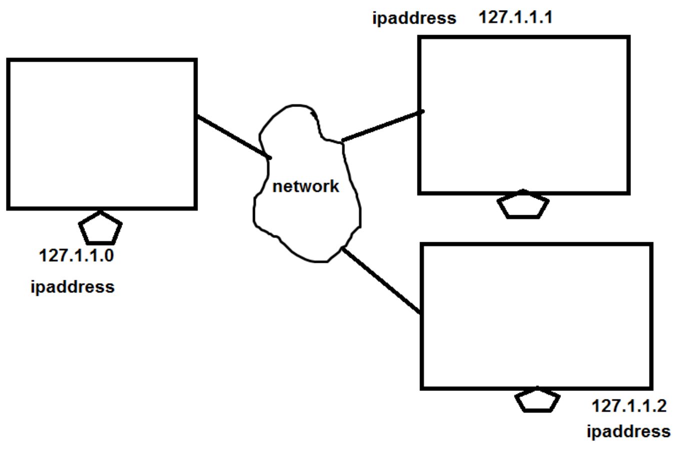
Networking or Socking Programming

Python provides a predefined module called “socket”, which is used for developing networking applications.

Q: What is networking?
Networking is logical or physical link between two or more devices is called networking.
Advantage of networking is sharing resources; these resources can hardware resources or software resources.

Q: What is ipaddress?
Every system in networking is given unique identity which is called ipaddress.

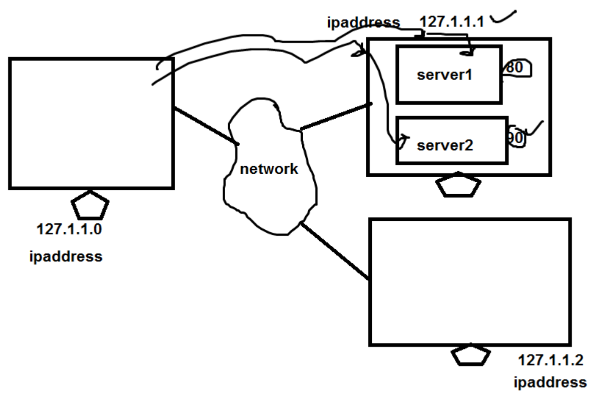


Q: What is protocol?
Protocol define set of rules and regulations to exchange data between two applications with network.
Application Protocols: HTTP,TCP,UDP,SMTP,POP,…

Q: What portno?
In networking there will be two programs.

1. Client
2. Server

Each server program is identified with one unique number called portno.



What is socket?
A socket is an endpoint communication between two programs.

socket module provides two protocols in order to develop networking applications.

1. TCP (Transmission Protocol)
2. UDP (User Datagram Protocol)

The communication between two programs in networking can be connection oriented or connectionless.
TCP communication is connection oriented
UDP communication is connection less.

TCP implementation

socket.socket(*family=AF_INET*, ***type=SOCK_STREAM***, *proto=0*, *fileno=None*)
if type=SOCK_STREAM it is TCP implementation.
If type=SOCK_DGRAM it is UDP implementation.

socket.bind(*address*)
Bind the socket to *address*. The socket must not already be bound
This address consists of two things.

1. Ipaddress of system
2. Portno

At the time developing server program, we create socket and bind with portno and ipaddress of system.

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

socket.listen([*backlog*])
Enable a server to accept connections. If *backlog* is specified, it must be at least 0 (if it is lower, it is set to 0); it specifies the number of unaccepted connections that the system will allow before refusing new connections.

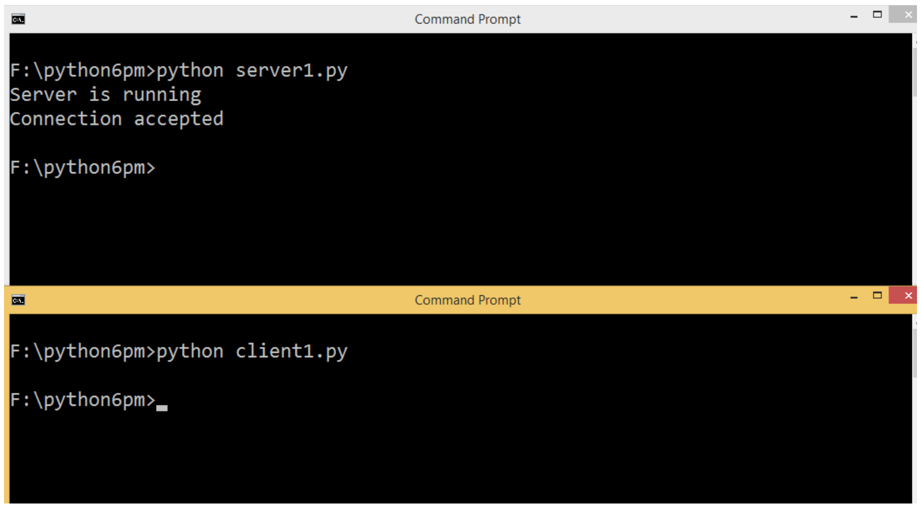
socket.connect(*address*)
Connect to a remote socket at *address*.

Example:

```
# server program
import socket
def main():
    s=socket.socket()
    s.bind(("localhost",60))
    print("Server is running")
    s.listen(5)
    cs=s.accept()
    print("Connection accepted")
main()
```

Example:

```
# client program
import socket
def main():
    s=socket.socket()
    s.connect(("localhost",60))
main()
```



```
F:\python6pm>python server1.py
Server is running
Connection accepted

F:\python6pm>
```

```
F:\python6pm>python client1.py

F:\python6pm>
```

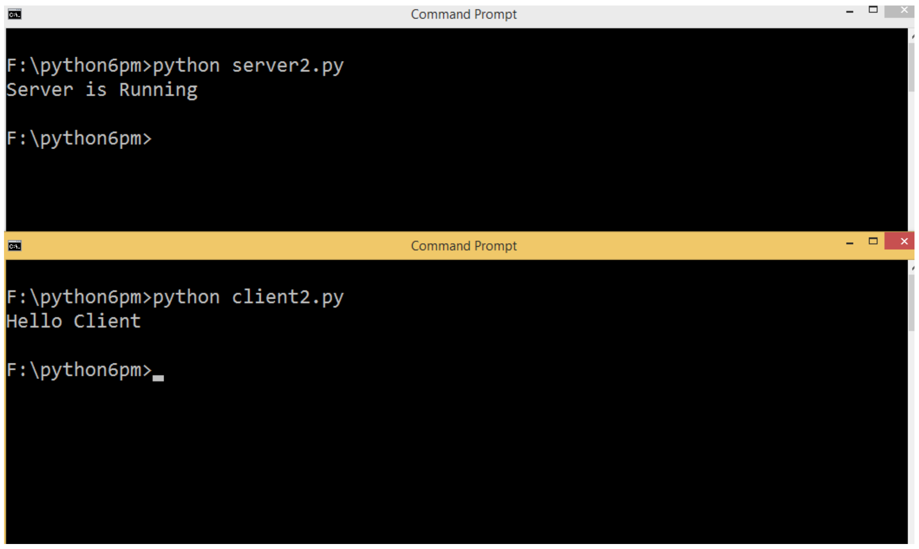
socket.send(*bytes)*
Send data to the socket. The socket must be connected to a remote socket.
socket.recv(bufsize)
Receive data from the socket. The return value is a bytes object representing the data received

Example:

```
# server which send message to client
import socket
def main():
   s=socket.socket()
   s.bind(("localhost",50))
   s.listen(10)
   print("Server is Running")
   sc,add=s.accept() # accept connection of client/server hold address of
client
msg=”Hello Client”
b=msg.encode()
   sc.send(b)
main()
```

Example:

```
# client program
import socket
def main():
   s=socket.socket()
   s.connect(("localhost",50))
   data=s.recv(1024)
   print(data.decode())
main()
```



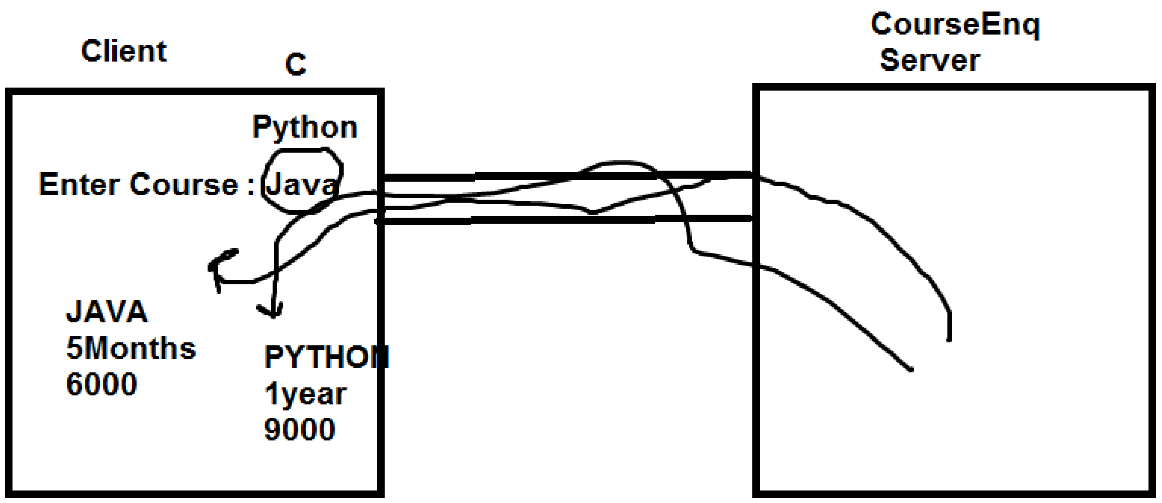
```
F:\python6pm>python server2.py
Server is Running

F:\python6pm>
```

```
F:\python6pm>python client2.py
Hello Client

F:\python6pm>_
```

Case Study



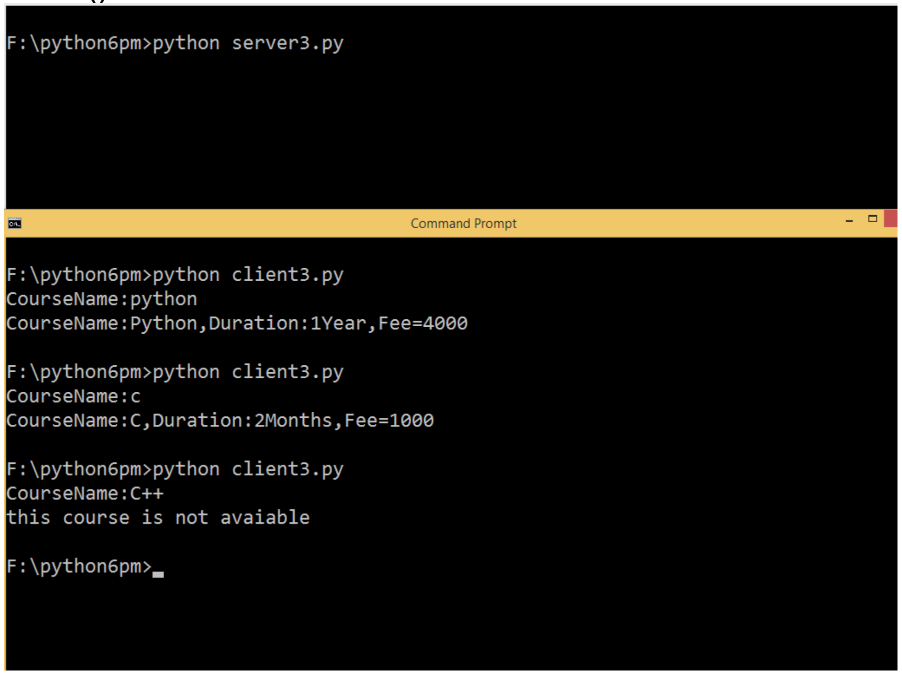
# server

```
import socket
def main():
    s=socket.socket()
    s.bind(("localhost",60))
    s.listen(10)
    while True:
        cs,add=s.accept()
        b=cs.recv(1024)
        course=b.decode()
        if course=="python":
            cs.send('CourseName:Python,Duration:1Year,Fee=4000'.encode())
        elif course=="java":
            cs.send('CourseName:Java,Duration:2Year,Fee=2000'.encode())
        elif course=="c":
            cs.send('CourseName:C,Duration:2Months,Fee=1000'.encode())
        else:
            cs.send("this course is not avaiable".encode())


main()
```

#client

```
import socket
def main():
    s=socket.socket()
    s.connect(("localhost",60))
    name=input("CourseName:")
    s.send(name.encode())
    b=s.recv(1024)
    d=b.decode()
    print(d)
main()
```



```
F:\python6pm>python server3.py
```

```
F:\python6pm>python client3.py
CourseName:python
CourseName:Python,Duration:1Year,Fee=4000

F:\python6pm>python client3.py
CourseName:c
CourseName:C,Duration:2Months,Fee=1000

F:\python6pm>python client3.py
CourseName:C++
this course is not avaiable

F:\python6pm>_
```

UDP Scokets

UDP stands for User Datagram Protocol, it is connectionless protocol. The data is transferred within network using UDP packet. This UDP packet consist of data and destination address. This UDP packet is send in network, It is response of network to send this UDP packet to destination. UDP implementation is not reliable.

At the time of creating socket, we need to define socket type. This type must be `SOCK_DGRAM`

socket.sendto(*bytes*, *address*)
Send data to the socket. The socket should not be connected to a remote socket, since the destination socket is specified by *address*
socket.recvfrom(*bufsize)*
Receive data from the socket. The return value is a pair (bytes,address) where *bytes* is a bytes object representing the data received and *address* is the address of the socket sending the data.

Example:
```
#UDP Receiver
import socket
def main():
    s=socket.socket(type=socket.SOCK_DGRAM)
    s.bind(('localhost',60))
    while True:
        b,add=s.recvfrom(1024)
        print(b.decode(),add)

main()
```

Example:
```
#UDP Sender
import socket
def main():
    s=socket.socket(type=socket.SOCK_DGRAM)
    s.bind(('localhost',80))
    s.sendto(b'Happy Birth Day Server',('localhost',60))
main()
```