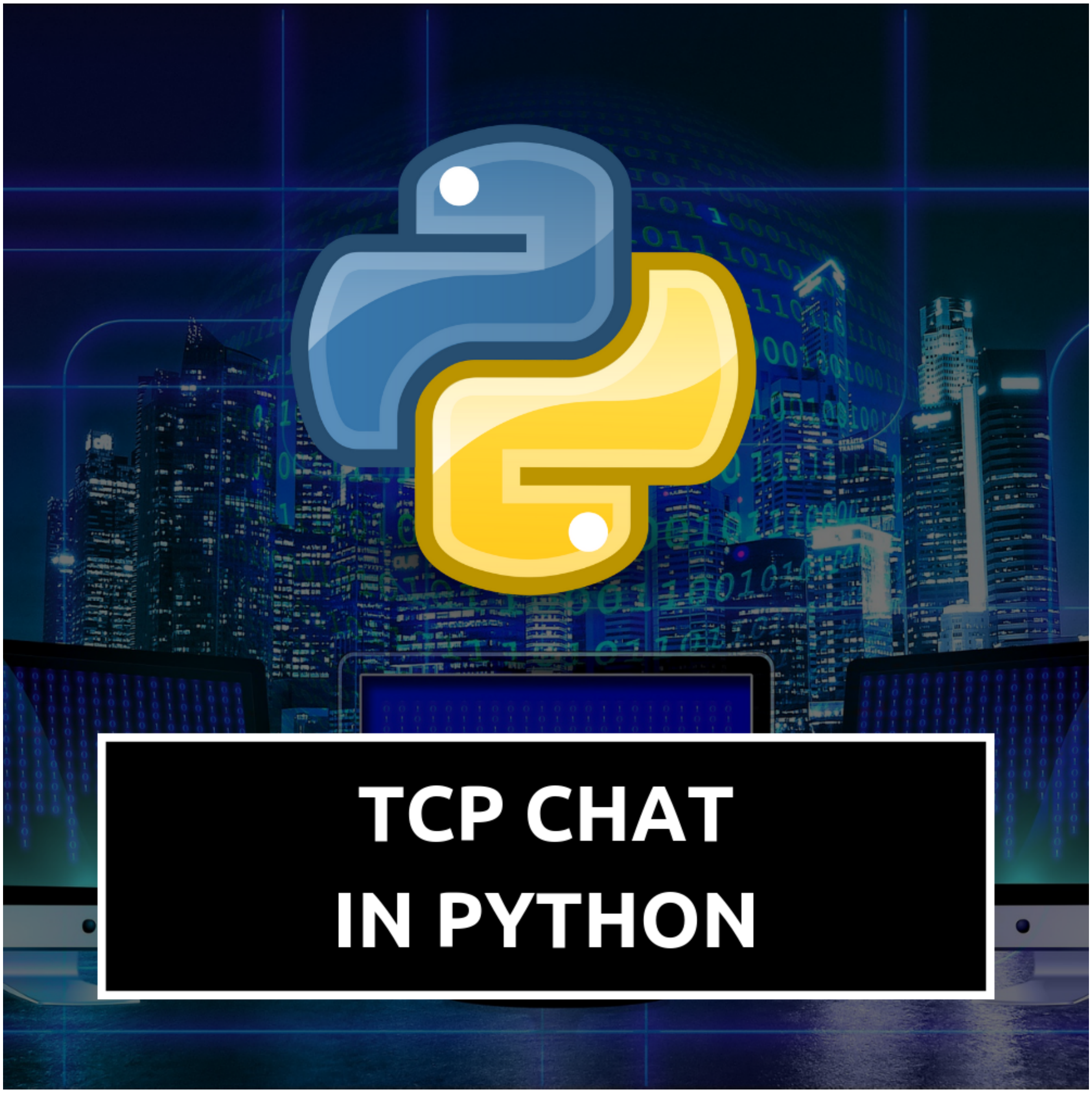
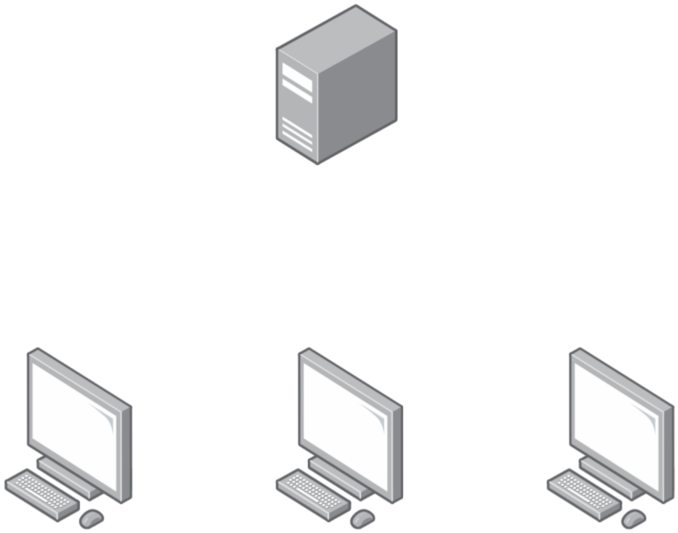


TCP CHAT IN PYTHON

TCP CHAT IN PYTHON

Client -server Architecture

For our application, we will use the client-server architecture. This means that we will have multiple clients (the users) and one central server that hosts everything and provides the data for these clients.



Therefore, we will need to write two Python scripts. One will be for starting the server and one will be for the client. We will have to run the server first, so that there is a chat, which the clients can connect to. The clients themselves are not going to directly communicate to each other but via the central server.

Implementing The client

A server is pretty useless without clients that connect to it. So now we are going to implement our client. For this, we will again need to import the same libraries. Notice that this is now a second separate script.

```
import socket
import threading
```

The first step of the client are to choose a nickname and to connect to our server . we will need to know the exact address and the poet at which our server is running .

```
# Choosing Nickname
nickname = input("Choose your nickname: ")

# Connecting To Server
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 55555))
```

As you can see, we using a different function here. Instead of binding the data and listing , we are connecting to an existing server.

Now, a client needs to have two threads that are running at the same time. The first one will constantly receive data from the server and the second one will send our own messages to the server. So we will need two functions here. Let’s start with the receiving part

```
# Listening to Server and Sending Nickname
def receive():
    while True:
        try:
            # Receive Message From Server
            # If 'NICK' Send Nickname
            message = client.recv(1024).decode('ascii')
            if message == 'NICK':
                client.send(nickname.encode('ascii'))
            else:
                print(message)
        except:
            # Close Connection When Error
            print("An error occured!")
            client.close()
            Break
```

Again we have an endless while-loop here. It constantly tries to receive messages and to print them onto the screen. If the message is ‘NICK’ however, it doesn’t print it but it sends its nickname to the server. In case there is some error, we close the connection and break the loop. Now we just need a function for sending messages and we are almost done

What it then does is receiving the message from the client (if he sends any) and broadcasting it to all connected clients. So when one client sends a message, everyone else can see this message. Now if for some reason there is an error with the connection to this client, we remove it and its nickname, close the connection and broadcast that this client has left the chat. After that we break the loop and this thread comes to an end. Quite simple. We are almost done with the server but we need one final function.

When we are ready to run our server, we will execute this receive function. It also starts an endless while-loop which constantly accepts new connections from clients. Once a client is connected it sends the string 'NICK' to it, which will tell the client that its nickname is requested. After that it waits for a response (which hopefully contains the nickname) and appends the client with the respective nickname to the lists. After that, we print and broadcast this information. Finally, we start a new thread that runs the previously implemented handling function for this particular client. Now we can just run this function and our server is done.

Notice that we are always encoding and decoding the messages here. The reason for this is that we can only send bytes and not strings. So we always need to encode messages (for example using ASCII), when we send them and decode them, when we receive them.

Implementing The Client

A server is pretty useless without clients that connect to it. So now we are going to implement our client. For this, we will again need to import the same libraries. Notice that this is now a second separate script.

Now, a client needs to have two threads that are running at the same time. The first one will constantly receive data from the server and the second one will send our own messages to the server. So we will need two functions here. Let's start with the receiving part

Again we have an endless while-loop here. It constantly tries to receive messages and to print them onto the screen. If the message is ‘NICK’ however, it doesn’t print it but it sends its nickname to the server. In case there is some error, we close the connection and break the loop. Now we just need a function for sending messages and we are almost done.

```
def write():
    while True:
        message = '{}: {}'.format(nickname, input(''))
        client.send(message.encode('ascii'))
```

The writing function is quite a short one. It also runs in an endless loop which is always waiting for an input from the user. Once it gets some, it combines it with the nickname and sends it to the server. That’s it. The last thing we need to do is to start two threads that run these two functions

```
receive_thread = threading.Thread(target=receive)
receive_thread.start()

write_thread = threading.Thread(target=write)
write_thread.start()
```

And now we are done. We have a fully-functioning server and working clients that can connect to it and communicate with each other.

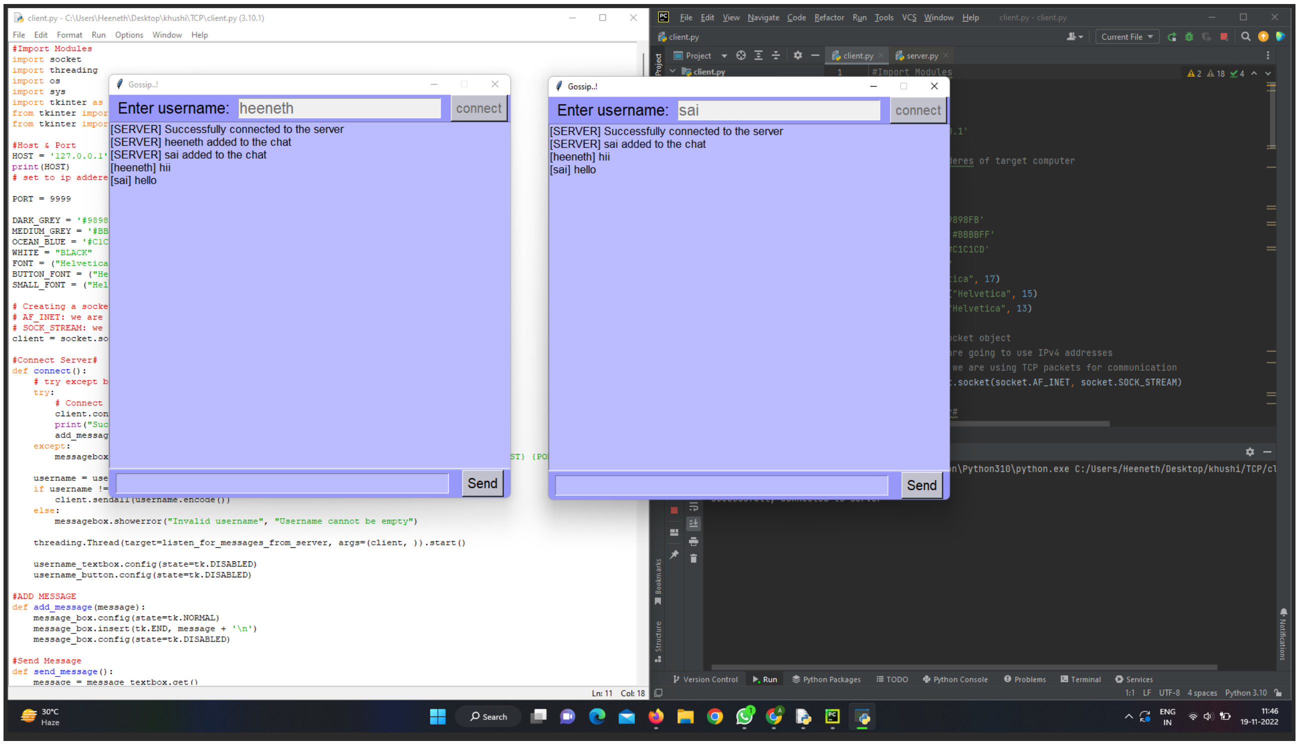
client.py

```
#Import Modules
import socket
import threading
import os
import sys
import tkinter as tk
from tkinter import scrolledtext
from tkinter import messagebox

#Host & Port
HOST = '127.0.0.1'
print(HOST)
# set to ip adderes of target computer

PORT = 9999

DARK_GREY = '#9898FB'
MEDIUM_GREY = '#BBBBFF'
OCEAN_BLUE = '#C1C1CD'
WHITE = "BLACK"
FONT = ("Helvetica", 17)
BUTTON_FONT = ("Helvetica", 15)
SMALL_FONT = ("Helvetica", 13)

# Creating a socket object
# AF_INET: we are going to use IPv4 addresses
# SOCK_STREAM: we are using TCP packets for communication
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Connect Server#
def connect():
    # try except block
    try:
        # Connect to the server
        client.connect((HOST, PORT))
        print("Successfully connected to server")
        add_message("[SERVER] Successfully connected to the server")
    except:
        messagebox.showerror("Unable to connect to server", f"Unable to connect
to server {HOST} {PORT}")

    username = username_textbox.get()
    if username != '':
        client.sendall(username.encode())
    else:
        messagebox.showerror("Invalid username", "Username cannot be empty")

    threading.Thread(target=listen_for_messages_from_server, args=(client,
)).start()

    username_textbox.config(state=tk.DISABLED)
    username_button.config(state=tk.DISABLED)

#ADD MESSAGE
def add_message(message):
    message_box.config(state=tk.NORMAL)
    message_box.insert(tk.END, message + '\n')
    message_box.config(state=tk.DISABLED)

#Send Message
def send_message():
    message = message_textbox.get()
    if message != '':
        client.sendall(message.encode())
        message_textbox.delete(0, len(message))
    else:
        messagebox.showerror("Empty message", "Message cannot be empty")

#TKINTER
root = tk.Tk()
root.geometry("600x600")
root.title("Gossip..!")
root.resizable(False, False)

root.grid_rowconfigure(0, weight=1)
root.grid_rowconfigure(1, weight=4)
root.grid_rowconfigure(2, weight=1)

top_frame = tk.Frame(root, width=600, height=100, bg=DARK_GREY)
top_frame.grid(row=0, column=0, sticky=tk.NSEW)

middle_frame = tk.Frame(root, width=600, height=400, bg=MEDIUM_GREY)
middle_frame.grid(row=1, column=0, sticky=tk.NSEW)

bottom_frame = tk.Frame(root, width=600, height=100, bg=DARK_GREY)
bottom_frame.grid(row=2, column=0, sticky=tk.NSEW)

username_label = tk.Label(top_frame, text="Enter username:", font=FONT,
bg=DARK_GREY, fg=WHITE)
username_label.pack(side=tk.LEFT, padx=10)

username_textbox = tk.Entry(top_frame, font=FONT, bg=MEDIUM_GREY, fg=WHITE,
width=23)
username_textbox.pack(side=tk.LEFT)
```

```
username_button = tk.Button(top_frame, text="connect", font=BUTTON_FONT,
bg=OCEAN_BLUE, fg=WHITE, command=connect)
username_button.pack(side=tk.LEFT, padx=15)

message_textbox = tk.Entry(bottom_frame, font=FONT, bg=MEDIUM_GREY, fg=WHITE,
width=38)
message_textbox.pack(side=tk.LEFT, padx=10)

message_button = tk.Button(bottom_frame, text="Send", font=BUTTON_FONT,
bg=OCEAN_BLUE, fg=WHITE, command=send_message)
message_button.pack(side=tk.LEFT, padx=10)

message_box = scrolledtext.ScrolledText(middle_frame, font=SMALL_FONT,
bg=MEDIUM_GREY, fg=WHITE, width=67, height=26.5)
message_box.config(state=tk.DISABLED)
message_box.pack(side=tk.TOP)

#================Listen for Incomming Message===============#
"""
The incomming message is encoded in utf-8 , which we need to decode to print
in the tkinter box.
"""
def listen_for_messages_from_server(client):
    while 1:
        message = client.recv(2048).decode('utf-8')
        if message != '':
            username = message.split("~")[0]
            content = message.split('~')[1]
            add_message(f"[{username}] {content}")

        else:
            messagebox.showerror("Error", "Message recevied from client is
empty")

#Main
def main():
    root.mainloop()

if __name__ == '__main__':
    main()
```

server.py

```
# Import required modules
import socket
import threading

# HOST = '127.0.0.1'
HOST = '127.0.0.1'

#Use same IPV4 adress
print(HOST)
PORT = 9999 # You can use any port between 0 to 65535
LISTENER_LIMIT = 10
active_clients = [] # List of all currently connected users


def main():

    # Creating the socket class object
    # AF_INET: we are going to use IPv4 addresses
    # SOCK_STREAM: we are using TCP packets for communication
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Creating a try catch block
    try:
        # Provide the server with an address in the form of
        # host IP and port
        server.bind((HOST, PORT))
        print(f"Running the server on {HOST} {PORT}")
    except:
        print(f"Unable to bind to host {HOST} and port {PORT}")

    # Set server limit
    server.listen(LISTENER_LIMIT)

    # This while loop will keep listening to client connections
    while 1:

        client, address = server.accept()
        print(f"Successfully connected to client {address[0]} {address[1]}")

        threading.Thread(target=client_handler, args=(client, )).start()

# Function to listen for upcoming messages from a client
def listen_for_messages(client, username):

    while 1:
        message = client.recv(2048).decode('utf-8')
        if message != '':

            final_msg = username + '~' + message
            send_messages_to_all(final_msg)

        else:
            print(f"The message send from client {username} is empty")


# Function to send message to a single client
def send_message_to_client(client, message):

    client.sendall(message.encode())

# Function to send any new message to all the clients that
# are currently connected to this server
def send_messages_to_all(message):

    for user in active_clients:

        send_message_to_client(user[1], message)

# Function to handle client
def client_handler(client):

    # Server will listen for client message that will
    # Contain the username
    while 1:

        username = client.recv(2048).decode('utf-8')
        if username != '':
            active_clients.append((username, client))
            prompt_message = "SERVER~" + f"{username} added to the chat"
            send_messages_to_all(prompt_message)
            break
        else:
            print("Client username is empty")

    threading.Thread(target=listen_for_messages, args=(client, username,
)).start()
if __name__ == '__main__':
    main()
```



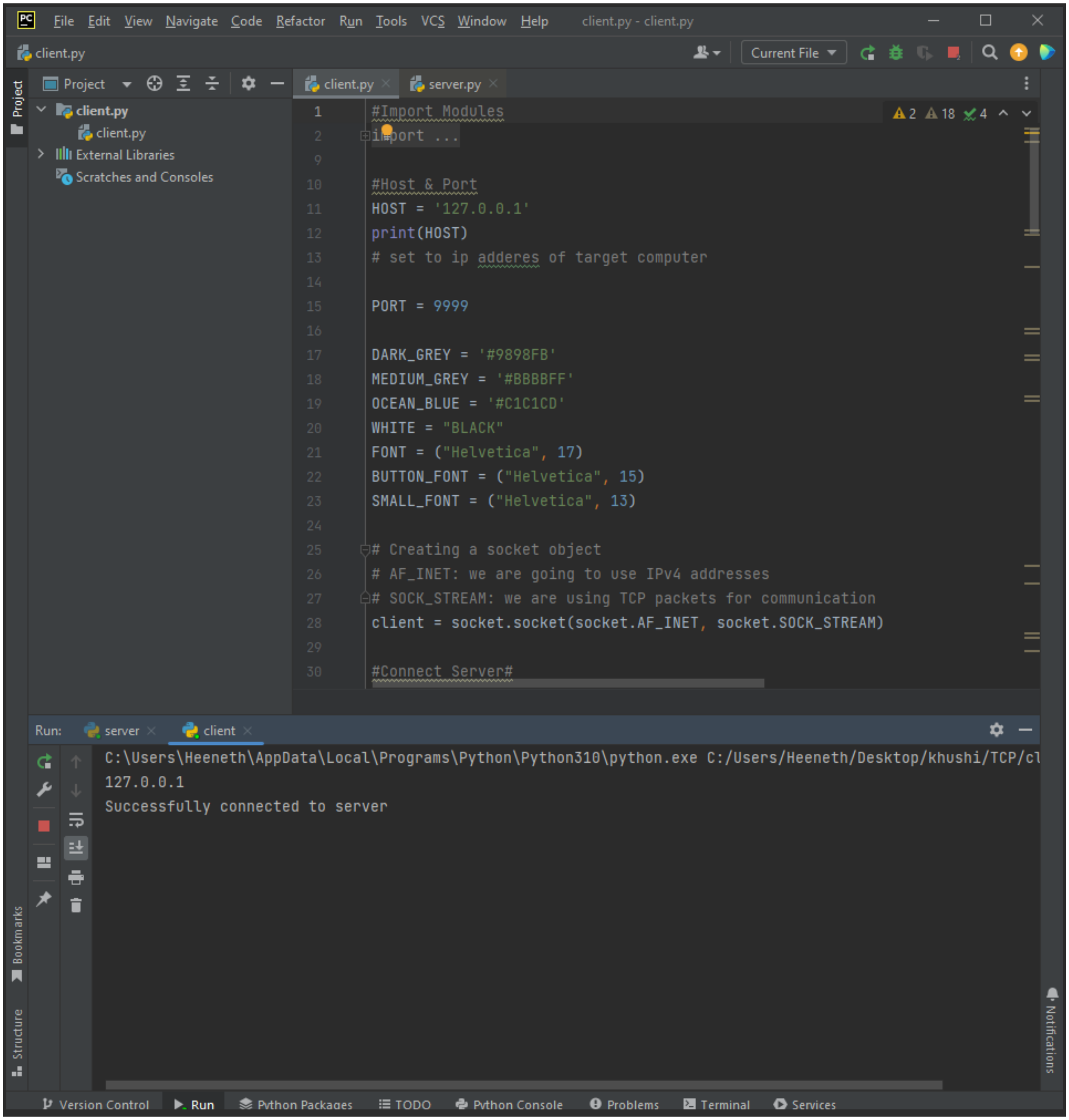
```
#Import Modules
import socket
import threading
import os
import sys
import tkinter as
from tkinter impor
from tkinter impor

#Host & Port
HOST = '127.0.0.1'
print(HOST)
# set to ip addere

PORT = 9999

DARK_GREY = '#9898
MEDIUM_GREY = '#BB
OCEAN_BLUE = '#C1C
WHITE = "BLACK"
FONT = ("Helvetica
BUTTON_FONT = ("He
SMALL_FONT = ("Hel

# Creating a socke
# AF_INET: we are
# SOCK_STREAM: we
client = socket.so

#Connect Server#
def connect():
    # try except b
    try:
        # Connect
        client.con
        print("Suc
        add_messag
    except:
        messagebox                                                  ST) {PO

    username = use
    if username !=
        client.sendall(username.encode())
    else:
        messagebox.showerror("Invalid username", "Username cannot be empty")

    threading.Thread(target=listen_for_messages_from_server, args=(client, )).start()

    username_textbox.config(state=tk.DISABLED)
    username_button.config(state=tk.DISABLED)

#ADD MESSAGE
def add_message(message):
    message_box.config(state=tk.NORMAL)
    message_box.insert(tk.END, message + '\n')
    message_box.config(state=tk.DISABLED)

#Send Message
def send_message():
    message = message_textbox.get()
```

```
[SERVER] Successfully connected to the server
[SERVER] heeneth added to the chat
[SERVER] sai added to the chat
[heeneth] hii
[sai] hello
```

```
[SERVER] Successfully connected to the server
[SERVER] sai added to the chat
[heeneth] hii
[sai] hello
```

```
#Import Modules
.1'
deres of target computer
898FB'
#BBBBFF'
C1C1CD'
ica", 17)
("Helvetica", 15)
'Helvetica", 13)
cket object
are going to use IPv4 addresses
we are using TCP packets for communication
.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
n\Python310\python.exe C:/Users/Heeneth/Desktop/khushi/TCP/cl
```



```
#Import Modules
import ...

#Host & Port
HOST = '127.0.0.1'
print(HOST)
# set to ip adderes of target computer

PORT = 9999

DARK_GREY = '#9898FB'
MEDIUM_GREY = '#BBBBFF'
OCEAN_BLUE = '#C1C1CD'
WHITE = "BLACK"
FONT = ("Helvetica", 17)
BUTTON_FONT = ("Helvetica", 15)
SMALL_FONT = ("Helvetica", 13)

# Creating a socket object
# AF_INET: we are going to use IPv4 addresses
# SOCK_STREAM: we are using TCP packets for communication
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Connect Server#
```

```
C:\Users\Heeneth\AppData\Local\Programs\Python\Python310\python.exe C:/Users/Heeneth/Desktop/khushi/TCP/cl
127.0.0.1
Successfully connected to server
```