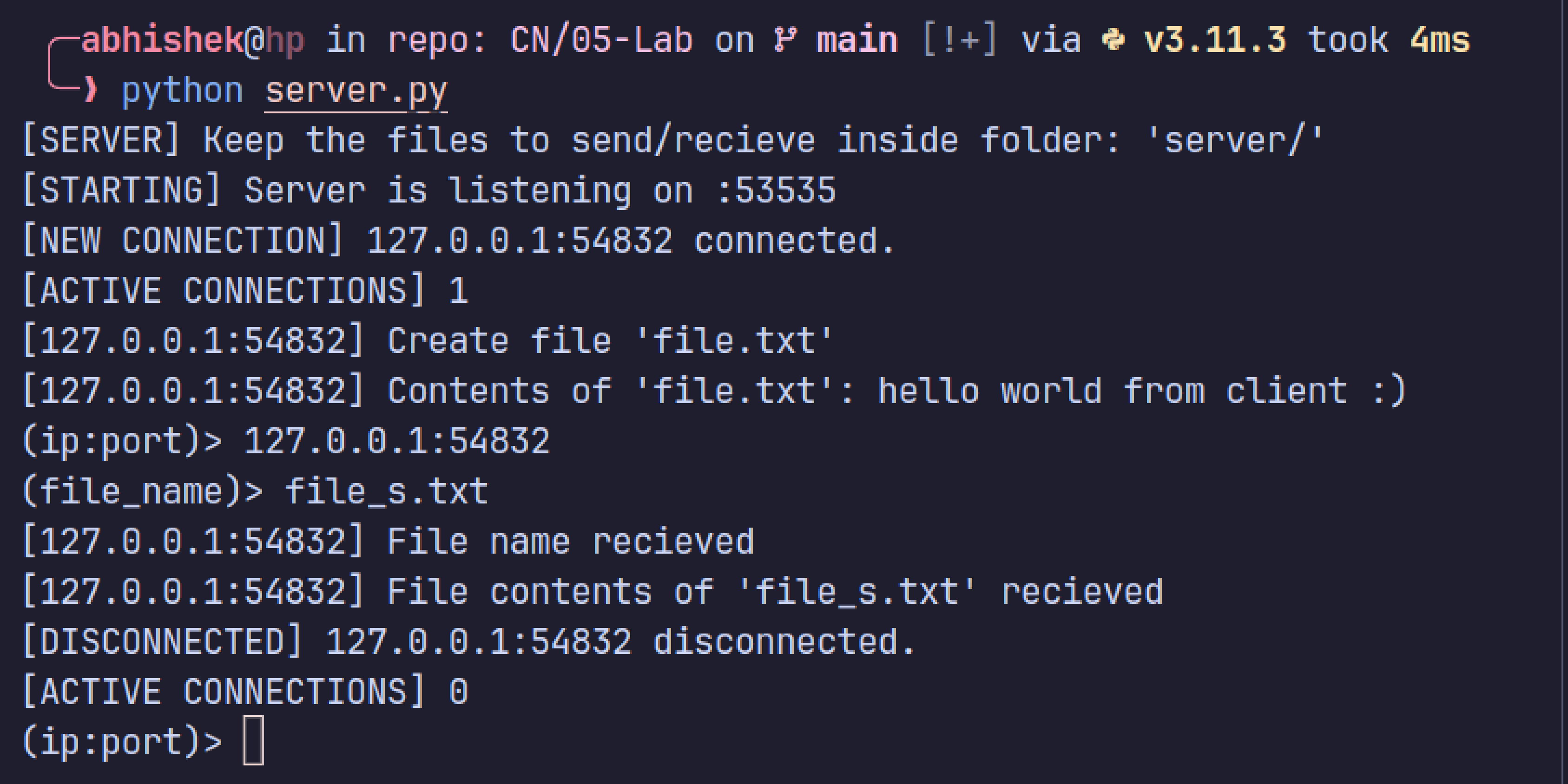
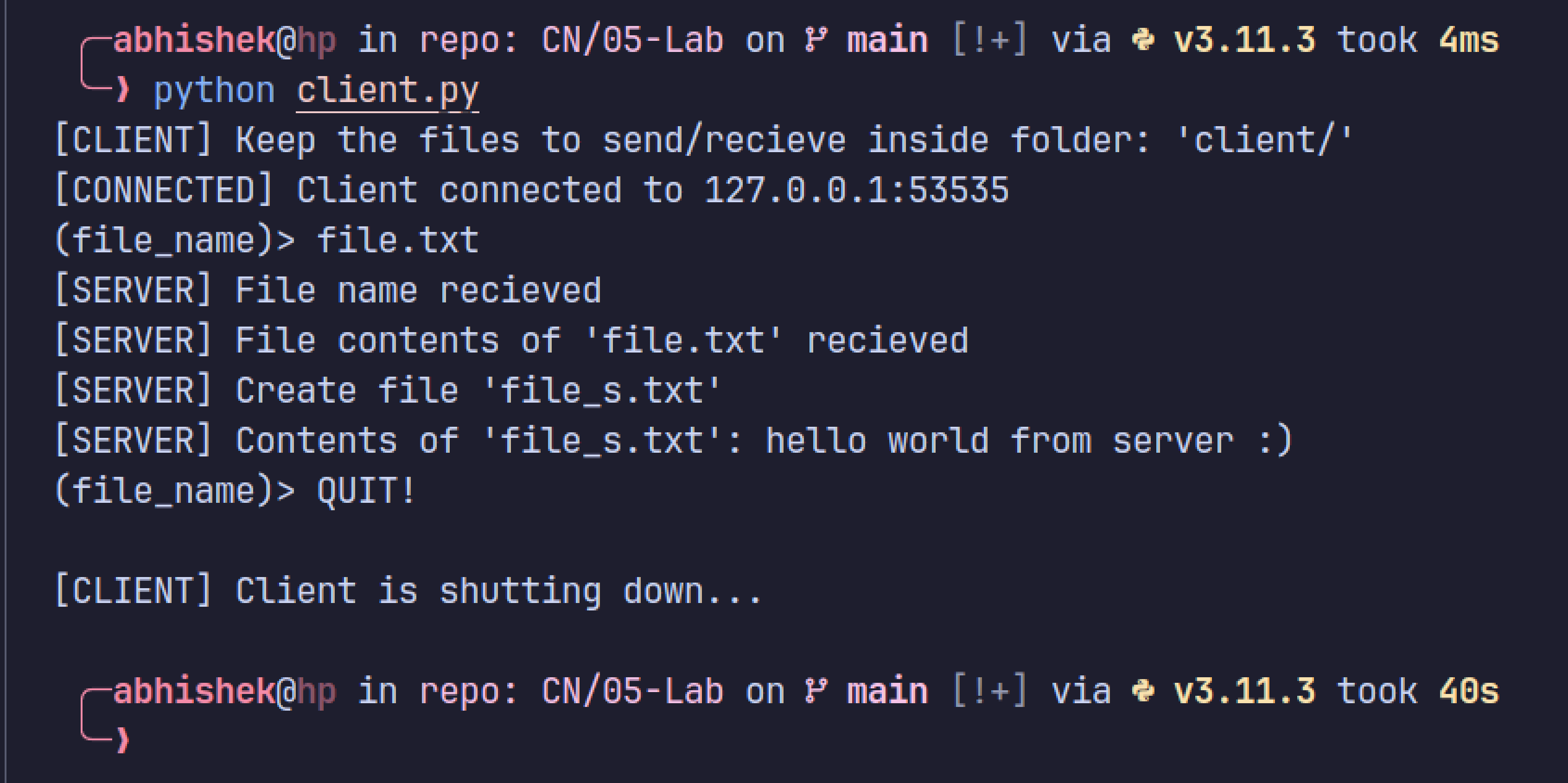


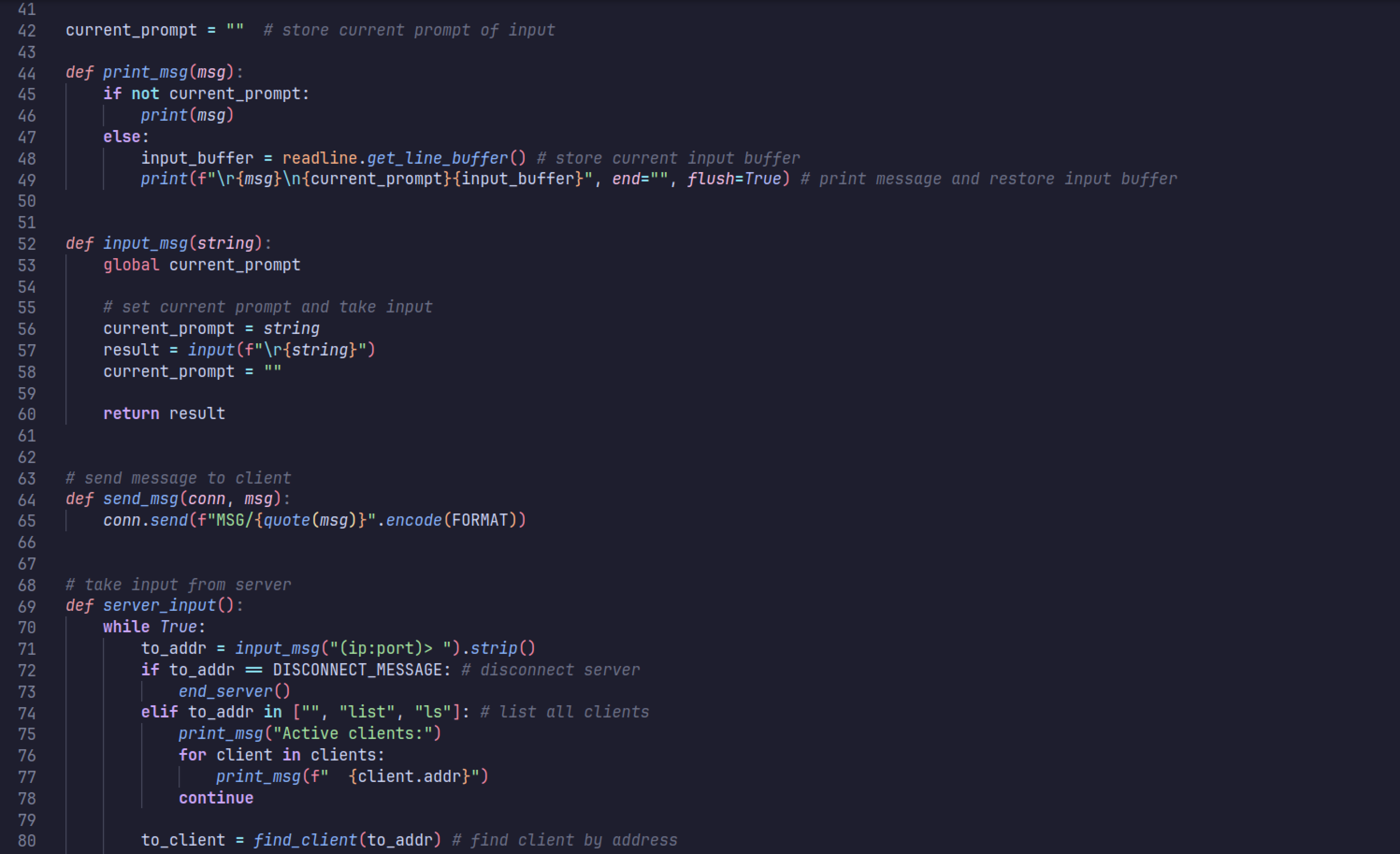
```
┌abhishek@hp in repo: CN/05-Lab on ᚠ main [!+] via  v3.11.3 took 4ms
└❯ python server.py
[SERVER] Keep the files to send/recieve inside folder: 'server/'
[STARTING] Server is listening on :53535
[NEW CONNECTION] 127.0.0.1:54832 connected.
[ACTIVE CONNECTIONS] 1
[127.0.0.1:54832] Create file 'file.txt'
[127.0.0.1:54832] Contents of 'file.txt': hello world from client :)
(ip:port)> 127.0.0.1:54832
(file_name)> file_s.txt
[127.0.0.1:54832] File name recieved
[127.0.0.1:54832] File contents of 'file_s.txt' recieved
[DISCONNECTED] 127.0.0.1:54832 disconnected.
[ACTIVE CONNECTIONS] 0
(ip:port)> 
```



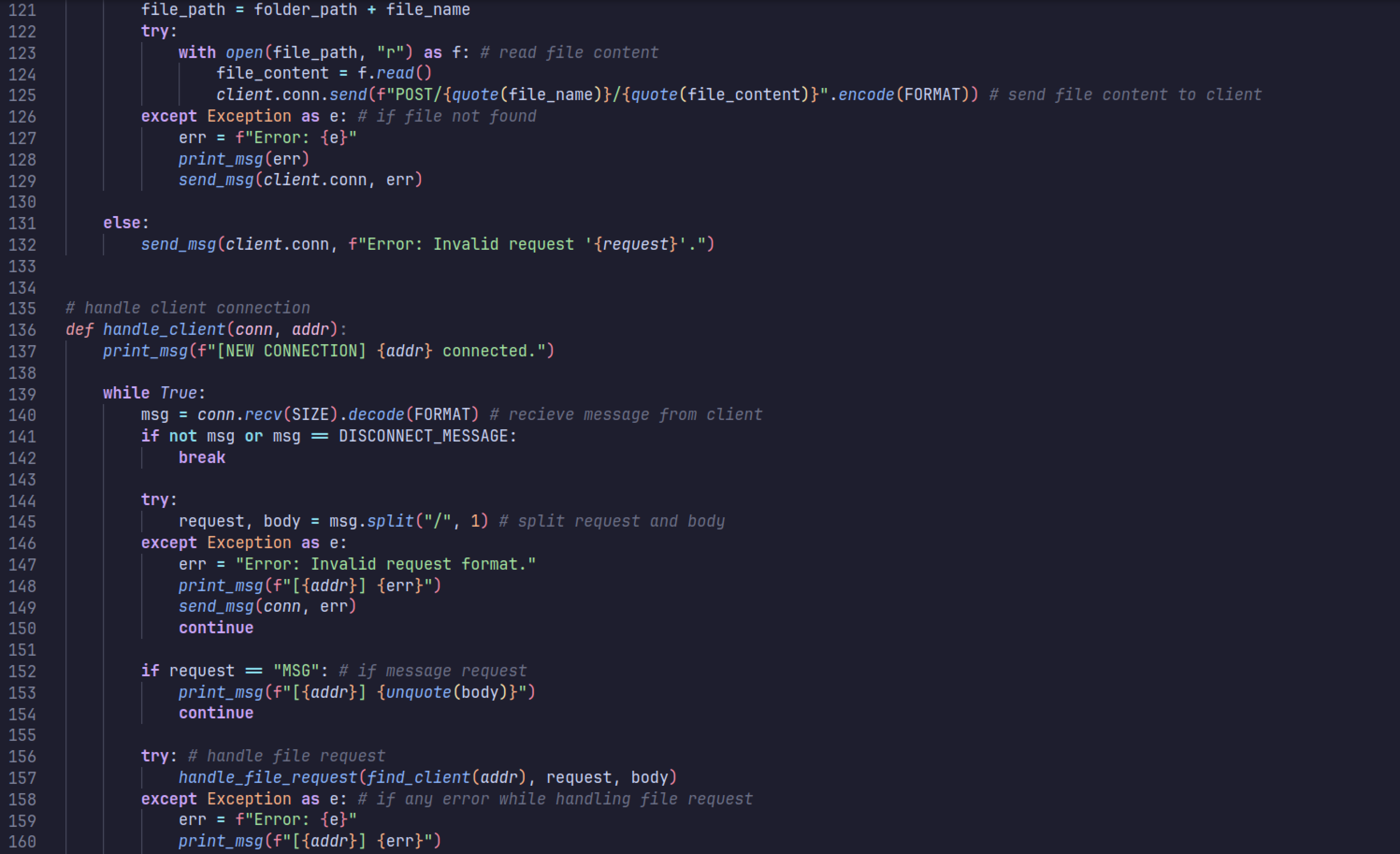
```
┌abhishek@hp in repo: CN/05-Lab on ᚠ main [!+] via  v3.11.3 took 4ms
└❯ python client.py
[CLIENT] Keep the files to send/recieve inside folder: 'client/'
[CONNECTED] Client connected to 127.0.0.1:53535
(file_name)> file.txt
[SERVER] File name recieved
[SERVER] File contents of 'file.txt' recieved
[SERVER] Create file 'file_s.txt'
[SERVER] Contents of 'file_s.txt': hello world from server :)
(file_name)> QUIT!

[CLIENT] Client is shutting down...

┌abhishek@hp in repo: CN/05-Lab on ᚠ main [!+] via  v3.11.3 took 40s
└❯
```

05-Lab > server.py > ...

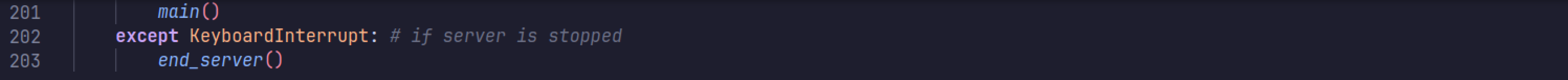
```python
import socket
import threading
import os
import time
import readline
from urllib.parse import quote, unquote # for encoding/decoding data in url format
from dataclasses import dataclass

# IP = socket.gethostbyname(socket.gethostname())
IP = ''
PORT = 53535
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
DISCONNECT_MESSAGE = "QUIT!"

# MSG Format:
#   MSG/<msg>
#   CREATE/<file_name>
#   POST/<file_name>/<file_content>
#   GET/<file_name>

folder_path = "server/" # folder to store files for server

clients = [] # list of all clients connected to server

# Client class to store client connection and address
@dataclass
class Client:
    conn: socket.socket
    addr: str


# find client by address
def find_client(addr):
    for client in clients:
        if client.addr == addr:
            return client
    return None

```



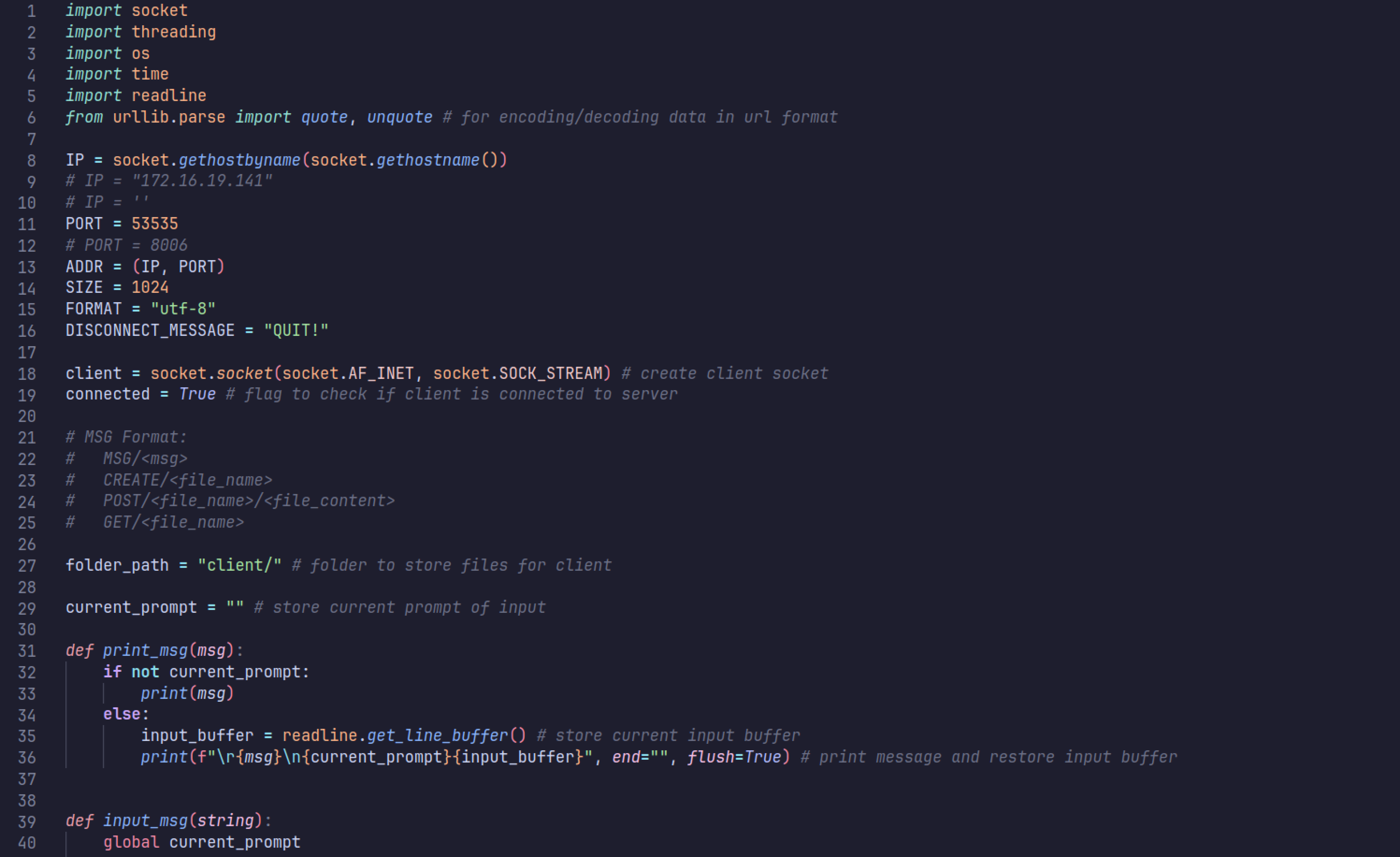
```python

current_prompt = ""  # store current prompt of input

def print_msg(msg):
    if not current_prompt:
        print(msg)
    else:
        input_buffer = readline.get_line_buffer() # store current input buffer
        print(f"\r{msg}\n{current_prompt}{input_buffer}", end="", flush=True) # print message and restore input buffer


def input_msg(string):
    global current_prompt

    # set current prompt and take input
    current_prompt = string
    result = input(f"\r{string}")
    current_prompt = ""

    return result


# send message to client
def send_msg(conn, msg):
    conn.send(f"MSG/{quote(msg)}".encode(FORMAT))


# take input from server
def server_input():
    while True:
        to_addr = input_msg("(ip:port)> ").strip()
        if to_addr == DISCONNECT_MESSAGE: # disconnect server
            end_server()
        elif to_addr in ["", "list", "ls"]: # list all clients
            print_msg("Active clients:")
            for client in clients:
                print_msg(f"  {client.addr}")
            continue

        to_client = find_client(to_addr) # find client by address
        if to_client is None:
            print_msg(f"Error: Client '{to_addr}' not found.")
            continue

        msg = input_msg("(file_name)> ") # take file name
        if msg == DISCONNECT_MESSAGE:
            end_server()
        elif msg == "":
            continue

        # send file name to client
        to_client.conn.send(f"CREATE/{quote(msg)}".encode(FORMAT))


# handle the file requests from client
def handle_file_request(client, request, body):
    # CREATE/<file_name>
    if request == "CREATE":
        file_name = unquote(body) # decode file name
        file_path = folder_path + file_name
        print_msg(f"[{client.addr}] Create file '{file_name}'")
        with open(file_path, "w"): # create file
            msg = f"File name recieved"
            send_msg(client.conn, msg) # send acknowledgement to client
            time.sleep(0.1)
            client.conn.send(f"GET/{quote(file_name)}".encode(FORMAT)) # send GET request to client to get file contents

    # POST/<file_name>/<file_content>
    elif request == "POST":
        file_name, file_content = map(unquote, body.split("/", 1)) # decode file name and content
        file_path = folder_path + file_name
        print_msg(f"[{client.addr}] Contents of '{file_name}': {file_content}")
        with open(file_path, "a") as f:
            f.write(file_content) # write file content to file
            msg = f"File contents of '{file_name}' recieved"
            send_msg(client.conn, msg) # send acknowledgement to client

    # GET/<file_name>
    elif request == "GET":
        file_name = unquote(body) # decode file name
```



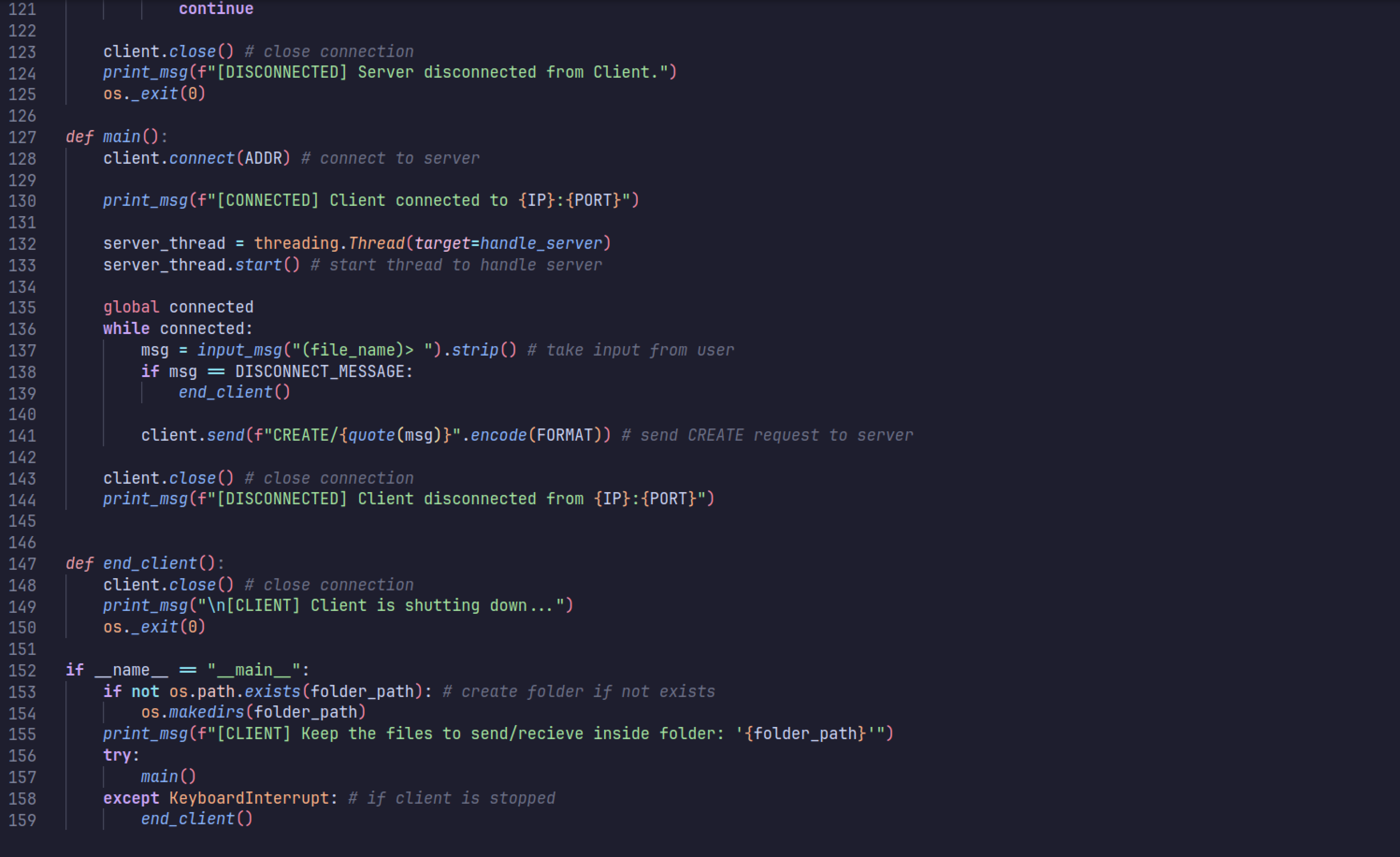
```python
        file_path = folder_path + file_name
        try:
            with open(file_path, "r") as f: # read file content
                file_content = f.read()
                client.conn.send(f"POST/{quote(file_name)}/{quote(file_content)}".encode(FORMAT)) # send file content to client
        except Exception as e: # if file not found
            err = f"Error: {e}"
            print_msg(err)
            send_msg(client.conn, err)

    else:
        send_msg(client.conn, f"Error: Invalid request '{request}'.")


# handle client connection
def handle_client(conn, addr):
    print_msg(f"[NEW CONNECTION] {addr} connected.")

    while True:
        msg = conn.recv(SIZE).decode(FORMAT) # recieve message from client
        if not msg or msg == DISCONNECT_MESSAGE:
            break

        try:
            request, body = msg.split("/", 1) # split request and body
        except Exception as e:
            err = "Error: Invalid request format."
            print_msg(f"[{addr}] {err}")
            send_msg(conn, err)
            continue

        if request == "MSG": # if message request
            print_msg(f"[{addr}] {unquote(body)}")
            continue

        try: # handle file request
            handle_file_request(find_client(addr), request, body)
        except Exception as e: # if any error while handling file request
            err = f"Error: {e}"
            print_msg(f"[{addr}] {err}")
            send_msg(conn, err)
            continue

    conn.close() # close connection
    clients.remove(Client(conn, addr)) # remove client from list
    print_msg(f"[DISCONNECTED] {addr} disconnected.")
    print_msg(f"[ACTIVE CONNECTIONS] {threading.active_count()-3}")

def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # create server socket
    server.bind(ADDR) # bind server socket to address
    server.listen() # start listening for connections
    print_msg(f"[STARTING] Server is listening on {IP}:{PORT}")

    server_input_thread = threading.Thread(target=server_input)
    server_input_thread.start() # start server input thread

    while True:
        conn, addr = server.accept() # accept connection
        addr = f"{addr[0]}:{addr[1]}"

        clients.append(Client(conn, addr)) # add client to list

        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start() # start client thread

        print_msg(f"[ACTIVE CONNECTIONS] {threading.active_count()-2}")


def end_server():
    print_msg("\n[EXITING] Server is shutting down...")
    for client in clients: # send disconnect message to all clients
        client.conn.send(DISCONNECT_MESSAGE.encode(FORMAT))
    os._exit(0)

if __name__ == "__main__":
    if not os.path.exists(folder_path): # create folder if not exists
        os.makedirs(folder_path)
    print_msg(f"[SERVER] Keep the files to send/recieve inside folder: '{folder_path}'")
    try:
```



```python
        main()
    except KeyboardInterrupt: # if server is stopped
        end_server()
```



```python
import socket
import threading
import os
import time
import readline
from urllib.parse import quote, unquote # for encoding/decoding data in url format

IP = socket.gethostbyname(socket.gethostname())
# IP = "172.16.19.141"
# IP = ''
PORT = 53535
# PORT = 8006
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
DISCONNECT_MESSAGE = "QUIT!"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # create client socket
connected = True # flag to check if client is connected to server

# MSG Format:
#   MSG/<msg>
#   CREATE/<file_name>
#   POST/<file_name>/<file_content>
#   GET/<file_name>

folder_path = "client/" # folder to store files for client

current_prompt = "" # store current prompt of input

def print_msg(msg):
    if not current_prompt:
        print(msg)
    else:
        input_buffer = readline.get_line_buffer() # store current input buffer
        print(f"\r{msg}\n{current_prompt}{input_buffer}", end="", flush=True) # print message and restore input buffer


def input_msg(string):
    global current_prompt

    # set current prompt and take input
    current_prompt = string
    result = input(f"\r{string}")
    current_prompt = ""

    return result


# send message to server
def send_msg(conn, msg):
    conn.send(f"MSG/{quote(msg)}".encode(FORMAT))


# handle file requests from server
def handle_file_request(request, body):
    # CREATE/<file_name>
    if request == "CREATE":
        file_name = unquote(body) # decode file name
        file_path = folder_path + file_name
        print_msg(f"[SERVER] Create file '{file_name}'")
        with open(file_path, "w"): # create file
            msg = f"File name recieved"
            send_msg(client, msg) # send acknowledgement to server
            time.sleep(0.1)
            client.send(f"GET/{quote(file_name)}".encode(FORMAT)) # send GET request to server to get file contents

    # POST/<file_name>/<file_content>
    elif request == "POST":
        file_name, file_content = map(unquote, body.split("/", 1)) # decode file name and file content
        file_path = folder_path + file_name
        print_msg(f"[SERVER] Contents of '{file_name}': {file_content}")
        with open(file_path, "a") as f:
            f.write(file_content) # write file content to file
            msg = f"File contents of '{file_name}' recieved"
            send_msg(client, msg) # send acknowledgement to server

    # GET/<file_name>
    elif request == "GET":
        file_name = unquote(body) # decode file name
        file_path = folder_path + file_name
        try:
            with open(file_path, "r") as f: # read file contents
                file_content = f.read()
                client.send(f"POST/{quote(file_name)}/{quote(file_content)}".encode(FORMAT)) # send file contents to server
        except Exception as e: # if file not found
            err = f"Error: {e}"
            print_msg(err)
            send_msg(client, err)

    else:
        send_msg(client, f"Error: Invalid request '{request}'.")


def handle_server():
    global connected
    while connected:
        msg = client.recv(SIZE).decode(FORMAT) # recieve message from server
        if not msg or msg == DISCONNECT_MESSAGE:
            connected = False
            end_client()

        try:
            request, body = msg.split("/", 1) # split request and body
        except Exception as e: # if invalid request format
            err = "Error: Invalid request format."
            print_msg(err)
            send_msg(client, err)
            continue

        if request == "MSG": # if message request
            print_msg(f"[SERVER] {unquote(body)}")
            continue

        try: # handle file request
            handle_file_request(request, body)
        except Exception as e: # if error occured while handling file request
            err = f"Error: {e}"
            print_msg(err)
            send_msg(client, err)
```


```python
            continue

    client.close() # close connection
    print_msg(f"[DISCONNECTED] Server disconnected from Client.")
    os._exit(0)

def main():
    client.connect(ADDR) # connect to server

    print_msg(f"[CONNECTED] Client connected to {IP}:{PORT}")

    server_thread = threading.Thread(target=handle_server)
    server_thread.start() # start thread to handle server

    global connected
    while connected:
        msg = input_msg("(file_name)> ").strip() # take input from user
        if msg == DISCONNECT_MESSAGE:
            end_client()

        client.send(f"CREATE/{quote(msg)}".encode(FORMAT)) # send CREATE request to server

    client.close() # close connection
    print_msg(f"[DISCONNECTED] Client disconnected from {IP}:{PORT}")


def end_client():
    client.close() # close connection
    print_msg("\n[CLIENT] Client is shutting down...")
    os._exit(0)

if __name__ == "__main__":
    if not os.path.exists(folder_path): # create folder if not exists
        os.makedirs(folder_path)
    print_msg(f"[CLIENT] Keep the files to send/recieve inside folder: '{folder_path}'")
    try:
        main()
    except KeyboardInterrupt: # if client is stopped
        end_client()
```