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

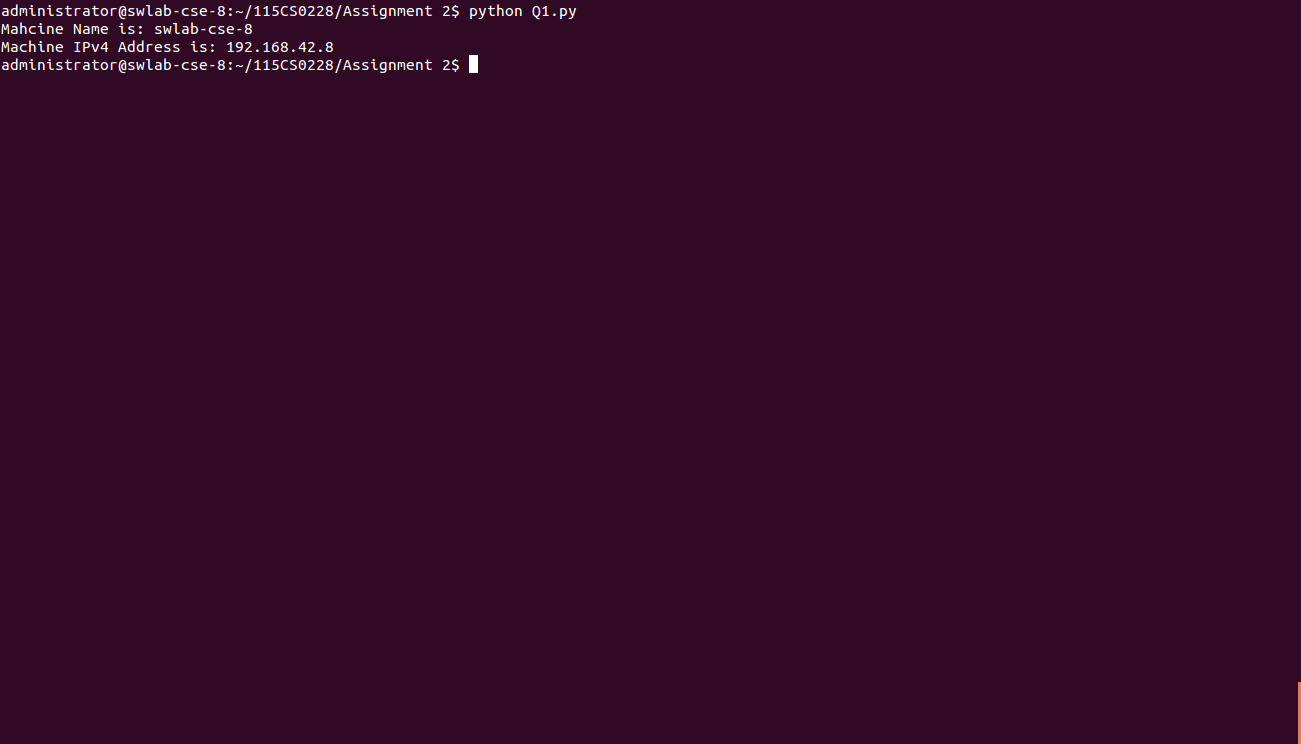
struct.pack('256s', ifname[:15])

)[20:24])

hostname = socket.gethostname()

print("Mahcine Name is: " + hostname)

print("Machine IPv4 Address is: " + get\_ip\_address('eth0'))



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)

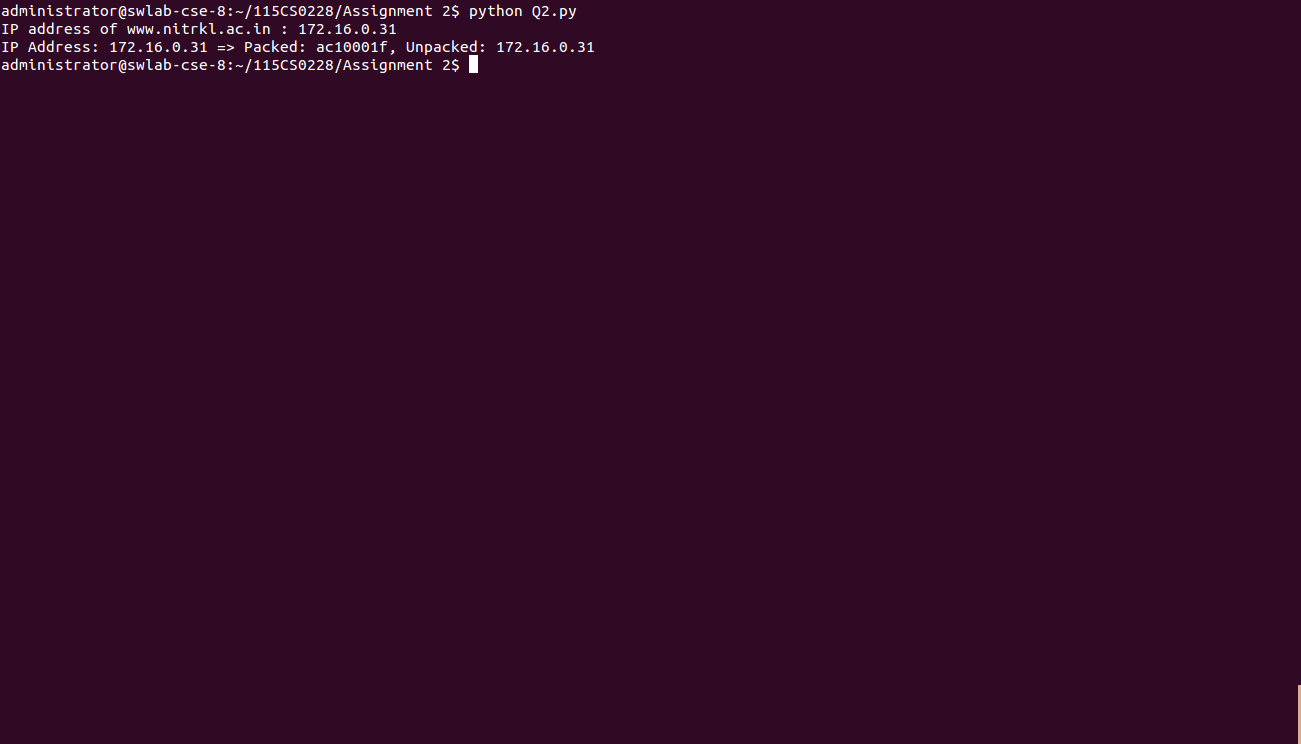
unpacked\_ip\_addr = socket.inet\_ntoa(packed\_ip\_addr)

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)



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):

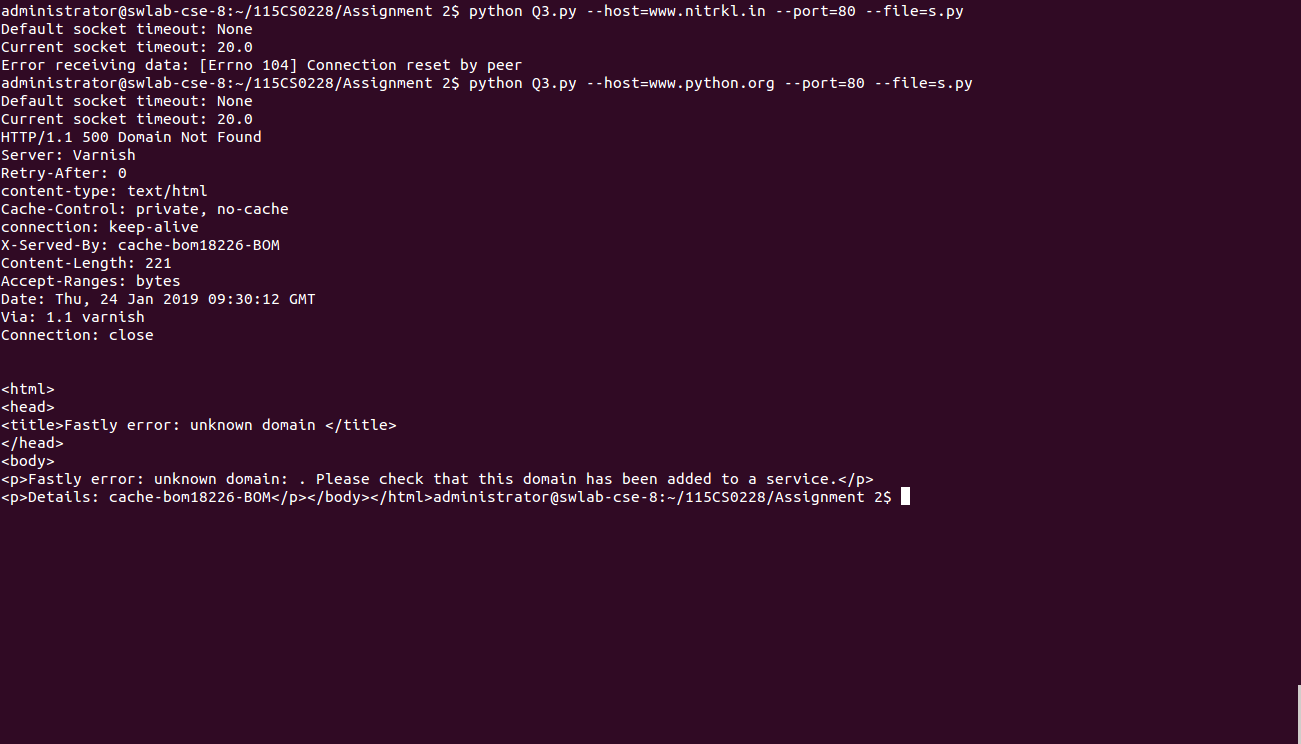
break

# write the received data

sys.stdout.write(buf)

socket\_timeout()

socket\_error()



Q4. Finding the service name, given the port and protocol of the remote host

(server)?

import socket

def get\_service\_name():

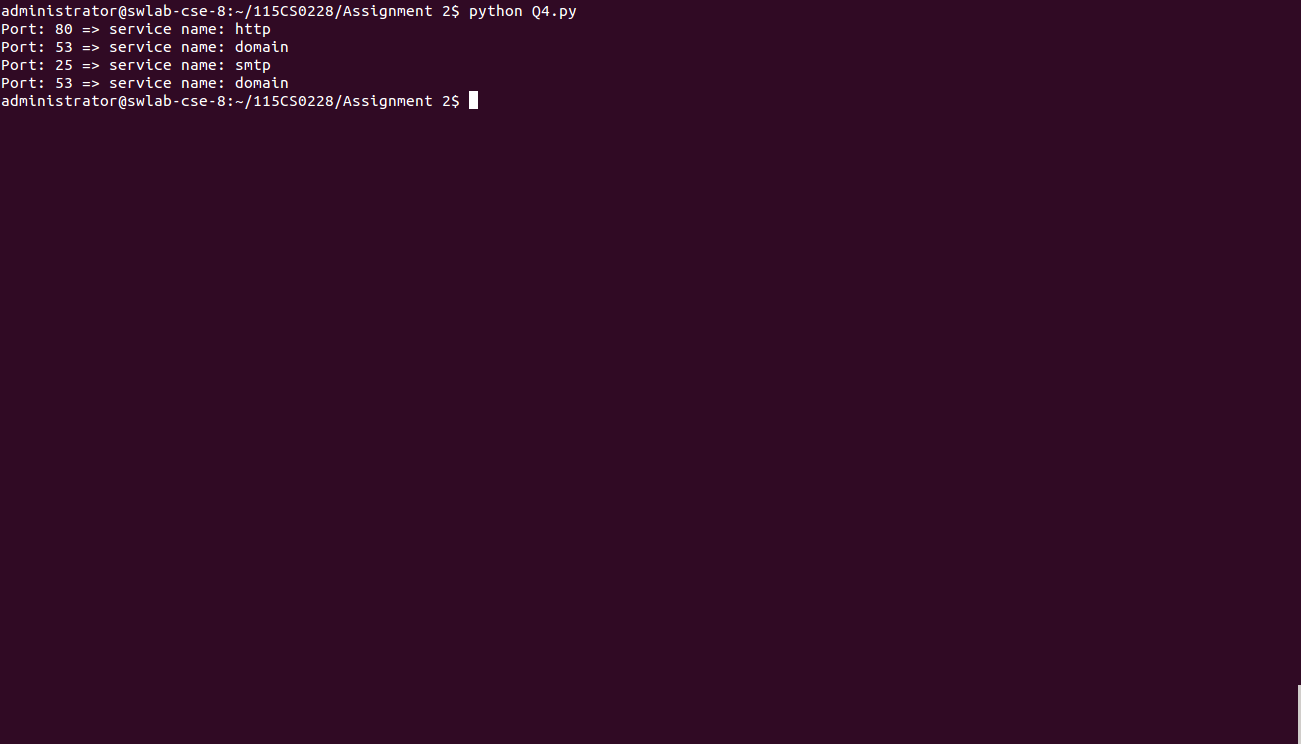
protocolname = 'tcp'

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()



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

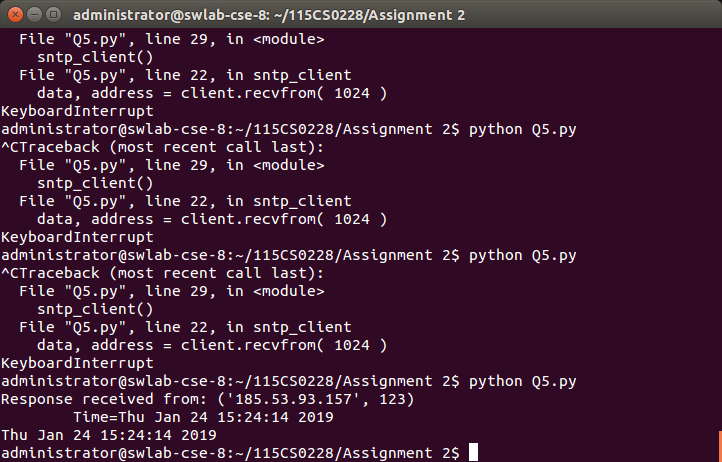
t = struct.unpack( '!12I', data )[10]

t -= TIME1970

print '\tTime=%s' % time.ctime(t)

sntp\_client()

get\_time()



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()

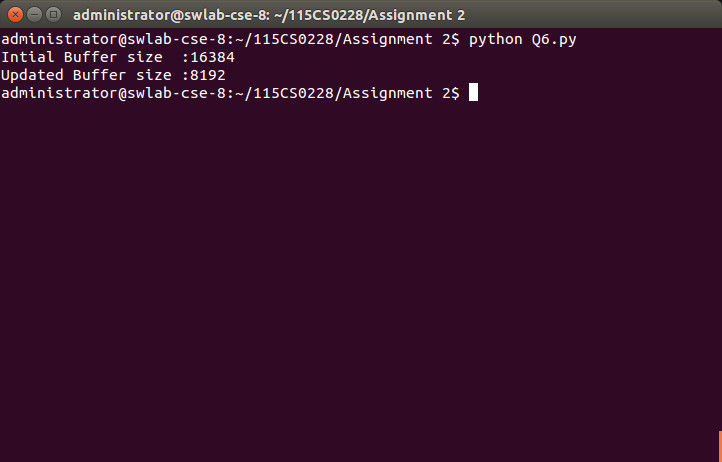
print "Trivial Server launched on socket: %s" %str(socket\_address)

#while(1):

#s.listen(1)

update\_buffer()

block\_modes()



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])

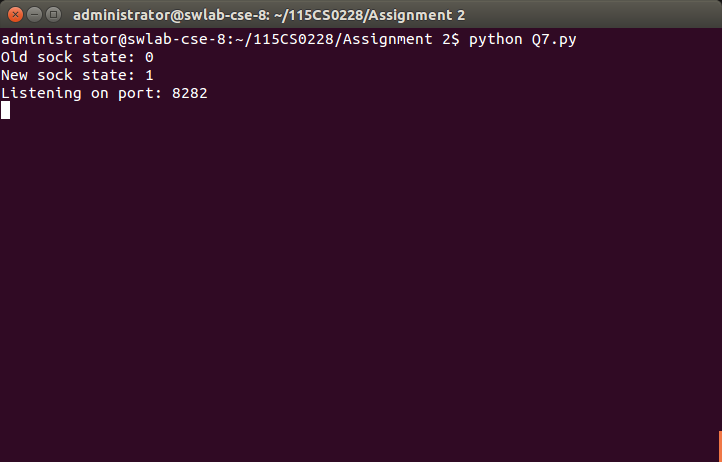
except KeyboardInterrupt:

break

except socket.error, msg:

print '%s' % (msg,)

socket\_reuse()



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