Assignment 14

Server:

import socket

import os

from \_thread import \*

ServerSideSocket = socket.socket()

host = socket.gethostname()

port = 5000

ThreadCount = 0

received\_data = [[3,2,1],[5,4,6],[]]

def merge(a,b):

x = a+b

return x

try:

ServerSideSocket.bind((host, port))

except socket.error as e:

print(str(e))

print('Socket is listening..')

ServerSideSocket.listen(2)

def multi\_threaded\_client(connection,info):

# connection.send(str.encode('Server is working:'))

st = str(info)

connection.sendall(str.encode(st))

while True:

data = connection.recv(2048)

response = data.decode('utf-8')

if not data:

break

print("sent data to client : ",info)

print("client response : ",response)

received\_data.append(eval(response))

connection.close()

received\_data = []

while True:

Client, address = ServerSideSocket.accept()

print('Connected to: ' + address[0] + ':' + str(address[1]))

test = [[3,2,1],[5,4,6],[]]

data1 = test[ThreadCount]

# data1 = []

# for row in range(ThreadCount\*20,ThreadCount\*20+20):

# data1.append(table[row])

# data1 = str(data1)

# data1 += str(table[0]) + "\n"

# for row in table:

# data1 += str(row[ThreadCount\*5:ThreadCount\*5+5]) + "\n"

start\_new\_thread(multi\_threaded\_client, (Client, data1))

ThreadCount += 1

print('Thread Number: ' + str(ThreadCount))

if ThreadCount == 3:

break

# print(received\_data)

print(merge(received\_data[0],received\_data[1]))

ServerSideSocket.close()

Client:

import socket

def client\_program():

host = socket.gethostname() # as both code is running on same pc

print(host)

port = 5000 # socket server port number

client\_socket = socket.socket() # instantiate

client\_socket.connect((host, port)) # connect to the server

message = "hi" # take input

while message.lower().strip() != 'bye':

# client\_socket.send(message.encode()) # send message

data = client\_socket.recv(1024).decode() # receive response

print('Received from server: \n' + data) # show in terminal

msg = input()

# client\_socket.send(msg.encode())

data = eval(data)

print(data)

data.sort()

data = str(data)

print(data)

client\_socket.send(data.encode())

msg = input()

client\_socket.send(data.encode())

client\_socket.close() # close the connection

if \_\_name\_\_ == '\_\_main\_\_':

client\_program()

Output:



  
