Mawlana Bhashani Science & Technology University

Lab Report No : 03

Course Code : ICT3208

Course Title : Computer Network Lab

Report Name : Python for networking

Submitted by

Name : Maskur Al Shal Sabil

Id: IT18021

3rd year 2nd Semester

Session 2017-2018

Dept of ICT

MBSTU

Submitted To

Nazrul Islam

Assistant Professor

Dept of ICT

MBSTU

Networking Glossary: Before we begin discussing networking with any depth, we must define some common terms that you will see throughout this guide, and in other guides and documentation regarding networking.

Connection: In networking, a connection refers to pieces of related information that are transferred through a network. This generally infers that a connection is built before the data transfer (by following the procedures laid out in a protocol) and then is deconstructed at the end of the data transfer

- **Packet:** A packet is, generally speaking, the most basic unit that is transfered over a network. When communicating over a network, packets are the envelopes that carry your data (in pieces) from one end point to the other. Packets have a header portion that contains information about the packet including the source and destination, timestamps, network hops, etc. The main portion of a packet contains the actual data being transfered. It is sometimes called the body or the payload.

Network Interface: A network interface can refer to any kind of software interface to networking hardware. For instance, if you have two network cards in your computer, you can control and configure each network interface associated with them individually.A network interface may be associated with a physical device, or it may be a representation of a virtual interface. The "loopback" device, which is a virtual interface to the local machine, is an example of this.

LAN: LAN stands for "local area network". It refers to a network or a portion of a network that is not publicly accessible to the greater internet. A home or office network is an example of a LAN.

WAN: WAN stands for "wide area network". It means a network that is much more extensive than a LAN. While WAN is the relevant term to use to describe large, dispersed networks in general, it is usually meant to mean the internet, as a whole. If an interface is said to be connected to the WAN, it is generally assumed that it is reachable through the internet.

Protocol: A protocol is a set of rules and standards that basically define a language that devices can use to communicate. There are a great number of protocols in use extensively in networking, and they are often implemented in different layers. Some low level protocols are TCP, UDP, IP, and ICMP. Some familiar examples of application layer protocols, built on these lower protocols, are HTTP (for accessing web content), SSH, TLS/SSL, and FTP. Port: A port is an address on a single machine that can be tied to a specific piece of software. It is not a physical interface or location, but it allows your server to be able to communicate using more than one application.

Firewall: A firewall is a program that decides whether traffic coming into a server or going out should be allowed. A firewall usually works by creating rules for which type of traffic is acceptable on which ports. Generally, firewalls block ports that are not used by a specific application on a server.

NAT: NAT stands for network address translation. It is a way to translate requests that are incoming into a routing server to the relevant devices or servers that it knows about in the LAN. This is usually implemented in physical LANs as a way to route requests through one IP address to the necessary backend servers

- **VPN:** VPN stands for virtual private network. It is a means of connecting separate LANs through the internet, while maintaining privacy. This is used as a means of connecting remote systems as if they were on a local network, often for security reasons.

Interfaces: Interfaces are networking communication points for your computer. Each interface is associated with a physical or virtual networking device. Typically, your server will have one configurable network interface for each Ethernet or wireless internet card you have.
In addition, it will define a virtual network interface called the "loopback" or localhost interface. This is used as an interface to connect applications and processes on a single computer to other applications and processes. You can see this referenced as the "lo" interface in many tools. Many times,

administrators configure one interface to service traffic to the internet and another interface for a LAN or private network.

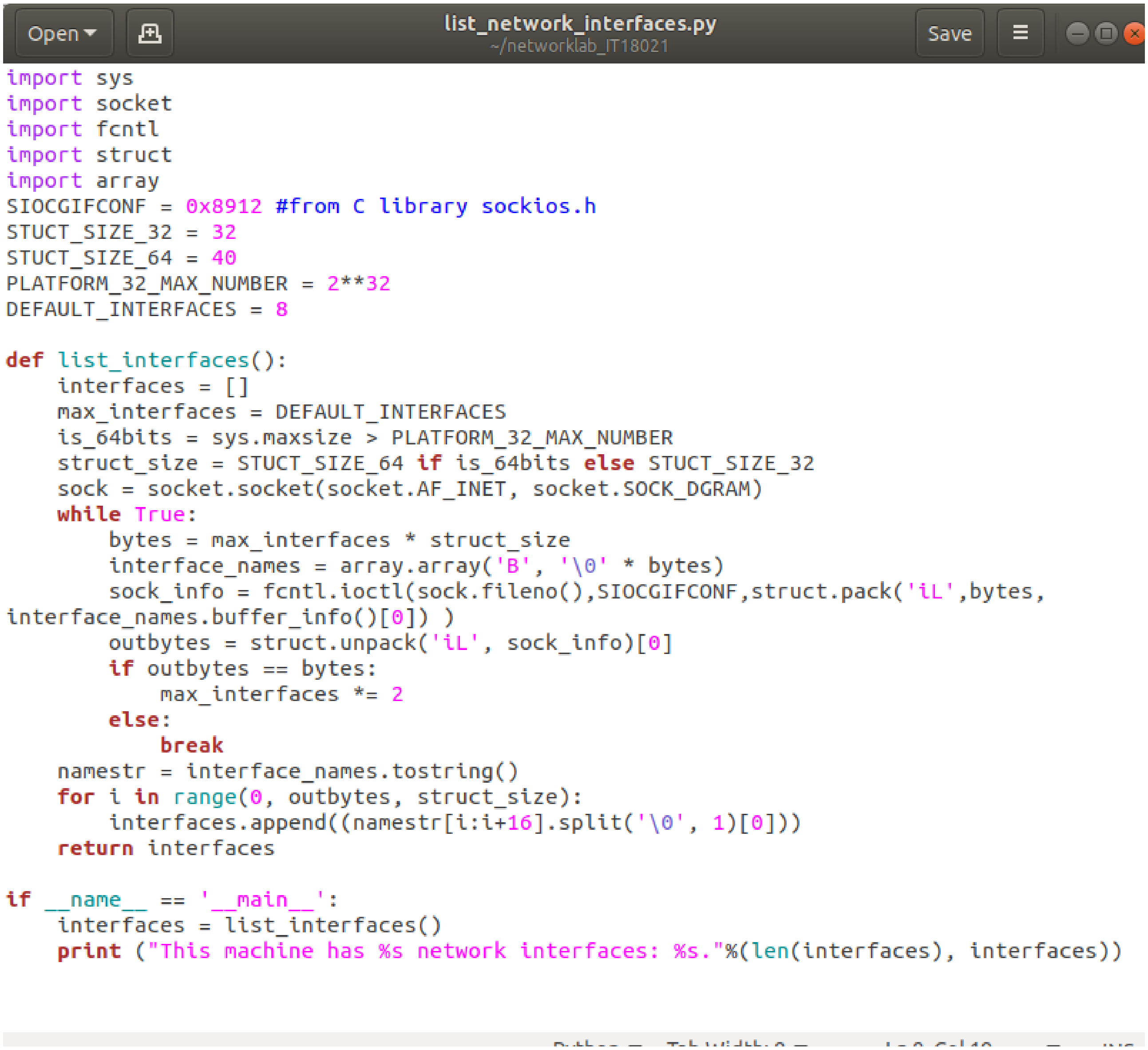
Protocols: Networking works by piggybacking a number of different protocols on top of each other. In this way, one piece of data can be transmitted using multiple protocols encapsulated within one another. We will talk about some of the more common protocols that you may come across and attempt to explain the difference, as well as give context as to what part of the process they are involved with. We will start with protocols implemented on the lower networking layers and work our way up to protocols with higher abstraction.

4. Exercises

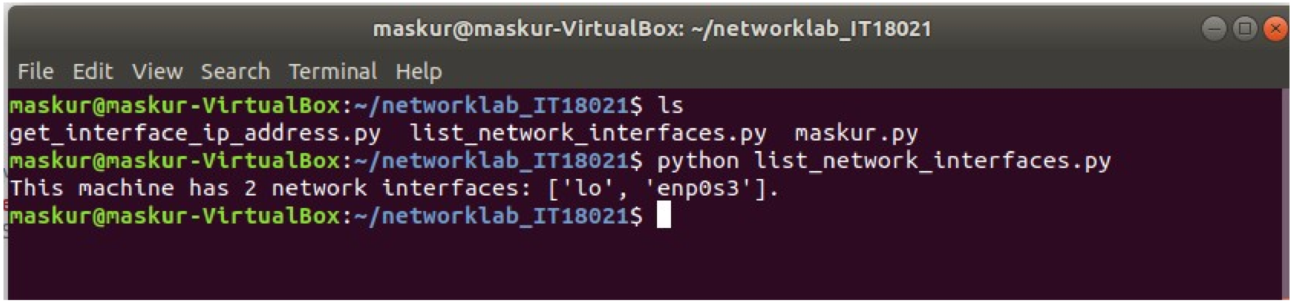
When importing a module if there is an error it means that the module needs to be installed.

Exercise 4.1: Enumerating interfaces on your machine

Create python scrip using the syntax below (save as list_network_interfaces.py):

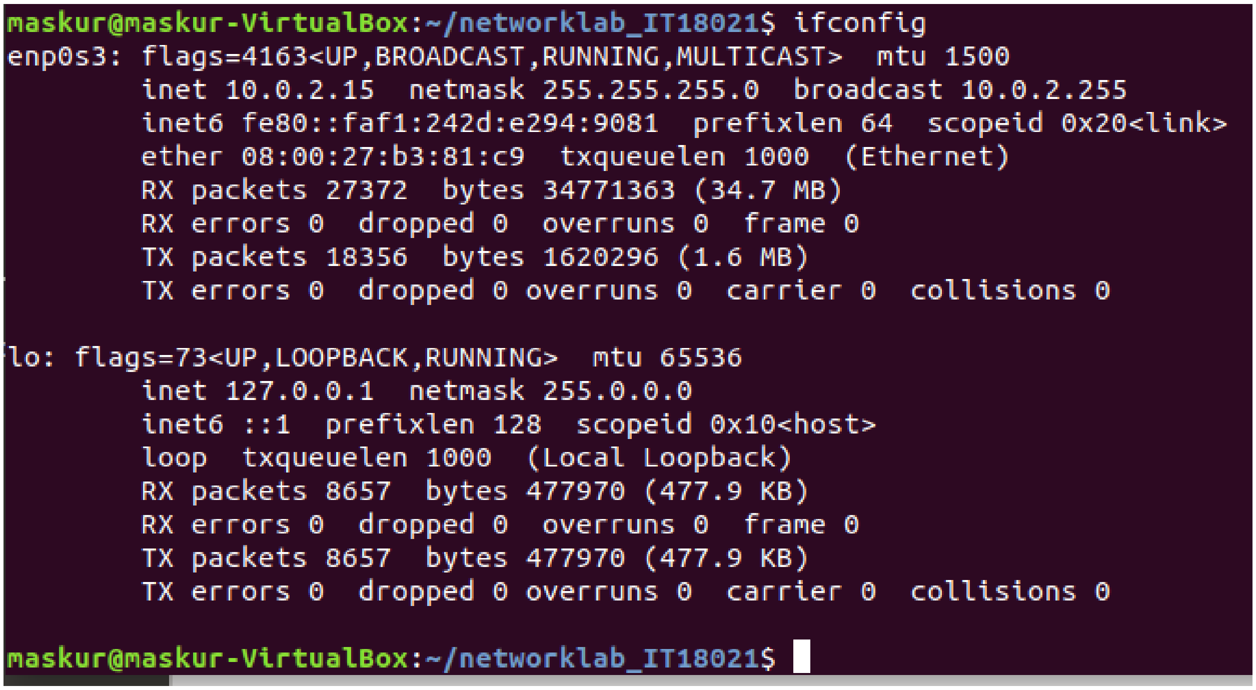


```
import sys
import socket
import fcntl
import struct
import array
SIOCGIFCONF = 0x8912 #from C library sockios.h
STUCT_SIZE_32 = 32
STUCT_SIZE_64 = 40
PLATFORM_32_MAX_NUMBER = 2**32
DEFAULT_INTERFACES = 8

def list_interfaces():
    interfaces = []
    max_interfaces = DEFAULT_INTERFACES
    is_64bits = sys.maxsize > PLATFORM_32_MAX_NUMBER
    struct_size = STUCT_SIZE_64 if is_64bits else STUCT_SIZE_32
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    while True:
        bytes = max_interfaces * struct_size
        interface_names = array.array('B', '\0' * bytes)
        sock_info = fcntl.ioctl(sock.fileno(),SIOCGIFCONF,struct.pack('iL',bytes,
interface_names.buffer_info()[0]) )
        outbytes = struct.unpack('iL', sock_info)[0]
        if outbytes == bytes:
            max_interfaces *= 2
        else:
            break
    namestr = interface_names.tostring()
    for i in range(0, outbytes, struct_size):
        interfaces.append((namestr[i:i+16].split('\0', 1)[0]))
    return interfaces

if __name__ == '__main__':
    interfaces = list_interfaces()
    print ("This machine has %s network interfaces: %s."%(len(interfaces), interfaces))
```


maskur@maskur-VirtualBox: ~/networklab_IT18021
File Edit View Search Terminal Help

```
maskur@maskur-VirtualBox:~/networklab_IT18021$ ls
get_interface_ip_address.py  list_network_interfaces.py  maskur.py
maskur@maskur-VirtualBox:~/networklab_IT18021$ python list_network_interfaces.py
This machine has 2 network interfaces: ['lo', 'enp0s3'].
maskur@maskur-VirtualBox:~/networklab_IT18021$ 
```

Verify output with ifconfig:



```
maskur@maskur-VirtualBox:~/networklab_IT18021$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::faf1:242d:e294:9081  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b3:81:c9  txqueuelen 1000  (Ethernet)
        RX packets 27372  bytes 34771363 (34.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18356  bytes 1620296 (1.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8657  bytes 477970 (477.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8657  bytes 477970 (477.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

maskur@maskur-VirtualBox:~/networklab_IT18021$ 
```

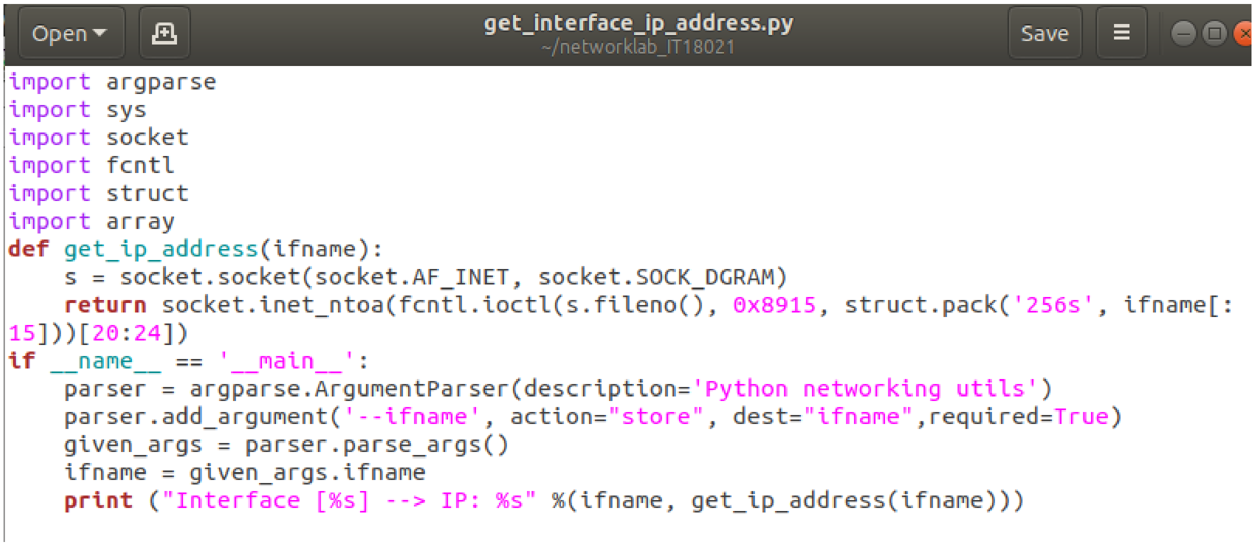
Exercise 4.2: Finding the IP address for a specific interface on your machine

Create python scrip using the syntax below (save as get_interface_ip_address.py):Run the script, which is the output? Which variable you need to provide? What is the

purpose of parse module?

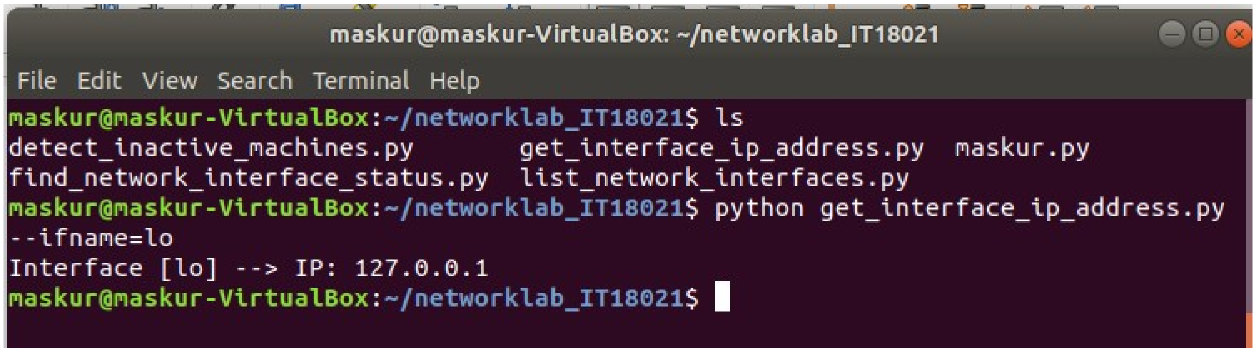
Here from the previous code my computer have two interface = ‘lo ,enp0s3’ . We need to pass the value of –ifname to get the interface ip.

A **parser** is a software component that takes input data (frequently text) and builds a data structure – often some kind of **parse** tree, abstract syntax tree or other hierarchical structure, giving a structural representation of the input while checking for correct syntax.



```
import argparse
import sys
import socket
import fcntl
import struct
import array
def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915, struct.pack('256s', ifname[:
15]))[20:24])
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Python networking utils')
    parser.add_argument('--ifname', action="store", dest="ifname",required=True)
    given_args = parser.parse_args()
    ifname = given_args.ifname
    print ("Interface [%s] --> IP: %s" %(ifname, get_ip_address(ifname)))
```

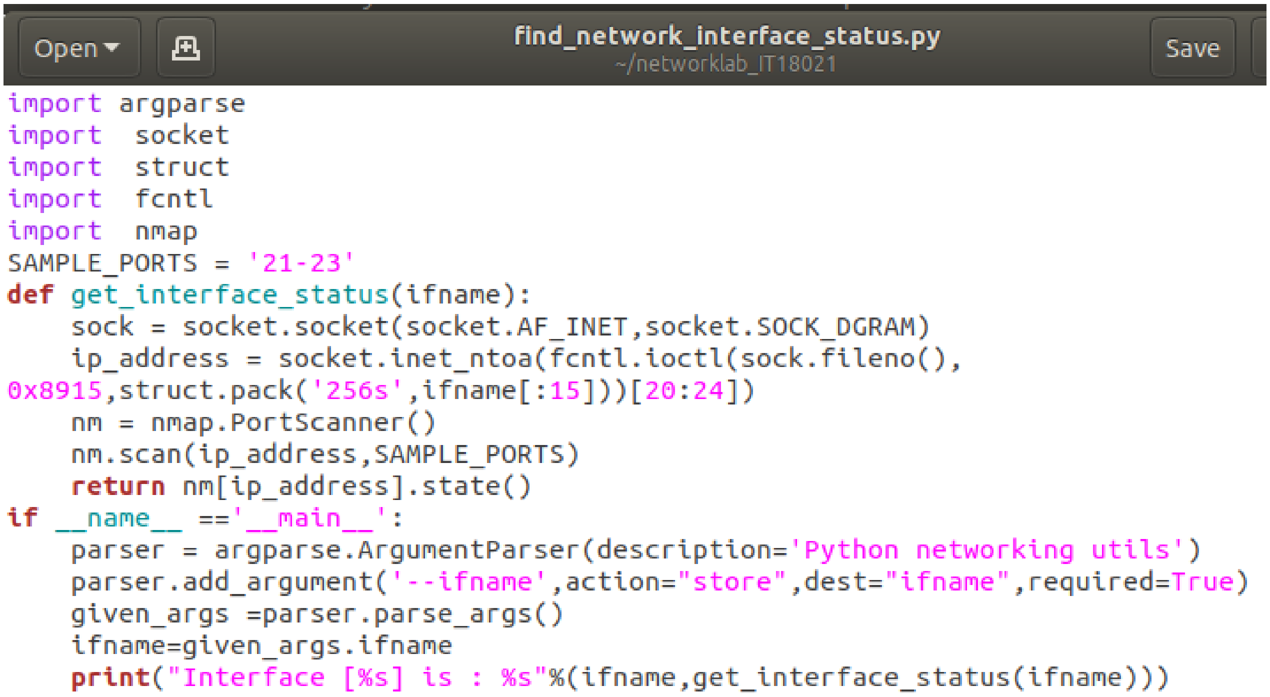
Output :


maskur@maskur-VirtualBox: ~/networklab_IT18021
File Edit View Search Terminal Help

```
maskur@maskur-VirtualBox:~/networklab_IT18021$ ls
detect_inactive_machines.py       get_interface_ip_address.py  maskur.py
find_network_interface_status.py  list_network_interfaces.py
maskur@maskur-VirtualBox:~/networklab_IT18021$ python get_interface_ip_address.py
--ifname=lo
Interface [lo] --> IP: 127.0.0.1
maskur@maskur-VirtualBox:~/networklab_IT18021$
```

Exercise 4.3: Finding whether an interface is up on your machine

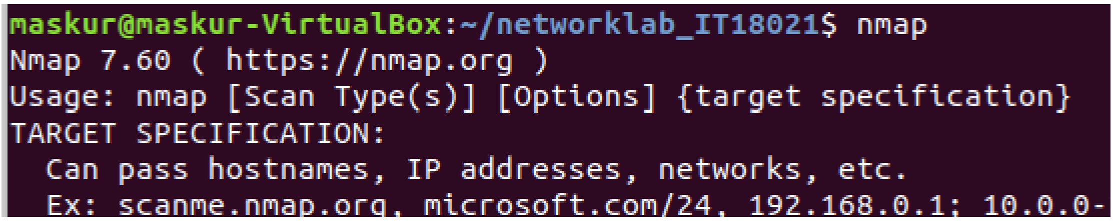
Create python scrip using the syntax below (save as find_network_interface_status.py):Run the script providing the interface to be tested, which is the output?



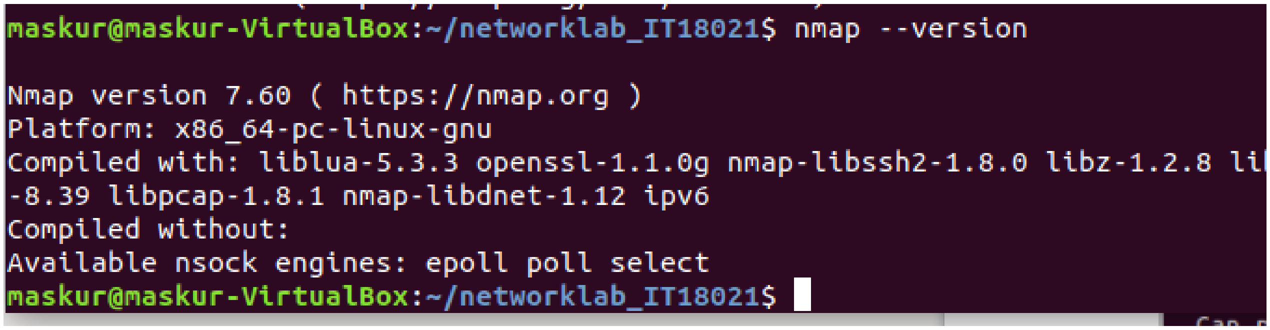
```
import argparse
import  socket
import  struct
import  fcntl
import  nmap
SAMPLE_PORTS = '21-23'
def get_interface_status(ifname):
    sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    ip_address = socket.inet_ntoa(fcntl.ioctl(sock.fileno(),
0x8915,struct.pack('256s',ifname[:15]))[20:24])
    nm = nmap.PortScanner()
    nm.scan(ip_address,SAMPLE_PORTS)
    return nm[ip_address].state()
if __name__ =='__main__':
    parser = argparse.ArgumentParser(description='Python networking utils')
    parser.add_argument('--ifname',action="store",dest="ifname",required=True)
    given_args =parser.parse_args()
    ifname=given_args.ifname
    print("Interface [%s] is : %s"%(ifname,get_interface_status(ifname)))
```

Here we need to install the nmap module . To do this Use : sudo apt-get install nmap

or : `$ pip install python-nmap`

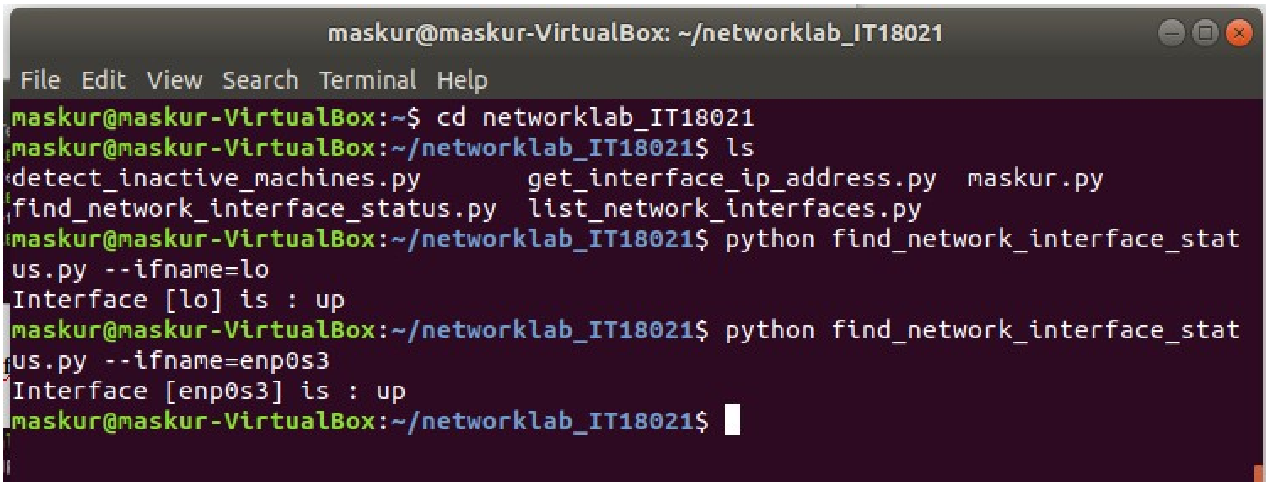


```
maskur@maskur-VirtualBox:~/networklab_IT18021$ nmap
Nmap 7.60 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-
```



```
maskur@maskur-VirtualBox:~/networklab_IT18021$ nmap --version

Nmap version 7.60 ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.3.3 openssl-1.1.0g nmap-libssh2-1.8.0 libz-1.2.8 li
-8.39 libpcap-1.8.1 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
maskur@maskur-VirtualBox:~/networklab_IT18021$
```

Now we run the program and the output is :



maskur@maskur-VirtualBox: ~/networklab_IT18021
File Edit View Search Terminal Help

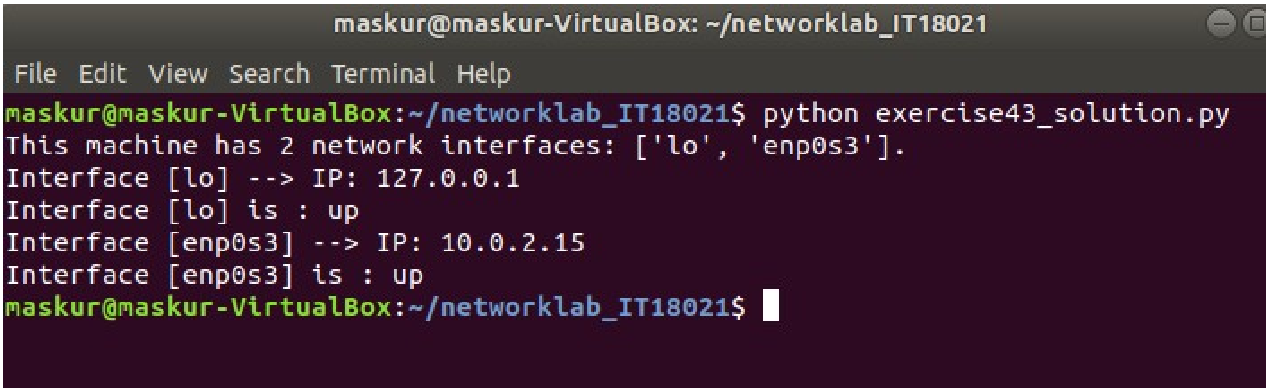
```
maskur@maskur-VirtualBox:~$ cd networklab_IT18021
maskur@maskur-VirtualBox:~/networklab_IT18021$ ls
detect_inactive_machines.py       get_interface_ip_address.py  maskur.py
find_network_interface_status.py  list_network_interfaces.py
maskur@maskur-VirtualBox:~/networklab_IT18021$ python find_network_interface_stat
us.py --ifname=lo
Interface [lo] is : up
maskur@maskur-VirtualBox:~/networklab_IT18021$ python find_network_interface_stat
us.py --ifname=enp0s3
Interface [enp0s3] is : up
maskur@maskur-VirtualBox:~/networklab_IT18021$ 
```

Write a script that provides the interfaces, IP and status (save as exercise43_solution.py)?

The code is given below :

```
import sys

import socket

import fcntl

import struct

import array

import argparse

import sys

import  nmap

#this is the starting  part of finding the interface

SIOCGIFCONF = 0x8912 #from C library sockios.h

STUCT_SIZE_32 = 32

STUCT_SIZE_64 = 40

PLATFORM_32_MAX_NUMBER = 2**32

DEFAULT_INTERFACES = 8


def list_interfaces():

    interfaces = []
    max_interfaces = DEFAULT_INTERFACES
    is_64bits = sys.maxsize > PLATFORM_32_MAX_NUMBER
    struct_size = STUCT_SIZE_64 if is_64bits else STUCT_SIZE_32
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    while True:
        bytes = max_interfaces * struct_size
        interface_names = array.array('B', '\0' * bytes)
        sock_info = fcntl.ioctl(sock.fileno(),SIOCGIFCONF,struct.pack('iL',bytes,
interface_names.buffer_info()[0]) )
        outbytes = struct.unpack('iL', sock_info)[0]
        if outbytes == bytes:
            max_interfaces *= 2
        else:
            break
    namestr = interface_names.tostring()
    for i in range(0, outbytes, struct_size):
        interfaces.append((namestr[i:i+16].split('\0', 1)[0]))
    return interfaces
#this method is to find the ip address of interface
def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915, struct.pack('256s', ifname[:15]))[20:24])
#this is the mthod for finding the status of interface
SAMPLE_PORTS = '21-23'
def get_interface_status(ifname):
    sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    ip_address = socket.inet_ntoa(fcntl.ioctl(sock.fileno(),0x8915,struct.pack('256s',ifname[:15]))
[20:24])
    nm = nmap.PortScanner()
    nm.scan(ip_address,SAMPLE_PORTS)
    return nm[ip_address].state()


if __name__ == '__main__':
    interfaces = list_interfaces()
    print ("This machine has %s network interfaces: %s."%(len(interfaces), interfaces))
    for i in interfaces:
        print ("Interface [%s] --> IP: %s" %(i, get_ip_address(i)))
            print("Interface [%s] is : %s"%(i,get_interface_status(i)))
```

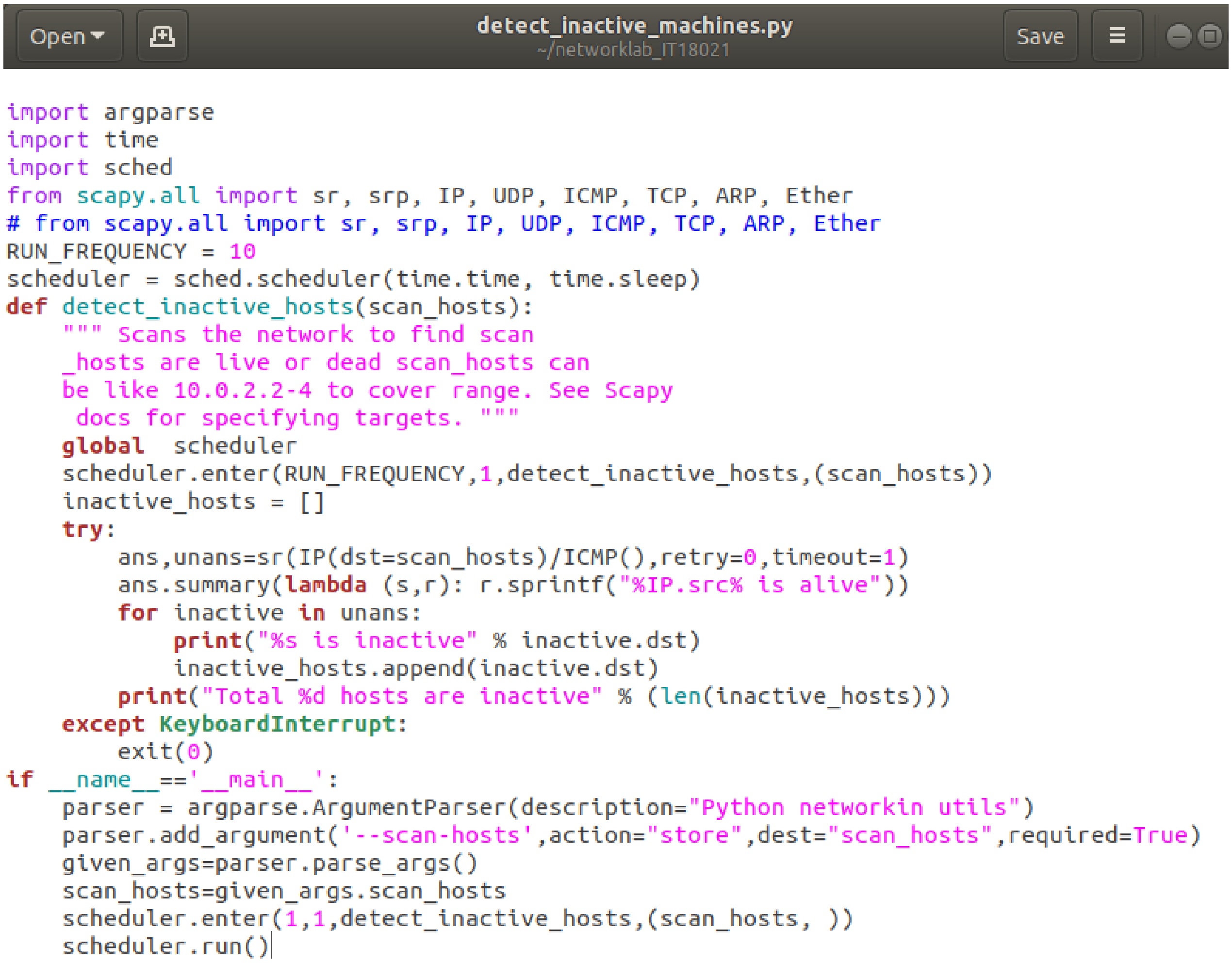
Output :


maskur@maskur-VirtualBox: ~/networklab_IT18021
File Edit View Search Terminal Help

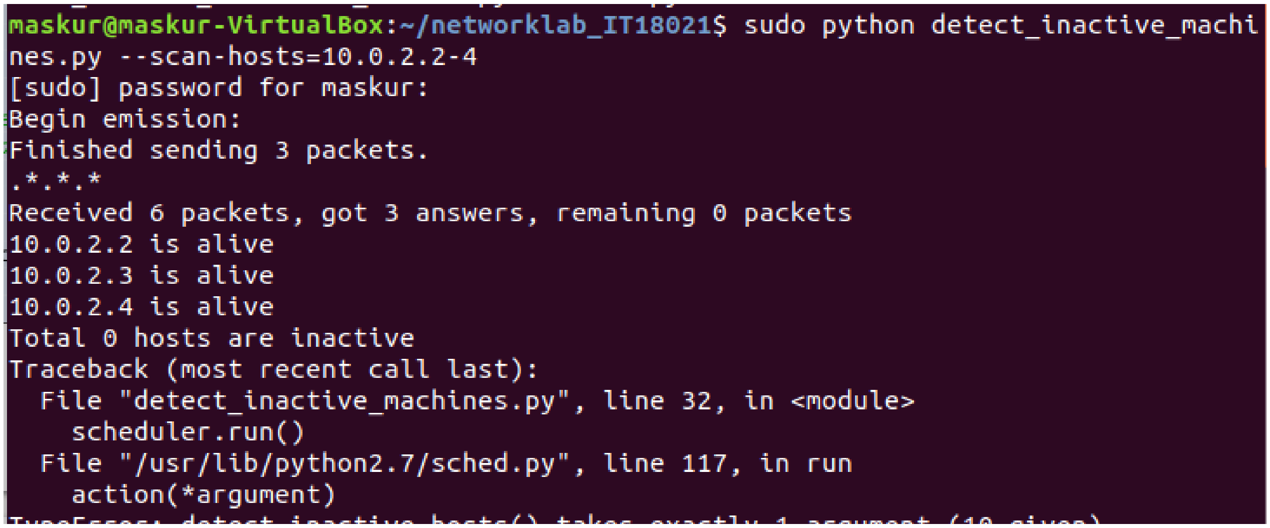
```
maskur@maskur-VirtualBox:~/networklab_IT18021$ python exercise43_solution.py
This machine has 2 network interfaces: ['lo', 'enp0s3'].
Interface [lo] --> IP: 127.0.0.1
Interface [lo] is : up
Interface [enp0s3] --> IP: 10.0.2.15
Interface [enp0s3] is : up
maskur@maskur-VirtualBox:~/networklab_IT18021$
```

Exercise 4.4: Detecting inactive machines on your network

Create python scrip using the syntax below (save as detect_inactive_machines.py):



```python
import argparse
import time
import sched
from scapy.all import sr, srp, IP, UDP, ICMP, TCP, ARP, Ether
# from scapy.all import sr, srp, IP, UDP, ICMP, TCP, ARP, Ether
RUN_FREQUENCY = 10
scheduler = sched.scheduler(time.time, time.sleep)
def detect_inactive_hosts(scan_hosts):
    """ Scans the network to find scan
     _hosts are live or dead scan_hosts can
    be like 10.0.2.2-4 to cover range. See Scapy
     docs for specifying targets. """
    global  scheduler
    scheduler.enter(RUN_FREQUENCY,1,detect_inactive_hosts,(scan_hosts))
    inactive_hosts = []
    try:
        ans,unans=sr(IP(dst=scan_hosts)/ICMP(),retry=0,timeout=1)
        ans.summary(lambda (s,r): r.sprintf("%IP.src% is alive"))
        for inactive in unans:
            print("%s is inactive" % inactive.dst)
            inactive_hosts.append(inactive.dst)
        print("Total %d hosts are inactive" % (len(inactive_hosts)))
    except KeyboardInterrupt:
        exit(0)
if __name__=='__main__':
    parser = argparse.ArgumentParser(description="Python networkin utils")
    parser.add_argument('--scan-hosts',action="store",dest="scan_hosts",required=True)
    given_args=parser.parse_args()
    scan_hosts=given_args.scan_hosts
    scheduler.enter(1,1,detect_inactive_hosts,(scan_hosts, ))
    scheduler.run()
```



```
maskur@maskur-VirtualBox:~/networklab_IT18021$ sudo python detect_inactive_machi
nes.py --scan-hosts=10.0.2.2-4
[sudo] password for maskur:
Begin emission:
Finished sending 3 packets.
.*.*.*
Received 6 packets, got 3 answers, remaining 0 packets
10.0.2.2 is alive
10.0.2.3 is alive
10.0.2.4 is alive
Total 0 hosts are inactive
Traceback (most recent call last):
  File "detect_inactive_machines.py", line 32, in <module>
    scheduler.run()
  File "/usr/lib/python2.7/sched.py", line 117, in run
    action(*argument)
TypeError: detect_inactive_hosts() takes exactly 1 argument (10 given)
```

Discussion : Really this lab report is not so easy to me as the previous It takes time and hard work to do this . I was stuck at the first moment how to install the fcntl module .In my pycharm all module can easily be installed except this one . It shows error . After trying long hours I came back to my virual linux and write the code again and hopefully completed this .