Computer Networks Package

Simulation of Telnet Protocol

Team Members:

1. **S Subhikshaa (19pd36)**
2. **Shania Job (19pd32)**

Both of us contributed equally to the project and completed the project by staying connected through gmeet.

Abstract:

In this project, we tried simulating the Telnet protocol using sockets to establish a connection-oriented communication between the telnet server and the telnet client. This project was implemented in Python.
Modules used: Socket, MySQL connector
Operating system: Windows

Description:

TELNET is an abbreviation Telecommunication Network and is simply a connection protocol that allows a user to connect to a remote server that is listening for commands. Once the connection is established, the user can then issue commands to the server computer and examine the responses that are sent back. In general, TELNET uses 23 port for its command operations.

Uses of TELNET:-

Telnet is a type of client-server protocol that can be used to open a command line on a remote computer, typically a server. Users can utilize this tool to ping a port and find out whether it is open.

Telnet can be used for a variety of activities on a server, including editing files, running various programs, and checking email. Some servers enable remote connections using Telnet to access public data to play simple games or look up weather reports. Many of these features exist for nostalgic fun or because they still have compatibility with older systems that need access to specific data. Users are also able to connect to any software that utilizes text-based, unencrypted protocols via Telnet, from web servers to ports. Users can open a command prompt on the remote machine, type the word telnet and the remote machine's name or IP address, and the telnet connection will ping the port to see if it is open or not.

OUTPUT:

In this section, we will be seeing the different commands implemented by us as a part of this project along with their output on both the client and server-side.

Server

```
In [1]: runfile('D:/subhikshaa/sem5/Computer Networks/LAB/Package/Telnet_Protocol/Server/
telnet_server.py', wdir='D:/subhikshaa/sem5/Computer Networks/LAB/Package/Telnet_Protocol/Server')
Connected to mysql database.......
TELNET SERVER STARTING...
LISTENING...
**Connected to ('192.168.56.1', 52903)**
**********************************************************
```

Client

```
In [1]: runfile('D:/subhikshaa/sem5/Computer Networks/LAB/Package/Telnet_Protocol/Client/
telnet_client.py', wdir='D:/subhikshaa/sem5/Computer Networks/LAB/Package/Telnet_Protocol/Client')
*********************************************
Message from server :-
Welcome from Server
*********************************************

Type help to see available commands :)


telnet>
```

1. **help** command:- this command lets the client know the available commands that can be executed in the telnet server.

   Server

```
**********************************************************
Command : help
history table updated
**********************************************************
```

Client

```
telnet> help
**********************************************
Message from server :-
Available commands:-
upload <filename> : Upload the file
download <filename> : download the file
exec <command>:execute command and send output
send <text> : send plain text message
encrypt <text> : send encrypted text message
GET/HEAD <host>: send request to web server
scan <host> <start port> <end port>:scan ports start_port to end_port
history : show the history

**********************************************
```

2. **send** command:- this command sends a message to the server.

Server

```
**********************************************************
Command : send
Message from client :  Hello from client to server
history table updated
**********************************************************
```

Client

```
telnet> send Hello from client to server
**********************************************
Message from server :-
Message recieved
**********************************************
```

3. **encrypt** command:- this command sends an encrypted message to the server which is decrypted at the server-side.

Server

```
**********************************************************
Command : encrypt
Encrypted Message from client :  Pmttw12346^amzDmz_<[+
Decrypted Message from client :  Hello12346 Server*%$#
history table updated
**********************************************************
```

Client

```
telnet> encrypt Hello12346 Server*%$#
**********************************************
Message from server :-
Message recieved
**********************************************
```

4. **exec** command:- executes the command specified by the client on the server-side and returns the output to the client.

Server

```
**********************************************************
Command : exec
Command executed and sent output to client successfully
history table updated
**********************************************************
```

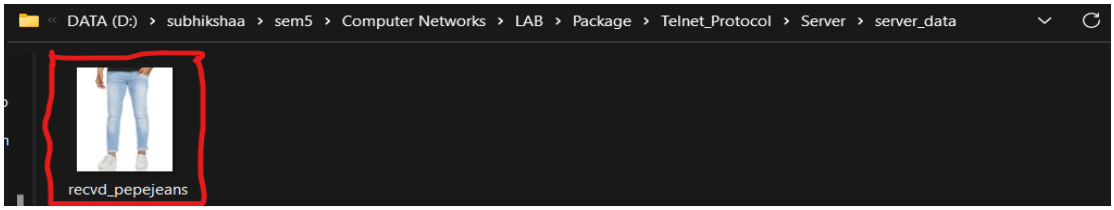
Client

```
telnet> exec ping www.google.com
**********************************************
Message from server :-
Command executed
Pinging www.google.com [2404:6800:4007:824::2004] with 32 bytes of data:
Reply from 2404:6800:4007:824::2004: time=26ms
Reply from 2404:6800:4007:824::2004: time=25ms
Reply from 2404:6800:4007:824::2004: time=28ms
Reply from 2404:6800:4007:824::2004: time=35ms

Ping statistics for 2404:6800:4007:824::2004:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 25ms, Maximum = 35ms, Average = 28ms

**********************************************
```

5. **upload** command:- this command uploads a file provided by the client present in client-side into the server.

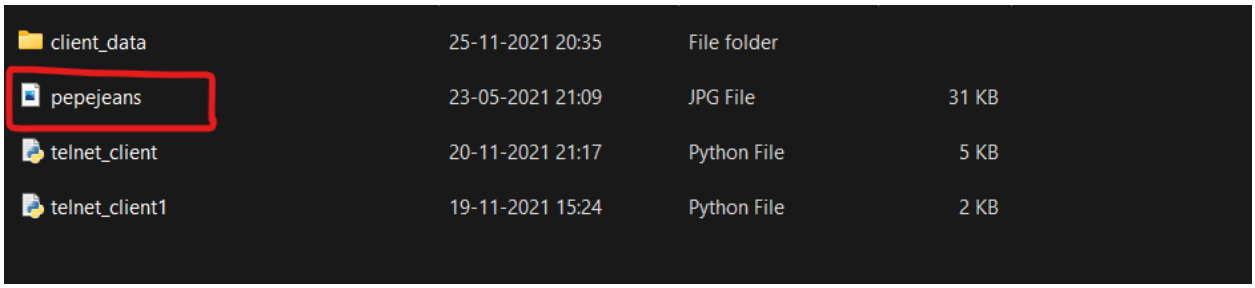
Server

```
*********************************************************
Command : upload
4000  bytes have been received so far
8000  bytes have been received so far
12000  bytes have been received so far
16000  bytes have been received so far
20000  bytes have been received so far
24000  bytes have been received so far
28000  bytes have been received so far
30976  bytes have been received so far
File uploaded at Server successfully!!
File saved at locaton :  server_data\recvd_pepejeans.jpg
history table updated
*********************************************************
```



Client

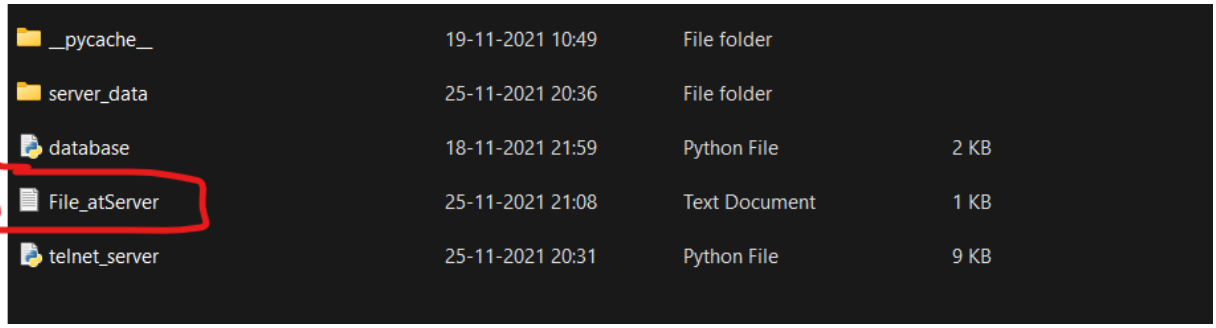
```
telnet> upload pepejeans.jpg
Filename and filesize received
Data received
Data received
Data received
Data received
Data received
Data received
Data received
Data received
**************************************************
Message from server :-
File is uploaded at Server successfully!!
**************************************************
```



6. **download** command:- this command helps the client to download a file present at the server.

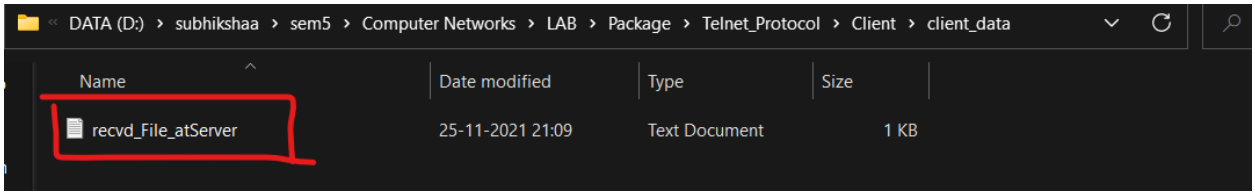
Server

```
**************************************************
Command : download
File downloaded from server Successfully!!
history table updated
**************************************************
```



Client

```
telnet> download File_atServer.txt
368  bytes have been received so far
File downloaded from Server successfully!!
File saved at locaton :  client_data\recvd_File_atServer.txt
************************************************
Message from server :-
DATA OVER
************************************************
```



7. **GET/HEAD** command:- In this command, the server tries to fetch details from the host specified by client (HTTP GET/HEAD).

   Link to HTTP website:- http://www.barc.gov.in/

   Server

```
************************************************************
Command : GET
history table updated
************************************************************
```

```
************************************************************
Command : HEAD
history table updated
************************************************************
```

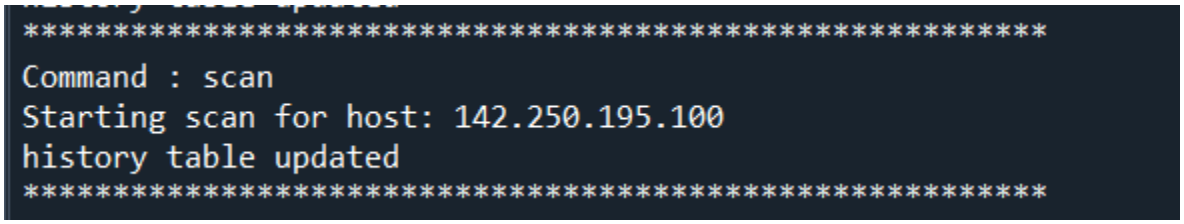
Client

```
Telnet> GET www.barc.gov.in
*********************************************
Message from server :-
HTTP/1.1 200 OK
Date: Thu, 25 Nov 2021 15:24:27 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Last-Modified: Thu, 25 Nov 2021 11:36:19 GMT
ETag: "266096d-bc93-5d19b62176694"
Accept-Ranges: bytes
Content-Length: 48275
X-Content-Type-Options: nosniff
X-XSS-Protection: 1;mode=block
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<meta name="MobileOptimized" content="width" />
<meta name="HandheldFriendly" content="true" />
<!--<meta http-equiv="Pragma" content="no-cache"/>-->
<meta name="description" content="A premier multi-disciplinary Nuclear Research Centre of India having
excellent infrastructure for advanced Research and Development with expertise covering the entire
spectrum of Nuclear Science and Engineering and related areas"/>
<meta name="robot" content="index,follow"/>
<meta name="refresh" content="30"/>
<meta name="copyright" content="Copyright © Bhabha Atomic Research Centre . All Rights Reserved."/>
<meta name="revisit-after" content="10"/>
```

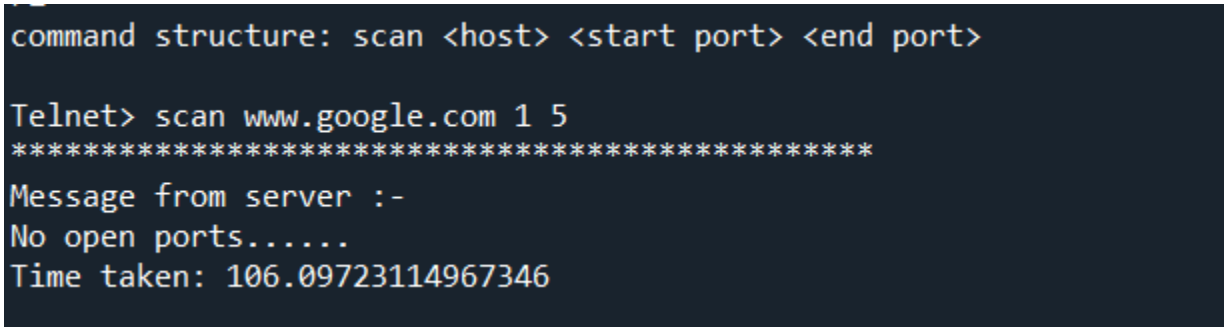
```
Telnet> HEAD www.barc.gov.in
*********************************************
Message from server :-
HTTP/1.1 200 OK
Date: Thu, 25 Nov 2021 15:26:50 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Last-Modified: Thu, 25 Nov 2021 11:36:19 GMT
ETag: "266096d-bc93-5d19b62176694"
Accept-Ranges: bytes
Content-Length: 48275
X-Content-Type-Options: nosniff
X-XSS-Protection: 1;mode=block
Connection: close
Content-Type: text/html; charset=UTF-8
```

8. **scan** command:- this command scans the host from start port to end port at the server-side specified by the client and returns the number of open ports.

   Server



   Client



9. **history** command:- this command keeps track of all the previously executed telnet commands with the help of a MySQL database. When the client requests this command server sends the history table.

   Server

Client

```
Port: 54535 Command: send   Time: 15:28:51  Date: 19/11/2021
Port: 54535 Command: encrypt    Time: 15:29:49  Date: 19/11/2021
Port: 56501 Command: encrypt    Time: 15:31:14  Date: 19/11/2021
Port: 59048 Command: encrypt    Time: 15:35:21  Date: 19/11/2021
Port: 59048 Command: encrypt    Time: 15:35:42  Date: 19/11/2021
Port: 54056 Command: exec   Time: 15:52:23  Date: 19/11/2021
Port: 60106 Command: encrypt    Time: 16:00:35  Date: 19/11/2021
Port: 60106 Command: exec   Time: 16:01:04  Date: 19/11/2021
Port: 60106 Command: exec   Time: 16:01:17  Date: 19/11/2021
Port: 63882 Command: help   Time: 16:04:53  Date: 19/11/2021
Port: 63882 Command: history    Time: 16:07:39  Date: 19/11/2021
Port: 56421 Command: help   Time: 20:02:11  Date: 20/11/2021
Port: 56421 Command: upload Time: 20:03:18  Date: 20/11/2021
Port: 56421 Command: download   Time: 20:04:36  Date: 20/11/2021
Port: 56421 Command: send   Time: 20:04:56  Date: 20/11/2021
Port: 56421 Command: encrypt    Time: 20:05:11  Date: 20/11/2021
Port: 56421 Command: exec   Time: 20:05:47  Date: 20/11/2021
Port: 56453 Command: GET    Time: 20:06:58  Date: 20/11/2021
Port: 56453 Command: scan   Time: 20:08:27  Date: 20/11/2021
Port: 56534 Command: history    Time: 20:09:13  Date: 20/11/2021
Port: 57338 Command: help   Time: 21:11:06  Date: 20/11/2021
Port: 57338 Command: history    Time: 21:11:11
*********************************************
```

10. **quit** command:- this command quits the connection between telnet client and server.

Server

```
************************************************************
Command : quit
Closing connection......
```

Client

```
telnet> quit

In [2]:
```

CODE:

database.py

```
import mysql.connector


db_conn=mysql.connector.connect(host="localhost",user="root",password="Subu
@2001",database="CN_DB")


db_cursor=db_conn.cursor()

db_cursor.execute("CREATE DATABASE IF NOT EXISTS CN_DB")

sql_query="CREATE TABLE history(port INT NOT NULL,command
VARCHAR(50),cmd_time CHAR(10),cmd_date CHAR(10))"

#Uncomment below line if you have to create table for first time

#db_cursor.execute(sql_query)


print("Connected to mysql database.....")


def insert_record(cursor, port, command, cmd_time, cmd_date):
```

```
    sql = "INSERT INTO history (port, command, cmd_time, cmd_date) VALUES
(%s, %s, %s, %s)"

    val = (port, command, cmd_time, cmd_date)

    cursor.execute(sql, val)

    db_conn.commit()

    print("history table updated")



def print_history_table(cursor):

    cursor.execute("SELECT * FROM history")

    res = cursor.fetchall()



    s = ""

    for row in res:

        s += "Port: " + str(row[0]) + "\t" + "Command: " + str(row[1]) + "\t" + "Time: "
+ str(row[2]) + "\t" + "Date: " \

             + str(row[3]) + "\n"

    return s
```

##

telnet_server.py

```
import threading

import os

import subprocess

import socket

from database import *

from datetime import datetime, date

import time

import string



HOST = socket.gethostbyname(socket.gethostname())

PORT = 23

address = (HOST, PORT)

buffer_size = 4000



all_letters= string.ascii_letters

spl_char=[' ','~', ':', "'", '+', '[', '\\', '@', '^', '{', '%', '(', '-', '"', '*', '|', ',', '&', '<', '`', '}', '.',
'_', '=', ']', '!', '>', ';', '?', '#', '$', ')', '/']

dict2 = {}
```

```
def client_thread(conn,addr):
    print(f"**Connected to {addr}**")
    conn.send("Welcome from Server".encode('utf-8'))
    while True:
        print("**********************************************************")
        data = conn.recv(buffer_size).decode()
        data = data.split()
        command = data[0]
        print(f"Command : {command}")


        conn_accepted=False


        if command=="exec":
            exec_msg = " ".join(data[i] for i in range(1,len(data)))
            result = subprocess.run(exec_msg, stdout=subprocess.PIPE)
            output = result.stdout.decode()
            output = "Command executed" + output
```

```
            conn.send(output.encode('utf-8'))

            print("Command executed and sent output to client successfully")

            conn_accepted=True



        elif command=="send":

            client_msg="Message recieved"

            print("Message from client : ",*data[1:])

            conn.send(client_msg.encode())

            conn_accepted = True



        elif command=="help":

            help_str = "Available commands:-\nupload <filename> : Upload the
file\ndownload <filename> : download the file\nexec <command>:execute
command and send output\n" \

                       "send <text> : send plain text message\nencrypt <text> : send
encrypted text message" \

                       "\nGET/HEAD <host>: send request to web server\nscan <host>
<start port> <end port>:scan ports start_port to end_port"\

                       "\nhistory : show the history\n"

            conn.send(help_str.encode('utf-8'))
```

```
        conn_accepted = True


    elif command=="history":
        histories = print_history_table(db_cursor)
        conn.send(("History table:-\n" + histories).encode('utf-8'))
        conn_accepted = True


    elif command == "quit":
        print("Closing connection.....")
        conn.send(b"SERVER DISCONNECTED.....")
        conn_accepted = True
        break


    elif command=="encrypt":
        key=int(data[1])
        enc_txt=" ".join(data[i] for i in range(2,len(data)))
        print("Encrypted Message from client : ",enc_txt)
        for i in range(len(all_letters)):
```

```
        dict2[all_letters[i]] = all_letters[(i-key)%(len(all_letters))]
    for i in range(len(spl_char)):
        dict2[spl_char[i]] = spl_char[(i-key)%len(spl_char)]
    dec_txt=""
    for char in enc_txt:
        if char in all_letters:
            temp = dict2[char]
        elif char in spl_char:
            temp = dict2[char]
        else:
            temp = char
        dec_txt+=temp
    client_msg="Message recieved"
    print("Decrypted Message from client : ",dec_txt)
    conn.send(client_msg.encode())
    conn_accepted=True

elif command == "upload":
```

```
path = data[1]

filename, filesize = path.split(":")

filesize = int(filesize)

conn.send("Filename and filesize received".encode())

size_tmp = 0

name = f"recvd_{filename}"

filepath = os.path.join("server_data", name)

with open(filepath, "wb") as f:
    while True:
        file_data = conn.recv(buffer_size)
        size_tmp += len(file_data)
        print(size_tmp, " bytes have been received so far")
        if not file_data:
            break
        if file_data == b'DATA OVER':
            break
        f.write(file_data)
        conn.send("Data received".encode('utf-8'))
```

```
            if size_tmp >= filesize:
                break
    conn.send("File is uploaded at Server successfully!!".encode())
    print("File uploaded at Server successfully!!")
    print("File saved at locaton : ",filepath)
    conn_accepted = True

elif command=="download":
    filename=data[1]
    if os.path.isfile(filename):
        conn.send("File exists".encode("utf-8"))
        f_size=os.path.getsize(filename)
        conn.send(str(f_size).encode("utf-8"))
        with open(filename, "rb") as f:
            while True:
                dt = f.read(buffer_size)

                if not dt:
```

```
                conn.send("DATA OVER".encode())
                break
            conn.sendall(dt)
            msg1 = conn.recv(buffer_size).decode()
            print(msg1)
    else:
        conn.send("File does not exist".encode("utf-8"))
    conn_accepted=True

elif command=="GET" or command=="HEAD":
    ip = data[1]
    target_port = 80
    http_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        http_client.connect((ip, target_port))
    except socket.gaierror:
        conn.send(b'Hostname could not be resolved')
        print("Hostname could not be resolved")
```

```
            break

        request = command + " / HTTP/1.1\r\nHost:+ " + ip + "\r\nAccept:
text/html\r\n\r\n"

        http_client.sendall(request.encode('utf-8'))

        recvd_data=http_client.recv(buffer_size)

        conn.send(recvd_data)

        http_client.close()

        conn_accepted=True



    elif command=="scan":

        host=data[1]

        startport=int(data[2])

        endport=int(data[3])

        start_time = time.time()

        try:

            target_host = socket.gethostbyname(host)

        except socket.gaierror:

            conn.send(b'Hostname could not be resolved')
```

```
    break


print("Starting scan for host:", target_host)
flag=0
for i in range(startport, endport+1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


    conn1 = s.connect_ex((target_host, i))
    if conn1 == 0:
        flag=1
        op=f"Port {i}: OPEN"
        conn.send(op.encode("utf-8"))
    s.close()
if flag==0:
    op=f"No open ports.....\nTime taken: {time.time() - start_time}"
else:
  op=f"Time taken: {time.time() - start_time}"
conn.send(op.encode("utf-8"))
```

```
            conn_accepted=True


        else:
            conn.send(b'Invalid command...')


        if conn_accepted:
            now = datetime.now()
            curr_time = now.strftime("%H:%M:%S")
            today = date.today()
            tdy_date = today.strftime("%d/%m/%Y")
            insert_record(db_cursor, str(addr[1]), command, curr_time, tdy_date)


    conn.close()




def start(address,HOST):
    print("TELNET SERVER STARTING...")
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(address)
    server_socket.listen(4)
    print("LISTENING...")
    while True:
        conn, addr = server_socket.accept()
        thread = threading.Thread(target=client_thread, args=(conn, addr))
        thread.start()



start(address,HOST)
```

##

telnet_client.py

```
import os
import socket
import sys
import string
```

```
import random



HOST = socket.gethostbyname(socket.gethostname())

PORT = 23

address = (HOST, PORT)

buffer_size = 4000



all_letters= string.ascii_letters

spl_char=[' ','~', ':', "'", '+', '[', '\\', '@', '^', '{', '%', '(', '-', '"', '*', '|', ',', '&', '<', '`', '}', '.',
'_', '=', ']', '!', '>', ';', '?', '#', '$', ')', '/']

dict1 = {}



while True:

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    sock.connect(address)

    while True:

        server_data = sock.recv(buffer_size).decode()

        if server_data:
```

```
    print("************************************************")
    print("Message from server :-")
    print(server_data)
print("************************************************")
print("\nType help to see available commands :)\n")
user_inp = input("telnet> ")
data = user_inp.split()
command = data[0]

if command=="exec":
    sock.send(user_inp.encode('utf-8'))

elif command=="send":
    sock.send(user_inp.encode('utf-8'))

elif command=="encrypt":
    key = random.randint(1,15)
    for i in range(len(all_letters)):
```

```
        dict1[all_letters[i]] = all_letters[(i+key)%len(all_letters)]
    for i in range(len(spl_char)):
        dict1[spl_char[i]] = spl_char[(i+key)%len(spl_char)]


    msg=" ".join(data[i] for i in range(1,len(data)))
    enc_txt=""
    for char in msg:
        if char in all_letters:
            temp = dict1[char]
        elif char in spl_char:
            temp = dict1[char]
        else:
            temp = char
        enc_txt+=temp
    final_msg="encrypt "+str(key)+" "+enc_txt
    sock.send(final_msg.encode('utf-8'))


elif command=="help":
```

```
    sock.send(user_inp.encode('utf-8'))


elif command=="history":
   sock.send(user_inp.encode('utf-8'))


elif command == "quit":
    sock.send(user_inp.encode('utf-8'))
    sys.exit()


elif command == "upload":
    f_name = data[1]
    if os.path.isfile(f_name):
        f_size = os.path.getsize(f_name)
        user_inp += ":" + str(f_size)
        sock.send(user_inp.encode())
        msg = sock.recv(buffer_size).decode()
        print(msg)
```

```
        with open(f_name, "rb") as f:
            while True:
                dt = f.read(buffer_size)


                if not dt:
                    #sock.send("DATA OVER".encode())
                    break
                sock.sendall(dt)
                msg1 = sock.recv(buffer_size).decode()
                print(msg1)
    else:
        print("File does not exist.....\n")
        break


elif command=="download":
    sock.send(user_inp.encode('utf-8'))
    fileexists=sock.recv(buffer_size).decode()
    if fileexists=="File exists":
```

```
name=f"recvd_{data[1]}"
filepath = os.path.join("client_data", name)
filesize=sock.recv(buffer_size).decode()
filesize=int(filesize)
size_tmp=0
with open(filepath, "wb") as f:
    while True:
        file_data = sock.recv(buffer_size)
        size_tmp += len(file_data)
        print(size_tmp, " bytes have been received so far")
        if not file_data:
            break
        if file_data == b'DATA OVER':
            break
        f.write(file_data)

        if size_tmp >= filesize:
            break
```

```
            print("File downloaded from Server successfully!!")
            print("File saved at locaton : ",filepath)
            sock.send("File downloaded from server Successfully!!".encode('utf-8'))
        elif fileexists=="File does not exist":
            print("Provided file name does not exist in the server...")
            break
    else:
        print("Invalid command.....")
        break

sock.close()
```

##

telnet_client1.py

```
import os
import socket
import sys
```

```
HOST = socket.gethostbyname(socket.gethostname())

PORT = 23

address = (HOST, PORT)

buffer_size = 4000



while True:

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    sock.connect(address)

    while True:

        server_data = sock.recv(buffer_size).decode()

        if server_data:

            print("************************************************")

            print("Message from server :-")

            print(server_data)



        print("\n1-Send Request to http web server(GET/HEAD)\n2-scan ip for open
ports\n3-quit")
```

```
c = input(">")


if c == '1':
    print("command structure: <Request type> <host>")
    cmd = input("Telnet> ")
    sock.send(cmd.encode('utf-8'))


elif c == '2':
    print("command structure: scan <host> <start port> <end port>")
    cmd = input("Telnet> ")
    sock.send(cmd.encode('utf-8'))


elif c=='3':
    sock.send(b'quit')
    sys.exit()


else:
    print("Invalid option....")
```

```
        break



    sock.close()
```

##

THANK YOU