Code a simple socket server in Python

By Silver Moon | July 4, 2013 25 Comments

Python sockets

In a previous tutorial we learnt how to do basic socket programming in python. The tutorial explained how to code a socket server and client in python using low level socket api. Check out that tutorial if you are not through on the basics of socket programming in python.

To recap, sockets are virtual endpoints of a communication

channel that takes place between 2 programs or processes on the same or different machines. This is more simply called network communication and sockets are the fundamental things behind network applications. For example when you open google.com in your browser, your browser creates a socket and connects to google.com server. There is a socket on google.com server also that accepts the connection and sends your browser the webpage that you see.

Socket Servers in python

In this post we shall learn how to write a simple socket server in python. This has already been covered in the previous tutorial. In this post we shall learn few more things about programming server sockets like handling multiple connections with the select method.

So lets take a look at a simple python server first. The things to do are, create a socket, bind it to a port and then accept connections on the socket.

```
1. Create socket with socket.socket function
2. Bind socket to address+port with socket.bind function
3. Put the socket in listening mode with socket.listen function
3. Accept connection with socket.accept function
```

Now lets code it up.

```
1  '''
2      Simple socket server using threads
3  '''
4
5  import socket
6  import sys
7
8  HOST = ''   # Symbolic name, meaning all available interfaces
```

```
PORT = 8888 # Arbitrary non-privileged port

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Socket created'

#Bind socket to local host and port
try:
    s.bind((HOST, PORT))
except socket.error as msg:
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message
    sys.exit()

print 'Socket bind complete'

#Start listening on socket
s.listen(10)
print 'Socket now listening'

#now keep talking with the client
while 1:
    #wait to accept a connection - blocking call
    conn, addr = s.accept()
    print 'Connected with ' + addr[0] + ':' + str(addr[1])

s.close()
```


The accept function is called in a loop to keep accepting connections from multiple clients.

Run it from the terminal.

```
$ python server.py
Socket created
Socket bind complete
Socket now listening
```

The output says that the socket was created, binded and then put into listening mode. At this point try to connect to this server from another terminal using the telnet command.

```
$ telnet localhost 8888
```

The telnet command should connect to the server right away and the server terminal would show this.

```
$ python server.py
Socket created
Socket bind complete
Socket now listening
Connected with 127.0.0.1:47758
```

So now our socket client (telnet) is connected to the socket server program.

> Telnet (socket client) ==========> Socket server

Handle socket clients with threads

The socket server shown above does not do much apart from accepting an incoming connection. Now its time to add some functionality to the socket server so that it can interact with the connected clients.

```
'''
    Simple socket server using threads
'''

import socket
import sys
from thread import *

HOST = ''   # Symbolic name meaning all available interfaces
PORT = 8888 # Arbitrary non-privileged port

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Socket created'

#Bind socket to local host and port
try:
    s.bind((HOST, PORT))
except socket.error as msg:
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message
    sys.exit()

print 'Socket bind complete'

#Start listening on socket
s.listen(10)
print 'Socket now listening'

#Function for handling connections. This will be used to create t
def clientthread(conn):
    #Sending message to connected client
    conn.send('Welcome to the server. Type something and hit ente

    #infinite loop so that function do not terminate and thread c
    while True:

        #Receiving from client
        data = conn.recv(1024)
        reply = 'OK...' + data
        if not data:
            break

        conn.sendall(reply)

    #came out of loop
    conn.close()

```

```
47  #now keep talking with the client
48  while 1:
49      #wait to accept a connection - blocking call
50      conn, addr = s.accept()
51      print 'Connected with ' + addr[0] + ':' + str(addr[1])
52
53      #start new thread takes 1st argument as a function name to be
54      start_new_thread(clientthread ,(conn,))
55
56  s.close()
```

Run the above server program and connect once again with a telnet from another terminal. This time if you type some message, the socket server will send it back with OK prefixed.

```
$ telnet localhost 8888
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Welcome to the server. Type something and hit enter
hello
OK...hello
how are you
OK...how are you
```

The socket server can handle multiple clients simultaneously by allotting a separate thread to each.

Handle socket clients with select function

Threads appear the most natural way of handling multiple socket connections and clients. However there are other techniques of doing this. Polling is one such technique. In polling, the socket api will continuously check a bunch of sockets for some activity or

event. And if an event occurs in one or multiple sockets, the function returns to the application the list of sockets on which the events occurred.

Such a kind of polling is achieved with the select function. The syntax of the select function is as follows

```
read_sockets,write_sockets,error_sockets = select(read_fds , write
```

The select function takes 3 different sets/arrays of sockets. If any of the socket in the first set is readable or any socket in the second set is writable, or any socket in the third set has an error, then the function returns all those sockets. Next the application can handle the sockets returned and do the necessary tasks.

```
# Socket server in python using select function

import socket, select

if __name__ == "__main__":

    CONNECTION_LIST = []    # list of socket clients
    RECV_BUFFER = 4096 # Advisable to keep it as an exponent of 2
    PORT = 5000

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

    # Add server socket to the list of readable connections
    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 through
        read_sockets,write_sockets,error_sockets = select.select(

        for sock in read_sockets:
```

```
27
28              #New connection
29              if sock == server_socket:
30                  # Handle the case in which there is a new connect
31                  sockfd, addr = server_socket.accept()
32                  CONNECTION_LIST.append(sockfd)
33                  print "Client (%s, %s) connected" % addr
34
35              #Some incoming message from a client
36              else:
37                  # Data recieved from client, process it
38                  try:
39                      #In Windows, sometimes when a TCP program clo
40                      # a "Connection reset by peer" exception will
41                      data = sock.recv(RECV_BUFFER)
42                      # echo back the client message
43                      if data:
44                          sock.send('OK ... ' + data)
45
46                  # client disconnected, so remove from socket list
47                  except:
48                      broadcast_data(sock, "Client (%s, %s) is offl
49                      print "Client (%s, %s) is offline" % addr
50                      sock.close()
51                      CONNECTION_LIST.remove(sock)
52                      continue
53
54      server_socket.close()
```

The select function is given the list of connected sockets CONNECTION_LIST. The 2nd and 3rd parameters are kept empty since we do not need to check any sockets to be writable or having errors.

Output

```
$ python server.py
Chat server started on port 5000
Client (127.0.0.1, 55221) connected
```

Last Updated On : 3rd August 2013

Related Post

Raw socket programming in python (Linux)

Code a network packet sniffer in python for Linux

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

CATEGORY: PYTHON

TAGS: NETWORK PROGRAMMING , PYTHON , SOCKET PROGRAMMING

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 →

25 thoughts on “Code a simple socket server in Python ”

Sarthak

November 22, 2017 at 12:12 pm

If we want to send the address in a different way, is it possible? Something like “{address}:{port}/test.mjpg”. I wish to add the ‘/test.mjpg’ along the IP address and PORT number. Any suggestions?

shenbagaraman

November 17, 2017 at 5:05 pm

```
import socket
import time
import select

sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sock.bind(('192.168.161.31',20001))
```

```
sock.listen(1)
sock.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
clientsocket,addr = sock.accept()
def square(X):
return X*X
def add(a,b):
return a+b
#task = 'sqrt of 16 is,.math.sqrt'
try:
while true:
#square = 10
#for i in range(1,100):
#square = square + 10
clientsocket.sendall(square(X))
time.sleep(10)
```

the above code is correct? and i need to pass function server to client

---

graket

August 4, 2017 at 10:10 pm

So every time I press a key, it will say
"ok.....
I press h
OkH... OkI....
and it doesnt even show up on the server either

---

aysadk

July 22, 2017 at 6:06 pm

you have to update your code with right syntax for print command
print (“my test is cool now”)

Manish

June 18, 2017 at 1:01 am

If i want to keep my socket open forever weather there is client connection or not what should i Do? i want to do my listener socket keep up and listen on port 35520 and insert received values in mysql database. what should i do?

Jessica Levy

December 1, 2016 at 1:01 pm

hi

maybe can help me out…
mine literally doesn’t run
why could that be?

Zachay

May 17, 2018 at 5:05 am

Do you have python?? If not Go to: https://python.org/getit

Yogeesh Seralathan

May 1, 2016 at 12:12 pm

Polling is a nice little hack of handling multiple clients. But is it faster than multithreading.? I assume polling will use fewer resources than multithreading.

Sagar kumar

March 15, 2016 at 10:10 pm

Thanks for tutorial. I haven't tried it yet but I want to ask one thing. Will it work for images like On android app, I want to send image to python script at backend server and server process it and rply me back to the app.???

ted

February 19, 2016 at 2:02 pm

Great tutorial, very helpful, thank you very much. One question, if the pipe was broken due to network connection, how could a client reconnect to the server? Thanks in advance.

hypixus

July 29, 2015 at 5:05 pm

PERFECT tutorial for me. Only things i needed were transforming it to python 3.X and adding Polish letters support, but that was easy with this.

def strToBytes(strToConvert):

return str.encode(strToConvert, 'UTF-8')
def bytesToStr(dataToConvert):
return str(dataToConvert, 'UTF-8')

In every place bugs of TypeError were replaced with these functions – everything was fine.

sly

November 17, 2014 at 6:06 am

This is my second post of the day which should also serve as the answer to my previous question. I found the event handling APIs, asyncore.loop and asyncore.dispatcher. Thank you though

---

sly

November 17, 2014 at 5:05 am

Your programs often used the "while true,," loop to listen to client request(s). Isn't there an event driven check like ~ Server Socket.Listen is ready before Socket.Accept. I think loop is resource intensive. Please advise !

sly

November 17, 2014 at 5:05 am

Your programs often used the “while true,,” loop to listen to client request(s). Isn’t there an event driven check like ~ Server Socket.Listen is ready before Socket.Accept. I think loop is resource intensive. Please advise !

Balaji

October 16, 2014 at 6:06 pm

Hi, I need a help on python programming by creating Server connectivity. Any one kindly suggest me on the following is feasible . Server1 and Server2 will have the single socket connection and the connection should always active. Now the Server1 listen to Client 1 and get the messages and forward to Server 2 by using the available connection and get the response and send it back to the client.

BabyCakes

October 16, 2014 at 6:06 am

Hi Silver Moon, thanks for the tutorials. What’s going on with the broadcast_data function? There is no such function provided in your code.

alva

July 4, 2014 at 6:06 pm

Thats you. I landed in this page and solved my problem. Great tutorial, thanks!

Silver Moon

January 9, 2014 at 10:10 am

tutorial on coding chat server and client can be found here
https://www.binarytides.com/code-chat-application-server-client-sockets-python/

---

test

September 18, 2013 at 3:03 am

Traceback (most recent call last):

File "./test.py", line 48, in

broadcast_data(sock, "Client %s: %s is offline" % addr)

NameError: name 'broadcast_data' is not defined

BabyCakes

October 16, 2014 at 6:06 am

Perhaps a function that the author neglected to include? I checked, it's not a socket method.

Steve N

November 27, 2014 at 6:06 pm

it can be written as :

```
def broadcast(connected):

while connected:
data = connected.recv(BUFFER_SIZE)

if data:
data = str(connected.getpeername()) + 'says –>' + data
for c in client_list:
if c is not connected:
c.sendall(data)
else:
client_index = client_list.index(connected)
print 'Client' + str(connected.getpeername()) + ' has left the chat.'
client_list.pop(client_index)
print 'The are ' + str(len(client_list)) + ' remaining clients'

break
```

Edwardo

September 10, 2013 at 1:01 am

Hi, How I can disconnect a client? I close (with the "x" gui button) but It does not broadcast the except error… Also, I want to made a quit from the client, like hitting q

for quit. I made in line 43 if data != ‘q’ then send otherwise close. Why it does not work?

anu nivas

August 28, 2013 at 10:10 am

Sorry, I had wrongly named my file as select.py.
Thank you

anu nivas

August 28, 2013 at 10:10 am

At line 24 I get the error, in
read_sockets,write_sockets,error_sockets = select(CONNECTION_LIST,[],[])
TypeError: ‘module’ object is not callable
Could you please assist?

s dubbya

December 27, 2014 at 6:06 am

select.select
