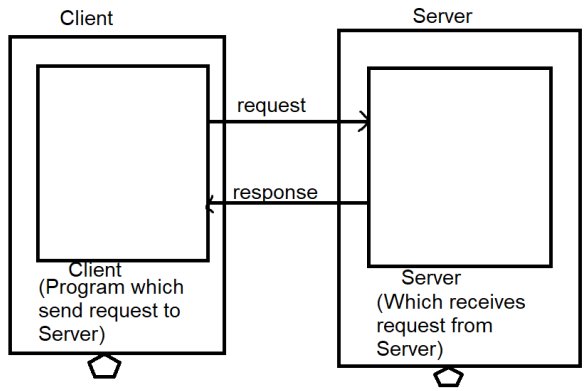
Networking (socket module)

Python is a general purpose programming language; this language is used to develop different types of applications. Python is used for developing networking applications or network enabled applications.

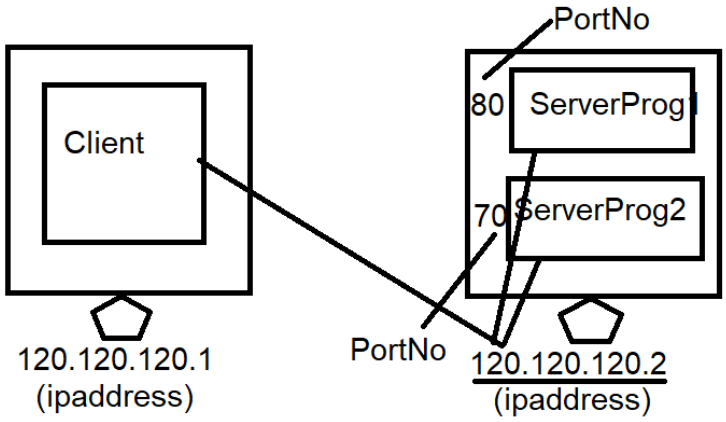
Network application required two programs.

1. Client program
2. Server program

Networking is used to share resources. These resources can be hardware or software.



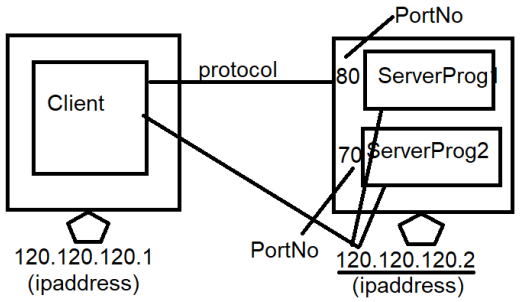
Socket module allows developing client program and server program.



What is ip-address?

In networking each system is identified with unique number called ip-address. This ip-address is given by system admins.

What is portno?
Each server program is identified with unique number called portno. This portno is given by a person who develop server program.



What is protocol?
A protocol defines set of rules and regulations to have communication between two programs.
HTTP, FTP,TCP,UDP,POP,….

Socket module is used to develop client program and server program.

Socket module provides predefined protocols to develop network enabled applications.

What is socket?
A socket is end-point communication between two programs.
A socket is an implementation of client program and server program.

class socket.socket(family=AF_INET, type=SOCK_STREAM)

Create a new socket using the given address family, socket type.
There two socket types

1. SOCK_STREAM → TCP (Transmission Control Protocol)
2. SOCK_DGRAM → UDP (User Datagram Protocol)

TCP implementation is connection oriented
UDP implementation is connectionless.

Methods of socket class

socket.accept()
Accept a connection. The socket must be bound to an address and listening for connections

socket.bind(*address*)
Bind the socket to *address*. The socket must not already be bound.

socket.close()
Mark the socket closed.

socket.connect(*address*)
Connect to a remote socket at *address*.

socket.recv(*bufsize*[, *flags*])
Receive data from the socket. The return value is a bytes object representing the data received

socket.send(*bytes*[, *flags*])
Send data to the socket. The socket must be connected to a remote socket.

socket.listen([*backlog*])
Enable a server to accept connections.

Open two IDLE

1. In one IDLE run server program
2. In another IDLE run client program

Example:

```
# Server program

import socket

s=socket.socket()
s.bind(("localhost",60))
s.listen(10)
print("Server is running...")
s.accept() # to accept connection of client
print("Client connection accepted...")
```

Example:

```
# Client Program
```

```
import socket

s=socket.socket()
s.connect(("localhost",60))
```

Example:

```
# Server program which receives data from client

import socket

s=socket.socket()
s.bind(("localhost",50))
s.listen(10)
print("Server Running...")
t=s.accept() # tuple contains connection of client (socket) (socket,address)
c=t[0] # connection of client
b=c.recv(1024) # 1024bytes --> 1kb
print(b.decode())
s.close()
```

Example

```
# Client Program

import socket

s=socket.socket()
s.connect(("localhost",50))
msg="Hello Server"
s.send(msg.encode()) # converting string to bytes
```

Example:

```
# Calc Server/Math Server

import socket

s=socket.socket()
s.bind(("localhost",40))
s.listen(10)
print("Math Server is Running")
```

```
t=s.accept()
c=t[0]
b=c.recv(1024)
cmd=b.decode()
res=eval(cmd)
res="Result is "+str(res)
b=res.encode()
c.send(b)
s.close()
```

Example:

```
# Math Client

import socket

s=socket.socket()
s.connect(("localhost",40))
cmd=input("Enter Command ")
cmd=cmd.encode()
s.send(cmd)
b=s.recv(1024)
res=b.decode()
print(res)
s.close()
```

numpy and pandas
tkinter (GUI) using backend database