GUI based chat application

Server:

*from* socket *import* AF\_INET, socket, SOCK\_STREAM

*from* threading *import* Thread

def accept\_incoming\_connections():

*while* True:

        client, client\_address = SERVER.accept()

        print("%s:%s has connected." % client\_address)

        client.send(

            bytes("Greetings from the cave! Now type your name and press enter!", "utf8"))

        addresses[client] = client\_address

*# print("addresses: ", addresses)*

        HANDLE\_THREAD = Thread(target=handle\_client, args=(client,))

        HANDLE\_THREAD.start()

def handle\_client(client):  *# Takes client socket as argument.*

*"""Handles a single client connection."""*

    name = client.recv(BUFSIZ).decode("utf8")

    welcome = 'Welcome %s! If you ever want to quit, type {quit} to exit.' % name

    client.send(bytes(welcome, "utf8"))

    msg = "%s has joined the chat!" % name

    broadcast(bytes(msg, "utf8"))

    clientsNames[client] = name

*while* True:

        msg = client.recv(BUFSIZ)

*if* msg != bytes("{quit}", "utf8"):

            broadcast(msg, name+": ")

*else*:

            client.send(bytes("{quit}", "utf8"))

            client.close()

*del* clientsNames[client]

            broadcast(bytes("%s has left the chat." % name, "utf8"))

*break*

def broadcast(msg, prefix=""):  *# prefix is for name identification.*

*"""Broadcasts a message to all the clients."""*

*# for cl in clients.values():*

*# print("All connected clients: ",  cl)*

    print("All connected clients: ",  clientsNames.values())

*for* sock *in* clientsNames:

        print("sock: ", sock)

        sock.send(bytes(prefix, "utf8") + msg)

clientsNames = {}

addresses = {}

HOST = ''

PORT = 33000

BUFSIZ = 1024

ADDR = (HOST, PORT)

SERVER = socket(AF\_INET, SOCK\_STREAM)

SERVER.bind(ADDR)

*if* \_\_name\_\_ == "\_\_main\_\_":

    SERVER.listen(5)

    print("Waiting for connection...")

    ACCEPT\_THREAD = Thread(target=accept\_incoming\_connections)

    ACCEPT\_THREAD.start()

    ACCEPT\_THREAD.join()

    SERVER.close()

Client 1:

*from* socket *import* AF\_INET, socket, SOCK\_STREAM

*from* threading *import* Thread

*import* tkinter

def receive():

*"""Handles receiving of messages."""*

*while* True:

*try*:

            msg = client\_socket.recv(BUFSIZ).decode("utf8")

            msg\_list.insert(tkinter.END, msg)

*except* OSError:  *# Possibly client has left the chat.*

*break*

def send(event=None):  *# event is passed by binders.*

*"""Handles sending of messages."""*

    msg = my\_msg.get()

    my\_msg.set("")  *# Clears input field.*

    client\_socket.send(bytes(msg, "utf8"))

*if* msg == "{quit}":

        client\_socket.close()

        top.quit()

def on\_closing(event=None):

*"""This function is to be called when the window is closed."""*

    my\_msg.set("{quit}")

    send()

top = tkinter.Tk('300x900')

top.title("Chatter")

messages\_frame = tkinter.Frame(top)

my\_msg = tkinter.StringVar()  *# For the messages to be sent.*

my\_msg.set("Type your Name here.")

scrollbar = tkinter.Scrollbar(messages\_frame)  *# To navigate through past messages.*

*# Following will contain the messages.*

msg\_list = tkinter.Listbox(messages\_frame, height=15, background='#063579', fg='White', width=50, yscrollcommand=scrollbar.set, font=("Calibri",14))

scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)

msg\_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)

msg\_list.pack()

messages\_frame.pack()

entry\_field = tkinter.Entry(top, textvariable=my\_msg, width=30,font=("Calibri",14))

entry\_field.bind("<Return>", send) *# send the content of the entry if {enter} button pressed*

entry\_field.pack()

send\_button = tkinter.Button(top, text="Send", command=send,background='#063579', fg='White', font=("Calibri",14))

send\_button.pack()

top.protocol("WM\_DELETE\_WINDOW", on\_closing)

HOST = '127.0.0.1'

PORT = 33000

BUFSIZ = 1024

ADDR = (HOST, PORT)

client\_socket = socket(AF\_INET, SOCK\_STREAM)

client\_socket.connect(ADDR)

receive\_thread = Thread(target=receive)

receive\_thread.start()

tkinter.mainloop()

Client 2:

*from* socket *import* AF\_INET, socket, SOCK\_STREAM

*from* threading *import* Thread

*import* tkinter

def receive():

*"""Handles receiving of messages."""*

*while* True:

*try*:

            msg = client\_socket.recv(BUFSIZ).decode("utf8")

            msg\_list.insert(tkinter.END, msg)

*except* OSError:  *# Possibly client has left the chat.*

*break*

def send(event=None):  *# event is passed by binders.*

*"""Handles sending of messages."""*

    msg = my\_msg.get()

    my\_msg.set("")  *# Clears input field.*

    client\_socket.send(bytes(msg, "utf8"))

*if* msg == "{quit}":

        client\_socket.close()

        top.quit()

def on\_closing(event=None):

*"""This function is to be called when the window is closed."""*

    my\_msg.set("{quit}")

    send()

top = tkinter.Tk('300x900')

top.title("Chatter")

messages\_frame = tkinter.Frame(top)

my\_msg = tkinter.StringVar()  *# For the messages to be sent.*

my\_msg.set("Type your Name here.")

scrollbar = tkinter.Scrollbar(messages\_frame)  *# To navigate through past messages.*

*# Following will contain the messages.*

msg\_list = tkinter.Listbox(messages\_frame, height=15, background='#063579', fg='White', width=50, yscrollcommand=scrollbar.set, font=("Calibri",14))

scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)

msg\_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)

msg\_list.pack()

messages\_frame.pack()

entry\_field = tkinter.Entry(top, textvariable=my\_msg, width=30,font=("Calibri",14))

entry\_field.bind("<Return>", send)

entry\_field.pack()

send\_button = tkinter.Button(top, text="Send", command=send,background='#063579', fg='White', font=("Calibri",14))

send\_button.pack()

top.protocol("WM\_DELETE\_WINDOW", on\_closing)

HOST = '127.0.0.1'

PORT = 33000

BUFSIZ = 1024

ADDR = (HOST, PORT)

client\_socket = socket(AF\_INET, SOCK\_STREAM)

client\_socket.connect(ADDR)

receive\_thread = Thread(target=receive)

receive\_thread.start()

tkinter.mainloop()

OutPut:

