

# Smashing the Roadblock in Chat System Project

## Socket/connection related

[Address already in use](#)

[Node name nor server name provided \(Windows OS\)](#)

[Open terminal window at a certain folder](#)

## Running Program from terminal

[faster way to open terminal window at a specific folder/directory](#)

[Type error: "super\(\)" takes at least one argument \(0 given\) \(chat\\_server.py\)](#)

[FileNotFoundError: \[Error 2\] No such file or directory: "AllSonnets.txt.idx" \(chat\\_server.py\)](#)

## Sound/Music/Voice related

[Error using SpeechRecognition library](#)

## GUI

Why threading

[How to get away with not learning how to thread for GUI](#)

[One easy way to construct input and output without threading: \(jy2363\)](#)

How to keep texts aligned on the left on tkinter.label

[Modifying GUI get\\_message](#)

[Destroying the previous page](#)

[Passing message from GUI to the server](#)

[Connecting Two User Interface together using PyQt5](#)

[Choosing the main thread using PyQt5](#)

[Sending Images from the GUI to the server using Tkinter](#)

[Changing the background image of GUI or login system](#)

[Updating the information on the Label in Tkinter](#)

[Embed GUI and Pygame into Chat System](#)

How travel between GUI windows with PyQt5 while keeping the same socket

Gaming (pygame)

[Pygame window not opening/closing](#)

[Pygame window all black](#)

[How to quit Pygame window without quitting python](#)

[Code a game which contains various phases](#)

[Pygame quit in windows system and macOS system](#)

[Handle characters who can have various states in a game](#)

[Check collision and update player's position in a game](#)

**Encryption related:**

["OverflowError: Python int too large to convert to C long"](#)

[Sending keys properly](#)

[Error complaining json cannot parse binary message when using AES for encryption](#)

[Error: "Non-hexadecimal digit found"](#)

[Search feature after encryption](#)

[Error 8: Node name nor server name provided \(Windows OS\)](#)

**File Transfer(lc3913)**

Unable to send binary data through json.dumps()

Errno 35 in macOS when receiving binary data from client

Sending file through JSON:

**[User Input From Console](#)****[Weird Dictionary Issue](#)**

Address already in use

Solution 1: in chat\_utils.py, change CHAT\_PORT = 1112 to a different number, i.e. 3356 and run the server again.

Solution 2: kill the process. in terminal window, run the following command to check python process that are running: `ps -ef|grep python`

```
JoyceNYUSH1743LP-MX:ICS_chat_joyce ecspubli$ ps -ef|grep python
502  4920  4817   0  3:04PM ttys000    8:09.50 python server.py
502  5252  4817   0  3:47PM ttys000    0:00.01 grep python
502  5077  5071   0  3:15PM ttys002    0:00.09 python chat_server.py
```

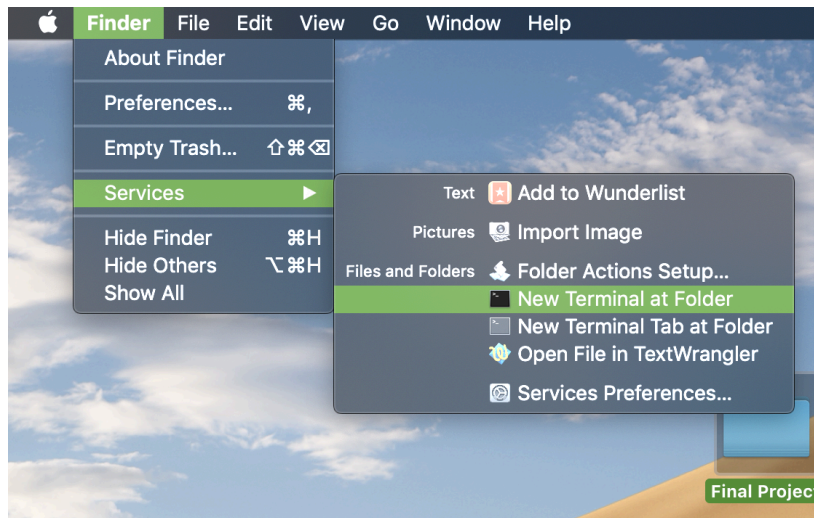
Then kill the process using `kill <process number (2nd column)>`

```
JoyceNYUSH1743LP-MX:tyler ecspubli$ kill 4920
[1]+  Terminated: 15      python server.py (wd: ~/Google Drive/ICS
wen/ICS_chat_joyce)
(wd now: ~/Google Drive/ICS_Spring_2019/PP_joyce/tyler)
JoyceNYUSH1743LP-MX:tyler ecspubli$
JoyceNYUSH1743LP-MX:tyler ecspubli$
JoyceNYUSH1743LP-MX:tyler ecspubli$
JoyceNYUSH1743LP-MX:tyler ecspubli$ ps -ef|grep python
502  5983  4817   0  7:28PM ttys000    0:00.00 grep python
502  5077  5071   0  3:15PM ttys002    0:00.09 python chat_server.py
```

(yf28) [back to top](#)

Open terminal window at a certain folder(Michael)

If you want to open terminal at a certain folder, you can operate as follows:



You can save a lot of time!

If you don't have this button, you can open the Services Preferences and check these two options. Hope it will help!

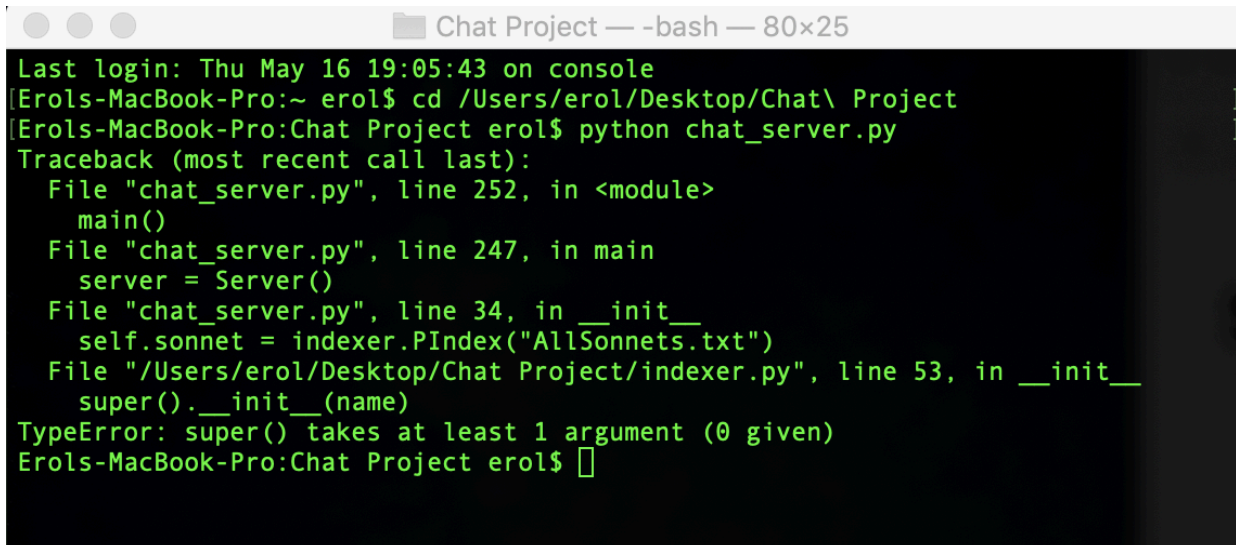
(jlz290) [back to top](#)

Type Error: “super()” takes at least one argument (0 given)

When attempting to run my program, I had difficulty running my chat\_server.py in my terminal. I was attempting to run the following which then resulted in this error message:

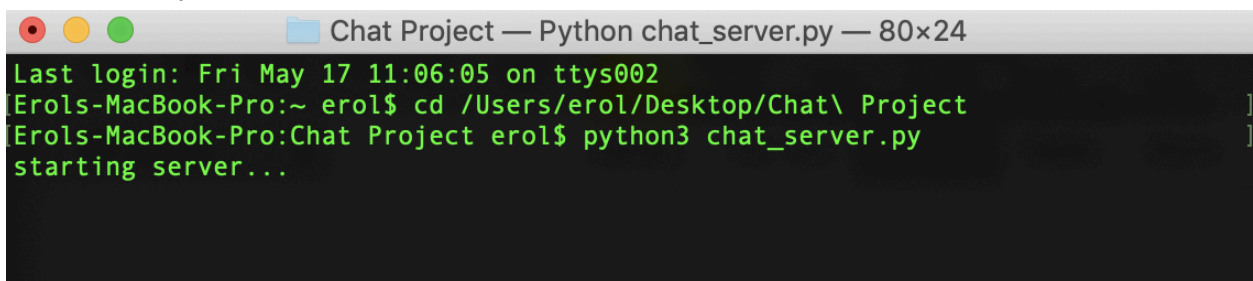
(After specifying my folder, used python chat\_server.py)

This error came up because of me using the python chat\_server.py instead of python3 chat\_server.py. My built-in terminal has python2 in

A terminal window titled "Chat Project — -bash — 80x25". The prompt is "Erols-MacBook-Pro:~ erol\$". The user enters "cd /Users/erol/Desktop/Chat\ Project" and then "python chat\_server.py". The terminal shows a traceback error: "File 'chat\_server.py', line 252, in <module>: main(); File 'chat\_server.py', line 247, in main: server = Server(); File 'chat\_server.py', line 34, in \_\_init\_\_: self.sonnet = indexer.PIndex('AllSonnets.txt'); File '/Users/erol/Desktop/Chat Project/indexer.py', line 53, in \_\_init\_\_: super().\_\_init\_\_(name); TypeError: super() takes at least 1 argument (0 given)". The prompt returns to "Erols-MacBook-Pro:Chat Project erol\$".

```
Last login: Thu May 16 19:05:43 on console
[Erols-MacBook-Pro:~ erol$ cd /Users/erol/Desktop/Chat\ Project
[Erols-MacBook-Pro:Chat Project erol$ python chat_server.py
Traceback (most recent call last):
  File "chat_server.py", line 252, in <module>
    main()
  File "chat_server.py", line 247, in main
    server = Server()
  File "chat_server.py", line 34, in __init__
    self.sonnet = indexer.PIndex("AllSonnets.txt")
  File "/Users/erol/Desktop/Chat Project/indexer.py", line 53, in __init__
    super().__init__(name)
TypeError: super() takes at least 1 argument (0 given)
Erols-MacBook-Pro:Chat Project erol$
```

stalled so by using python chat\_server.py I am telling the terminal to run my program using python2 rather than python3. Once I specify that I want to use python3 by typing python3 chat\_server.py, the result occurs:

A terminal window titled "Chat Project — Python chat\_server.py — 80x24". The prompt is "Erols-MacBook-Pro:~ erol\$". The user enters "cd /Users/erol/Desktop/Chat\ Project" and then "python3 chat\_server.py". The terminal shows the output "starting server...".

```
Last login: Fri May 17 11:06:05 on ttys002
[Erols-MacBook-Pro:~ erol$ cd /Users/erol/Desktop/Chat\ Project
[Erols-MacBook-Pro:Chat Project erol$ python3 chat_server.py
starting server...
```

So in summary, if you have difficulty running your program and the error previously stated comes up, try using python3 chat\_server.py instead of python chat\_server.py

(emb689) [back to top](#)

FileNotFoundError: [Error 2] No such file or directory: "AllSonnets.txt.idx"

Our team also had the problem of opening 'chat\_server.py' using the terminal.

```
FileNotFoundError: [Errno 2] No such file or directory: 'AllSonnets.txt.idx'
```

The terminal kept complaining about the errors of different '.pk' and '.idx' files not found in the directory, though we made sure that there were such files with the correct names in the folder. Later we found out that clearing all the spaces in the folder names of the directory might solve this problem. We also suggest launching the '.py' files in the folder where it belongs. You need to access the folder layer by layer using 'cd' command.

(cc5709 and cx523) [back to top](#)

## Error using SpeechRecognition library

Downgrade python to 3.6 or below because pyaudio does not work with python 3.7 → enter following commands in terminal (creates new environment with downgraded version of python in anaconda):

- pip install SpeechRecognition
- conda install -c anaconda pyaudio
- conda create -n nameOfEnvironment python=3.5
- conda activate nameOfEnvironment
  - Repeat this command every time you open terminal, check left side of terminal to tell which environment you're in

```
(base) Phobes-MacBook-Pro:~ phoebeharmon$ conda activate nameOfEnvironment
(nameOfEnvironment) Phobes-MacBook-Pro:~ phoebeharmon$
```

- pip install --upgrade ibm-watson
  - only if you're using ibm watson API
  - note: if using API that requires password + username or api key + url, delete them from code before uploading to github

(prh262) [back to top](#)

## Pygame window not opening/closing

- If your Pygame file does not have a main loop function, like mine, or if calling the main loop function doesn't work, you can use "import os" to import the entire pygame file into the client\_state\_machine file.

Then, you can call the pygame file by using "os.system('python file\_name')".

This way, pygame should be running without error.

(yy2564) [back to top](#)

## Pygame window all black

- If you finish coding an interface and turns out to be all black after running it, check if you have use the function `pygame.display.update()` after coding, without updating the data, the game cannot function.

(zy1223) [back to top](#)

## How to close Pygame window without quitting python

- Usually, we use `pygame.quit()` and `sys.exit()` to quit pygame window, however, as we need to return back to the chat system, it's better not to use `sys.exit()`. This may cause pygame error: video system not initialized in some cases. Then try to apply 'try' and 'except' to the specific code.

(ys3062) [back to top](#)

## Pygame quit in windows system and macOS system

- In windows system, after using for loop to catch the event type of `pygame.QUIT`, first type in "`pygame.quit()`", then type in "`quit()`" in the following line. In the OSX system, the quit mechanism is a bit special: first type in "`pygame.quit()`", then type in "`os._exit(0)`". Of course you have to import `os` first.

(xw1744) [back to top](#)

## Code a game which contains various phases

- A game can be really difficult to code if it has many phases, going from one to one and cycling within themselves. The game we create is "Werewolves", which has days and nights. In the nights there are the rounds of wolves, of the seer, of the witch, etc. In the days there are the rounds of discussion and of voting. And the game circles between days and nights. These various stages make the game very complex. To solve this, we use state machine to help switch the days and nights. State 1 is distributing the identities, state 2 is the night, state 3 is the day. The game first goes from state 1 to 2 and then cycle between 2 and 3 until it ends. And for the various phases inside the nights or inside the days, we create a lot of True or False flags and if-else statements. After finishing, we think that it may be more convenient to use another state machine(s) there. Overall, if the game has various stages, it is super important to think thoroughly and clearly about how to represent different stages and how to go from one to another before you start coding.

(wj621) [back to top](#)

## Handle characters who can have various states in a game

- In a lot of games, characters have different skills which they can use only once in a turn. And characters can be alive or dead. All these features make the coding of the game complex. In the game "Werewolves" we code, we use a state machine to help change

the states of the characters. State 1 means this character hasn't used his skill this turn, state 2 means this character has already used his skill this turn, state 3 means this character is dead. It makes the code clearer. But after we finish coding, we think that it would be better if we use multiple state machines. One is to switch the condition of using skill or not. Another is about alive or dead and this one should have three stages instead of two: "alive", "temporarily dead (which can still be saved)", "dead". It is important to create multiple state machines and many stages categorized as detailed as possible, since it can make the logic far clearer.

(zd662) [back to top](#)

Check collision and update player's position in a game

```
collide_list = pygame.sprite.spritecollide(player, block_list, False)
if len(collide_list) != 0:
    player.rect.bottom -= speed+1
    player.rect.y -= speed+1
```

- One of the most fundamental part of the game is to check collision between different objects. However, the simple method for detection of collision by writing if (same coordinate) then (collision) is not very accurate when dealing with moving objects. Furthermore, there are usually delays, resulting in poor game experience. The most convenient way is to use the sprite feature to convert the object to rect, and use the built-in functions of the sprite class to detect collisions and update player's information. Above is an example for your reference.

(yc 2949) [back to top](#)

## File Transfer(lc3913)

Unable to send binary data through json.dumps()

- Create your own functions to send and receive binary data

Errno 35 in macOS when receiving binary data from client

- Change the self.setblocking(0) in chat\_server.py to self.setblocking(1), in order to allow blocking connection, which would enable file transfer

File damaged when transfer finishes (No solution yet)

The receiving while loop in server could never reach an end, as the "if not" statement was never executed(No solution yet)



```

while True:
    file_data = myrecvF(from_sock)
    print("yes")
    if not file_data:
        break
    mysendF(to_sock, file_data)

```

Sending file through JSON:

JSON does not accept binary data, so encode the binary data into a string first. My solution: binary\_data → b64encode (still binary) → utf-8 encode (now a string) → split the string for transferring large files → now the string is ready to be sent within the dictionary sent to the server, just like how regular messages work. Reverse the process on the receiving end.(yh2689) [back to top](#)

## User Input From Console

The get\_msg function would be skipped and there is no way for user to input message

- Since the server and chat\_client\_class both run in an infinite loop, you need to add an “if len(msg) > 0” block to ensure the following code would only be executed when there is input from the user  
(lc3913) [back to top](#)
- Also, in the main part, you can use if \_\_name\_\_ == '\_\_main\_\_' to ensure when you import the file, it won't run automatically and cause problems.  
(yl5726) [back to top](#)

## GUI

How to get away with not learning how to thread for GUI: [back to top](#)

- My solution using tkinter: First, launch the mainloop for GUI in chat\_cmdl\_client. Then bind the mainloop for chat\_cmdl\_client as an event in the GUI. (eg. chat\_cmdl\_client launches the GUI. Then, say, the user moves the cursor into the chat window, and that triggers the mainloop for chat\_cmdl\_client, launching the client.) (yh2689)
- The most difficult part for doing a GUI is how to take care of the infinite loop. I wrote this loop in the chat\_client\_class.py since I only wanted the clients to see my GUI. And the best way to check GUI, is to write “\_\_name\_\_ == \_\_main\_\_” in the code, since it can help you stop the infinite loop directly. (sz2530)
- I tried to make our chatting interface more like a slide. Each time the output function is called in client\_class, I use the tkinter update() function to refresh the window.



As mentioned above by yh2689, the solution is great. But pay attention to the import part when running chat\_cmdl\_client and editing chat\_client\_class. Be sure to write if `__name__ == '__main__':` (jx1038) [back\\_to\\_top](#)

- When i tried to put GUI in our chat system, i still launching the mainloop for GUI in chat\_cmdl\_client.py, and change the variable in chat\_client\_class.py. If the GUI file can run itself, be sure to check if you write `__name__ == '__main__':`. But then i find a bug which is even if i didn't login successfully, i still can chat if i close the window of GUI immediately. Because we run the chat in chat\_cmdl\_class when we close the GUI. So the solution we find is add a boolean in GUI and also add in if in chat\_cmdl\_class.py. (wy815) [back\\_to\\_top](#)

One easy way to construct input and output without threading: (jy2363)

Typically since the terminal and the GUI is the thing that would continuous refresh itself, you need to open another threading to combine the GUI into the chat system. But there is an easy way to do that, with the price of your system might be a little bit of “卡”.

1.

```
def read_input(self):  
    while True:  
        text = sys.stdin.readline()[:-1]  
        self.console_input.append(text) # no need for lock, append is thread safe
```

This is originally how the terminal reads your input at terminal, the function `sys.stdin`. So since you know it's keep refreshing, you will need to control it, say only after you enter something in you GUI input box and press “Enter” will the message be read and send to the server.

My way of doing it is doing something like this 🙌 :

```
def read_input(self):  
    while True:  
        #text = sys.stdin.readline()[:-1]  
        if len( self.interface.text_out) > 0:  
            text=self.interface.text_out[0]  
            del self.interface.text_out[0]  
            self.system_msg += text  
            self.console_input.append(text) # no r
```

So the logic is simple. You design a output box at the GUI interface. When you write something and press “Enter”, you will send the message to the box [ in python it might just be a string variable or a list], and when the condition {`self.interface.text_out`} is bigger than 0, namely it has something, you help send it to the server, and then clear the box.

But the price is that, compared with 2 threadings keep refreshing , you are taking turns here, as if you refresh the chat system once and then the GUI interface once and again and again, so it might get a little bit “卡”.

2.

```
def output(self):
    if len(self.system_msg) > 0:
        print(self.system_msg)
        self.system_msg = ''
```

This is originally how the terminal outputs the message from the server. So if you want to see the message on your GUI interface. You have to find a way to transfer the self.system\_msg to the GUI system.

Surely you need design a Textbox to hold these message on your GUI interface. So the way I do this is as this 📌:

```
def output(self):
    if len(self.system_msg) > 0:
        # print(self.system_msg)
        # GUI.output=self.system_msg
        self.interface.get_message_print('\n'+self.system_msg)
        self.system_msg = ''
```

So you really need to import the GUI as an attribute for the client class. And design a function to display the message on your GUI. This is much easier than the read\_input thing I've mentioned above!

----(jy2363) [back to top](#)

### Modifying GUI get\_message

- Instead of using the original sys.stdin.readline()[:-1] to get message from the console, I set a shared variable final\_msg between the main loop and the GUI loop to transfer information from the interface to the chat program.  
(zz1763) [back to top](#)
- Besides, we can set up a function in the GUI that return the message and import it in the client\_class.py. To show the message, since all printed message is through on function output(), we can add GUI.insert(END, message) in that function to print the message both in terminal and GUI window.  
(yl5726) [back to top](#)

## Destroying the previous page

- If you have a button in your GUI that is linked to the next window, do not forget to totally destroy the previous window. I used the `grid_slaves()` method to first collect all the elements in the previous window in a list, then applied the `destroy()` method to each element in the list to totally destroy the page.  
(jc8135) [back to top](#)
- Instead of `destroy()`, if we have a login window, we can use `quit()` to just close the window of login but pass the user name to `client_class.py`.  
(yl5726) [back to top](#)
- Both `destroy()` and `quit()` work. Sometimes, we can also try `hide()` to conceal the previous window temporarily. Remember applying `destroy()` method in proper position in your code, or the user may completely quit the whole chat system when he/she just wants to close the previous window. (jb6614) [back to top](#)

## Passing message from GUI to the server

- We can set up a function `send(event)`, which can get the input from the GUI input and trans the message as a variable into the `client_class.py`, to set the variable `my_msg` as the message from GUI.  
(yl5726) [back to top](#)

## Connecting Two User Interface together using PyQt5:

- By creating a new attribute in the first Interface class as `self.name`, in the second class, which is the main thread, I import this attribute, in this way these two user interfaces can connect with each other. (xh1082) [back to top](#)

## Choosing the main thread using PyQt5:

- According to the PyQt5 Documentation, for PyQt5, only the User Interface Class can be the main thread, otherwise, it would raise an error. All the other threading such as reading messages from the server and client and display them, reading names should be attributes in the main thread class. (xh1082) [back to top](#)

## Sending Images from the GUI to the server using Tkinter

- `TclError: couldn't recognize data in image file`. We encountered this minor problem when attempting to send a jpg or png file through the chat. We later realized in the latest version of Python, you have to open the file using `Image.open()` and then pass it through `ImageTk.PhotoImage()`. Tkinter does not support modifying the file after you have passed it through `ImageTk.PhotoImage()`, so you should modify it beforehand. (sc6354) [back to top](#)

```

def send_img(self):
    f = tkinter.filedialog.askopenfilename(parent=self.m2, initialdir='C:/Tutorial',
    title='Choose Image', filetypes=[('png images', '.png'), ('jpeg files', '*.jpg'), ('gif images', '.gif')])
    openImg = Image.open(f)
    img = openImg.resize((300, 300), Image.ANTIALIAS)
    img = ImageTk.PhotoImage(img)

    panel = Label(self.text, image=img)
    panel.image = img
    panel.pack()

    return

```

Changing the background image of GUI or login system:

- When we create the GUI window, we can set canvas to add image to it  
`canvas = tk.Canvas(window, height = 300, width = 500)`  
`imagefile = tk.PhotoImage(file = 'xk.png')`  
`image = canvas.create_image(0,0,anchor = 'nw', image = imagefile)`  
`canvas.pack(side = 'top')`

and then add put labels on it to get username and other information or develop other features. (jb6614) [back to top](#)

Updating the information on the Label in Tkinter

- While Tkinter is executing the main interface, `mainloop()` is a pretty popular method to trigger the appearance of the Tkinter. However, when I want to update the information in Tkinter based on the action of user, it is impossible to change the component directly because the `mainloop()` does not end and python, by default, will not run another loop at the same time. To solve this problem, there are two solutions. (1). Use `update()` to execute the Tkinter instead of `mainloop()`. `Update()` is not an infinite loop and it could help us to both show the Tkinter interface as well as change the label of the Tkinter. (2). Threading could be added to run two infinite loops at the same time. While `mainloop()` is working, the function of changing information could also be executed at the same time. (yz4115) [back to top](#)

```

def main(to_sock):
    global GAME_STATE
    c = 0
    # window.update_idletasks()
    Button(to_sock).create_row()
    window.update()
    for i in GAME_STATE:
        if i == 0:
            c += 1
        if c == 9:
            break

```

## Embed GUI and Pygame into Chat System

- I would like to achieve an interactive game platform, many necessary information and parameters should transports from one tkinter to another one. Generally speaking, the information is from Pygame file, Tkinter file, chat server, client interface. So, it is challenging and important to pass these parameter through different files safely and carefully. To achieve this, I set up some class to inherit parameters from other files so that every file will pass the correct parameter to other files.(yz4115) [back\\_to\\_top](#)

```

class play_game:
    global counter
    global player_1, player_2
    GAME_STATE = {0: 1, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1}
    # counter = 0
    def __init__(self, to_sock, state, game_num, game): # game 是导入module的触发函数
        self.state = state
        self.game_num = game_num
        self.message = 'If you win, you will get ' + str(self.game_num) + ' point; otherwise, get nothing'
        self.game = game
        self.s = to_sock

    def update_point(self): #用于发送信息的时候...更新分数
        GAME_STATE[self.game_num] = 0
        total1 = 0
        total2 = 0
        for i in player_1:
            total1 += i
        for j in player_2:
            total2 += j

        l1 = tk.Label(frame_4_1, text=str(total1), bg='white', font=('楷体', 20), width=15, height=1)
        l1.pack(side='top', padx=50, pady=0)

        l2 = tk.Label(frame_4_2, text=str(total2), bg='white', font=('楷体', 20), width=15, height=1)
        l2.pack(side='top', padx=50, pady=0)

    def check(self):
        global counter

        if self.state == 1:
            a = tkinter.messagebox.askyesno(title='Hi, Warrior',
                                             message='If you win, you will get points; otherwise, get nothing.')
            if a == True: #player确定进入游戏

```

## Encryption related

“OverflowError: Python int too large to convert to C long”

If the encrypted blocks are stored as integers, it is likely that the code can't handle the value then it is first converted from a python3 int/python2 long to a C long. Then the C long is converted to a C int. So on a platform with 32-bit long, the conversion fails at trying to convert 10000000000 to a long, before it tries converting it to an int. Comments according to experience: if this message comes up, it is very likely that the key deployed is a wrong one. In terms of the public key encryption method, always remember: encrypt with the public key, decrypt with the private key.

### Sending keys properly

Some note on sending keys: It would NOT be proper encryption if the keys/encrypted texts are stored in LOCAL files--in this case, there is no point of encrypting it in the first place! Never leave any trace locally. Anyone in the chat system can possibly find a way to access the local file and decode the message. Make sure that the keys are sent through the socket exclusively. It is also very important to note to where the keys were sent to: people encrypt the message using OTHER'S public key, and they decrypt and read their messages received with their OWN private key! (Personally, the logic is quite clear but it gets mixed up in terms of practicing coding.)

(Simone sy2221) [back to top](#)

### Error complaining json cannot parse binary message when using AES for encryption

When using AES for encryption, there might be an error relating to the switch between the original message and encrypted binary message. Json cannot parse a binary message to the chat\_system. One way to solve the problem is using “message.decode()” and “message.encode()” after encrypting or decrypting the message

(xg764) [back to top](#)

### Error: “Non-hexadecimal digit found”

After you implement the encryption function to the chat system, there may appear the error saying that, “Non-hexadecimal digit found”, this is probably because the time is added before the message after we receive it. Therefore, you need to slice the string, only decrypting the part without time.

(ms11141) [back to top](#)

## Search feature with encryption

After encryption, the search feature failed to work as the messages could not be seen by the server. This problem was overcome by implementing in the client state machine and by storing the individual user history files on the personal machines of the users as opposed to the server. (Enver ek2874) [back to top](#)

## Error 8: Node name nor server name provided (Windows OS)

Error: line8 in chat\_utils.py CHAT\_IP = socket.gethostname(socket.gethostname()) [Error 8] nodename nor server name provided, or not known

My solution: change CHAT\_IP = socket.gethostname(socket.gethostname()) to CHAT\_IP="" (ml5386) [back to top](#)

## How travel between GUI windows with PyQt5 while keeping the same socket(tw1614):

So, in our project we implemented both a GUI and a login system. This means that we technically had to create 3 different GUI's and then based on user input communicate between the GUI's. We had 1. A login window 2. A registration window 3. A chatting window. So when you want to open up a window from a different one I think the easiest way is through a button. When the button is clicked, your code should quit the current window and open the other.

The way we did this was when the button is clicked in the GUI, you have to call a function. Here is a picture of that:

```
#what buttons do if clicked
register_button.clicked.connect(self.open_registration)
login_button.clicked.connect(self.open_client)
```

After you will be taken to either the open\_client window function, or the open\_registration window function. Here is the open\_client function.

Here depending on the message we have received from the server, we will either quit our window or display an error.



```

#opens a chat window when login, or displays error message
def open_client(self):
    global login_msg

    lgn_info = json.dumps({"action": "login", "name": self.user_input.text(),
    self.send(lgn_info)
    response = json.loads(self.recv())

    if response["status"] == "success":
        self.client_w()
        self.hide()

    elif response["status"] == "error":
        login_msg = response["errmsg"]
        login_btn_response = QLabel(login_msg)
        login_btn_response.setStyleSheet("color: red;")
        self.layout.addRow(login_btn_response)

    #clears text from type box
    self.user_input.clear()
    self.pw_input.clear()

```

You can see here when our `self.client_w()` is ran, we get an instance of our client class from then run the function in that class that creates our new window. This is what creates our chatting window, while the `self.hide()` in the previous picture is what quits our current window.

```

def client_w(self):

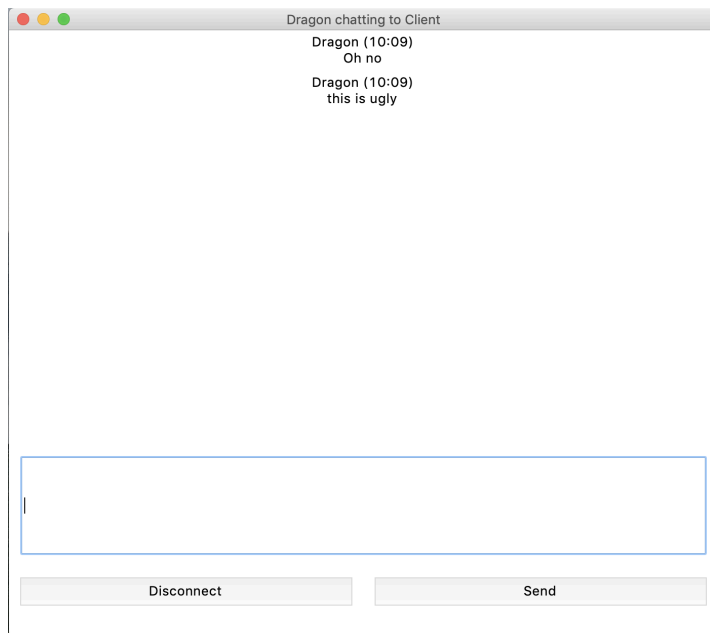
    self.cw = client(self.socket, self.user_input.text())
    self.cw.client_window()

```

Not a very hard concept, but it was tricky for us to implement, hope this can help someone!  
 (tw1614) [back to top](#)

What if i want to keep my text on the left side, but the default place for a text to place is at the center of the label?(tl2564)

When constructing a GUI interface, especially for a chatting interface, sometimes you would like your text to show on the left side of the screen. However, if you simply use pack() function, tkinter will automatically place the text on the center, making your interface a mess. Like shown in the following picture:



And you might find that even if you add the parameter or change pack() into grid function, the text still appear on the center(at least in my case, they don't work)

The solution i come up with, is to use a Text variable, the code is shown as following:

```
import ...

class Chat:
    def __init__(self, socket, client, user):
        self.client = client
        self.user = user

        #self.mySocket = socket
        #self.mySocket.connect(SERVER)

        packedMessage = json.dumps({"action": "newSocket"})
        mysend(self.mySocket, packedMessage)

        self.window = Tk()
        self.window.title("{} chatting to {}".format(self.user, self.client))
        self.window.geometry('700x600')

        self.frame1 = Frame(self.window)
        self.frame2 = Frame(self.window)
        self.frame3 = Frame(self.window)

        self.entry = Entry(self.frame2)
        self.text = Text(self.frame1)

        self.quitButton = Button(self.frame3, text = "Disconnect", command = self.quitChatting)
        # self.searchButton = Button(self.frame3, text = "Search", command = self.search)
        self.entryButton = Button(self.frame3, text = "Send", command = self.send)

        self.frame1.pack()
        self.frame2.place(x = 0, y = 380, width = 700, height = 160)
        self.frame3.place(x = 0, y = 540, width = 700, height = 60)

        self.entry.place(x = 10, y = 40, width = 680, height = 100)
        self.entryButton.place(x = 615, y = 145, width = 75, height = 30)

        self.quitButton.place(x = 10, y = 0, width = 330, height = 30)
        #self.searchButton.place(x = 240, y = 0, width = 220, height = 30)
        self.entryButton.place(x = 360, y = 0, width = 330, height = 30)

        # thread = threading.Thread(target = self.receive, daemon = True)
        # thread.start()

        mainloop()

    def send(self):
        typeInMessage = self.entry.get()
        self.entry.delete(0, END)

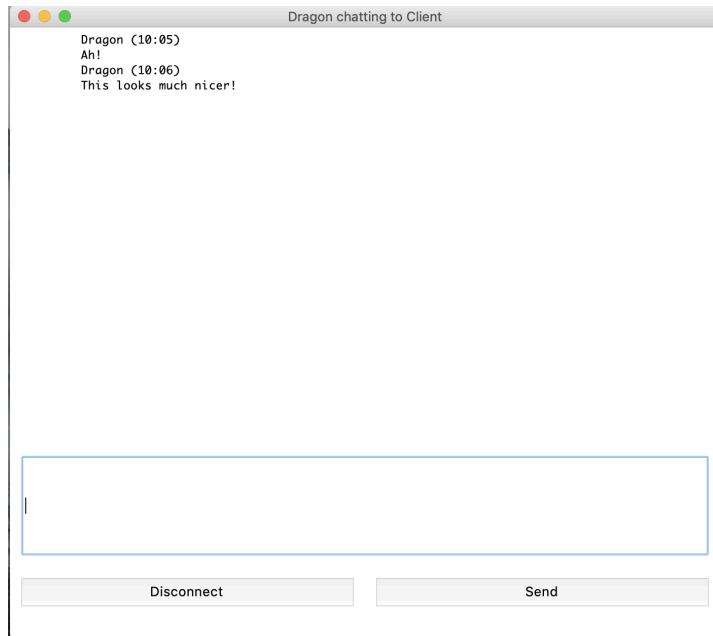
        sendTime = time.strftime('%H:%M', time.localtime())
        messageStamp = "{} ({})".format(self.user, sendTime)

        message = "{}\n{}".format(messageStamp, typeInMessage)

        self.text.insert(INSERT, "{}\n".format(message))
        self.text.pack()

        packedMessage = json.dumps({"action": "chat", "from": self.user, "to": self.client, "message": typeInMessage})
        mysend(self.mySocket, packedMessage)
```

And the text will then show on the place you want:



[back\\_to\\_top](#)

What should I do if my dictionary was chopped into two when using myrecv function?

When doing GUI, you sometimes encounter problems if you are building your code based on UP3, and this problem has to do with it. When creating a new GUI interface, sometimes it will create multiple threads(which I don't fully understand). And that will result in your dictionary got chopped into two parts when trying to receive it for the server, if you are using myrecv()function in UP3.

The solution is simple, just replace myrecv() with the following code:

```
def getClient(self):  
  
    packedMessage = json.dumps({"action": "getAllUser"})  
  
    mysend(self.mySocket, packedMessage)  
    result = json.loads(self.mySocket.recv(1000000).decode()[5:])  
    clientList = result["clientList"]  
    return clientList
```

(tl2564) [back\\_to\\_top](#)

## Weird Dictionary Issue

- So I ran into many issues, some that bugged me for hours and yet I've forgotten them after I fixed the problem. One that I still am stumped by is a seemingly simple one, that I have yet to figure out. I had battleship.py file that was just the backbone for the game. In the state machine file I imported battleship.py and created instances for each player. These instances include 5 data attributes of each ship which were equal to a dictionary with an empty string as the key and a number as the value. This was intended to keep track of how many hit points were left on each ship and subsequently count whenever the ship sank. Now when I ran the game self contained within the battleship.py file, this worked! I was quite proud honestly. Yet, to my surprise and frustration, when calling the function that decreased hit points within the state machine file it did not work! Seemingly the functionality of the dictionary broke?? It was like the data attribute was acting as a normal variable or that it was resetting, which it was doing neither. Nonetheless, I changed my strategy. More tedious, however, but it worked. I just used strings to "tag" each ship and then checked if the boat hit was that ship and decreased the data attribute which was now a number.

(pmn258 or Patrick Newland) [back to top](#)