Python - Simple Networking Tutorial

By - Michael Tubinis (mtubinis)

Networking Basics

What is Networking?

- Simply the communication between multiple machines

Networking Basics

How does Networking work in Python?

- Sockets!

What is a Socket?!

Well there are generally two types of sockets, they look something like this:

What is a Socket?!

Okay, yeah, they look like pipes. What does that mean?

- Notice how one pipe had one start point and one end point.
- The other had one starting point (the center), and multiple ending points.

The Client Socket

The “pipe” with one starting point and one ending point is used by clients to talk to a server.

The Server Socket

The multiple ending “pipe” aka the “listening” socket, is used by the server to listen for, accept, and create sockets for each of the clients attempting to connect.

Server Socket Code - server.py

```
import socket,json
from threading import Thread
from urllib2 import urlopen

# returns your external IP Address
def GetIp():
    try:
        return json.load(urlopen('http://httpbin.org/ip'))['origin']
    except:
        print("Could not obtain IP Address")
        return "0.0.0.0"
```

Function that returns your external IP Address. Otherwise binds to 0.0.0.0 if it cannot get your external IP. 0.0.0.0 is equivalent to “localhost”

```
self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
self.host = GetIp()        # GetIp() returns your external IP Address
self.port = 6119
self.socket.bind((self.host, self.port))
print(self.host + " Listening on port: " + str(self.port))
self.socket.listen(10)
```

Create a socket, bind it to a host and port (host is your external ip, port should be in the thousands somewhere), and have the socket listen for connections.

Do this code in an init function.

Server Socket Code - server.py

```
try:
    connection, address = self.socket.accept()      #Note - this function blocks
                                                    #execution, the code below will
                                                    #not run until a client
                                                    #actually connects

    thread = ClientThread(connection,address)

    thread.start()                                  #the connection socket should be
                                                    #stored in a thread, as the function
                                                    #that processes data sent by clients
    self.server.clients[address[0]] = thread        #also blocks execution
    print(address[0] + " connected!")
except socket.error as msg:
    print(msg)
    break
```

Have this code run in the main loop

Server Socket Code - server.py

```
class ClientThread(Thread):
    def __init__(self,socket,address):
        super(ClientThread, self).__init__()
        self.running = True
        self.connection = socket
        self.address = address

    def run(self):
        while self.running:
            try:
                data = self.connection.recv(2048)          #Note - this function blocks execution
                                                           #until the client actually sends data over
                if data == “”:
                    print("Client: " + self.address[0] + " disconnected.")
                    self.running = False
                else:
                    #process data here
                    print(self.address[0] + " said: " + data)
            except socket.error as msg:
                print(msg)
                self.running = False
        print("Closing " + self.address[0] + "'s connection.")
        self.conn.close()
```

The client thread class seen in the previous slide. There will be one of these for each client connected.

Client Socket Code - client.py

```
import socket,sys


sys.stdout.write("Specify server ip to connect to: ")
server = raw_input()
sys.stdout.write("Specify port to connect to: ")
port = raw_input()

self.connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    self.connection.connect((server, int(port)))
    self.running = True
except:
    print("Could not connect to the server.")
    self.running = False
```

Run this in an init function

Client Socket Code - client.py

```
while self.running:
    userinput = raw_input()
    if userinput == "quit":
        print("Closing...")
        self.connection.close()
        break
    self.connection.send(userinput)
```

Run this in the main function

Questions?

https://github.com/mstubinis/SimplePythonNetworking

^ for server.py, client.py, and these slides