



# MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh, Tangail-1902

# LAB REPORT

Lab No : 03

Lab name : Python for Networking

Course Title : Network Planning and Designing Lab

Course Code : ICT-3208

Submitted by,

Zahid Hasan Chowdhury  
IT-18017  
3rd year 2nd semester  
Dept. of ICT

Submitted to,

Nazrul Islam  
Assistant professor  
Dept. of ICT,  
MBSTU

# Lab 3: Python for Networking

## 1. Objectives

The objective of the lab 3 is to:

1. Install python and use third-party libraries
2. Interact with network interfaces using python
3. Getting information from internet using Python

## 2. Theory

Third-party libraries: Although the Python's standard library provides a great set of awesome functionalities, there will be times that you will eventually run into the need of making use of third party libraries. Can you imagine building a webserver from scratch? Or making a port to a database driver? Or, maybe, coming up with an image manipulation tool?. Third party libraries are welcome in a way that they prevent you from reinventing the something that exist. They save you time to focus on finishing and delivering your application.

The following is a list of those third-party libraries with their download URLs used in networking:

- ntplib: <https://pypi.python.org/pypi/ntplib/>
- diesel: <https://pypi.python.org/pypi/diesel/>
- nmap: <https://pypi.python.org/pypi/python-nmap>
- scapy: <https://pypi.python.org/pypi/scapy>
- netifaces: <https://pypi.python.org/pypi/netifaces/>
- netaddr: <https://pypi.python.org/pypi/netaddr>
- pyopenssl: <https://pypi.python.org/pypi/pyOpenSSL>
- pygeocoder: <https://pypi.python.org/pypi/pygocoder>
- pyyaml: <https://pypi.python.org/pypi/PyYAML>
- requests: <https://pypi.python.org/pypi/requests>
- feedparser: <https://pypi.python.org/pypi/feedparser>
- paramiko: <https://pypi.python.org/pypi/paramiko/>
- fabric: <https://pypi.python.org/pypi/Fabric>
- supervisor: <https://pypi.python.org/pypi/supervisor>
- xmlrpclib: <https://pypi.python.org/pypi/xmlrpclib>
- SOAPpy: <https://pypi.python.org/pypi/SOAPpy>
- bottlenose: <https://pypi.python.org/pypi/bottlenose>
- construct: <https://pypi.python.org/pypi/construct/>
- pyserial: <https://pypi.python.org/pypi/pyserial>

**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 transferred 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 transferred. 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.

### 3. Methodology

Installing Python Third-party includes:

Python Third-party includes a setup.py file, it is usually distributed as a tarball (.tar.gz or .tar.bz2 file). The instructions for installing these generally look like:

- Download the file from website.
- Extract the tarball.
- Change into the new directory that has been newly extracted.
- Run `sudo python setup.py build`
- Run `sudo python setup.py install`

### 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`):

Run the script, which is the output?

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

[Running] python -u "c:\Users\User-PC\list_network_interfaces.py"
This machine has 2 network interfaces: ['lo', 'wifi0']

[Done] exited with code=0 in 0.069 seconds
```

Verify the output using the command line ifconfig, which is the output?

```
zahid@DESKTOP-DF9QV8M: ~
zahid@DESKTOP-DF9QV8M:~$ ifconfig
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::61aa:bda4:c241:8485 prefixlen 64 scopeid 0xfd<compat,link,site,host>
    ether 0a:00:27:00:00:07 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 1500
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0xfe<compat,link,site,host>
    loop (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wifi0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.102 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::1186:ba95:e44b:5a42 prefixlen 64 scopeid 0xfd<compat,link,site,host>
    ether 44:03:2c:e9:6c:68 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

zahid@DESKTOP-DF9QV8M:~$
```

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

Create python script using the syntax below (save as get\_interface\_ip\_address.py):

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

[Running] python -u "c:\Users\User-PC\list_network_interfaces.py"
Interface [wifi0] --> IP: 192.168.0.102

[Done] exited with code=0 in 0.071 seconds
```

Run the script, which is the output? Which variable do you need to provide? What is the purpose of parse module?

```
zahid@DESKTOP-DF9QV8M: ~  
zahid@DESKTOP-DF9QV8M:~$ ifconfig  
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255  
    inet6 fe80::61aa:bda4:c241:8485 prefixlen 64 scopeid 0xfd<compat,link,site,host>  
    ether 0a:00:27:00:00:07 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 1500  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0xfe<compat,link,site,host>  
    loop (Local Loopback)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wifio: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.102 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::1186:ba95:e44b:5a42 prefixlen 64 scopeid 0xfd<compat,link,site,host>  
    ether 44:03:2c:e9:6c:68 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
zahid@DESKTOP-DF9QV8M:~$
```

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

Create python script using the syntax below (save as find\_network\_interface\_status.py):

Run the script providing the interface to be tested, which is the output? Write a script that provides the interfaces, IP and status (save as exercise43\_solution.py)?

```
C:\Users\User-PC>list_network_interfaces.py  
Traceback (most recent call last):  
  File "C:\Users\User-PC\list_network_interfaces.py", line 4, in <module>  
    import fcntl  
ModuleNotFoundError: No module named 'fcntl'  
  
C:\Users\User-PC>
```

#### Exercise 4.4: Detecting inactive machines on your network

Create python script using the syntax below (save as detect\_inactive\_machines.py):

Add a range of IP addresses and run the script, which is the output? Ask your tutor for help.

```
C:\Users\User-PC>list_network_interfaces.py
Traceback (most recent call last):
  File "C:\Users\User-PC\list_network_interfaces.py", line 4, in <module>
    import fcntl
ModuleNotFoundError: No module named 'fcntl'

C:\Users\User-PC>
```

#### Exercise 4.5: Pinging hosts on the network with ICMP

Create python script using the syntax below (save as ping\_remote\_host.py):

How many functions does the code have? Which is the command for running the script (Target host is WWW.GOOGLE.COM)

```
$ sudo python 3_2_ping_remote_host.py --target-host=www.google.com
Ping to www.google.com... Get pong in 7.5634ms
Ping to www.google.com... Get pong in 7.2694ms
Ping to www.google.com... Get pong in 7.8254ms
Ping to www.google.com... Get pong in 7.7845ms
```

#### Exercise 4.6: Pinging hosts on the network with ICMP using pc resources

Create python scrip using the syntax below (save as ping\_subprocess.py):

Add the IP address of your PC in the code and run the script, which is the output?

Test the script with the IP of you classmate?

What is the role of subprocess?

```
Pinger > send_ping() > try
Run: ping_remote_host x
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/tanvi/Py
File "C:/Users/tanvi/PycharmProjects/Network_Lab/src/Lab-03_Networking_with_Python
except socket.error, (errno, msg):
    ^
SyntaxError: invalid syntax
```

#### Exercise 4.7: Scanning the broadcast of packets

Create python script using the syntax below (save as broadcast\_scanning.py):

Run the script, which is the output?

```
C:\Users\User-PC>list_network_interfaces.py
Traceback (most recent call last):
  File "C:\Users\User-PC\list_network_interfaces.py", line 4, in <module>
    import fcntl
ModuleNotFoundError: No module named 'fcntl'

C:\Users\User-PC>
```

#### Exercise 4.8: Sniffing packets on your network

Tcpdump is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to

which the computer is attached. Distributed under the BSD license,[3] tcpdump is free Software.

- o Open a linux terminal and check the usage of tcpdump using the command  
line `tcpdump -help`
- o Using tcpdump get the traffic present in the Ethernet interface of your pc  
(10 packet only), which is the command line?
- o Using the subprocess write a program for sniffing 1 packet of the  
Ethernet interface? (Save as `packet_sniffer.py`).

```
C:\Users\User-PC>list_network_interfaces.py
Traceback (most recent call last):
  File "C:\Users\User-PC\list_network_interfaces.py", line 4, in <module>
    import fcntl
ModuleNotFoundError: No module named 'fcntl'

C:\Users\User-PC>
```



