

Socket Programming

多人聊天室

資管四A

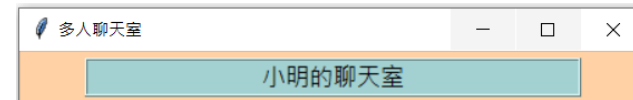
108403501

劉柔辰

功能簡單描述

Client端


- 多Client連接(Multithreading)
- 每位User進入聊天室前，可以輸入自己的姓名
- 進入後，聊天室視窗會顯示視窗名稱
- 每位User剛進入時，其他人的聊天室會顯示是誰進入
- 聊天過程中，User可清楚分辨他人訊息和自己的訊息
- 每位User離開時，其他人的聊天室會顯示是誰離開



Server端

- 可以知道client有誰連線及其addr(SERVER, PORT)
- 可以知道client有誰離開及其addr(SERVER, PORT)
- 只要有user加入，就會更新目前聊天室人數
- 可以知道誰在聊天室留言什麼

GUI介面_tkinter

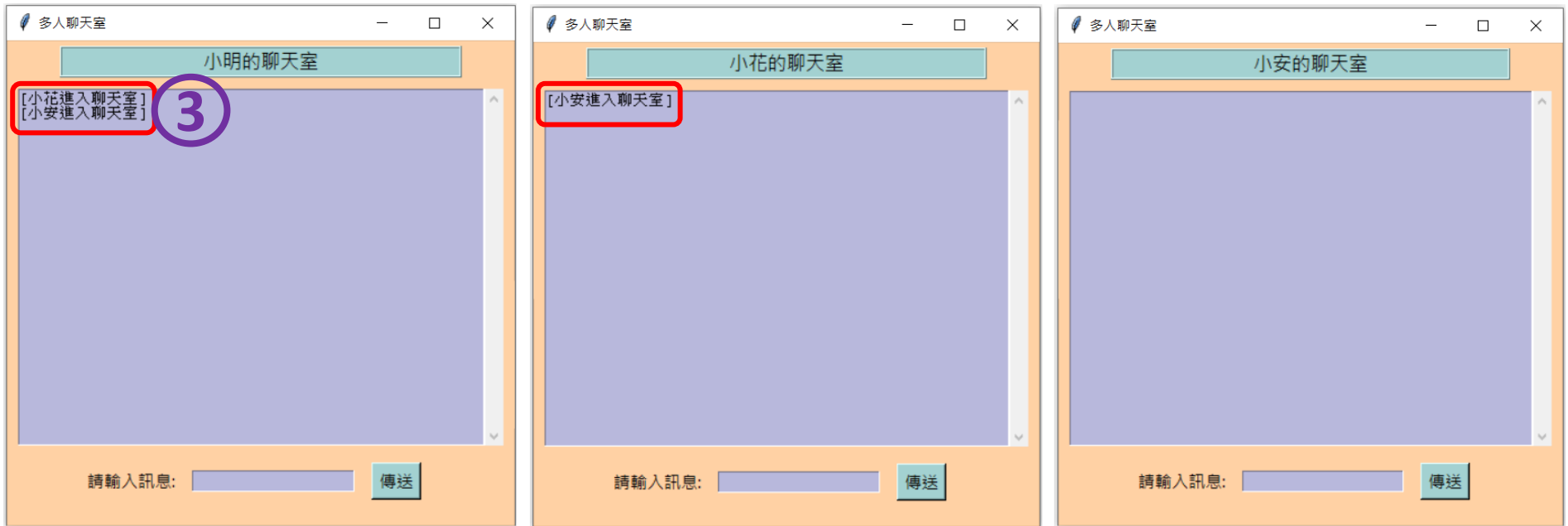
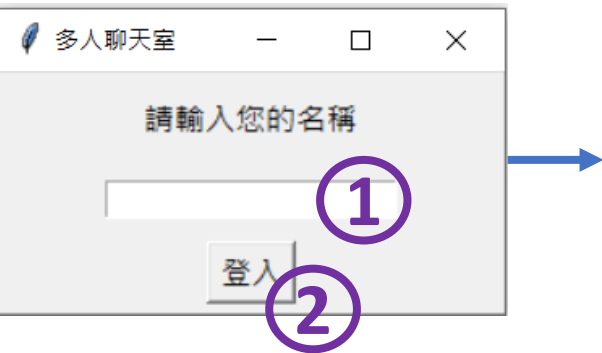
- background 顏色設計(#FFD1A4, #B8B8DC, #A3D1D1)
- Label 樣式設計(relief="ridge") 
- tkinter.scrolledtext.ScrolledText 卷軸文字圖形物件
 - wrap=tkinter.WORD
該行的尾端如果有單字跨行列印，則把此單字放到下一行顯示，方便閱讀
 - yview('end')
將閱讀焦點直接顯示在最下方的文字
- User 可以點選離開按鈕或輸入exit，以離開聊天室
- tkinter.messagebox.askokcancel
 - 幫使用者再次確認是否要離開聊天室(以防是不小心按到離開按鈕)

Client使用者流程&介面設計

Server對應的接收資訊

//User: 小明 小花 小安 //輸入名稱依序進入聊天室 //原使用者視窗會顯示新使用者進入的通知訊息

(Client)



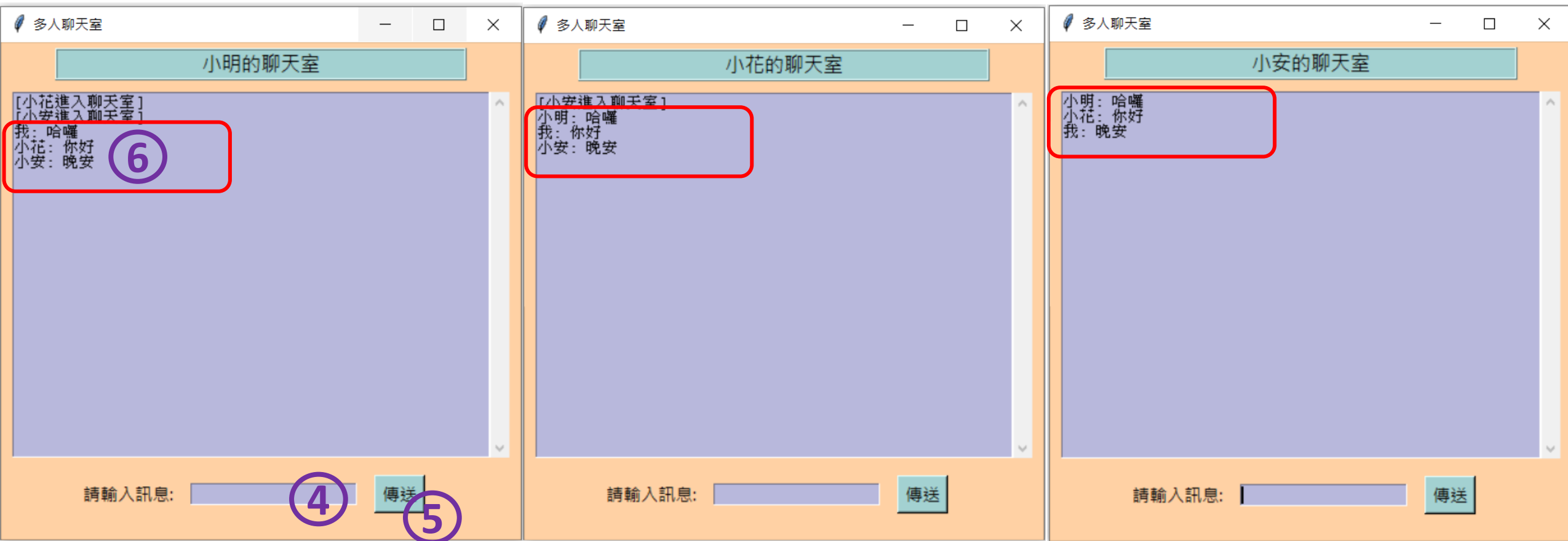
(Server)

//知道client有誰連線
 &其addr(SERVER, PORT)

//只要有User加入
 就會更新聊天室人數

```
PS C:\Users\AnnaLiu\python\socket> python server.py
[STARTING] Server is starting...
[LISTENING] Server is listening on 192.168.141.1
client_name = 小明
[NEW CONNECTION] ('192.168.141.1', 55504) connected.
聊天室人數： 1
[LISTENING] Server is listening on 192.168.141.1
client_name = 小花
[NEW CONNECTION] ('192.168.141.1', 55510) connected.
聊天室人數： 2
[LISTENING] Server is listening on 192.168.141.1
client_name = 小安
[NEW CONNECTION] ('192.168.141.1', 55520) connected.
聊天室人數： 3
[LISTENING] Server is listening on 192.168.141.1
```

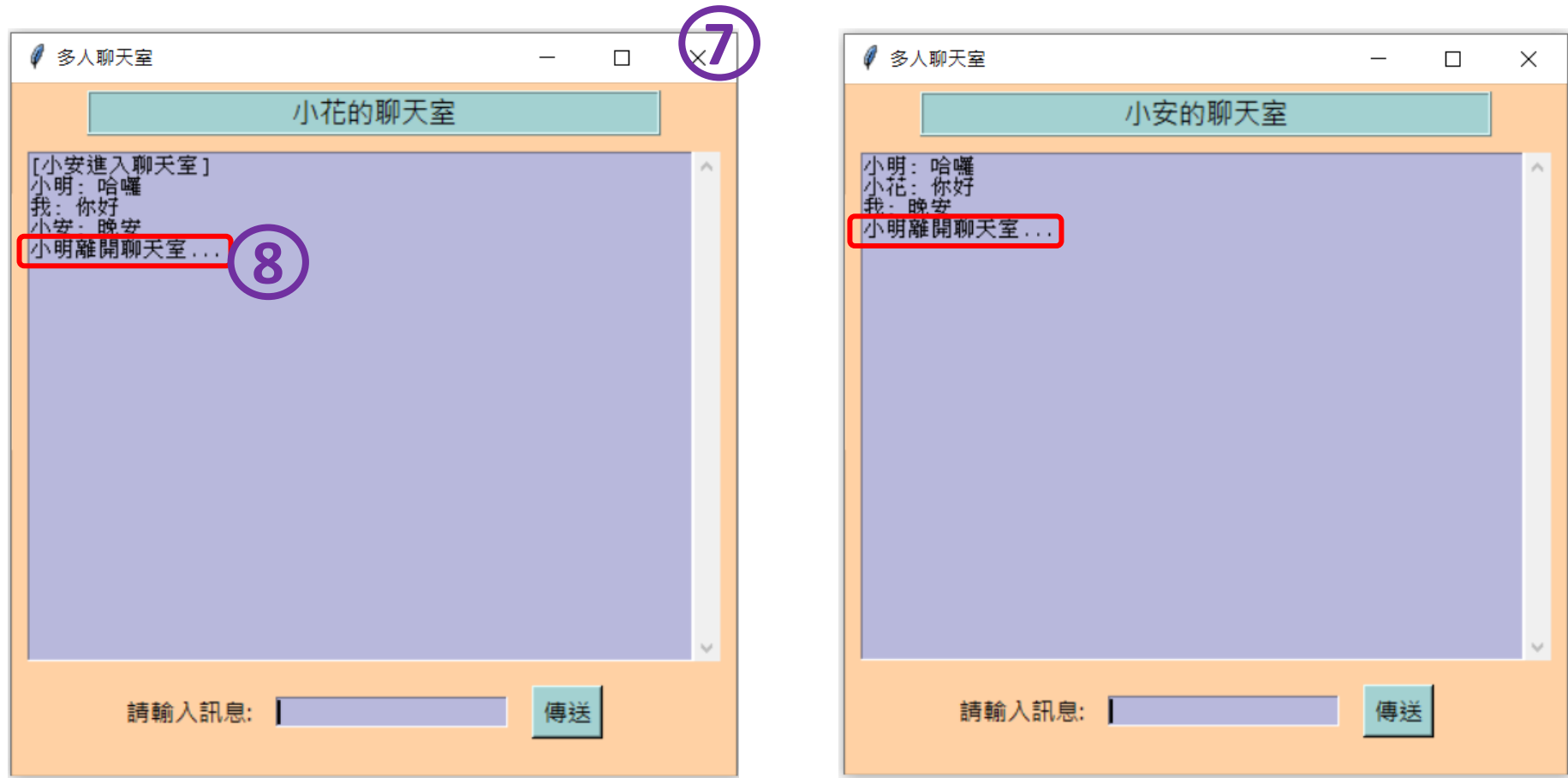
(Client) //User可清楚分辨他人訊息和自己的訊息



(Server) //知道誰在聊天室留言什麼

```
小明: 哈囉 from ('192.168.141.1', 55504)
小花: 你好 from ('192.168.141.1', 55510)
小安: 晚安 from ('192.168.141.1', 55520)
```


(Client) //每位User離開時，其他人的聊天室會顯示是誰離開



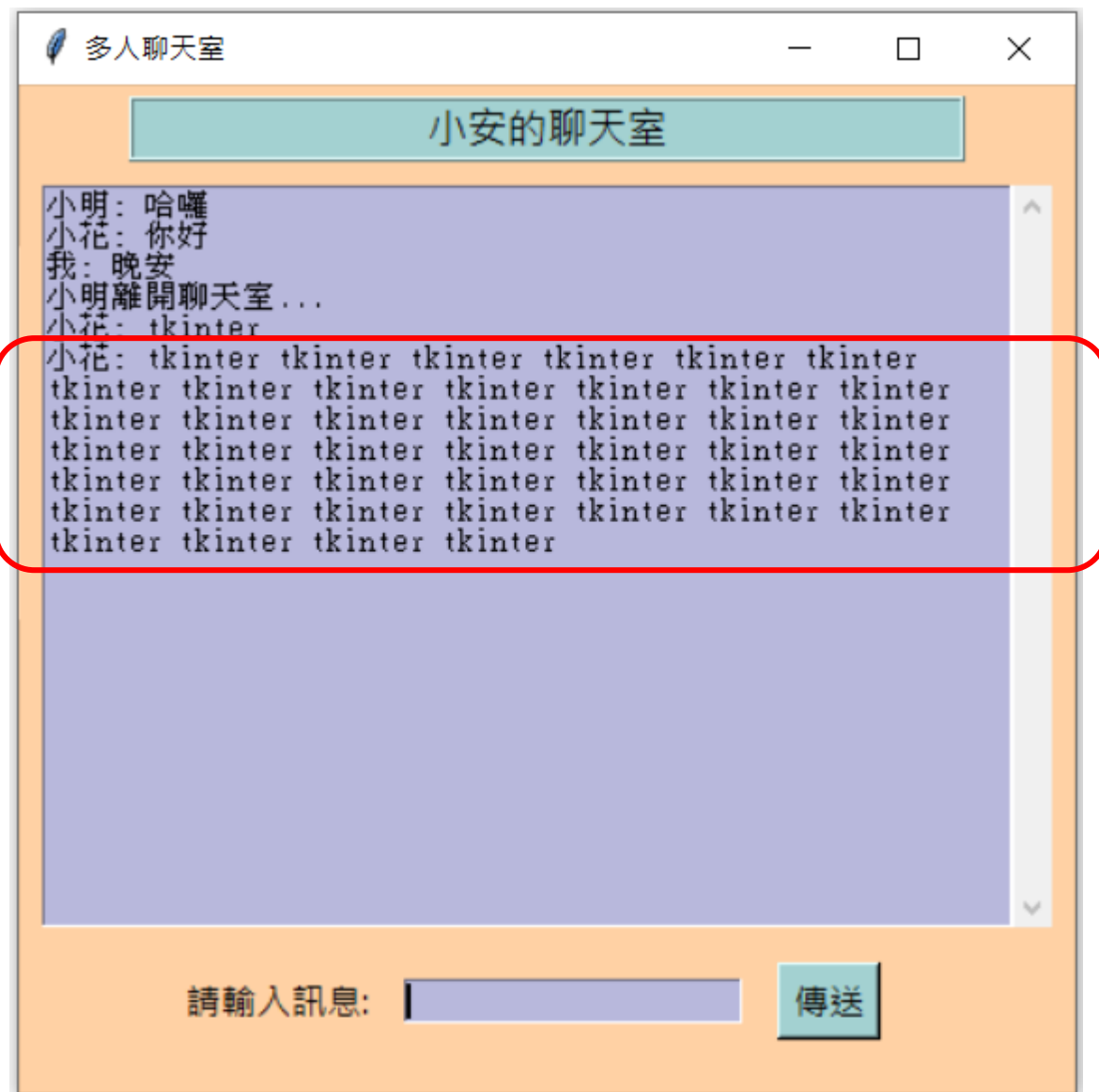
(Server) //知道client有誰連線 & 其addr(SERVER, PORT)

```
小明: 哈囉 from ('192.168.141.1', 55504)
小花: 你好 from ('192.168.141.1', 55510)
小安: 晚安 from ('192.168.141.1', 55520)
[CLOSED] ('192.168.141.1', 55504) closed.
```

//離開聊天室的再次確認視窗



//該行的尾端如果有單字跨行列印，則把此單字放到下一行顯示，方便閱讀



Code分配説明

(Client.py)

```
1  import socket
2  import sys
3  import tkinter as tk
4  import tkinter.messagebox
5  import tkinter.scrolledtext
6  import threading
7
8  SERVER = "140.115.140.103"
9  PORT = 7000
10 ADDR = (SERVER, PORT)
11 FORMAT = 'utf-8'
12
13 c = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # 創建socket對象(Ipv4,TCP宣告)
14 c.connect(ADDR) # 建立連接 # 客戶端指定要串接的IP位址跟Port號
```

```

16 # 傳送訊息給socket對象
17 def send(soc):
18     ... if string != "":
19         ... message = name + ": " + string # 誰(
20         ... data = message.encode(FORMAT)
21         ... soc.send(data) # 傳送資料過去給串接對
22         ... if string.lower() == "EXIT".lower():
23             ... exit()

```

傳送訊息給socket對象

```

24
25 # 將訊息插入ScrolledText聊天室框框內
26 def chatblock(s):
27     ... output_area.config(state='normal')
28     ... output_area.insert('end', s) # 插入
29     ... output_area.yview('end') # foc
30     ... output_area.config(state='disabled')

```

將接收與送出的訊息插入
聊天室視窗上方ScrolledText框框內
(插入後，設定使用者無法編輯state='disabled')

```

31
32 # 接受訊息(threading)
33 def recv(soc):
34     ... soc.send(name.encode(FORMAT))
35     ... while True:
36         ... data = soc.recv(1024) # 接收資料
37         ... s = data.decode(FORMAT) + '\n'
38         ... chatblock(s) # 將訊息插入ScrolledTe

```

接收訊息

```

39
40 # 將自己傳送的訊息插入到聊天室框框內，且傳送給sock
41 def myMsg():
42     ... global string # 宣告函數定義外的全域變數(
43     ... string = input_area.get() # 字串變數，取得
44     ... send(c) # 傳送資料過去給串接對象(se
45     ... if string != "":
46         ... s = '我: ' + string + '\n'
47         ... chatblock(s) # 將訊息插入ScrolledTe
48         ... input_area.set('') # 輸入傳送訊息的

```

傳送訊息到聊天室框框內

建立聊天室視窗

```
50
51 def create():
52     ... global name
53     ... name = user_name.get()
54
55     ... # 接收進程 # Threading
56     ... tr = threading.Thread(target=recv, args=(c,), daemon=True) # recv(sock) # daemon=True 表示創
57     ... tr.start()
58
59     ... # 關閉登入的視窗
60     ... enter_label.destroy()
61     ... enter_name.destroy() # 關閉登入視窗
62     ... login_button.destroy()
63     ... # 建立聊天室視窗
64     ... window.title("多人聊天室")
65     ... window.configure(bg='#FFD1A4') # 建立聊天室視窗
66     ... window.geometry("450x430+500+150")
67
68     ... # 聊天室名稱之Label # 設定聊天室名稱之Label
69     ... roomname = tk.Label(window, text="%s的聊天室" % name, width=35, bg='#A3D1D1', relief="ridge")
70     ... roomname.config(font=("微軟正黑體", 13))
71     ... roomname.pack(pady=5)
72
73     ... # 上方聊天室框框ScrolledText
74     ... output_area.pack(padx=10, pady=5)
75     ... output_area.config(state='disabled') # 設定聊天室視窗上方框框ScrolledText
```

```
77 ... # 建立聊天室視窗下方傳送文字的框架
78 ... frame = tk.Frame(window, bg='#FFD1A4')
79 ... # 請輸入訊息之Label # 請輸入訊息之Label
80 ... input_label = tk.Label(frame, text="請輸入訊息:", anchor='w', width=10, bg='#FFD1A4')
81 ... input_label.config(font=("微軟正黑體", 11))
82 ... input_label.pack(side=tk.LEFT) # pack: 採用塊的方式組織配件 # side=tk.LEFT: 靠左邊
83 ... # User輸入想傳送的文字之Entry
84 ... input_entry = tk.Entry(frame, bg='#B8B8DC', textvariable=input_area) # User輸入想傳送的文字之Entry
85 ... input_entry.pack(side=tk.LEFT) # pack: 採用塊的方式組織配件 # side=tk.LEFT: 靠左邊
86 ... # 傳送訊息之Button
87 ... input_button = tk.Button(frame, text="傳送", command=myMsg, bg='#A3D1D1') # 執行myMsg
88 ... input_button.config(font=("微軟正黑體", 11)) # 傳送訊息之Button
89 ... input_button.pack(side=tk.LEFT, padx=15) # pack: 採用塊的方式組織配件 # side=tk.LEFT: 靠左邊
90 ...
91 ... frame.pack(anchor='center', pady=10) # 框架位置設定 # center: 在整個視窗(window)向下pack的
```

```
93 # 再次確認是否關閉聊天室視窗
94 def quit():
95     msgBox = tkinter.messagebox.askokcancel("剛才點擊了關閉按鈕", "確定要離開嗎?")
96     if msgBox == True: # 確定關閉視窗
97         c.send('close'.encode(FORMAT)) # 傳送關閉視窗的通知給socket對象
98         exit(0) # 關閉視窗
```



```

100 # 宣告tkinter視窗父容器
101 window = tk.Tk() ... #宣告
102 window.title("多人聊天室") ... # 父容器標題
103 window.geometry("250x120+500+250") ... # 設置父容器視窗初始大小，如果沒有這個設
104
105 # 建立User登入視窗
106
107 enter_label = tk.Label(window, text="請輸入您的名稱", width=40, height=2)
108 enter_label.config(font=("微軟正黑體", 11))
109 enter_label.pack(pady=2) ... # pack: 採用塊的方式組織配件 # 加入版面時預設會由
110 # 輸入名字之Entry
111 user_name = tk.StringVar() ... # 字串參數
112 enter_name = tk.Entry(window, width=20, textvariable=user_name) ... # textva
113 name = user_name.get() ... # 字串變數 取得文字內容
114 enter_name.pack(pady=6)
115 # 登入之Button
116 login_button = tk.Button(window, text="登入", command=create) ... # 執行cre
117 login_button.config(font=("微軟正黑體", 11))
118 login_button.pack(padx=20, pady=5)
119
120 print("Client connect to server")

```

請輸入名稱之Label

輸入名字之Entry

登入之Button

```
122 # 聊天室視窗
123 # input_area: 輸入傳送訊息的文字框 的文字
124 input_area = tk.StringVar()
125 # 聊天室框框scrolledtext 卷軸文字圖形物件
126 output_area = tkinter.scrolledtext.ScrolledText(window, bg='#B8B8DC', wrap=tk.WORD)
127
128 window.protocol("WM_DELETE_WINDOW", quit) # 定義當用戶使用窗口管理器顯式關閉窗口時發生的情況 # 執行quit()
129 window.mainloop() ... # 自動刷新畫面
130
131 c.close() ... # 關閉Socket連接，並釋放所有相關資源
```

(Server.py)

```
1  import socket
2  import threading
3
4  SERVER = socket.gethostbyname(socket.gethostname())
5  PORT = 7000 # Port 範圍介於1024~65535，其中0~1023為系統保留不可使用
6  ADDR = (SERVER, PORT)
7  FORMAT = 'utf-8'
8
9  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) ... # 創建socket對象(Ipv4,TCP宣告)
10 s.bind(ADDR) ... # 綁定地址和埠
11 s.listen() ... # 伺服器端監聽socket串接
```

```

13 def handle_client(conn, addr): # 等待接收client端訊息存放在2個變數conn和addr裡
14     # 如果addr不在user字典裡
15     if not addr in user:
16         print(f"[NEW CONNECTION] {addr} connected.") # 新的user連線成功
17
18         for scs in ser_cli_soc: # 從user依序取出address(key:addr)
19             ser_cli_soc[scs].send("[".encode(FORMAT) + client_name + "進入聊天室"].encode(FORMAT))
20             user[addr] = client_name.decode(FORMAT) # client_name是最新進入聊天室的client，解壓後放入user
21             ser_cli_soc[addr] = conn # 將伺服器與伺服器埠號為addr的socket對象(conn)放入字典ser_cli_soc裡
22
23     # Start to chat
24     while True:
25         msg = conn.recv(1024) # 接受client端訊息 from the client
26         # 如果EXIT在發送的数据(msg)裡，或user點選關閉視窗按鈕
27         if (('EXIT'.lower() in msg.decode(FORMAT)) | (msg.decode(FORMAT) == 'close')):
28             name = user[addr] # 變數name值為user字典addr鍵(key)對應的值(client_name)
29             user.pop(addr) # 刪除離開聊天室的user(client)
30             ser_cli_soc.pop(addr) # 刪除離開聊天室的user的addr
31
32             for scs in ser_cli_soc: # 從user依序取出address(key:addr)
33                 ser_cli_soc[scs].send((name + "離開聊天室...").encode(FORMAT)) # 依序發送data(client_name)
34                 print(f"[CLOSED] {addr} closed.") # 離開聊天室的user(client)的addr(SERVER, PORT)
35
36             global num # 宣告函數定義外的全域變數(global variable)，使該全域變數可以在函數中進行處理
37             num = num - 1 # 聊天室人數-1
38             break
39         else:
40             print(f"{msg.decode(FORMAT)} from {addr}") # 哪位user傳了什麼訊息到聊天室
41             for scs in ser_cli_soc: # 從user依序取出address(key:addr)
42                 if ser_cli_soc[scs] != conn: # 若address不等於目前傳送訊息的user的地址
43                     ser_cli_soc[scs].send(msg) # 就發送data(msg)到client端聊天室

```

新user加入

user離開聊天室

開始聊天

傳送訊息
給其它user

```

45 num = 0 # 聊天室人數
46 user = {} # 存放字典{addr:name}
47 ser_cli_soc = {} # 存放{socket:不同線程(thread)的socket對象}
48
49 print("[STARTING] Server is starting...")
50
51 while True: # user加入
52     try:
53         print(f"[LISTENING] Server is listening on {SERVER}")
54         conn, addr = s.accept() # conn存串接對象、addr存連線資訊 # 等待接收cl
55     except ConnectionResetError: # 掛斷時會報錯
56         print("Someone left unexcept.")
57
58     client_name = conn.recv(1024) # receive data(client name) from the client
59     if client_name.decode() == 'close': # user沒有登入，直接關閉視窗
60         print(addr, "關閉了登入視窗...")
61         continue
62     print("client_name =", client_name.decode()) # 新user的name # decode c
63     num = num + 1 # 聊天室人數+1
64
65     # 為server分配線程
66     r = threading.Thread(target=handle_client, args=(conn, addr), daemon=True)
67     r.start()
68
69     print("聊天室人數:", num) # 當新user加入聊天室，更新聊天室人數

```

server接受socket串接

user關閉登入視窗

user進入聊天室

分配Thread