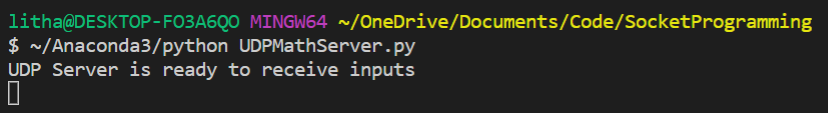
# Socket Programming for UDP:

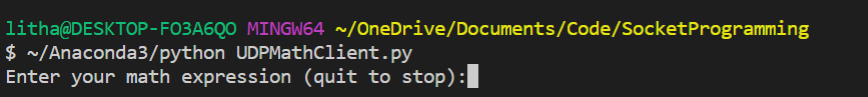
Server Program : UDPMathServer.py

Client Program : UDPMathClient.py

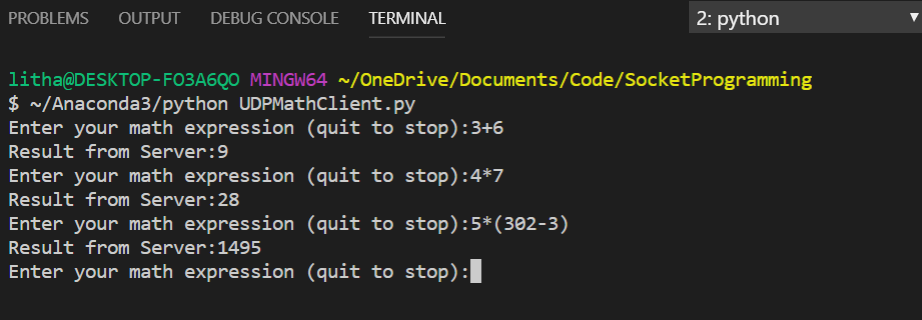
Server Program Execution:



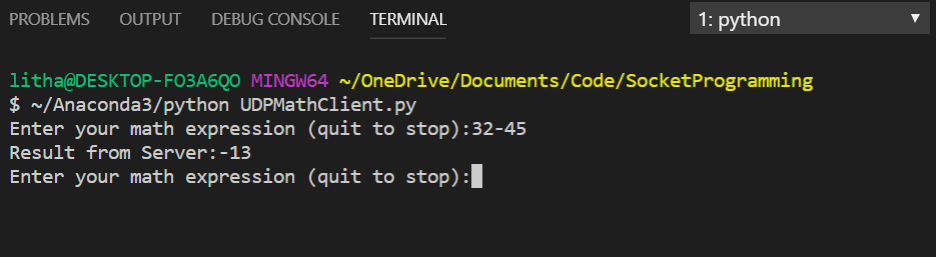
Client Program Execution (First Client):



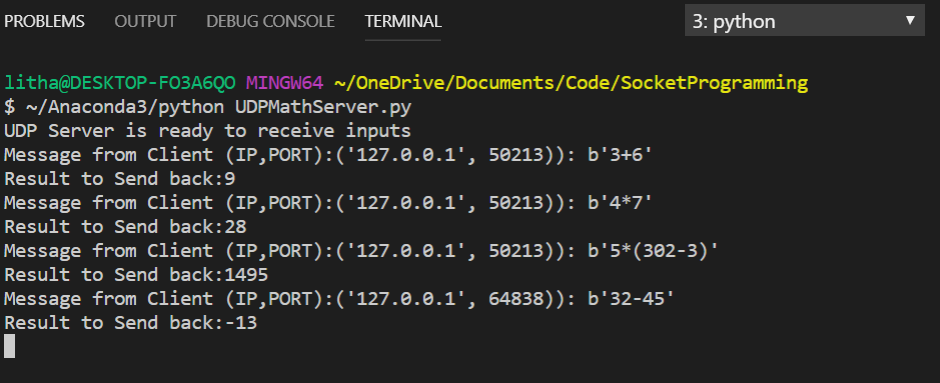
First Client sending inputs to server is shown below



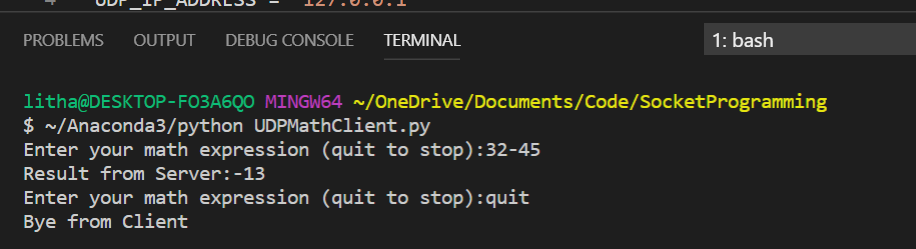
Second Client establishes and sends connection to server as shown below



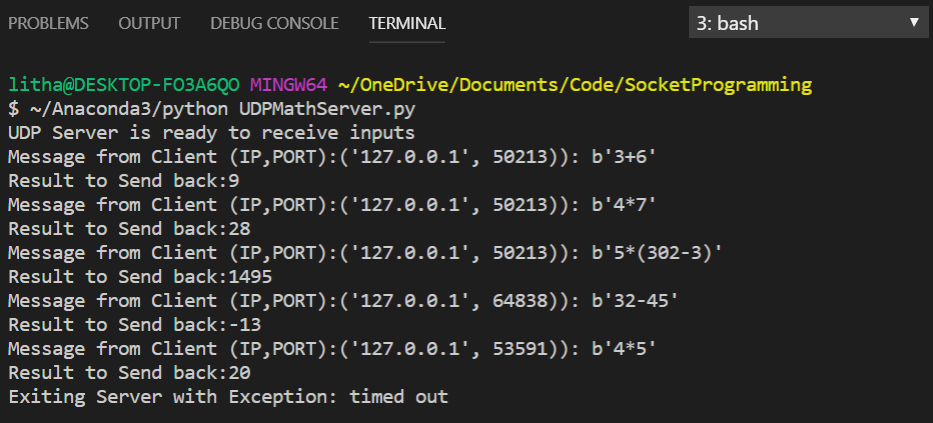
Server side responses to First Client (port : 50213) and Second Client (port : 64838):



Client gracefully quits by providing ‘quit’ command.



Server times out if no new connection till timeout (2 minutes).

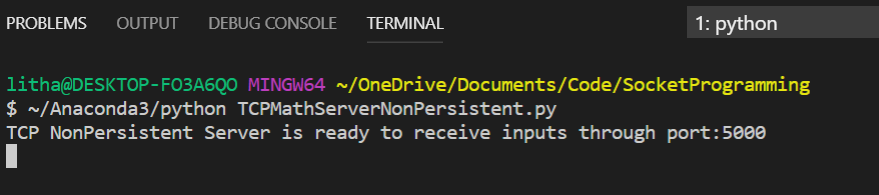


# Socket Programming for TCP (Non Persistent):

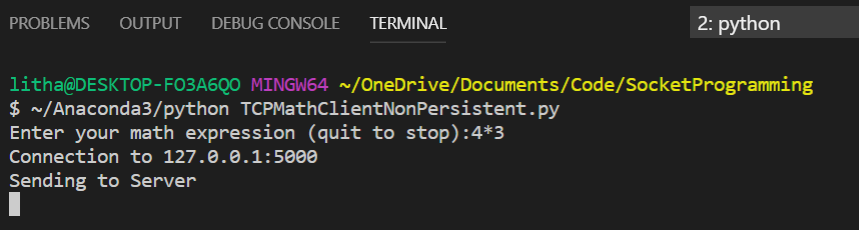
Server Program : TCPMathServerNonPersistent.py

Client Program : TCPMathClientNonPersistent.py

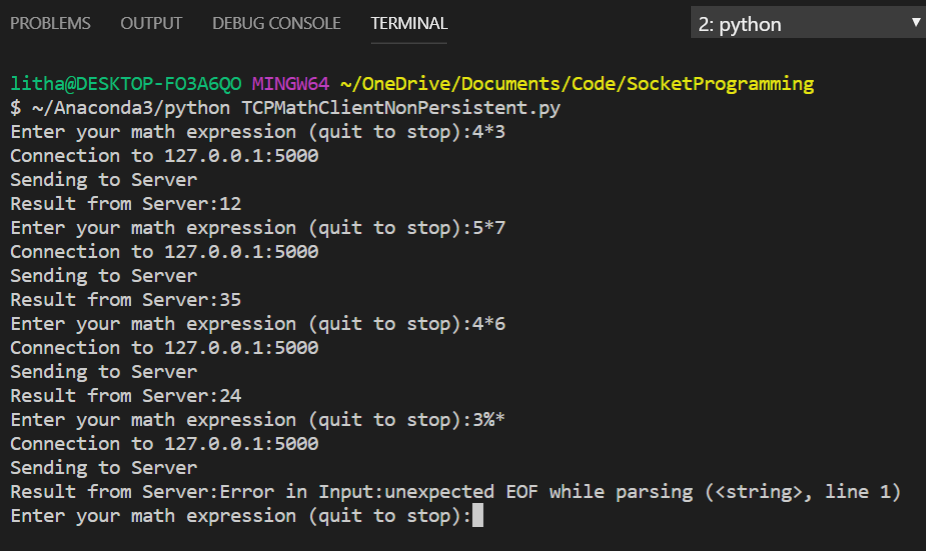
Server Program Execution:



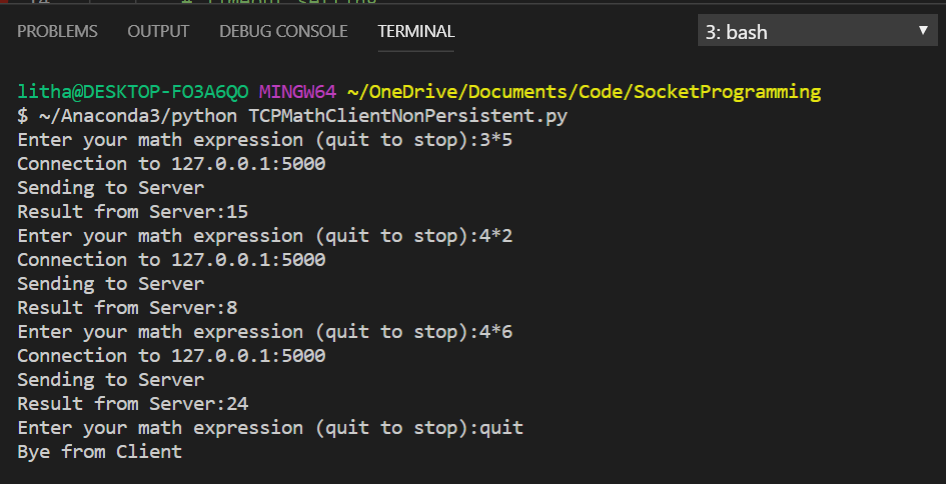
First Client Program execution (waiting for server response/ Server delay of 10 seconds)



Wrong Input handled by server

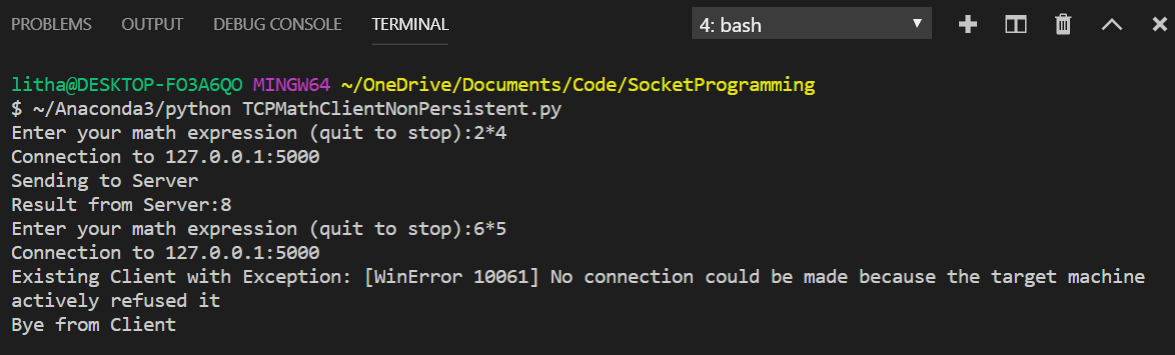


Client 2 sending inputs and gracefully quitting

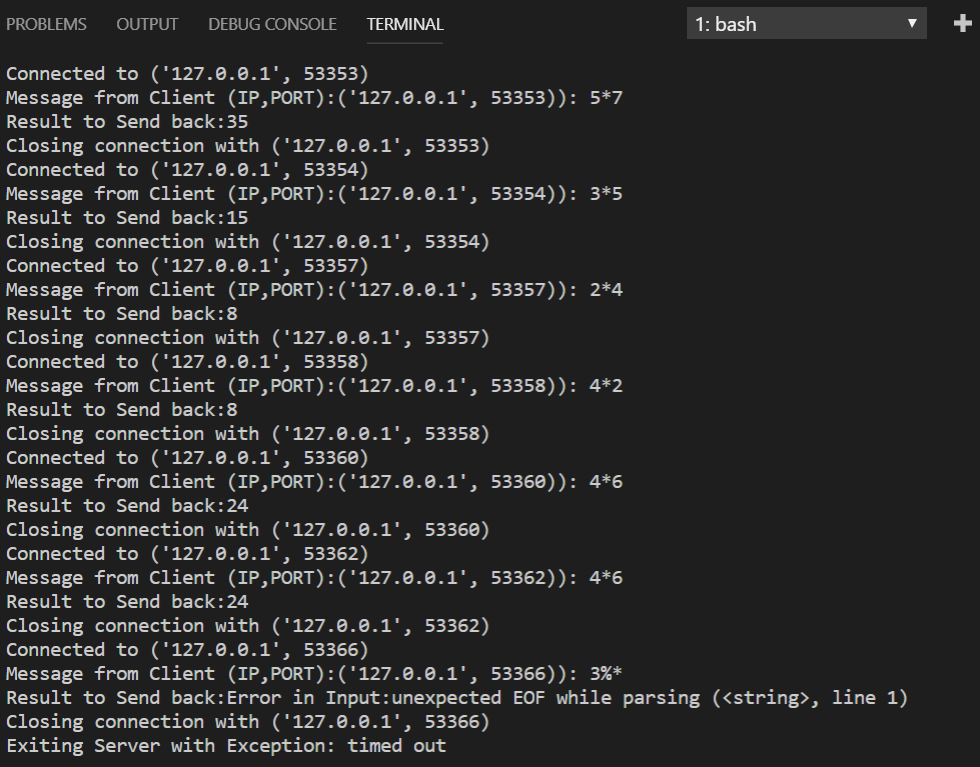


Client 3 trying to connect while server is working on client 1 and Client 2 in queue.

Since Queue size is 1 , Server rejects further connections.



Server responses to different connections (each time new connection and port since the connection is non-persistent). Server times out is no new connections for 2 minutes.

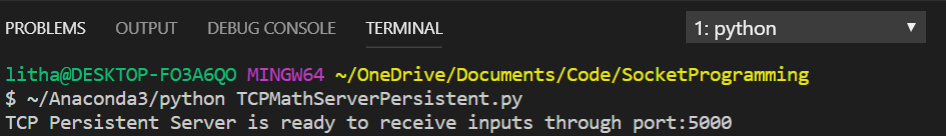


# Socket Programming for TCP (Persistent):

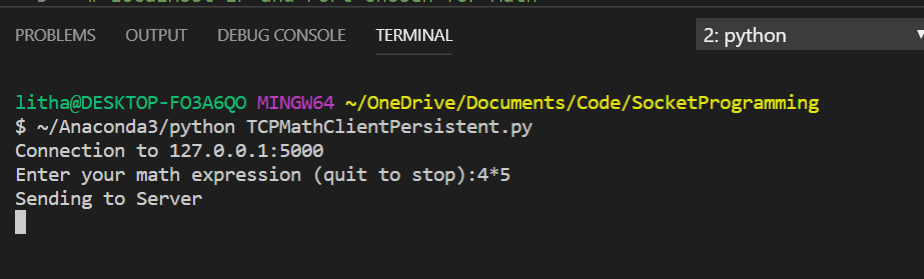
Server Program : TCPMathServerPersistent.py

Client Program : TCPMathClientPersistent.py

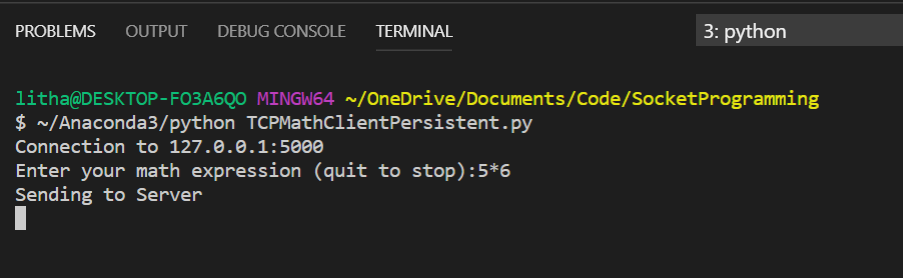
Server Program Execution:



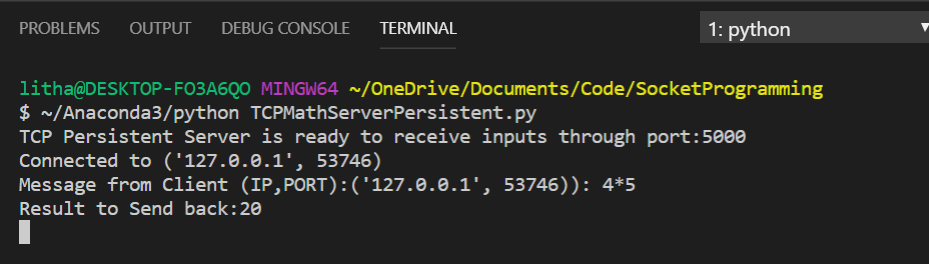
Client 1 connecting and sending input. 10 second delay induced at Server



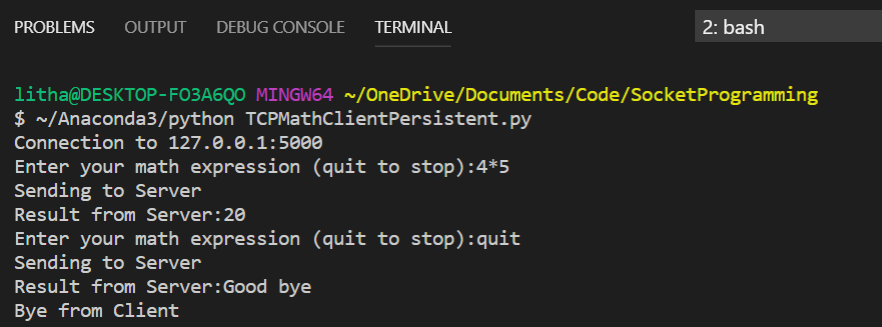
Client 2 connecting and sending input. Waiting for response from server which is serving client 1



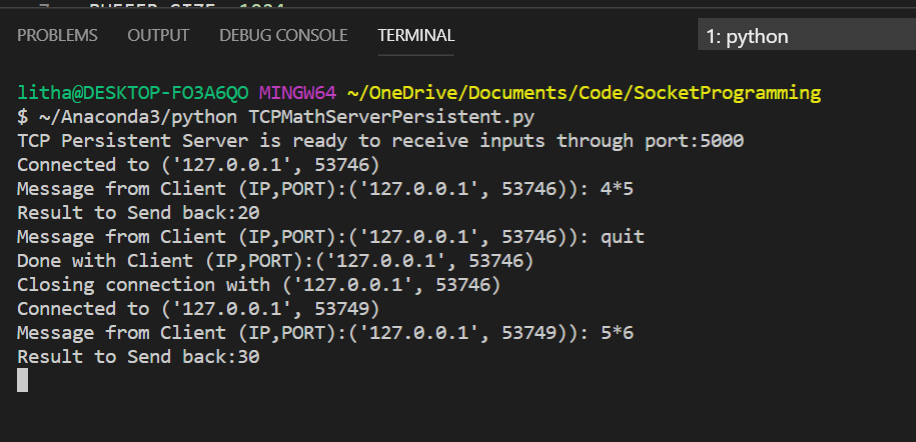
Server sending back response to Client 1.



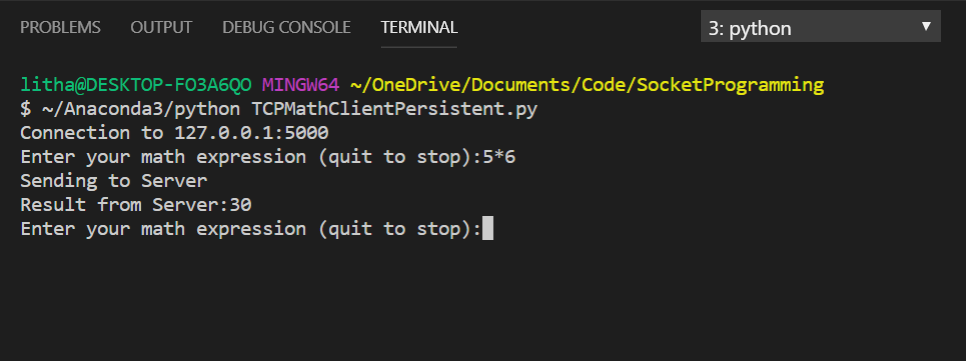
Client 1 gracefully quitting. Server responds back with a Good bye.



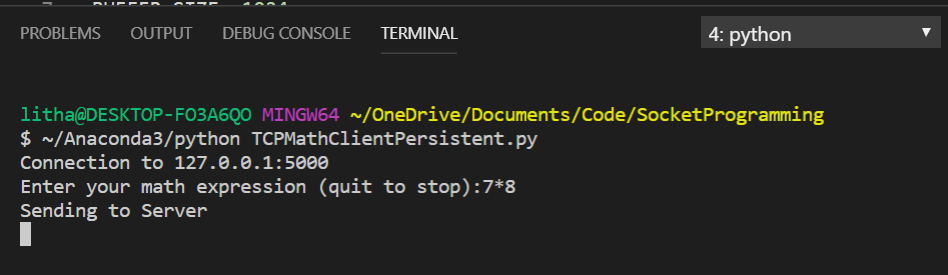
Client 2 is now picked up by server once it was done with Client 1. Same Connection and port is used throughout the lifetime of a connection.Server now send response back to Client 2



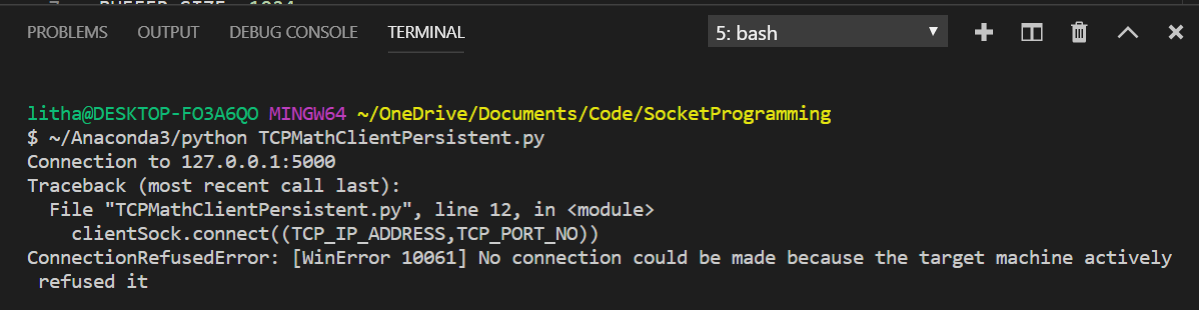
Client 2 received back the response from Server



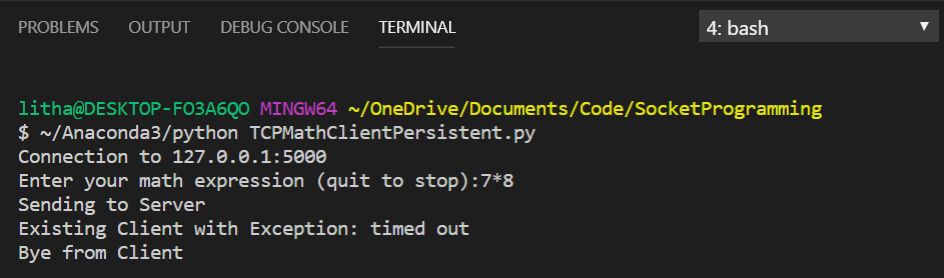
Client 3 is connecting to the server(queued)



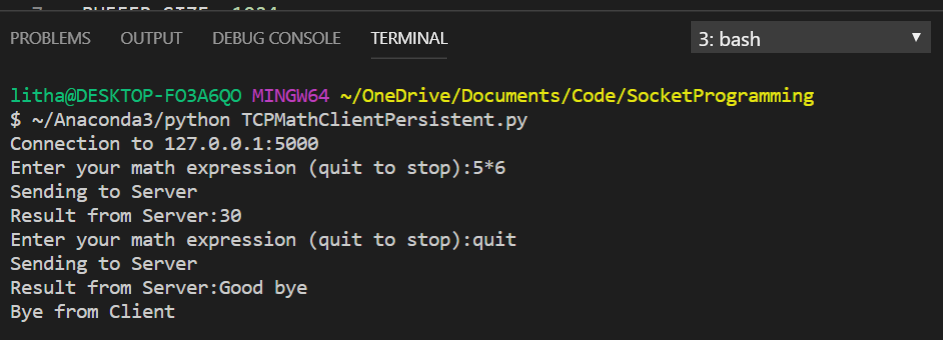
Client 4 tried to connect server . Since this attempt is over the queue limit (1) , server refuses further connections.



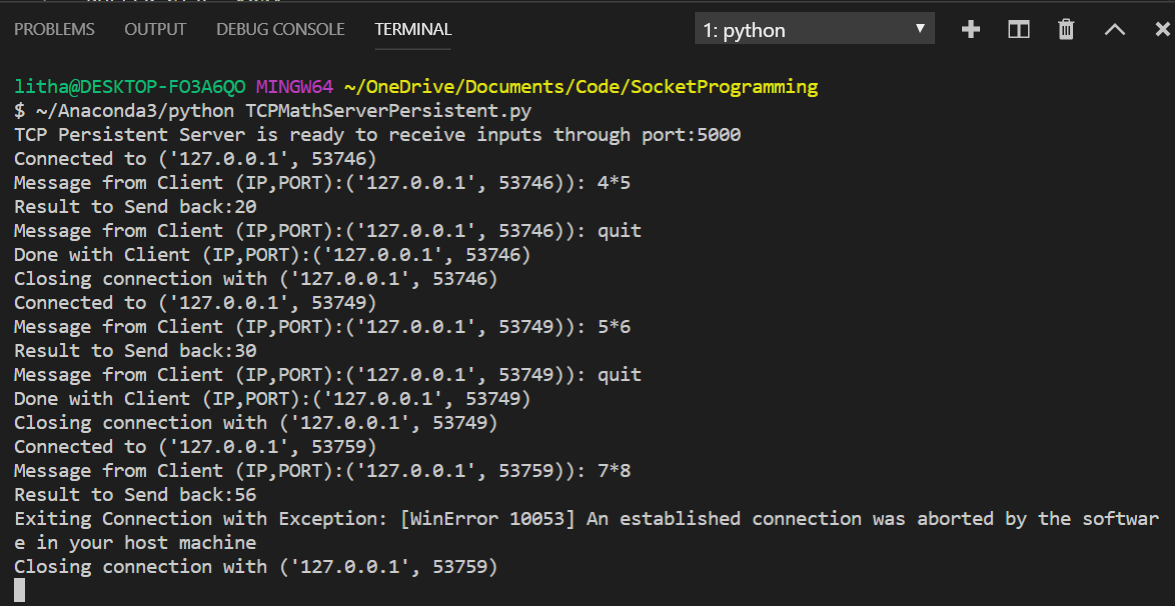
Since ,Client 2 holds up connection for a long time. Client 3 times out after client-side timeout (1 minute)



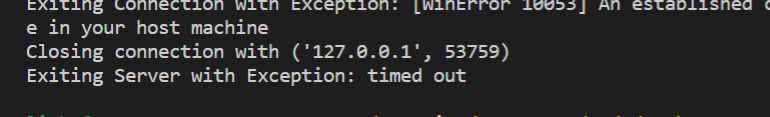
Client 2 gracefully quits the server



Server now picks Client 3 from queue and realize that it timed out before getting response and ended connection.



Server times out if not getting connection for 2 minutes.

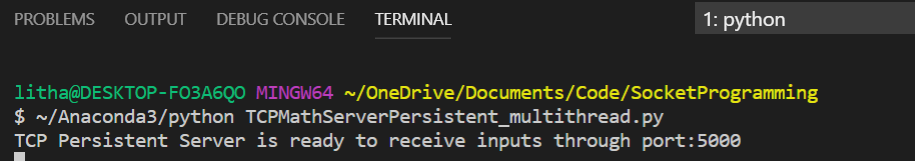


# Socket Programming for TCP (Persistent & multi-threaded):

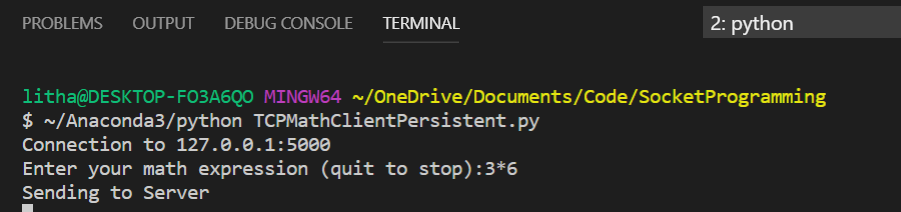
Server Program : TCPMathServerPersistent\_multithreaded.py

Client Program : TCPMathClientPersistent.py

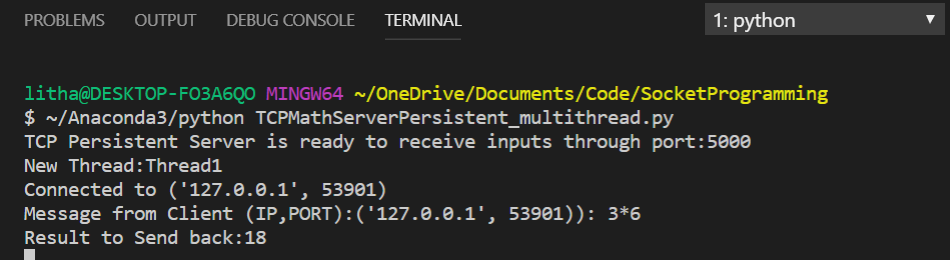
Server Program Execution:



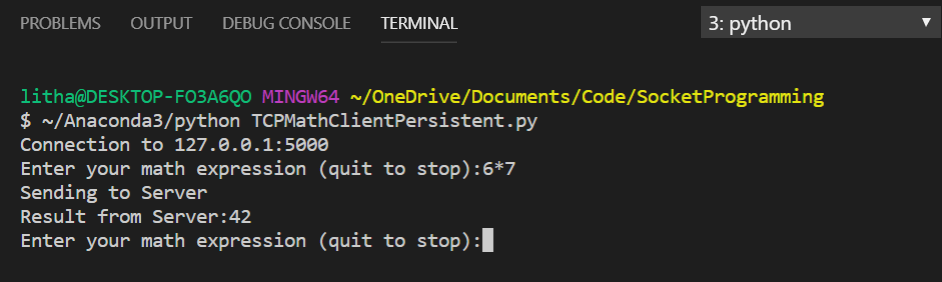
Client 1 connect to the server and provides input



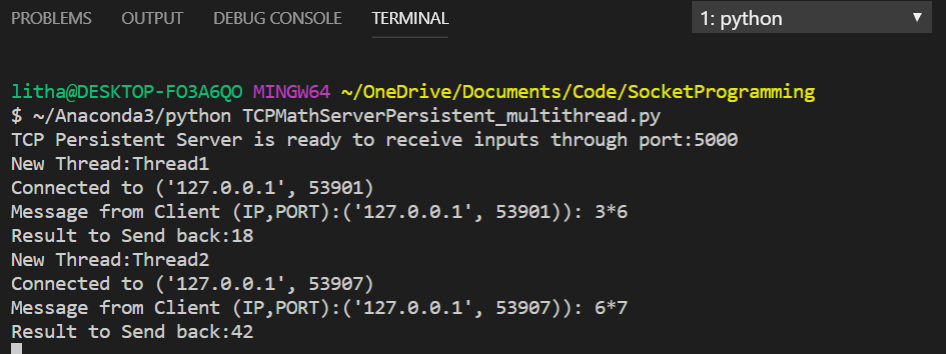
Server creates a new thread to handle Client 1 and provides response



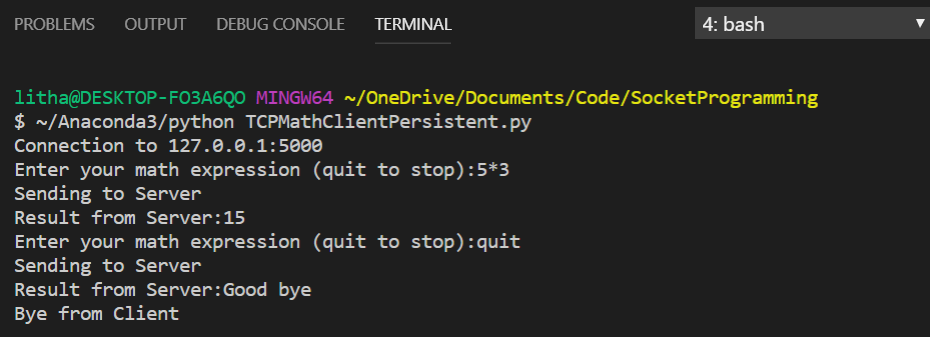
Client 2 connect to the server and provides input



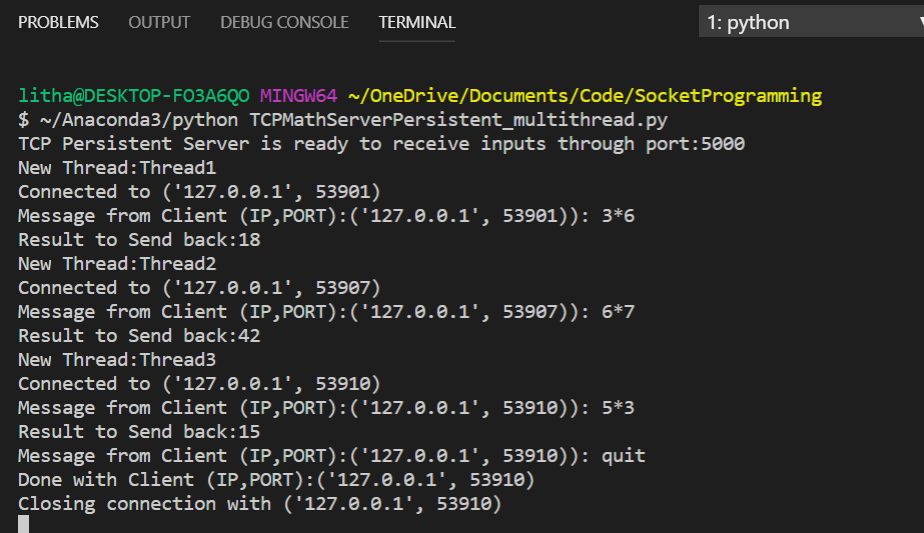
Server creates a new thread (number 2) to handle client 2 and provides response. Client 2 does not need to wait for Client 1 since it is a different connection using a separate thread.



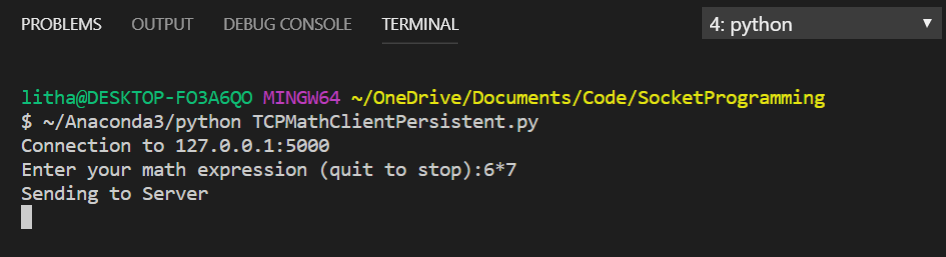
Client 3 connects to the server provides input and gets response and gracefully quits. Server send back Goodbye indicating connection closure.



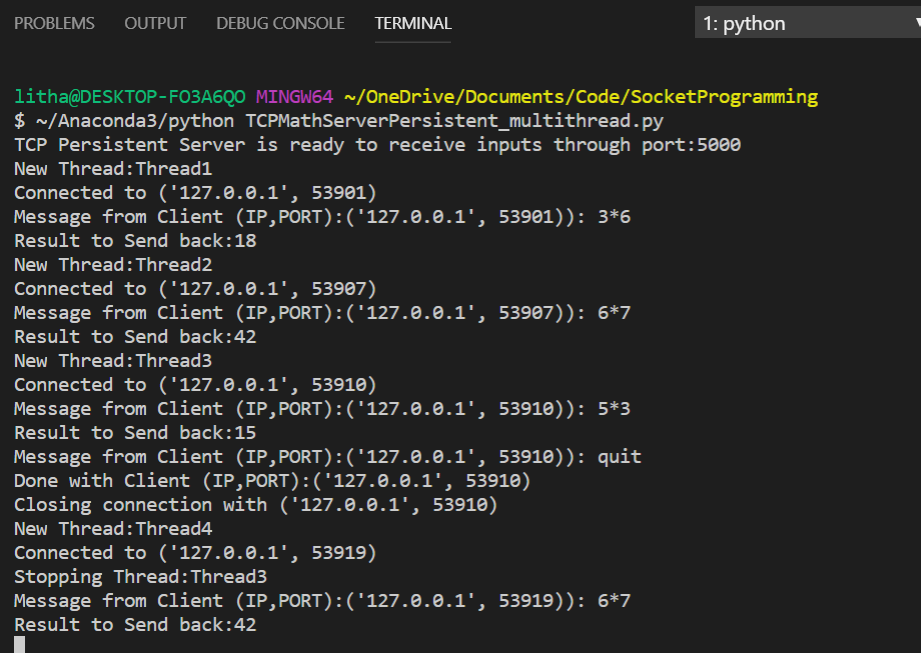
Server creates a new thread (number 3) for Client 3 and closes connection after receiving quit message from client as it is done with that client.



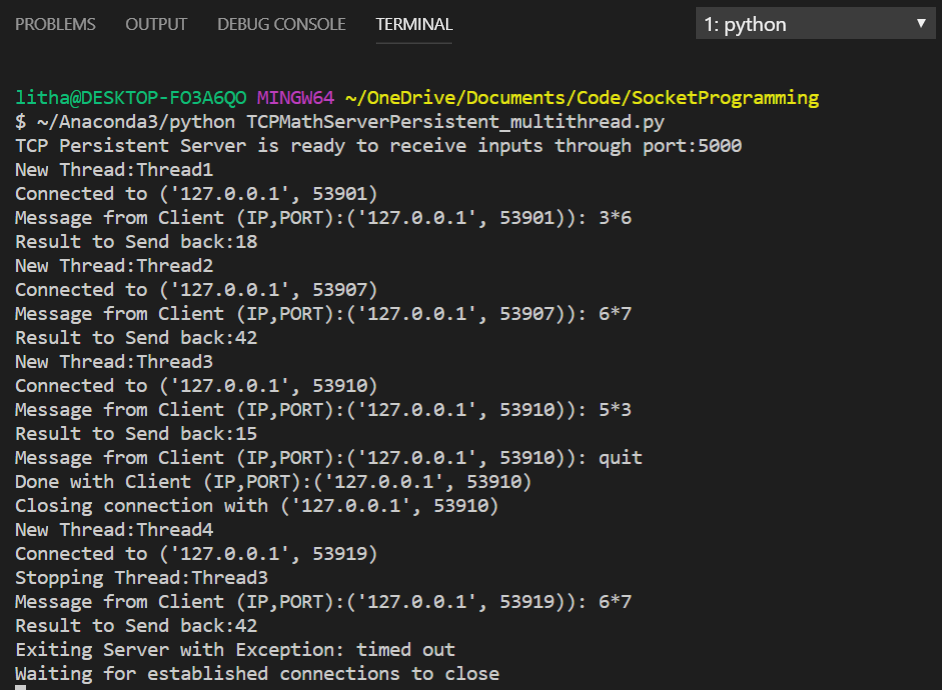
Client 4 connects to the server and provides input



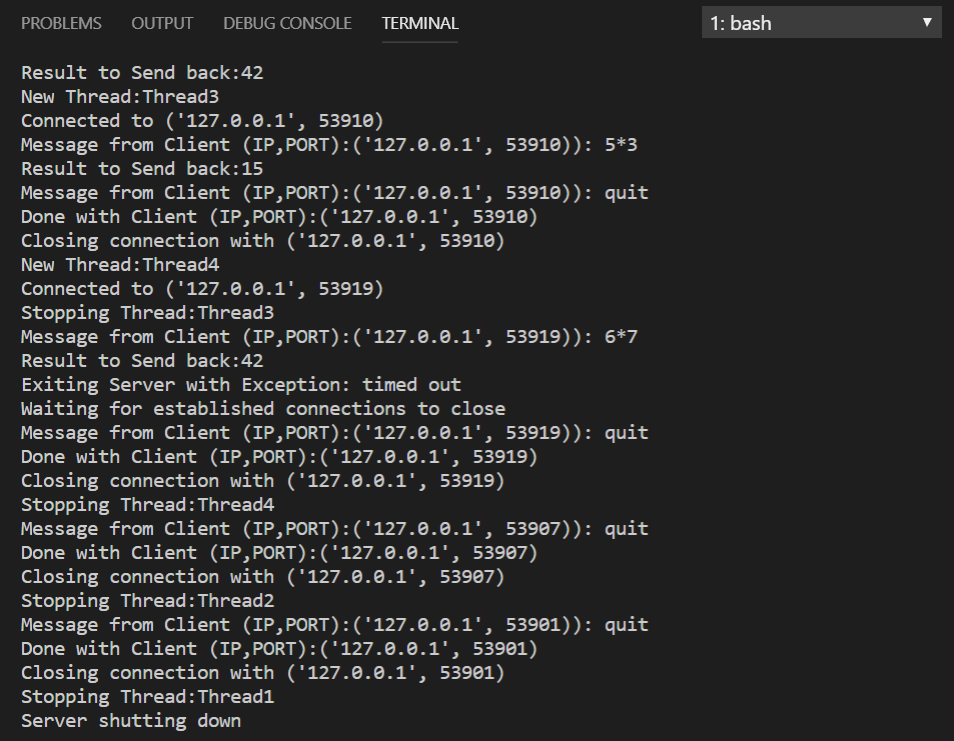
Server creates a new thread (number 4) for client 4 and also stops thread 3 as it is no longer serving client 3.



Since server is not getting any new connection for 2 minutes, it times out. But keeps the threads alive for established connections



Once each client that are connected to the server gracefully quits, corresponding thread is also stopped. Once all threads are done, the server shuts itself down.



# Environment Used:

**OS**: Windows 10

**Language** : Python 3.7 (Anaconda 3)

**IDE** : Visual Studio Code

**Note**: Server Timeout is set for Server programs to close the server one the timeout is reached and no client tries to access that server (Idle Server). This can be removed by commenting the below line in server programs. This line was originally added since the mysys/bash does not take Keyboard interrupt in Windows machines.

