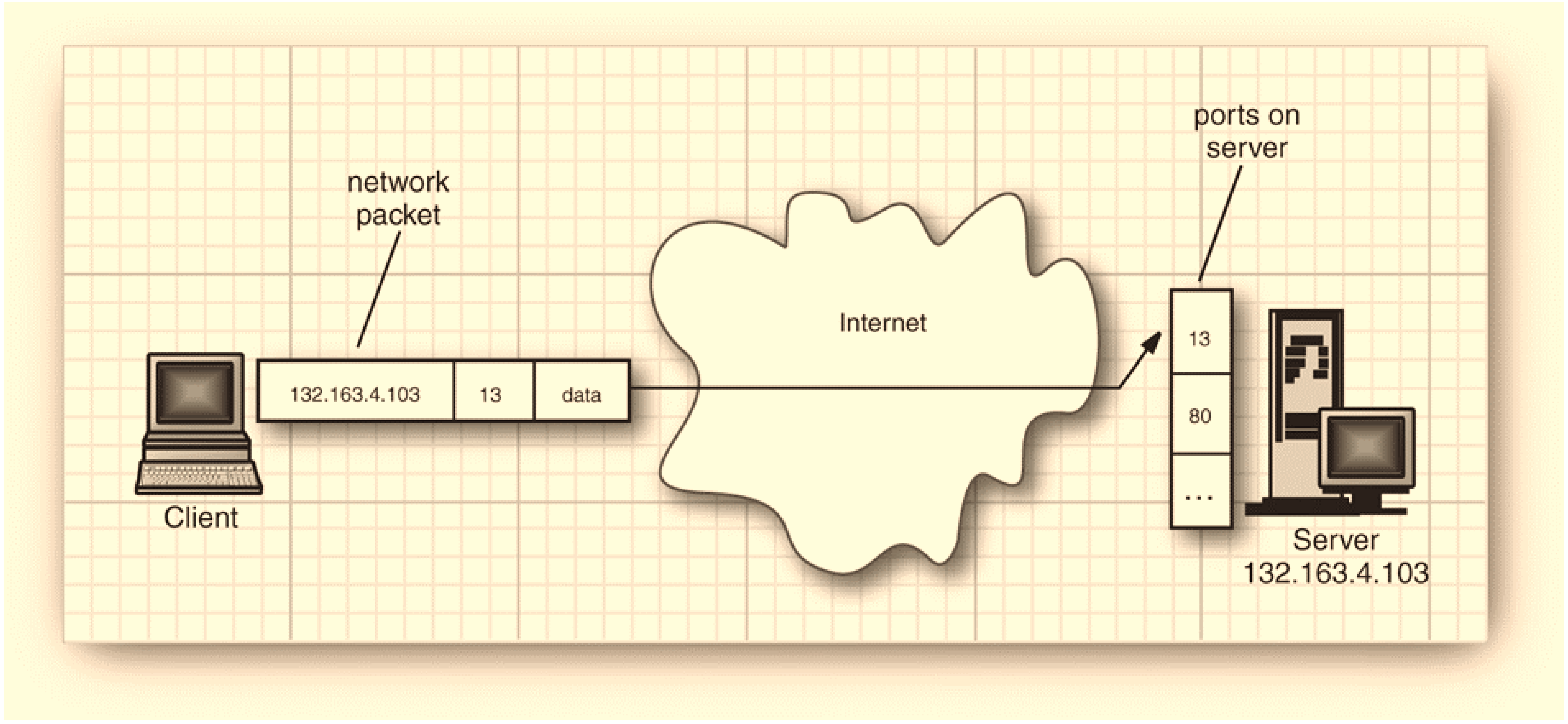
NETWORKING

SOCKET PROGRAMMING


network
packet
132.163.4.103
13
data
Client
Internet
ports on
server
13
80
...
Server
132.163.4.103

Overview

- Sockets are the end-point of a two-way communication link. An endpoint is a combination of IP address and the port number.
- Sockets allow communication between processes that lie on the same machine, or on different machines.
- For Client-Server communication, sockets are to be configured at the two ends to initiate a connection:
  - listen for incoming messages
  - send the responses at both ends

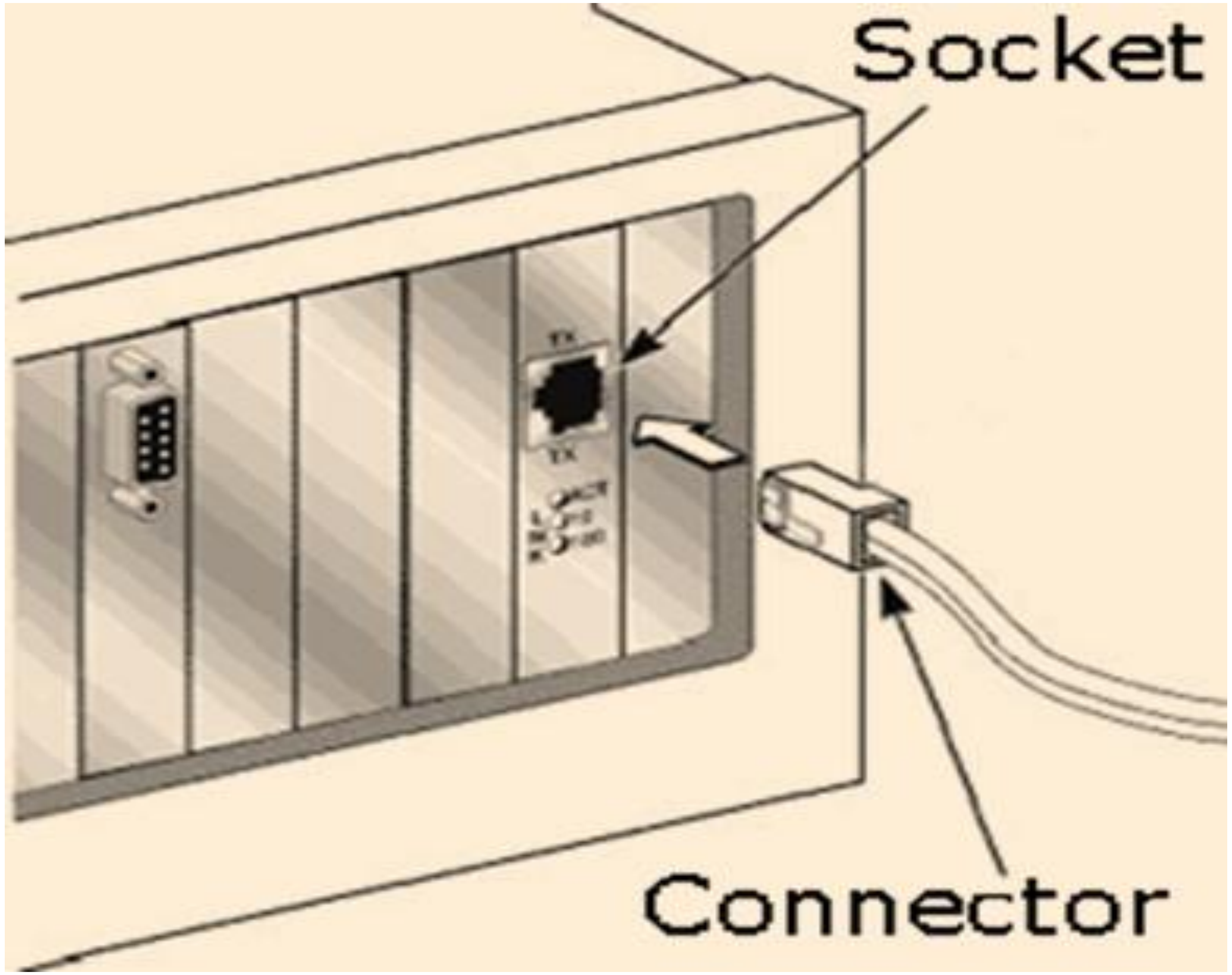
Terminology

- *A Socket* is an *abstraction* of a "communications link" between machines over some network.

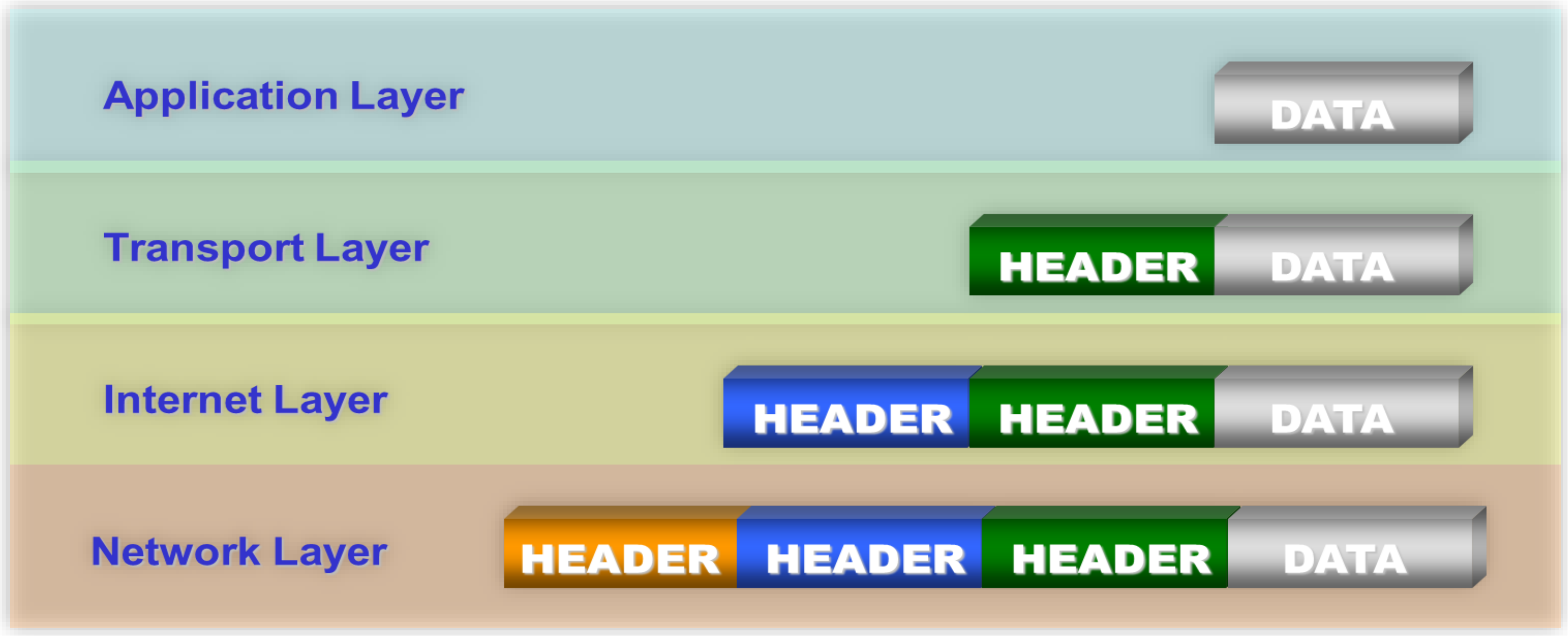
- *Socket* communication is the same *regardless* of whether the network connection is via a phone line, cable modem, ethernet, or fiber-optic line.

- *A Packet* is a discrete quantity of information suitable for routed transport over a shared network.

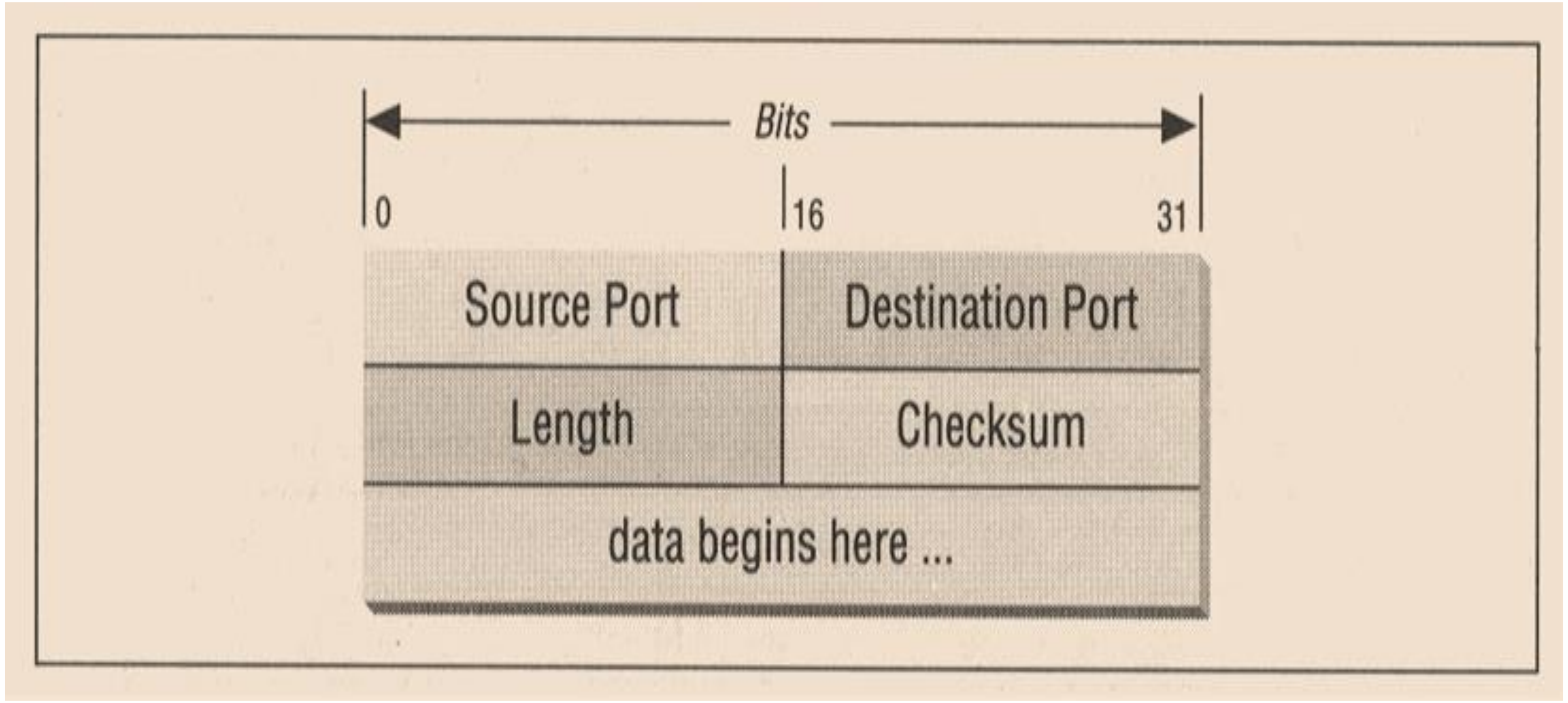
- *Packet* sizes are limited, so a packet may be a fragment of a large file or message.


Socket
Connector

Communication Architecture



User Data Protocol (UDP/TCP)



Common Connections

Service	Port no.
echo	7
daytime	13
ftp	21
telnet	23
smtp	25
finger	79
http	80
pop3	110

Create a socket object in Python

```
sock_obj = socket.socket( socket_family, socket_type, protocol=0)
```

- Socket_family: Either AF_UNIX, IPv4 or IPv6
- Socket_type: TCP or UDP
- Protocol: Typically efault this field to zero.

Connect Example

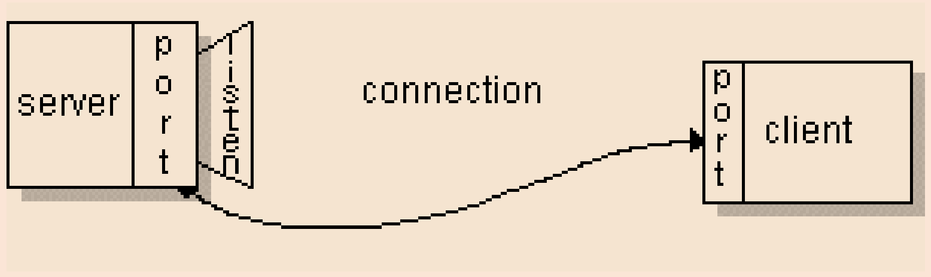
```
import socket   #for sockets
import sys  #for exit

try:
    #create an AF_INET, STREAM socket (TCP)
    sock_obj = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error as err:
    print ('Error code: ' + str(err[0]) + err_msg[1])
    sys.exit()

print ('Socket Initialized')
```

Client Socket

- The client knows the hostname of the server and the port number on which the server is listening.
- To connect, the client tries to “handshake” with the server on the server's machine and port. The client identifies itself to the server to bind to a local port number to the connection.

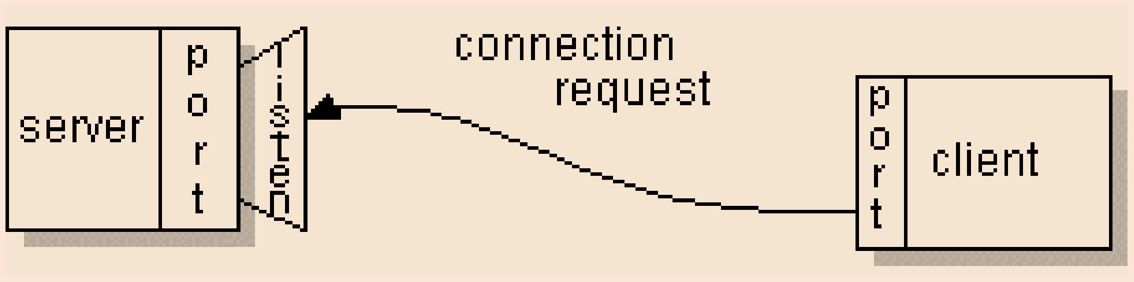


Client Socket Methods

- sock_object.bind(address)
- sock_object.listen(logfile)
- sock_object.accept()

Server Socket

- A server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.



Server Socket Methods

- sock_object.connect()

General Socket Methods

- sock_object.recv() *TCP*
- sock_object.send() *TCP*
- sock_object.recvfrom() *UDP*
- sock_object.sendto() *UDP*
- sock_object.gethostname()
- sock_object.close()

Server Example

```
def Server(host,port):
		mySocket = socket.socket()
		mySocket.bind((host,port))
		mySocket.listen(1)
		conn, addr = mySocket.accept()
		while True:
				data = conn.recv(1024).decode()
		if not data:
				break
		data = str(data).upper()
		data = input(" ? ")
		conn.send(data.encode())
		conn.close()
```

Client Example

```
def Client(host,port):
        mySocket = socket.socket()
        mySocket.connect((host,port))
        message = input(" ? ")
        mySocket.send(message.encode())
        data = mySocket.recv(1024).decode()
        mySocket.close()
```