PRO-C211

File Sharing App - 4

What is our GOAL for this MODULE?

In this class, we have applied our knowledge of FTP and we did the third part of the File sharing app. During the third part of the module, we discussed how to make an FTP server. The goal of this module is to learn how to make FTP Server and how to browse files.

What did we ACHIEVE in the class TODAY?

- FTP Server
- Browse file on FTP

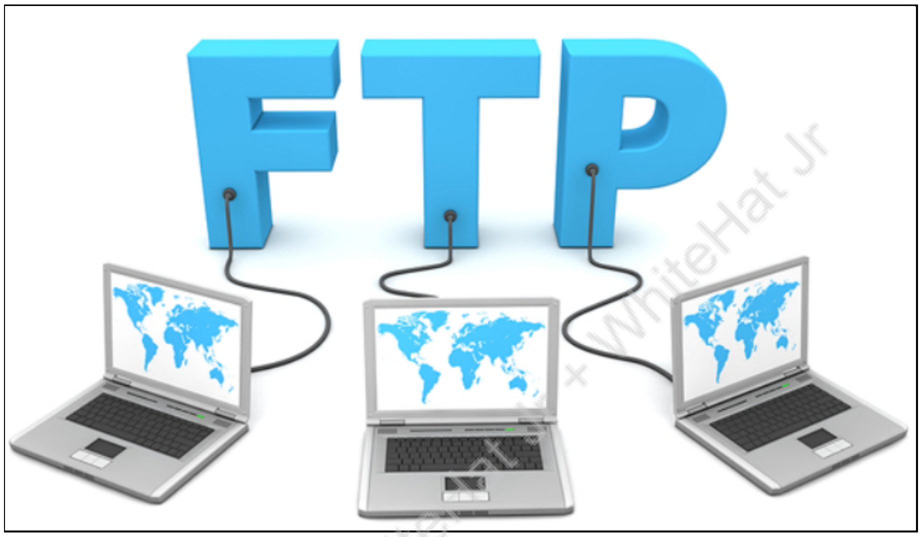
Which CONCEPTS/CODING BLOCKS did we cover today?

- We learned to how to make FTP Servers
- We learned how to browse files

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

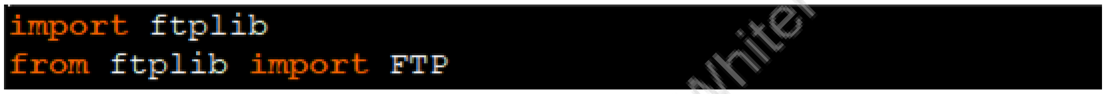


File transfer protocol (FTP) is a set of rules that computers follow for the transferring of files from one system to another over the internet. It may be used by a business to transfer files from one computer system to another, or websites may use FTP to upload or download files from a website's server.

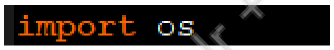
How did we DO the activities?

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

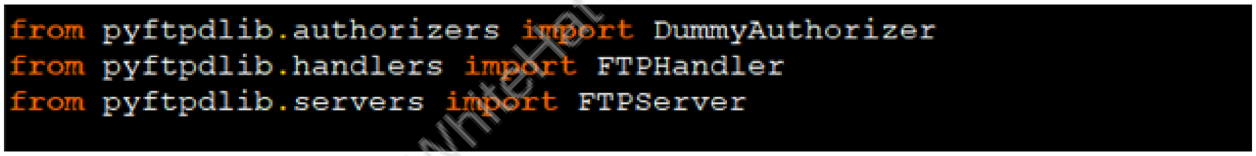
2. Import Python's ftplib module to upload and download the file, also import os .



```
import ftplib
from ftplib import FTP
```

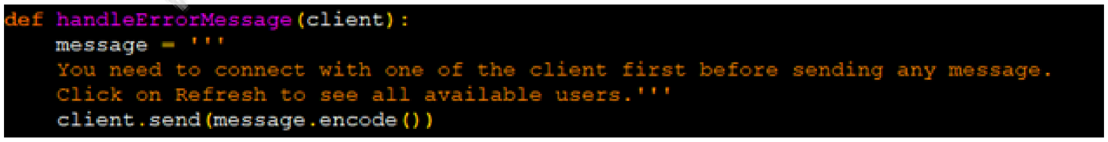


```
import os
```



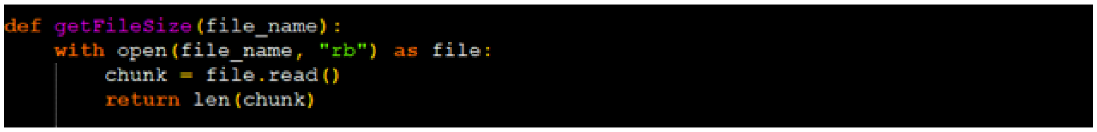
```
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer
```

3. Create function **handleErrorMessage()** pass the argument client also write a function getFileSize()
    - Create variable message which will store "error message", client.send will send this message in encoded form



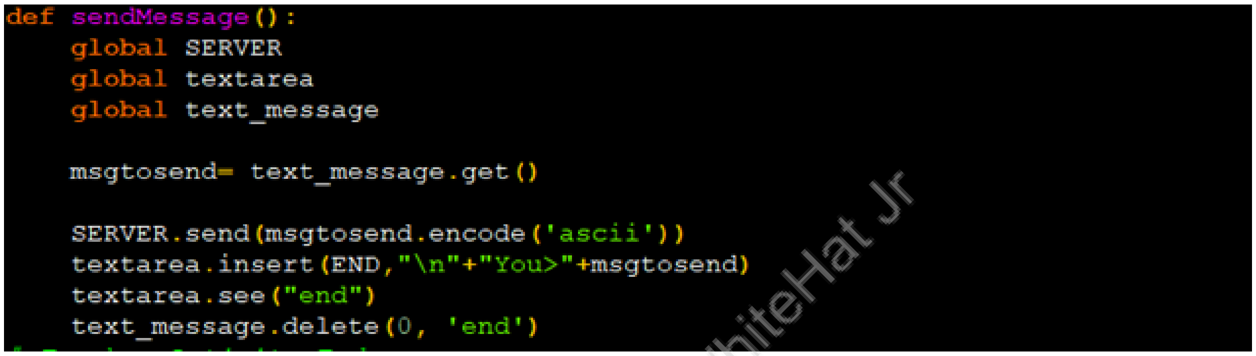
```
def handleErrorMessage(client):
    message = '''
    You need to connect with one of the client first before sending any message.
    Click on Refresh to see all available users.'''
    client.send(message.encode())
```

4. Create getFileSize() function

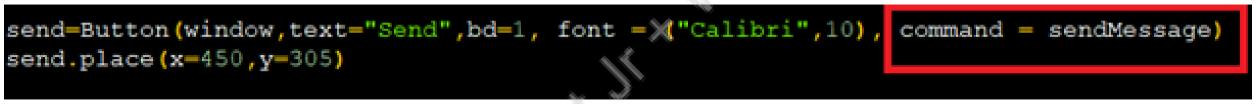


```
def getFileSize(file_name):
    with open(file_name, "rb") as file:
        chunk = file.read()
        return len(chunk)
```

5. Create a function ***sendMessage()*** and call it at the user- interface.

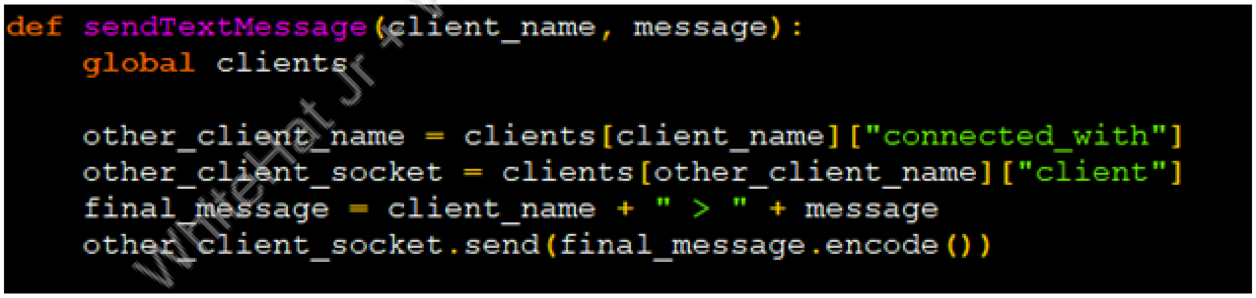


```
def sendMessage():
    global SERVER
    global textarea
    global text_message

    msgtosend= text_message.get()

    SERVER.send(msgtosend.encode('ascii'))
    textarea.insert(END,"\n"+"You>"+msgtosend)
    textarea.see("end")
    text_message.delete(0, 'end')
```



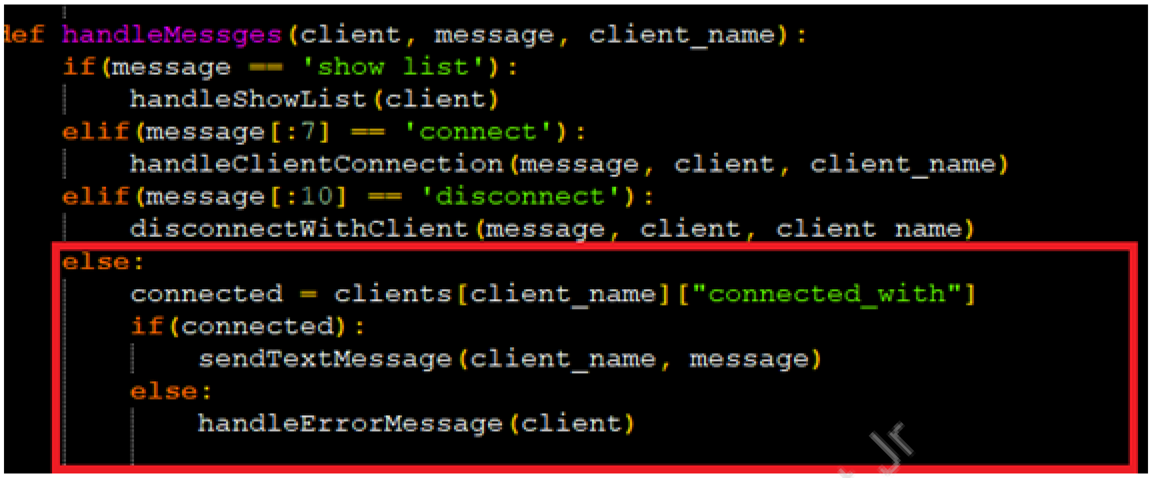
```
send=Button(window,text="Send",bd=1, font =("Calibri",10), command = sendMessage)
send.place(x=450,y=305)
```

6. Create function name ***sendTextMessage()*** where we pass two arguments client_name and message.



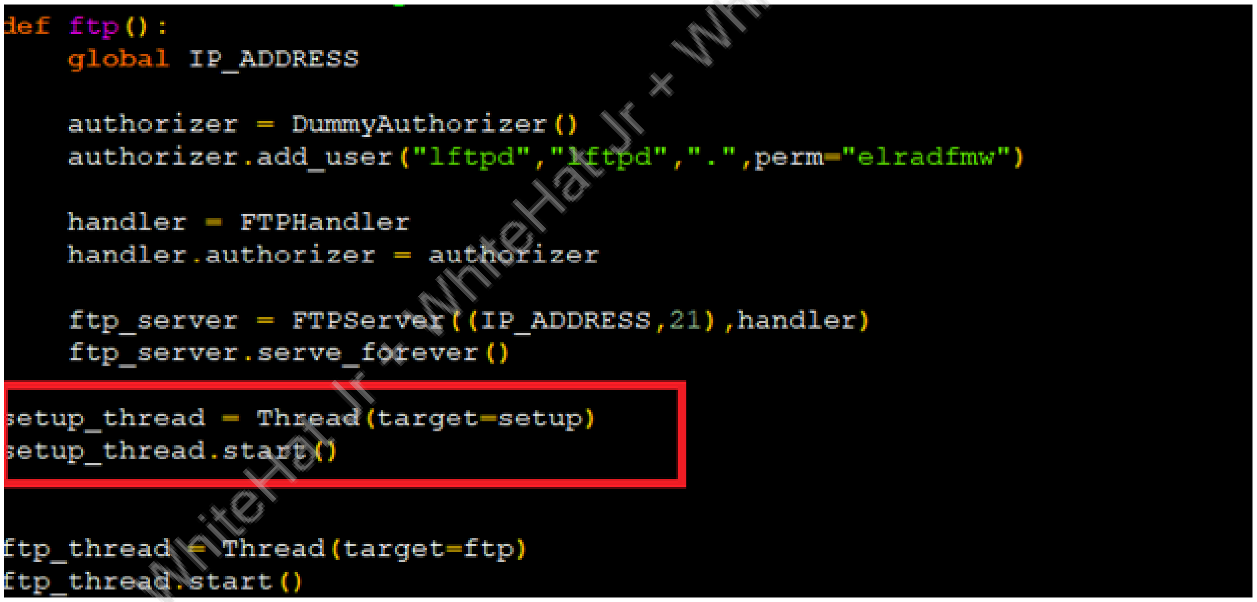
```
def sendTextMessage(client_name, message):
    global clients

    other_client_name = clients[client_name]["connected_with"]
    other_client_socket = clients[other_client_name]["client"]
    final_message = client_name + " > " + message
    other_client_socket.send(final_message.encode())
```

7. Make changes in ***handleMessages()*** function to send messages and also handle error



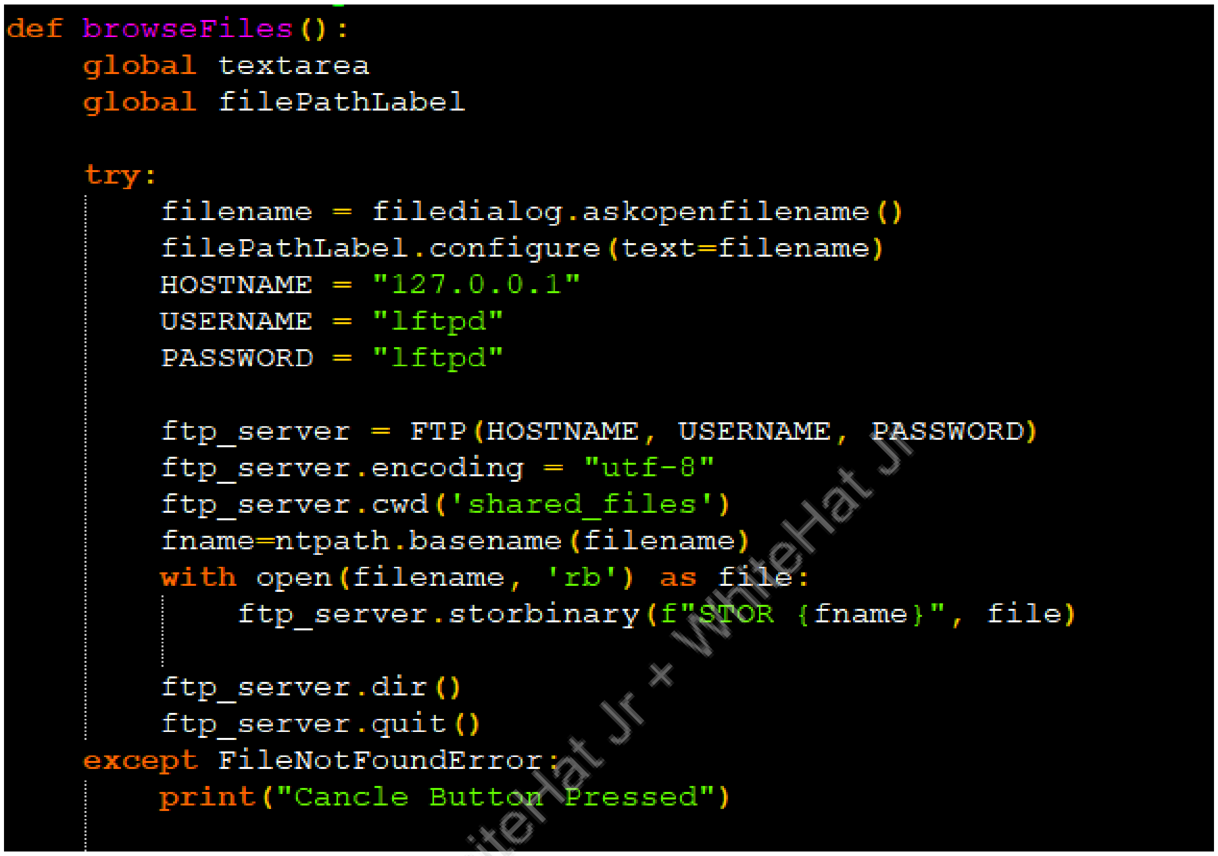
```
def handleMessges(client, message, client_name):
    if(message == 'show list'):
        handleShowList(client)
    elif(message[:7] == 'connect'):
        handleClientConnection(message, client, client_name)
    elif(message[:10] == 'disconnect'):
        disconnectWithClient(message, client, client name)
    else:
        connected = clients[client_name]["connected_with"]
        if(connected):
            sendTextMessage(client_name, message)
        else:
            handleErrorMessage(client)
```

8. Create a function **ftp()** to configure FTP Server

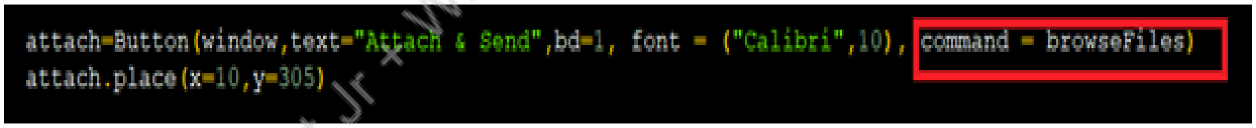


```
def ftp():
    global IP_ADDRESS

    authorizer = DummyAuthorizer()
    authorizer.add_user("lftpd","lftpd",".",perm="elradfmw")

    handler = FTPHandler
    handler.authorizer = authorizer

    ftp_server = FTPServer((IP_ADDRESS,21),handler)
    ftp_server.serve_forever()

setup_thread = Thread(target=setup)
setup_thread.start()

ftp_thread = Thread(target=ftp)
ftp_thread.start()
```

9. Create a function **browseFiles()** to browse the files in the server and call it at user-interface.

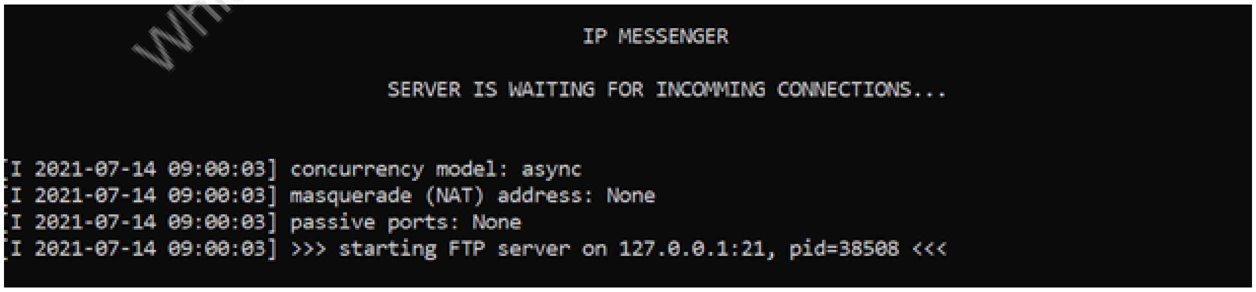


```
def browseFiles():
    global textarea
    global filePathLabel

    try:
        filename = filedialog.askopenfilename()
        filePathLabel.configure(text=filename)
        HOSTNAME = "127.0.0.1"
        USERNAME = "lftpd"
        PASSWORD = "lftpd"

        ftp_server = FTP(HOSTNAME, USERNAME, PASSWORD)
        ftp_server.encoding = "utf-8"
        ftp_server.cwd('shared_files')
        fname=ntpath.basename(filename)
        with open(filename, 'rb') as file:
            ftp_server.storbinary(f"STOR {fname}", file)

        ftp_server.dir()
        ftp_server.quit()
    except FileNotFoundError:
        print("Cancle Button Pressed")
```



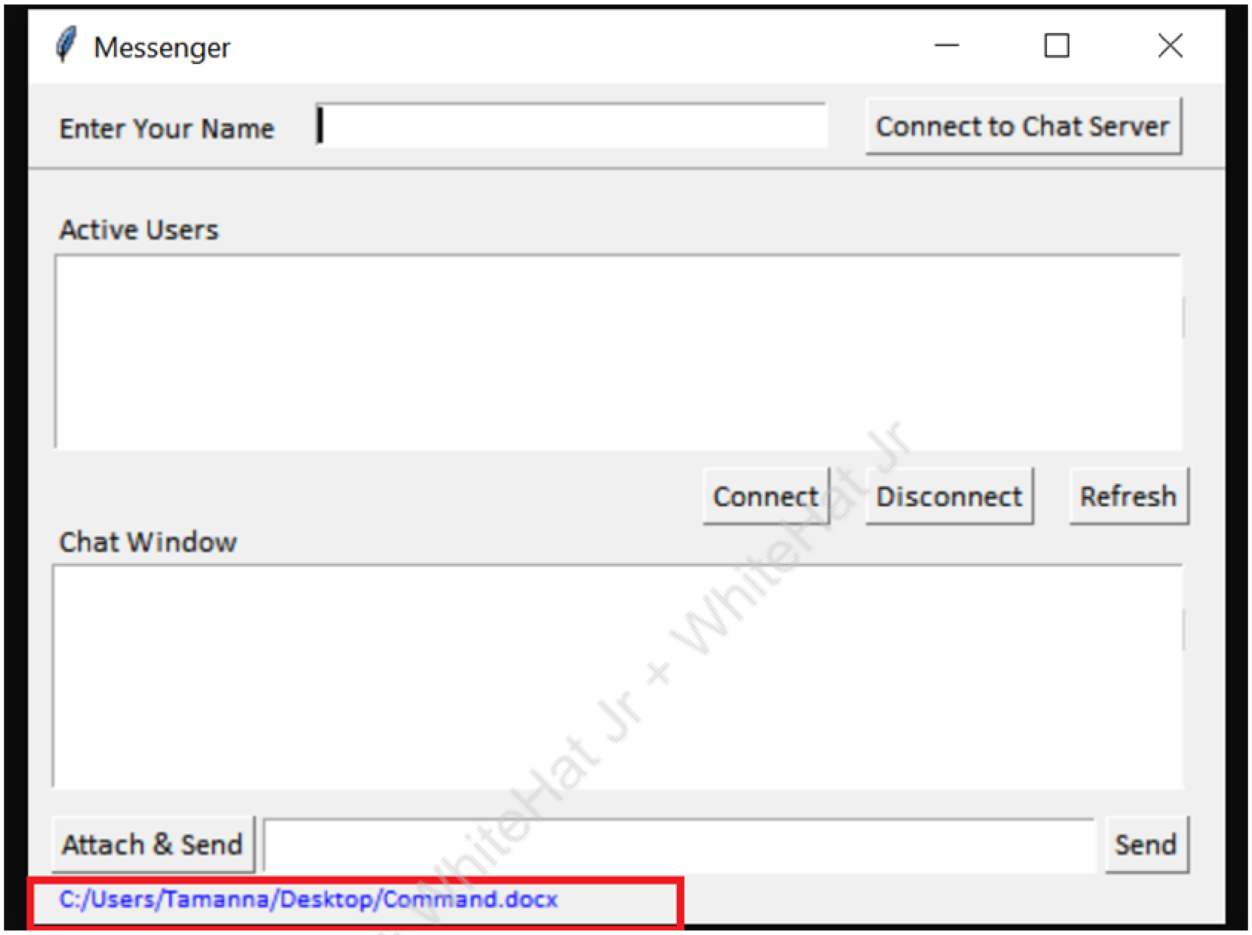
```
attach=Button(window,text="Attach & Send",bd=1, font = ("Calibri",10), command = browseFiles)
attach.place(x=10,y=305)
```

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



```
                                   IP MESSENGER

                    SERVER IS WAITING FOR INCOMMING CONNECTIONS...

[I 2021-07-14 09:00:03] concurrency model: async
[I 2021-07-14 09:00:03] masquerade (NAT) address: None
[I 2021-07-14 09:00:03] passive ports: None
[I 2021-07-14 09:00:03] >>> starting FTP server on 127.0.0.1:21, pid=38508 <<<
```

client.py in the terminal/cmd looks like -



We have completed the next part of the app, where we created buttons to send and receive files from the server using FTP protocol.

What's NEXT?

In the next class we will design the functionality for the same.

EXTEND YOUR KNOWLEDGE

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