Refer to tutorialspoint,

geeks and bogotobogo and other things for networking in Python and https://docs.python.org/3.0/library/socket.html

Programming abstraction for network code

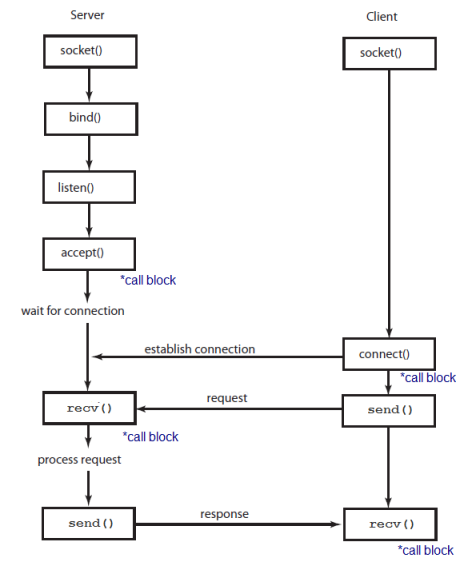
• Socket: A communication endpoint 12 socket socket

• Supported by socket library module •

Allows connections to be made and data to be transmitted in either direction

Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The *socket* library provides specific classes for handling the common transports as well as a generic interface for handling the rest.



he socket Module

To create a socket, you must use the socket.socket() function available in socket module, which has the general syntax −

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

Here is the description of the parameters −

socket\_family − This is either AF\_UNIX or AF\_INET, as explained earlier.

socket\_type − This is either SOCK\_STREAM or SOCK\_DGRAM.

protocol − This is usually left out, defaulting to 0.

Once you have socket object, then you can use required functions to create your client or server program. Following is the list of functions required −

Server Socket Methods

Sr.No. Method & Description

1

s.bind()

This method binds address (hostname, port number pair) to socket.

2

s.listen()

This method sets up and start TCP listener.

3

s.accept()

This passively accept TCP client connection, waiting until connection arrives (blocking).

Client Socket Methods

Sr.No. Method & Description

1s.connect()

This method actively initiates TCP server connection.

General Socket Methods

Sr.No. Method & Description

1 s.recv()

This method receives TCP message

2

s.send()

This method transmits TCP message

3

s.recvfrom()

This method receives UDP message

4

s.sendto()

This method transmits UDP message

5

s.close()

This method closes socket

6

socket.gethostname()

Returns the hostname.

A Simple Server

To write Internet servers, we use the socket function available in socket module to create a socket object. A socket object is then used to call other functions to setup a socket server.

Now call bind(hostname, port) function to specify a port for your service on the given host.

Next, call the accept method of the returned object. This method waits until a client connects to the port you specified, and then returns a connection object that represents the connection to that client.

#!/usr/bin/python # This is server.py file

import socket # Import socket module

s = socket.socket() # Create a socket object

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

port = 12345 # Reserve a port for your service.

s.bind((host, port)) # Bind to the port

s.listen(5) # Now wait for client connection.

while True:

c, addr = s.accept() # Establish connection with client.

print 'Got connection from', addr

c.send('Thank you for connecting')

c.close() # Close the connection

A Simple Client

Let us write a very simple client program which opens a connection to a given port 12345 and given host. This is very simple to create a socket client using Python's socket module function.

The socket.connect(hosname, port ) opens a TCP connection to hostname on the port. Once you have a socket open, you can read from it like any IO object. When done, remember to close it, as you would close a file.

The following code is a very simple client that connects to a given host and port, reads any available data from the socket, and then exits −

#!/usr/bin/python # This is client.py file

import socket # Import socket module

s = socket.socket() # Create a socket object

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

port = 12345 # Reserve a port for your service.

s.connect((host, port))

print s.recv(1024)

s.close # Close the socket when done

Now run this server.py in background and then run above client.py to see the result.

# Following would start a server in background.

$ python server.py &

# Once server is started run client as follows:

$ python client.py

This would produce following result −

Got connection from ('127.0.0.1', 48437)

Thank you for connecting