Python socket – chat server and client with code example

By Silver Moon | March 31, 2013 49 Comments

Socket based chat application

In our previous article on socket programming in python we learned about the basics of creating a socket server and client in python. In this post we are going to write a very simple chat application in python that is powered by sockets.

The chat application we are going to make will be more like a chat room, rather than a peer to peer chat. So this means that multiple users can connect to the chat server and send their messages. Every message is broadcasted to every connected chat user. The construction is as simple as the theory.

The code consists of 2 python scripts. First is the server and the other is the chat client.

Chat server

The chat server does the following things

1. Accept multiple incoming connections for client.
2. Read incoming messages from each client and broadcast them to all other connected clients.

Here is the code of the chat server. It server opens up port 5000 to listen for incoming connections. The chat client must connect to this same port. You can change the port number if you want.

The server handles multiple chat clients with select based multiplexing. The select function monitors all the client sockets and the master socket for readable activity. If any of the client socket is readable then it means that one of the chat client has send a message.

```
Code

# Get the list sockets which are ready to be read through
read_sockets,write_sockets,error_sockets = select.select(
```

When the select function returns, the read_sockets will be an array consisting of all socket descriptors that are readable. So if the master socket is readable, the server would accept the new connection. If any of the client socket is readable, the server would read the message, and broadcast it back to all clients except the one who send the message. The following function broadcasts the message to all chat clients.

```
Code

def broadcast_data (sock, message):
    #Do not send the message to master socket and the cli
    for socket in CONNECTION_LIST:
        if socket != server_socket and socket != sock :
            try :
                socket.send(message)
            except :
                # broken socket connection may be, chat c
                socket.close()
                CONNECTION_LIST.remove(socket)
```

If the broadcast function fails to send message to any of the client, the client is assumed to be disconnected and the connection is closed and the socket is removed from the connection list.

Rest of the program is quite self explanatory. Here is the full code of the chat client.

Code

```
# Tcp Chat server

import socket, select

#Function to broadcast chat messages to all connected cli
def broadcast_data (sock, message):
    #Do not send the message to master socket and the cli
    for socket in CONNECTION_LIST:
        if socket != server_socket and socket != sock :
            try :
                socket.send(message)
            except :
                # broken socket connection may be, chat c
                socket.close()
                CONNECTION_LIST.remove(socket)

if __name__ == "__main__":

    # List to keep track of socket descriptors
    CONNECTION_LIST = []
    RECV_BUFFER = 4096 # Advisable to keep it as an expor
    PORT = 5000

    server_socket = socket.socket(socket.AF_INET, socket.
    # this has no effect, why ?
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO
    server_socket.bind(("0.0.0.0", PORT))
    server_socket.listen(10)

    # Add server socket to the list of readable connectio
    CONNECTION_LIST.append(server_socket)

    print "Chat server started on port " + str(PORT)

    while 1:
        # Get the list sockets which are ready to be read
        read_sockets,write_sockets,error_sockets = select
```

```
        for sock in read_sockets:
            #New connection
            if sock == server_socket:
                # Handle the case in which there is a new
                sockfd, addr = server_socket.accept()
                CONNECTION_LIST.append(sockfd)
                print "Client (%s, %s) connected" % addr

                broadcast_data(sockfd, "[%s:%s] entered r

            #Some incoming message from a client
            else:
                # Data recieved from client, process it
                try:
                    #In Windows, sometimes when a TCP pro
                    # a "Connection reset by peer" except
                    data = sock.recv(RECV_BUFFER)
                    if data:
                        broadcast_data(sock, "\r" + '<' +

                except:
                    broadcast_data(sock, "Client (%s, %s)
                    print "Client (%s, %s) is offline" %
                    sock.close()
                    CONNECTION_LIST.remove(sock)
                    continue

    server_socket.close()
```

Run the server in a console.

```
$ python chat_server.py
Chat server started on port 5000
```

Chat Client

Now lets code the chat client that will connect to the above chat server. The client is based on the telnet program in python. It connects to a remote server, sends messages and receives messages.

The chat client does the following 2 things :

1. Listen for incoming messages from the server.
2. Check user input. If the user types in a message then send it to the server.

Now here is something tricky. The client has to actually listen for server message and user input at the same time. To do this, we use the select function. The select function can monitor multiple sockets or file descriptors for some "interesting activity" which is this case is readable. When a message comes from the server on the connected socket, it is readable and when the user types a message and hits enter, the stdin stream is readable.

So the select function has to monitor 2 streams. First is the socket that is connected to the remote webserver, and second is stdin or terminal input stream. The select function blocks till something happens. So after calling select, it will return only when either the server socket receives a message or the user enters a message. If nothing happens it keeps on waiting.

```
Code

socket_list = [sys.stdin, s]

# Get the list sockets which are readable
read_sockets, write_sockets, error_sockets = select.selec
```

We simply create an array of the stdin file descriptor that is available from the sys module, and the server socket s. Then we call the select function passing it the list. The select function returns a list of arrays that are readable, writable or had an error. The readable sockets will be again a list of sockets that is readable.

So in this case, the read_sockets array will contain either the server socket, or stdin or both. Then the next task is to do relevant processing based on which socket is readable. If the server socket is readable, it means that the server has send a message on that socket and so it should be printed. If stdin is readable, it means that the user typed a message and hit enter key, so that message should be read and send to server as a chat message.

Here is the python code that implements the above logic using select function

```
Code

# telnet program example
import socket, select, string, sys

def prompt() :
    sys.stdout.write('<You> ')
    sys.stdout.flush()
```

```
#main function
if __name__ == "__main__":

    if(len(sys.argv) < 3) :
        print 'Usage : python telnet.py hostname port'
        sys.exit()

    host = sys.argv[1]
    port = int(sys.argv[2])

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(2)

    # connect to remote host
    try :
        s.connect((host, port))
    except :
        print 'Unable to connect'
        sys.exit()

    print 'Connected to remote host. Start sending messag
    prompt()

    while 1:
        socket_list = [sys.stdin, s]

        # Get the list sockets which are readable
        read_sockets, write_sockets, error_sockets = sele

        for sock in read_sockets:
            #incoming message from remote server
            if sock == s:
                data = sock.recv(4096)
                if not data :
                    print '\nDisconnected from chat serve
                    sys.exit()
                else :
                    #print data
                    sys.stdout.write(data)
                    prompt()

            #user entered a message
            else :
                msg = sys.stdin.readline()
                s.send(msg)
                prompt()
```

Run the client from multiple consoles.

```
$ python telnet.py localhost 5000
Connected to remote host. Start sending messages
<You> hello
<You> I am fine
<('127.0.0.1', 38378)> ok good
<You>
```

on another console

```
<You> [127.0.0.1:39339] entered room
<('127.0.0.1', 39339)> hello
<('127.0.0.1', 39339)> I am fine
<You> ok good
```

So the messages send by one client are seen on the consoles of other clients. Logic is quite simple. Run it and check it out.

Note

The above shown chat client is not going to work on windows. It uses the select function to read data from both the socket and the input stream. This works on linux but not on windows.

The python documentation on select mentions this

```
File objects on Windows are not acceptable, but sockets a
```

Linux treats sockets and file descriptors in the same manner, therefor the select function is able to read from stdin. On windows the select function will not read anything except sockets created by the winsock socket functions.

There is another drawback that the above shown chat program suffers. If in the chat client a user is typing a message and while typing a message comes from the server, then the server message shall be printed rightaway and the message that the user was typing would be lost. That is the expected behaviour of this program and there is nothing that can be done to fix this properly.

Only solution is to use better terminal libraries like ncurses to keep the user input separate from terminal output. Or write a gui program.

Last Updated On : 23rd April 2019

Related Post

Code a simple telnet client using sockets in pytho...

Syn flood program in python using raw sockets (Lin...

Programming udp sockets in python

CATEGORY: PYTHON

TAGS: PYTHON SOCKETS

About Silver Moon

Php developer, blogger and Linux enthusiast. He can be reached at binarytides@gmail.com. Or find him on Google+

View all posts by Silver Moon →

49 thoughts on “ Python socket – chat server and client with code example ”

kodebaca

June 15, 2019 at 8:08 am

Thanks you for this tutorial

harish

December 14, 2017 at 11:11 pm

client goes offline once it runs

ktulx

December 14, 2017 at 8:08 pm

If anyone wonders why the server does not handle a client disconnect properly:

just move or copy the lines 60-64 from “except” block up to the “if – else” block.

```
if data:
broadcast_data(sock, “\r” + ‘ ‘ + data)

else:
broadcast_data(sock, “Client (%s, %s) is offline” % addr)
print “Client (%s, %s) is offline” % addr
sock.close()
CONNECTION_LIST.remove(sock)
continue
```

Very nice tutorial though, Silver Moon. Thanks a lot.

Sumit Aher

December 7, 2017 at 5:05 pm

I am trying to implement Tcp server socket code which accepts n number of connections and process the data received from n number of clients. I am creating the multiple threads for processing the data. Currently multiple connections are creating but after 10-15 min the data is getting interchanged between the threads while processing. How do I fix this ? Please help me to resolve this issue.

Sumit Aher

December 7, 2017 at 5:05 pm

I am trying to implement Tcp server socket code which accepts n number of connections and process the data received from n number of clients. I am creating the multiple threads for processing the data. Currently multiple connections are creating but after 10-15 min the data is getting interchanged between the threads while processing and resulting wrong output.How do I fix this ? Please help me to resolve this issue.

biruk

October 14, 2017 at 7:07 pm

hi
if I want to get communication between two separate systems, not on a single , one as a server and the other as a cliant.
thanks!

paul

July 19, 2017 at 7:07 pm

wow your a great programer

paul

July 19, 2017 at 7:07 pm

i like how it works you run the server then you start the chat server
it's really nice.

Deniz

May 19, 2017 at 6:06 pm

if(len(sys.argv) < 3) :
print ('Usage : python chat_client.py hostname port')
sys.exit()

Whenever I start the client program after I start the server program, the only output I get is the print of the if statement above. How can I make the program work ?

ben maccini

April 19, 2017 at 7:07 am

how to code messaging app

ben maccini

April 19, 2017 at 7:07 am

good comment

bhargav

December 26, 2016 at 2:02 pm

the client side enters the if loop and exits
if(len(sys.argv) < 3):

print("usage : python telenet.py hostname port")
sys.exit()
i tried varying the number it gave me an array index out of rage error

---

al

November 20, 2016 at 11:11 pm

Hi, I tried to follow your code almost entirely (I had only to use encode-decode utf-8 for string). It works well between server and each client but I can not see the message sent back from the server on the other client (the one is only viewing the chat). Do you know what could be the reason?

thank

Kritika Garg

March 13, 2018 at 11:11 pm

same thing is happening to me.

Vitor

November 15, 2016 at 6:06 pm

I need some help, first of all, when i close one of the terminals conected with server, it crashes shortly after.
Second, how can i do other devices access my server? it only works if i try to connect on the same device
Thanks

deadbool

October 16, 2016 at 7:07 pm

server,py works flawless, client.py gives me an error:

Connected to remote host. Start sending messages
Traceback (most recent call last):
File “client.py”, line 35, in
read_sockets, write_sockets, error_sockets =
select.select(socket_list , [], [])
OSError: [WinError 10038]

what’s the matter??
thx

Yuval Keren

October 23, 2016 at 5:05 pm

“[WinError 10038] An operation was attempted on something that is not a socket” will occur when applying a non-socket object in a command which only accepts socket objects. As mentioned in the article, when using the Select library on windows it only accepts socket objects since it uses the Winsock library and will not accept any other objects (in your case, the stdin file).
If you’re interested in applying that kind of code on a windows machine, consider reading on the ‘msvcrt’ library which allows you to read user input in a non-blocking way while still using the select command for the client socket, or consider building a multi threaded client side.
Good Luck :)

Metonimie

April 22, 2016 at 3:03 am

Thanks to this article I managed to learn something about sockets and update + enhance this code to py3
https://github.com/Metonimie/python-networking

siva

May 8, 2016 at 2:02 pm

Do you know how to deploy this chat application in web using heroku or some other? If so, could you please explain me how to do it?

pratibha uphade

March 4, 2016 at 9:09 pm

hi,

Could you explain what this line does in the server:

server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

Could you explain what this line does in the server:

broadcast_data(sockfd, "[%s:%s] entered room\n" % addr)

pratibha uphade

March 4, 2016 at 9:09 pm

hi,very nice code….
Could you explain what this line does in the server app:

server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

and also thins line does in the client :

broadcast_data(sockfd, "[%s:%s] entered room\n" % addr)

Giridhar G Nair

April 7, 2016 at 3:03 pm

Hey
The setsockopt line is required to make the socket reuseable. Simply put, if you close the program and then open it immediately and try to access the same socket, the "in use" error can be avoided.

GilV

December 31, 2015 at 11:11 pm

Could you explain-me why this line doen't work?

read_sockets, write_sockets, error_sockets =
select.select(socket_list , [], [])

it gives me this error:

Traceback (most recent call last):

File "D:UE2º Anol SemestreRedes de
ComputadoresTrabalhoTP2client_test1.py", line 57, in

read_sockets, write_sockets, error_sockets =
select.select(socket_list , [], [])

io.UnsupportedOperation: fileno

I will sent you a picture of my code (for a work):

Indra Cahya

December 28, 2015 at 7:07 pm

sir, i get message
"Usage : python telnet.py hostname port"

can you give me solution ?

---

tiferrei

January 30, 2016 at 1:01 am

When you start the client you have to input the ip and port with it, so imagine my script is called example.py and i'm using a localhost with port 5000, that would be python example.py localhost 5000, got it? And of course, your server needs to be running first.

bhargav

December 26, 2016 at 2:02 pm

ta for the hint

bhargav

December 26, 2016 at 3:03 pm

the client disconnects as soon as it connects and goes offline

---

Jason

December 8, 2014 at 5:05 pm

how can I make client input to nio. If we do not input something, the program will wait in there. Anyone has an idea. I am working on it with multithread.

jai

November 14, 2014 at 8:08 am

its not working

Dash Kiwing

July 8, 2014 at 9:09 am

Excuse me
I have one question
when i type
netstat -tanp | grep python
i have got
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:5000 0.0.0.0:* LISTEN 22725/python

tcp 0 0 127.0.0.1:56271 127.0.0.1:5000 ESTABLISHED 22727/python
tcp 0 0 127.0.0.1:56272 127.0.0.1:5000 ESTABLISHED 22728/python
tcp 0 0 127.0.0.1:5000 127.0.0.1:56272 ESTABLISHED 22725/python
tcp 0 0 127.0.0.1:5000 127.0.0.1:56271 ESTABLISHED 22725/python
we can see on right there. how can i change that program name in above program

thanks in advance…

José Rosendo

March 2, 2014 at 12:12 pm

great example, it is works! :) thanks!

moonman

February 10, 2014 at 5:05 pm

ansi commands to protect tty text is awesome if you like retro. here's how.
self.rows, self.columns = os.popen('stty size', 'r').read().split()
rows = int(self.columns) – 5
print '33[5;'+str(rows)+'r'

linux only and remember to import.

Schorsch

January 23, 2014 at 10:10 pm

Even worse: Start the server, connect two clients. Terminate one client, then type a few lines in the second… server crashes with

CONNECTION_LIST.remove(sock)
ValueError: list.remove(x): x not in list

Schorsch

January 23, 2014 at 10:10 pm

Disconnecting does not work for me. It never removes anything from the connection list or sends offline notifications…

Fred99

December 1, 2013 at 4:04 pm

Hi, I have a problem with "read_sockets,write_sockets,error_sockets = select.select(CONNECTION_LIST,[],[])" in the server code. When I connect from other cients I get socket.error (9 'bad file descriptor') after a while and the server stops working :(. If you know what it can be please let me know.

sabaka

August 29, 2013 at 1:01 am

Hi! How to setup HOST and PORT when i want to use your app on two computers connected to different routers?

Nhân

May 30, 2013 at 11:11 pm

"The above shown chat client is not going to work on windows."

Why's not work??

i use Eclipse (PyDev) to write your code in Windows. And i get this : "Usage : python telnet.py hostname port"
Why??

tattata

December 1, 2013 at 12:12 am

replace the lines 11-16 with Host = host as string and port as int

---

Indra Cahya

December 28, 2015 at 7:07 pm

i already replace like u said, but still not working sir,
can u show your code please

beeznicity

May 15, 2013 at 5:05 am

Hi, after trying your code on Windows I have spent the last couple days trying to figure out how to implement what you're doing in line 35: read_sockets, write_sockets, error_sockets = select.select(socket_list , [], []) since 'select' can't handle sys.stdin on windows. Do you know how to do something similar that will work in Windows?

Thanks

Silver Moon

May 15, 2013 at 9:09 am

Thanks for pointing it out.
I updated the post with details about the select function limitation on windows.

Sammy

April 11, 2013 at 10:10 pm

I cannot seem to get it to work on my machine as it does not like sys.exit(). Could you please send me the files, perhaps with a patch to my email: sammy.herring@me.com. Many Thanks, Sam

Silver Moon

May 15, 2013 at 8:08 am

does it show any error ?

Jack

May 27, 2013 at 8:08 pm

I have the same problem when i start the server it works fine but when i run the client it gives this error
Traceback (most recent call last):
where the file is located on my computer
Line 11, in
sys.exit()
SystemExit

Silver Moon

May 27, 2013 at 9:09 pm

what version of python are you using ?

alan

April 30, 2014 at 7:07 pm

hi yes I did that but it still not working

Echito
November 13, 2014 at 10:10 am

The problem is that it's a Windows OS. It happens to me as well 1st of all, it's not an error. It's explaining to you that it's using (System Exit) To close the program from your active programs.

James
April 2, 2013 at 2:02 am

Hi. Excellent posts on sockets. Really enjoying them.

Could you explain what this line does in the server app:

server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

Thanks

Anthony Straetman
April 3, 2013 at 1:01 pm

When a socket is used it gets a TIME_WAIT state which prevents it from being used again.
That line makes sure that the socket can be used when it's in the TIME_WAIT state.