ENPM685 – Python Exercises

Version 3.2 – January 4th 2022

Grab The Code Examples

1. Create a directory to work out of

2. **`git clone https://github.com/kts262/enpm685.git`**

- OR -

2. **`wget https://github.com/kts262/enpm685/archive/master.zip`**
   `unzip master.zip`

Python Background

We'll be using Python 3 for these examples.

Where to get help:

- https://www.google.com/
- https://stackoverflow.com/

Books/online knowledge:

- Automated The Boring Stuff With Python by Al Sweigart (http://automatetheboringstuff.com/ and in print too)
- A Whirlwind Tour of Python - https://github.com/jakevdp/WhirlwindTourOfPython
- Coding for Penetration Testers by Jason Andress and Ryan Linn
- http://learnpythonthehardway.org/
- http://www.codecademy.com/en/tracks/python

Start off your script:

`#!/usr/bin/python`

(or replace with the location of where Python is installed). Once you have done this you can make your script executable with "**chmod +x scriptname.py**" and run it with "**./scriptname.py**" or you can run it with "**python scriptname.py**"

Comments:

You can use the "**#**" and anything after that on the line is considered a comment and will be ignored

```
#!/usr/bin/python
#
# myscript.py – my cool script

mystring = "Professor Shivers is a bad Python teacher." # Truth!
```

Whitespace

White space matters in Python. A tab (8 spaces is the default for vi) is different from pressing the space bar 8 times! 4 spaces is different than 5 spaces. **Watch your white space!**

Import Modules

```
import module_name
from module_name import function
```

The first imports everything and you call with module_name.function() the other you can call with just function(). Typically, not a big issue but you can potentially run into name space issues in larger scripts.

Ex:

```
import sys
value = sys.argv[]

from sys import argv
value = argv[]
```

Variables

Variables must be declared before they are used.

```
var = 0              # integer
var2 = "string"      # string (text)
var3 = ["1", "2"]    # list
```

Some variables (like integers, strings, and floating numbers) can be converted into other variables types with their corresponding functions.

Ex: `var_as_string = str(var)`
Ex: `string = "685"`
`string_as_int = int(string)`

Comparisons:

Operation	Meaning
`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Equal to
`!=`	Not equal to
`is`	Object identity (string is "685")
`is not`	Negated object identity (string is not "686")

Lists vs Tuples vs Dicts

Lists – list of values, each numbered from 0 with a value
Ex: `people = ['Bob','Sue','Beth']`

Tuples – lists but you can't change their values
Ex: `months = ('Jan','Feb',…'Dec')`

Dictionary – "index" of words and with each one a definition
Ex: `phonebook – {'Bob':1112222, 'Jenny':8675309, 'Joe': 1234567}`

Output Text

```
print("text to output")
print("text " + var + "\n")
```

Input Text

```
input = raw_input()

input = raw_input("Enter something: ")
```

File Input/Output

```
var = open(filename)
var2 = open("filename", "r")

r = read, w = write, a = append, r+ = read+write
b = binary mode, used on Windows

var.read() = reads file character by character
var.readlines() = read file line by line

var.write("text") = write to file
var.close() = close file
```

Conditionals

```
if something == something_else:
      #do something
elif something < something_else:
      #do something else
else:
      #do this other thing
```

Loops

```
for line in lines:
      #do something with/to each line

temp = 220
while temp > 212:
      #do something
      temp – temp -1
```

Functions

Create a function:

```
def function_name():
      # do stuff

def function_name2(input_var):
      #do stuff with input_var

def function_name3(var):
      return var*var
```

Call a function:

```
# run function_name()
function_name()

# run function_name2()
function_name(variable)

#run function_name3()
result = function_name(variable)
```

wget Example

Code:

```
#!/usr/bin/python3

import urllib.request
import sys

# sys.argv[1] = url
# sys.argv[2] = file name

response = urllib.request.urlretrieve(sys.argv[1], sys.argv[2])
```

---

httpbanner.py Example

Pseudocode:

1. Connect to server on port 80/443
2. Get a request for the main page (HEAD request to save time)
3. Scan the response for the “Server” tag.
4. Server = HTTP Banner

Code:

```
#!/usr/bin/python3
import sys
import requests

httpsvr = sys.argv[1]
response = requests.get(sys.argv[1])
```

```
print(sys.argv[1] + ' web server is: ' + response.headers['Server'])
```

Results:

```
[oitsec:ENPM685 kts$ python httpbanner.py www.umd.edu
 www.umd.edu HTTP server is: 'CloudFront'
[oitsec:ENPM685 kts$ python httpbanner.py www.google.com
 www.google.com HTTP server is: 'gws'
[oitsec:ENPM685 kts$ python httpbanner.py advancedengineering.umd.edu
 advancedengineering.umd.edu HTTP server is: 'Apache/2.2'
```

Wordpress Scanning Example

Idea: Write a script that checks to see if a site might be a Wordpress site.

Pseudocode:

1. Look for **/readme.txt**
2. Look for **/wp-admin/admin-ajax.php**
3. Look for and output **robots.txt** (might turn up other Wordpress links/interesting info)

We'll use the Python requests library for this.

Example:

```
import requests

url = "http://www.umd.edu"
r = requests.get(url)
if r.status_code == 200:
      print("The URL was found.")
```

V1 code (check for **readme.html**)

```
#!/usr/bin/python

import sys
import requests


def check_url(url):
    r = requests.get(url + "/readme.html")
    if r.status_code == 200:
        print(url + " - FOUND -- POSSIBLE WORDPRESS SITE")
    else:
```

```
        print(url + " - NOT found")

httpsvr = sys.argv[1]

print("Checking " + httpsvr + "...")
check_url(httpsvr)
```

Now let's look for **/wp-admin/admin-ajax.php**

We could copy and paste the code from before and change it but what if we make the code more functional so we can request different URLs with it…

Change the adding to the URL into the main body

```
check_url(httpsvr)
```

Becomes…

```
check_url(httpsvr + "/readme.html")
check_url(httpsvr + "/wp-admin/admin-ajax.php")
```

Let's also look for robots.txt

If we find it let's output it. Might offer more clues of Wordpress URLs or other interesting URLs…

Ex.

```
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
```

(We could modify check_url to see if robots is in the URL or add a second variable to the function but I'm taking the easy route of duplicating check_url but changing what it does. Plus I've always wanted to have a function called "print_robots")

check_url()

```
def check_url(url):
  r = requests.get(url)
  if r.status_code == 200:
    print(url + " - FOUND -- POSSIBLE WORDPRESS SITE")
  else:
```

```
    print(url + " - NOT found")
```

print_robots()

```
def print_robots(url):
  r = requests.get(url)
  if r.status_code == 200:
    print ("\n" + url + " - FOUND -- robots.txt content:\n")
    print(r.text)
  else:
    print(url + " - NOT found")
```

Final Version Code:

```
#!/usr/bin/python3

import sys
import requests


def check_url(url):
  r = requests.get(url)
  if r.status_code == 200:
    print(url + " - FOUND -- POSSIBLE WORDPRESS SITE")
  else:
    print(url + " - NOT found")

def print_robots(url):
  r = requests.get(url)
  if r.status_code == 200:
    print ("\n" + url + " - FOUND -- robots.txt content:\n")
    print(r.text)
  else:
    print(url + " - NOT found")

httpsvr = sys.argv[1]

print("Checking " + httpsvr + "...")
check_url(httpsvr + "/readme.html")
check_url(httpsvr + "/wp-admin/admin-ajax.php")
print_robots(httpsvr + "/robots.txt")
```

Tor Exit Nodes

You'd like to get a list of Tor Exit nodes and format them so you can include them into various firewalling/security tools.

Tor makes an exit node list available here: https://check.torproject.org/cgi-bin/TorBulkExitList.py

You can query it like so: https://check.torproject.org/cgi-bin/TorBulkExitList.py?ip=1.1.1.1&port=80

Initial code:

```
---------------
#!/usr/bin/python3

import requests

exit_list_url = requests.get('http://check.torproject.org/cgi-
bin/TorBulkExitList.py?ip=1.1.1.1&port=80')
exit_list = exit_list_url.text
exit_list = exit_list.split("\n")
exit_list.pop()  # delete that last blank line

for line in exit_list:
        if line[0] != "#":
                print(line + ",")
---------------
```

To make it TippingPoint/CSV friendly add **+ ","** to that last line.

Ex: **`print line + ","`**

Save it to a pre-determined file:

```
---------------
#!/usr/bin/python3

import requests

f = open("tor_exit_nodes.txt", "w")

exit_list_url = requests.get('http://check.torproject.org/cgi-
bin/TorBulkExitList.py?ip=1.1.1.1&port=80')

exit_list = exit_list_url.text
exit_list = exit_list.split("\n")
```

```
exit_list.pop()  # delete that last blank line

i = 0
print("Pulling Tor exit node list...")

for line in exit_list:
        if line[0] != "#":
                f.write(line + ",\n")
                i = i + 1

print("Done, " + str(i) + " exit nodes")
```

Command Line Options

```
import sys
```

sys.argv[x] where x =
0 = script name (or 'python script' if invoked with python)
1 = first argument
2 = second argument
N = nth argument

User specified file name code:

```
#!/usr/bin/python3

import requests
import sys

if len(sys.argv) < 2:
    print("You need to specify a file name")
    sys.exit(0)

f = open(sys.argv[1], "w")

exit_list_url = requests.get('http://check.torproject.org/cgi-
bin/TorBulkExitList.py?ip=1.1.1.1&port=80')

exit_list = exit_list_url.text
exit_list = exit_list.split("\n")
exit_list.pop()  # delete that last blank line

i = 0
print("Pulling Tor exit node list...")
```

```
for line in exit_list:
        if line[0] != "#":
                f.write(line + ",\n")
                i = i + 1

print("Done, " + str(i) + " exit nodes")
```

Getting fancier with OptParse

More info: https://docs.python.org/3/library/optparse.html

```
#!/usr/bin/python3

import requests
from optparse import OptionParser

parser = OptionParser(usage="Usage %prog [-f format] file")
parser.add_option("-f", dest="format", help="output format to use [csv
,palo, iptables]")

options, args = parser.parse_args()

if len(args) < 1:
    parser.error("You need to specify a format and a file name")
    sys.exit(0)

f = open(args[0], "w")

exit_list_url = requests.get('http://check.torproject.org/cgi-
bin/TorBulkExitList.py?ip=1.1.1.1&port=80')

exit_list = exit_list_url.text
exit_list = exit_list.split("\n")
exit_list.pop()  # delete that last blank line

i = 0
print("Pulling Tor exit node list...")
print("Output option selected: " + options.format)

for line in exit_list:
        if line[0] != "#":
                if options.format == "csv":
```

```
                    f.write(line + ",\n")
                i = i + 1

print("Done, " + str(i) + " exit nodes")
```

Adding another option… CIDR/Palo Alto format

Palo Alto needs IP addresses/subnets to have the CIDR block with them.

1.1.1.1/32 = 1.1.1.1

Code:

```
for line in exit_list:
        if line[0] != "#":
                if options.format == "csv":
                        line_output = line + ",\n"
                elif options.format == "palo":
                        line_output = line + "/32\n"
                f.write(line_output)
                i = i+1
```

Multiple options:

```
elif (options.format == "palo") or (options.format == "cidr"):
```

iptables Format

Format: **iptables -A INPUT -s ip.address -j DROP**

```
elif options.format == "iptables":
     line_output = "sudo iptables -A INPUT -s " + line + "-j DROP\n"
```

Code:

```
#!/usr/bin/python3

import requests
from optparse import OptionParser

parser = OptionParser(usage="Usage %prog [-f format] file")
```

```
parser.add_option("-f", dest="format", help="output format to use [csv
,palo, iptables]")

options, args = parser.parse_args()

if len(args) < 1:
    parser.error("You need to specify a format and a file name")
    sys.exit(0)

f = open(args[0], "w")

exit_list_url = requests.get('http://check.torproject.org/cgi-
bin/TorBulkExitList.py?ip=1.1.1.1&port=80')

exit_list = exit_list_url.text
exit_list = exit_list.split("\n")
exit_list.pop()  # delete that last blank line

i = 0
print("Pulling Tor exit node list...")
print("Output option selected: " + options.format)

for line in exit_list:
        if line[0] != "#":
                if options.format == "csv":
                    line_output = line + ",\n"
                elif (options.format == "palo") or (options.format ==
"cidr"):
                    line_output = line + "/32\n"
                elif options.format == "iptables":
                    line_output = "sudo iptables -A INPUT -s " + line
+ " -j DROP\n"
                f.write(line_output)
                i = i + 1

print("Done, " + str(i) + " exit nodes")
```

Exercise: add support for **ufw** output.

Format: **sudo ufw deny from *ip.address***