

File Sharing App- 1



What is our GOAL for this MODULE?

In this class, we have applied our knowledge of FTP and we did the first part of the File sharing app. During the first part of the module, we discussed how socket client connections work and created GUIs for File Sharing Application. The goal of this module is to learn how to make connections and GUI for desktop File Sharing Application.

What did we ACHIEVE in the class TODAY?

- Socket Client Connection
- GUI for FTP

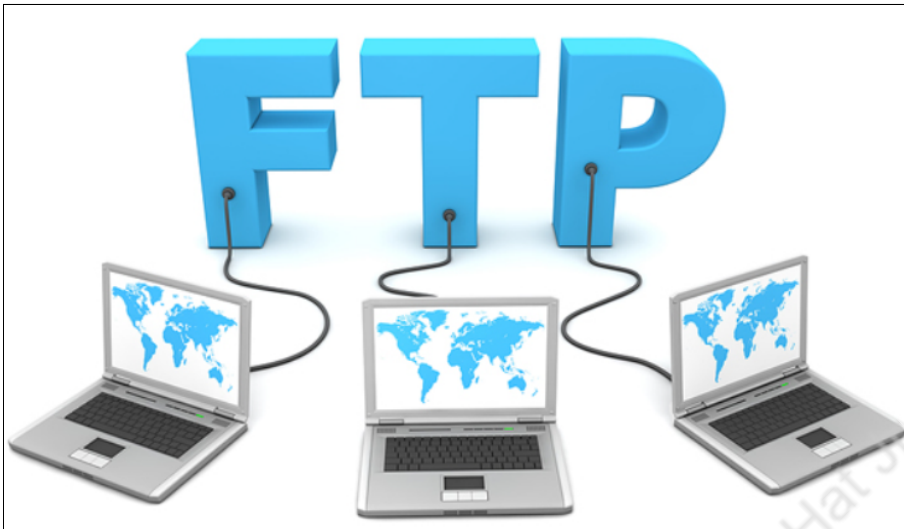
Which CONCEPTS/CODING BLOCKS did we cover today?

- We learned to understand about Socket & Client Connection
- We learned how to make GUI

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. Create a server using boilerplate code

- Import sockets library from Python into server.py to create sockets.

```
import socket
```

- Import thread library, for the server and client to communicate simultaneously.

```
from threading import Thread
```

- Declare some global variables such as SERVER, PORT, and IP_ADDRESS and set their values to None. Also declare an empty clients dictionary that contains the information about clients that have connected to the server.

```
import socket
from threading import Thread
IP_ADDRESS = '127.0.0.1'
PORT = 8080
SERVER = None
clients = {}
```

- Create a function called setup() to setup the server

```
def setup():
    print("\n\t\t\t\t\tIP MESSENGER\n")

    # Getting global values
    global PORT
    global IP_ADDRESS
    global SERVER

    SERVER = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    SERVER.bind((IP_ADDRESS, PORT))

    # Listening incoming connections
    SERVER.listen(100)

    print("\t\t\t\tSERVER IS WAITING FOR INCOMING CONNECTIONS...")
    print("\n")
```

- Create a function **acceptConnections()** and call the same in our server setup() function

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

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

```
def setup():
    print("\n\t\t\t\t\tIP MESSENGER\n")

    # Getting global values
    global PORT
    global IP_ADDRESS
    global SERVER

    SERVER = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    SERVER.bind((IP_ADDRESS, PORT))

    # Listening incoming connections
    SERVER.listen(100)

    print("\t\t\t\t\tSERVER IS WAITING FOR INCOMING CONNECTIONS...")
    print("\n")

    acceptConnections()

setup_thread = Thread(target=setup)          #receiving multiple messages
setup_thread.start()
```

3. Create a client.py file.

- imports socket & threading and declared variables

```
import socket
from threading import Thread
PORT = 8080
IP_ADDRESS = '127.0.0.1'
SERVER = None
```

4. Using the connect() function, we establish a connection between the server and client

```
def setup():
    global SERVER
    global PORT
    global IP_ADDRESS

    SERVER = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    SERVER.connect((IP_ADDRESS, PORT))
setup()
```

5. For GUI, import the Tkinter library on top of the code, along with Tkinter we will import other Tkinter libraries like the file dialog, to open a file, a directory, save as file, and more

```
from tkinter import *
from tkinter import ttk
from tkinter import filedialog
```

6. Declare all variables necessary for our user interface, name listbox, textarea, labelchat, text_message and will set their value to None.

```
name = None
listbox = None
textarea= None
labelchat = None
text_message = None
```

7. Create this GUI within a function openChatWindow()

```
def openChatWindow():
```

8. In order to make an FTP app , we need the following: Entry box, Button, Listbox, Listbox Scrollbar, Connect, Disconnect, test-area, text-area, scroller for test-area, Send button for message, Attach & send button.

9. Initialize the window with the **Tk()** method and assign it to a variable. A blank window is created with close, maximize, and minimize buttons on top, as a normal GUI should. Set the window title using title() . Let's name it "Messenger".

10. Set the geometrical configuration of the window using **geometry()**, which in our case is ("500 * 300").

```
window=Tk()
window.title('Messenger')
window.geometry("500x350")
```

11. Create Name label: :

Create a name label using the widget **Label()**. Call the main window first, then set the Label attributes, such as text and font, and then place it on the screen using the place() method.

```
namelabel = Label(window, text= "Enter Your Name", font = ("Calibri",10))
namelabel.place(x=10, y=8)
```

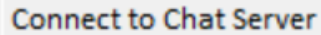
12. Create Text Input:

Create an input box using **Entry()** widget. Set the attributes for Entry, call the main window first and then set attributes for the Entry i.e. width, font and then place it on the screen using place() method.

Add focus using focus() to highlight the entry box.

```
name = Entry(window,width =30,font = ("Calibri",10))
name.place(x=120,y=8)
name.focus()
```

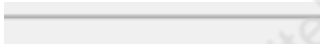
13. Create Connect to Chat Server Button:



Button() widget can be used to create a button. Set the Button attributes, call the main window and then set the Button attributes, such as the text and font, and then place it on the screen using place().

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

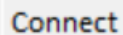
14. Create Separator:



Use the **Separator()** widget to create a Separator. Assign attributes to Separator. Call the main window first and then assign attributes to Separator, i.e. orient, and then place it on the screen by using place() method. The click event will be handled using Command.

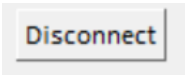
```
separator = ttk.Separator(window, orient='horizontal')
separator.place(x=0, y=35, relwidth=1, height=0.1)
```

15. Connect Button :



Create a button by using the **Button()** widget. Setting the button's attributes involves calling the main window first, setting the button's attributes such as text and font, and placing it on screen using the **place()** method.

```
connectButton=Button(window,text="Connect",bd=1, font = ("Calibri",10))
connectButton.place(x=282,y=160)
```



16. Disconnect Button:

Using the **Button()** widget, create a disconnect button. Call the main window first, and then set the attributes for Button - i.e. text, font, and then place it on the screen using the **place()** method. This event will be handled by Command later on.

```
disconnectButton=Button(window,text="Disconnect",bd=1, font = ("Calibri",10))
disconnectButton.place(x=350,y=160)
```



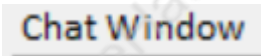
17. Refresh Button

Create a Refresh button using the **Button()** widget. Set the attributes for the button, first call the main window, and then set the text, font, and then place it on the screen using the **place()** method. Eventually, we'll use Command for the click event.

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

In order to make your U-I visible on screen, we must call **mainloop**.

```
window.mainloop()
```



14. Chat Label

Using the **Label()** widget, create a chat name label. Call the main window first and then set the Label's attributes, such as text and font, and then place it on the screen using the **place()** method.

```
labelchat = Label(window, text= "Chat Window", font = ("Calibri",10))
labelchat.place(x=10, y=180)
```

15. Use the **Text()** widget to create a Text Area. Setting the attributes for a Scroll Bar, calling the main window, setting the attributes for a Text, such as the size, height, and font, and then placing the text on the screen with the **place()** method.

```
textarea = Text(window, width = 67,height = 6,font = ("Calibri",10))
textarea.place(x=10,y=200)
```

16. ScrollBar

Use the **Scrollbar()** widget to create a scroll bar. Set the attributes for the Scrollbar, call the Text area this time since we want a scrollbar for the text area and then place it on the screen using place (). Position is determined by relative height and relative x. Using the config() command, configure the scrollbar. Since we need a vertical view, use the yview

```
scrollbar2 = Scrollbar(textarea)
scrollbar2.place(relheight = 1,relx = 1)
scrollbar2.config(command = listbox.yview)
```

17. Message input to write messages



Use the **Entry()** widget to create a message input box. You can set attributes for the Entry by calling the main window first, then setting width and font attributes, and then putting it on the screen by using the place() method. Use the pack() method to pack the box.

```
text_message = Entry(window, width =43, font = ("Calibri",12))
text_message.pack()
text_message.place(x=98,y=306)
```

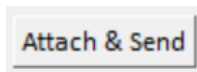
18. Send Button:



Create a send button using the Button() widget. Call the main window first, and then set the attributes for Button - i.e. text, font, and then place it on the screen using the place() method. We will later use Command for this click event.

```
send=Button(window,text="Send",bd=1, font = ("Calibri",10))
send.place(x=450,y=305)
```

19. Attach & Send Button:



To fetch files from the computer, we need an attach & send button. Create an attach button using the **Button()** widget. Call the main window first, and then set attributes for the button, such as text and font, before placing it on the screen using **place()**

method. We will later use Command for this click event.

```
attach=Button(window,text="Attach & Send",bd=1, font = ("Calibri",10))
attach.place(x=10,y=305)
```

20. File path label

This label will display the path of the fetched file. Create a file path label using **Label()** widget. Set the attributes for Label, call the main window first and then set attributes for the Label i.e. text, font and then place it on the screen using place() method.

```
filePathLabel = Label(window, text= "",fg= "blue", font = ("Calibri",8))
filePathLabel.place(x=10, y=330)
```

21. Call the **openChatWindow()** function inside the setup

```
def setup():
    global SERVER
    global PORT
    global IP_ADDRESS

    SERVER = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    SERVER.connect((IP_ADDRESS, PORT))

    openChatWindow()

setup()
```

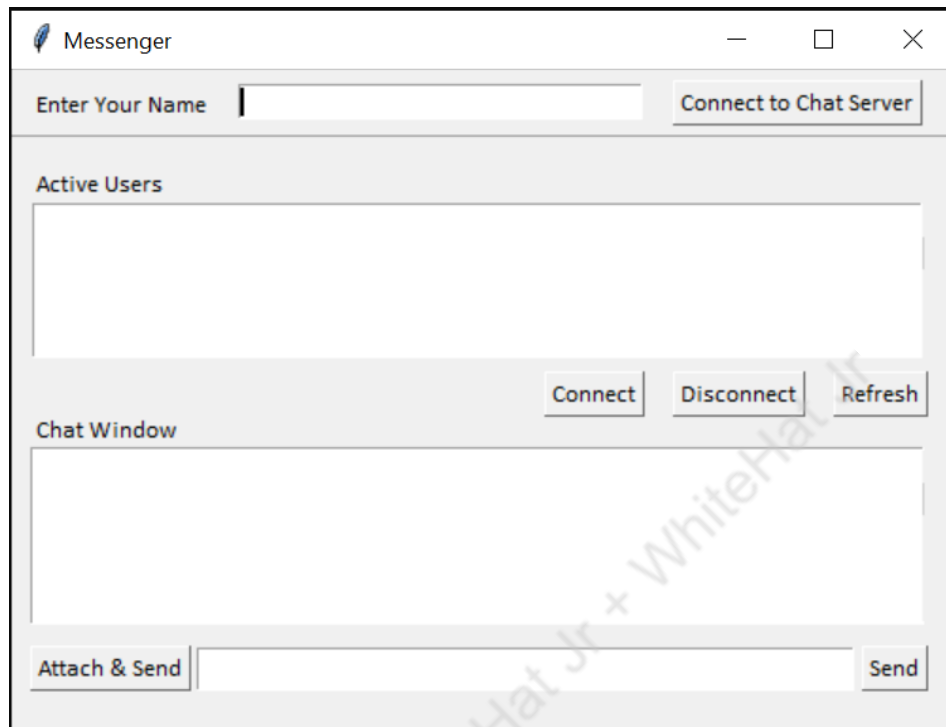
22. server.py in terminal/cmd looks like -

```
IP MESSENGER

SERVER IS WAITING FOR INCOMING CONNECTIONS...
```

23. client.py in the terminal/cmd looks like -

```
IP MESSENGER
```



We have completed the first part of the app!

What's NEXT?

In the next class we will now work on the button's functionality.

EXTEND YOUR KNOWLEDGE

You can learn more about messaging from

https://en.wikipedia.org/wiki/Windows_Messenger_service.