Python Internet Services

Source: https://www.tutorialspoint.com

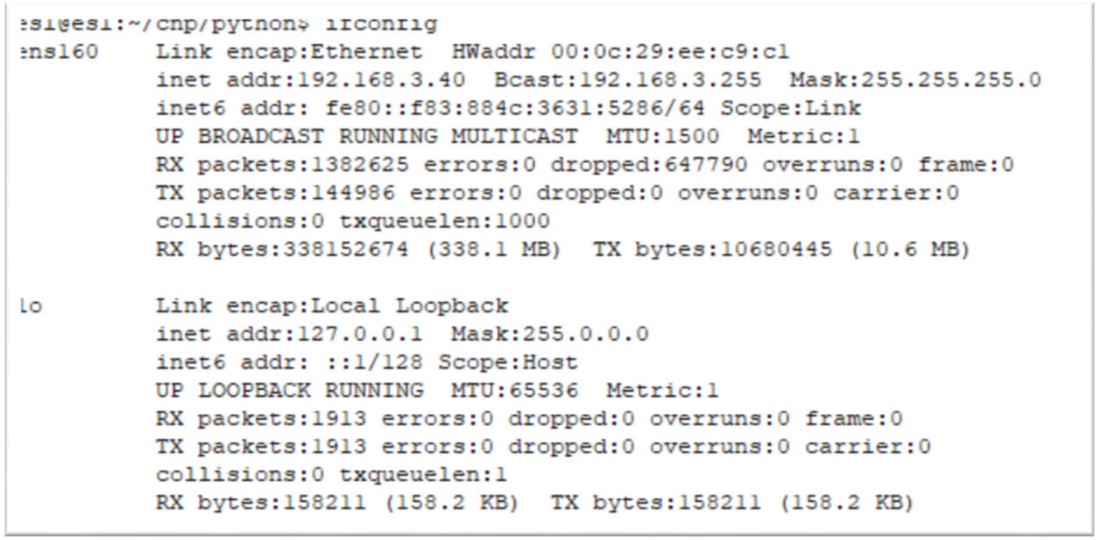
Outline

- Python - Network Interface
- Python HTTP Services
    - HTTP Server
    - HTTP Client
- Python FTP Services
    - Listing the Files
    - Changing the Directory
    - Fetching the Files
    - Reading the Data
- Python NTP Client
- Packet Sniffer

Python - Network Interface

Python - Network Interface (listNIC.py)

- When we have multiple interfaces in a machine we need to keep track of their names, status etc. When we using the net-tool ifconfig .
    - $ ifconfig



```
esi@esi:~/cnp/python$ ifconfig
ens160    Link encap:Ethernet  HWaddr 00:0c:29:ee:c9:c1
          inet addr:192.168.3.40  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::f83:884c:3631:5286/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1382625 errors:0 dropped:647790 overruns:0 frame:0
          TX packets:144986 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:338152674 (338.1 MB)  TX bytes:10680445 (10.6 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1913 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1913 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:158211 (158.2 KB)  TX bytes:158211 (158.2 KB)
```

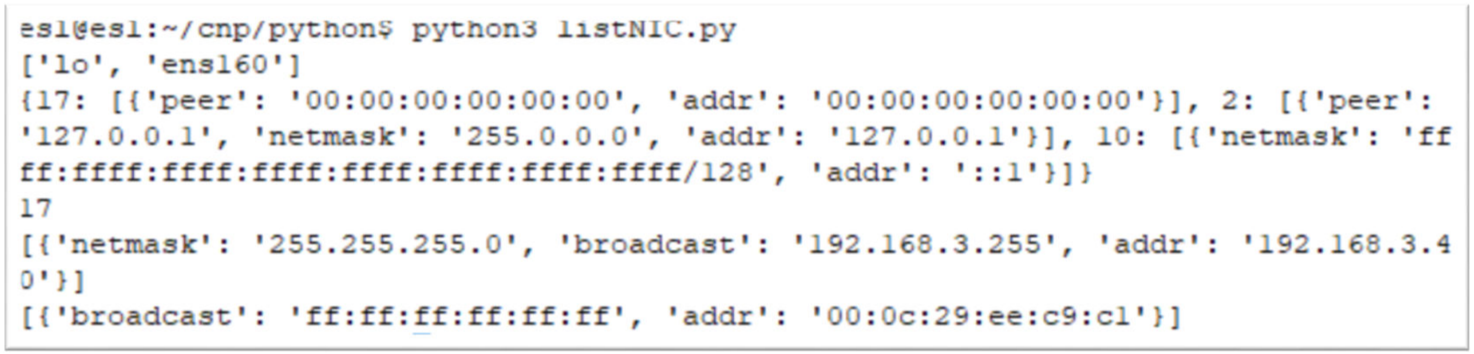
Python - Network Interface (listNIC.py)

- In Python we can list the interfaces and their status. We need to install the module netifaces to acheive this.
  - sudo pip3 install netifaces
- In the below example we use the python module **netifaces** which gives the details of the interfaces and their status. The methods used are very simple and straight forward.

```
import netifaces
print (netifaces.interfaces())
print (netifaces.ifaddresses('lo'))
print (netifaces.AF_LINK)
addrs = netifaces.ifaddresses('ens160')
print(addrs[netifaces.AF_INET])
print(addrs[netifaces.AF_LINK])
```

Python - Network Interface (listNIC.py)

- When we run the above program, we get the following output –



```
es1@es1:~/cnp/python$ python3 listNIC.py
['lo', 'ens160']
{17: [{'peer': '00:00:00:00:00:00', 'addr': '00:00:00:00:00:00'}], 2: [{'peer':
'127.0.0.1', 'netmask': '255.0.0.0', 'addr': '127.0.0.1'}], 10: [{'netmask': 'ff
ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff/128', 'addr': '::1'}]}
17
[{'netmask': '255.255.255.0', 'broadcast': '192.168.3.255', 'addr': '192.168.3.4
0'}]
[{'broadcast': 'ff:ff:ff:ff:ff:ff', 'addr': '00:0c:29:ee:c9:c1'}]
```

Python HTTP Services

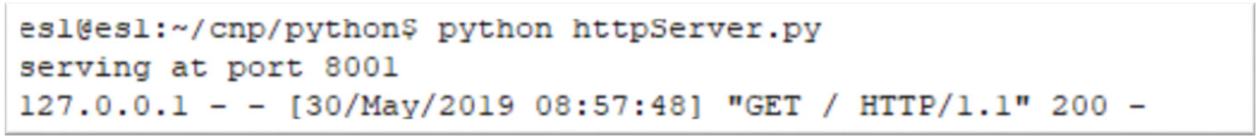
Python - HTTP Server (httpServer.py)

- Python standard library comes with a in-built webserver which can be invoked for simple web client server communication.
- The port number can be assigned programmatically and the web server is accessed through this port. Though it is not a full featured web server which can parse many kinds of file, it can parse simple static html files and serve them by responding them with required response codes.
- The below program starts a simple web server and opens it up at port 8001. The successful running of the server is indicated by the response code of 200 as shown in the program output.

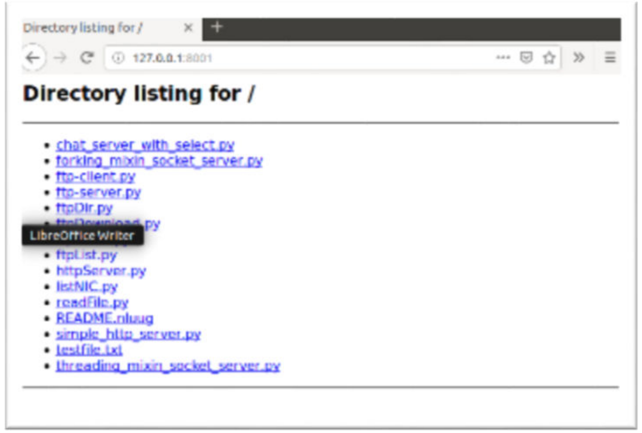
Python - HTTP Server (httpServer.py)

```
import SimpleHTTPServer
import SocketServer
PORT = 8001
Handler = SimpleHTTPServer.SimpleHTTPRequestHandler
httpd = SocketServer.TCPServer(("", PORT), Handler)
print "serving at port", PORT
httpd.serve_forever()
```

- When we run the above program, we get the following output –



```
esl@esl:~/cnp/python$ python httpServer.py
serving at port 8001
127.0.0.1 - - [30/May/2019 08:57:48] "GET / HTTP/1.1" 200 -
```



Directory listing for /

127.0.0.1:8001

Directory listing for /

- chat_server_with_select.py
- forking_mixin_socket_server.py
- ftp-client.py
- ftp-server.py
- ftpDir.py
- [illegible].py
- [illegible]
- ftpList.py
- httpServer.py
- listNIC.py
- readFile.py
- README.nluug
- simple_http_server.py
- testfile.txt
- threading_mixin_socket_server.py

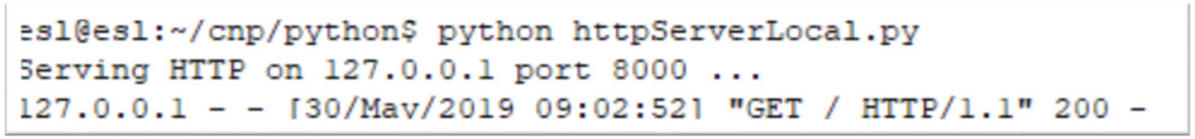
Serving a localhost (httpServerLocal.py)

- If we decide to make the python server as a local host serving only the local host, then we can use the following program to do that.

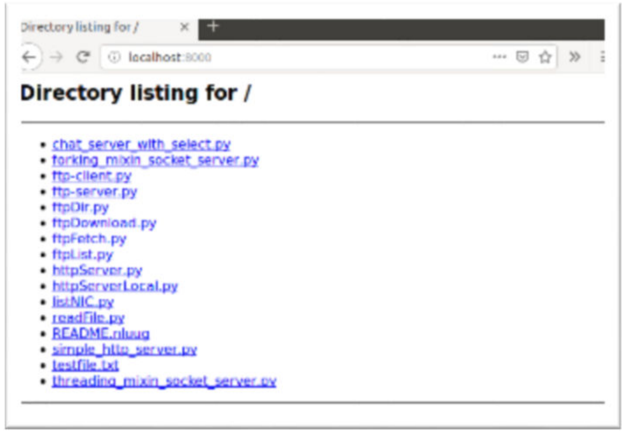
```
import sys
import BaseHTTPServer
from SimpleHTTPServer import SimpleHTTPRequestHandler
HandlerClass = SimpleHTTPRequestHandler
ServerClass  = BaseHTTPServer.HTTPServer
Protocol     = "HTTP/1.0"
if sys.argv[1:]:
    port = int(sys.argv[1])
else:
    port = 8000
server_address = ('127.0.0.1', port)
HandlerClass.protocol_version = Protocol
httpd = ServerClass(server_address, HandlerClass)
sa = httpd.socket.getsockname()
print "Serving HTTP on", sa[0], "port", sa[1], "..."
httpd.serve_forever()
```

Serving a localhost (httpServerLocal.py)

- When we run the above program, we get the following output –



```
esl@esl:~/cnp/python$ python httpServerLocal.py
Serving HTTP on 127.0.0.1 port 8000 ...
127.0.0.1 - - [30/May/2019 09:02:52] "GET / HTTP/1.1" 200 -
```



Directory listing for /

- chat_server_with_select.py
- forking_mixin_socket_server.py
- ftp-client.py
- ftp-server.py
- ftpDir.py
- ftpDownload.py
- ftpFetch.py
- ftpList.py
- httpServer.py
- httpServerLocal.py
- listNIC.py
- readFile.py
- README.nluug
- simple_http_server.py
- testfile.txt
- threading_mixin_socket_server.py

Python - HTTP Client (httpClient1.py)

- In the http protocol, the request from the client reaches the server and fetches some data and metadata assuming it is a valid request. We can analyze this response from the server using various functions available in the python requests module. Here the below python programs run in the client side and display the result of the response sent by the server.

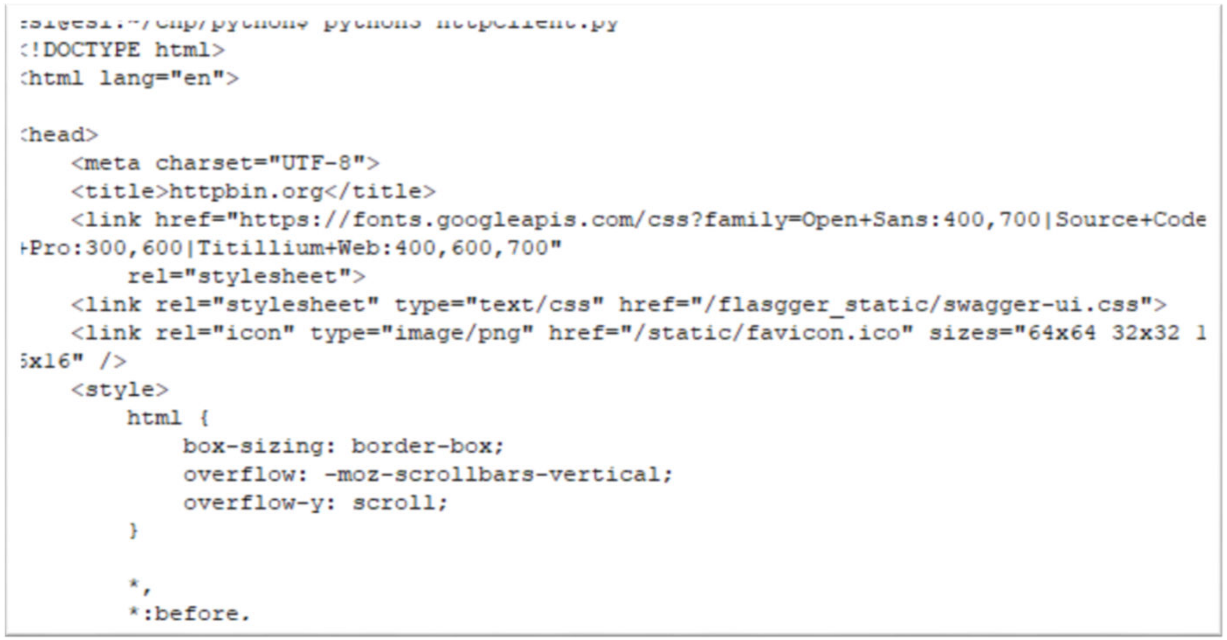
Get Initial Response (httpClient1.py)

- In the below program the get method from requests module fetches the data from a server and it is printed in plain text format. So we install the module into our python environment with the below command.
  - sudo pip3 install requests

```
import requests
r = requests.get('https://httpbin.org’)
print(r.text)[:200]
```

Get Initial Response (httpClient1.py)

- When we run the above program, we get the following output –



```
:sieesi:~/tmp/python$ python3 httpclient.py
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>httpbin.org</title>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans:400,700|Source+Code
+Pro:300,600|Titillium+Web:400,600,700"
        rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="/flasgger_static/swagger-ui.css">
    <link rel="icon" type="image/png" href="/static/favicon.ico" sizes="64x64 32x32 1
6x16" />
    <style>
        html {
            box-sizing: border-box;
            overflow: -moz-scrollbars-vertical;
            overflow-y: scroll;
        }

        *,
        *:before,
```

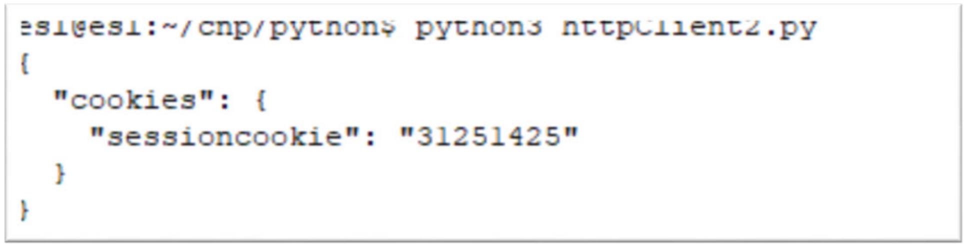
Get Session Object Response (httpClinet2.py)

- The Session object allows you to persist certain parameters across requests. It also persists cookies across all requests made from the Session instance. If you’re making several requests to the same host, the underlying TCP connection will be reused.

```
import requests
s = requests.Session()
s.get('http://httpbin.org/cookies/set/sessioncookie/31251425')
r = s.get('http://httpbin.org/cookies')
print(r.text)
```

Get Session Object Response (httpClinet2.py)

- When we run the above program, we get the following output –



```
esi@esi:~/cnp/python$ python3 httpClient2.py
{
  "cookies": {
    "sessioncookie": "31251425"
  }
}
```

Handling Error

- In case some error is raised because of issue in processing the request by the server, the python program can gracefully handle the exception raised using the timeout parameter as shown below.
- The program will wait for the defined value of the timeout error and then raise the time out error.
    - requests.get('http://github.com', timeout=10.001)

Python FTP Services

File Transfer Protocol

- **FTP** or File Transfer Protocol is a well-known network protocol used to transfer files between computers in a network.
- It is created on client server architecture and can be used along with user authentication.
- It can also be used without authentication but that will be less secure. FTP connection which maintains a current working directory and other flags, and each transfer requires a secondary connection through which the data is transferred.
- Most common web browsers can retrieve files hosted on FTP servers.

File Transfer Protocol

- The **ftplib** module is a simple interface to FTP. The module offers high level abstractions and task based routines to handle the FTP needs. So we install the module into our python environment with the below command.
    - sudo pip3 install ftplib

The Methods in FTP class

pwd()	Current working directory.
cwd()	Change current working directory to path.
dir([path[,...[,cb]])	Displays directory listing of path. Optional call-back cb passed to retrlines().
storlines(cmd, f)	Uploads text file using given FTP cmd - for example, STOR file name.
storbinary(cmd,f[, bs=8192])	Similar to storlines() but is used for binary files.
delete(path)	Deletes remote file located at path.
mkd(directory)	Creates remote directory.
exception ftplib.error_temp	Exception raised when an error code signifying a temporary error (response codes in the range 400–499) is received..
exception ftplib.error_perm	Exception raised when an error code signifying a permanent error (response codes in the range 500–599) is received..
connect(host[, port[, timeout]])	Connects to the given host and port. The default port number is 21, as specified by the FTP protocol..
quit()	Closes connection and quits.

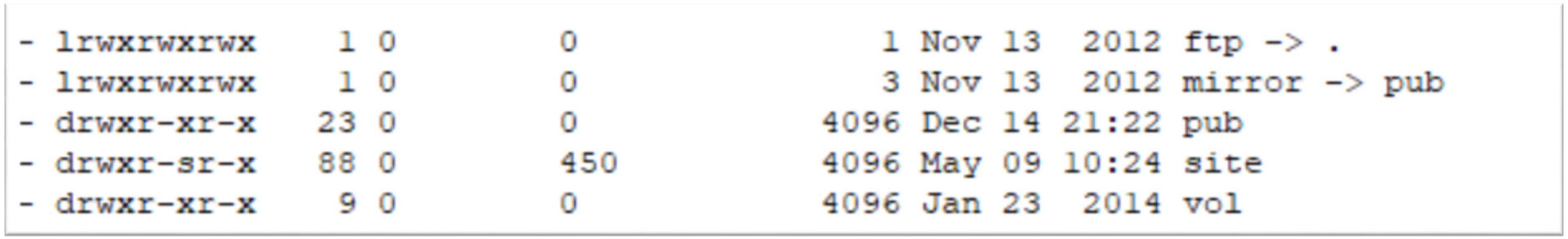
Listing the Files (ftpList.py)

- The below example uses anonymous login to the ftp server and lists the content of the current directory. It treates through the name of the files and directories and stores them as a list. Then prints them out.

```
import ftplib
 ftp = ftplib.FTP("ftp.nluug.nl")
ftp.login("anonymous", "ftplib-example-1")
 data = []
 ftp.dir(data.append)
 ftp.quit()
 for line in data:
    print "-", line
```

Listing the Files (ftpList.py)

- When we run the above program, we get the following output –



```
- lrwxrwxrwx    1 0         0                1 Nov 13  2012 ftp -> .
- lrwxrwxrwx    1 0         0                3 Nov 13  2012 mirror -> pub
- drwxr-xr-x   23 0         0             4096 Dec 14 21:22 pub
- drwxr-sr-x   88 0         450           4096 May 09 10:24 site
- drwxr-xr-x    9 0         0             4096 Jan 23  2014 vol
```

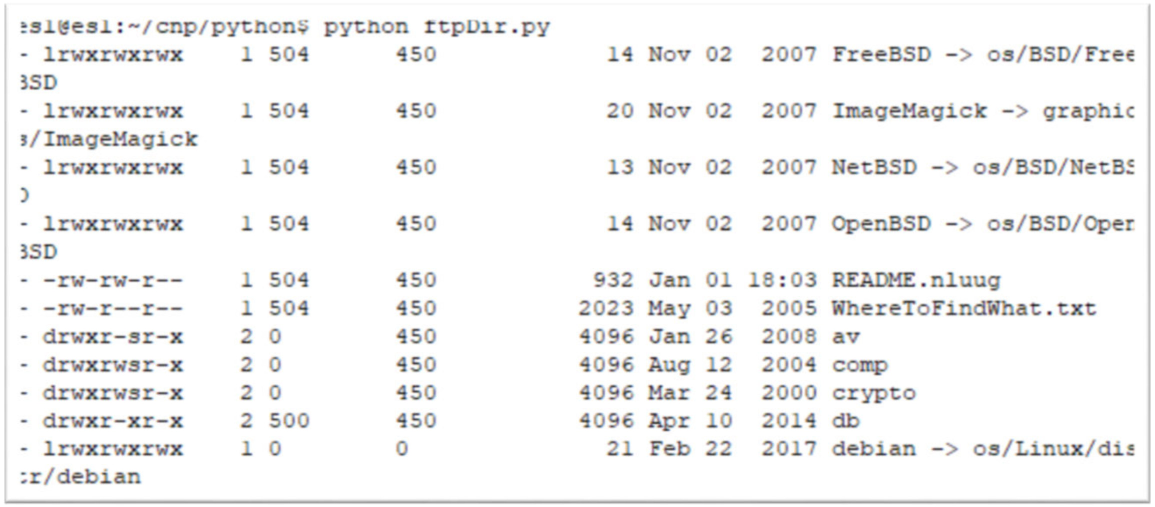
Changing the Directory (ftpDir.py)

- The below program uses the cwd method available in the ftplib module to change the directory and then fetch the required content.

```
import ftplib
ftp = ftplib.FTP("ftp.nluug.nl")
ftp.login("anonymous", "ftplib-example-1")
data = []
ftp.cwd('/pub/')        change directory to /pub/
ftp.dir(data.append)
ftp.quit()
for line in data:
    print "-", line
```

Changing the Directory (ftpDir.py)

- When we run the above program, we get the following output –



```
esl@esl:~/cnp/python$ python ftpDir.py
- lrwxrwxrwx    1 504      450           14 Nov 02  2007 FreeBSD -> os/BSD/Free
BSD
- lrwxrwxrwx    1 504      450           20 Nov 02  2007 ImageMagick -> graphic
s/ImageMagick
- lrwxrwxrwx    1 504      450           13 Nov 02  2007 NetBSD -> os/BSD/NetBS
D
- lrwxrwxrwx    1 504      450           14 Nov 02  2007 OpenBSD -> os/BSD/Open
BSD
- -rw-rw-r--    1 504      450          932 Jan 01 18:03 README.nluug
- -rw-r--r--    1 504      450         2023 May 03  2005 WhereToFindWhat.txt
- drwxr-sr-x    2 0        450         4096 Jan 26  2008 av
- drwxrwsr-x    2 0        450         4096 Aug 12  2004 comp
- drwxrwsr-x    2 0        450         4096 Mar 24  2000 crypto
- drwxr-xr-x    2 500      450         4096 Apr 10  2014 db
- lrwxrwxrwx    1 0        0             21 Feb 22  2017 debian -> os/Linux/dis
tr/debian
```

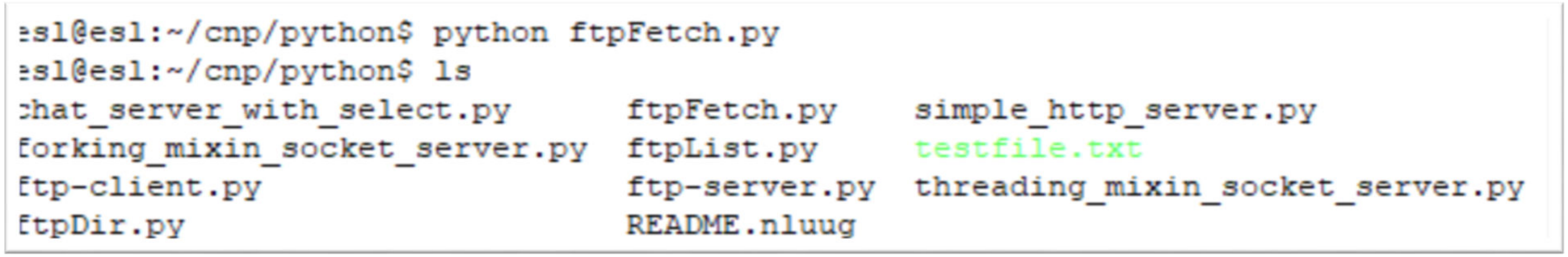
Fetching the Files (ftpFetch.py)

- After getting the list of files as shown above, we can fetch a specific file by using the **getfile** method. This method moves a copy of the file from the remote system to the local system from where the ftp connection was initiated.

```
import ftplib
import sys
def getFile(ftp, filename):
    try:
        ftp.retrbinary("RETR " + filename ,open(filename, 'wb').write)
    except:
        print "Error"
ftp = ftplib.FTP("ftp.nluug.nl")
ftp.login("anonymous", "ftplib-example-1")
ftp.cwd('/pub/')         change directory to /pub/
getFile(ftp,'README.nluug')
ftp.quit()
```

Fetching the Files (ftpFetch.py)

- After getting the list of files as shown above, we can fetch a specific file by using the **getfile** method. This method moves a copy of the file from the remote system to the local system from where the ftp connection was initiated.



```
esl@esl:~/cnp/python$ python ftpFetch.py
esl@esl:~/cnp/python$ ls
chat_server_with_select.py      ftpFetch.py     simple_http_server.py
forking_mixin_socket_server.py  ftpList.py      testfile.txt
ftp-client.py                   ftp-server.py   threading_mixin_socket_server.py
ftpDir.py                       README.nluug
```

- When we run the above program, we find the file README.nlug to be present in the local system from where the connection was initiated.

File Downloader

- We can download data from a serer using python's module which handle ftp or File Transfer Protocol. We can also read the data and later save it to the local system.

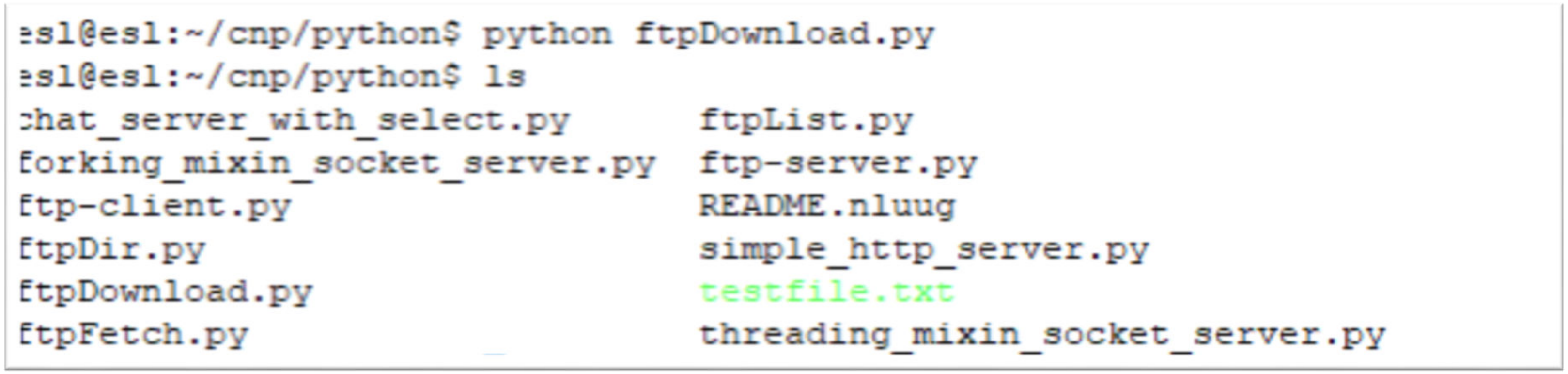
Fetching the Files (ftpDownload.py)

- As FTP File download, we can fetch a specific file by using the **getfile** method. This method moves a copy of the file from the remote system to the local system from where the ftp connection was initiated.

```
import ftplib
import sys
def getFile(ftp, filename):
    try:
        ftp.retrbinary("RETR " + filename ,open(filename, 'wb').write)
    except:
        print "Error"
ftp = ftplib.FTP("ftp.nluug.nl")
ftp.login("anonymous", "ftplib-example-1")
ftp.cwd('/pub/')
getFile(ftp,'README.nluug')
ftp.quit()
```

Fetching the Files (ftpDownload.py)

- When we run the above program, we find the file README.nlug to be present in the local system from where the connection was initiated.



```
esl@esl:~/cnp/python$ python ftpDownload.py
esl@esl:~/cnp/python$ ls
chat_server_with_select.py      ftpList.py
forking_mixin_socket_server.py  ftp-server.py
ftp-client.py                   README.nluug
ftpDir.py                       simple_http_server.py
ftpDownload.py                  testfile.txt
ftpFetch.py                     threading_mixin_socket_server.py
```

- When we run the above program, we find the file README.nlug to be present in the local system from where the connection was initiated.

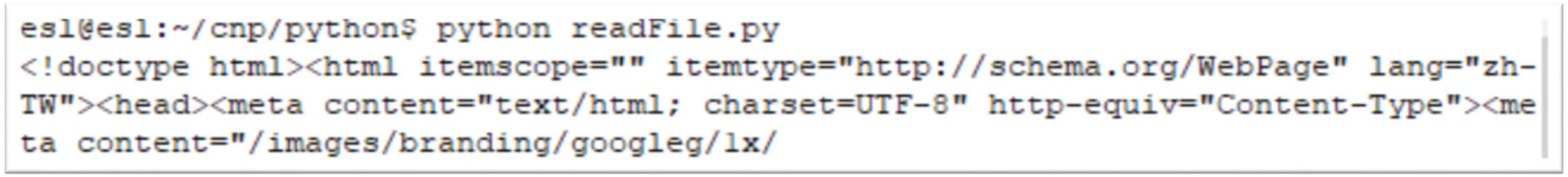
Reading the Data (readFile.py)

- In the below example we use the module urllib2 to read the required portion of the data which we can copy and save it to local system.
- When we run the above program, we get the following output −

```
import urllib2
response = urllib2.urlopen('http://www.google.com')
html = response.read(200)
print html
```

Reading the Data (readFile.py)

- When we run the above program, we get the following output –



```
esl@esl:~/cnp/python$ python readFile.py
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="zh-
TW"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><me
ta content="/images/branding/googleg/1x/
```

Python NTP Client

Python NTP Client (NTP.py)

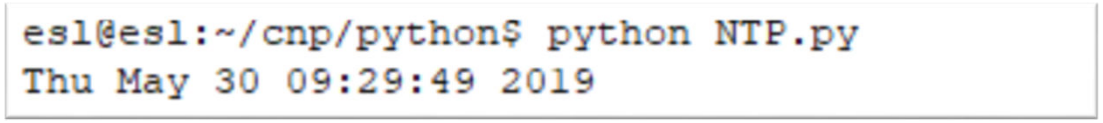
- NTP (Network Time Protocol), a means of synchronizing clocks over the internet. After you configure the NTP feature, the PC can get time from an NTP server automatically.
- Following example is try to query an NTP server (pool.ntp.org) to get the current time from a python script.
- There are some python modules you can install which will take care of the heavy lifting (one is ntplib), but this example wanted to be able to do these NTP queries without requiring the host to install 3rd party python packages.

Python NTP Client (NTP.py)

```
from socket import AF_INET, SOCK_DGRAM
import sys
import socket
import struct, time
def getNTPTime(host = "pool.ntp.org"):
    port = 123
    buf = 1024
    address = (host,port)
    msg = '\x1b' + 47 * '\0'
    # reference time (in seconds since 1900-01-01 00:00:00)
    TIME1970 = 2208988800L # 1970-01-01 00:00:00
    # connect to server
    client = socket.socket( AF_INET, SOCK_DGRAM)
    client.sendto(msg, address)
    msg, address = client.recvfrom( buf )
    t = struct.unpack( "!12I", msg )[10]
    t -= TIME1970
    return time.ctime(t).replace("  "," ")
if __name__ == "__main__":
    print getNTPTime()
```

Python NTP Client (NTP.py)

- When we run the above program, we get the following output –



```
esl@esl:~/cnp/python$ python NTP.py
Thu May 30 09:29:49 2019
```

Packet Sniffer in Python

Packet Sniffer in Python (SimpleSniffer.py)

- For sniffing with socket module in python we have to create a socket.socket class object with special configuration.
- In simple words, we have to configure socket.socket class object to capture low-level packets from the network so that it can capture packet from low-level networks and provides us output without doing any type of changes in capture packets.

Packet Sniffer in Python (SimpleSniffer.py)

- If You Are Using Windows Then, Use This Codes,
  - s = socket.socket(socket.AF_INET,socket.SOCK_RAW,socket.IPPROTO_IP)
  - s.bind(("YOUR_INTERFACE_IP",0))
  - s.setsockopt(socket.IPPROTO_IP,socket.IP_HDRINCL,1)
  - s.ioctl(socket.SIO_RCVALL,socket.RCVALL_ON)
- or if you are using *nix then, use this code.
  - s=socket.socket(socket.PF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0800))

Packet Sniffer in Python (SimpleSniffer.py)

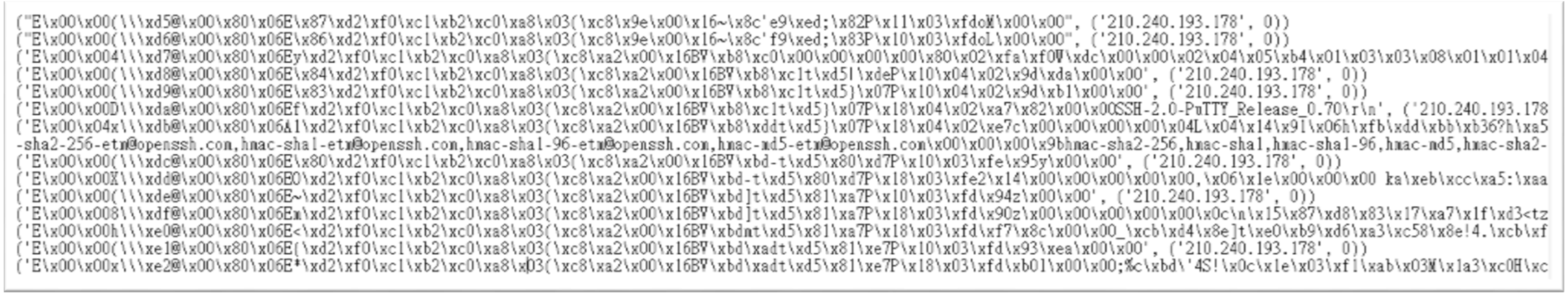
- Now our socket object is ready to capture packets. Let's create a simplest packet sniffer script in python.

```
1.    #!/usr/bin/python
2.    import socket
3.    #create an INET, raw socket
4.    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
5.    # receive a packet
6.    while True:
7.        # print output on terminal
8.        print s.recvfrom(65565)
```

- Line 2 is for importing modules
- Line 4 for creating a socket.socket class object.
- Line 6 for while loop
- Line 8 for printing output on the terminal.

Packet Sniffer in Python (SimpleSniffer.py)

- To Run this example, just run it with sudo permission.
- When we run the above program, we get the following output −



```
("E\x00\x00(\\\xd5@\x00\x80\x06E\x87\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\x9e\x00\x16~\x8c'e9\xed;\x82P\x11\x03\xfdoM\x00\x00", ('210.240.193.178', 0))
("E\x00\x00(\\\xd6@\x00\x80\x06E\x86\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\x9e\x00\x16~\x8c'f9\xed;\x83P\x10\x03\xfdoL\x00\x00", ('210.240.193.178', 0))
('E\x00\x004\\\xd7@\x00\x80\x06Ey\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xb8\xc0\x00\x00\x00\x00\x80\x02\xfa\xf0V\xdc\x00\x00\x02\x04\x05\xb4\x01\x03\x03\x08\x01\x01\x04
('E\x00\x00(\\\xd8@\x00\x80\x06E\x84\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xb8\xc1t\xd5l\xdeP\x10\x04\x02\x9d\xda\x00\x00', ('210.240.193.178', 0))
('E\x00\x00(\\\xd9@\x00\x80\x06E\x83\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xb8\xc1t\xd5}\x07P\x10\x04\x02\x9d\xb1\x00\x00', ('210.240.193.178', 0))
('E\x00\x00D\\\xda@\x00\x80\x06Ef\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xb8\xc1t\xd5}\x07P\x18\x04\x02\xa7\x82\x00\x00SSH-2.0-PuTTY_Release_0.70\r\n', ('210.240.193.178
('E\x00\x04x\\\xdb@\x00\x80\x06A1\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xb8\xddt\xd5}\x07P\x18\x04\x02\xe7c\x00\x00\x00\x00\x04L\x04\x14\x91\x06h\xfb\xdd\xbb\xb36?h\xa5
-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1-96-etm@openssh.com,hmac-md5-etm@openssh.com\x00\x00\x00\x9bhmac-sha2-256,hmac-sha1,hmac-sha1-96,hmac-md5,hmac-sha2-
('E\x00\x00(\\\xdc@\x00\x80\x06E\x80\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xbd-t\xd5\x80\xd7P\x10\x03\xfe\x95y\x00\x00', ('210.240.193.178', 0))
('E\x00\x00X\\\xdd@\x00\x80\x06E0\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xbd-t\xd5\x80\xd7P\x18\x03\xfe2\x14\x00\x00\x00\x00\x00,\x06\x1e\x00\x00\x00 ka\xeb\xcc\xa5:\xaa
('E\x00\x00(\\\xde@\x00\x80\x06E~\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xbd]t\xd5\x81\xa7P\x10\x03\xfd\x94z\x00\x00', ('210.240.193.178', 0))
('E\x00\x008\\\xdf@\x00\x80\x06Em\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xbd]t\xd5\x81\xa7P\x18\x03\xfd\x90z\x00\x00\x00\x00\x00\x0c\n\x15\x87\xd8\x83\x17\xa7\x1f\xd3<tz
('E\x00\x00h\\\xe0@\x00\x80\x06E<\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xbdmt\xd5\x81\xa7P\x18\x03\xfd\xf7\x8c\x00\x00_\xcb\xd4\x8e]t\xe0\xb9\xd6\xa3\xc58\x8e!4.\xcb\xf
('E\x00\x00(\\\xe1@\x00\x80\x06E(\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xbd\xadt\xd5\x81\xe7P\x10\x03\xfd\x93\xea\x00\x00', ('210.240.193.178', 0))
('E\x00\x00x\\\xe2@\x00\x80\x06E*\xd2\xf0\xc1\xb2\xc0\xa8\x03(\xc8\xa2\x00\x16BV\xbd\xadt\xd5\x81\xe7P\x18\x03\xfd\xb01\x00\x00;%c\xbd\'4S!\x0c\x1e\x03\xf1\xab\x03M\x1a3\xc0H\xc
```