Computer Programming Fall 2018
#Final Project (Group 10)
Python Mini Chat Room

Student:
吳禎軒 B07901026
林予康 B07901027
蘇磐洋 B07901093

Motivation:

A lot of other groups choose to design games. We choose to design a chat room since we think it's more useful in our daily life, and we want to design a program that involves communication with other people, not just a one-person program.

Also, we want to learn how line and chat room work. We have a desire to learn new things while working on our final project. Though we know that it will be a hard work, we march forward bravely.

User guide:

Enter your computer's IP and port in the code
Make sure the server and the clients(no limited number) use the same internet
The server will request you to enter your name and let everyone else in the chat room know
Press 'Enter' or 'Send' Button to send your message

Other Functions:

- Thumb up: randomly generate positive comments

- Encode and Decode:

encode:After striping the punctuation and lower the alphabets, encode your message with '0' and '1' with designed Huffman code (26 alphabets, 0-9, spaces are valid inputs)

decode:Convert the code into alphabets after entering the correct code (default : {0814})

☌ Private talk:

Private talk:You can choose a user and have private talk with him.Remember that you should follow the instructions that server provides you.ex: you should type {private} to get into private talk and {over} to quit and if you agree to talk with the user that tries to connect with you privately, please type {Yes} (the user's name) with a space between two words.

☌ Send files and photos:

send files:You can choose a file in your computer, and send it to all the other people in the chat room. However, you can only send txt.
send photos:You can choose a photo in your computer, and send it to all the other people in the chatroom. However, the size of the photo can't be too large.

☌ Calculator:

You can type an equation, and press 'OK'. Then the program will show the answer. This function isn't related to chat room, which means nobody else will see what you have calculated.

☊Some difficult codes(server):

☌ Private talk:

The following code is to handle the user who wants to have private talk with other users.

```
elif msg == bytes("{private}", "utf8"):
    client.send(bytes("(Who do you want to connect with?)", "utf8"))
    msg = client.recv(BUFSIZ)
    for sock in clients:
        flag=1
        if clients[sock][0]==msg:
            flag=0
            private[client]=1
            msg="(%s connect with you privately and please type({Yes} his/hername))" %clients[client][1]
            sock.send(bytes(msg, "utf8"))
            client.send(bytes("(If you want to quit private talk,please type {over})", "utf8"))
            while True:
                flag2=1
                private_clients={}
                msg = client.recv(BUFSIZ)
                if private[sock]==0:
                    client.send(bytes("(server:the user doesn't want to connect with you!)", "utf8"))
                    break
                if msg==bytes("{over}", "utf8"):
                    private[client]=0
                    break
                private_clients[sock]=clients[sock]
                private_clients[client]=clients[client]
                broadcast(msg,flag2,private_clients,client,str(clients[client][1])+": ")
    if flag==1:
        client.send(bytes("(The user doesn't exit!)", "utf8"))
```

First, after you type{private}, you will enter the “private” status. Then enter the name of the user that you want to have private talk with. The code will go into a for loop to search the user’s socket in “clients” dictionary in order to connect the current thread with the user’s socket. However, if you don’t enter the user’s name appropriately, the value of “flag” won’t change, which will lead to the result that you get the message “the user doesn’t exist ”.

In addition, there’s a dictionary called “private” that can share information with other threads. Therefore, once you go into {private} status, the value in this dictionary will change, which means you are currently in private status. The reason for doing this is because the other users’ thread should know who are in {private} status too so that his thread knows who it should send message and who shouldn’t.

Also, there’s a dictionary called “private_clients” to record the sockets that are in “private” status.

Finally, if you or the other leave the “private” status, the value in “private” dictionary will change again so that you know the fact that the user doesn’t want to talk with you privately.

The following code is to handle the situation when other user wants

to have private talk with you, how can you reject or accept him/her.

```
elif msg[:5] == bytes("{Yes}", "utf8"):
    private[client]=1
    sock_name=msg[6:]
    for sock in clients:
        if clients[sock][0]==sock_name:
            while True:
                msg = client.recv(BUFSIZ)
                flag2=1
                private_clients={}
                if private[sock]==0:
                    client.send(bytes("(server:the user doesn't want to connect with you!)", "utf8"))
                    break
                if msg==bytes("{over}", "utf8"):
                    private[client]=0
                    break
                print(sock)
                private_clients[sock]=clients[sock]
                private_clients[client]=clients[client]
                broadcast(msg,flag2,private_clients,client,str(clients[client][1])+": ")
```

If you enter {Yes} (user's name), you will go into the above code. Then, the code gets the user's name too, so that it can find the corresponding user's socket. As for the remaining part, it is the same as the part that deals with the user who sends the "private" invitation.

♂ Send_file and send_photo:

```
elif msg[:10] == bytes('[===] 100%','utf8'):
    flag2=0
    print(msg)
    sleep(0.1)
    broadcast(msg,flag2,clients,client)
elif msg[:5] == bytes("#####", "utf8") or flag2==2:
    flag2=2
    broadcast(msg,flag2,clients,client)
elif msg[:5] == bytes("?????", "utf8") or flag2==2:
    flag2=2
    broadcast(msg,flag2,clients,client)
```

When the server receives '#####', it means one of the client has sent a file. When the server receives '?????', it means one of the client has sent a photo. When the server receives '[===] 100%', it means the sending of the file or photo has finished.

We use flag2 to deal with different types of sending of messages.

Some difficult codes(client):

Send_file:

```
def send_file(event=None):
    root = tkinter.Tk()
    root.withdraw()

    file_path = filedialog.askopenfilename()
    #print(file_path)

    f = open (file_path, "rb")
    l = f.read(1024)
    client_socket.send(bytes('#####', "utf8"))
    sleep(1)
    while (l):
        client_socket.send(l)
        l = f.read(1024)
    sleep(1)
    client_socket.send(bytes('[===] 100%', "utf8"))
```

The first line opens a window, and the second line closes the window. The third line will open your computer's file dialog, and file_path will record what you have clicked. If you are not sure about what you have open, you can print file_path.

Then, the program will open the file you want to send. '#####' is like a signal, telling the server that the client is now sending a file. '[===] 100%' is another signal that tells the server that the file sending has finished.

Here, we use sleep to separate the sending of the signals and the sending of the file content. In other words, it makes sure the signals are properly sent, not to be mixed with other messages.

As for file's name, we import datetime and name the file in terms of the time the user creates the file, so that the files can be distinguished.

♁ send_file:

The concept is like send_file, except that the starting signal is '?????' rather than '#####'.

```
def send_photo(event=None):
    root = tkinter.Tk()
    root.withdraw()

    photo_path = filedialog.askopenfilename()

    f = open (photo_path, "rb")
    l = f.read(1024)
    client_socket.send(bytes('?????', "utf8"))
    sleep(1)
    while (l):
        client_socket.send(l)
        l = f.read(1024)
    sleep(1)
    client_socket.send(bytes('[===] 100%', "utf8"))
```

♁ Calculator:

```
def calculator():
    root = Tk()
    root.withdraw()
    d = number(root)
    root.wait_window(d.top)
```

The first line opens a window, and the second line closes the window. The third line calls number, a class.

```
class number:

    def __init__(self, parent):

        top = self.top = Toplevel(parent)

        Label(top, text="please enter an equation").pack()
        self.e = Entry(top)
        self.e.pack(padx=10)

        b = Button(top, text="OK", command=self.ok)
        b.pack(pady=10)
```

Parent is the window we have opened. Then another window jumps out, and you can enter the equation. After entering the equation, press 'OK'.

```
def ok(self):
    c=self.e.get()
    print( "the value is",c)
    self.top.destroy()
    answer=Tk()
    try:
        Button(answer, text="the answer is:%s"%eval(c)).pack()
    except:
        Button(answer, text="you use the calculator in a wrong way").pack()
```

Self.top.destroy() will close the equation window. Then another window with the answer jumps out. We use eval() to calculate the equation, and we also deal with exceptions.

Other module required to setup: None

Modules we have used:

Socket: enable message transfer and connection between server and client.
Thread: create many threads as servers to serve their clients.
Tkinter: the interface of the chat room.
Datetime: current time is displayed after every message.
Random: used in Thumb up.

Time: enable the code to pause so that the message can be sent appropriately.
Vpython: to create welcome page of our chat room.

Obstacles we have encountered:

1. We used NTU_peap for internet connection and failed, attempting to disable the Firewall of the laptops.
2. Handle exception situations of private talk.
3. The part that binds our image handling code and file handling code with tkinter.
4. At first, we have a hard time understanding tkinter and socket

Review:

This project is not a simple task. We are like explorers, fumbling in a forest of unknown. Tkinter, Socket, Thread, etc. are unfamiliar to us, and are difficult for python beginners to understand. Therefore, we search a lot of information, and ask a lot of people.

After a long and circuitous road, we finally finish our final project. We have to frankly confess that the interface is not beautiful and eye-catching, and there are still a lot of bugs if you use the python miniline in a wrong way. However, we can proudly say that we have accomplished a lot of missions we previously thought impossible, like private talk, sending and receiving file and photo, and group chat.

Though we sometimes tumble, sometimes detour, we have learned and made progress along the road. Never give up. Keep on going.

Work Division:

蘇磐洋:照片的接收與發送(與 tkinter 結合)、檔案的接收與發送(研究 filedialog、與 tkinter 結合)、小組程式整合、Tkinter 介面處理、計算機製作、vpython 頁面製作、上臺報告與回答問題

吳禎軒: Tkinter 介面處理、小組程式整合、時間顯示、按鈕處理、按讚功能、encode+decode 功能、字母頻率計算與 huffman code 編碼設計、topic+final ppt 製作

林予康: 檔案的接收與發送(與 tkinter 結合)、照片的接收與發送(與 tkinter 結合)、指定私人訊息(private talk)的傳收、小組程式的最後整合、研究 socket 和 thread module、上臺報告與回答問題

Reference:

1.https://github.com/lunemec/python-chat?fbclid=IwAR3qNCH4bItfX-DNGKGuUPJQ843Uzp6LAvPvpPRuVIkEe8hZ_n5Kooi1isk

2. https://en.wikipedia.org/wiki/Huffman_coding

3.https://realpython.com/python-sockets/?fbclid=IwAR1D14LTwejkSVEOGVrfg4Fswe3zsI9Ee0N0UY7TxxQxmS9UdGBT92kIXnc

4.https://docs.python.org/2/library/tkinter.html

5. https://docs.python.org/3/library/datetime.html

6. https://www.python-course.eu/threads.php

7.https://medium.com/swlh/lets-write-a-chat-app-in-python-f6783a9ac170

8.https://stackoverflow.com/questions/9382045/send-a-file-through-sockets-in-python

9.https://stackoverflow.com/questions/6289474/working-with-utf-8-encoding-in-python-source

10.https://medium.com/swlh/lets-write-a-chat-app-in-python-f6783a9ac170

11.https://docs.python.org/3/library/socket.html

12.https://hk.saowen.com/a/91e710867237885420ddc11405f824

49cd5733d50508418faf8f2690d7a8b772
13.https://keelii.com/2018/09/24/socket-programming-in-python/
14.https://pythonspot.com/tk-file-dialogs/
15.https://stackoverflow.com/questions/9239514/filedialog-tkinter-and-opening-files
16. https://stackoverflow.com/questions/9319317/quick-and-easy-file-dialog-in-python
17. https://blog.gtwang.org/programming/opencv-basic-image-read-and-write-tutorial/
18. https://stackoverflow.com/questions/16242782/change-words-on-tkinter-messagebox-buttons