

# Python程序设计大作业实验报告

——刘滨瑞 2021012579 未央-水木12

## 功能实现

本聊天室软件成功实现了以下功能：

**所有基本功能：**

- 账户注册与登录功能
- 聊天室聊天功能
- 好友私聊功能
- 登出功能
- 保存聊天记录功能（数据库）

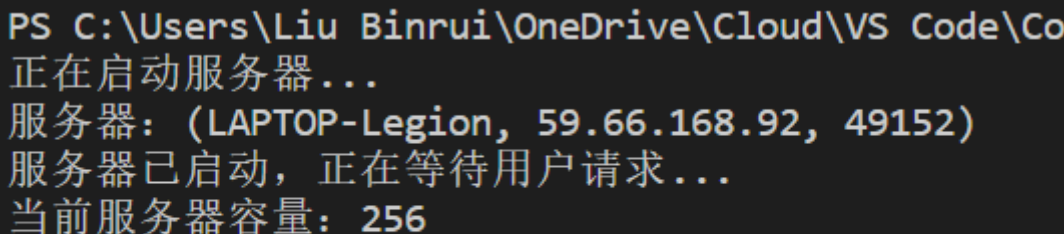
**下列提高功能：**

- 图形化用户界面
- 数据库信息储存
- 查看聊天记录

## 软件演示

下面展示软件的部分功能。 部分不便以截图形式展示的功能（如登出）还望用户自行探索。

- 服务器启动

A terminal window with a dark background and light-colored text. The text shows the command prompt 'PS C:\Users\Liu Binrui\OneDrive\Cloud\VS Code\Co' followed by the output '正在启动服务器...' (Starting server...). Then, the server address is displayed as '服务器: (LAPTOP-Legion, 59.66.168.92, 49152)'. Next, it says '服务器已启动, 正在等待用户请求...' (Server started, waiting for user requests...). Finally, it shows '当前服务器容量: 256' (Current server capacity: 256). A cursor is visible at the end of the last line.

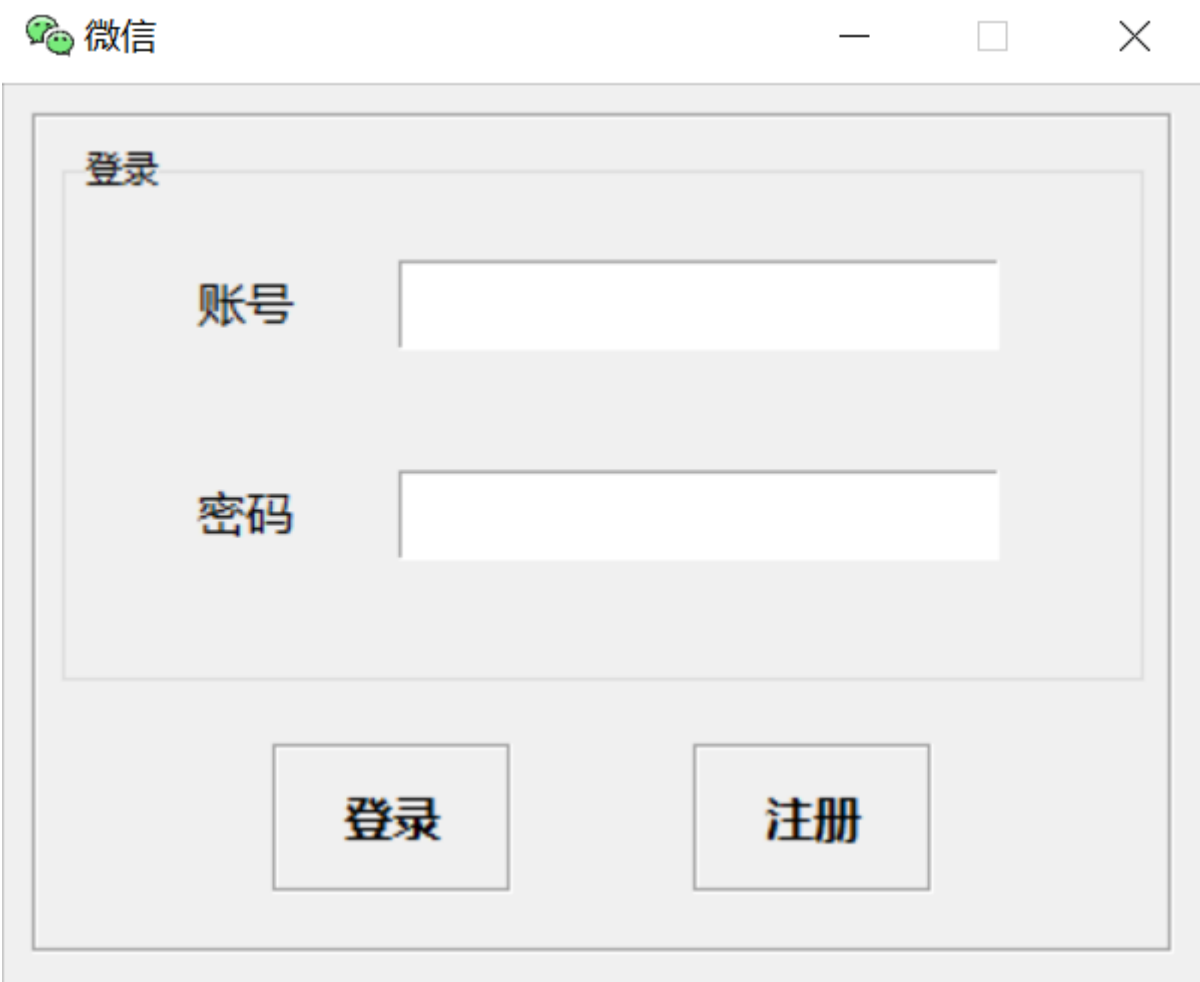
```
PS C:\Users\Liu Binrui\OneDrive\Cloud\VS Code\Co
正在启动服务器...
服务器: (LAPTOP-Legion, 59.66.168.92, 49152)
服务器已启动, 正在等待用户请求...
当前服务器容量: 256
```

- 客户端启动并连接服务器

```
PS C:\Users\Liu Binrui\OneDrive\Cloud\VS Code\Codes\Python\Wechat> & D:/Python/Python310/python.exe "c:/Users/Liu Binrui/OneDrive/Cloud/VS Code/Codes/Python/Wechat/server/main.py"
正在启动服务器...
服务器: (LAPTOP-Legion, 59.66.168.92, 49152)
服务器已启动, 正在等待用户请求...
当前服务器容量: 256
('59.66.168.92', 50846)已连接!
主编号: 488
副编号: 492
█
```

```
PS C:\Users\Liu Binrui\OneDrive\Cloud\VS Code\Codes\Python\Wechat> & D:/Python/Python310/python.exe "c:/Users/Liu Binrui/OneDrive/Cloud/VS Code/Codes/Python/Wechat/client/main.py"
客户端: (LAPTOP-Legion, 59.66.168.92, 50846)
成功连接服务器! 服务器为 (LAPTOP-Legion, 59.66.168.92, 49152)
当前客户端主编号: 488
当前客户端副编号: 492
正在接收消息...
客户端工作状态: 正常
█
```

- 登录界面



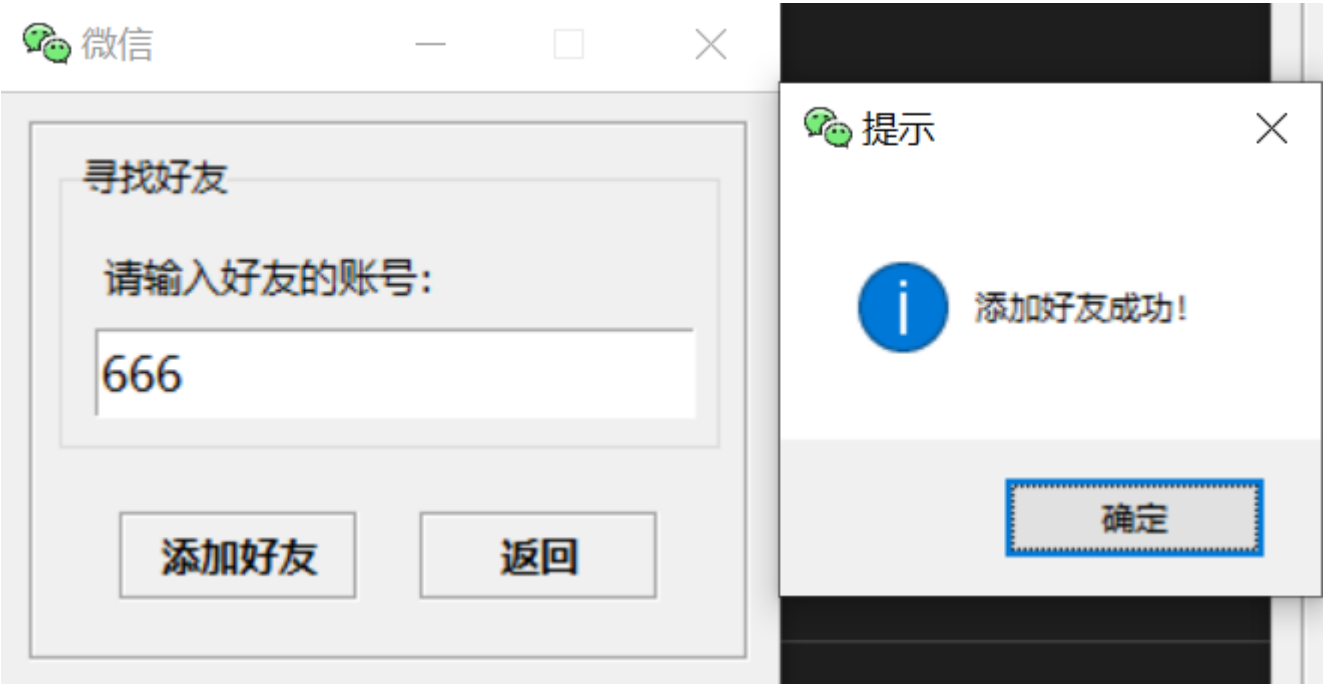
- 账号注册



- 聊天目录



- 添加好友



• 好友私聊



• 聊天室聊天



- 查看聊天记录



- 服务器和客户端后台

```
正在接收消息...
客户端工作状态: 正常
尝试登录...
账号、密码已上传服务器!
登录成功!
接收完成!
已加载GUI缓冲!
正在接收消息...
收到一条消息!
(4)Dave:
1

尝试发送一条消息...
消息已上传服务器!
接收完成!
发送成功!
已加载GUI缓冲!
正在接收消息...
收到一条消息!
(10086)Administrator:
Hello!

尝试登出...
账号已登出!
尝试登录...
账号、密码已上传服务器!
登录成功!
查询聊天室成员信息中...
查询成功!
尝试加入一个聊天室...
加入聊天室成功!
查询聊天室成员信息中...
查询成功!
接收完成!
已加载GUI缓冲!
正在接收消息...
收到一条消息!
(4)Dave:
Hello!

尝试发送一条消息...
消息已上传服务器!
发送成功!
接收完成!
已加载GUI缓冲!
正在接收消息...
收到一条消息!
(666)QB:
Yee~

登录成功!
4已登录!
('59.66.168.92', 62945)已连接!
主编号: 508
副编号: 512
508号客户端提交了一个登录申请...
登录账号: 10086
登录密码: admin
登录成功!
10086已登录!
488号客户端提交了一个消息...
发送消息成功!
508号客户端提交了一个消息...
发送消息成功!
488号客户端提交了一个添加好友申请...
好友账号不存在!
488号客户端提交了一个添加好友申请...
添加好友成功!
现在4和666是好友了!
488号客户端提交了一个检查聊天室成员信息的请求..
请求接受!
488号客户端提交了一个加入聊天室申请...
请求接受!
1号聊天室现有1位成员:
['4']
488号客户端提交了一个检查聊天室成员信息的请求..
请求接受!
508号客户端提交了一个登出申请...
账号已登出!
508号客户端提交了一个登录申请...
登录账号: 666
登录密码: 1234
登录成功!
666已登录!
508号客户端提交了一个检查聊天室成员信息的请求..
请求接受!
508号客户端提交了一个加入聊天室申请...
请求接受!
1号聊天室现有2位成员:
['4', '666']
508号客户端提交了一个检查聊天室成员信息的请求..
请求接受!
488号客户端提交了一个检查聊天室成员信息的请求..
请求接受!
488号客户端提交了一个消息...
发送消息成功!
508号客户端提交了一个消息...
发送消息成功!
[1671965661.793883, 1671965669.455387, 1671978669.7
34842, 1671978676.771842], ['111', '111', '4', '666'
已加载GUI缓冲!
正在接收消息...
收到一条消息!
(4)Dave:
Hello!

接收完成!
已加载GUI缓冲!
正在接收消息...
收到一条消息!
(666)QB:
Yee~

查询聊天记录中...
查询成功!
正在退出客户端...
PS C:\Users\Liu Binrui\OneDrive\Cloud\VS Code\Codes
\Python\Wechat> []
```

# 软件结构设计

## 服务器

## 文件架构

```
└─ server
    │ database.py
    │ main.py
    │ server.py
    └─ data
        │ History.db
        └─ Info.db
```

## 功能简介

```
server\server.py
```

- 为**服务器主文件**，负责连接所有的客户端，并利用不同的接口为客户端的各种请求提供服务。

在 `server.py` 中共有12个接口，为客户端的不同请求提供相应的服务，其功能分别如下所示：



```

#1
login(connection:socket.socket) -> tuple:
'''
接收客户端上传的账号和密码，在数据库中检验其正确性
传回验证码，1表示密码正确，0表示密码错误，-1表示不存在该账号，-8080表示账号已经登陆
若密码正确，传回其昵称和好友账号、昵称的元组
返回登录的账户信息
'''

#2
logout(connection:socket.socket):
'''接收客户端提交的登出申请，执行登出账户操作'''

#3
register(connection:socket.socket):
'''
接收客户端上传的账号、密码和昵称，尝试注册一个新的账户
传回验证码，1表示注册成功，0表示账号已存在
'''

#4
add_friend(connection:socket.socket):
'''
接收客户端上传的账号，尝试添加一个新的好友关系，传回好友昵称
传回验证码，1表示添加成功，0表示好友关系已存在，-1表示好友账号不存在
'''

#5
exchange_message(connection:socket.socket, connection_b:socket.socket):
'''
[供私聊使用的接口]
接收客户端上传的账号和消息，向目标账户和源账户的后台端口发送发送者的账号、昵称和消息
传回验证码，1表示发送成功，0表示好友不在线
'''

#6
broadcast_message(connection:socket.socket):
'''
[供聊天室使用的接口]
接收客户端上传的聊天室编号和消息
向当前聊天室中的所有后台端口发送发送者的账号、昵称、消息和聊天室编号
'''

#7
getin_chatroom(connection:socket.socket):
'''接收客户端上传的聊天室编号，处理其进入聊天室的请求'''

#8
getout_chatroom(connection:socket.socket):
'''接收客户端上传的聊天室编号，处理其退出聊天室的请求'''

#9
check_roominfo(connection:socket.socket):
'''接收客户端上传的聊天室编号，返回聊天室当前成员信息'''

#10
ask_privatehistory(connection:socket.socket):
'''
接收客户端上传的账号，传回验证码，1表示查询成功，0表示不存在聊天记录
返回私聊的聊天记录
'''

```

```
'''  
#11  
ask_chatroomhistory(connection:socket.socket):  
'''接收客户端上传的聊天室编号, 返回聊天室的聊天记录'''  
#12  
sendfile(connection:socket.socket, file:str) -> int:  
'''  
读取文件并发送到接收端  
返回验证码, 1代表发送成功, 返回0代表发送失败  
(用于传输图片的接口, 尚未完全实现)  
'''
```

另外, `server.py` 提供了服务器启动接口 `start()`, `main.py` 可以通过调用该函数完成服务器的初始化和启动。

服务器启动后, 会基于默认 `socket` (默认地址为 `(localhost,49152)`) 等待用户连接。每当有一个新的客户端成功连接, 服务器会开启一个子线程执行指令接收函数 `toserver(*args)`, 等待客户端的请求并为其提供服务。各子线程之间通过一个互斥锁来保证互不干扰。

#### `server/database.py`

- 为数据库操作文件, 提供了9个数据库的操作接口, 其功能如下所示:

```

#1
create_info()
'''创建主信息数据库，含账户信息表、好友信息表和聊天室信息表'''
#2
create_History()
'''创建聊天记录数据库，每一位用户对应着一个聊天记录表'''
#3
search_account(account:str) -> tuple
'''根据账号，在数据库中查找账户信息，以元组格式返回'''
#4
search_friends(account:str) -> tuple
'''根据账号，在数据库中查找好友的帐户信息，以元组格式返回账号列表和昵称列表'''
#5
search_privatehistory(account:str, account_friend:str) -> tuple:
'''
根据两个账号，在数据库中查找两账户间的私聊聊天记录
以元组格式返回时间戳列表、发送者账号列表和消息内容列表
'''
#6
search_chatroomhistory(roomid:str) -> tuple:
'''
根据聊天室编号，在数据库中查找聊天室的聊天记录
以元组格式返回时间戳列表、发送者账号列表和消息内容列表
'''
#7
add_account(account:str, password:str, name:str) -> int:
'''在数据库中添加账户信息，返回1表示成功，返回0表示账号重复'''
#8
add_friend(account:str, account_friend:str) -> int:
'''在数据库中添加一条好友关系，返回1表示成功，返回0表示好友关系重复或不存在好友账号'''
#9
add_history(account:str, content:str, position:str, isroom:bool) -> int:
'''在数据库中添加一条聊天记录，返回1表示成功，返回0表示失败'''

```

主文件 `server.py` 通过调用这些接口，实现了对数据库的访问、查找和添加操作。

在首次调用 `database.py` 时会检查数据库是否存在，若不存在会自动创建数据库和其中的基本表。

#### server\data

- Info.db **信息数据库**，储存了所有用户的账户信息、好友关系和聊天室信息。 内含基本表AccountInfo, FriendInfo, ChatroomInfo。
- History.db **聊天记录数据库**，储存了所有用户的聊天信息。 内含基本表ChatHistory。

#### server/main.py

- 为**服务器启动文件**。

在终端启动本文件即可开启服务器，同时完成服务器的初始化。

# 客户端

## 文件架构

```
└─ client
    │ client.py
    │ main.py
    │ GUI_framework.py
    └─ pics
        └─ icon.ico
```

## 功能简介

### client\client.py

- 为**客户端主文件**，负责监测用户在图形界面的操作，并根据用户的指令向服务器请求相应的服务。

在 `client.py` 中共有**12**个接口，和 `server.py` 中的12个接口按顺序一一对应。用户在图形界面中的操作将被转义为各种请求，然后通过对应的接口向服务器请求服务，服务器则会调用与之相对应的接口处理该请求。

本软件通过**指令码**完成上述过程。例如，用户在图形界面中点击了“登录”按钮，客户端就会调用 `login(*args)` 函数，在 `login(*args)` 中，客户端首先会向服务器发送“登录”对应的**指令码**——“**login**”，然后提交请求信息。服务器中负责为本客户端提供服务的子线程 `to_server(*args)` 在接收到该**指令码**后，便会调用 `server.py` 中的 `login(*args)` 接口处理客户端发送的请求。

这12个接口的功能和相应的指令码如下所示：

```

#1
login(sock_client:socket.socket, account:str, password:str) -> int:
'''
向服务器上传账号和密码，尝试登录账户，指令码：__login__
接收验证码，1表示密码正确，0表示密码错误，-1表示不存在该账号，-8080表示该账号已登录
若密码正确，接收用户昵称和好友账号的元组，并修改客户端的全局变量
返回验证码
'''

#2
logout(sock_client:socket.socket):
'''向服务器提交登出申请，指令码：__logout__'''

#3
register(sock_client:socket.socket, account:str, password:str, name:str) -> int:
'''
向服务器上传账号、密码和昵称，尝试注册一个新的账户，指令码：__register__
接收验证码，1表示注册成功，0表示账号已存在
返回验证码
'''

#4
add_friend(sock_client:socket.socket, account_friend:str) -> int:
'''
向服务器上传账号，尝试添加一个好友，接收好友昵称并修改本地的全局变量，指令码：__add__
接收验证码，1表示添加成功，0表示好友关系已存在，-1表示好友账号不存在，返回验证码
'''

#5
send_message(sock_client:socket.socket, account_friend:str, message:str) -> int:
'''
[供私聊使用的接口]
向服务器上传好友账号和消息，请求向好友发送该消息，指令码：__chat__
接收验证码，1表示发送成功，0表示好友不在线，返回验证码
'''

#6
call_message(sock_client:socket.socket, roomid:str, message:str):
'''
[供聊天室使用的接口]
向服务器上传聊天室编号和消息，请求在聊天室发送该消息，指令码：__broadcast__
'''

#7
getin_chatroom(sock_client:socket.socket, roomid:str):
'''向服务器上传聊天室编号，请求加入聊天室，指令码：__getin__'''

#8
getout_chatroom(sock_client:socket.socket, roomid:str):
'''向服务器上传聊天室编号，请求退出聊天室，指令码：__getout__'''

#9
check_roominfo(sock_client:socket.socket, roomid:str) -> tuple:
'''
向服务器上传聊天室编号，接收聊天室成员信息，指令码：__cr__
以元组格式返回聊天室成员账号和昵称的列表
'''

#10
ask_privatehistory(sock_client:socket.socket, account_friend:str) -> tuple:

```

```

'''
向服务器上传好友的账号，接收聊天记录，指令码：__private__
以元组格式返回时间戳列表、发送者账号列表和消息内容列表
'''

#11
ask_chatroomhistory(sock_client:socket.socket, roomid:str) -> tuple:
'''
向服务器上传聊天室编号，接收聊天记录，指令码：__room__
以元组格式返回时间戳列表、发送者账号列表和消息内容列表
'''

#12
recvfile(file:str) -> int:
'''
接收来自发送端的文件
返回验证码，1代表接收成功，返回0代表接收失败
（用于传输图片的接口，尚未完全实现）
'''

```

在确认了和服务器的连接后，后台消息接口 `recv_message(*arg)` 将在一个子线程中启动，随时接收服务器发送的消息，并载入缓冲区，最终在图形界面中加载。

### client\GUI\_framework.py

- 为**图形界面文件**，是软件中直接和用户交互的部分，同时负责将用户的操作提交至 `client.py` 中的相应接口。

登录、注册、聊天室等窗口均以函数的形式封装，以便调用。共计8个图形界面函数，如下所示：

```

login()'''启动登录界面'''
register()'''启动注册界面'''
menu()'''打开菜单界面'''
add_friend()'''启动添加好友界面'''
privatechat(account_friend:str)'''启动私聊界面'''
chatroom(roomid:str)'''启动聊天室界面'''
showhistory(history:tuple)'''展示聊天记录'''
neterror():'''弹窗，提示和服务端断开了连接'''

```

图形界面采用 `tkinter.ttk` 制作，使用 `place` 方法布局。

### client\pics

- 内含图形界面所需的**图形元素**。

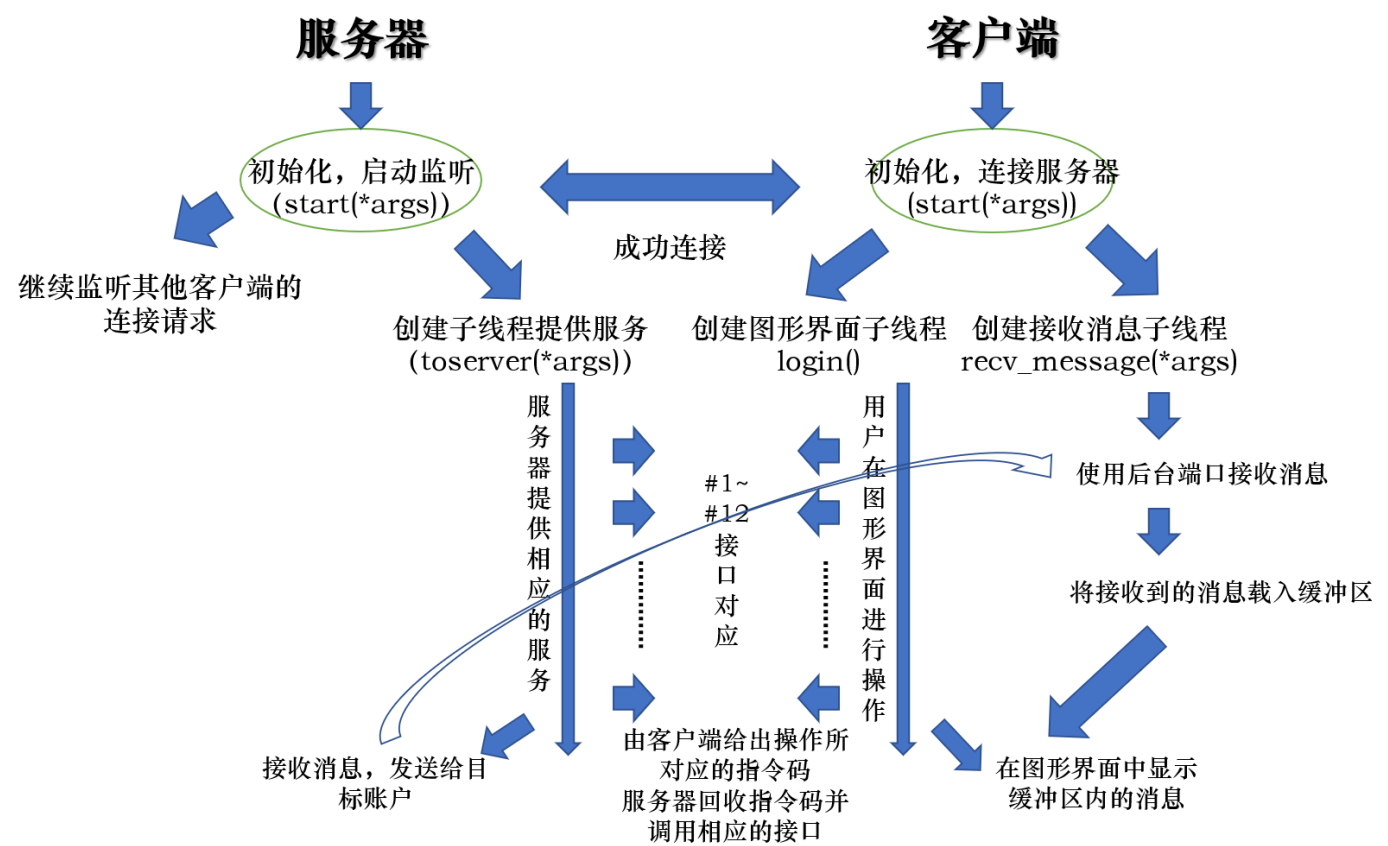
### client\main.py

- 为**客户端启动文件**。

在终端启动本文件即可开启客户端。

首先将调用 `client.py` 中的启动接口 `start()`，启动客户端，并连接服务器。成功连接服务器后，将开启两个子线程，分别运行 `client.py` 中的后台消息接口 `recv_message(*arg)` 和图形用户界面。

## 总体运行结构图



## 其他

- 由于采用了部署在服务器的数据库系统存储数据，因此服务器和客户端的重启不会导致数据的丢失。
- 关于服务器和客户端的任何操作，服务器和客户端的后台均有完善的信息提示，这为软件的调试工作和监测工作提供了便利。
- 作者已经在能力范围内对本软件作了尽量充分的测试，绝大部分情况可以保证软件不崩溃、不卡死，或者有正确的错误处理机制。