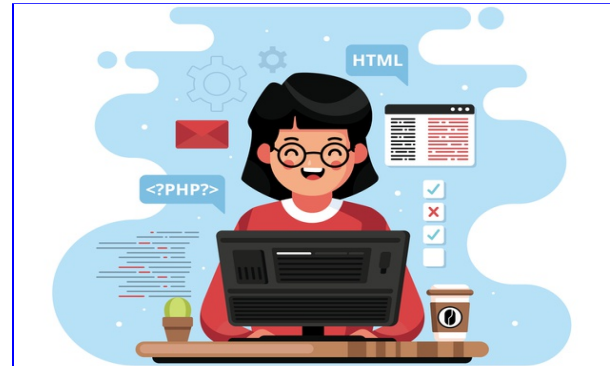


GUI BASED CHAT APP



What is our GOAL for this CLASS?

In this class, we have applied our knowledge to add a GUI to the chat app we built earlier using Tkinter. We break the app down into two parts, the login window and the chat interface and work on the chat window using Object Oriented Programming.

What did we ACHIEVE in the class TODAY?

- Added a GUI to our chat app
- Built the chat window for users

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Tkinter
- Object Oriented Programming
- **self** keyword

How did we DO the activities?

In the last class, we learnt about Tkinter and how it's used to create GUI based applications for desktop. Today, we added the chat window to our chat app and will check the functionality.

Activity:

1. In the code that we completed in C-202, open **client.py** and start making a chat window.
2. Make a function def layout. Pass the two arguments self,name -
3. In Tkinter, we can create a main window with the **Tk()** class, or we can create a top level window for secondary things with the **Toplevel()** class which we created in the last session
 - a. Since our main screen is the chat interface, we will keep the **self.Window** that is a **Tk()** object for the chat interface.
 - b. We will use the **deiconify()** function on the **Tk()** object. The **deiconify ()** method will raise this chat window and give it the focus after the login window.
 - c. Set the Title for the chat window.
 - d. Set the properties for the chat screen with the **resizable()** and **configure()** functions.

```
def layout(self,name):  
  
    self.name = name  
    self.Window.deiconify()  
    self.Window.title("CHATROOM")  
    self.Window.resizable(width = False,  
                           height = False)  
    self.Window.configure(width = 470,  
                           height = 550,  
                           bg = "#17202A")
```

4. We will also create a label with the **Label()** class to display the name of the client on the chat window.

```
self.labelHead = Label(self.Window,  
                        bg = "#17202A",  
                        fg = "#EAECEE",  
                        text = self.name ,  
                        font = "Helvetica 13 bold",  
                        pady = 5)
```

5. Use the **place()** function on the label to place it on the chat window -

```
self.labelHead.place(relwidth = 1)
```

6. Create a line under the labelname field on the chat window and place it on the screen -

```
self.line = Label(self.Window,  
                  width = 450,  
                  bg = "#ABB2B9")  
  
self.line.place(relwidth = 1,  
                rely = 0.07,  
                relheight = 0.012)
```

7. Add a **Text()** widget below to the Line label so that the user can see the group chat area in the chat window.

```
self.textCons = Text(self.Window,  
    width = 20,  
    height = 2,  
    bg = "#17202A",  
    fg = "#EAECEE",  
    font = "Helvetica 14",  
    padx = 5,  
    pady = 5)  
  
self.textCons.place(relheight = 0.745,  
    relwidth = 1,  
    rely = 0.08)
```

8. Create a Label for Bottom part of the chat window, and place it on the chat screen -

```
self.labelBottom = Label(self.Window,  
    bg = "#ABB2B9",  
    height = 80)  
  
self.labelBottom.place(relwidth = 1,  
    rely = 0.825)
```

9. Use an **Entry()** widget to take input from the user that will add in the label Bottom. , and place it on the chat screen. Make sure to use a **focus()** function on it, so that as soon as the chat window opens, it will keep the input field selected so the user can directly send their message.

```
self.entryMsg = Entry(self.labelBottom,  
                        bg = "#2C3E50",  
                        fg = "#EAECEE",  
                        font = "Helvetica 13")  
  
self.entryMsg.place(relwidth = 0.74,  
                    relheight = 0.06,  
                    rely = 0.008,  
                    relx = 0.011)  
  
self.entryMsg.focus()
```

10. Create a send button using **Button()** and place it on the screen.

```
self.buttonMsg = Button(self.labelBottom,  
                        text = "Send",  
                        font = "Helvetica 10 bold",  
                        width = 20,  
                        bg = "#ABB2B9",  
                        command = lambda: self.sendButton(self.entryMsg.get()))  
  
self.buttonMsg.place(relx = 0.77,  
                    rely = 0.008,  
                    relheight = 0.06,  
                    relwidth = 0.22)
```

11. Add a cursor in the entry() widget to show the text position in the chat text area using **config()** command.

```
self.textCons.config(cursor = "arrow")
```

12. Create a scroll bar to move the text screen area up and down using **scrollbar()** widget() and then place it using **place()** command

```
scrollbar = Scrollbar(self.textCons)

scrollbar.place(relheight = 1,
               |   |   |   |   |
               relx = 0.974)
```

13. Create the **send button()** function in the class. It takes the **msg** as an argument, along with the keyword **self**, since this function belongs to the class. Save the message into a variable called **self.msg**. Take the data from starting to end. We can start writing messages, so create a thread that starts the **self.write()** function.

```
def sendButton(self, msg):
    self.textCons.config(state = DISABLED)
    self.msg=msg
    self.entryMsg.delete(0, END)
    snd= Thread(target = self.write)
    snd.start()
```

14. Create the **showmessage()** function in the class. It takes one as an argument, along with the keyword **self**, since this function belongs to the class. Insert all the data inside text area using **insert()** method. Make text area in disabled state while entering data.

```
def show_message(self, message):
    self.textCons.config(state = NORMAL)
    self.textCons.insert(END, message+"\n\n")
    self.textCons.config(state = DISABLED)
    self.textCons.see(END)
```

15. Create the **write()** function in the class. It takes the **message** as an argument, along with the keyword **self**, since this function belongs to the class. Disable will freeze the user text widget means it's not clickable. Send the client message using **send()** function and encode the message in "UTF-8" format and then call the **showmessage()** function with an argument i.e message from the client.

```
def write(self):
    self.textCons.config(state=DISABLED)
    while True:
        message = (f"{self.name}: {self.msg}")
        client.send(message.encode('utf-8'))
        self.show_message(message)
        break
```

16. Go to receive function which we created in last class, just we need to change else statement, as we will call send_message function
Earlier it was like this :

```
try:
    message = client.recv(2048).decode('utf-8')
    if message == 'NICKNAME':
        client.send(self.name.encode('utf-8'))
    else:
        pass
except:
    print("An error occurred!")
    client.close()
    break
```

Now it will look like this :

```
def receive(self):  
    while True:  
        try:  
            message = client.recv(2048).decode('utf-8')  
            if message == 'NICKNAME':  
                client.send(self.name.encode('utf-8'))  
            else:  
                self.show_message(message)  
        except:  
            print("An error occurred!")  
            client.close()  
            break
```

17. Run the **server.py** and the **client.py** in separate command prompts/terminals -

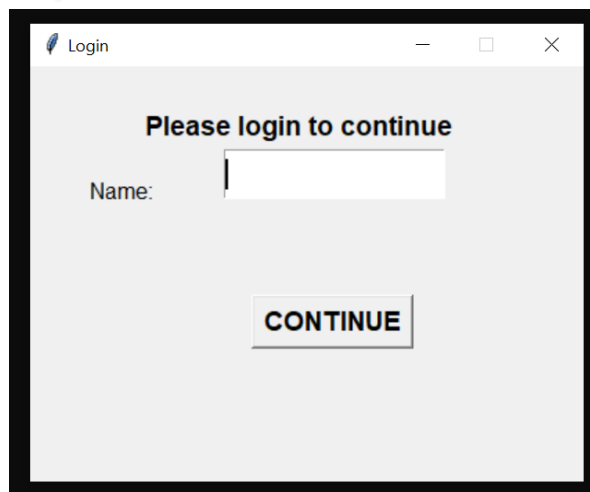
Server.py

Server has started...

client.py

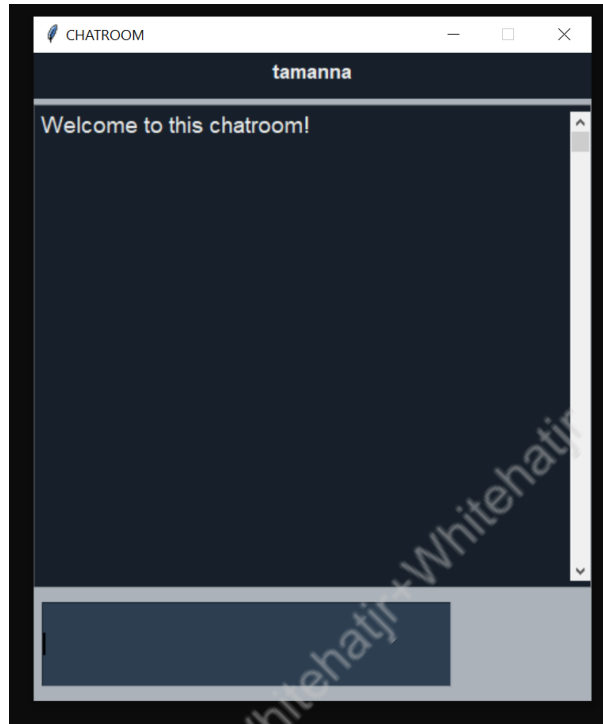
Connected with the server...

18. The login window opens up and looks like -



A screenshot of a web browser window titled "Login". The window has a light gray background. At the top, it says "Please login to continue". Below this, there is a label "Name:" followed by a text input field. At the bottom of the window, there is a button labeled "CONTINUE".

19. The chat window opens up and looks like -



What's NEXT?

In the next class, we will complete the chat interface of this app and have a fully functional GUI based chat applications

Expand Your Knowledge:

Explore more about Tkinter's library [here](#)