Python for Network Engineers

Onsite Training Session

$ whoami

Kirk Byers
Network Engineer:
CCIE #6243 (emeritus)

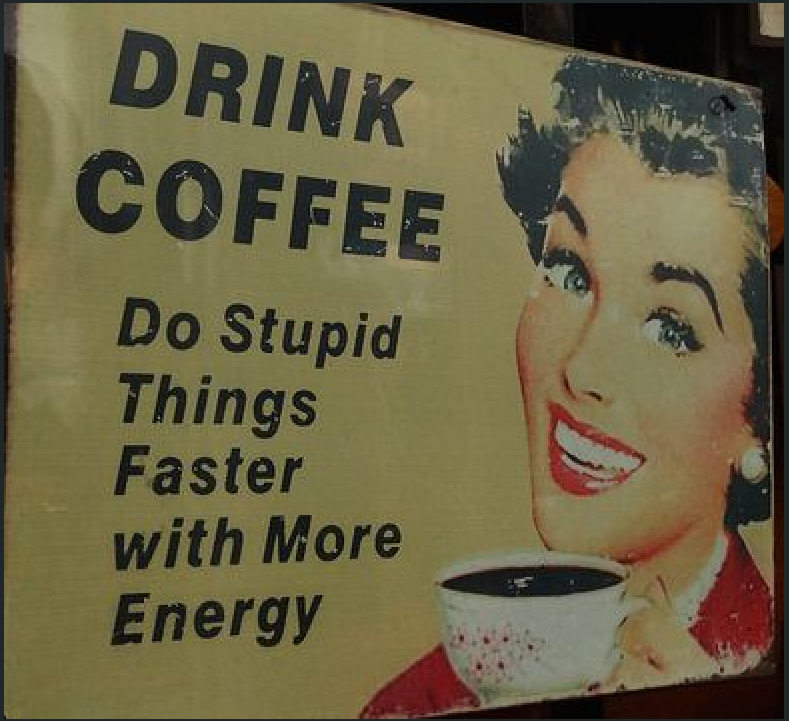
Programmer:
Netmiko
NAPALM

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
11. Exceptions
12. Regular Expressions

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

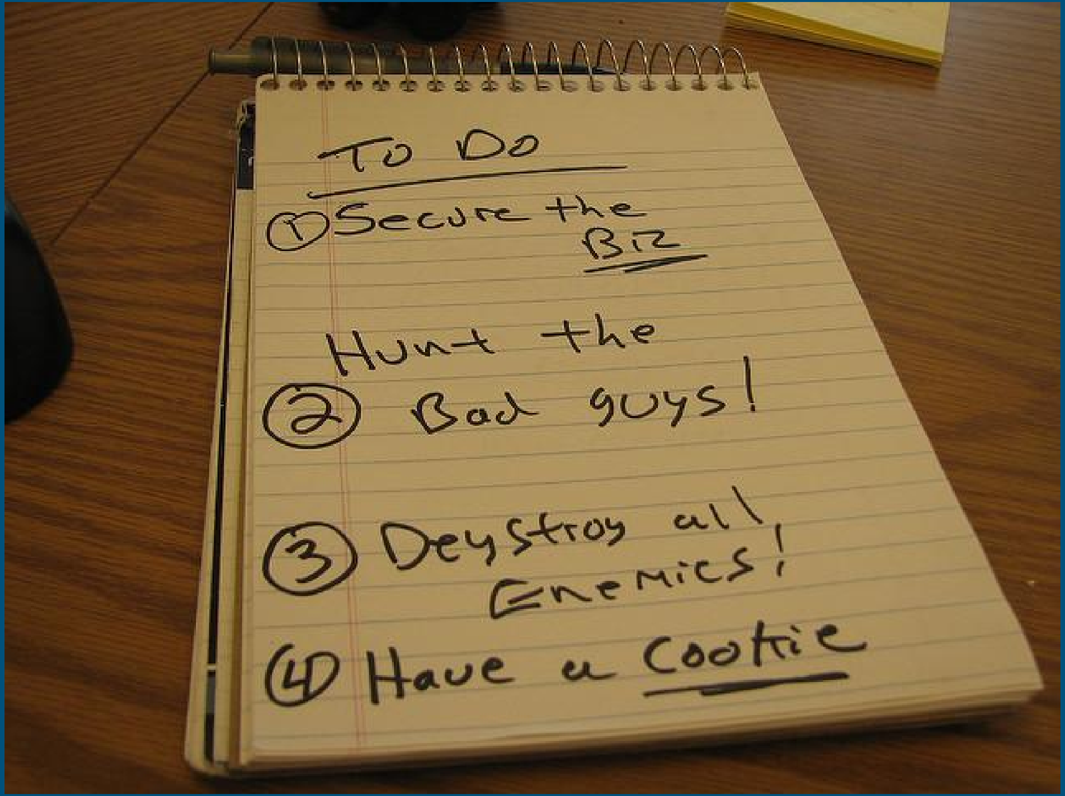


Photo: Purple Slog (Flickr)

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)

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 (.*?)

Day2

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

----

9. Python + SNMP
10. Sending Email Notifications
11. CiscoConfParse
12. Serialization: JSON and YAML

Family Church
Building The Church Through Building The Family
ANGER MGT CLASS POSTPONED FOR CLASSROOM REPAIRS

Flickr: au_tiger01

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

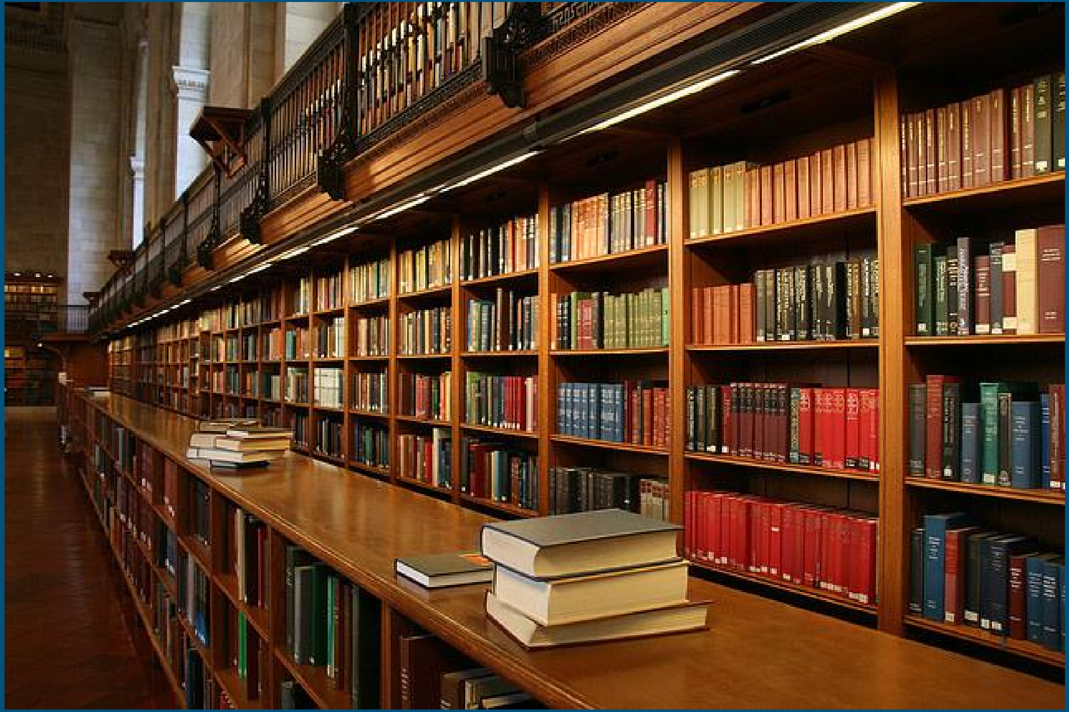


Photo: Viva Vivanista (Flickr)

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==1.4.2
```

Python + SNMP

Using PySNMP library

Using helper library I created, see:

~/python-libs/snmp_helper.py

Email notifications

Using helper library I created, see:

    ~/python-libs/email_helper.py

```
------------------
from email_helper import send_mail

sender = 'twb@twb-tech.com'
recipient = 'ktbyersx@gmail.com'
subject = 'This is a test message.'
message = '''Whatever'''

send_mail(recipient, subject, message, sender)
```

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
```

Data Serialization

Why do we need data serialization?

Characteristics of JSON

Characteristics of YAML

Day3 Schedule

1. Python and SSH
2. Concurrency Basics
   - Threads
   - Processes
3. Netmiko Tools
4. Arista eAPI
5. Juniper, NETCONF, and PyEZ
   - What is NETCONF
   - PyEZ
   - PyEZ get operations
5. Integrating to a database
6. NAPALM

Flickr: Pierre-Olivier Carles

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	Limited testing	Experimental
Arista vEOS	Alcatel AOS6/AOS8	A10
Cisco ASA	Avaya ERS	Accedian
Cisco IOS	Avaya VSP	Alcatel-Lucent SR-OS
Cisco IOS-XE	Brocade VDX	Aruba
Cisco IOS-XR	Brocade ICX/FastIron	Ciena SAOS
Cisco NX-OS	Brocade MLX/NetIron	Cisco Telepresence
Cisco SG300	Cisco WLC	CheckPoint Gaia
HP Comware7	Dell-Force10 DNOS9	Enterasys
HP ProCurve	Dell PowerConnect	Extreme EXOS
Juniper Junos	Huawei	Extreme Wing
Linux	Mellanox	F5 LTM
	Palo Alto PAN-OS	Fortinet
	Pluribus	MRV Communications OptiSwitch
	Vyatta VyOS	

Threads/Processes

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

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
```

Arista eAPI

```
import ssl
import jsonrpclib
from getpass import getpass

ssl._create_default_https_context = ssl._create_unverified_context
ip = '184.105.247.72'
username = 'admin1'
password = getpass()
url = 'https://{}:{}@{}:{}/command-api'.format(username, password, ip, port=’443’)

eapi_connect = jsonrpclib.Server(url)
response = eapi_connect.runCmds(1, ['show version'])
```

Using pyeapi library

```
import pyeapi

pynet_sw = pyeapi.connect_to("pynet-sw2")
show_version = pynet_sw.enable("show version")
```

~/.eapi.conf file contains connection definition information

Data Gathering and Config Automation using eAPI

- Data Gathering using eAPI

- Configuration Automation using eAPI

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()
```

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()
```

Integrating to a DB

- Django ORM
- Defining the DB
- Creating the DB
- Primary Keys, Foreign Keys
- CRUD Operations

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
fortios
nxos
ios
pluribus
panos

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()
```

Day4 Schedule

1. Ansible Overview
2. Ansible Inventory System
3. Simple Ansible Playbooks
4. Ansible Variables
5. Ansible Conditionals and Loops
6. Core Networking Modules

7. Ansible Config Templating
8. NAPALM + Ansible
9. Ansible Roles
10. Additional Playbook Topics
11. Dynamic Inventory
12. Writing custom Ansible Modules

Ansible Overview

- Ansible Introduction
- Ansible Terminology: Playbook, Play, Task
- Ansible Inventory

  - /etc/ansible/hosts

  - Overridden with -i option

  - host_vars

  - group_vars

Ansible Core Networking Modules

platform_facts
platform_command
platform_config

match: line/strict/exact
replace: line/block
parents
before

Ansible Config Templating

```
---
- name: Configuration templating
  hosts: localhost
  tasks:
  - name: Generate configuration files
    template: src=access_switch.j2 dest=CFGS/{{ item.hostname }}.txt
    with_items:
      - {hostname: pynet-sw1, ip_addr: 10.10.10.20}
      - {hostname: pynet-sw2, ip_addr: 10.10.20.20}
```

Ansible Config Templating

access_switch.j2

```
!
!
service timestamps debug datetime msec localtime show-timezone
service timestamps log datetime msec localtime show-timezone
!
hostname {{ item.hostname }}
!
logging buffered 32000
no logging console
```

Ansible Config Templating

Jinja2

```
{% if item.field %}
ip access-list extended TEST-ACL
  permit ip host 1.1.1.1 any log
  permit ip host 2.2.2.2 any log
{% elif item.otherfield %}
ip access-list extended TEST-ACL
  permit ip host 3.3.3.3 any log
{% else %}
ip access-list extended TEST-ACL
  permit ip host 4.4.4.4 any log
{% endif %}
```

Jinja2

```
{% for port_number in range(1,25) %}
interface FastEthernet0/{{ port_number }}
 switchport access vlan {{ item.access_vlan }}
!
{% endfor %}
```

NAPALM + Ansible

```
- name: NAPALM default configuration (Arista)
  hosts: arista
  gather_facts: False
  tasks:
    - napalm_install_config:
        hostname: "{{ ansible_host }}"
        username: "{{ username }}"
        password: "{{ password }}"
        dev_os: eos
        config_file: "CFGS/{{ inventory_hostname }}.txt"
        commit_changes: True
        replace_config: True
        get_diffs: True
        diff_file: "DIFFS/{{ inventory_hostname }}.diff"
      tags: arista
```

Ansible roles / adding structure

```
- name: Build Python + Ansible (Group A)
  hosts: server1
  sudo: yes

  roles:
    - server
    - applied_python
    - netmiko
    - arista
    - django
    - juniper
```

Directories

```
./roles/access_switch/files
./roles/access_switch/handlers
./roles/access_switch/tasks
./roles/access_switch/templates
./roles/access_switch/vars
```

Additional Ansible Topics

Handlers
Tags
Register
Limit (--limit)

Ansible Filters / Custom Filters
Ansible Lookups

```
- set_fact:
    show_arp_new: "{{ show_arp.stdout_lines[0] }}"
```

Ansible Dynamic Inventory

```
$ ansible-playbook my_playbook.yml -i ./dyn_inv.py
```

The --list option must list out all of the groups and the associated hosts and group variables.
The --host option must either return an empty dictionary or a dictionary of variables relevant to that host.

https://github.com/ktbyers/pynet/blob/master/ansible/dyn_inv_v1.py

http://docs.ansible.com/ansible/dev_guide/developing_inventory.html

Creating an Ansible Module

```
from ansible.module_utils.basic import AnsibleModule

def main():
    module = AnsibleModule(
        argument_spec = dict(
            state     = dict(default='present', choices=['present', 'absent']),
            name      = dict(required=True),
            enabled   = dict(required=True, type='bool'),
            something = dict(aliases=['whatever'])
        )
    )

module.exit_json(changed=True, something_else=12345)

module.fail_json(msg="Something fatal happened")
```

Ansible Vault

NTC-Ansible Modules

ntc_config_command.py
ntc_file_copy.py
ntc_get_facts.py
ntc_install_os.py
ntc_reboot.py
ntc_rollback.py
ntc_save_config.py
ntc_show_command.py

The end…

Questions?

ktbyers@twb-tech.com
Twitter: @kirkbyers