ASSIGNMENT - 3

Date: 30.01.2019 - 13.02.2019

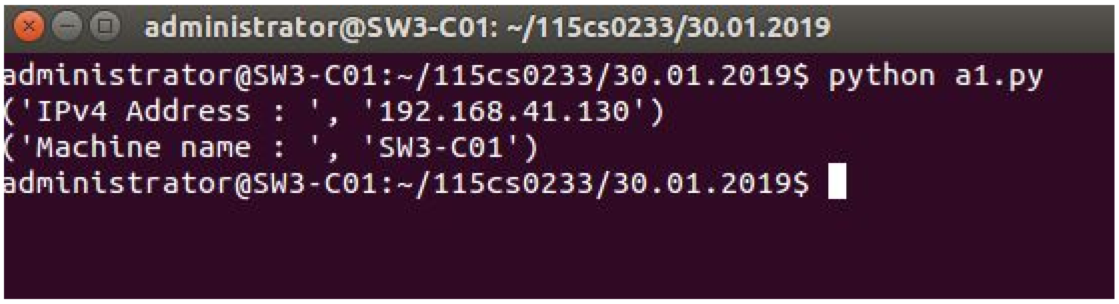
Topic : Programming with Python's socket modules.

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

Solution:

```
import socket

def get_Host_name_IP():
    try:
        host_name = socket.gethostname()
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(("8.8.8.8", 80))
        print("IPv4 Address : ",s.getsockname()[0])
        print("Machine name : ",host_name)
    except:
        print("Unable to get Hostname and IP")


get_Host_name_IP()
```

Output:



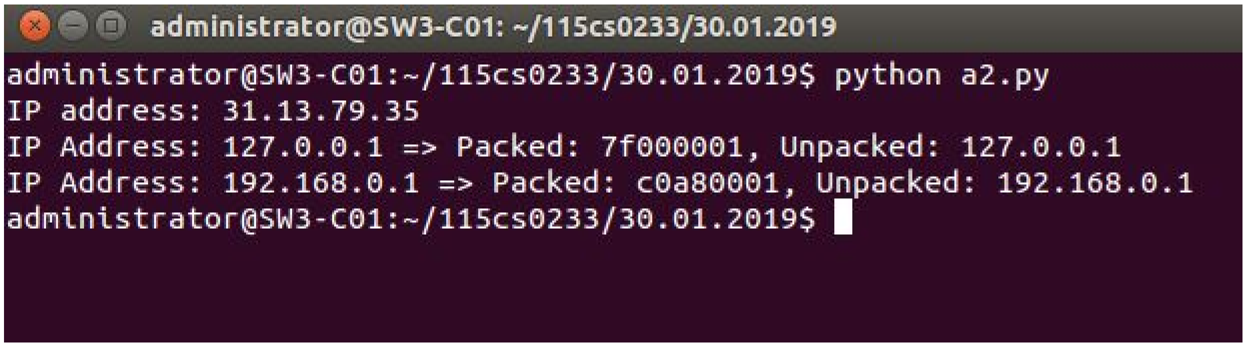
```
administrator@SW3-C01: ~/115cs0233/30.01.2019
administrator@SW3-C01:~/115cs0233/30.01.2019$ python a1.py
('IPv4 Address : ', '192.168.41.130')
('Machine name : ', 'SW3-C01')
administrator@SW3-C01:~/115cs0233/30.01.2019$
```

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

Solution:

```
import socket
from binascii import hexlify

def get_remote_machine_info():
   remote_host = 'www.facebook.com'
   try:
         print "IP address: %s" %socket.gethostbyname(remote_host)
   except socket.error, err_msg:
         print "%s: %s" %(remote_host, err_msg)

def convert_ip4_address():
   for ip_addr in ['127.0.0.1', '192.168.0.1']:
         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)

if __name__ == '__main__':
   get_remote_machine_info()
   convert_ip4_address()
```

Output:



```
administrator@SW3-C01: ~/115cs0233/30.01.2019
administrator@SW3-C01:~/115cs0233/30.01.2019$ python a2.py
IP address: 31.13.79.35
IP Address: 127.0.0.1 => Packed: 7f000001, Unpacked: 127.0.0.1
IP Address: 192.168.0.1 => Packed: c0a80001, Unpacked: 192.168.0.1
administrator@SW3-C01:~/115cs0233/30.01.2019$
```

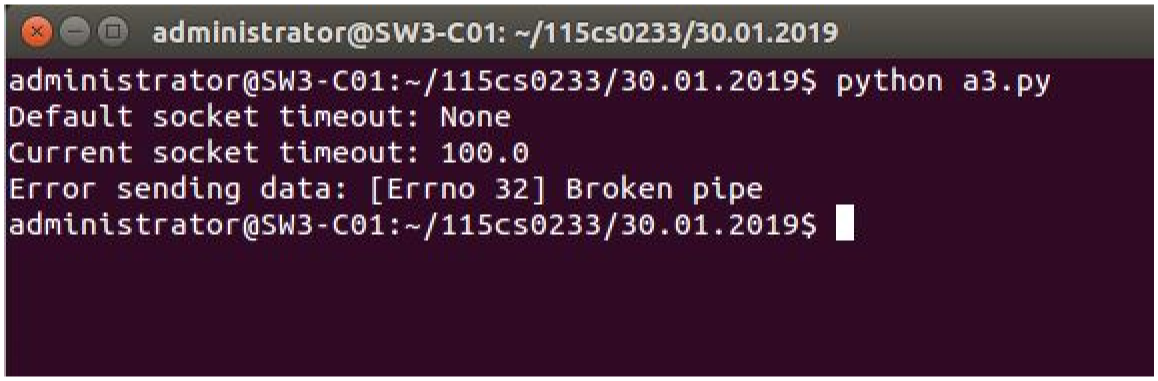
3. Setting and getting the default socket timeout, the program should include how to handle the socket error gracefully?

Solution:

```
import socket
import sys
import argparse

def test_socket_timeout():
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   print "Default socket timeout: %s" %s.gettimeout()
   s.settimeout(100)
   print "Current socket timeout: %s" %s.gettimeout()
def main():
        # 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 as e:
        print ("Error creating socket: %s" % e)
        sys.exit(1)


        # Third try-except block -- sending data
        try:
        msg = "GET %s HTTP/1.0\r\n\r\n" % filename
        s.sendall(msg.encode('utf-8'))
        except socket.error as e:
```

```
        print ("Error sending data: %s" % e)
        sys.exit(1)

        while 1:
        # Fourth tr-except block -- waiting to receive data from remote host
        try:
        buf = s.recv(2048)
        except socket.error as e:
        print ("Error receiving data: %s" % e)
        sys.exit(1)
        if not len(buf):
        break
        # write the received data
    sys.stdout.write(buf.decode('utf-8'))

if __name__ == '__main__':
    test_socket_timeout()
    main()
```

Output:



```
administrator@SW3-C01: ~/115cs0233/30.01.2019
administrator@SW3-C01:~/115cs0233/30.01.2019$ python a3.py
Default socket timeout: None
Current socket timeout: 100.0
Error sending data: [Errno 32] Broken pipe
administrator@SW3-C01:~/115cs0233/30.01.2019$
```

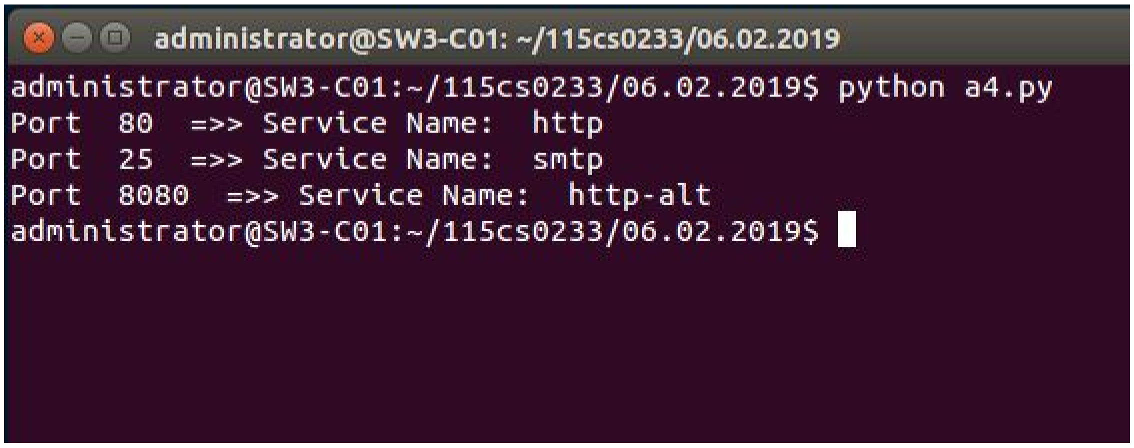
4. Finding the service name, given the port and protocol of the remote host (server)?

Solution:

```
import socket

protocol_name = 'tcp'
```

```
for port in [80,25,8080]:
    print "Port ",port," =>> Service Name: ",socket.getservbyport(port,protocol_name)
```

Output:



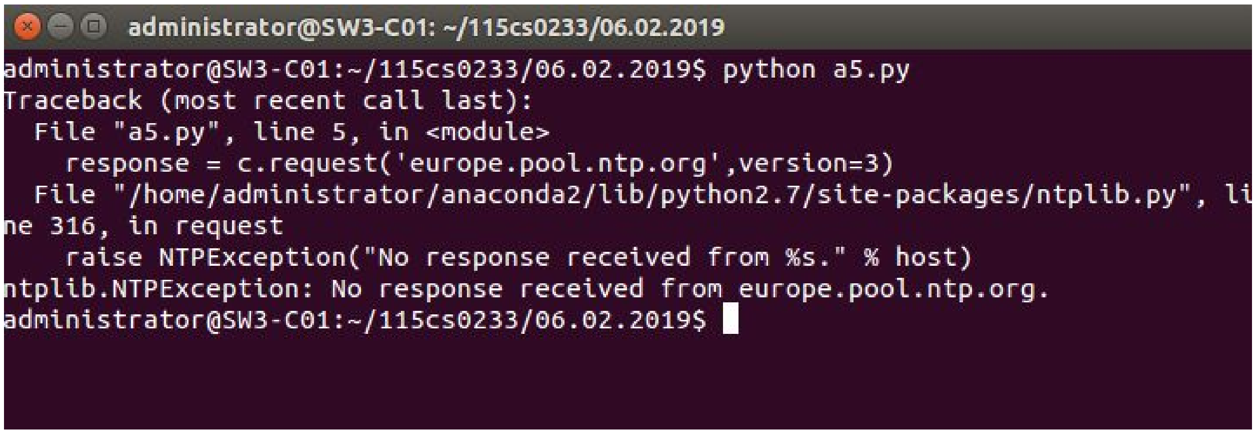
```
administrator@SW3-C01: ~/115cs0233/06.02.2019
administrator@SW3-C01:~/115cs0233/06.02.2019$ python a4.py
Port  80  =>> Service Name:  http
Port  25  =>> Service Name:  smtp
Port  8080  =>> Service Name:  http-alt
administrator@SW3-C01:~/115cs0233/06.02.2019$
```

5. 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?

Solution:

```
import ntplib
from time import ctime

c = ntplib.NTPClient()
response = c.request('europe.pool.ntp.org',version=3)
print(ctime(response.tx_time))
```

Output:


```
administrator@SW3-C01: ~/115cs0233/06.02.2019
administrator@SW3-C01:~/115cs0233/06.02.2019$ python a5.py
Traceback (most recent call last):
  File "a5.py", line 5, in <module>
    response = c.request('europe.pool.ntp.org',version=3)
  File "/home/administrator/anaconda2/lib/python2.7/site-packages/ntplib.py", li
ne 316, in request
    raise NTPException("No response received from %s." % host)
ntplib.NTPException: No response received from europe.pool.ntp.org.
administrator@SW3-C01:~/115cs0233/06.02.2019$
```

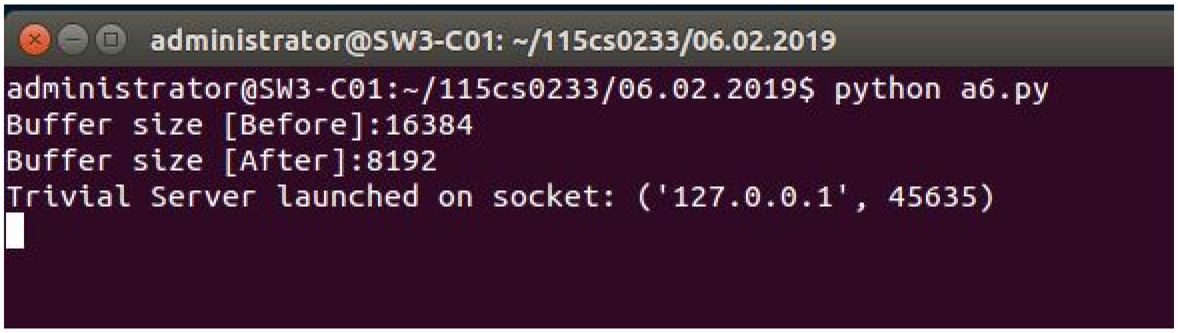
6. Modifying sockets send/receive buffer size and changing the socket to blocking/non-blocking mode?

Solution:

```
import socket

SEND_BUF_SIZE = 4096
RECV_BUF_SIZE = 4096

def modify_buff_size():
   sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM )
   # Get the size of the socket's send buffer
   bufsize = sock.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
   print "Buffer size [Before]:%d" %bufsize
   sock.setsockopt(socket.SOL_TCP, socket.TCP_NODELAY, 1)

   sock.setsockopt(
         socket.SOL_SOCKET,
         socket.SO_SNDBUF,
         SEND_BUF_SIZE)
   sock.setsockopt(
         socket.SOL_SOCKET,
         socket.SO_RCVBUF,
         RECV_BUF_SIZE)

   bufsize = sock.getsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF)
   print "Buffer size [After]:%d" %bufsize
```

```
def test_socket_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)
if __name__ == '__main__':
   modify_buff_size()
   test_socket_modes()
```

Output:



```
administrator@SW3-C01: ~/115cs0233/06.02.2019
administrator@SW3-C01:~/115cs0233/06.02.2019$ python a6.py
Buffer size [Before]:16384
Buffer size [After]:8192
Trivial Server launched on socket: ('127.0.0.1', 45635)
```

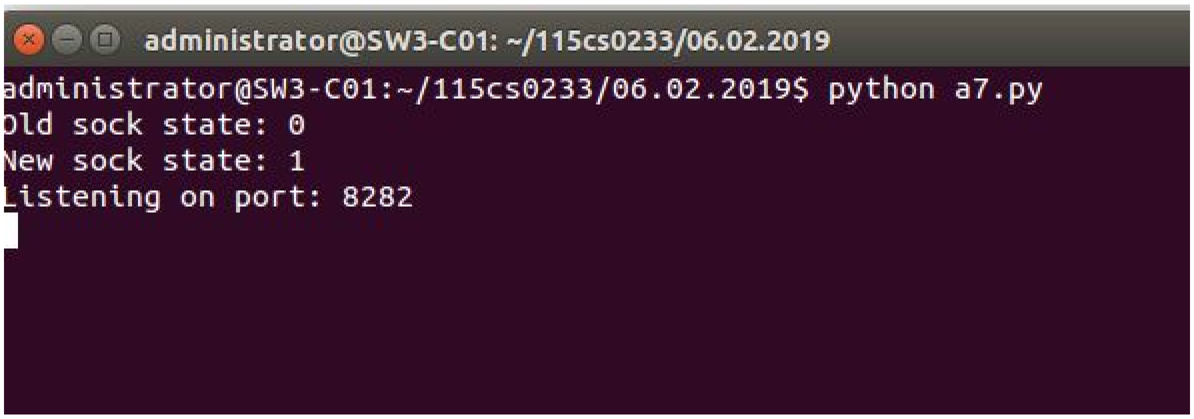
7. Write a program that demonstrates the reuse socket addresses?

Solution:

```
import socket
import sys
def reuse_socket_addr():
   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,)
if __name__ == '__main__':
    reuse_socket_addr()
```

Output:



```
administrator@SW3-C01: ~/115cs0233/06.02.2019
administrator@SW3-C01:~/115cs0233/06.02.2019$ python a7.py
Old sock state: 0
New sock state: 1
Listening on port: 8282
```

8. 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.

Solution:

Server:

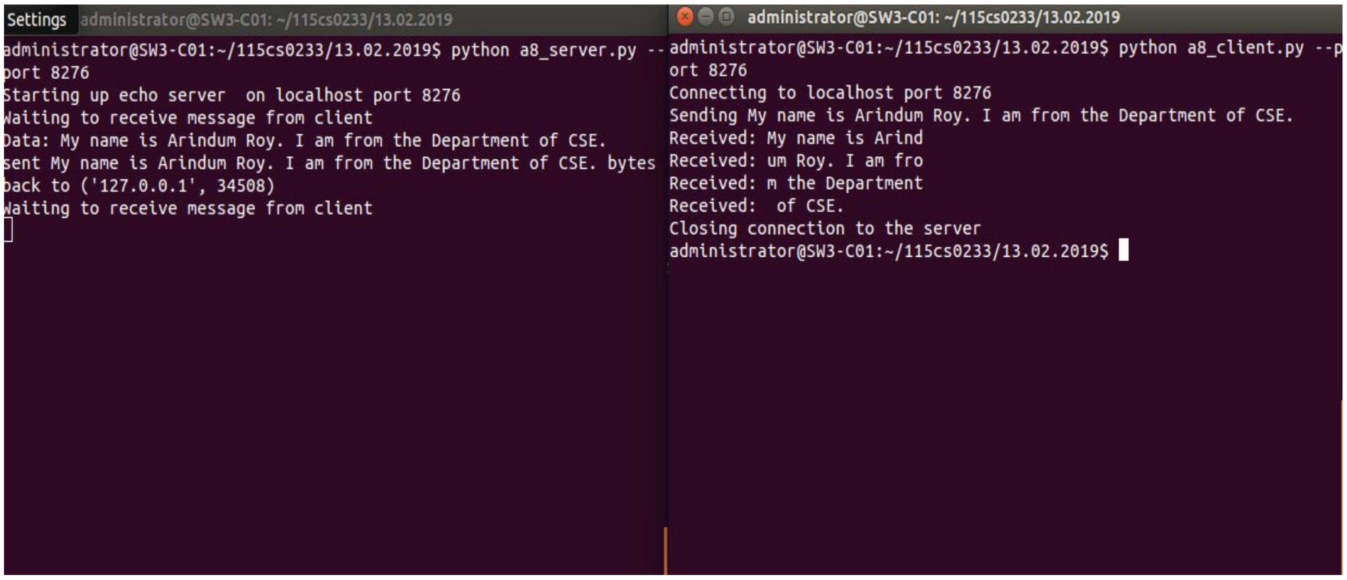
```
import socket
import sys
import argparse

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:

```
import socket
import sys

import argparse

host = 'localhost'

def echo_client(port):
        """ A simple echo client """
        # 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 = "My name is Arindum Roy. I am from the Department of CSE."
        print ("Sending %s" % message)
        sock.sendall(message.encode('utf-8'))
        # 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.error as e:
        print ("Socket error: %s" %str(e))
        except Exception as 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)
```

Output:



```
administrator@SW3-C01:~/115cs0233/13.02.2019$ python a8_server.py --
port 8276
Starting up echo server  on localhost port 8276
Waiting to receive message from client
Data: My name is Arindum Roy. I am from the Department of CSE.
sent My name is Arindum Roy. I am from the Department of CSE. bytes
back to ('127.0.0.1', 34508)
Waiting to receive message from client
```

```
administrator@SW3-C01:~/115cs0233/13.02.2019$ python a8_client.py --p
ort 8276
Connecting to localhost port 8276
Sending My name is Arindum Roy. I am from the Department of CSE.
Received: My name is Arind
Received: um Roy. I am fro
Received: m the Department
Received:  of CSE.
Closing connection to the server
administrator@SW3-C01:~/115cs0233/13.02.2019$
```

9. 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.

Solution:

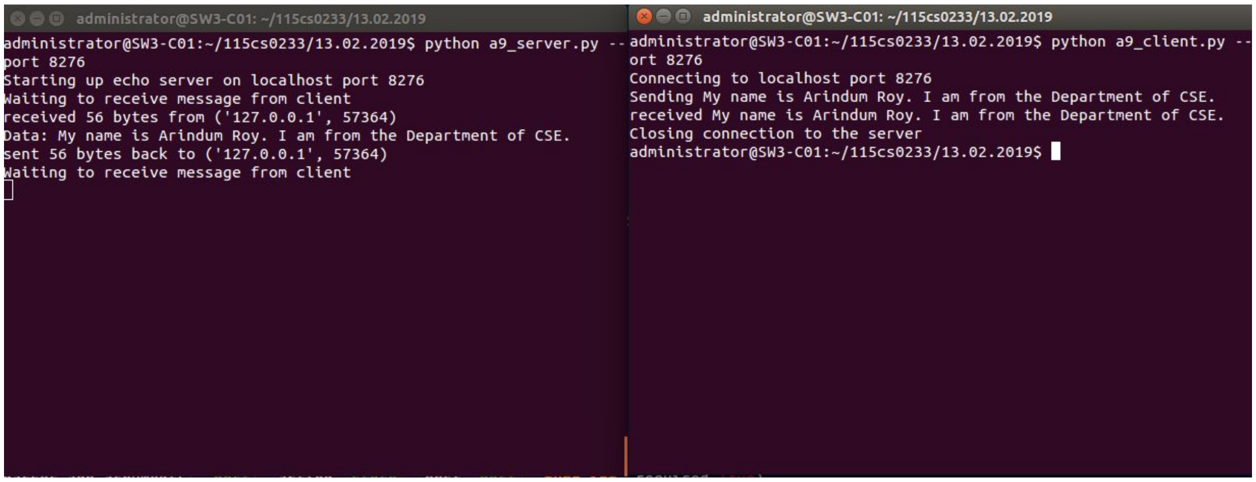
Server:

```
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 bytes from %s" % (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)
```

Client:

```
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 the message.  It will be repeated.'

        try:

        # Send data
        message = "My name is Arindum Roy. I am from the Department of CSE."
        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)
```

Output:



```
administrator@SW3-C01:~/115cs0233/13.02.2019$ python a9_server.py --port 8276
Starting up echo server on localhost port 8276
Waiting to receive message from client
received 56 bytes from ('127.0.0.1', 57364)
Data: My name is Arindum Roy. I am from the Department of CSE.
sent 56 bytes back to ('127.0.0.1', 57364)
Waiting to receive message from client
```

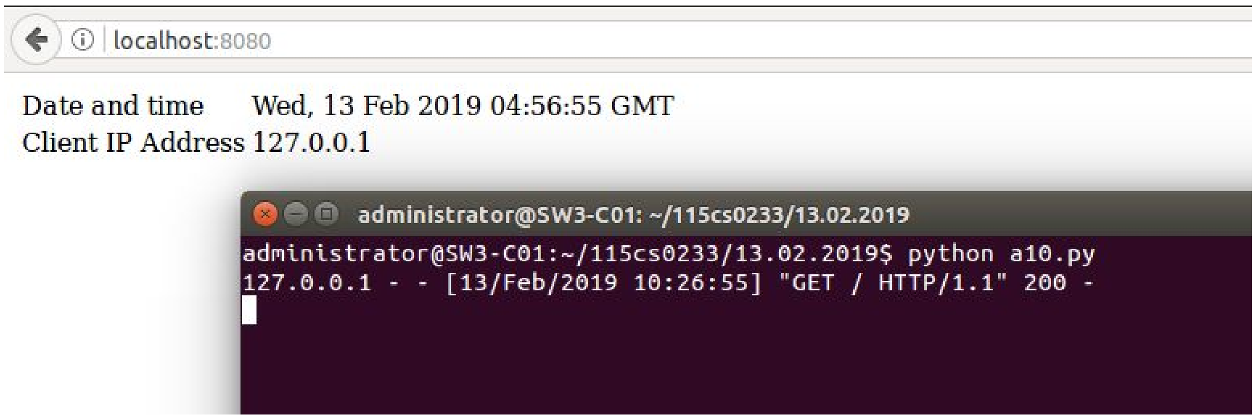
```
administrator@SW3-C01:~/115cs0233/13.02.2019$ python a9_client.py --port 8276
Connecting to localhost port 8276
Sending My name is Arindum Roy. I am from the Department of CSE.
received My name is Arindum Roy. I am from the Department of CSE.
Closing connection to the server
administrator@SW3-C01:~/115cs0233/13.02.2019$
```

10. 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.

Solution:

```
import BaseHTTPServer

class RequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):
   Page = '''\
<html>
<body>
<table>
<tr>  <td>Date and time</td>  <td>{date_time}</td>     </tr>
<tr>  <td>Client IP Address</td>   <td>{client_host}</td>  </tr>
</table>
</body>
</html>
'''
   def do_GET(self):
                  page = self.create_page()
                  self.send_page(page)
   def create_page(self):
                  values = {
                  'date_time'   : self.date_time_string(),
                   'client_host' : self.client_address[0],
                   'client_port' : self.client_address[1],
                  'command'   : self.command,
                  'path'            : self.path
                  }
                  page = self.Page.format(**values)
                  return page
   def send_page(self, page):
                  self.send_response(200)
                  self.send_header("Content-type", "text/html")
                  self.send_header("Content-Length", str(len(page)))
                  self.end_headers()
                  self.wfile.write(page)

if __name__ == '__main__':
          serverAddress = ('', 8080)
          server = BaseHTTPServer.HTTPServer(serverAddress, RequestHandler)
          server.serve_forever()
```

Output:



localhost:8080

```
Date and time      Wed, 13 Feb 2019 04:56:55 GMT
Client IP Address 127.0.0.1
```

administrator@SW3-C01: ~/115cs0233/13.02.2019

```
administrator@SW3-C01:~/115cs0233/13.02.2019$ python a10.py
127.0.0.1 - - [13/Feb/2019 10:26:55] "GET / HTTP/1.1" 200 -
```