基于 PYTHON 实现一个简单的 QQ 工具

一步步手把手教你实现

1、一个最简单的聊天程序：单向控制台聊天

```
from socket import *
ADDR = ('localhost', 21567)
udpCliSock = socket(AF_INET, SOCK_DGRAM)

data = raw_input('> ')
udpCliSock.sendto(data, ADDR)
udpCliSock.close()
```

客户端 1

```
from socket import *
ADDR = ('' , 21567)
udpSerSock = socket(AF_INET,SOCK_DGRAM)
udpSerSock.bind(ADDR)
data, addr = udpSerSock.recvfrom(1024)
print data
udpSerSock.close()
```

服务器 1

2、稍复杂的聊天程序：双向控制台聊天

修改第一个程序，实现双向聊天功能，及客户端和服务器都能收发信息。但此时是有序收发。

实现下列功能：

客户端：首先发一个消息，然后接收一个对方收到的消息，再发一个消息，再收一个消息，除非客户端发一个 END，结束。

服务器：接收消息

步骤：这个合适的做法，当然是要做循环。但在做循环之前，我们可以把收发程序复制三份(或多份)。然后，我们可以将我们的服务器改成循环。但客户端可以暂时不改。

3、具有简单图形用户界面（GUI）的聊天程序

修改客户端的程序，基于 TK 做一个简单界面。

注意 1：只对客户端实现界面，服务器不用界面。

注意 2：相对于实验 2 的客户端收发程序的复制，这里客户端需改为循环。

4、基于面向对象的图形用户界面（GUI）聊天程序

基本上是没有什么好修改的，就是加一个类，这个类里面有二个函数，一个是发送的程序，另一个是初始化的程序。发送程序是原来的程序，初始化程序实现原来界面的布局，以及 ip、端口号的设置。

这里需要注意的是：在相关的一些地方加上 SELF。SELF 的意思是类自己了，就是调用类自己的数据。

5、GUI+多线程的全双向聊天程序

其实这个也是很简单的，就是将原来的接收数据模块分离出来，做成一个专门的接收方法。然后，再写一个启动多线程的代码，就是一句话，启一个新的线程，而这个线程指明是调用上面的接收程序。最后在主程序中，加一句话，来调用上面写的类中的那个启动调用多线程的方法。

在 python 中，可以通过 thread 模块中的 start_new_thread(function,args[])函数来启动一个新线程，其中 function 参数是你将要调用的线程函数，args[]是传递给你的线程函数的参数。eg:

```
import thread
def Func1():
```

```
        print('hello')
    def Func2():
        thread.start_new_thread(Func1, ())    //启用单个线程
    if __name__=='__main__':
        Func2()
```

这里需要注意的是：其实也没有什么太多的东西了，就是用 UDP，可以起一个服务端来，接收服务器发送的消息，也就是要绑定一个新的端口。

6、QQ 登陆服务器

上面这个是 QQ 登陆，我们仿这个做一个。

界面已经做好了，后面是需要补充逻辑：

1）错误，清空输入并提示错误，用户重新输入

2）正确，登陆成功了，就新启一个聊天应用。然后关闭当前登录应用。

理念 1：专业程序员写程序，都不用 MOUSE，只用键盘。正如专业打字员，打字不看键盘，不看屏幕，只看文稿。

理念 2：从小到大一步步构建。