

Socket programming :

Deb 1 : server:1.113

Deb2 : client :1.142

When we call open function it returns an object which is stored in variable , we need socket module.

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

They are the real backbones behind web browsing. In simpler terms, there is a server and a client.

Server :

1.Import the socket :

```
>dir()
>import socket
>dir()
>dir(socket) (check socket module inside socket)
```

```
ethostbyname_ex', 'gethostname', 'getnameinfo', 'getprotobyname', 'getservbyname',
', 'getservbyport', 'has_ipv6', 'herror', 'htonl', 'htons', 'if_indextoname', 'i
f_nameindex', 'if_nametoindex', 'inet_aton', 'inet_ntoa', 'inet_ntop', 'inet_pto
n', 'io', 'ntohl', 'ntohs', 'os', 'selectors', 'setdefaulttimeout', 'sethostname
', 'socket', 'socketpair', 'sys', 'timeout']
>>>
```

2. Create the socket .

```
>TCPSocket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
AF_INET is the Internet address family for ipv4 . SOCK_STREAM is the socket type
for TCP, the protocol that will be used to transport messages in the network.
```

3.Bind the socket :

Bind takes two output and in tuple
>TCPSocket.bind(("0.0.0.0.",Port))
0.0.0.0 means it listens in all IP and port can be any

4. Listen for connection

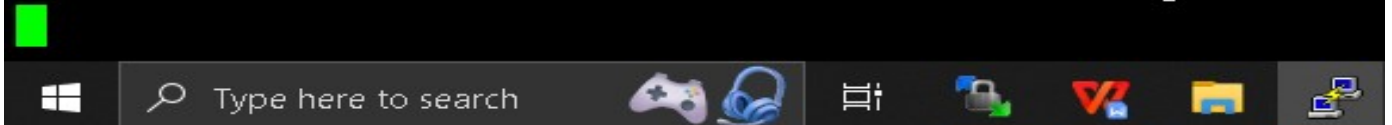
```
>TCPSocket.listen()
```

5. Accept

```
>(clientsocket,(clientip,clientport))=TCPSocket.accept()
```

In my practical I have given var1,var2,var3

```
>>> TCPSocket.listen()
>>> (var1,(var2,var3))=TCPSocket.accept()
```



As accept returns two things in a **tuple** first thing is client socket address and the other second thing itself consists of two things ip and port. Hence we store objects returned by accept into a variable.

No prompt will appear until connection is established

Client :

\$sudo apt-get install telnet

\$telnet server ip port

```
inet) in auto mode
shuhari@debian:~$ telnet 192.168.1.113 1314
Trying 192.168.1.113...
Connected to 192.168.1.113.
Escape character is '^]'.
```

At server side check what all is stored in var1, var2 and var3

```
shuhari@debian: ~
>>> print(var1)
<socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('192.168.1.113', 1314), raddr=('192.168.1.142', 37654)>
>>> print(var2)
192.168.1.142
>>> print(var3)
37654
>>>
```

Message customization :

6. Server side :

>>> var1.send(b"\nHey Buddy\n")

b=Python data has to be binary

Returns length also

```
Escape character is '^]'.
```

```
Hey Buddy
hi i am omkar
```

```
Hey Buddy how are you
i am good
```

>>> data = ClientSocket.recv(2048)

Go to client and type any msg

>>> print(data)

```
>>> var1.send(b"\n Hey Buddy how are you\n")
24
>>> data=var1.recv(2048)
>>> print(data)
b'i am good\r\n'
```

7.To close :

```
>>>var1.close
```

Connection is close at client side but server is in still listening state

To close listening as well

```
>>>TCPSocket.close()
```

```
>>> var1.close
<bound method socket.close of <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('192.168.1.113', 1314), raddr=('192.168.1.142', 37654)>>
>>> TCPSocket.close
<bound method socket.close of <socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('0.0.0.0', 1314)>>
>>>
```

Windows Telnet and Linux Telnet :

```
>>>(winsocket,(winip,winport))=TCPSocket.accept()
```

```
>>>winsocket.send(b"\n msg \n")
```

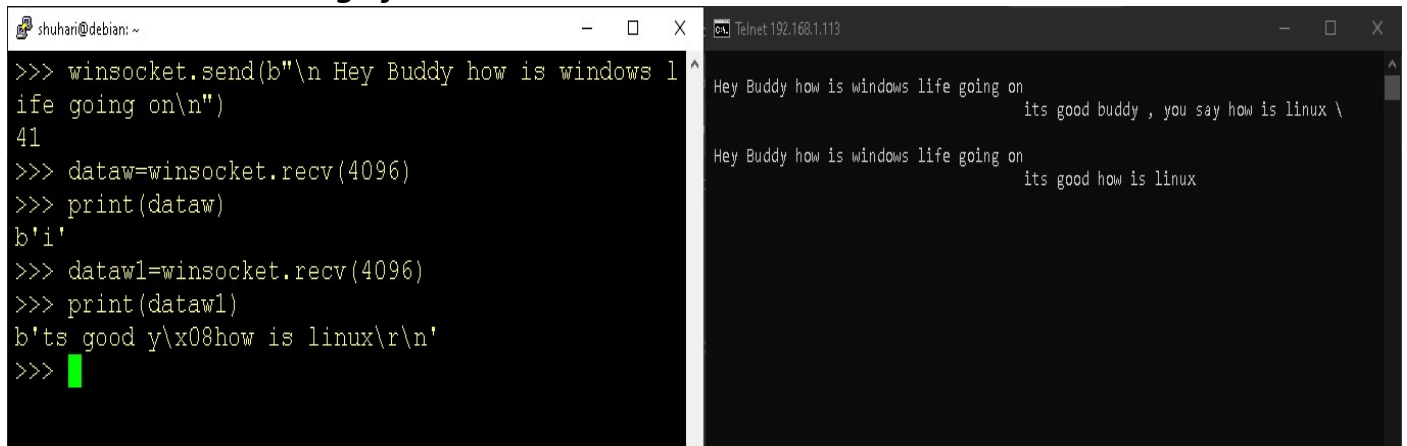
Msg received at client side

```
>>>data=winsocket.recv(4096)
```

Type msg at client side

```
>>>print(data)
```

Show msg of client



```
shuhari@debian:~
>>> winsocket.send(b"\n Hey Buddy how is windows life going on\n")
41
>>> dataw=winsocket.recv(4096)
>>> print(dataw)
b'i'
>>> dataw1=winsocket.recv(4096)
>>> print(dataw1)
b'ts good y\x08how is linux\r\n'
>>>
```

```
Telnet 192.168.1.113
Hey Buddy how is windows life going on
its good buddy , you say how is linux \
Hey Buddy how is windows life going on
its good how is linux
```

Socket syntax :

Socket takes 3 arguments

Socket(socket_family, socket_type, protocols)

Socket_family is either :

RPC and LPC:

Remote procedure call and local procedure call

AF_UNIX or : The **AF_UNIX** (also known as **AF_LOCAL**) socket family is used to communicate between processes on the same machine efficiently.

AF_INET : **AF_INET** is the Internet address family for **ipv4**

Socket_type:

SOCK_STREAM (TCP)

SOCK_DGRAM (UDP)

A socket is an endpoint in communication between networks, and socket programming enables these endpoints to transfer data, thereby supporting communication between networks and programs.

Remote procedure call:

In distributed computing , a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in a different address space (commonly on another computer on a shared network), which is written as if it were a normal (local) procedure call, without the programmer explicitly writing the details for the remote interaction.

Local Procedure call:

The message-passing facility in Windows is called the local procedure call (LPC) facility. LPC is used for communication between two processes on the same machine. It is similar to the standard remote procedure call (RPC) mechanism that is widely used, but it is optimized for and specific to Windows. Windows uses a port object to establish and maintain a connection between two processes, like Mach. Two types of ports are used by Windows: connection ports and communication ports.