File Sharing App - 2

What is our GOAL for this MODULE?

In this class, we have applied our knowledge of FTP and we did the second part of the File sharing app. During the second part of the module, we discussed how to connect to the chat server, and refresh button functionality. The goal of this module is to learn how to make our GUI buttons work.

What did we ACHIEVE in the class TODAY?

- Connect to chat server function
- Refresh working function

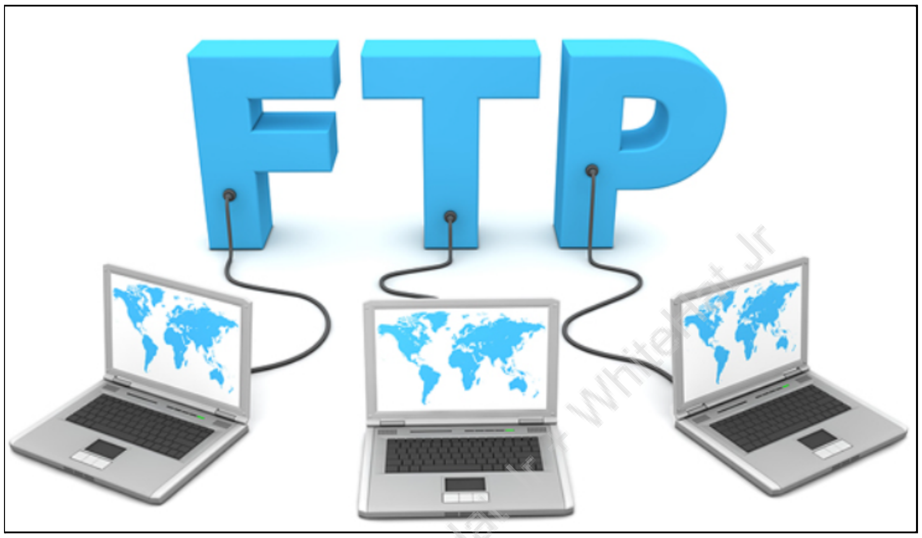
Which CONCEPTS/CODING BLOCKS did we cover today?

- We learned how to make connect to chat button functional
- We learned how to use list for refresh button

The KEY CONCEPT

1. What is FTP?

   The File Transfer Protocol is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network



File transfer protocol (FTP) is a set of rules that computers follow for the transferring of files from one system to another over the internet. It may be used by a business to transfer files from one computer system to another, or websites may use FTP to upload or download files from a website's server.

How did we DO the activities?

1. Devices needed to create a FTP
   - Server
   - Client
   - GUI
   - FTP
   - Other functions

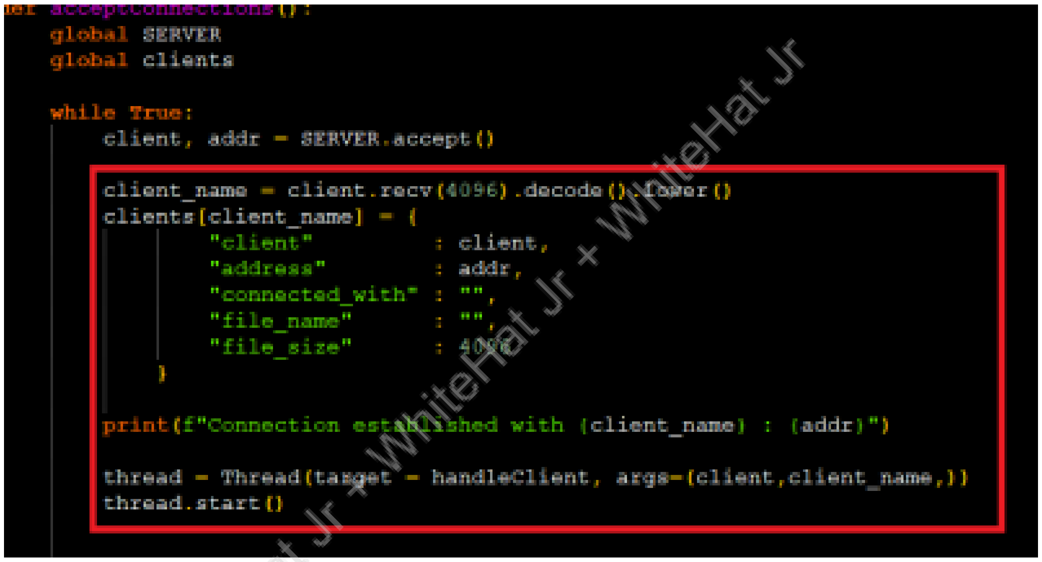
2. Download the boiler plate code, which contains the code to create a server and client

3. Enhance **acceptConnections()** in order to save client information

   - Create the variable client_name where it will store client information that will be received using **recv()** ,decode it and then convert it into lower using **lower()** method.

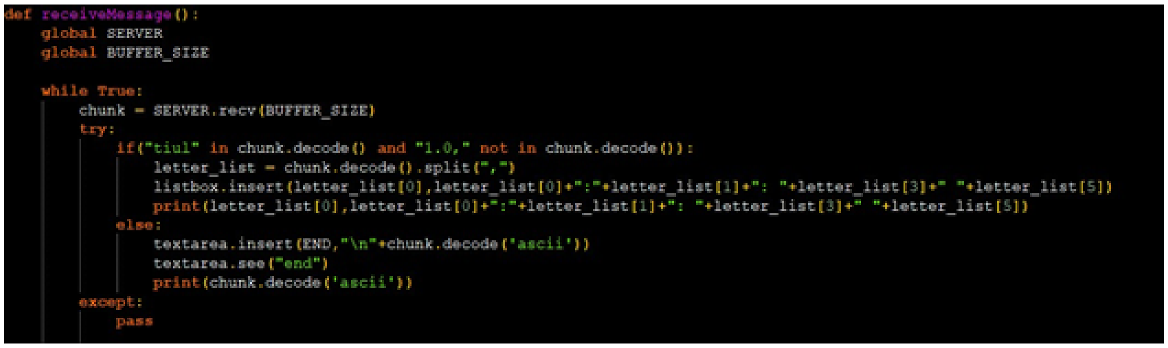
- Create a dictionary where it will store client name, address, connected with information, file name and file size. After getting all the information, display the message in the text area with the client name and address.

- Use threads on the server side so that whenever a client request comes, a separate thread can be assigned for handling each request. It will target the handleclient function and pass two arguments client and client name and use start() to start this thread process.



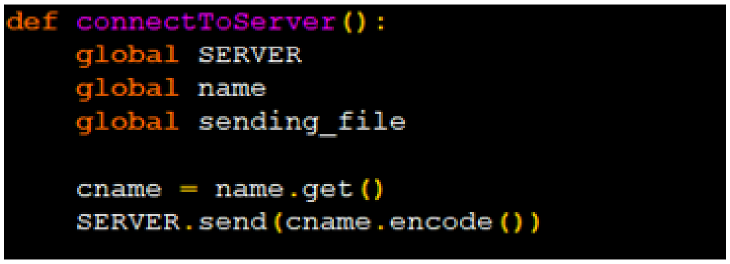
```
def acceptConnections():
    global SERVER
    global clients

    while True:
        client, addr = SERVER.accept()

        client_name = client.recv(4096).decode().lower()
        clients[client_name] = {
                "client"         : client,
                "address"        : addr,
                "connected_with" : "",
                "file_name"      : "",
                "file_size"      : 4096
            }

        print(f"Connection established with {client_name} : {addr}")

        thread = Thread(target = handleClient, args=(client,client_name,))
        thread.start()
```

4. Create a function **receiveMessage()** a client end function where the message received from a client or server is processed



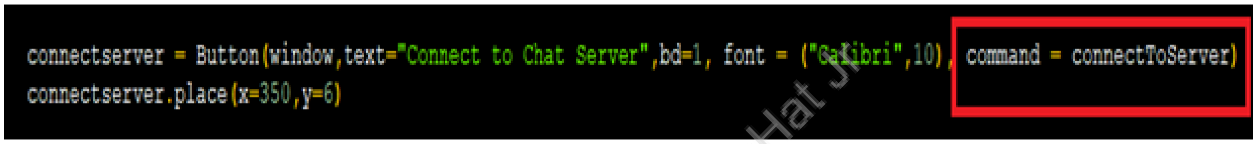
```
def receiveMessage():
    global SERVER
    global BUFFER_SIZE

    while True:
        chunk = SERVER.recv(BUFFER_SIZE)
        try:
            if("tiul" in chunk.decode() and "1.0," not in chunk.decode()):
                letter_list = chunk.decode().split(",")
                listbox.insert(letter_list[0],letter_list[0]+":"+letter_list[1]+": "+letter_list[3]+" "+letter_list[5])
                print(letter_list[0],letter_list[0]+":"+letter_list[1]+": "+letter_list[3]+" "+letter_list[5])
            else:
                textarea.insert(END,"\n"+chunk.decode('ascii'))
                textarea.see("end")
                print(chunk.decode('ascii'))
        except:
            pass
```

5. Make function **connectToServer()** at client side



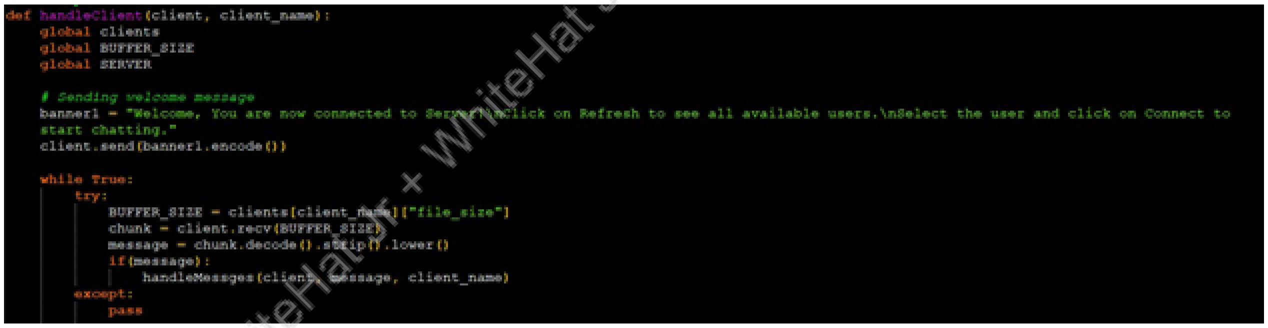
```
def connectToServer():
    global SERVER
    global name
    global sending_file

    cname = name.get()
    SERVER.send(cname.encode())
```

6. Add **connectToServer()** in the U-I Interface to which will act as an event.



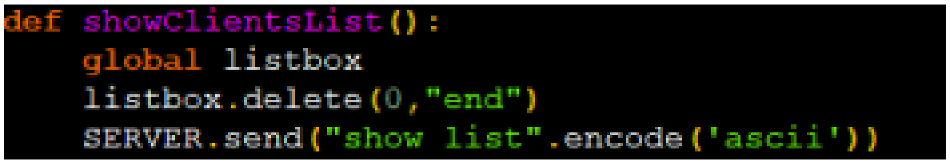
```
connectserver = Button(window,text="Connect to Chat Server",bd=1, font = ("Calibri",10), command = connectToServer)
connectserver.place(x=350,y=6)
```

7. Create a function **handleClient()** where we pass two parameters, client and client_name.



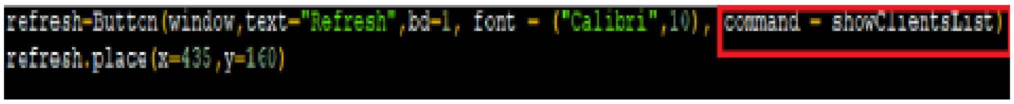
```
def handleClient(client, client_name):
    global clients
    global BUFFER_SIZE
    global SERVER

    # Sending welcome message
    banner1 = "Welcome, You are now connected to Server!\nClick on Refresh to see all available users.\nSelect the user and click on Connect to start chatting."
    client.send(banner1.encode())

    while True:
        try:
            BUFFER_SIZE = clients[client_name]["file_size"]
            chunk = client.recv(BUFFER_SIZE)
            message = chunk.decode().strip().lower()
            if(message):
                handleMessges(client, message, client_name)
        except:
            pass
```

8. Create a function **showClientList()**



```
def showClientsList():
    global listbox
    listbox.delete(0,"end")
    SERVER.send("show list".encode('ascii'))
```

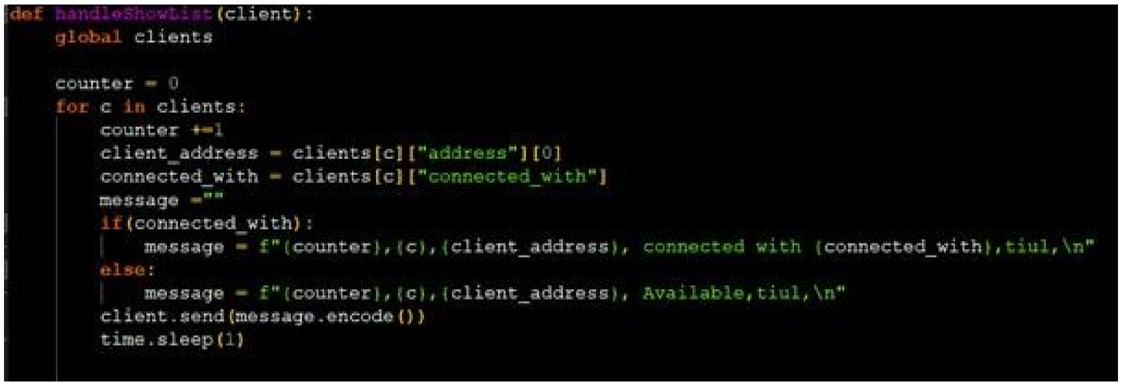
9. Call this function **showClientList()** at our user-interface side.



```
refresh=Button(window,text="Refresh",bd=1, font = ("Calibri",10), command = showClientsList)
refresh.place(x=435,y=160)
```

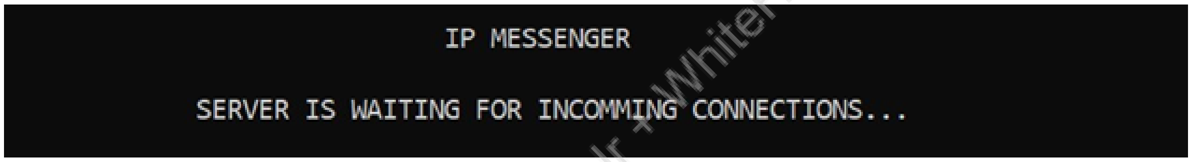
10. Create a function **handlesShowList()** a server-side function that retrieves a list of

clients whenever a client requests it.



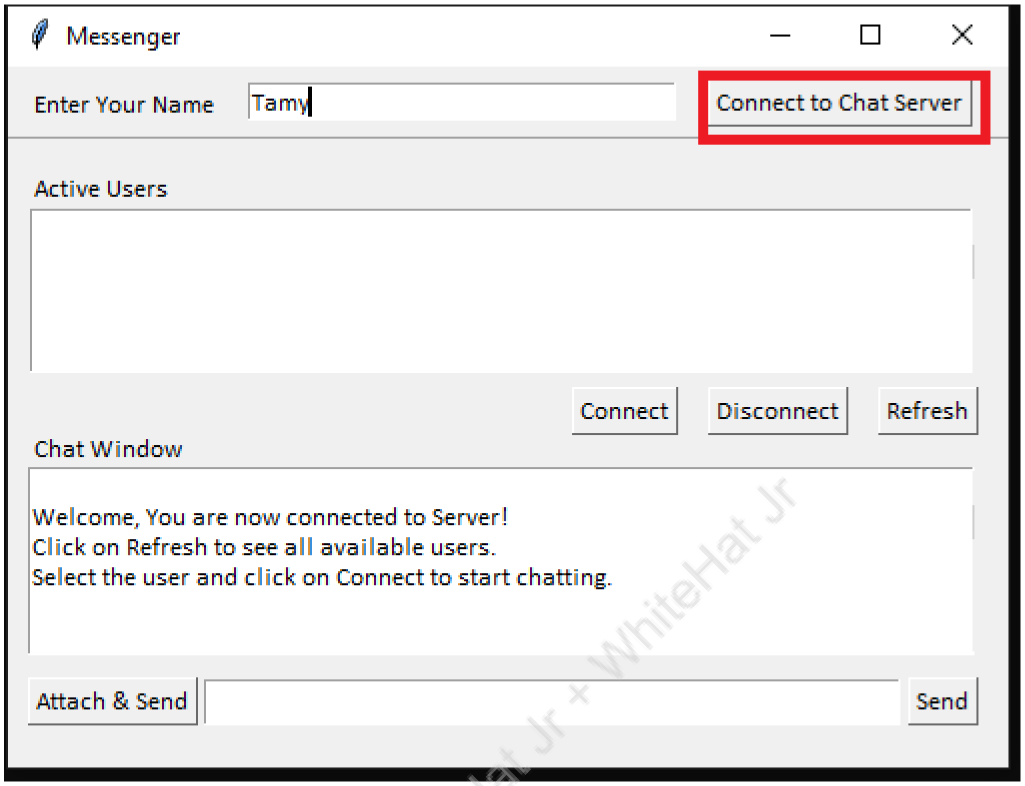
```
def handleShowList(client):
    global clients

    counter = 0
    for c in clients:
        counter +=1
        client_address = clients[c]["address"][0]
        connected_with = clients[c]["connected_with"]
        message =""
        if(connected_with):
            message = f"{counter},{c},{client_address}, connected with {connected_with},tiul,\n"
        else:
            message = f"{counter},{c},{client_address}, Available,tiul,\n"
        client.send(message.encode())
        time.sleep(1)
```

11. server.py in terminal/cmd looks like -



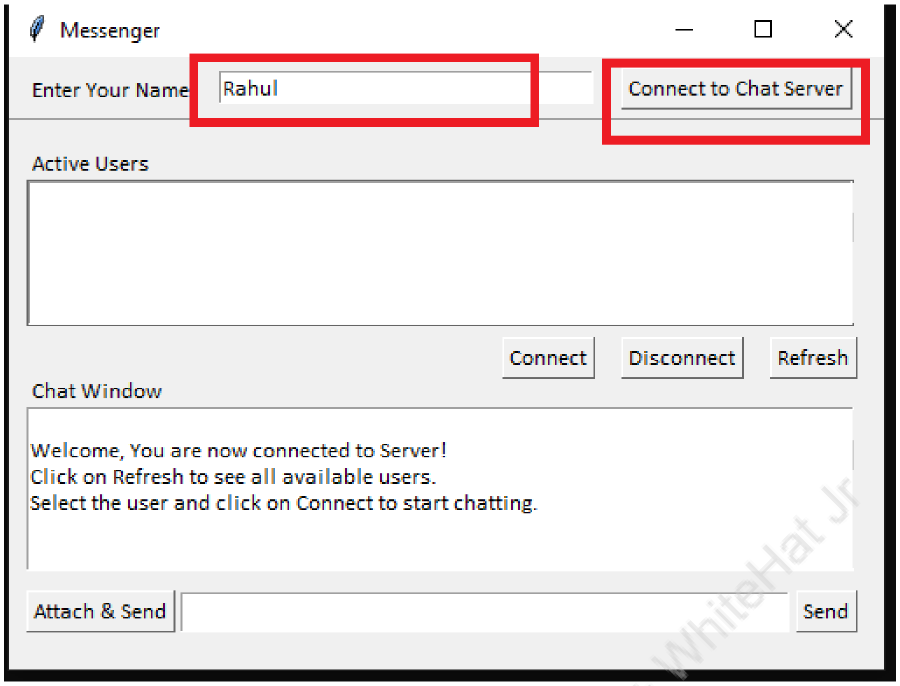
```
IP MESSENGER

SERVER IS WAITING FOR INCOMMING CONNECTIONS...
```

12. client.py in the terminal/cmd looks like -
    - Client Tamy



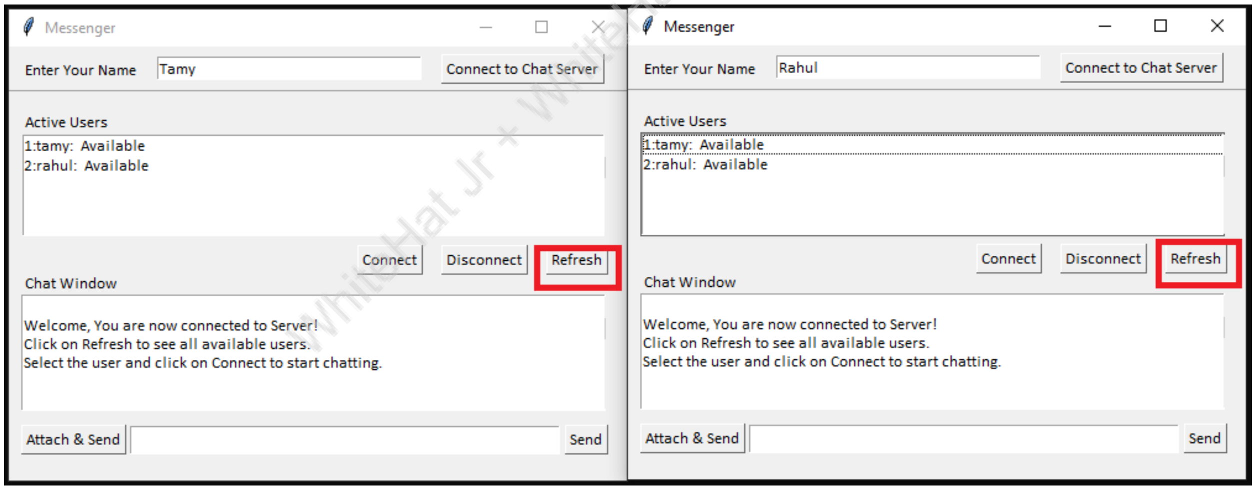
```
Welcome, You are now connected to Server!
Click on Refresh to see all available users.
Select the user and click on Connect to start chatting.
```

- Client Rahul



```
Welcome, You are now connected to Server!
Click on Refresh to see all available users.
Select the user and click on Connect to start chatting.
```

- Click on refresh button to see available users



```
1:tamy:  Available
2:rahul:  Available
```

```
Welcome, You are now connected to Server!
Click on Refresh to see all available users.
Select the user and click on Connect to start chatting.
```

```
1:tamy:  Available
2:rahul:  Available
```

```
Welcome, You are now connected to Server!
Click on Refresh to see all available users.
Select the user and click on Connect to start chatting.
```

We have completed the second part of the app!

What's NEXT?

In the next class we will _________

EXTEND YOUR KNOWLEDGE

You can learn more about messaging from https://en.wikipedia.org/wiki/Windows_Messenger_service.