Eunice Protasio
BSIT 3A
Event- Driven Programming

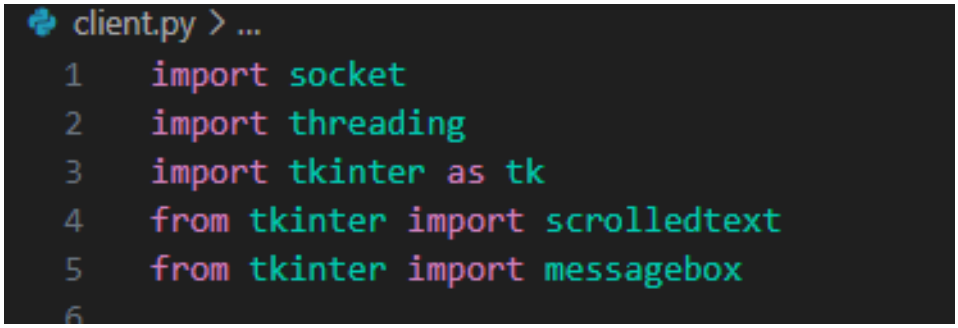
Building A Simple Real-Time-Chat in Python with Tkinter

Real-time communication has become an essential component of many applications in today's fast-paced digital era. Real-time chat functionality is in high demand, whether in collaboration tools, customer service systems, or social networking platforms. In this blog post, we'll look at how to create a real-time chat application with Python, a powerful and widely used programming language.

Understanding the Basics

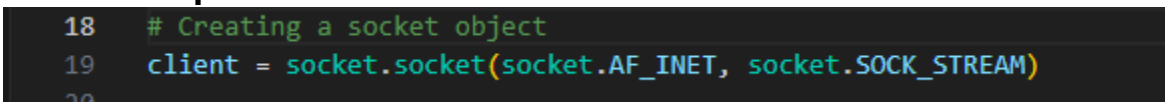
The Client-side code

1. **Imports and Constant**



We import the necessary libraries for networking, threading, and creating the GUI.

2. **Client Set up**



We create a socket object for the client, specifying that it will use IPv4 addresses and TCP for communication

3. **GUI Set up**

The code defines the GUI layout using tkinter. It includes frames for the top, middle, and bottom sections. As well as labels, entry boxes, buttons, and a scrolled text box for displaying messages.

4. **Event Handling Functions**
   - connect(): Connects the client to the server, sends a username, and starts a thread to listen for incoming messages.
   - send_message(): Sends a message to the server.
   - listen_for_messages_from_server(client): Listens for messages from the server in a separate thread.

5. **Main Function**

```
112     root.mainloop()
113 if __name__ == '__main__':
114     main()
```

The main function sets up the Tkinter window and starts the main event loop.

The Server-side code

1. **Imports and contacts**

```
server.py > listen_for_messag
1   import socket
2   import threading
```

Its similar to the client side, the server-side code begins with importing necessary libraries.

2. **Server Set up**

```
58      server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

A socket object is created for the server, specifying the use of IPv4 addresses and TCP for communication. The server is then bound to a host and port.

3. **Client Handling Functions**
   - listen_for_messages(client, username): Listens for messages from a specific client.
   - send_message_to_client(client, message): Sends a message to a specific client.
   - send_messages_to_all(message): Sends a message to all connected clients.
   - client_handler(client): Handles the connection with a client, receives the username, and starts a thread to listen for messages.

Eunice Protasio
BSIT 3A
Event- Driven Programming

4. **Main Function**

```
81 if __name__ == '__main__':
82     main()
```

The main function sets up the server, binds it to a host and port, and starts listening for client connections. Threads are created for each connected client to handle messages concurrently.

Running the Chat Application

1. Run the server: "**python server.py**"
   This starts the server and prepares it to accept client connections.

2. Run the client: "**python client.py**"
   The client GUI will appear, prompting you to enter a username. Once entered, click the "Join" button to connect to the server.

Chat away!