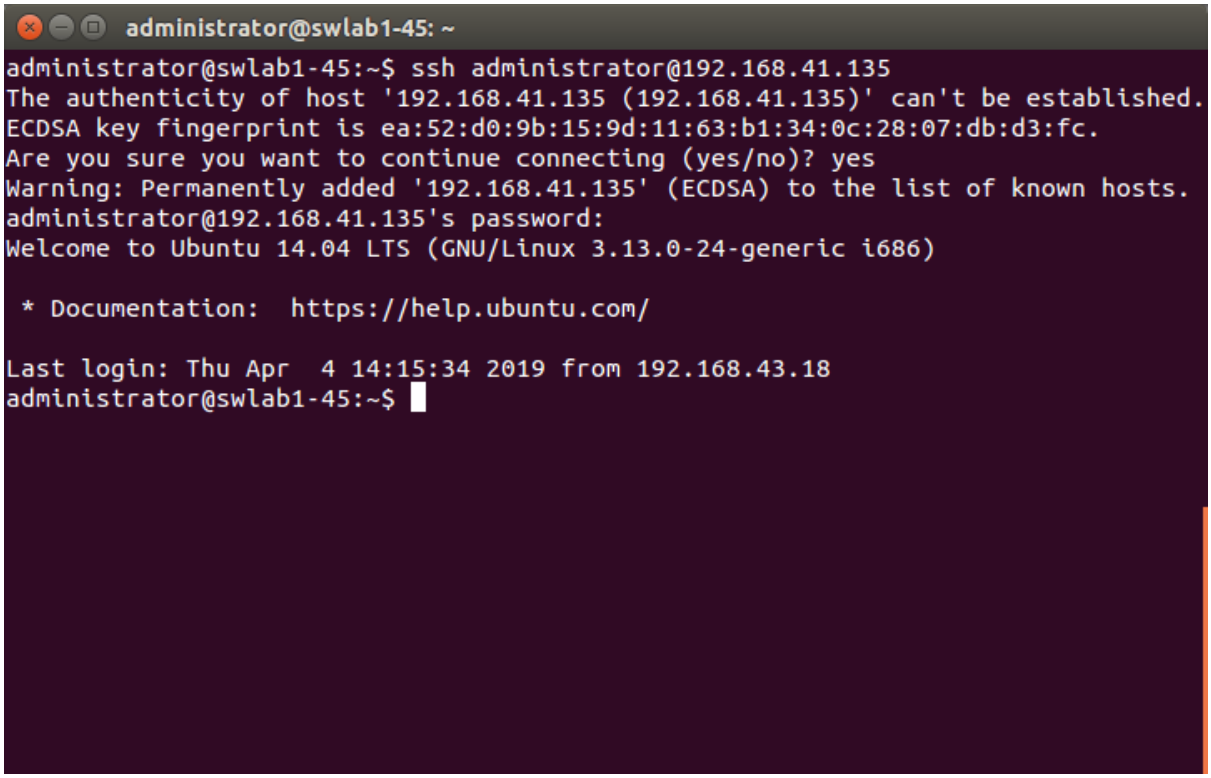
April 4, 2019

[27] Write a program that establish a connection with a remote machine over SSH. Then display the remote machine's CPU and Main memory information.

N.B. you need to install third party package ***Paramiko***




```
Last login: Thu Apr  4 14:15:34 2019 from 192.168.43.18
administrator@swlab1-45:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping        : 9
microcode       : 0x12
cpu MHz         : 1600.000
cache size      : 8192 KB
```

physical id : 0
siblings : 8
core id : 0
cpu cores : 4
apicid : 0
initial apicid : 0
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

processor : 1
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping : 9
microcode : 0x12
cpu MHz : 1600.000
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 1
cpu cores : 4
apicid : 2
initial apicid : 2
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2

ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

processor : 2
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping : 9
microcode : 0x12
cpu MHz : 1600.000
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 2
cpu cores : 4
apicid : 4
initial apicid : 4
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

processor : 3
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping : 9
microcode : 0x12
cpu MHz : 1600.000

cache size : 8192 KB
physical id : 0
siblings : 8
core id : 3
cpu cores : 4
apicid : 6
initial apicid : 6
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

processor : 4
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping : 9
microcode : 0x12
cpu MHz : 1600.000
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 0
cpu cores : 4
apicid : 1
initial apicid : 1
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc

aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

processor : 5
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping : 9
microcode : 0x12
cpu MHz : 1600.000
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 1
cpu cores : 4
apicid : 3
initial apicid : 3
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

processor : 6
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping : 9
microcode : 0x12

cpu MHz : 1600.000
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 2
cpu cores : 4
apicid : 5
initial apicid : 5
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

processor : 7
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping : 9
microcode : 0x12
cpu MHz : 1600.000
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 3
cpu cores : 4
apicid : 7
initial apicid : 7
fdiv_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht

tm pbe nx rdtscp lm constant_tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips : 6784.78
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

```
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping        : 9
microcode       : 0x12
cpu MHz         : 1600.000
cache size      : 8192 KB
physical id     : 0
siblings        : 8
core id         : 1
cpu cores       : 4
apicid          : 2
initial apicid  : 2
fdiv_bug        : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_
tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqd
q dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb
xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips        : 6784.78
clflush size    : 64
cache_alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
power management:

processor       : 2
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping        : 9
microcode       : 0x12
cpu MHz         : 1600.000
cache size      : 8192 KB
```

```
cache size      : 8192 KB
physical id     : 0
siblings        : 8
core id         : 2
cpu cores       : 4
apicid          : 4
initial apicid  : 4
fdiv_bug        : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant_
tsc arch_perfmon pebs bts xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqd
q dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb
xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips        : 6784.78
clflush size    : 64
cache_alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
power management:

processor       : 3
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping        : 9
microcode       : 0x12
cpu MHz         : 1600.000
cache size      : 8192 KB
physical id     : 0
siblings        : 8
core id         : 3
cpu cores       : 4
apicid          : 6
initial apicid  : 6
```

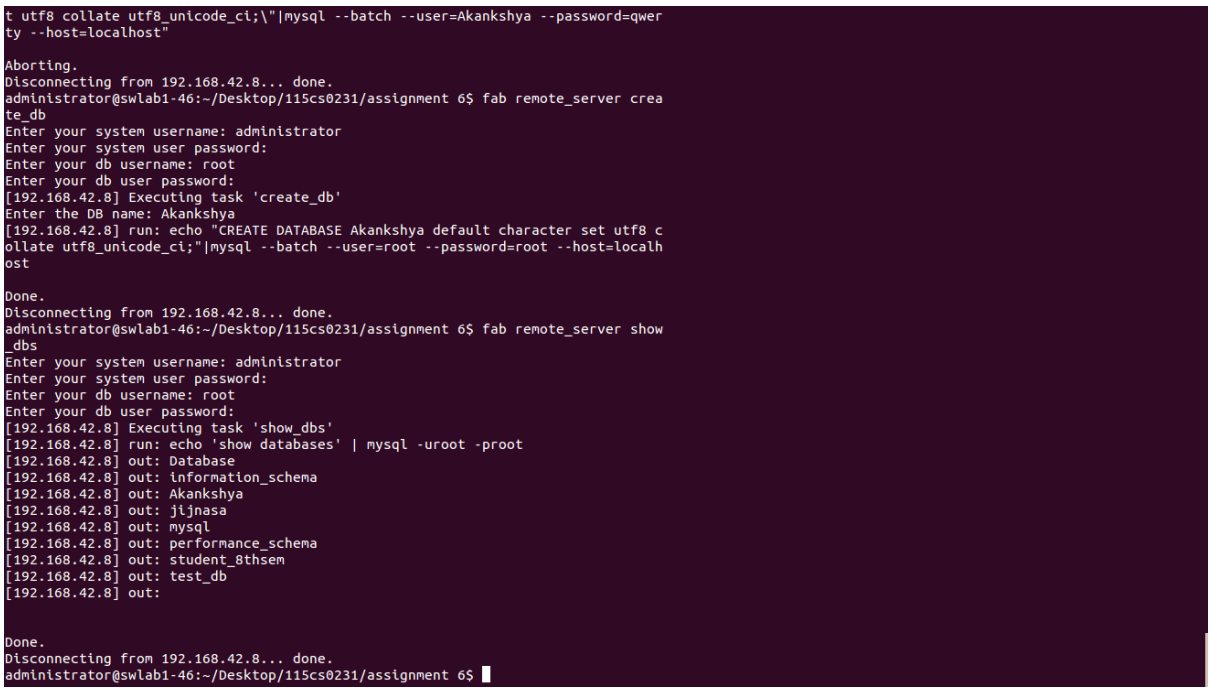
```
MemFree:         2235792 kB
Buffers:          101220 kB
Cached:           954068 kB
SwapCached:            0 kB
Active:           941372 kB
Inactive:         718280 kB
Active(anon):     606756 kB
Inactive(anon):   228352 kB
Active(file):     334616 kB
Inactive(file):   489928 kB
Unevictable:          32 kB
Mlocked:              32 kB
HighTotal:       3155288 kB
HighFree:        1773160 kB
LowTotal:         852736 kB
LowFree:          462632 kB
SwapTotal:       1967100 kB
SwapFree:        1967100 kB
Dirty:               108 kB
Writeback:             0 kB
AnonPages:        604684 kB
Mapped:           225476 kB
Shmem:            230748 kB
Slab:              70988 kB
SReclaimable:      48572 kB
SUnreclaim:        22416 kB
KernelStack:        4016 kB
PageTables:         8768 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     3971112 kB
Committed_AS:    3798232 kB
VmallocTotal:     122880 kB
VmallocUsed:       14896 kB
VmallocChunk:     100820 kB
HardwareCorrupted:     0 kB
AnonHugePages:    276480 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
```

[28] Demonstrate the use of executing a MYSQL command on remote server (i.e. you may create MYSQL database for the student,s of BTech 8th semester).

```
from getpass import getpass
from fabric.api import run, env, prompt, cd

def remote_server():
    env.hosts = ['192.168.42.8']
    env.user = prompt('Enter your system username: ')
    env.password = getpass('Enter your system user password: ')
    env.mysqlhost = 'localhost'
    env.mysqluser = prompt('Enter your db username: ')
    env.mysqlpassword = getpass('Enter your db user password: ')
    env.db_name = ''

def show_dbs():
    """ Wraps mysql show databases cmd"""
    q = "show databases"
    run("echo '%s' | mysql -u%s -p%s" %(q, env.mysqluser, env.mysqlpassword))


def run_sql(db_name, query):
    """ Generic function to run sql"""
```

```
    with cd('/tmp'):
        run("echo '%s' | mysql -u%s -p%s -D %s" %(query, env.mysqluser,
env.mysqlpassword, db_name))

def create_db():
    """Create a MySQL DB for App version"""
    if not env.db_name:
        db_name = prompt("Enter the DB name:")
    else:
        db_name = env.db_name
    run('echo "CREATE DATABASE %s default character set utf8 collate
utf8_unicode_ci;"|mysql --batch --user=%s --password=%s --host=%s'\
         % (db_name, env.mysqluser, env.mysqlpassword, env.mysqlhost),
pty=True)

def ls_db():
    """ List a dbs with size in MB """
    if not env.db_name:
        db_name = prompt("Which DB to ls?")
    else:
        db_name = env.db_name
    query = """SELECT table_schema                                    "DB Name",
       Round(Sum(data_length + index_length) / 1024 / 1024, 1) "DB Size in MB"
        FROM   information_schema.tables
        WHERE table_schema = \"%s\"
        GROUP  BY table_schema """ %db_name
    run_sql(db_name, query)


def empty_db():
    """ Empty all tables of a given DB """
    db_name = prompt("Enter DB name to empty:")
    cmd = """
    (echo 'SET foreign_key_checks = 0;';
    (mysqldump -u%s -p%s --add-drop-table --no-data %s |
     grep ^DROP);
     echo 'SET foreign_key_checks = 1;') | \
     mysql -u%s -p%s -b %s
```

```
    """ %(env.mysqluser, env.mysqlpassword, db_name, env.mysqluser,
env.mysqlpassword, db_name)
    run(cmd)
```

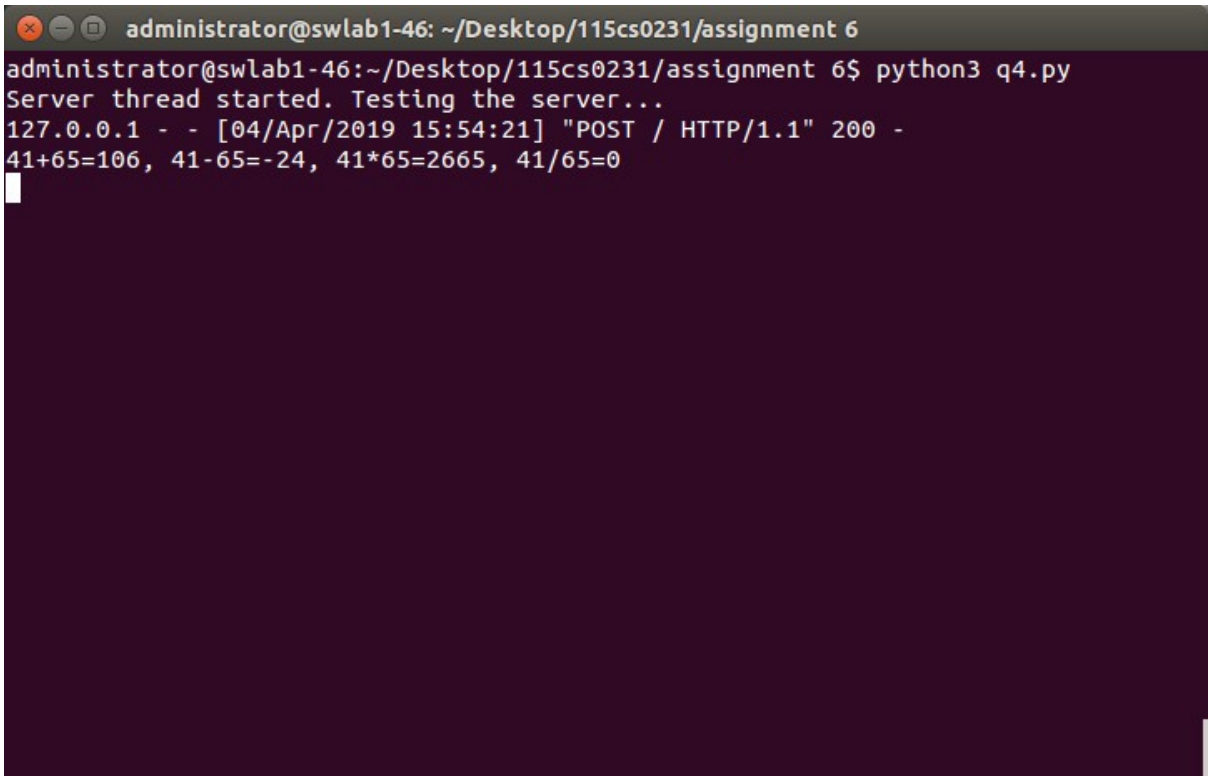


[29] Write a multithreaded, multi call XML-RPC server, so that multiple function calls can return a single result?

```
import argparse

import xmlrpc
# Comment out the above line and uncomment the below line for Python 2.x.
#import xmlrpclib
import threading

from xmlrpc.server import SimpleXMLRPCServer
# Comment out the above line and uncomment the below line for Python 2.x.
#from SimpleXMLRPCServer import SimpleXMLRPCServer

# some trivial functions
def add(x,y):
    return x+y

def subtract(x, y):
```

```
    return x-y

def multiply(x, y):
    return x*y

def divide(x, y):
    return x/y


class ServerThread(threading.Thread):
    def __init__(self, server_addr):
        threading.Thread.__init__(self)
        self.server = SimpleXMLRPCServer(server_addr)
        self.server.register_multicall_functions()
        self.server.register_function(add, 'add')
        self.server.register_function(subtract, 'subtract')
        self.server.register_function(multiply, 'multiply')
        self.server.register_function(divide, 'divide')

    def run(self):
        self.server.serve_forever()

def run_server(host, port):
    # server code
    server_addr = (host, port)
    server = ServerThread(server_addr)
    server.start() # The server is now running
    print ("Server thread started. Testing the server...")

def run_client(host, port):

    # client code
    proxy = xmlrpc.client.ServerProxy("http://%s:%s/" %(host, port))
    # Comment out the above line and uncomment the below line for Python 2.x.
    #proxy = xmlrpclib.ServerProxy("http://%s:%s/" %(host, port))

    multicall = xmlrpc.client.MultiCall(proxy)
    # Comment out the above line and uncomment the below line for Python 2.x.
    #multicall = xmlrpclib.MultiCall(proxy)

    multicall.add(41,65)
    multicall.subtract(41,65)
    multicall.multiply(41,65)
    multicall.divide(41,65)
    result = multicall()
    print ("41+65=%d, 41-65=%d, 41*65=%d, 41/65=%d" % tuple(result))

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Multithreaded multicall
XMLRPC Server/Proxy')
    parser.add_argument('--host', action="store", dest="host", default='localhost')
```

```
    parser.add_argument('--port', action="store", dest="port", default=8000,
type=int)
    # parse arguments
    given_args = parser.parse_args()
    host, port =  given_args.host, 5000
    run_server(host, port)
    run_client(host, port)
```



[30] In corporate a suitable authentication scheme to the serve designed in question number 29.

```
import argparse
import xmlrpc
from base64 import b64decode

from xmlrpc.server import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler


class SecureXMLRPCServer(SimpleXMLRPCServer):
```

```
    def __init__(self, host, port, username, password, *args, **kargs):
        self.username = username
        self.password = password

        class VerifyingRequestHandler(SimpleXMLRPCRequestHandler):

             def parse_request(request):
                if SimpleXMLRPCRequestHandler.parse_request(request):

                    if self.authenticate(request.headers):
                        return True
                    else:
                        # if authentication fails return 401
                        request.send_error(401, 'Authentication failed, Try agin.')
                return False
        # initialize
        SimpleXMLRPCServer.__init__(self, (host, port),
requestHandler=VerifyingRequestHandler, *args, **kargs)

    def authenticate(self, headers):
        headers = headers.get('Authorization').split()
        basic, encoded = headers[0], headers[1]
        if basic != 'Basic':
            print ('Only basic authentication supported')
            return False
        secret = b64decode(encoded).split(b':')

        username, password = secret[0].decode("utf-8"), secret[1].decode("utf-8")
        return True if (username == self.username and password == self.password)
else False


def run_server(host, port, username, password):
    server = SecureXMLRPCServer(host, port, username, password)

    def echo(msg):
        """Reply client in  uppser case """
        reply = msg.upper()
        print ("Client said: %s. So we echo that in uppercase: %s" %(msg, reply))
        return reply
    server.register_function(echo, 'echo')
    print ("Running a HTTP auth enabled XMLRPC server on %s:%s..." %(host,
port))
    server.serve_forever()


if __name__ == '__main__':

    host, port =  'localhost', 5500
    username, password = 'user1', 'pass1'
    run_server(host, port, username, password)
```

Client

```
import argparse
import xmlrpc

from xmlrpc.server import SimpleXMLRPCServer

def run_client(host, port, username, password):
    server = xmlrpc.client.ServerProxy('http://%s:%s@%s:%s' %(username,
password, host, port, ))
    password, host, port, ))
    msg = "hello server..."
    print ("Sending message to server: %s  " %msg)
    print ("Got reply: %s" %server.echo(msg))

if __name__ == '__main__':

    host, port =  'localhost', 5500
    username, password = 'user1', 'pass1'
    run_client(host, port, username, password)
```