Lab 4

In the previous lab you became familiar with socket programming in C. This time you will continue the socket programming using Python as we think it can be more practical for you in your career. Also, it provides a good opportunity for the students who have not tried programming in Python to learn this language. We believe that as students who have passed lots of CS courses, it shouldn't be difficult for you to start programming in Python even if it is the first time you are using it.

Click on the link below, to start with a very simple, clear, and concise tutorial and some experiments in Python socket programming:

http://www.binarytides.com/python-socket-programming-tutorial/

Start from the very first line, do the experiments, read all the parts, and follow all the steps to the end. (It is a perfect tutorial for socket programming in Python. Besides that, it will help you to understand socket programming and network concepts too. So please do not skip any step and study all the provided code. Pay close attention to the differences between each step.) Whenever you want to select a port number for a server, please use the four rightmost digits of your student ID. If your student ID is 861109596, you should assign 9596 as the port number of the server. (If your port number is less than 2000, add 2000 to it) When you want to make a connection to a website (in the tutorial, google.com is used), use the first result from a google search of your firstname. (Similar to previous labs.) Take screenshots of your experiments similar to the provided screenshots in the tutorial. **Note:** you can run your client.py and server.py scripts on your lab machine and telnet from your mininet VM to your lab machine; or run everything on mininet VM.

After finishing that tutorial, add the following enhancements to your server code:

1) If the text "!q" is sent from the client to the server, have the server close the connection. (For this, it is enough to just check the first two characters of the received text and ignore anything else.)
2) If the text "!sendall <message>" is sent to the server, have the server send the <message> to all connected clients.

Demo: All the screenshots of the tutorial experiments should be available to show to your TA. You should be able to explain every single line of your code similar to the explanations in the provided link. Your TA will use telnet to make two parallel connections and he will test normal and broadcast messages to the server.

Some Python code you may need:

#Define a new list

```
myList = []
```

#add a member to a list

```
myList.append(member)
```

#get substring of a string

```
print data[0:5]
```

if data is 0123456, it will print 01234

#for loop on a list

```
for member in myList:
    print member
```

#if condition

```
if condition :
    print 'Condition is true'
```

To avoid port reuse problem, you can use this right after calling the socket creation function:

```
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

Good Luck!