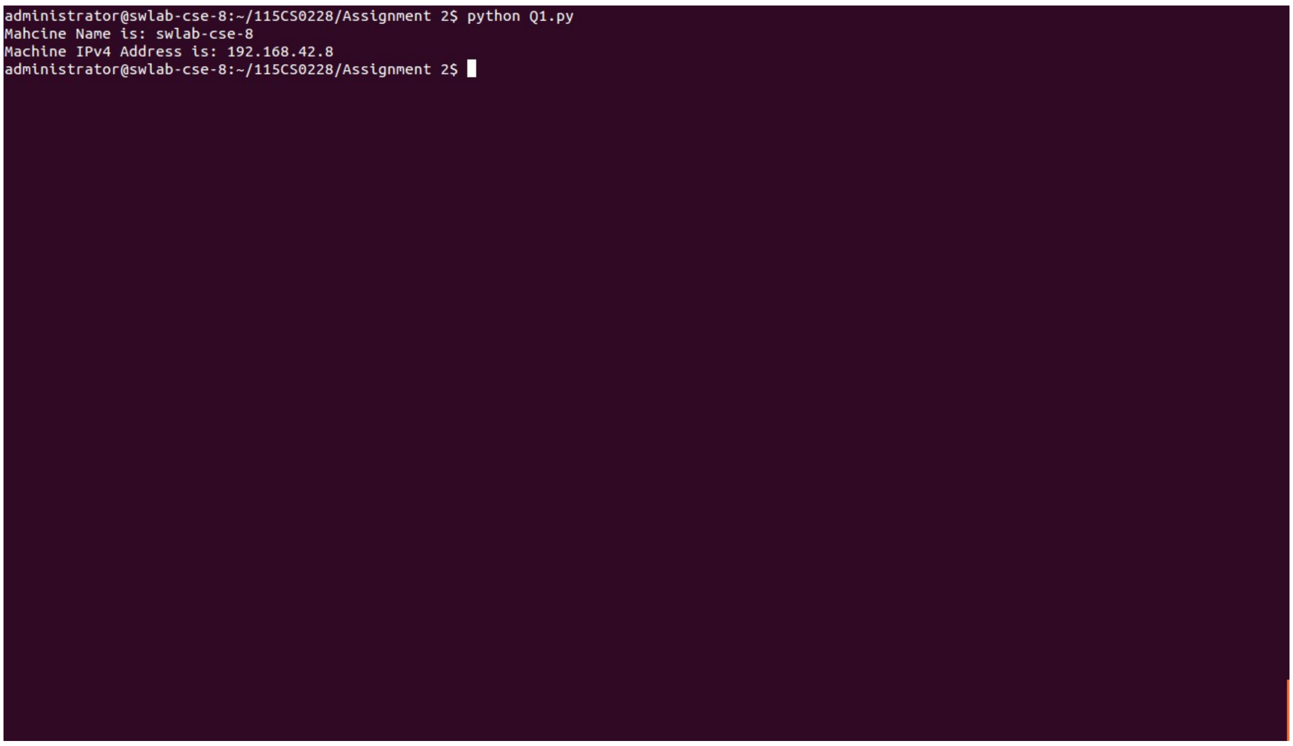
ASSIGNMENT 2

Q1. Printing your machine's name and IPv4 address?

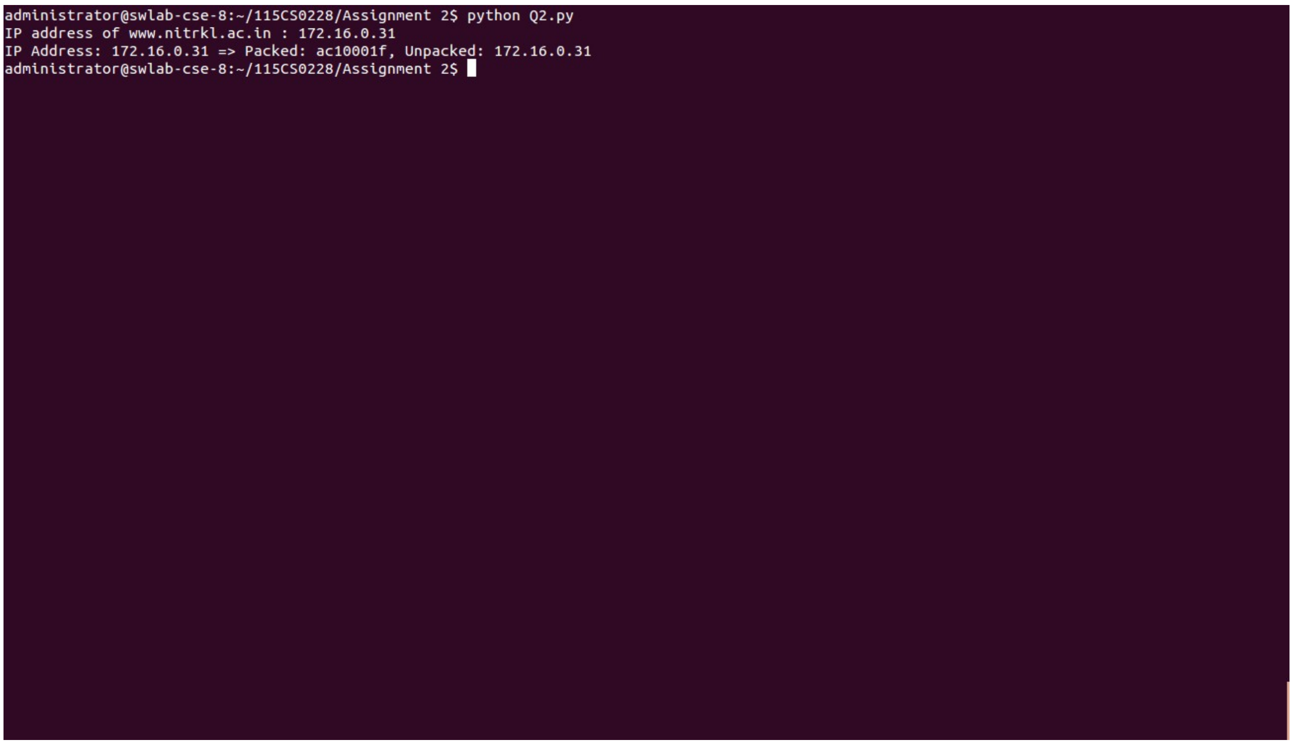
```
import socket
import fcntl
import struct

def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])

hostname = socket.gethostname()
print("Mahcine Name is: " + hostname)
print("Machine IPv4 Address is: " + get_ip_address('eth0'))
```



```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q1.py
Mahcine Name is: swlab-cse-8
Machine IPv4 Address is: 192.168.42.8
administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

Q2. Retrieve a remote machine's IP address and convert the IP address to different format?

```
import socket
from binascii import hexlify
def get_remote_machine_details():
        remote_host = "www.nitrkl.ac.in"
        ip_addr = socket.gethostbyname(remote_host)
        try:
                print "IP address of "+remote_host+" : "+ip_addr
                return ip_addr
        except socket.error, err_msg:
                print "%s: %s" %(remote_host, err_msg)

def convert_ip_addr(ip_addr):
        packed_ip_addr = socket.inet_aton(ip_addr)
        unpacked_ip_addr = socket.inet_ntoa(packed_ip_addr)
        print "IP Address: %s => Packed: %s, Unpacked: %s"\
        %(ip_addr, hexlify(packed_ip_addr), unpacked_ip_addr)

ip_addr = get_remote_machine_details()
convert_ip_addr(ip_addr)
```

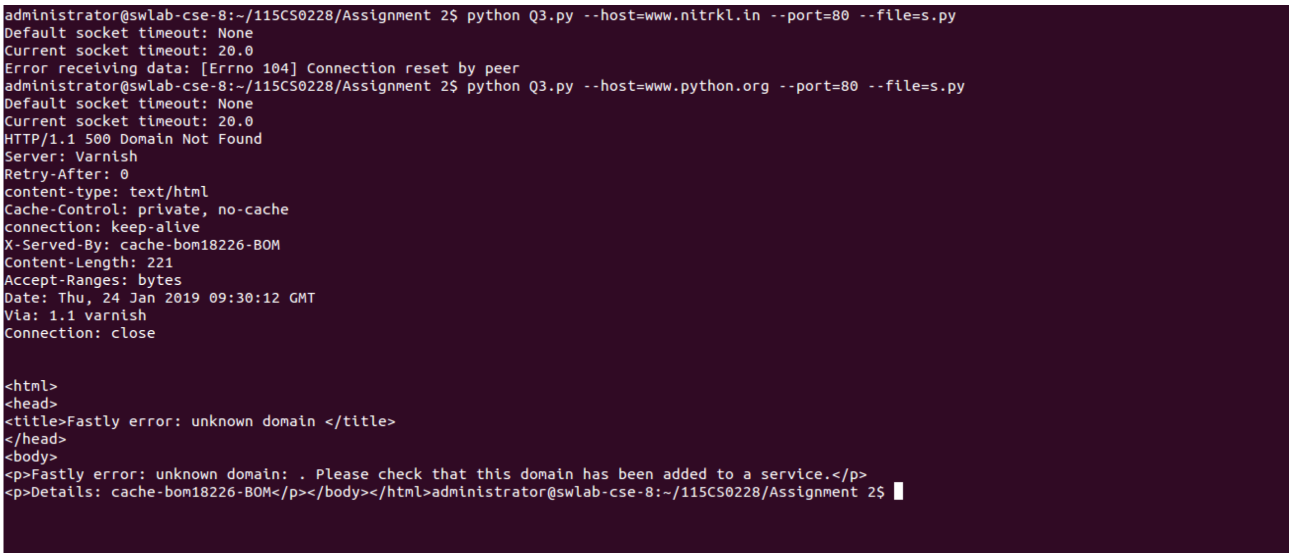


```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q2.py
IP address of www.nitrkl.ac.in : 172.16.0.31
IP Address: 172.16.0.31 => Packed: ac10001f, Unpacked: 172.16.0.31
administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

Q3. Setting and getting the default socket timeout, the program should include how to handle the socket error gracefully?

```
import sys
import socket
import argparse

def socket_timeout():
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print "Default socket timeout: %s" %s.gettimeout()
        s.settimeout(20)
        print "Current socket timeout: %s" %s.gettimeout()



def socket_error():
        # setup argument parsing
        parser = argparse.ArgumentParser(description='Socket Error Examples')
        parser.add_argument('--host', action="store", dest="host",required=False)
        parser.add_argument('--port', action="store", dest="port",type=int, required=False)
```

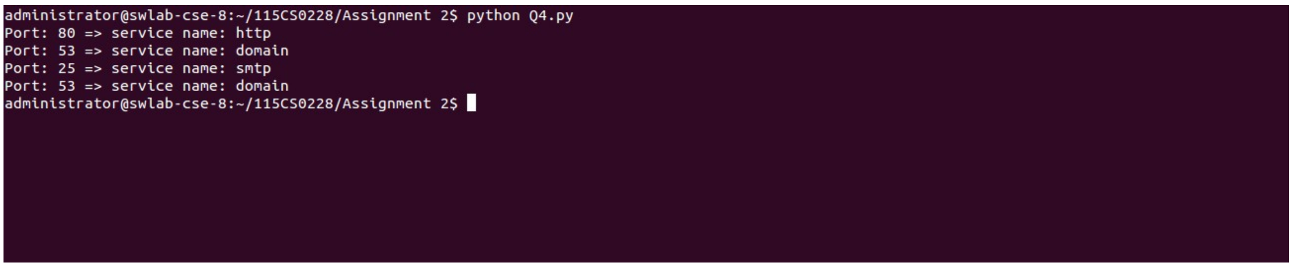
```
    parser.add_argument('--file', action="store", dest="file",required=False)
    given_args = parser.parse_args()
    host = given_args.host
    port = given_args.port
    filename = given_args.file
    # First try-except block -- create socket
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    except socket.error, e:
        print "Error creating socket: %s" % e
        sys.exit(1)
    # Second try-except block -- connect to given host/port
    try:
        s.connect((host, port))
    except socket.gaierror, e:
        print "Address-related error connecting to server: %s" % e
        sys.exit(1)
    except socket.error, e:
        print "Connection error: %s" % e
        sys.exit(1)
    # Third try-except block -- sending data
    try:
        s.sendall("GET %s HTTP/1.0\r\n\r\n" % filename)
    except socket.error, e:
        print "Error sending data: %s" % e
        sys.exit(1)
    while 1:
        # Fourth try-except block -- waiting to receive data from remote host
        try:
            buf = s.recv(2048)
        except socket.error, e:
            print "Error receiving data: %s" % e
            sys.exit(1)
        if not len(buf):
            break
        # write the received data
        sys.stdout.write(buf)

socket_timeout()

socket_error()
```



```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q3.py --host=www.nitrkl.in --port=80 --file=s.py
Default socket timeout: None
Current socket timeout: 20.0
Error receiving data: [Errno 104] Connection reset by peer
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q3.py --host=www.python.org --port=80 --file=s.py
Default socket timeout: None
Current socket timeout: 20.0
HTTP/1.1 500 Domain Not Found
Server: Varnish
Retry-After: 0
content-type: text/html
Cache-Control: private, no-cache
connection: keep-alive
X-Served-By: cache-bom18226-BOM
Content-Length: 221
Accept-Ranges: bytes
Date: Thu, 24 Jan 2019 09:30:12 GMT
Via: 1.1 varnish
Connection: close


<html>
<head>
<title>Fastly error: unknown domain </title>
</head>
<body>
<p>Fastly error: unknown domain: . Please check that this domain has been added to a service.</p>
<p>Details: cache-bom18226-BOM</p></body></html>administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

Q4. Finding the service name, given the port and protocol of the remote host (server)?

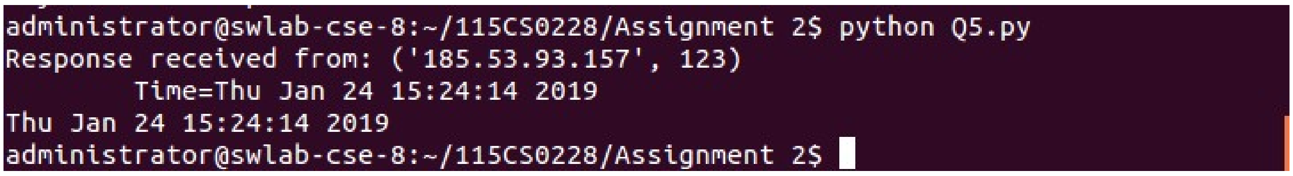
```
import socket
def get_service_name():
        protocolname = 'tcp'
        for port in [80, 25]:
                print "Port: %s => service name: %s" %(port, socket.getservbyport(port,
protocolname))
                print "Port: %s => service name: %s" %(53, socket.getservbyport(53, 'udp'))

get_service_name()
```



```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q4.py
Port: 80 => service name: http
Port: 53 => service name: domain
Port: 25 => service name: smtp
Port: 53 => service name: domain
administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

Q5. Printing the current time from the internet time server with the help of NTP? Also write an SNTP client that prints the current time from the internet time server received with the SNTP protocol?

```
import socket
import struct
import sys
import time
import ntplib
from time import ctime



def get_time():
        c = ntplib.NTPClient()
        response = c.request('pool.ntp.org')
        print(ctime(response.tx_time))


def sntp_client():
        NTP_SERVER = "0.uk.pool.ntp.org"
        TIME1970 = 2208988800L
        client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        data = '\x1b' + 47 * '\0'
        client.sendto(data, (NTP_SERVER, 123))
        data, address = client.recvfrom( 1024 )
        if data:
                print 'Response received from:', address
        t = struct.unpack( '!12I', data )[10]
        t -= TIME1970
        print '\tTime=%s' % time.ctime(t)

sntp_client()
get_time()
```

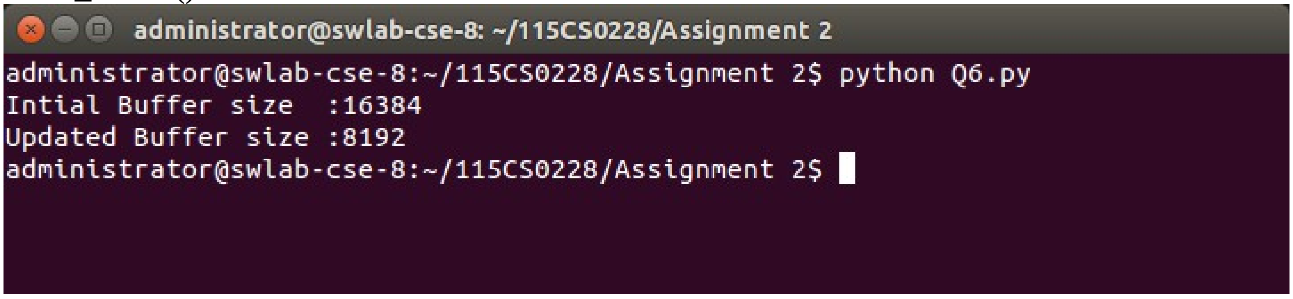


```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q5.py
Response received from: ('185.53.93.157', 123)
        Time=Thu Jan 24 15:24:14 2019
Thu Jan 24 15:24:14 2019
administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

Q6. Modifying sockets send/receive buffer size and changing the socket to blocking/non-blocking mode?

```
import socket

SEND_BUF_SIZE = 4096
RECV_BUF_SIZE = 4096

def update_buffer():
        sck = socket.socket(socket.AF_INET, socket.SOCK_STREAM )
        # Get the size of the socket's send buffer
        bufsize = sck.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
        print "Intial Buffer size  :%d" %bufsize
        sck.setsockopt(socket.SOL_TCP, socket.TCP_NODELAY, 1)
        sck.setsockopt(socket.SOL_SOCKET,socket.SO_SNDBUF,SEND_BUF_SIZE)
        sck.setsockopt(socket.SOL_SOCKET,socket.SO_RCVBUF,RECV_BUF_SIZE)
        bufsize = sck.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
        print "Updated Buffer size :%d" %bufsize

def block_modes():
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.setblocking(1)
        s.settimeout(0.5)
        s.bind(("127.0.0.1", 0))
        socket_address = s.getsockname()
        print "Trivial Server launched on socket: %s" %str(socket_address)
        #while(1):
```

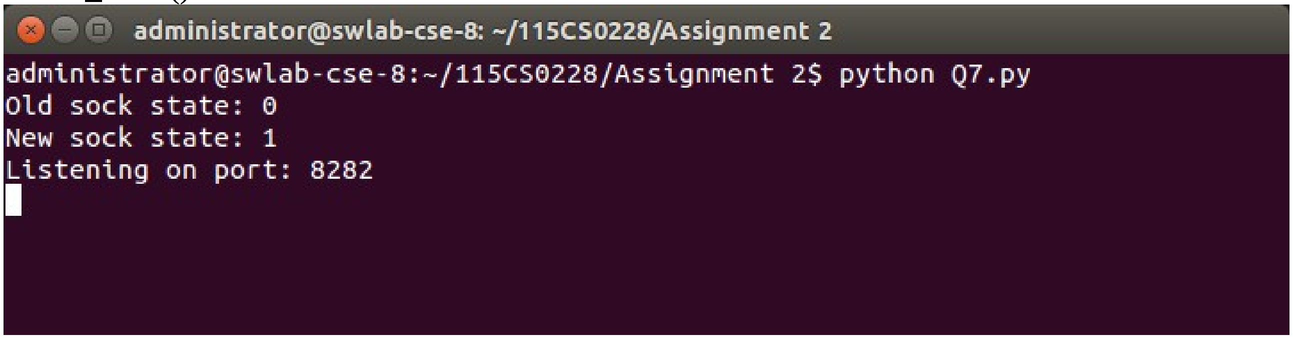
```
        #s.listen(1)


update_buffer()
block_modes()
```



```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q6.py
Intial Buffer size  :16384
Updated Buffer size :8192
administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

Q7. Write a program that demonstrates the reuse socket addresses?

```
import socket
import sys
def socket_reuse():
    sock = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
    # Get the old state of the SO_REUSEADDR option
    old_state = sock.getsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR)
    print "Old sock state: %s" %old_state
    # Enable the SO_REUSEADDR option
    sock.setsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR, 1 )
    new_state = sock.getsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR )
    print "New sock state: %s" %new_state
    local_port = 8282
    srv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    srv.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    srv.bind( ('', local_port) )
    srv.listen(1)
    print ("Listening on port: %s " %local_port)
    while True:
        try:
            connection, addr = srv.accept()
            print 'Connected by %s:%s' % (addr[0], addr[1])
        except KeyboardInterrupt:
            break
        except socket.error, msg:
            print '%s' % (msg,)

socket_reuse()
```



```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q7.py
Old sock state: 0
New sock state: 1
Listening on port: 8282
```

Q8. Write a simple TCP echo client/server application with the help of TCP socket object. The server wait for the client to be connected and send some data to the server. When the data is received, the server echoes the data to the client.

SERVER.PY

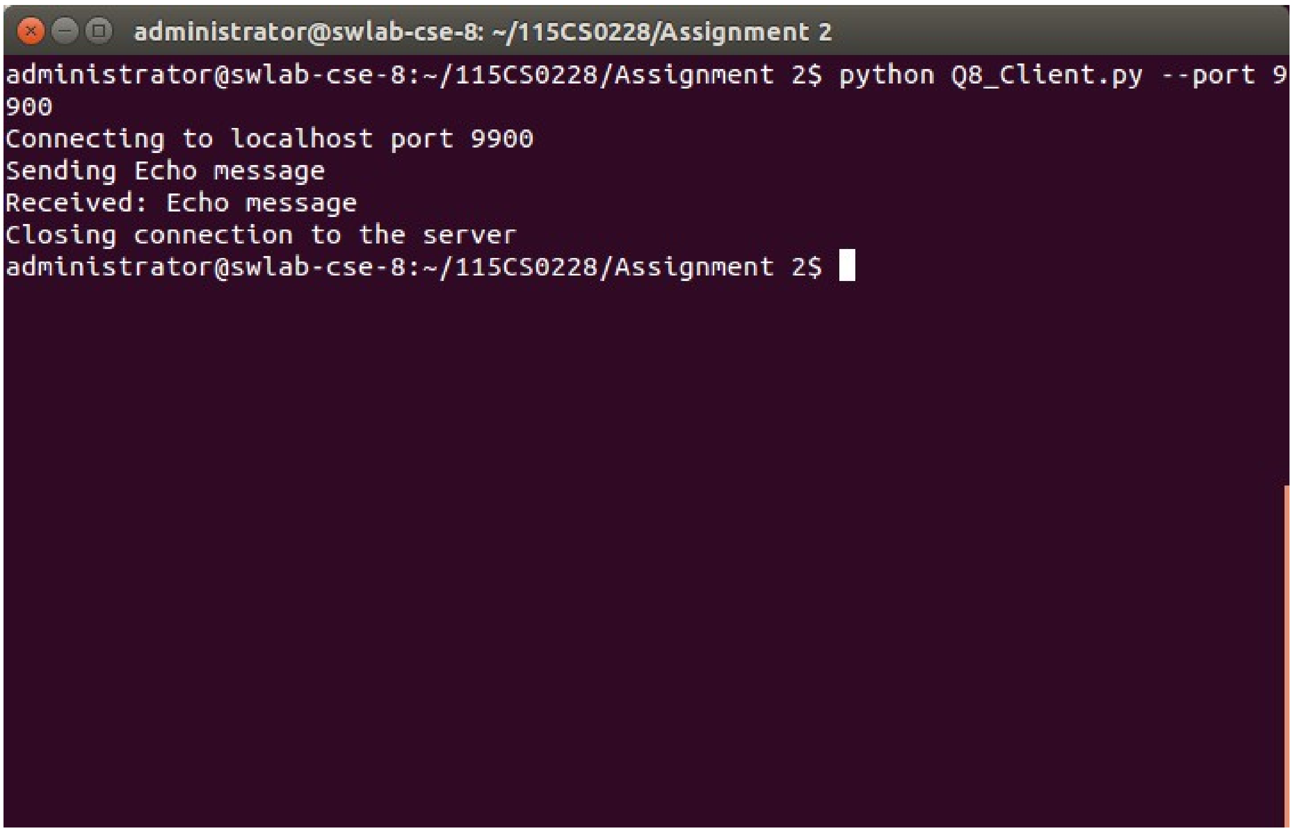
```
import socket
import sys
import argparse
host = 'localhost'
data_payload = 2048
backlog = 5

def echo_server(port):
	""" A simple echo server """
	# Create a TCP socket
	sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
	# Enable reuse address/port
	sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
	# Bind the socket to the port
	server_address = (host, port)
	print "Starting up echo server on %s port %s" % server_address
	sock.bind(server_address)
	# Listen to clients, backlog argument specifies the max no. of 	queued connections
	sock.listen(backlog)
	while True:
		print "Waiting to receive message from client"
		client, address = sock.accept()
		data = client.recv(data_payload)
		if data:
			print "Data: %s" %data
			client.send(data)
			print "sent %s bytes back to %s" % (data, address)
		# end connection
		client.close()

if __name__ == '__main__':
	parser = argparse.ArgumentParser(description='Socket Server Example')
	parser.add_argument('--port', action="store", dest="port",type=int, required=True)
	given_args = parser.parse_args()
	port = given_args.port
	echo_server(port)
```

CLIENT.PY

```
import socket
import sys
import argparse
host = 'localhost'
def echo_client(port):
	# Create a TCP/IP socket
	sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
	# Connect the socket to the server
	server_address = (host, port)
```

```
        print "Connecting to %s port %s" % server_address
        sock.connect(server_address)
        # Send data
        try:
                # Send data
                message = "Echo message"
                print "Sending %s" % message
                sock.sendall(message)
                # Look for the response
                amount_received = 0
                amount_expected = len(message)
                while amount_received < amount_expected:
                        data = sock.recv(16)
                        amount_received += len(data)
                        print "Received: %s" % data
        except socket.errno, e:
                print "Socket error: %s" %str(e)
        except Exception, e:
                print "Other exception: %s" %str(e)
        finally:
                print "Closing connection to the server"
                sock.close()

if __name__ == '__main__':
        parser = argparse.ArgumentParser(description='Socket Server Example')
        parser.add_argument('--port', action="store", dest="port",type=int, required=True)
        given_args = parser.parse_args()
        port = given_args.port
        echo_client(port)
```

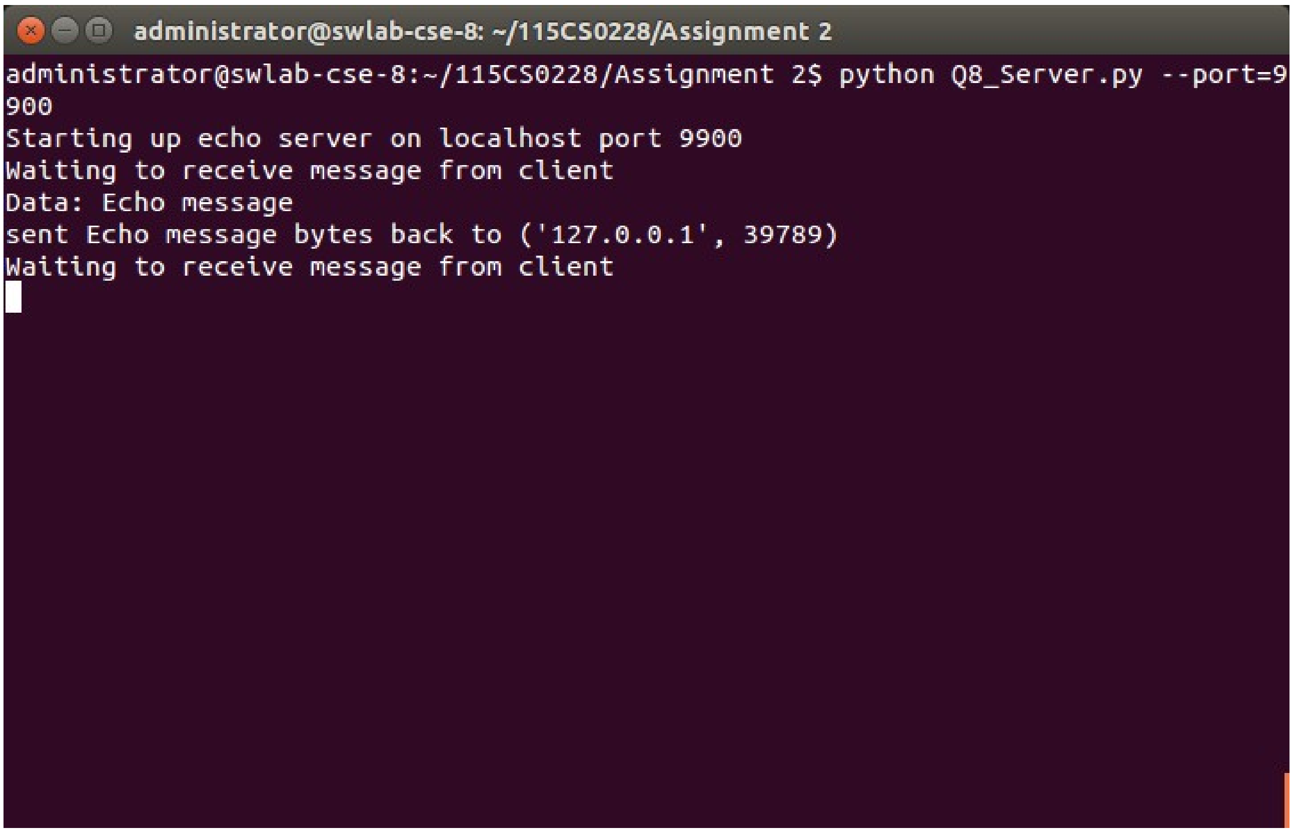
CLIENT SIDE



```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q8_Client.py --port 9
900
Connecting to localhost port 9900
Sending Echo message
Received: Echo message
Closing connection to the server
administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

`

SERVER SIDE



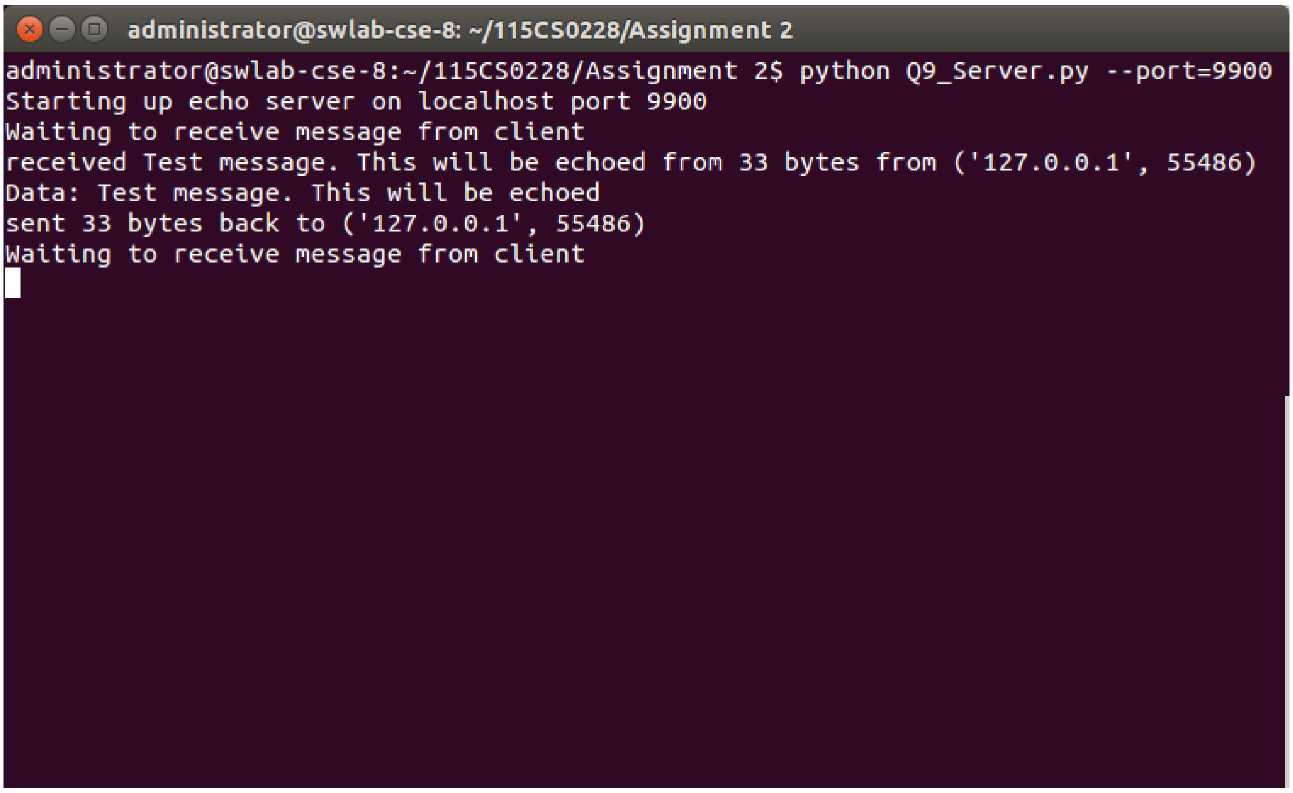
administrator@swlab-cse-8: ~/115CS0228/Assignment 2

```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q8_Server.py --port=9
900
Starting up echo server on localhost port 9900
Waiting to receive message from client
Data: Echo message
sent Echo message bytes back to ('127.0.0.1', 39789)
Waiting to receive message from client
```

Q9. Write a simple UDP echo client/server application with the help of TCP socket object. The server wait for the client to be connected and send some data to the server. When the data is received, the server echoes the data to the client.

SERVER.PY

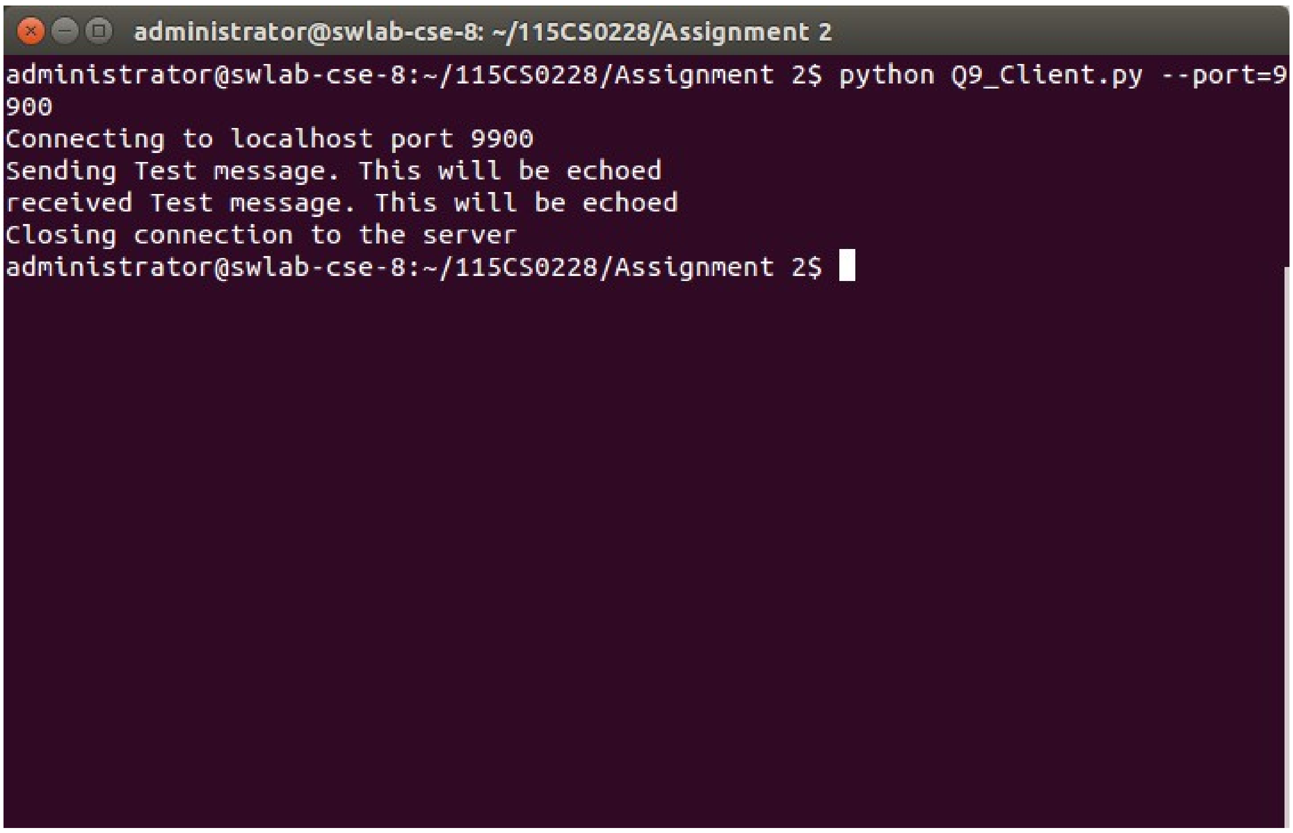
```
import socket
import sys
import argparse

host = 'localhost'
data_payload = 2048

def echo_server(port):
    """ A simple echo server """
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    # Bind the socket to the port
    server_address = (host, port)

    print ("Starting up echo server on %s port %s" % server_address)
    sock.bind(server_address)
    while True:
        print ("Waiting to receive message from client")
        data, address = sock.recvfrom(data_payload)
        print ("received %s from %s bytes from %s" % (data, len(data), address))
        print ("Data: %s" %data)
        if data:
            sent = sock.sendto(data, address)
            print ("sent %s bytes back to %s" % (sent, address))

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_server(port)
```



```
administrator@swlab-cse-8: ~/115CS0228/Assignment 2
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q9_Server.py --port=9900
Starting up echo server on localhost port 9900
Waiting to receive message from client
received Test message. This will be echoed from 33 bytes from ('127.0.0.1', 55486)
Data: Test message. This will be echoed
sent 33 bytes back to ('127.0.0.1', 55486)
Waiting to receive message from client
```

CLIENT.PY

```
import socket
import sys
import argparse

host = 'localhost'
data_payload = 2048

def echo_client(port):
    """ A simple echo client """
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_address = (host, port)
    print ("Connecting to %s port %s" % server_address)
    message = "This is a message"
    try:
        # Send data
        message = "Test message. This will be echoed"
        print ("Sending %s" % message)
        sent = sock.sendto(message.encode('utf-8'), server_address)
        # Receive response
        data, server = sock.recvfrom(data_payload)
        print ("received %s" % data)
    finally:
        print ("Closing connection to the server")
        sock.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_client(port)
```



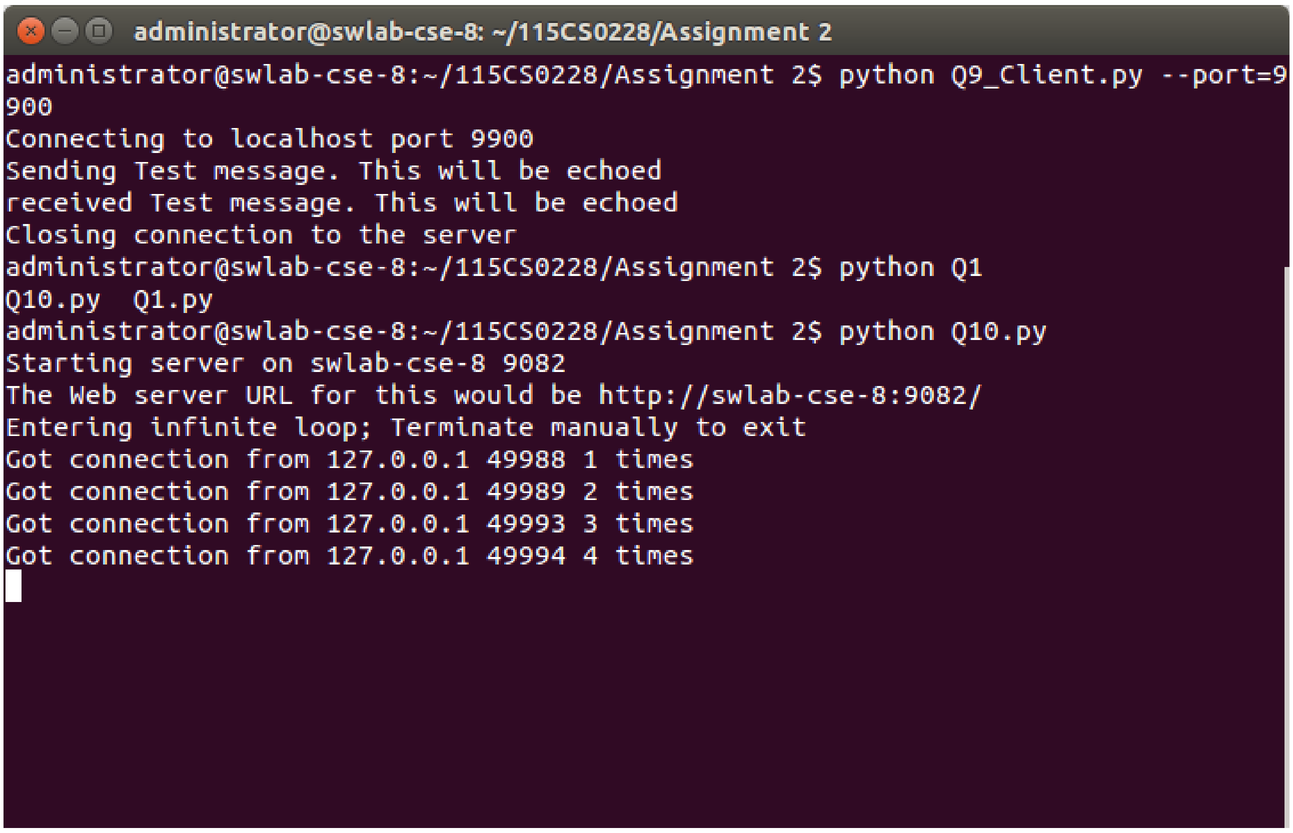
```
administrator@swlab-cse-8: ~/115CS0228/Assignment 2
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q9_Client.py --port=9
900
Connecting to localhost port 9900
Sending Test message. This will be echoed
received Test message. This will be echoed
Closing connection to the server
administrator@swlab-cse-8:~/115CS0228/Assignment 2$
```

Q10. Write a program that is a TCP server that returns a HTTP response to a browser that displays
the client's IP address and the number of times it has connected to the server. Test your program
with a standard Web browser like the Internet Explorer

CODE :

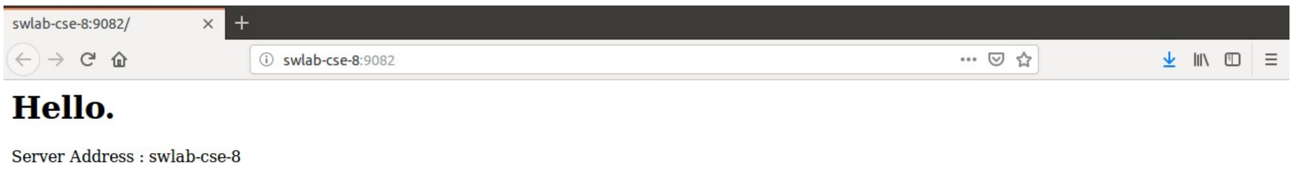
```
import socket

s = socket.socket()
host = socket.getfqdn()
port = 9082
s.bind((host, port))
```

```
print 'Starting server on', host, port
print 'The Web server URL for this would be http://%s:%d/' % (host, port)

s.listen(5)

print 'Entering infinite loop; Terminate manually to exit'

track = dict()

while True:
        c, (client_host, client_port) = s.accept()
        track[client_host] = track.get(client_host, 0) + 1
        c.recv(1000)
        c.send('HTTP/1.0 200 OK\n')
        c.send('Content-Type: text/html\n')
        c.send('\n')
        c.send("""
        <html>
        <body>
        <h1>Hello.</h1> Server Address : """+host+"""
        </body>
        </html>
        """)
        c.close()
        print 'Got connection from', client_host, client_port, track[client_host], 'times'
```



administrator@swlab-cse-8: ~/115CS0228/Assignment 2

```
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q9_Client.py --port=9
900
Connecting to localhost port 9900
Sending Test message. This will be echoed
received Test message. This will be echoed
Closing connection to the server
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q1
Q10.py  Q1.py
administrator@swlab-cse-8:~/115CS0228/Assignment 2$ python Q10.py
Starting server on swlab-cse-8 9082
The Web server URL for this would be http://swlab-cse-8:9082/
Entering infinite loop; Terminate manually to exit
Got connection from 127.0.0.1 49988 1 times
Got connection from 127.0.0.1 49989 2 times
Got connection from 127.0.0.1 49993 3 times
Got connection from 127.0.0.1 49994 4 times
```



Hello.

Server Address : swlab-cse-8