Python for Network Engineers

Onsite Training Session

$ whoami

Kirk Byers
Network Engineer:
CCIE #6243 (emeritus)

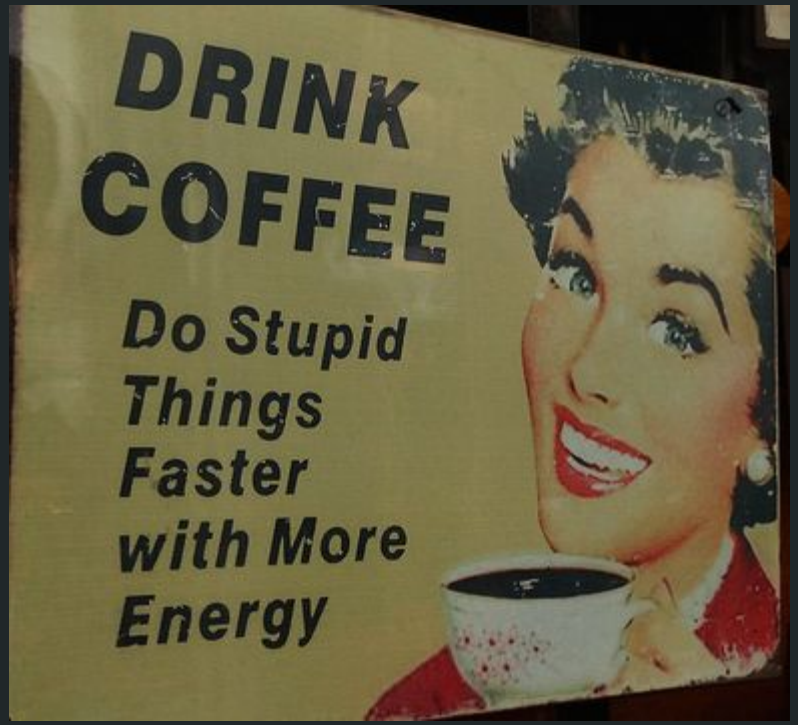
Programmer:
Netmiko
NAPALM

Teach Python and Ansible
SF Network Automation Meetup

General:

Lunch
Some breaks

Focused
Minimize Distractions
Exercises and Examples
Examples in the Python Shell
Try not to fall behind on day1 & 2



Flickr: Ben Sutherland

Day1 Schedule

1. Intro
2. GIT
3. Python Fundamentals - General
4. Strings
5. Numbers
6. Files
7. Lists / Tuples
7. Booleans / None
8. Conditionals
9. Loops
10. Dictionaries

Git

- Why care about Git?
- Git and GitHub
- Cloning a Project
- git init / git add / git rm / git commit
- git pull / git push
- Managing Git branches
- Making a Pull Request
- Git Rebase

Why Python?

- Widely supported (meaning lots of library support)
- Easily available on systems
- Language accommodates beginners through advanced
- Maintainable
- Allows for easy code reuse
- High-level

Python Characteristics

Indentation matters.

Use spaces not tabs.

Python programmers are particular.

Py2 or Py3.

General Items

The Python interpreter shell
Assignment and variable names
Python naming conventions
Printing to standard out/reading from standard in
Creating/executing a script
Quotes, double quotes, triple quotes
Comments
dir() and help()

Strings

- String methods
- Chaining
- split()
- strip()
- substr in string
- unicode
- raw strings
- format() method

Numbers

Integers
Floats
Math Operators (+, -, *, /, **, %)
Strange Behavior of Integer Division

Writing to a file/reading from a file:

```
with open(file_name, "w") as f:
    f.write(output)
```

```
with open(file_name) as f:
    output = f.read()
```

Lists

Zero-based indices

.append()

.pop()

.join()

List slices

Tuple

Copying a list

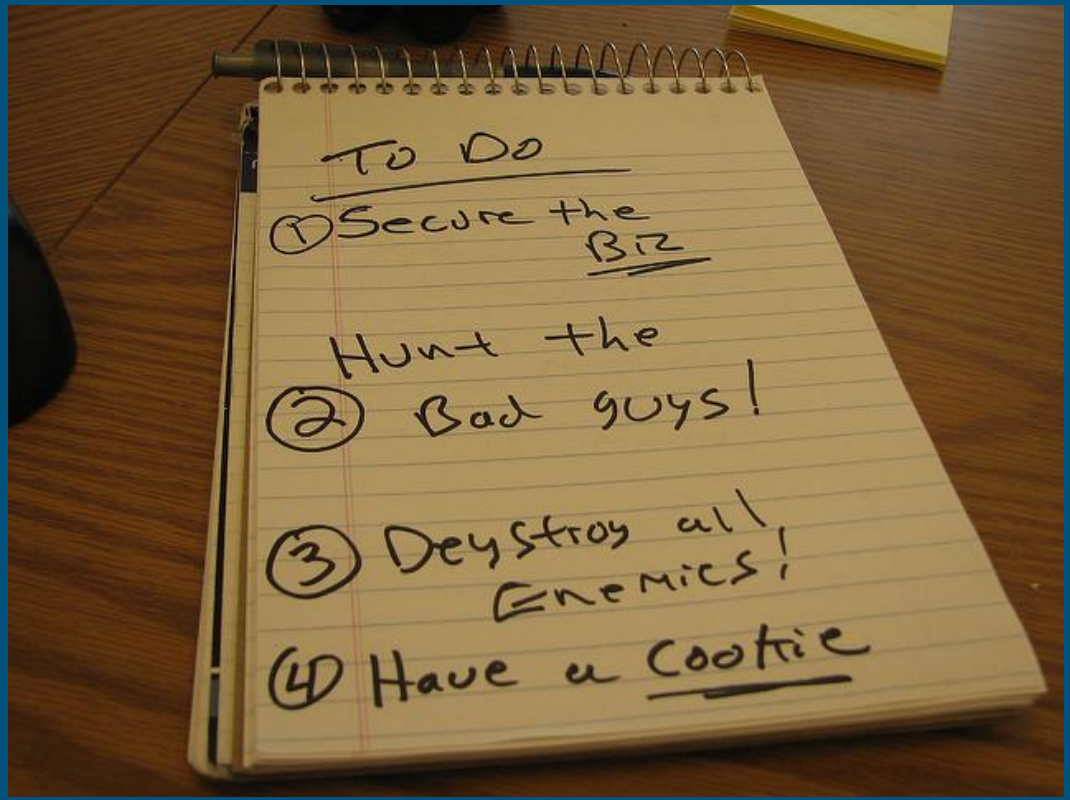


Photo: Purple Slog (Flickr)

Booleans and None

Boolean operators (and, or, not)

is

Truish

Comparison operators (==, !=, <, >, >=, <=)

None

Conditionals

```
if a == 15:
    print "Hello"
elif a >= 7:
    print "Something"
else:
    print "Nada"
```

Loops

- for
- while
- break
- continue
- range(len())
- enumerate

Photo: Mário Monte Filho (Flickr)

For/while syntax

```
for my_var in iterable:
    print my_var


i = 0
while i < 10:
    print i
    i += 1
```

Dictionaries

- Creating
- Updating
- get()
- pop()
- Iterating over keys
- Iterating over keys and values

Photo: Holger Zscheyge (Flickr)

Day2

1. Review Exercises
2. Exceptions
3. Regular Expressions
4. Functions
5. Python Code Structure
6. Classes and Objects
7. Managing Python Libraries
8. Modules and Packages
9. Writing reusable code
10. Virtualenv

Flickr: au_tiger01

Exception Handling

```
try:
    my_dict[‘missing_key’]
except KeyError:
    do_something
```

- Trying to gracefully handle errors.
- finally: - always ran if you have a cleanup condition.

Python Regular Expresions

import re

Other re methods

re.split()

re.sub()

re.findall()

re.search(pattern, string)

- always use raw strings
- re.M/re.MULTILINE
- re.DOTALL
- re.I
- Parenthesis to retain patterns
- greedy/not greedy (.*?)

Functions:

- Defining a function
- Positional arguments
- Named arguments
- Mixing positional and named arguments
- Default values
- Passing in *args, **kwargs
- Functions and promoting the reuse of code

Python Code Structure:

- Imports at top of the file
- CONSTANTS
- Functions / classes
- if __name__ == “__main__”:
- Main code or main() function call

Classes and Objects

```
class NetDevice(object):
    def __init__(self, ip_addr, username, password):
        self.ip_addr = ip_addr
        self.username = username
        self.password = password

    def test_method(self):
        print "Device IP is: {}".format(self.ip_addr)
        print "Username is: {}".format(self.username)


    rtr1 = NetDevice('10.22.1.1', 'admin', 'passw')
    rtr1.test_method()
```

Libraries

sys.path

PYTHONPATH

Installing packages (pip)

import x

from x import y

Photo: Viva Vivanista (Flickr)

Modules and Packages

Python Module

A Python file that you can import into another Python program

Example, storing device's definitions in an external file.

Python Package

An importable Python directory

__init__.py

Writing reusable code

Basic Building Blocks (functions/classes)
Python Modules
if __name__
Python Packages
Don’t repeat yourself

Flickr: Koka Sexton

Virtualenv

virtualenv -p /usr/bin/python27 test_venv

source /path/to/virtualenv/bin/activate

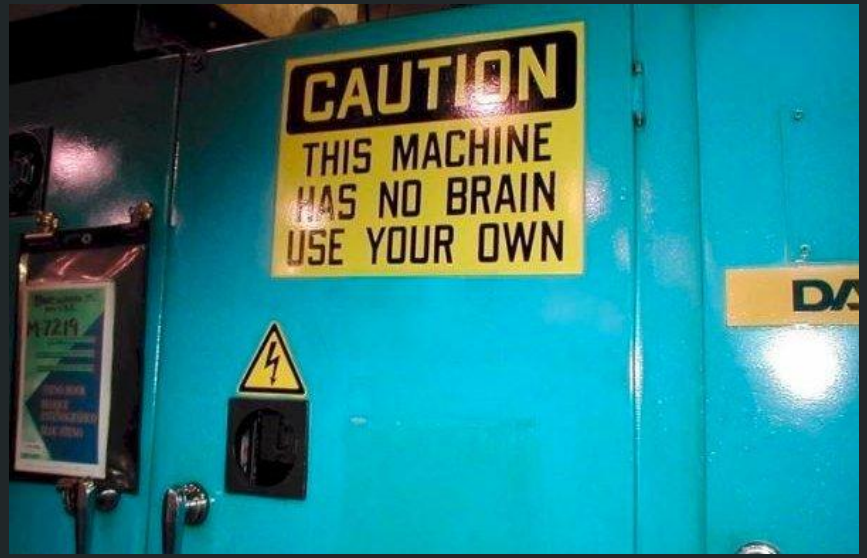
deactivate

pip list

pip install netmiko==1.4.1

Day3 Schedule

1. Serialization: JSON and YAML
2. Python and SSH
3. Concurrency Basics
    - Threads
    - Processes
4. Juniper, NETCONF, and PyEZ
    - What is NETCONF
    - PyEZ
    - PyEZ get operations
5. Jinja2 Templating



Flickr: Pierre-Olivier Carles

Data Serialization

Why do we need data serialization?

Characteristics of JSON

Characteristics of YAML

Paramiko & Netmiko

Paramiko is the standard Python SSH library.

Netmiko is a multi-vendor networking library based on Paramiko.

Netmiko example

```
#!/usr/bin/env python
from getpass import getpass
from netmiko import ConnectHandler

if __name__ == "__main__":
    password = getpass("Enter password: ")
    srx = {
        'device_type': 'juniper_junos',
        'ip':   '184.105.247.76',
        'username': 'pyclass',
        'password': password
    }

    net_connect = ConnectHandler(**srx)
    print net_connect.find_prompt()
```

Key Netmiko Methods

.send_command()
.send_command_timing()

.send_config_set()
.send_config_from_file()

.commit()
.enable()
.disconnect()

.write_channel()
.read_channel()

FileTransfer Class

Netmiko Vendors

Regularly tested
Arista vEOS
Cisco ASA
Cisco IOS
Cisco IOS-XE
Cisco IOS-XR
Cisco SG300
HP Comware7
HP ProCurve
Juniper Junos
Linux

Limited testing
Avaya ERS
Avaya VSP
Brocade VDX
Brocade ICX/FastIron
Brocade MLX/NetIron
Cisco NX-OS
Cisco WLC
Dell-Force10 DNOS9
Huawei
Palo Alto PAN-OS
Vyatta VyOS

Experimental
A10
Alcatel-Lucent SR-OS
Ciena SAOS
Cisco Telepresence
Dell PowerConnect
Enterasys
Extreme
F5 LTM
Fortinet

Threads/Processes

- Concurrency
- Python and the GIL
- Example with threads
- Example with processes
- Example with a queue

Juniper, NETCONF, and PyEZ

- What is NETCONF?
- PyEZ
- PyEZ get operations
- PyEZ config operations

PyEZ simple connect / facts

```
from jnpr.junos import Device
from getpass import getpass
from pprint import pprint

juniper_srx = {
        "host": “184.105.247.76”,
        "user": "pyclass",
        "password": getpass(),
    }

a_device = Device(**juniper_srx)
a_device.open()
pprint(a_device.facts)
```

PyEZ table operations

```
from jnpr.junos import Device
from jnpr.junos.op.ethport import EthPortTable
from getpass import getpass

juniper_srx = {
    "host": "184.105.247.76",
    "user": "pyclass",
    "password": getpass(),
}
a_device = Device(**juniper_srx)
a_device.open()
eth_ports = EthPortTable(a_device)
eth_ports.get()
```

Jinja2 Templating

```
import jinja2

bgp_template = """
protocols {
    bgp {
        group EBGP {
            type external;
            peer-as {{ remote_as }};
            neighbor {{ peer_ip }};
        }
    }
}
"""
```

```
my_vars = {
    'remote_as': '100',
    'peer_ip': '10.10.10.2',
}

template = jinja2.Template(bgp_template)
print template.render(my_vars)
```

Netmiko Tools

git clone https://github.com/ktbyers/netmiko_tools

In your .bashrc file if you want to retain it
export PATH=~/netmiko_tools/netmiko_tools:$PATH

~/.netmiko.yml

netmiko-grep
netmiko-show
netmiko-cfg

Day4 Schedule

1. PyEZ Config Operations.
2. Jinja2 Templating (expanded) / Pushing Configurations to Devices.
3. CiscoConfParse
4. Verifying and proceduralizing changes.
5. NAPALM & Juniper
6. Lab Environment Buildout
7. Misc items

PyEZ config operations

```
#!/usr/bin/env python
from jnpr.junos import Device
from jnpr.junos.utils.config import Config
from getpass import getpass

juniper_srx = {
    "host": "184.105.247.76",
    "user": "pyclass",
    "password": getpass(),
}
a_device = Device(**juniper_srx)
a_device.open()
cfg = Config(a_device)
```

```
cfg.load("set system host-name test1", format="set", merge=True)
cfg.load(path="load_hostname.conf", format="text", merge=True)
cfg.load(path="load_hostname.xml", format="xml", merge=True)

cfg.diff()
cfg.rollback(0)
cfg.commit()
```

CiscoConfParse

```
#!/usr/bin/env python
from ciscoconfparse import CiscoConfParse

cisco_file = 'cisco_config.txt'
cisco_cfg = CiscoConfParse(cisco_file)
intf_obj = cisco_cfg.find_objects(r"^interf")
print
for intf in intf_obj:
    print intf.text
    for child in intf.children:
        print child.text
    print
```

Verifying and Proceduralizing Changes

1. Use tooling to make your life easier.
    a. Lint checks
    b. Unit testing
    c. System testing
    d. CI tools
2. Think about the same principles for your configuration artifacts as for your code.
    a. Test network devices
    b. Test configuration build-outs from templates
    c. Test against virtual network devices / or lab devices

Verifying and Proceduralizing Changes

3. Think about your processes for doing a given task.
    a. Put your process into code.
    b. It doesn't have to be all-or-nothing; automate parts of the process.
    c. Across time automate more and more parts of a given process.
4. Do human beings really need to be involved in doing this.
5. Pick low risk, but time consuming tasks.
    a. Information gathering is much lower risk than config changes.
    b. Balance the risk / time saved / time required

NAPALM

Purpose of NAPALM: create a standard set of operations across a range of platforms.

Operations fall into two general categories: Config Operations + Getter Operations.

NAPALM Vendors

eos
junos
iosxr
nxos
ios
fortios
pluribus
panos
mikrotik
vyos

NAPALM Getters

get_facts
get_environment
get_snmp_information
get_ntp_peers
get_ntp_stats
get_mac_address_table
get_arp_table
get_interfaces
get_interfaces_ip
get_lldp_neighbors
get_lldp_neighbors_detail
get_bgp_neighbors
get_bgp_neighbors_detail
get_bgp_config
get_route_to
get_probes_config
get_probes_results
get_users
get_optics

NAPALM Config Operations

device.load_merge_candidate()
device.load_replace_candidate()

device.compare_config()
device.discard_config()

device.commit_config()

device.rollback()

Lab Environment Buildout and Misc Items

Python + Email

Python + SNMP

Reading CSV Data

Argparse

Subprocess

The end...

Questions?

ktbyers@twb-tech.com
Twitter: @kirkbyers