**Networking**

Python is provinding two level of networks servicess acces ,first level is with respect create the sockets ,another is access the newtork apllications like FTP,SSH,HTTP etc.

Now will see the details with respect to socket

**Socket:**

Sockets are endpoints which are used to create the communiaction between within same network or in anothe network

Mainly sockect can develop different number of channels they are unix domain,TCP,UDP etc.

Sockets are having there pre defined library apis and macros.They are

**1.Domain:**

It is kind of protocol family i.e AF\_INET, AF\_UNIX, AF\_LOCAL,AF\_INET6 etc

**2.type:**

It is tell the kind of communication protocol type i.e

SOCK\_STREAM: Transmission Control Protocol(TCP,connection oriented,reliable )

SOCK\_DGRAM: User Datagram Protocol(UDP,connectionless,unreliable)

**3**.protocol:

It is a protocol type ,by default mostly all sockets are used internet procol(IP),value is '0'

4.hostname:

Which is string variable like domain names,ip addresses,loop back addresses etc.

But in network we have some pre defined macros they are

INADDR\_BROADCAST

INADDR\_ANY

**5.port:**

port is nothing but it is fixed integer number,which is used to connect a particular application like HTTP ,FTP ets

HTTP port number is 80

FTP port number is 21

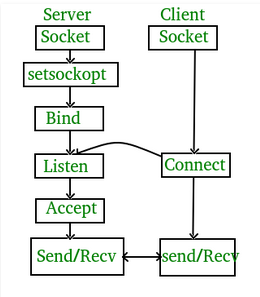
**import socket module:**

import socket

**Syntax to create a socket:**

socket= socket.socket (socket\_family, socket\_type, protocol=0)

**Schematci diagram between server and client:**

****

**Below are the server socket methods:**

**1.socket.bind()**

This method is used to binds the address (hostname,port number ) to socket.

**2.socket.listen()**

This method is used to create a setup and start TCP listener.

**3.socket.accept()**

This method is used to accept TCP client connection, waiting until connection arrives (blocking).

**Below are the client socket methods:**

**socket.connect()**

This method is used to initiates communication protocol server connection.

**Comman socket methods:**

**1.socket.recv()**

This method is used to receives Transmission Control Protocol message

**2.socket.send()**

This method is used to receives transmits Transmission Control Protocol message

**3.socket.recvfrom()**

This method is used to receives receives User Datagram Protocol message

**4.socket.sendto()**

This method is used to receives transmits User Datagram Protocol message

**5.socket.close()**

This method is used to receives closes socket

**6.socket.gethostname()**

Returns the hostname.

**https://www.journaldev.com/15906/python-socket-programming-server-client**

**Ex:server.py**

import socket

def server\_program():

# get the hostname

host = socket.gethostname()

port = 5000 # initiate port no above 1024

server\_socket = socket.socket() #Creating the socket

# look closely. The bind() function takes tuple as argument

server\_socket.bind((host, port)) # bind host address and port together

# configure how many client the server can listen simultaneously

server\_socket.listen(2)

conn, address = server\_socket.accept() # accept new connection

print("Connection from: " + str(address))

while True:

# receive data stream. it won't accept data packet greater than 1024 bytes

data = conn.recv(1024).decode()

if not data:

# if data is not received break

break

print("from connected user: " + str(data))

data = input(' -> ')

conn.send(data.encode()) # send data to the client

conn.close() # close the connection

server\_program()

**Ex:client.py**

import socket

def client\_program():

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

port = 5000 # socket server port number

client\_socket = socket.socket() # instantiate

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

message = input(" -> ") # 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: ' + data) # show in terminal

message = input(" -> ") # again take input

client\_socket.close() # close the connection

client\_program()

Explantion:

1.open two new terminals,in one terminal you can execute the server script,in another terminal you can execute the client script once you both executed start the chatting