DELHI TECHNOLOGICAL UNIVERSITY

COMPUTER NETWORKS (MC-308)

Midterm Innovative Project

SUBMITTED BY:

Aiman Siddiqua - 2K18/MC/008

Apoorva - 2K18/MC/019

SUBMITTED TO:

Sumedha Seniaray

File Transfer Application

TCP File Transfer Application in Python with GUI

INDEX

INTRODUCTION

TRANSMISSION CONTROL PROTOCOL

Transmission Control Protocol is a transport layer protocol that facilitates the transmission of packets from source to destination. It is a **connection-oriented** protocol that means it establishes the connection prior to the communication that occurs between the computing devices in a network. This protocol is used with an **IP protocol**, so together, they are referred to as a TCP/IP.

The main functionality of the TCP is to take the data from the application layer. Then it divides the data into several packets, provides numbering to these packets, and finally transmits these packets to the destination. The TCP, on the other side, will reassemble the packets and transmit them to the application layer.

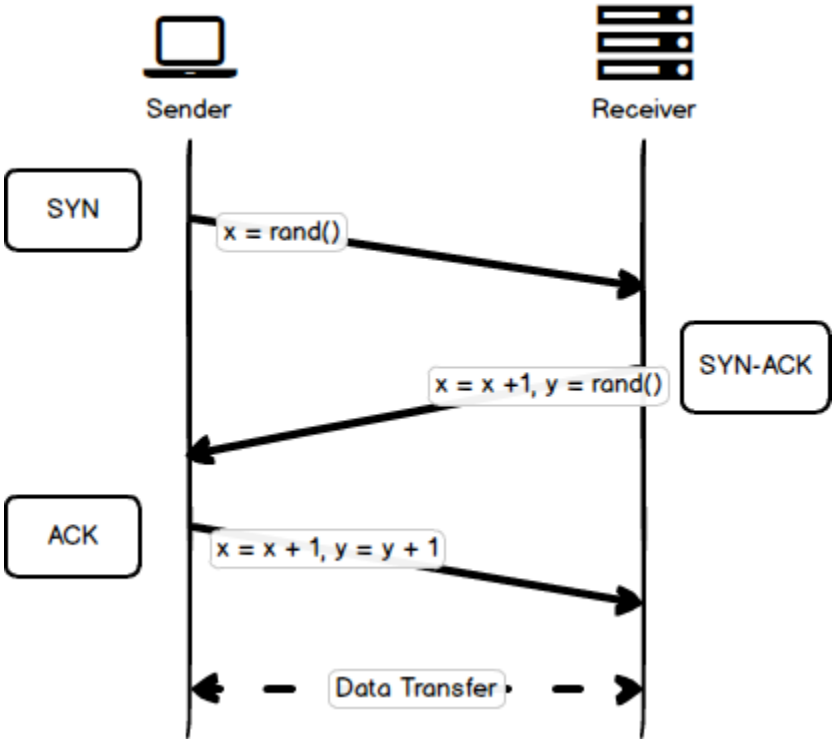
FEATURES OF TCP

- **Reliable:** TCP is a reliable protocol as it follows the flow and error control mechanism. It also supports the acknowledgment mechanism, which checks the state and sound arrival of the data.
- **Order of the data is maintained:** This protocol ensures that the data reaches the intended receiver in the same order in which it is sent.
- **Full duplex:** Data can transfer in both directions at the same time.
- **Connection-Oriented:** Data exchange occurs only after the connection establishment.
- **Stream-oriented:** TCP allows the sender to send the data in the form of a stream of bytes and also allows the receiver to accept the data in the form of a stream of bytes. TCP creates an environment in which both the sender and receiver are connected by an imaginary tube known as a virtual circuit.

WORKING OF TCP

In TCP, the connection is established by using **three-way handshaking**.

- ❏ The client sends the segment with its sequence number.
- ❏ The server, in return, sends its segment with its own sequence number as well as the acknowledgment sequence, which is one more than the client sequence number.
- ❏ When the client receives the acknowledgment of its segment, then it sends the acknowledgment to the server.



SOCKET PROGRAMMING

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

WHAT ARE SOCKETS?

Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The socket library(or various languages) provides specific classes for handling the common transports as well as a generic interface for handling the rest.

Sockets have their own vocabulary, these include:

Domain: The family of protocols that is used as the transport mechanism. These values are constants such as AF_INET, PF_INET, PF_UNIX, PF_X25, and so on.

Type: The type of communications between the two endpoints, typically SOCK_STREAM for connection-oriented protocols(TCP) and SOCK_DGRAM for connectionless protocols(UDP).

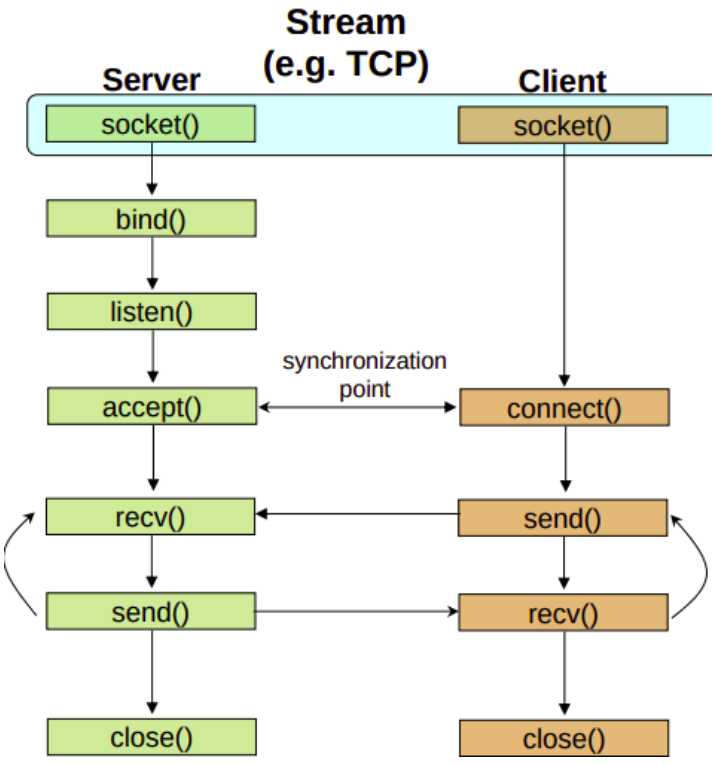
Protocol: Typically zero, this may be used to identify a variant of a protocol within a domain and type.

Hostname: The identifier of a network interface –

- A string, which can be a host name, a dotted-quad address, or an IPV6 address in colon (and possibly dot) notation
- A string "<broadcast>", which specifies an INADDR_BROADCAST address.
- A zero-length string, which specifies INADDR_ANY, or
- An Integer, interpreted as a binary address in host byte order.

Port : Each server listens for clients calling on one or more ports. A port may be a Fixnum port number, a string containing a port number, or the name of a service.

Socket Programming between a Server and a Client follows a specific order. This order is illustrated by:



IMPLEMENTATION

We implemented a simple file transfer application in Python via Socket Programming. The project was implemented in Linux.

The steps undertaken during the timeline of project included:

1. Learning the basics of Socket Programming.
2. Implementing a basic Server-Client connection via Sockets.
3. Outlining a simple process for the File Transfer Server to follow.
4. Creating a GUI in Python using tkinter module.
5. Implementing Socket Programming in the project.
6. Verifying Output.

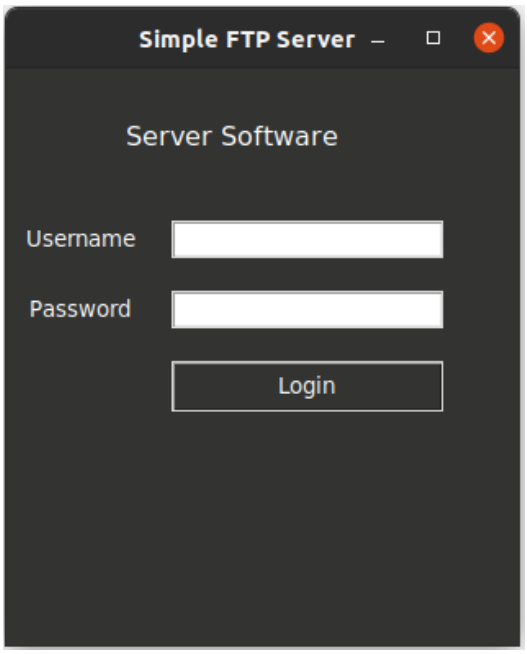
The File Transfer server completes a simple objective.

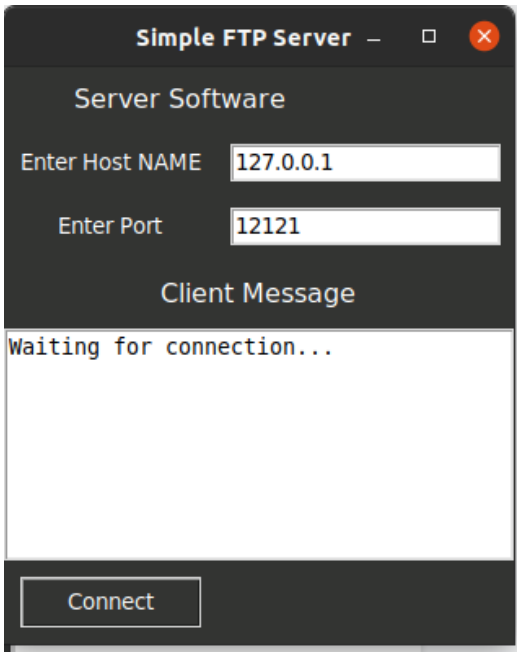
- We login to the server and create an open socket. The server then waits for any client to connect.
- The client logs in and connects to the server host and port.
- Client sends confirmation for the file to be sent.
- Server sends a .png to the client's local directory.
- Client confirms receipt of the file
- The connection is then closed on both Server and client side.
- The server can handle multiple clients during one session.

OUTPUTS

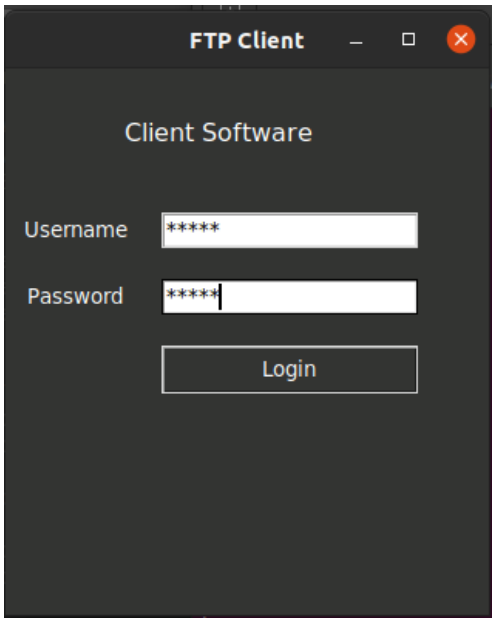
Following screenshots show the working of the project.

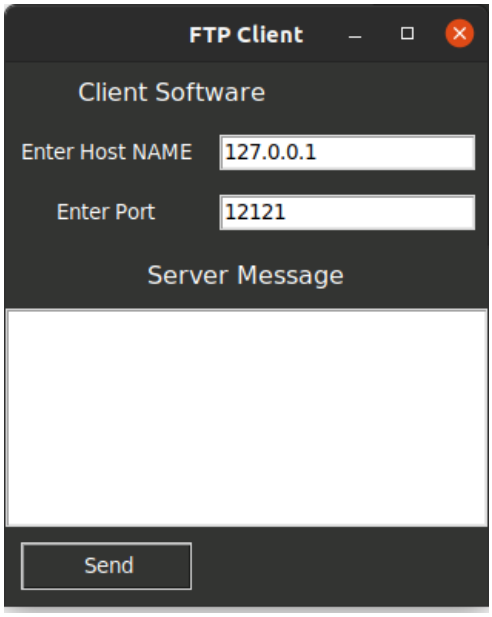
Server Login Screen



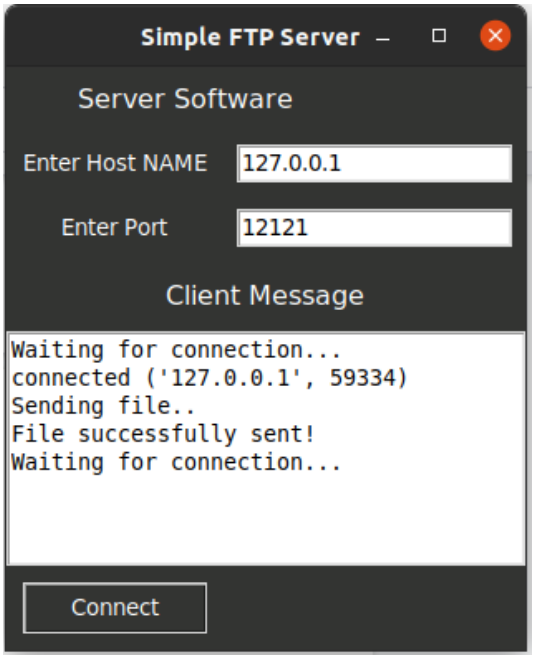


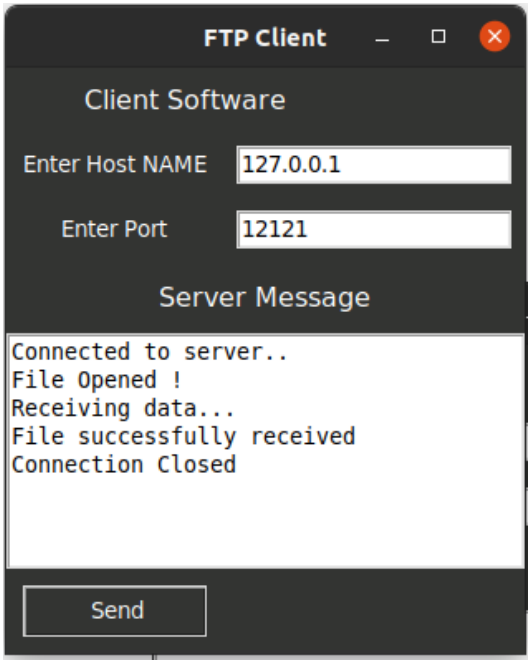
Client Login Screen



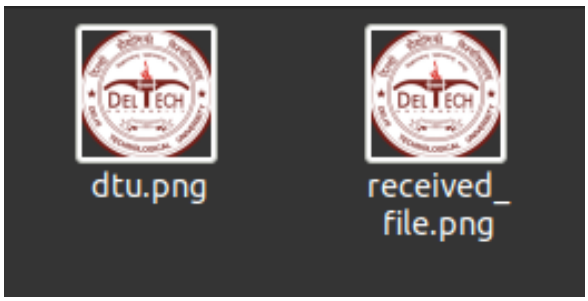


Final output screens





File Sent and File received



REFERENCES

- https://www.javatpoint.com/tcp
- https://www.geeksforgeeks.org/socket-programming-cc/#:~:text=Socket%20programming%20is%20a%20way,other%20to%20form%20a%20connection.
- https://www.tutorialspoint.com/unix_sockets/client_server_model.htm
- https://www.tutorialspoint.com/python/python_networking.htm
- https://www.geeksforgeeks.org/python-program-that-sends-and-recieves-message-from-client/
- https://www.tutorialspoint.com/python3/python_gui_programming.htm
- https://www.geeksforgeeks.org/python-gui-tkinter/
- https://medium.com/vaidikkapoor/understanding-non-blocking-i-o-with-python-part-1-ec31a2e2db9b
- https://www.geeksforgeeks.org/socket-programming-multi-threading-python/

APPENDIX

Server.py

```
import tkinter as tk
from tkinter import messagebox
from socket import *
import _thread

mycolor= '#333432'
def my_server(show_1,HOST,PORT):
    BUFSIZE = 1024
    ADDR = (HOST, PORT)

    tcpTimeSrvrSock = socket(AF_INET,SOCK_STREAM)
    tcpTimeSrvrSock.bind(ADDR)
    tcpTimeSrvrSock.listen(5)


    while True:
        show_1.insert(tk.END,"Waiting for connection...")
        show_1.insert(tk.END,"\n")

        tcpTimeClientSock, addr = tcpTimeSrvrSock.accept()

        show_1.insert(tk.END,"connected {}".format(addr))
        show_1.insert(tk.END,"\n")

        filename='dtu.png'
        f = open(filename,'rb')
        l = f.read(1024)
        show_1.insert(tk.END,"Sending file..")
        show_1.insert(tk.END,"\n")

        while (l):
            tcpTimeClientSock.send(l)
            print('Sent ',repr(l))
            l = f.read(1024)

        f.close()
        show_1.insert(tk.END,'File successfully sent!')
        show_1.insert(tk.END,"\n")
        tcpTimeClientSock.close()

class Page(tk.Tk):
```

```
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand = True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        self.frames = {}

        for F in (LoginPage, PageOne):
            frame = F(container, self)
            self.frames[F] = frame
            frame.grid(row=0, column=0, sticky="nsew")
            frame.configure(bg = mycolor)
        self.show_frame(LoginPage)

    def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()


class LoginPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        l_title=tk.Label(self, text="Server Software",font="Ubuntu,12",bg = mycolor,
fg = 'white')
        l_title.grid(row=0,column=0,columnspan=3, sticky="NSEW",padx=30,pady=30)

        label_username = tk.Label(self, text="Username",bg = mycolor, fg = 'white')
        label_password = tk.Label(self, text="Password",bg = mycolor, fg = 'white')

        entry_username = tk.Entry(self,show="*")

        entry_password = tk.Entry(self, show="*")

        label_username.grid(row=2, column=0,sticky='NSEW',padx=10,pady=10)
        label_password.grid(row=3, column=0,sticky='NSEW',padx=10,pady=10)
        entry_username.grid(row=2, column=1,sticky='NSEW',padx=10,pady=10)
        entry_password.grid(row=3, column=1,sticky='NSEW',padx=10,pady=10)

        logbtn = tk.Button(self, text="Login", bg = mycolor,
fg="White",command=lambda: login_btn_clicked())
        logbtn.grid(row=5, column=1,sticky='NSEW', padx=10, pady=10)

        def login_btn_clicked():
            # print("Clicked")
            username = entry_username.get()
```

```
            password = entry_password.get()

            if len(username) and len(password) > 2:
                # print(username, password)

                if username == "admin" and password == "admin":
                    controller.show_frame(PageOne)
                # display a ,essage if username and password is incorrect!
                else:
                    messagebox.showerror("An Error has occurred","Invalid username or
password ! ")

            else:
                messagebox.showerror("An Error has occurred","Enter Username and
Password")



class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        flag = True

        label = tk.Label(self, text="Server Software ", font="Ubuntu,16",bg=
mycolor,fg="White")
        label.grid(row=0, column=0, columnspan=2, padx=8, pady=8, sticky="NSNESWSE")

        l_host=tk.Label(self,text="Enter Host NAME", bg= mycolor,fg="White")
        l_host.grid(row=1, column=0, padx=8, pady=8, sticky="NSNESWSE")

        e_host=tk.Entry(self)
        e_host.grid(row=1, column=1, columnspan=2, padx=8, pady=8, sticky="NSNESWSE")
        e_host.insert(tk.END,'127.0.0.1')


        l_port=tk.Label(self,text="Enter Port", bg= mycolor,fg="White")
        l_port.grid(row=2, column=0, padx=8, pady=8, sticky="NSNESWSE")

        e_port=tk.Entry(self)
        e_port.grid(row=2, column=1, columnspan=2, padx=8, pady=8, sticky="NSNESWSE")
        e_port.insert(tk.END,12121)

        message_label=tk.Label(self,text="Client Message",font=("Ubuntu,12"),bg=
mycolor,fg="White")
        message_label.grid(row=3,column=0,columnspan=3,padx=10,pady=10,sticky="NSEW")
```

```
        #scrollbar_y = tk.Scrollbar(self)
        #scrollbar_y.grid(row=4, column=3,rowspan=6)
        #yscrollcommand=scrollbar_y.set
        show_1=tk.Text(self,height=8, width=35, bg='white',fg="black")
        show_1.grid(row=4, column=0,rowspan=3,columnspan=3,sticky="NSEW")

        b_connect=tk.Button(self,text="Connect",command=lambda: connect(),bg=
mycolor,fg="White")
        b_connect.grid(row=14,column=0,padx=10,pady=10,sticky="nsew")


        def runner():
            global after_id
            global secs
            secs += 1
            if secs % 2 == 0:  # every other second
                e_host_v=e_host.get()
                e_port_v=int(e_port.get())


        def connect():
            # CONNECT COM PORT
            e_host_v=e_host.get()
            e_port_v=int(e_port.get())
            _thread.start_new_thread(my_server,(show_1,e_host_v,e_port_v))
            global secs
            secs = 0


app = Page()
app.title("Simple FTP Server")
app.mainloop()
```

---

client.py

```
import tkinter as tk
from tkinter import messagebox
import socket
import sys
from tkinter import PhotoImage
from tkinter import PhotoImage,BitmapImage

mycolor= '#333432'

class Page(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand = True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        self.frames = {}

        for F in (LoginPage, PageOne):
            frame = F(container, self)
            self.frames[F] = frame
            frame.grid(row=0, column=0, sticky="nsew")
            frame.configure(bg = mycolor)
        self.show_frame(LoginPage)

    def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()

class LoginPage(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        l_title=tk.Label(self, text="Client Software",font="Ubuntu,12",bg = mycolor,
fg = 'white')
        l_title.grid(row=0,column=0,columnspan=3, sticky="NSEW",padx=30,pady=30)
        label_username = tk.Label(self, text="Username",bg = mycolor, fg = 'white')
        label_password = tk.Label(self, text="Password",bg = mycolor, fg = 'white')
        entry_username = tk.Entry(self,show="*")
        entry_password = tk.Entry(self, show="*")
        label_username.grid(row=2, column=0,sticky='NSEW',padx=10,pady=10)
        label_password.grid(row=3, column=0,sticky='NSEW',padx=10,pady=10)
        entry_username.grid(row=2, column=1,sticky='NSEW',padx=10,pady=10)
        entry_password.grid(row=3, column=1,sticky='NSEW',padx=10,pady=10)
```

```
        logbtn = tk.Button(self, text="Login", bg = mycolor,
fg="White",command=lambda: login_btn_clicked())
        logbtn.grid(row=5, column=1,sticky='NSEW', padx=10, pady=10)
        def login_btn_clicked():
            # print("Clicked")
            username = entry_username.get()
            password = entry_password.get()

            if len(username) and len(password) > 2:
                # print(username, password)

                if username == "admin" and password == "admin":
                    controller.show_frame(PageOne)
                # display a ,essage if username and password is incorrect!
                else:
                    messagebox.showerror("An Error has occurred","Invalid username or
password ! ")
            else:
                messagebox.showerror("An Error has occurred","Enter Username and
Password")

class PageOne(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        label = tk.Label(self, text="Client Software ", font="Ubuntu,16",bg=
mycolor,fg="White")
        label.grid(row=0, column=0, columnspan=2, padx=8, pady=8, sticky="NSNESWSE")

        l_host=tk.Label(self,text="Enter Host NAME",bg= mycolor,fg="White")
        l_host.grid(row=1, column=0, padx=8, pady=8, sticky="NSNESWSE")
        self.e_host=tk.Entry(self)
        self.e_host.grid(row=1, column=1, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")
        self.e_host.insert(tk.END,'127.0.0.1')
        l_port=tk.Label(self,text="Enter Port",bg= mycolor,fg="White")
        l_port.grid(row=2, column=0, padx=8, pady=8, sticky="NSNESWSE")
        self.e_port=tk.Entry(self)
        self.e_port.grid(row=2, column=1, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")
        self.e_port.insert(tk.END,12121)
        message_label=tk.Label(self,text="Server Message",font=("Ubuntu,12"),bg=
mycolor,fg="White")
        message_label.grid(row=3,column=0,columnspan=3,padx=10,pady=10,sticky="NSEW")
        self.show_1=tk.Text(self,height=8, width=35,bg='white',fg="black")
        self.show_1.grid(row=4, column=0,rowspan=3,columnspan=3,sticky="NSEW")
```

```
        b_connect=tk.Button(self,text=" Send",command=lambda: self.my_server(),bg=
mycolor,fg="White")
        b_connect.grid(row=14,column=0,padx=10,pady=10,sticky="nsew")

    def my_server(self):
        e_host_v=self.e_host.get()
        e_port_v=int(self.e_port.get())

        HOST, PORT = e_host_v, e_port_v
            # Create a socket (SOCK_STREAM means a TCP socket)
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                # Connect to server and send data
            s.connect((HOST, PORT))
            #s.sendall(bytes(data + "\n", "utf-8"))
            self.show_1.insert(tk.END, 'Connected to server..\n')

            with open('received_file.png','wb') as f:
                self.show_1.insert(tk.END,'File Opened !')
                self.show_1.insert(tk.END,'\n')

                data = s.recv(1024)
                self.show_1.insert(tk.END,'Receiving data...')
                self.show_1.insert(tk.END,'\n')
                while data:

                    print("data recv")
                    f.write(data)
                    data = s.recv(1024)
            f.close()
            self.show_1.insert(tk.END, 'File successfully received')
            self.show_1.insert(tk.END,'\n')

        s.close()
        self.show_1.insert(tk.END, 'Connection Closed')
        self.show_1.insert(tk.END,'\n')

app = Page()
app.title("FTP Client")
app.mainloop()
```
