Networking Programming using Python

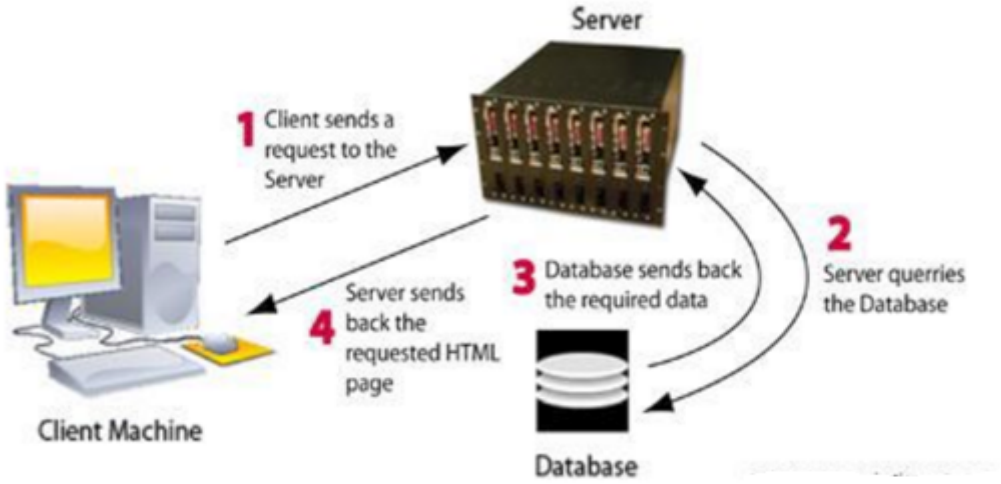
Computer Network: A computer network is a set of connected computers. Computers on a network are called **nodes**. The connection between computers can be done via cabling, or wirelessly through radio waves. Connected computers can share resources, like access to the Internet, printers, file servers, and others. A network is a multipurpose connection, which allows a single computer to do more.

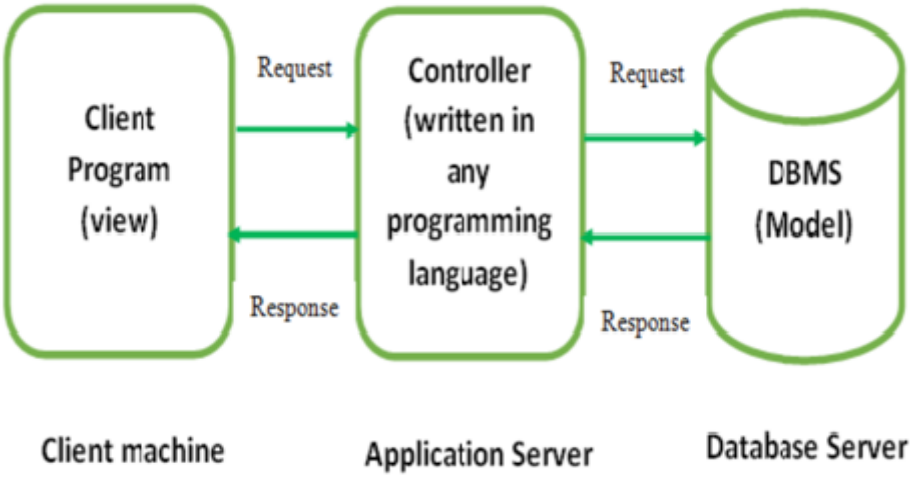
Socket

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 other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server. They are the real backbones behind web browsing. In simpler terms there is a server and a client. Socket programming is started by importing the socket library and making a simple socket.

Client:A client is the receiving end of a service or the requestor of a service in a client/server model type of system.

Server: A server is a machine that waits for client requests and serves or processes them.This is one or more multi-user processors with a higher capacity of shared memory which provides connectivity and the database services along with interfaces relevant to the business procedures.





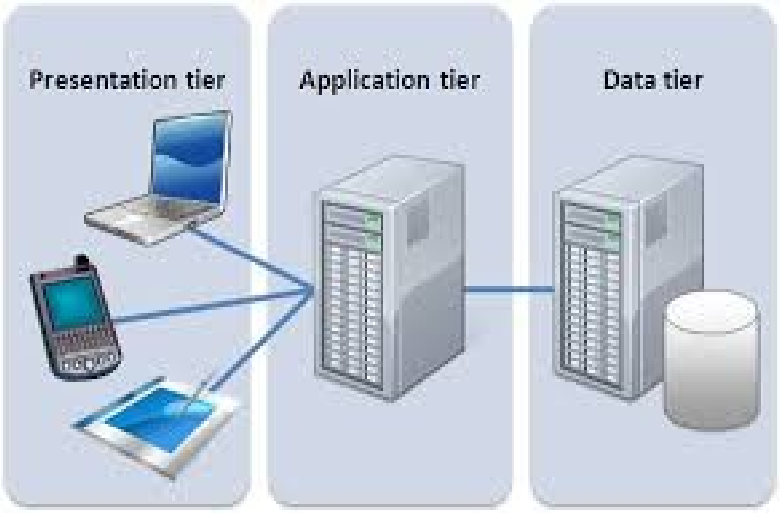
Client Side vs. Server Side

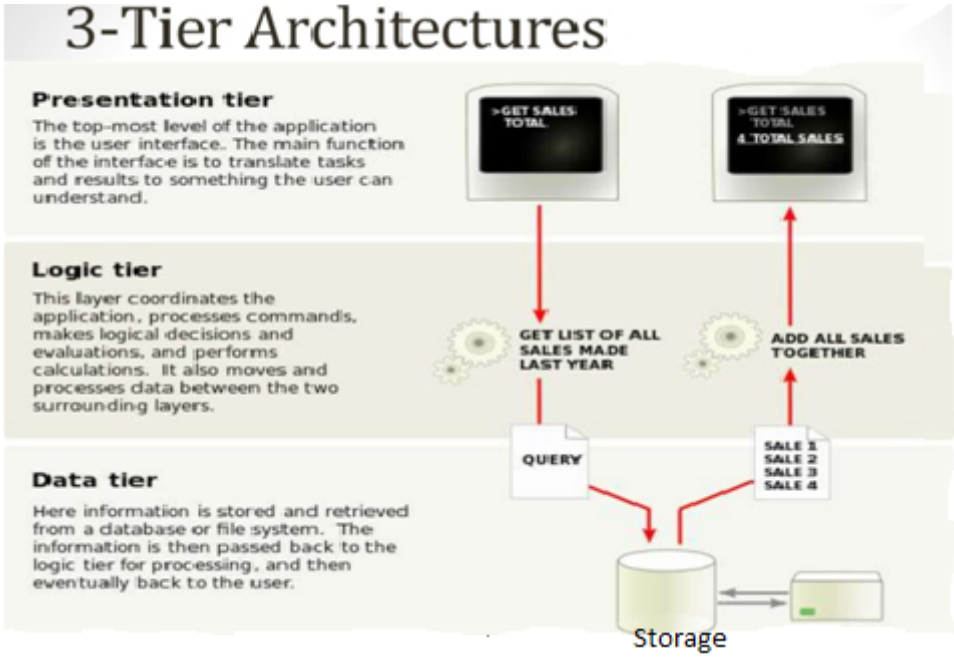
When a client (your computer) makes a request for a web page that information is processed by the web server. If the request is a server side script (e.g. Perl or PHP) before the information is returned to the client the script is executed on the server and the results of the script is returned to the client.

Once the client recieves the returned information from the server if it contains a client side script (e.g. JavaScript) your computer browser executes that script before displaying the web page.

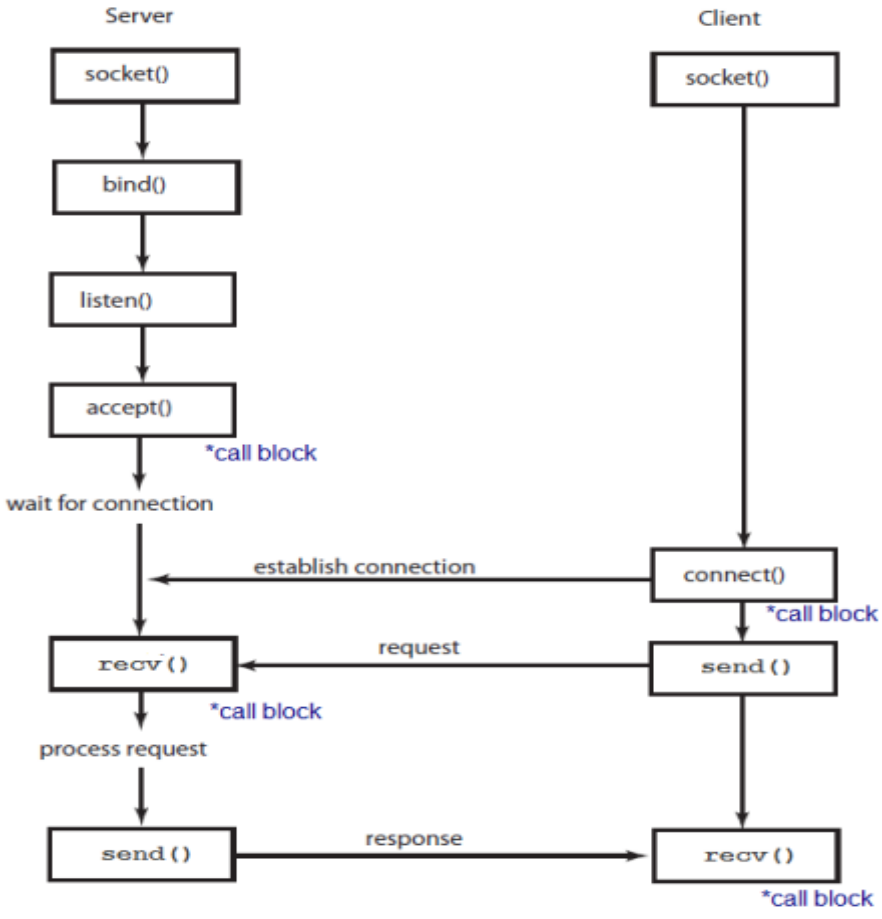
Three-Tier Architecture

Client -server architecture we can call it as a three-tier architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. Three-tier architecture is a software design pattern and a well-established software architecture.





we are building a simple server and a client where server will open a socket and wait for clients to connect. Once client is connected he can send a message and server will process the message and reply back the same message but converted into upper case to demonstrate the server side processing and transmission.



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

Method	Description
s.bind()	This method binds address (hostname, port number pair) to socket.
s.listen()	This method sets up and start TCP listener.
s.accept()	This passively accept TCP client connection, waiting until connection arrives (blocking).

Client Socket Methods

Method	Description
s.connect()	This method actively initiates TCP server connection.

General Socket Methods

Method	Description
s.recv(recv_size)	This method receives TCP message
s.send()	This method transmits TCP message
s.recvfrom()	This method receives UDP message
s.sendto()	This method transmits UDP message
s.close()	This method closes socket
socket.gethostname()	Returns the hostname.

Note:*TCP(Transmission Control Protocol)* works with the Internet Protocol (IP), which defines how computers send packets of data to each other.User Datagram Protocol (**UDP**)is used to send short messages

We can connect to a server using this socket.Socket programming is started by importing the socket library and making a simple socket.

Connecting to a server:

```
# An example script to connect to Google server using socket
# programming in Python
import socket # for socket
import sys
s = socket.socket()
print "Socket successfully created"
port = 80
try:
```

```
    host_ip = socket.gethostbyname('hub.rgukt.ac.in')#to know the server ip
except socket.gaierror:
    # this means could not resolve the host
    print "there was an error resolving the host"
    sys.exit()
# connecting to the server
s.connect((host_ip, port))
print "the socket has successfully connected to rgukt hub port no {}".format(host_ip)
```

Output:

```
Socket successfully created
the socket has successfully connected to google on port == 172.217.26.228
```

- First of all we made a socket.
- Then we resolved google's ip and lastly we connected to google.
- Now we need to know how can we send some data through a socket.
- For sending data the socket library has a *sendall* function. This function allows you to send data to a server to which the socket is connected and server can also send data to the client using this function.

A simple server-client program :

Server :
A server has a bind() method which binds it to a specific ip and port so that it can listen to incoming requests on that ip and port.A server has a listen() method which puts the server into listen mode. This allows the server to listen to incoming connections. And last a server has an accept() and close() method. The accept method initiates a connection with the client and the close method closes the connection with the client.

Example

```
# first of all import the socket library

import socket

# next create a socket object

s = socket.socket()

print "Socket successfully created"

# reserve a port on your computer in our

# case it is 12345 but it can be anything

port = 12345

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

#s.bind((host, port))

# Next bind to the port
```

```
# we have not typed any ip in the ip field
# instead we have inputted an empty string
# this makes the server listen to requests
# coming from other computers on the network
s.bind(('', port))#bind takes a Tuple only
print "socket binded to %s" %(port)
# put the socket into listening mode
s.listen(5)
print "socket is listening"
# a forever loop until we interrupt it or
# an error occurs
while True:
  # Establish connection with client.
  c, addr = s.accept()#C is the socket object which holds the ip and port
  print c
  print 'Got connection from', addr
  # send a thank you message to the client.
  c.send('Thank you for connecting')
  # Close the connection with the client
  c.close()
```

- First of all we import socket which is necessary.
- Then we made a socket object and reserved a port on our pc.
- After that we binded our server to the specified port. Passing an empty string means that the server can listen to incoming connections from other computers as well. If we would have passed 127.0.0.1 then it would have listened to only those calls made within the local computer.
- After that we put the server into listen mode.5 here means that 5 connections are kept waiting if the server is busy and if a 6th socket trys to connect then the connection is refused.
- At last we make a while loop and start to accept all incoming connections and close those connections after a thank you message to all connected sockets.

Client :

Now we need something with which a server can interact. We could believe to the server like this just to know that our server is working. Type these commands in the terminal:

```
# Import socket module

import socket

# Create a socket object

s = socket.socket()

# Define the port on which you want to connect

port = 12345

# connect to the server on local computer

s.connect(('127.0.0.1', port))

# receive data from the server

print s.recv(1024)

# close the connection

s.close()
```

- First of all we make a socket object.
- Then we connect to localhost on port 12345 (the port on which our server runs) and lastly we receive (Maximum 1024 bytes)data from the server and close the connection.
- Now save this file as client.py and run it from the terminal after starting the server script.

Ouput

```
# start the server:
$ python server.py
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 52717)

# start the client:
$ python client.py
Thank you for connecting
```

Example 2

Server

```
import socket
host="127.0.0.1"
port=5000
s=socket.socket()
s.bind((host,port))
s.listen(1)
```

```
c,addr=s.accept()
print"Cnnection froom",(addr)
while True:
                data=c.recv(1024)
                if not data:
                        break
                print "from connected user:",(data)
                data=(data).upper()

                print "sending ",(data)
                c.send(data)
c.close()
```

Client

```
import socket
host="127.0.0.1"
port=5000
s=socket.socket()
s.connect((host,port))
message=raw_input("Type here:- ")
while message !="q":
                s.send(message)
                data=s.recv(1024)
                print "reveied from server: ",(data)
                message=raw_input("Type here:-  ")
s.close()
```