Solved-1: Socket programming using connection-oriented approach

 # Echo server program

```
import socket              # Import socket module
host = socket.gethostname() # Get local machine name
port = 1112                # Reserve a port for your service.
s = socket.socket()        # Create a socket object
s.bind((host, port))       # Bind to the port
s.listen(5)                # Now wait for client connection.
conn, addr = s.accept()    # Establish connection with client.
print('Got connection from', addr[0], '(', addr[1], ')')
print('Thank you for connecting')
while True:
    data = conn.recv(1024)
    if not data: break
    conn.sendall(data)
print('Received message is sent back to client')
conn.close()               # Close the connection
```

Output produced at the server terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 server1.py
Got connection from 127.0.0.1 ( 57831 )
Thank you for connecting
Received message is sent back to client
venkatesh@MAHEFATYL0766:~/DSL$
```

 # Echo client program

```
import socket              # Import socket module
host = socket.gethostname() # Get local machine name
port = 1112                # Reserve a port for your service.
s = socket.socket()        # Create a socket object
s.connect((host, port))
s.sendall(b'Welcome user!')
print('Client is connected to server and it has sent the message to server')
data = s.recv(1024)
print('This is printing the message received back by the server')
print(repr(data))
s.close()                  # Close the socket when done
```

Output produced at the client terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 client1.py
Client is connected to server and it has sent the message to server
This is printing the message received back by the server
b'Welcome user!'
venkatesh@MAHEFATYL0766:~/DSL$
```

Execution method

Step 1: Run server.py. It would start a server in background.
Step 2: Run client.py. Once server is started run client.

Solved-2: Socket programming using connection-less approach

```
# Echo server program
import socket
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      # For UDP
udp_host = socket.gethostname()            # Host IP
udp_port = 12345                    # specified port to connect
sock.bind((udp_host,  udp_port))
while True:
  print('Waiting for client...')
  data,addr = sock.recvfrom(1024)          #receive data from client
  print('Received Messages:',data.decode(),'from',addr)
sock.close()
```

Output produced at the server terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 server2.py
Waiting for client...
Received Messages: UDP Program third time! from ('127.0.0.1', 55561)
Waiting for client...
Received Messages: UDP Program third time! from ('127.0.0.1', 55561)
Waiting for client...
^Z
[1]+  Stopped                 python3 server2.py
venkatesh@MAHEFATYL0766:~/DSL$
```

 # Echo client program

```
import socket              # Import socket module
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      # For UDP

udp_host = socket.gethostname()    # Host IP
udp_port = 12345              # specified port to connect

msg = "UDP Program third time!"
print('UDP target IP:', udp_host)
print('UDP target Port:', udp_port)

sock.sendto(msg.encode(),(udp_host,udp_port))
sock.sendto(msg.encode(),(udp_host,udp_port))
sock.close()
```

Output produced at the client terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 client2.py
UDP target IP: MAHEFATYL0766
UDP target Port: 12345
venkatesh@MAHEFATYL0766:~/DSL$
```

Execution method

Step 1: Run server.py. It would start a server in background.

Step 2: Run client.py. Once server is started run client.

Solved-3:

Write a program where client can send a message to the server and the server an receive the message and send, or echo, it back to the client.

```
# Server program: server.py

import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 2053     # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
  s.bind((HOST, PORT))
  s.listen()
  conn, addr = s.accept()

  with conn:
     print('Connected by', addr)
     while True:
        data = conn.recv(1024)
        if  data:
          print("Client: ",data.decode())
          data = input("Enter message to client:");
        if not data:
          break;
        # sending message as bytes to client.
        conn.sendall(bytearray(data,'utf-8'));
  conn.close()
```

Output produced at the server terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 server3.py
Connected by ('127.0.0.1', 57855)
Client:  Hello, world
Enter message to client: Hello I am server
venkatesh@MAHEFATYL0766:~/DSL$
```

```
# Client program: client.py
import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 2053      # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world')
    data = s.recv(1024)
    print('Received Connection')
    print('Server:', data.decode())
```

Output produced at the client terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 client3.py
Received Connection
Server: Hello I am server
venkatesh@MAHEFATYL0766:~/DSL$
```

Execution method

Step 1: Run server.py. It would start a server in background.

Step 2: Run client.py. Once server is started run client.

Solved-4: Write a program to create TCP time server in Python

```
# Server program: Time server

import socket
import time

# create a socket object
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# get local machine name
host = socket.gethostname()

port = 9991

# bind to the port
serversocket.bind((host, port))

# queue up to 5 requests
serversocket.listen(5)

while True:
  # establish a connection
  clientsocket,addr = serversocket.accept()

  print("Got a connection from %s" % str(addr))
  currentTime = time.ctime(time.time()) + "\r\n"
  clientsocket.send(currentTime.encode('ascii'))
  clientsocket.close()
```

Output produced at the server terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 server4.py
Got a connection from ('127.0.0.1', 58576)
^Z
[1]+  Stopped                 python3 server4.py
venkatesh@MAHEFATYL0766:~/DSL$
```

```
# Client program: Time client

#client.py
import socket
# create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# get local machine name
host = socket.gethostname()
port = 9991
# connection to hostname on the port.
s.connect((host, port))
# Receive no more than 1024 bytes
tm = s.recv(1024)
print(' Current time from Sever :', tm.decode())
s.close()
```

Output produced at the client terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 client4.py
Current time from Sever : Sun Mar 20 17:03:39 2022
venkatesh@MAHEFATYL0766:~/DSL$
```

Execution method

Step 1: Run server.py. It would start a server in background.

Step 2: Run client.py. Once server is started run client.

Solved-5: Write a TCP chat server in python using socket programming.

 # Server program: Server Chat

```
import socket
HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 31621    # Port to listen on (non-privileged ports are > 1023)

s = socket.socket()

s.bind((HOST, PORT))

s.listen()
print("\nWaiting for incoming connections...\n")
conn, addr = s.accept()
print("Received connection from ", addr[0], "(", addr[1], ")\n")

s_name = conn.recv(1024)
s_name = s_name.decode()
print(s_name, "has connected to the chat room\nEnter [e] to exit chat room\n")
name = input(str("Enter your name: "))
conn.send(name.encode())

while True:
  message = input(str("Me : "))
  if message == "[e]":
    message = "Left chat room!"
    conn.send(message.encode())
    print("\n")
    break
  conn.send(message.encode())
  message = conn.recv(1024)
  message = message.decode()
  print(s_name, ":", message)
```

Output produced at the server terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 server5.py

Waiting for incoming connections...

Received connection from  127.0.0.1 ( 58599 )

STUDENT has connected to the chat room
Enter [e] to exit chat room

Enter your name: VENKATESH
Me : Hi
STUDENT : Hello sir
Me : ARE YOU FINE WITH DS LAB
STUDENT : YES SIR !
Me : GOOD
STUDENT : Thank you sir.
Me : OK KEEP WORKING..
STUDENT : ok sir.
Me : [e]

venkatesh@MAHEFATYL0766:~/DSL$
```

 # Client program: Client Chat

```
import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 31621    # Port to listen on (non-privileged ports are > 1023)
s = socket.socket()
name = input(str("\nEnter your name: "))

print("\nTrying to connect to ", HOST, "(", PORT, ")\n")

s.connect((HOST, PORT))
print("Connected...\n")

s.send(name.encode())
s_name = s.recv(1024)
s_name = s_name.decode()
print(s_name, "has joined the chat room\nEnter [e] to exit chat room\n")

while True:
   message = s.recv(1024)
   message = message.decode()
   print(s_name, ":", message)
   message = input(str("Me : "))
   if message == "[e]":
      message = "Left chat room!"
      s.send(message.encode())
      print("\n")
      break
   s.send(message.encode())
```

Output produced at the client terminal

```
venkatesh@MAHEFATYL0766:~/DSL$ python3 client5.py

Enter your name: STUDENT

Trying to connect to  127.0.0.1 ( 31621 )

Connected...

VENKATESH has joined the chat room
Enter [e] to exit chat room

VENKATESH : Hi
Me : Hello sir
VENKATESH : ARE YOU FINE WITH DS LAB
Me : YES SIR !
VENKATESH : GOOD
Me : Thank you sir.
VENKATESH : OK KEEP WORKING..
Me : ok sir.
VENKATESH : Left chat room!
Me : [e]
venkatesh@MAHEFATYL0766:~/DSL$
```

Execution method

Step 1: Run server.py. It would start a server in background.

Step 2: Run client.py. Once server is started run client.

Solved-6: Forking/ Threading (Concurrent Server)

// **Tutorial Reference :**
// https://codezup.com/socket-server-with-multiple-clients-model-multithreading-python/

Forking/ Threading (Concurrent Server)

// SERVER PROGRAM

```
import socket
import os
from _thread import *

ServerSocket = socket.socket()
host = '127.0.0.1'
port = 1233
ThreadCount = 0
try:
    ServerSocket.bind((host, port))
except socket.error as e:
    print(str(e))

print('Waitiing for a Connection..')
ServerSocket.listen(5)


def threaded_client(connection):
    connection.send(str.encode('Welcome to the Servern'))
    while True:
        data = connection.recv(2048)
```

```
        reply = 'Server Says: ' + data.decode('utf-8')
        if not data:
            break
        connection.sendall(str.encode(reply))
    connection.close()

while True:
    Client, address = ServerSocket.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(threaded_client, (Client, ))
    ThreadCount += 1
    print('Thread Number: ' + str(ThreadCount))
ServerSocket.close()
```

// CLIENT PROGRAM

```
import socket

ClientSocket = socket.socket()
host = '127.0.0.1'
port = 1233

print('Waiting for connection')
try:
    ClientSocket.connect((host, port))
except socket.error as e:
    print(str(e))

Response = ClientSocket.recv(1024)
while True:
    Input = input('Say Something: ')
    ClientSocket.send(str.encode(Input))
    Response = ClientSocket.recv(1024)
    print(Response.decode('utf-8'))

ClientSocket.close()
```

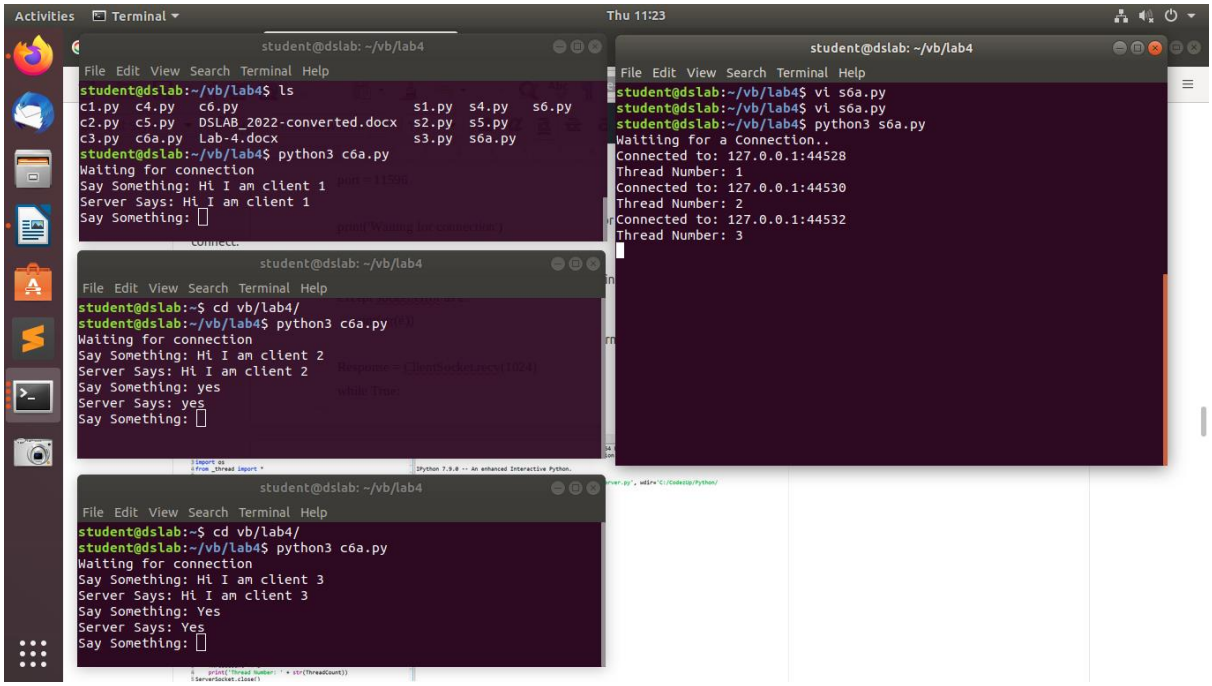
Run Client-Server Model (Execution method)

First, we need to run the Server from our terminal to create a connection or port to which the client can connect.

After running the Server, keep the terminal open and open 3 new terminals to check whether the 3 clients can directly communicate with our Server or not.

After running Client Script in these 3 terminals, you can check Server Terminal you got the 3 threads running in the background with a unique thread number.

Sample output at both client and server side



https://aaronjohn2.github.io/2019/03/03/tcp-udp/