**[Network](https://cn.bing.com/dict/search?q=Network&FORM=BDVSP6&cc=cn) [Communication](https://cn.bing.com/dict/search?q=Communication&FORM=BDVSP6&cc=cn) about scoket for python**

# server.py

import socket

import threading

class ChatServer:

def \_\_init\_\_(self, host='0.0.0.0', port=5555):

self.host = host

self.port = port

self.server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server.bind((host, port))

self.server.listen()

self.clients = []

self.nicknames = []

def broadcast(self, message):

for client in self.clients:

client.send(message)

def handle\_client(self, client):

while True:

try:

message = client.recv(1024)

self.broadcast(message)

except:

index = self.clients.index(client)

self.clients.remove(client)

client.close()

nickname = self.nicknames[index]

self.broadcast(f'{nickname} left the chat!'.encode('utf-8'))

self.nicknames.remove(nickname)

break

def receive(self):

print("Server is running and listening...")

while True:

client, address = self.server.accept()

print(f"Connected with {str(address)}")

client.send('NICK'.encode('utf-8'))

nickname = client.recv(1024).decode('utf-8')

self.nicknames.append(nickname)

self.clients.append(client)

print(f"Nickname of the client is {nickname}!")

self.broadcast(f"{nickname} joined the chat!".encode('utf-8'))

client.send('Connected to the server!'.encode('utf-8'))

thread = threading.Thread(target=self.handle\_client, args=(client,))

thread.start()

if \_\_name\_\_ == "\_\_main\_\_":

server = ChatServer()

server.receive()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# client.py

import socket

import threading

class ChatClient:

def \_\_init\_\_(self, host='127.0.0.1', port=5555):

self.host = host

self.port = port

self.client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.client.connect((host, port))

def receive(self):

while True:

try:

message = self.client.recv(1024).decode('utf-8')

if message == 'NICK':

self.client.send(input("Choose your nickname: ").encode('utf-8'))

else:

print(message)

except:

print("An error occurred!")

self.client.close()

break

def write(self):

while True:

message = f"{input('')}"

self.client.send(message.encode('utf-8'))

def run(self):

receive\_thread = threading.Thread(target=self.receive)

receive\_thread.start()

write\_thread = threading.Thread(target=self.write)

write\_thread.start()

if \_\_name\_\_ == "\_\_main\_\_":

client = ChatClient()

client.run()

# server.py for tkinter

import socket

import threading

import tkinter as tk

from tkinter import scrolledtext, messagebox

class ChatServer:

def \_\_init\_\_(self):

self.server\_socket = None

self.clients = {}

self.setup\_ui()

def setup\_ui(self):

self.root = tk.Tk()

self.root.title("聊天服务器")

# 服务器控制区域

control\_frame = tk.Frame(self.root)

control\_frame.pack(pady=10)

tk.Label(control\_frame, text="端口:").grid(row=0, column=0)

self.port\_entry = tk.Entry(control\_frame, width=10)

self.port\_entry.grid(row=0, column=1)

self.port\_entry.insert(0, "12345")

self.start\_btn = tk.Button(control\_frame, text="启动服务器", command=self.start\_server)

self.start\_btn.grid(row=0, column=2, padx=5)

self.stop\_btn = tk.Button(control\_frame, text="停止服务器", command=self.stop\_server, state=tk.DISABLED)

self.stop\_btn.grid(row=0, column=3)

# 日志区域

tk.Label(self.root, text="服务器日志:").pack(anchor=tk.W, padx=10)

self.log\_area = scrolledtext.ScrolledText(self.root, width=60, height=20, state='disabled')

self.log\_area.pack(padx=10, pady=5)

self.root.protocol("WM\_DELETE\_WINDOW", self.on\_closing)

def start\_server(self):

try:

port = int(self.port\_entry.get())

if port < 1 or port > 65535:

raise ValueError("端口号必须在1-65535之间")

self.server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server\_socket.bind(('0.0.0.0', port))

self.server\_socket.listen(5)

self.log\_message(f"服务器已启动，监听端口 {port}")

self.start\_btn.config(state=tk.DISABLED)

self.stop\_btn.config(state=tk.NORMAL)

self.port\_entry.config(state='readonly')

# 启动接受客户端连接的线程

accept\_thread = threading.Thread(target=self.accept\_clients, daemon=True)

accept\_thread.start()

except Exception as e:

messagebox.showerror("错误", f"启动服务器失败: {str(e)}")

def stop\_server(self):

try:

if self.server\_socket:

# 通知所有客户端服务器关闭

for client\_socket in list(self.clients.keys()):

try:

client\_socket.send("服务器关闭".encode('utf-8'))

client\_socket.close()

except:

pass

self.server\_socket.close()

self.server\_socket = None

self.clients.clear()

self.log\_message("服务器已停止")

self.start\_btn.config(state=tk.NORMAL)

self.stop\_btn.config(state=tk.DISABLED)

self.port\_entry.config(state='normal')

except Exception as e:

messagebox.showerror("错误", f"停止服务器失败: {str(e)}")

def accept\_clients(self):

while True:

try:

client\_socket, client\_address = self.server\_socket.accept()

self.log\_message(f"新客户端连接: {client\_address}")

# 接收客户端用户名

username = client\_socket.recv(1024).decode('utf-8')

self.clients[client\_socket] = username

# 广播新用户加入

self.broadcast\_message(f"{username} 加入了聊天室", exclude=client\_socket)

client\_socket.send(f"欢迎来到聊天室, {username}!".encode('utf-8'))

# 为每个客户端启动接收消息的线程

client\_thread = threading.Thread(

target=self.handle\_client,

args=(client\_socket, client\_address),

daemon=True

)

client\_thread.start()

except Exception as e:

if self.server\_socket is None:

break

self.log\_message(f"接受客户端连接时出错: {str(e)}")

continue

def handle\_client(self, client\_socket, client\_address):

while True:

try:

message = client\_socket.recv(1024).decode('utf-8')

if not message:

break

username = self.clients[client\_socket]

formatted\_message = f"{username}: {message}"

self.log\_message(formatted\_message)

self.broadcast\_message(formatted\_message, exclude=client\_socket)

except Exception as e:

self.log\_message(f"与客户端 {client\_address} 通信时出错: {str(e)}")

break

# 客户端断开连接

if client\_socket in self.clients:

username = self.clients[client\_socket]

self.log\_message(f"客户端断开连接: {username} ({client\_address})")

del self.clients[client\_socket]

self.broadcast\_message(f"{username} 离开了聊天室", exclude=client\_socket)

client\_socket.close()

def broadcast\_message(self, message, exclude=None):

for client\_socket in list(self.clients.keys()):

if client\_socket != exclude:

try:

client\_socket.send(message.encode('utf-8'))

except:

pass

def log\_message(self, message):

self.log\_area.config(state='normal')

self.log\_area.insert(tk.END, message + "\n")

self.log\_area.config(state='disabled')

self.log\_area.see(tk.END)

def on\_closing(self):

if self.server\_socket:

if messagebox.askokcancel("退出", "服务器正在运行，确定要退出吗？"):

self.stop\_server()

self.root.destroy()

else:

self.root.destroy()

def run(self):

self.root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

server = ChatServer()

server.run()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# client.py for tkinter

import socket

import threading

import tkinter as tk

from tkinter import scrolledtext, messagebox, simpledialog

class ChatClient:

def \_\_init\_\_(self):

self.client\_socket = None

self.running = False

self.setup\_ui()

def setup\_ui(self):

self.root = tk.Tk()

self.root.title("聊天客户端")

# 连接控制区域

connect\_frame = tk.Frame(self.root)

connect\_frame.pack(pady=10)

tk.Label(connect\_frame, text="服务器地址:").grid(row=0, column=0)

self.host\_entry = tk.Entry(connect\_frame, width=15)

self.host\_entry.grid(row=0, column=1)

self.host\_entry.insert(0, "localhost")

tk.Label(connect\_frame, text="端口:").grid(row=0, column=2, padx=(10,0))

self.port\_entry = tk.Entry(connect\_frame, width=8)

self.port\_entry.grid(row=0, column=3)

self.port\_entry.insert(0, "12345")

self.connect\_btn = tk.Button(connect\_frame, text="连接", command=self.connect\_to\_server)

self.connect\_btn.grid(row=0, column=4, padx=5)

self.disconnect\_btn = tk.Button(connect\_frame, text="断开", command=self.disconnect\_from\_server, state=tk.DISABLED)

self.disconnect\_btn.grid(row=0, column=5)

# 聊天显示区域

tk.Label(self.root, text="聊天内容:").pack(anchor=tk.W, padx=10)

self.chat\_area = scrolledtext.ScrolledText(self.root, width=60, height=20, state='disabled')

self.chat\_area.pack(padx=10, pady=5)

# 消息发送区域

message\_frame = tk.Frame(self.root)

message\_frame.pack(pady=10)

tk.Label(message\_frame, text="消息:").grid(row=0, column=0)

self.message\_entry = tk.Entry(message\_frame, width=50)

self.message\_entry.grid(row=0, column=1)

self.message\_entry.bind("<Return>", self.send\_message)

self.send\_btn = tk.Button(message\_frame, text="发送", command=self.send\_message, state=tk.DISABLED)

self.send\_btn.grid(row=0, column=2, padx=5)

self.root.protocol("WM\_DELETE\_WINDOW", self.on\_closing)

def connect\_to\_server(self):

try:

host = self.host\_entry.get()

port = int(self.port\_entry.get())

if not host:

raise ValueError("请输入服务器地址")

if port < 1 or port > 65535:

raise ValueError("端口号必须在1-65535之间")

# 获取用户名

username = simpledialog.askstring("用户名", "请输入您的用户名:", parent=self.root)

if not username:

return

self.client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.client\_socket.connect((host, port))

# 发送用户名

self.client\_socket.send(username.encode('utf-8'))

# 接收欢迎消息

welcome\_msg = self.client\_socket.recv(1024).decode('utf-8')

self.display\_message(welcome\_msg)

self.running = True

self.connect\_btn.config(state=tk.DISABLED)

self.disconnect\_btn.config(state=tk.NORMAL)

self.send\_btn.config(state=tk.NORMAL)

self.host\_entry.config(state='readonly')

self.port\_entry.config(state='readonly')

# 启动接收消息的线程

receive\_thread = threading.Thread(target=self.receive\_messages, daemon=True)

receive\_thread.start()

except Exception as e:

messagebox.showerror("错误", f"连接服务器失败: {str(e)}")

if self.client\_socket:

self.client\_socket.close()

self.client\_socket = None

def disconnect\_from\_server(self):

if self.client\_socket:

self.running = False

try:

self.client\_socket.close()

except:

pass

finally:

self.client\_socket = None

self.display\_message("已断开与服务器的连接")

self.connect\_btn.config(state=tk.NORMAL)

self.disconnect\_btn.config(state=tk.DISABLED)

self.send\_btn.config(state=tk.DISABLED)

self.host\_entry.config(state='normal')

self.port\_entry.config(state='normal')

def receive\_messages(self):

while self.running:

try:

message = self.client\_socket.recv(1024).decode('utf-8')

if not message:

break

self.display\_message(message)

except Exception as e:

if self.running:

self.display\_message(f"与服务器通信时出错: {str(e)}")

break

self.disconnect\_from\_server()

def send\_message(self, event=None):

if not self.client\_socket:

return

message = self.message\_entry.get()

if not message:

return

try:

self.client\_socket.send(message.encode('utf-8'))

self.message\_entry.delete(0, tk.END)

except Exception as e:

self.display\_message(f"发送消息失败: {str(e)}")

self.disconnect\_from\_server()

def display\_message(self, message):

self.chat\_area.config(state='normal')

self.chat\_area.insert(tk.END, message + "\n")

self.chat\_area.config(state='disabled')

self.chat\_area.see(tk.END)

def on\_closing(self):

if self.client\_socket:

if messagebox.askokcancel("退出", "您已连接到服务器，确定要退出吗？"):

self.disconnect\_from\_server()

self.root.destroy()

else:

self.root.destroy()

def run(self):

self.root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

client = ChatClient()

client.run()