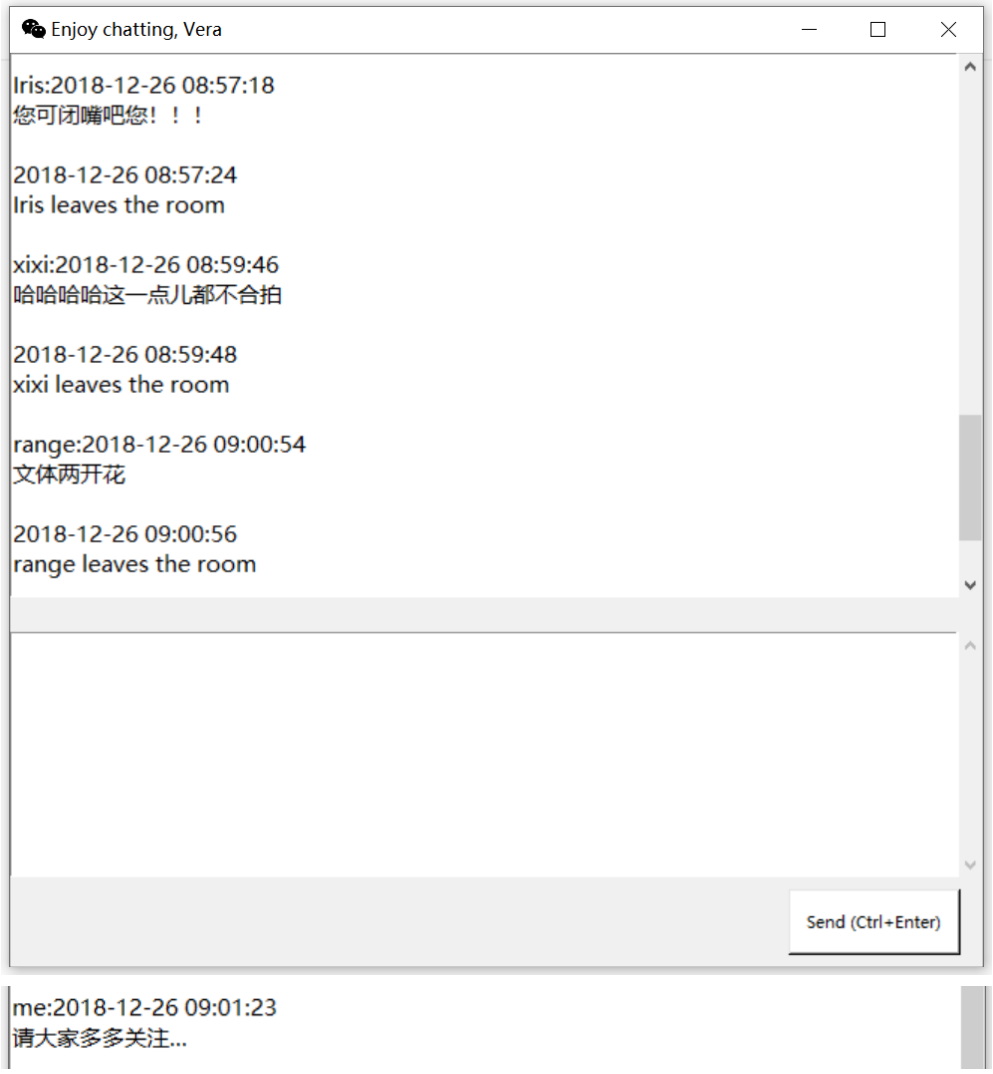


依次登录四个 client



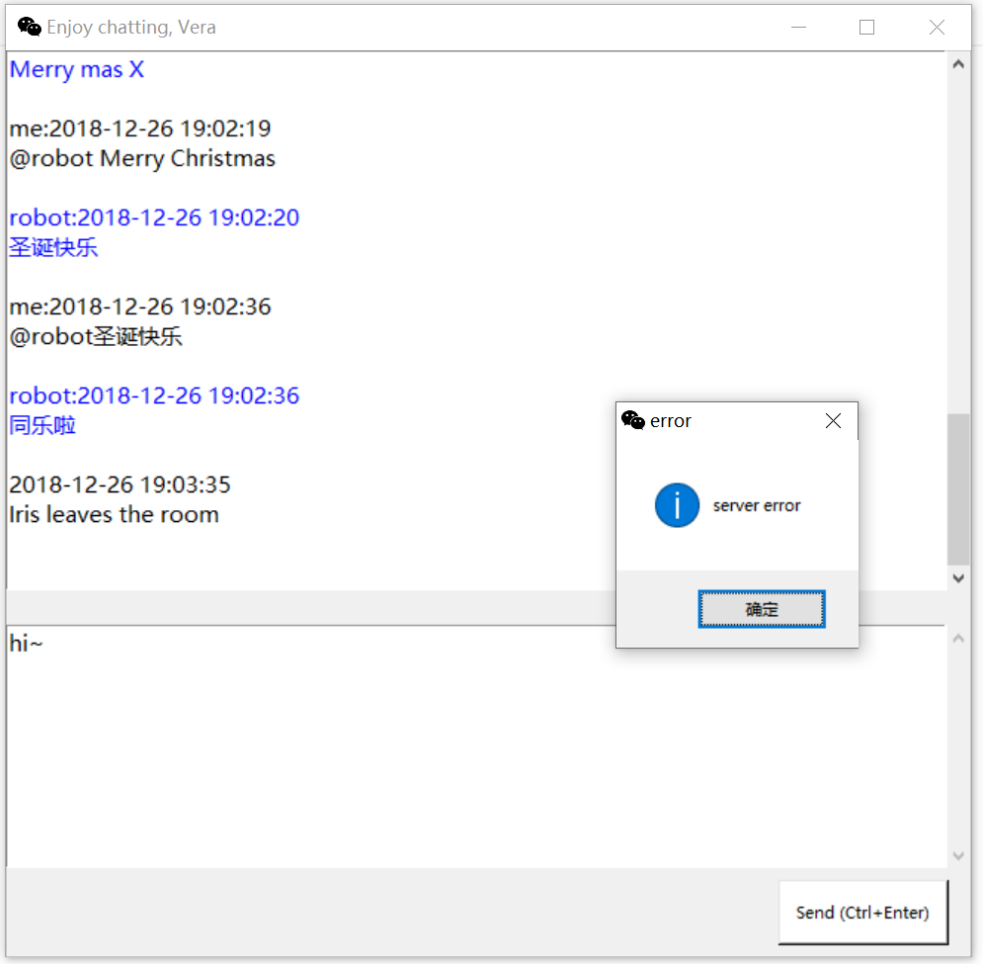
四人聊天 client 界面

依次退出聊天室



四人聊天 server 界面

如果先关掉 server，再发送消息会提示 server error



设计思路

采用 C/S 模式，基于 Socket 编程的方式，使得各个用户通过服务器转发实现聊天的功能。分为两大模块：服务器端模块和客户端模块。

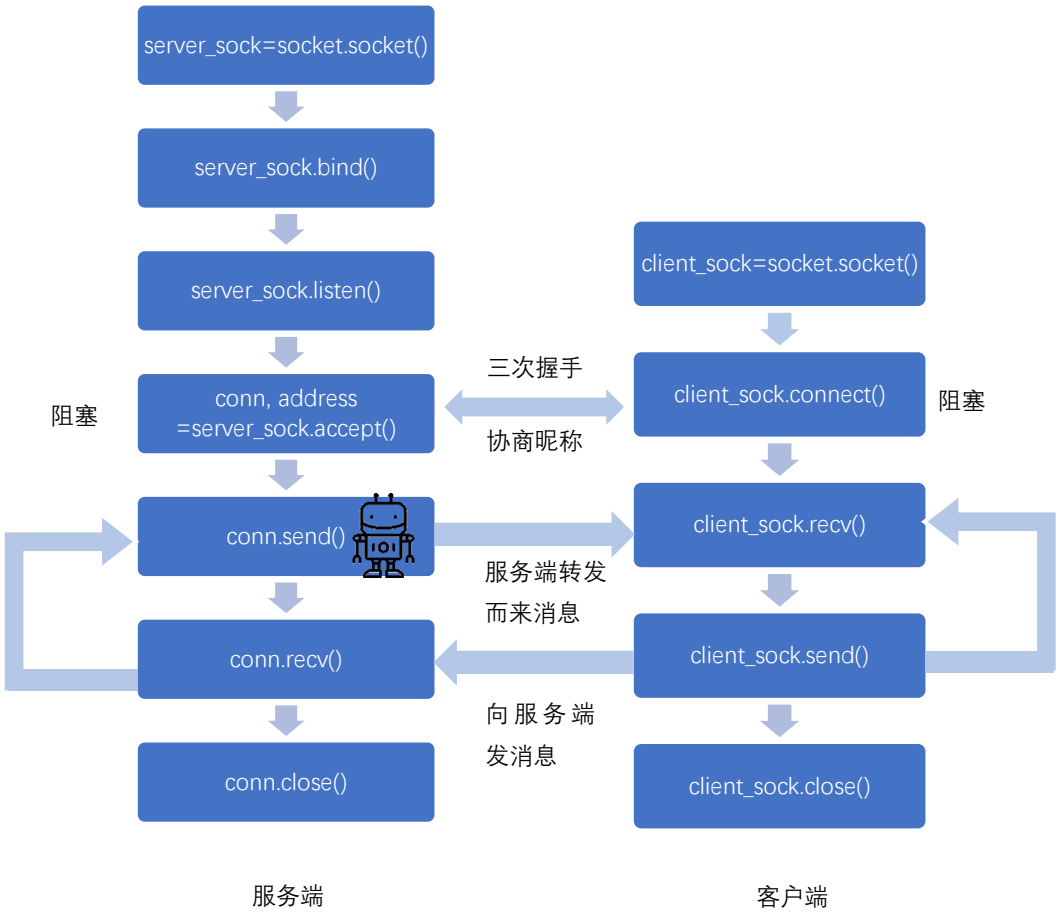
服务器协议解析：

首先建立 socket 套接字，创建成功后持续监听用户请求，并启动两个线程分别负责收发消息。当监听到有客户端连接到服务器端主机 IP 地址和对应端口时，与客户端协商登录信息（昵称），协商成功后绑定其连接，并将其昵称添加到已有昵称列表，广播欢迎信息。

当收到某个客户端发来的消息，简单地将其转发给其他所有用户。同时服务器端通过调用图灵机器人 API，当检测到用户发来的消息中包含"@robot"时，对消息进行解析并传送给机器人处理，将机器人的回复转发给所有用户。服务器检测到客户端连接异常，广播用户离开房间信息。

客户端协议解析：

首先进入登录页面，后台已自动建立到服务器的连接（若连接建立失败则弹窗提示并退出程序），通过已建立的连接与服务器协商昵称，以保证昵称的合法性和唯一性。若登录成功则跳转至聊天主界面，启动两个线程负责收发消息。用户在输入框中输入信息，消息框中实时显示服务器转发而来的消息。用户点击发送按钮或按下 Ctrl+Enter，将消息发送给服务器并清空输入框。通过@robot 唤醒机器人，可收到其自动回复，并以蓝色字体显示。若服务器异常关闭，在下一次试图发送消息时弹窗提示并退出程序。



进一步思考

- 图片的传输
  sock.sendall()只接受 bytes 类型的数据。对于文字消息的处理是依赖于 bytes 与 utf-8 格式的互相转换，但由于利用 Pillow 库读到的二进制图片数据有很多不是 utf-8 字符，必须将其转化为 base64 再传输，而且很多图片比较大，需要循环接收并判断消息边界，这就给信息包的封装与解封装带来了困难。未来可以设计一种消息协议，采用 key-value 形式分别指定消息的形式和消息的内容。
- 私聊的实现
  在协议中增加一个字段用以标识目的用户 ID。
- 数据加密问题
  目前没有实现数据加密功能，客户端和服务器之间信息的传输只是纯粹的二进制流，故面临着被第三方攻击或窃取机密信息的风险。考虑使用信息安全原理课上学到的一些加密算法，如 RSA 公钥密码体制和 DES 加密算法等，每建立一条连接，先让客户端和服务器进行密钥的交换，后续数据都进行加密传输。还可以考虑增加签名认证机制，以预防中间人攻击。
- 添加数据库保存用户登录信息及历史消息
  目前的聊天室是无状态的，而要涉及到保存每个用户的注册登录信息或历史消息就必须依赖数据库。未来可以考虑采用 python+mysql+django 技术栈实现网页版聊天室，服务器端本地存储数据库。