**重庆理工大学**

**智能数据分析程序设计**

**项目报告**

**题 目：**基于Python socket UDP和Tkinter编

写的多功能即时聊天室

**专 业：电子信息工程**

**姓 名：**

**学 号：**

**电子信息工程系**

**二Ｏ二二年 十月 五日**

**目 录**

[一、概述 1](#_Toc151928846)

[二、项目需求分析及方案设计 1](#_Toc151928847)

[2.1 开发环境 1](#_Toc151928848)

[2.2 需求分析 1](#_Toc151928849)

[2.3 方案设计 1](#_Toc151928850)

[2.3.1客户端 1](#_Toc151928851)

[2.3.2服务器 2](#_Toc151928852)

[2.3.3用户界面 2](#_Toc151928853)

[2.3.4网络安全 2](#_Toc151928854)

[2.3.5多用户支持 2](#_Toc151928855)

[三、项目关键技术 2](#_Toc151928856)

[3.1 Python Socket网络编程接口 2](#_Toc151928857)

[3.2 python Tkinter库 2](#_Toc151928858)

[3.3 UDP协议 3](#_Toc151928859)

[3.4 C/S结构（客户端/服务端模式） 3](#_Toc151928860)

[四、项目详细设计 4](#_Toc151928861)

[4.1 服务器功能模块 4](#_Toc151928862)

[4.2 客户端功能模块 4](#_Toc151928863)

[4.2.1 登录子模块 4](#_Toc151928864)

[4.2.2 注册子模块 6](#_Toc151928865)

[4.2.3 聊天子模块 7](#_Toc151928866)

[4.2.4 表情包子模块 9](#_Toc151928867)

[4.2.5 邮件发送与FTP子模块 9](#_Toc151928868)

[五、调试与运行结果 10](#_Toc151928869)

[六、总结 12](#_Toc151928870)

# 一、概述

随着互联网技术的不断发展，网络通讯已经成为最为重要的应用之一。腾讯公司自主研发的聊天软件QQ，以较为明显的优势在市场上得到了广泛而深刻的运用，成为了几乎每台电脑必装软件之一。即使分处两地，人们依然可以通过互联网实现即时通讯。但是在很多情况下由于各方面的限制我们无法连接互联网，只能组建局域网，在不需要与外接通讯，而只想实现局域网内的即时通讯的情况之下，就需要一款与QQ同类型的能够在局域网内运行的聊天软件。

我们可以从网上下载到很多局域网聊天的软件，它们大多运行在Windows平台下，虽然能够实现的功能很多，但是基本都是非开源的。互联网的发展为我们提供了很多的应用，而聊天服务是互联网提供的重要服务之一。本文希望通过程序设计，并且熟练掌握Windows和Mac OS平台下的Socket网络编程技巧，从Socket网络编程中理解TCP/IP协议和UDP协议的具体应用方式，并能应用Tkinter来开发软件界面，便于用户操作。

# 二、项目需求分析及方案设计

2.1 开发环境

处理器：AMD Ryzen 7 5800H / Apple M1

内存：16GB / 8GB

网络环境：局域网

操作系统：Windows / MacOS

开发平台：PyCharm

开发语言：Python

2.2 需求分析

此次设计实现的网络聊天室系统作为局域网的交流工具使用，创建一个支持多人聊天的网络环境，使得多个用户可以在同一个平台上实时聊天。因此要求简洁、便利、信息响应速度快，系统应该能够显示正在聊天的在线用户列表，并且可以让用户给其他用户发送消息。聊天室应具备一定的安全机制，例如账号注册和登录，防止未授权用户的入侵。本系统采用C/S模式，由服务器端程序和客户端程序两部分组成。服务器模块可以实现启动和关闭服务器、建立连接、监听客户端等操作；客户端模块能够完成用户注册、用户登录、私聊与群聊等功能。为提高用户体验，提供一个图形化的聊天界面，便于用户操作和管理。除此之外，本聊天室系统的通信原理基于UDP协议。

2.3 方案设计

2.3.1客户端

客户端是运行在每个用户设备上的程序，它负责与服务器建立连接、发送消息以及接收来自服务器和其他用户的回复消息。客户端可以使用Python的socket模块来创建Socket对象，并使用UDP协议进行数据传输。

2.3.2服务器

服务器是运行在一个固定位置的程序，它负责接收客户端发来的消息并转发给其他客户端。服务器也需要使用Python的socket模块来创建Socket对象，并使用UDP协议进行数据传输。

2.3.3用户界面

用户界面是客户端的一部分，它负责向用户提供一个简单的接口来输入消息并显示接收到的消息。用户界面可以使用Python的Tkinter库来创建窗口、按钮和文本框等组件。

2.3.4网络安全

为了确保数据的安全性，使用用户名和密码来进行身份验证，以防止未授权的用户接入系统。

2.3.5多用户支持

为了支持多个用户同时在线并相互发送消息，在服务器端存储当前在线的用户及其状态信息。当某个用户发送一条消息时，服务器将把该消息转发给所有其他在线用户。

# 三、项目关键技术

## **3.1 Python Socket网络编程接口**

使用标准库中的socket模块来实现。Socket是计算机网络中进行数据交换的基础结构，它可以用来实现不同计算机之间的通信。在Python中，我们可以使用Socket来实现客户机-服务器模式、多播、广播等多种网络通信方式。Python中的Socket分为两种类型：面向连接的TCP Socket和面向无连接的UDP Socket。其中，TCP Socket是可靠的，因为它提供了数据包的确认机制、重传机制以及流量控制和拥塞控制机制，但速度较慢；而UDP Socket则是快速的，但它不提供任何可靠性的保证。使用Python中的Socket，我们可以轻松地实现各种复杂的网络应用，如聊天室、电子邮件、文件传输等。

## **3.2 python Tkinter库**

Tkinter库是Python的标准库，它是Python接口的Tk GUI工具包的标准Python接口，采用模块化设计。Tkinter库提供了一系列GUI控件，如按钮、标签、文本框和滚动条等，可以帮助开发者快速创建出美观、友好的图形用户界面。

Tkinter库的优点包括：

1. 跨平台兼容性好，可以在Windows、Mac OS X和Linux等操作系统上运行。

2. 使用简单，只需要几行代码就可以实现一个简单的图形用户界面。

3. 功能强大，支持各种类型的控件和布局管理器，可以满足大多数的界面需求。

## **3.3 UDP协议**

UDP协议（User Datagram Protocol）是一种传输层协议，它是基于IP协议的一种无连接、不可靠、速度快的数据报协议。UDP协议不需要建立连接就可以发送数据，因此它的速度比TCP协议更快，但可靠性较差。UDP协议由两部分组成：头部和数据。头部包含源地址、目的地址、长度和校验和等信息；数据部分则包含了用户想要传输的实际内容。由于UDP协议不建立连接，因此它没有确认机制和重传机制，如果数据在传输过程中丢失或者出现错误，就无法重新发送或纠正。另外，UDP协议也没有流量控制和拥塞控制机制，这意味着它不能有效地处理大量的并发请求，而且可能会导致网络拥塞。但是，由于UDP协议的速度快、开销小，所以常用于实时性强、数据量较小的应用场景，如视频会议、网络游戏、实时流媒体等。

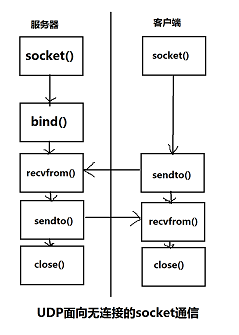


图 1 UDP通信流程

## **3.4 C/S结构（客户端/服务端模式）**

Client/Server结构（C/S结构）是大家熟知的客户机和服务器结构，通过它可以充分利用两端硬件环境的优势，将任务合理分配到Client端和Server端来实现，降低了系统的通讯开销。目前大多数应用软件系统都是Client/Server形式的两层结构。

服务端负责初始化软件运行环境，加载用户信息，为用户之间的文字传送提供平台。客户端由用户操作，可进行注册、登录、密码修改和文字聊天等功能。客户端将用户的操作请求发送至服务端，信息经过服务端处理，再发往对应的客户端。

# 四、项目详细设计

通过需求分析，可以直观地得到网络聊天室的系统结构，方便用户进行操作和使用。依据上述需求分析，可以将网络聊天室系统分为服务器端和客户端两大功能模块。网络聊天室系统的功能模块如图2所示。

网络聊天室

服务端

客户端

启动和关闭

建立连接

监听客户端

用户注册

用户登陆

私发消息

群发消息

图 2 网络聊天室功能模块图

## **4.1 服务器功能模块**

网络聊天室服务器模块主要用于实现服务器对客户端的监听。服务器监听主要是为了获取客户端的相关信息，如端口号，用户发送的信息等，这些都是服务器初始化时需要监听的内容。

1. s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) # 创建socket对象
2. s\_addr = ('127.0.0.1', 9999)
3. s.bind(s\_addr)  # 绑定地址和端口
4. logging.info('UDP Server on %s:%s...', s\_addr[0], s\_addr[1])
5. user = {}  # 存放字典{name:addr}
6. **print**('----------服务器已启动-----------')
7. **print**('Bind UDP on ' + str(s\_addr))
8. **print**('等待客户端数据...')

## **4.2 客户端功能模块**

### 4.2.1 登录子模块

登录子模块主要实现用户的登录功能，用户在登录界面输入用户名和密码后点击登录按钮，如果信息错误，就会显示错误提示“用户名或密码错误！”；登录成功后便进入聊天室界面。登录界面的实现，用了2个Label标签，分别用来显示用户名和密码字段；用了2个Entry标签，用来收集用户输入的数据，即用户名和密码，再用于在数据库中匹配相对应的用户信息；最后，用了3个Button按钮，用来表示相关的单击事件（即Click事件）。

部分核心代码：

1. **def** creatlogin(self):
2. self.fr2 = Frame(self.root)
3. self.fr2.pack()
4. self.fr1 = Frame(self.root)
5. self.fr1.pack(pady=10)
6. self.benner\_list = ['images/benner/banner-1.jpg', 'images/benner/banner-2.jpg',]
7. self.benner\_img = random.choice(self.benner\_list)#随机一张背景图
8. # 图片大小：690x300
9. self.pic = Image.open(self.benner\_img)
10. self.login\_benner = ImageTk.PhotoImage(self.pic)
11. # 标签 图片
12. self.imgLabel = Label(self.fr2, image=self.login\_benner)
13. self.imgLabel.pack()
14. # 标签 用户和密码
15. self.label\_usr = Label(self.fr1, text="用户名：")
16. self.label\_usr.grid(row=0, column=0, pady=10)
17. self.label\_pwd = Label(self.fr1, text="密  码：")
18. self.label\_pwd.grid(row=1, column=0)
19. # 文本框 用户名
20. self.var\_usr\_name = StringVar()
21. self.entry\_name = Entry(self.fr1, textvariable=self.var\_usr\_name)
22. self.entry\_name.grid(row=0, column=1)
23. self.entry\_name.focus\_set()  # 获得焦点
24. # 文本框 密码
25. self.var\_usr\_pwd = StringVar()
26. self.entry\_pwd = Entry(self.fr1, textvariable=self.var\_usr\_pwd, show="\*")
27. self.entry\_pwd.grid(row=1, column=1)
28. self.saved\_msg()
29. self.fr3 = Frame(self.root)
30. self.fr3.pack()
31. self.rd\_login = IntVar()
32. self.rd\_Passwd = IntVar()
33. self.checkboxLogin = Checkbutton(self.fr3, text="自动登录", variable=self.rd\_login)
34. self.checkboxPasswd = Checkbutton(self.fr3, text="记住密码", variable=self.rd\_Passwd)
36. self.la = Label(self.fr3, width=5)
37. self.la.grid(row=0, column=0)
38. self.checkboxLogin.grid(row=0, column=1)
39. self.checkboxPasswd.grid(row=0, column=2)
40. # 登录
41. self.root.bind('<Return>', self.check\_login)  # 绑定回车键
42. self.bt\_login = Button(self.fr3,text="登录",command=**lambda**: self.check\_login())
43. self.bt\_login.grid(row=1, column=1, pady=5)
44. self.bt\_quit = Button(self.fr3,text="退出",command=sys.exit)
45. self.bt\_quit.grid(row=1, column=2)

### 4.2.2 注册子模块

用户注册的实质是向数据库中的表中添加一行数据，如果是新用户或者用户名称不存在，则可以在注册页面上完成相关操作。注册页面的实现，同样用了若干个Label标签和entry标签，但是最重要的控件还是填写注册信息后的确定按钮，即Button按钮。在该Click事件下，同样需要在数据库连接成功的前提下，用SQL语句向USER表中添加用户信息。经过验证，如果在表中添加一行数据，则注册成功，跳转到登录页面。

部分核心代码：

1. **def** creatregister(self):
2. self.fr2 = Frame(self.root)
3. self.fr2.pack()
4. self.fr1 = Frame(self.root)
5. self.fr1.pack(pady=10)
6. self.benner\_list = ['images/benner/banner-1.jpg', 'images/benner/banner-2.jpg', ]
7. self.benner\_img = random.choice(self.benner\_list)#随机一张背景图
8. # 图片大小：690x300
9. self.pic = Image.open(self.benner\_img)
10. self.register\_benner = ImageTk.PhotoImage(self.pic)
11. # 标签 图片
12. self.imgLabel = Label(self.fr2, image=self.register\_benner)
13. self.imgLabel.pack()
14. # 标签 用户和密码
15. self.label\_usr = Label(self.fr1, text="用户名：")
16. self.label\_usr.grid(row=0, column=0)
17. self.label\_pwd = Label(self.fr1, text="密  码：")
18. self.label\_pwd.grid(row=1, column=0, pady=5)
19. self.label\_repwd = Label(self.fr1, text="确认密码：")
20. self.label\_repwd.grid(row=2, column=0)
21. # 文本框 用户名
22. self.var\_usr\_name = StringVar()
23. self.entry\_name=Entry(self.fr1,textvariable=self.var\_usr\_name)
24. self.entry\_name.grid(row=0, column=1)
25. self.entry\_name.focus\_set()  # 获得焦点
26. self.docheck1 = self.entry\_name.register(self.usercheck)  # 自带验证功能，usercheck自定义函数
27. self.entry\_name.config(validate='all', validatecommand=(self.docheck1, '%P'))
28. # 文本框 密码
29. self.var\_usr\_pwd = StringVar()
30. self.entry\_pwd = Entry(self.fr1, textvariable=self.var\_usr\_pwd, show="\*")
31. self.entry\_pwd.grid(row=1, column=1)
32. self.docheck2 = self.entry\_pwd.register(self.passwordcheck)
33. self.entry\_pwd.config(validate='all', validatecommand=(self.docheck2, '%d', '%S'))
34. # 文本框 确认密码
35. self.var\_usr\_repwd = StringVar()
36. self.entry\_repwd = Entry(self.fr1, textvariable=self.var\_usr\_repwd, show="\*")
37. self.entry\_repwd.grid(row=2, column=1)
38. self.fr3 = Frame(self.root)
39. self.fr3.pack()
40. # 注册
41. self.root.bind('<Return>', self.reg)  # 绑定回车键
42. self.bt\_register = Button(self.fr3, text="注册", command=**lambda**: self.reg())
43. self.bt\_register.grid(row=1, column=1, pady=5, padx=35)
44. # self.la = Label(self.fr3, width=5)
45. # self.la.grid(row=0, column=0)
46. self.bt\_quit = Button(self.fr3, text="退出",command=sys.exit)
47. self.bt\_quit.grid(row=1, column=2)

### 4.2.3 聊天子模块

聊天子模块里包含私聊和群聊功能。私聊通过单击聊天室页面上在线好友的名称，即可实现与好友私聊；而群聊功能，则是单击页面上的群发按钮，进而消息内容可以被聊天室的所有在线用户看到。聊天室页面的设计用到了1个Treeview控件，用来显示当前登录进来聊天界面的用户名称；用了1个Frame控件，用来显示当前聊天内容和接收用户输入的聊天信息；有8个Button按钮，表情包按钮用于发送标签，发送按钮用于发送消息，群发按钮用于群发消息，私发按钮用于与其他用户私聊，传文件按钮用于发送文件等。

部分核心代码：

1. **def** chat(self, usename):
2. self.name = usename
3. self.root.title('聊天室--用户名:' + self.name)
4. sw = self.root.winfo\_screenwidth()  # 计算水平距离
5. sh = self.root.winfo\_screenheight()  # 计算垂直距离
6. w = 1120  # 宽
7. h = 720  # 高
8. x = (sw - w) / 2
9. y = (sh - h) / 2
10. self.root.geometry("%dx%d+%d+%d" % (w, h, (x + 160), y))
11. self.root.iconbitmap(r'images/icon/chat.ico')  # 设置左上角窗口图标
12. self.root.resizable(0, 0)  # 窗口设置为不可放大缩小
13. # 告诉操作系统使用程序自身的dpi适配
14. ctypes.windll.shcore.SetProcessDpiAwareness(1)
15. # 获取屏幕的缩放因子
16. ScaleFactor = ctypes.windll.shcore.GetScaleFactorForDevice(0)
17. # 设置程序缩放
18. self.root.tk.call('tk', 'scaling', ScaleFactor / 75)
19. self.root.resizable(1, 1)  # 窗口设置为不可放大缩小
20. self.scr1 = scrolledtext.ScrolledText(self.root, height=18, font=('黑体', 13))
21. self.scr1.tag\_config('green', foreground='#008C00', font=('微软雅黑', 10))  # 设置组件字体颜色
22. self.scr1.tag\_config('red', foreground='red')
23. self.scr1.tag\_config('zise', foreground='#aaaaff')
24. self.scr1.tag\_config('shengzise', foreground='#9d4cff')
25. self.scr1.tag\_config('chengse', foreground='#ff7f27')
26. # 创建树形列表
27. self.fri\_list = Treeview(self.root, height=30, show="tree")
28. self.fri\_list.insert('', 0, 'online\_user', text='在线用户'.center(10, '-'), values=("1"), tags='在线用户')
29. **if** self.name **not** **in** self.fri\_list.get\_children():#如果不在列表中
30. self.fri\_list.insert('', 1, 'me', text=self.name.center(24), values=("1"), tags='自己')  # 自己在列表中颜色为红色
31. self.fri\_list.grid(row=1, column=2, rowspan=7, sticky=N)
32. self.fri\_list.tag\_configure('在线用户', foreground='#aa5500', font=('黑体', 13))  # 设置组件字体颜色
33. self.fri\_list.tag\_configure('自己', foreground='red', font=('微软雅黑', 10))  # 设置组件字体颜色
34. self.fri\_list.tag\_configure('其他用户', font=('微软雅黑', 10))  # 设置组件字体颜色
35. self.scr1.grid(row=1, column=1)
36. l0 = Label(self.root, text='')
37. l0.grid(row=2)
38. l1 = Label(self.root, text='下框输入你要的发送的内容：')
39. l1.grid(row=3, column=1)
40. self.scr2 = scrolledtext.ScrolledText(self.root, height=6, font=('黑体', 13))
41. self.scr2.grid(row=4, column=1)
42. l2 = Label(self.root, text='')
43. l2.grid(row=5)
44. tf = Frame(self.root)
45. tf.grid(row=6, column=1)
46. obj\_emoji = Emoji(self.root, self.send\_mark)
47. chat = ChatClient(self.name, self.scr1, self.scr2, self.fri\_list, obj\_emoji)

### 4.2.4 表情包子模块

用于发送表情包，关联在聊天界面的Button按钮。

部分核心代码：

1. **class** Emoji(object):
2. **def** \_\_init\_\_(self,root,send\_mark):
3. self.root=root
4. self.emoji\_img()
5. self.send\_mark=send\_mark
6. self.ee = 0
7. # 用字典将标记与表情图片一一对应, 用于后面接收标记判断表情贴图
8. self.dics = {}
9. self.bb\_list=[]
10. self.stickers\_code=['[aa\*\*]','[bb\*\*]','[cc\*\*]','[dd\*\*]','[ee\*\*]','[ff\*\*]','[gg\*\*]','[hh\*\*]','[ii\*\*]','[jj\*\*]','[kk\*\*]','[ll\*\*]','[mm\*\*]','[nn\*\*]','[oo\*\*]','[pp\*\*]','[qq\*\*]','[rr\*\*]','[ss\*\*]','[tt\*\*]','[uu\*\*]','[vv\*\*]','[ww\*\*]','[xx\*\*]','[yy\*\*]','[zz\*\*]','[aaa\*\*]','[bbb\*\*]']
11. **print**(len(self.stickers\_code))
12. **for** index,code **in** enumerate(self.stickers\_code):
13. self.dics[code]=self.pic\_list[index]
14. self.bb\_list.append(f"self.bb{index+1}")
15. **print**(self.bb\_list)
16. **print**(len(self.bb\_list))
17. **def** emoji\_img(self):
18. emoji\_path = 'images/emoji/'
19. filelist = os.listdir(emoji\_path)  # 该方法列出path路径下的文件和文件夹列表
20. self.pic\_list=[]
21. **for** i **in** range(0,28):
22. self.pic\_list.append(ImageTk.PhotoImage(file=emoji\_path + filelist[i]))

### 4.2.5 邮件发送与FTP子模块

邮件发送：以QQ邮箱为例，可直接跳转到QQ邮箱，关联在聊天界面的Button按钮。部分核心代码：

1. **def** open\_url(self):
2. url = 'https://mail.qq.com/'
3. webbrowser.open(url)

FTP：提供了一个接口，可用于接入其他文件传输服务关联在聊天界面的Button按钮。部分核心代码：

1. **def** demo(self):
2. messagebox.showinfo(title='提示',message='前面的区域以后再来探索吧')

# 五、调试与运行结果

注册页面：



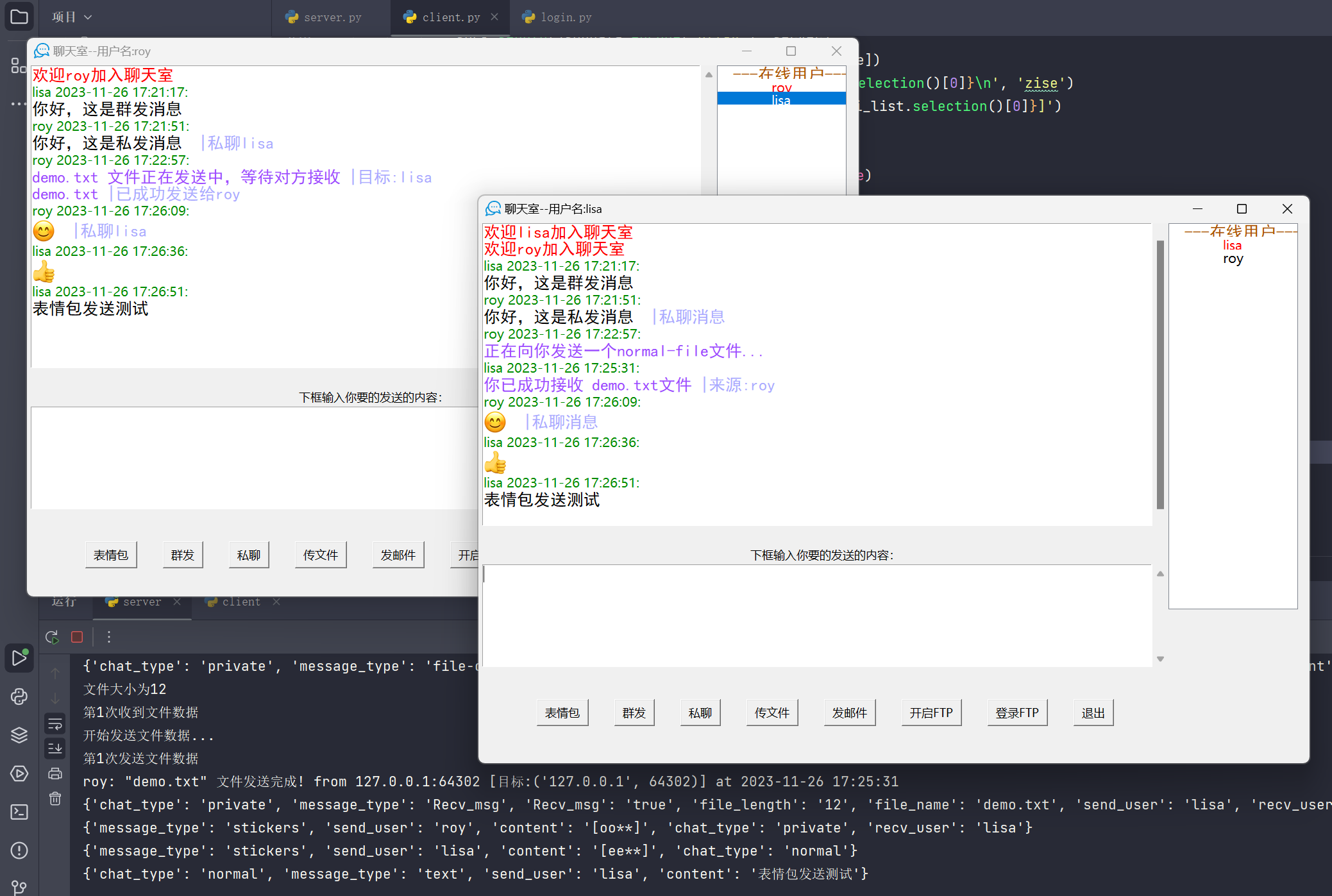
登陆页面：



聊天与文件发送：



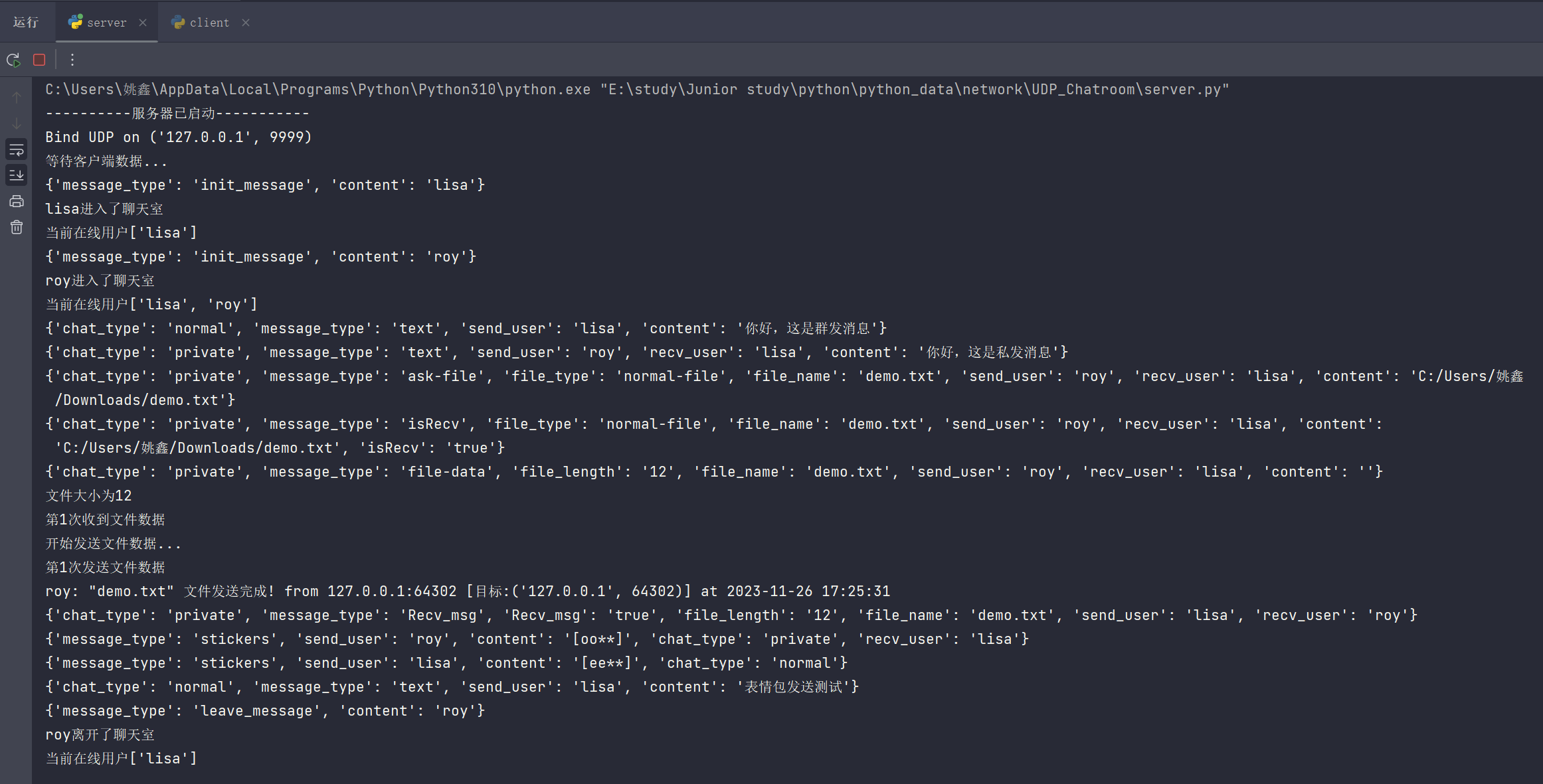
表情包发送；



邮件发送功能点击后将跳转到QQ邮箱，同时本聊天室提供接口可接入其他的邮箱，FTP功能，聊天室提供接口可接入其他FTP程序



服务端后台可实现对客户端的监听



# 六、总结

在这次实验中，我们学习了如何使用Python和UDP协议来构建一个简单的聊天室系统。

首先，我们在客户端创建了一个Socket对象，并使用UDP协议与服务器建立了连接。然后，我们在服务器端也创建了一个Socket对象，并监听客户端的连接请求。接着，我们设计了一个简单的用户界面，允许用户输入消息并向服务器发送。

在服务器端，我们收到了客户端发送的消息后，将其转发给了所有其他在线的客户端。我们也实现了简单的安全性措施，比如验证用户的身份。

通过这次实验，我们了解了Python和UDP协议的基本原理，并学会了如何利用它们来构建一个简单的聊天室系统。我们也发现，虽然UDP协议简单快速，但由于其不可靠性和缺乏流量控制机制，因此不适合在高并发和大流量的场景中使用。

虽然目前的局域网聊天软件种类繁多，几乎能满足各个用户的需求，但是依然存在很多可以改进的地方。先从功能方面来看，现在的局域网聊天软件几乎都有即时聊天、文件传输、语音、视频、表情等功能，但是这些功能在使用的时候可能会出现很多问题。比如聊天的时候会出现数据丢失，传送文件的时候传输速度的快慢。这些情况将来一定能够得到合理的解决，提供给用户最舒适的人机交互。再从开发平台来看，大部分的软件都只能支持单系统，比如能在Windows平台下运行的软件，无法在MacOS平台下正常运行。随着IT行业的迅猛发展，支持多系统的局域网聊天软件会越来越多。这样使用不同的操作系统的用户可以使用同一款软件进行即时通讯。最后从软件的开发环境来看，从面向对象语言产生以来，由于面向对象能做界面，而界面能更好的实现人机交互，所以被广泛应用。