DEVELOPING CHAT APPLICATION IN PYTHON
(TCP Protocol implementation)

The Chat Application is very common today offered either via a web application or mobile application. Learning to write a Chat Application is good for understanding many network communication concepts and can be useful to build other network applications. Chat Application provides communication between two parties i.e. sender and receiver. The sender is someone who initiates and sends a message to another known as receiver; receiver at other end receives the message. The role of sender and receiver is not fixed and keeps exchanging during communication, so in simple words, at a point, someone who sends the message is a sender and who receives the message is called receiver. In networking terms, sender and receiver are denoted as source and destination respectively.

Communication can be of many types depending upon the method of communication and the number of parties involved. Some of the scenarios are :

1. **Simplex or one-way communication:** Only one party is able to send the message and other parties can only receive.
2. **Duplex or two-way communication:** Both parties can send and receive messages.

In chat application we have two write two scripts

1. Server Scripts
2. Client Scripts

Server Scripts

Server program has all the logic to control and regulate the Chat, so most of the chat logic is implemented with a server program. So the first step of communication is to identify the users, how does the server do this? In network communication, users are identified by a socket which is nothing but a combination of IP address and port address. So, for human understanding, Alice and Bob will be chatting but for a network, it is two sockets processes which are sending and receiving bytes. Steps involved in this process is as follows:

1. Create socket
2. Communicate the socket address
3. Keep waiting for an incoming connection request/s
4. Connect to client
5. Receive the message

6. Decode the destination user and select the socket
7. Send a message to the intended client
8. Keep repeating step 5 & 6 as per users wish
9. Exit i.e. end the communication by terminating the connection

server-chat.py

```
import socket,select
port = 12345
socket_list = []
users = {}
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind(('',port))
server_socket.listen(5)
socket_list.append(server_socket)
while True:
   ready_to_read,ready_to_write,in_error = select.select(socket_list,[],[],0)
   for sock in ready_to_read:
       if sock == server_socket:
           connect, addr = server_socket.accept()
           socket_list.append(connect)
           connect.send("You are connected from:" + str(addr))
       else:
           try:
               data = sock.recv(2048)
               if data.startswith("#"):
                   users[data[1:].lower()]=connect
                   print "User " + data[1:] +" added."
                   connect.send("Your user detail saved as : "+str(data[1:]))
               elif data.startswith("@"):

users[data[1:data.index(':')].lower()].send(data[data.index(':')+1:])
           except:
               continue
server_socket.close()
```

Client script: Client script is run by the user, so the same client code will be run by a different user but each will have a separate socket so they will have their unique communication channel. Client script used to be thin because it has very less work i.e. it only connects with the server and sends and receives messages. The steps involved in client script are:

1. Create a unique client socket per instance/user
2. Connect to the server with given socket address (IP and port)
3. Send and receive messages
4. Repeat step 3 as per configuration

5. Close the connection

Client-chat.py

```
import socket
client_socket = socket.socket()
port = 12345
client_socket.connect(('127.0.0.1',port))
#recieve connection message from server
recv_msg = client_socket.recv(1024)
print recv_msg
#send user details to server
send_msg = raw_input("Enter your user name(prefix with #):")
client_socket.send(send_msg)
#receive and send message from/to different user/s
while True:
  recv_msg = client_socket.recv(1024)
  print recv_msg
  send_msg = raw_input("Send your message in format [@user:message] ")
  if send_msg == 'exit':
     break;
  else:
     client_socket.send(send_msg)
client_socket.close()
```

In the above program, each user has to run the **client script** separately after the **server script** is running. Once the client program **connects** to the **server** the **client** has to register itself as a user by giving a username, so the rest of the communication will be done using the username.

Steps for running the sample Chat application:

1. Open a terminal and Run the server-chat.py

2. Open a new terminal and run client-chat.py

a) Enter the username with a '#' prefix. Example: *#ali*

b) Now, send the message to a user by following the format @username:message. Example: @Ahmad:Hello, Ahmad! This is alice

3. Repeat *step 2* for other users. (Maximum 5 users is allowed with server configuration i.e. server_socket.listen(5)

Tasks

Write chat application that can connect 10 client and these 10 client communicate with each other