

TCP-Peer to Peer

Input :

Server.py

```
import socket
from threading import Thread, Lock
import sys

def receiver(sock):
    global flag
    while flag:
        data=sock.recv(size) #receive client data
        '''Receive data from client''' #Receive client data
        if data == 'quit':
            sys.exit(0)
        print "\t\t"+data
    flag = True
    print "Thread Exiting"

host = '172.19.229.227'          #Server IP address
port = 50004                    #Port address
backlog = 5                     #For binding connections
size = 1024                     #Size of packet data to receive
flag = True

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Creating a Socket
s.bind((host,port))           #Binding Port and IP address
s.listen(backlog)              #Listen to connection

while 1:
    print "Waiting for Peer to Join..."
    client,address=s.accept()
    '''Accept a connection'''    #Accept a client connecion
    print "Client "+str(address)+"joined chat.\n"

    # Start channel receiver thread
    '''Create a thread'''
    recv_thread=Thread(target=receiver,args=(client,))
    recv_thread.daemon = True
    recv_thread.start()

    data = "random" #Entering any value so it is not quit initially
    while data != 'quit': #User will enter 'quit' to exit server
        data = raw_input()
        client.send(data) #Enter message to send to client
        '''Send data to client'''    #Sending data to client
    flag = False
    client.close()    #Closing the socket
```

Client.py

```
import socket
from threading import Thread, Lock
import sys

def receiver(sock):
    while flag:
        data = sock.recv(1024) #Receive data from server
        if data == 'quit':
            sys.exit(0)
        print "\t\t"+data

host = '172.19.229.227'          #Server IP address
port = 50004                    #Port address
size = 1024                     #Size of packet data to receive
flag = True

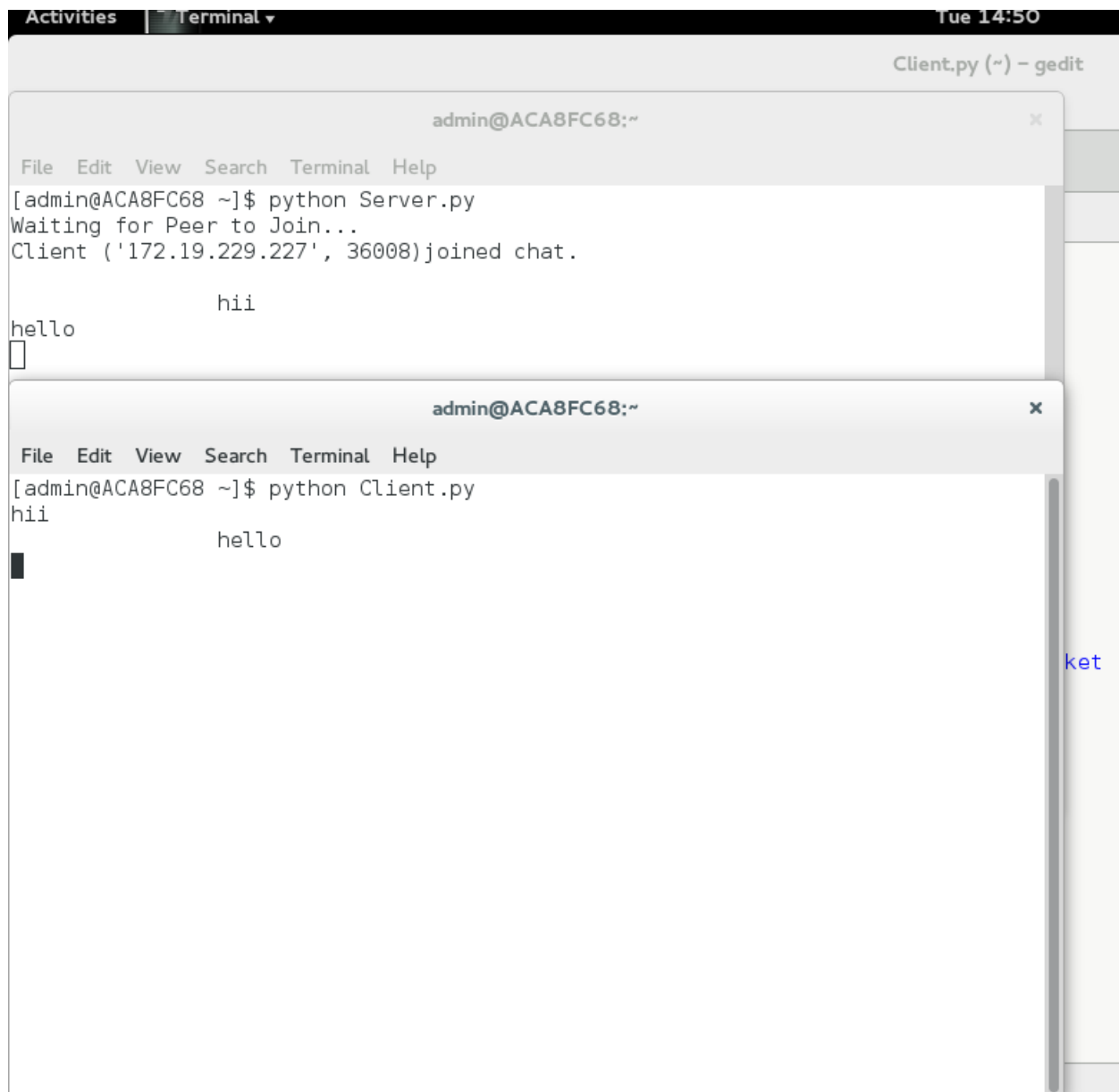
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Creating a
Socket
sock.connect((host,port))      #Connect to Server

# Start Receiver Thread
hop = Thread(target=receiver, args=(sock,))
hop.daemon = True
hop.start()

data = "random" #Entering any value so it is not quit initially

while data != 'quit': #User will enter 'quit' to exit server
    data = raw_input()#Enter message to send to server
    sock.send(data)
    '''Send data to server''' #Sending data to server
sock.close() #Closing the socket.
```

Output :



The screenshot shows a Linux desktop environment with a terminal window open. The terminal title bar indicates the user is 'admin@ACA8FC68' and the current directory is '~'. The terminal output shows the execution of a Python script 'Server.py' which starts a chat server. It then receives a connection from a client at IP '172.19.229.227' on port 36008. The server prompts for a name and receives 'hello'. The client then sends a message 'hii'. A second terminal window is also shown, titled 'admin@ACA8FC68:~', where the script 'Client.py' is executed. The client prompts for a name and receives 'hii', then sends a message 'hello'.

```
admin@ACA8FC68:~  
File Edit View Search Terminal Help  
[admin@ACA8FC68 ~]$ python Server.py  
Waiting for Peer to Join...  
Client ('172.19.229.227', 36008)joined chat.  
  
hello  
hii  
[  
  
admin@ACA8FC68:~  
File Edit View Search Terminal Help  
[admin@ACA8FC68 ~]$ python Client.py  
hii  
hello  
[
```

ket