این کد یک سرور ساده برای یک برنامه پیام‌رسان پیاده‌سازی می‌کند. سرور از پروتکل TCP برای ارتباط با کلاینت‌ها استفاده می‌کند و از رابط کاربری tkinter برای نمایش وضعیت سرور استفاده می‌کند.

حال خط به خط توضیحات این کد به شرح زیر است:

1. \*\*import socket, threading, tkinter as tk, from tkinter import scrolledtext\*\*: وارد کردن ماژول‌های مورد نیاز برای ایجاد سرور، مدیریت رشته‌ها، و نمایش رابط کاربری.

2. \*\*SERVER\_HOST = '127.0.0.1'\*\*: تعیین آدرس IP میزبان سرور.

3. \*\*SERVER\_PORT = 12345\*\*: تعیین پورت سرور برای ارتباط با کلاینت‌ها.

4. \*\*clients = {}\*\*: یک دیکشنری برای ذخیره اطلاعات کلاینت‌های متصل به سرور.

5. \*\*class ServerUI\*\*: تعریف یک کلاس برای مدیریت رابط کاربری سرور.

6. \*\*def \\_\\_init\\_\\_(self, master)\*\*: سازنده کلاس `ServerUI` که یک پنجره tkinter برای نمایش وضعیت سرور ایجاد می‌کند.

7. \*\*def broadcast\_message(message, sender\_socket=None)\*\*: تابعی که پیام را به همه کلاینت‌های متصل ارسال می‌کند با استثنای کلاینتی که پیام را ارسال کرده است.

8. \*\*def handle\_client(client\_socket, client\_address, server\_ui)\*\*: تابعی که برای پردازش ارتباط با یک کلاینت ایجاد می‌شود.

9. \*\*def start\_server(server\_ui)\*\*: تابعی که سرور را شروع می‌کند و منتظر اتصال کلاینت‌ها می‌ماند.

10. \*\*def main()\*\*: تابع اصلی برنامه که رابط کاربری سرور را ایجاد کرده و سرور را شروع می‌کند.

11. \*\*if \\_\\_name\\_\\_ == "\\_\\_main\\_\\_": main()\*\*: اجرای تابع `main()` اگر برنامه به عنوان یک اسکریپت اجرا شود.