PYTHON
FOR NETWORK ENGINEERS

Onsite Training Session
February 2019

$ whoami

Kirk Byers
Network Engineer:
CCIE #6243 (emeritus)

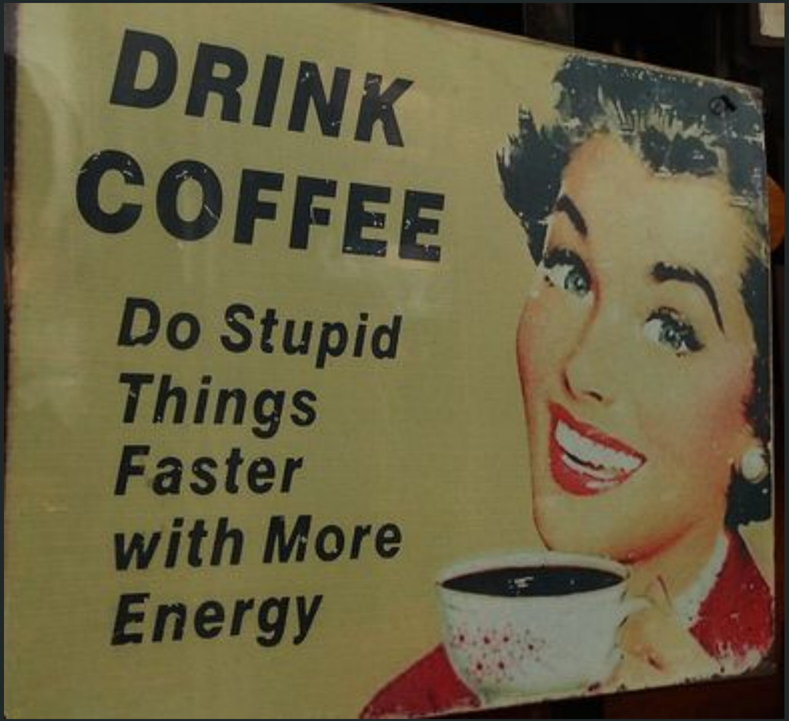
Programmer:
Netmiko
NAPALM
Nornir

Teach Python and Ansible in a Network Automation context

General:

8:30AM - 4:30PM
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

1a. GIT Basics

1b. VI in five minutes

2. Python Fundamentals - General

3. Strings

4. Numbers

5. Files

6. Lists / Tuples

7. Booleans / None

8. Conditionals

9. Loops

10. Dictionaries

11. Exceptions

12. Functions

13. Regular Expressions

Git

- Why care about Git?
- Git and GitHub
- Some principles of how Git works
  - Tracking files and directories across time
  - All objects are stored in the .git directory
  - You can swap your working set of files
  - Distributed
- Creating a repository on GitHub
- Cloning a repository
- git init
- Files have four different states: untracked, modified, staged, committed

Git Adding/Removing Files

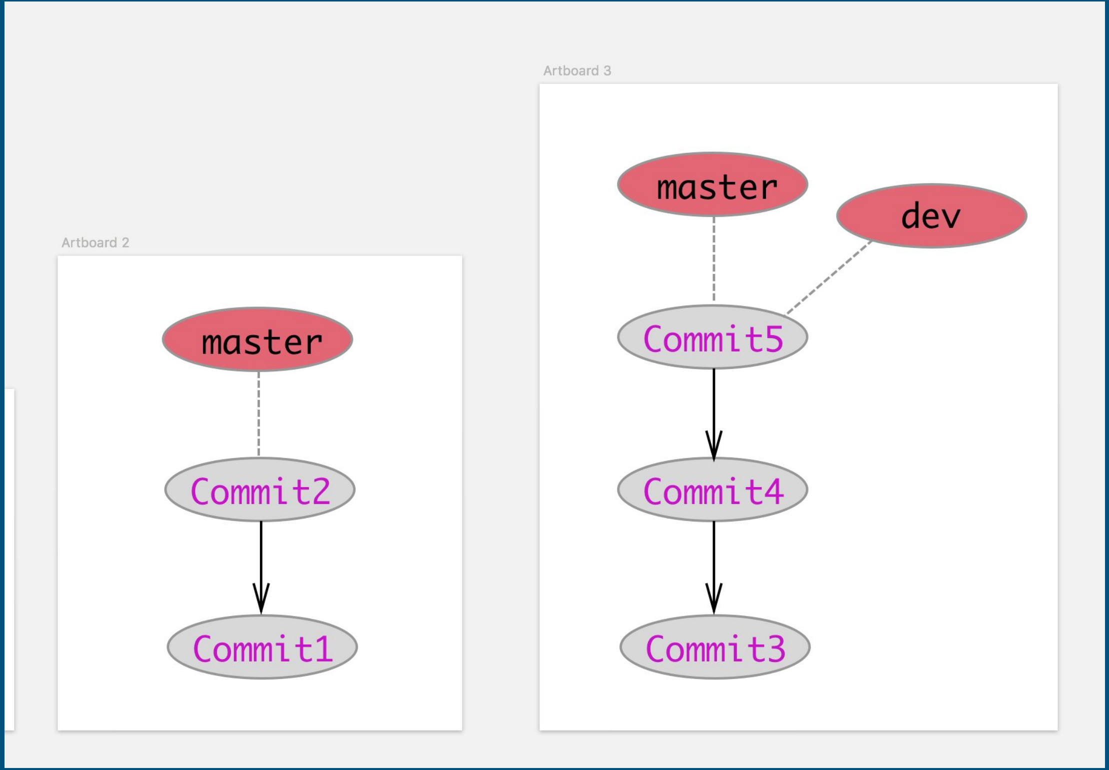
- git status *# basically what is the current state of this repository*
- git branch *# which branches are there and which branch am I working on*
- Adding/Removing files
  - git add / git rm / git commit
  - git diff *# to see what changed on a file or set of files*
- git log *# to see the history of commits*

Git Push & Pull

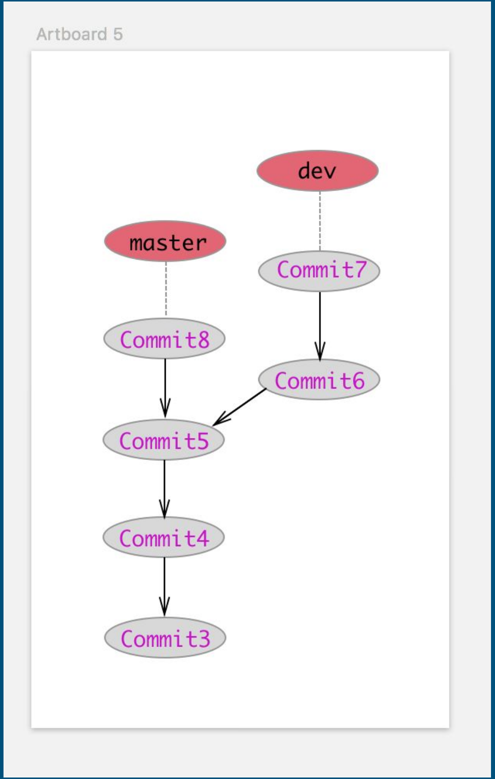
Changes have been committed locally, but haven’t been pushed up to GitHub

- git pull / git push
    - git remote -v
    - git remote add
    - git branch -vv

Git Branches



Git Branches



Git Branches

Creating a branch

- git checkout -b dev origin/master
- git branch dev2
- git checkout dev2
- git branch *# Look at your current branches*
- Switching branches
  - Underlying files in the working directory change

Merge operation

- Checkout the branch you want to merge into
- git merge dev2

Git Handling Merge Conflicts

A set of changes that Git can't reconcile

```
    $ git merge dev
    Auto-merging test2.py
    CONFLICT (content): Merge conflict in test2.py
    Automatic merge failed; fix conflicts and then
commit the result.
```

```
$ cat test2.py
------------
    while True:
        print("Hello world")
        break

    for x in range(10):
        x = 0
    <<<<<<< HEAD
        y = 1 * x
        z = 3
        print(y)

    print("Foo")
    =======
        y += 1
        z = 3

    >>>>>>> dev
```

Git Pull Requests / Git Rebase

Pull Request - Submit changes from your copy of a repository for review and potentially integration into the main repository for the project.

Rebase - One of your branches has become out of date (relative to another copy of the repository) and you want to bring it back up to date.

Git Exercises

Reference Commands:

{{ github_repo }}/git_notes/git_commands.md

Exercises:
./day1/git_ex1.txt
./day1/git_ex2.txt

VI in five minutes

SSH into lab environment

vi test1.txt

Two modes: edit-mode and command-mode (ESC is your path to safety).

i - insert (switch to edit-mode)
a - append (switch to edit-mode)

Never, absolutely never, hit caps-lock it is the path to destruction and ruin.

Use h, j, k, l to navigate (in command-mode)

VI in five minutes

Use h, j, k, l to navigate (in command-mode)

h - move left one space
j - move down one space
k - move up one space
l - move right one space

Arrow keys will also probably work.

x - delete a character
dw - delete a word
dd - delete a line

<u>To exit</u>

:wq - saves file and exits
:q!   - exits WITHOUT saving

u - undo the last command
yy - yank a line
p - put a line

REMEMBER:
<esc> is your friend

Why Python?

PYTHON FOR NETWORK ENGINEERS

- 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. *# The battle is now largely over: use Python3.*

Python2 support ends on Jan1, 2020.

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
- f-strings

Exercises:
./day1/str_ex1.txt
./day1/str_ex2.txt

Numbers

Integers

Floats

Math Operators (+, -, *, /, **, %)

~~Strange Behavior of Integer Division~~

Exercises:
./day1/numbers_ex1.txt

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()
```

Exercises:
./day1/files_ex1.txt

Lists

Zero-based indices

.append()

.pop()

.join()

List slices

Tuple

Copying a list

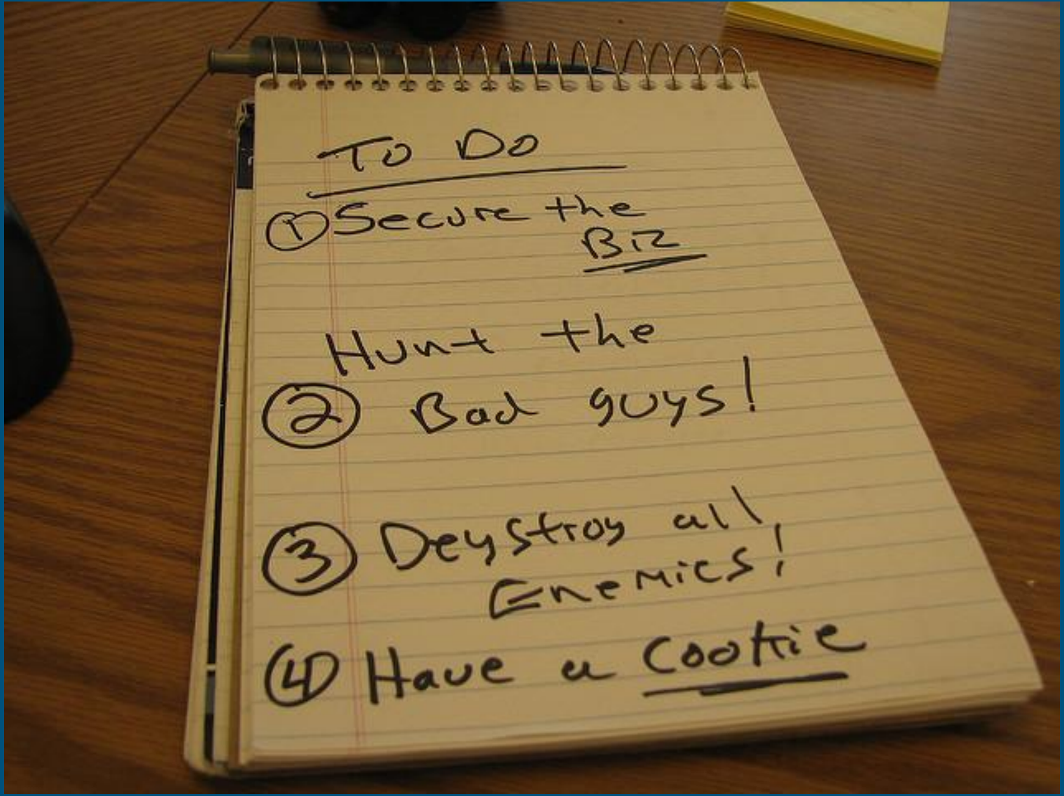


Photo: Purple Slog (Flickr)

Exercises:
./day1/lists_ex1.txt
./day1/lists_ex2.txt

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
```

Exercises:
./day1/loops_ex1.txt
./day1/loops_ex2.txt

Exercise:

Exercises:
./day1/for_bgp_parse_ex1.txt

Show IP BGP Parsing

Read the ‘show_ip_bgp.txt’ file.

Strip out the header information so you are just left with the routes.

Parse each BGP line such that you retrieve the prefix and the as_path.

Save the prefix and as_path to a file.

Exercise:

Exercises:
./day1/for_cond_show_ver_ex1.txt

Show Version Exercise

----

a. Read a show version output from a router (in a file named, “show_version.txt”.

b. Find the router serial number in the output.

c. Parse the serial number and return it as a variable. Use .split() and substr in str to accomplish this.

Dictionaries

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

Exercises:
./day1/dict_ex1.txt

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.

Exercises:
./day1/except_dict_ex1.txt

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

Exercises:
./day1/func_ex1.txt
./day1/func_ex2.txt
./day1/func_ex3.txt
./day1/func_ex4.txt

Python Regular Expresions

import re

Other re methods

re.split()

re.sub()

re.findall()

Exercises:
./day1/regex_ex1.txt
./day1/regex_ex2.txt

re.search(pattern, string)

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

match.group(0)
match.groups()
match.groupdict()

Named patterns

(?P<software_ver>Ver.*)

Regular Expression Resources

Regular Expression Tutorial
https://regexone.com/lesson/introduction_abcs
This is a good resource if you are new to regular expressions.

Online Regular Expression Tester
https://regex101.com/
Select 'Python' on the left-hand side.

Python Regular Expression HowTo
https://docs.python.org/2/howto/regex.html
This is a good overview of regular expression special characters.
Start at the very top of the page and read through the 'Repeating Things' section.

Day2

1. Python Classes and Objects
2. Python Code Structure
3. Libraries
4. Modules
5. Linters
6. Email Notifications
7. CiscoConfParse
8. Netmiko
9. Netmiko Tools

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

Flickr: au_tiger01

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()
```

Exercises:
./day2/classes_ex1.txt
./day2/classes_ex2.txt

Python Code Structure:

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

Exercises:
./day2/reuse_ex1.txt

Libraries

import x

from x import y

sys.path

PYTHONPATH

Installing packages (pip)

Virtual Environments

Photo: Viva Vivanista (Flickr)

Virtualenv

```
virtualenv-3.6 -p /usr/bin/python36 test_venv

source /path/to/virtualenv/bin/activate

deactivate

pip list

pip install netmiko==2.3.0
pip install pycodestyle

pip freeze
```

Exercises:
./day2/virtualenv_ex1.txt

Modules and Packages

Python Module

A Python file that you can import into another Python program

Example, storing device definition in an external file.

Python Package

An importable Python directory

__init__.py

Exercises:
./day2/reuse_ex2.txt
./day2/reuse_ex3.txt

Python Linters

Auto formatting with Python Black

pylint or pycodestyle

Consistency and conventions make your life easier.

Finds obvious errors. Finds problems you might not be aware of (reuse of builtins).

```
pylint my_file.py
pycodestyle my_file.py
pylama my_file.py
```

Review Exercise

Process the 'show_ip_int_brief.txt' file and create a data structure from it.

1. Create a dictionary of dictionaries.
2. The keys for the outermost dictionary should be the interface names.
3. The value corresponding to this interface name is another dictionary with the fields 'ip_address', 'line_status', and 'line_protocol'.
4. Use pretty-print to print out your data structure.

Your output should be similar to the following:

```
{'FastEthernet0': {'ip_address': 'unassigned',
                   'line_protocol': 'down',
                   'line_status': 'down'},
 ... }
```

Exercises:
./day2/review_ex1.txt

Review Exercise

Process the 'show_arp.txt' file and create a data structure from it.

1. Create a dictionary where the keys are the ip addresses and the corresponding values are the mac-addresses.

2. Create a second dictionary where the keys are the mac-addresses and the corresponding values are the ip addresses.

3. Use pretty print to print these two data structures to the screen.

Exercises:
./day2/review_ex2.txt

Email notifications

Reference Material in:

{{ github_repo }}/email_example

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
```

Reference Material in:
{{ github_repo }}/confparse_example

Exercises:
./day2/confparse_ex1.txt
./day2/confparse_ex2.txt

Netmiko

Paramiko is the standard Python SSH library.

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

Netmiko Vendors

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

Limited testing
- Alcatel AOS6/AOS8
- Avaya ERS
- Avaya VSP
- Brocade VDX
- Brocade MLX/NetIron
- Calix B6
- Cisco WLC
- Dell-Force10
- Dell PowerConnect

Limited testing
- Huawei
- Mellanox
- NetApp cDOT
- Palo Alto PAN-OS
- Pluribus
- Ruckus ICX/FastIron
- Ubiquity EdgeSwitch
- Vyatta VyOS

Experimental
- A10
- Accedian
- Aruba
- Ciena SAOS
- Cisco Telepresence
- CheckPoint GAiA
- Coriant
- Eltex
- Enterasys
- Extreme EXOS
- Extreme Wing
- F5 LTM
- Fortinet
- MRV OptiSwitch
- Nokia SR-OS
- QuantaMesh

Key Netmiko Methods

.send_command()
.send_command_timing()

.send_config_set()
.send_config_from_file()

.save_config()

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

.write_channel()
.read_channel()

FileTransfer Class

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',
        'host':   '184.105.247.76',
        'username': 'pyclass',
        'password': password
    }

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

Reference Material in:

{{ github_repo }}/netmiko_example
{{ github_repo }}/paramiko_example
{{ github_repo }}/pexpect_example

Exercises:
./day2/netmiko_ex1.txt
./day2/netmiko_ex2.txt

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

Day3 Schedule

- Serialization: JSON and YAML
- XML
- NX-API
- Requests and using a REST-API

Flickr: Pierre-Olivier Carles

Data Serialization

Why do we need data serialization?

Characteristics of JSON

Characteristics of YAML

Reference Material in:

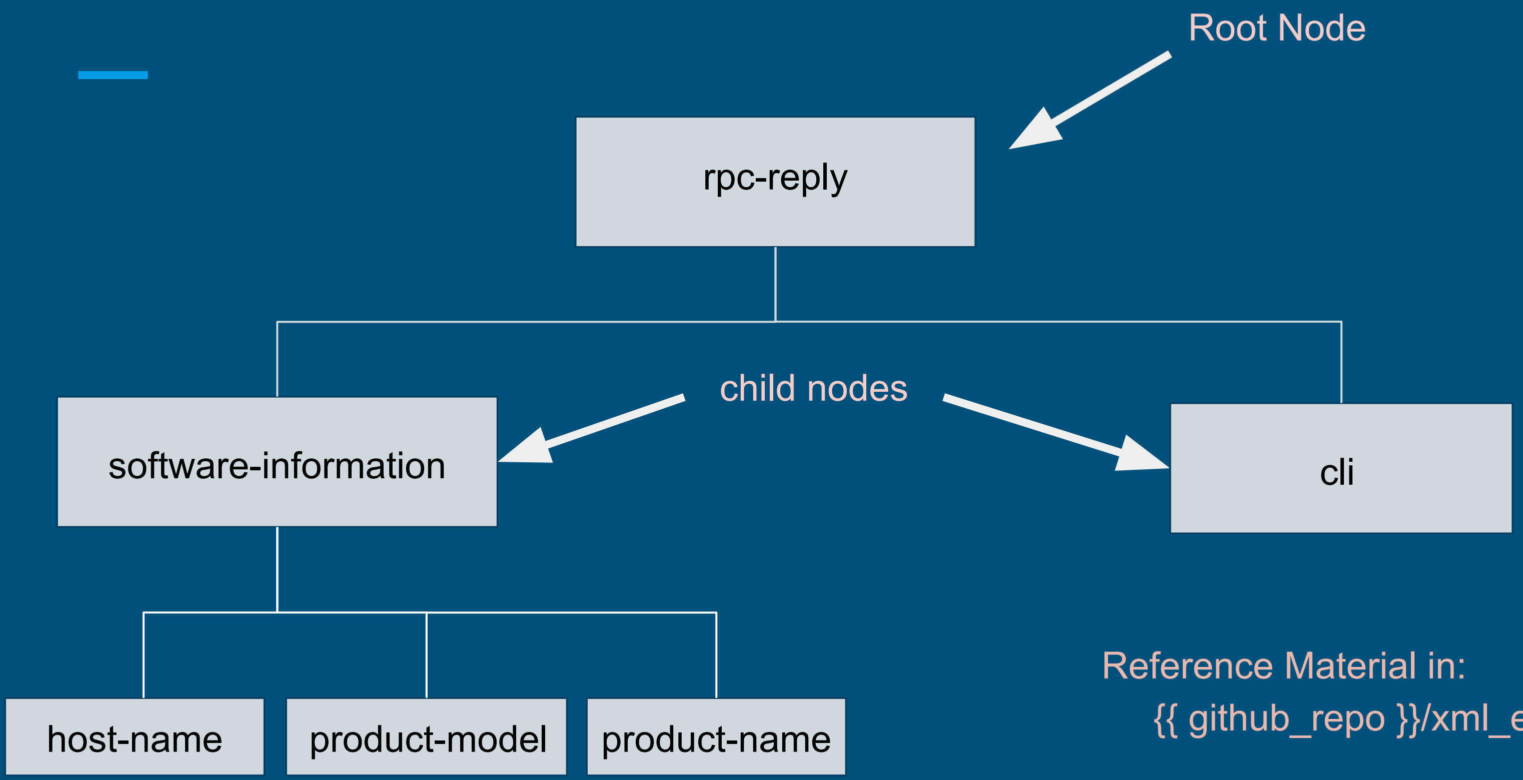
{{ github_repo }}/json_yaml

Exercises:
./day3/yaml_ex1.txt
./day3/yaml_ex2.txt
./day3/netmiko_ex3.txt

XML: the good, the bad, and the ugly

```
> show version | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/12.1X44/junos">
    <software-information>
        <host-name>pynet-jnpr-srx1</host-name>
        <product-model>srx100h2</product-model>
        <product-name>srx100h2</product-name>
        <jsr/>
        <package-information>
            <name>junos</name>
            <comment>JUNOS Software Release [12.1X44-D35.5]</comment>
        </package-information>
    </software-information>
    <cli>
        <banner></banner>
    </cli>
</rpc-reply>
```

XML - Think of it as a tree of nodes



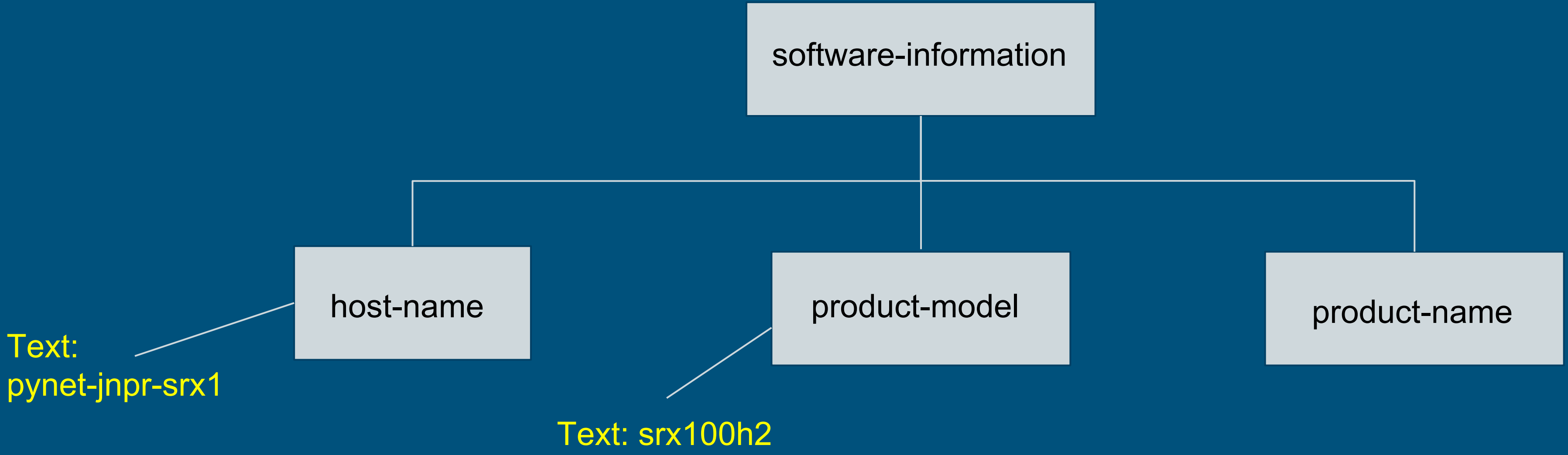
Reference Material in:
{{ github_repo }}/xml_example

XML Text “Nodes”

```
> show version | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/12.1X44/junos">
    <software-information>
        <host-name>pynet-jnpr-srx1</host-name>
        <product-model>srx100h2</product-model>
        <product-name>srx100h2</product-name>
        <jsr/>
        <package-information>
            <name>junos</name>
            <comment>JUNOS Software Release [12.1X44-D35.5]</comment>
        </package-information>
    </software-information>
    <cli>
        <banner></banner>
    </cli>
</rpc-reply>
```

XML Text “Nodes” (ElementTree/lxml Perspective)

Treat the Text as an attribute of the Element Node

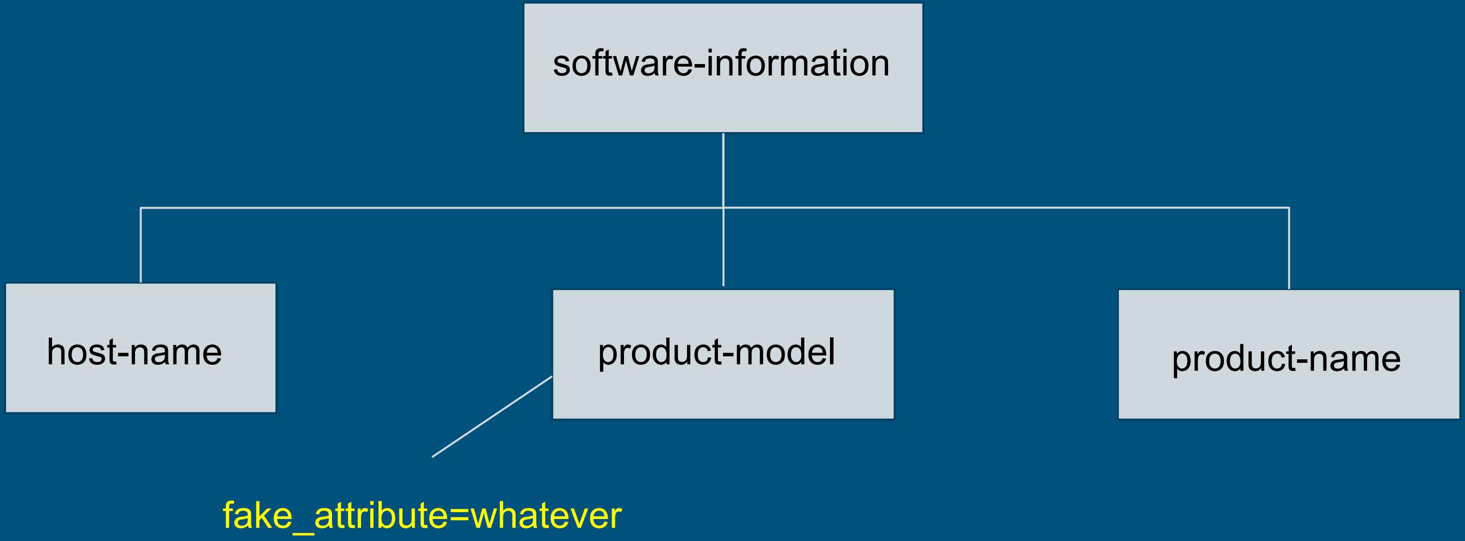


XML Attribute “Nodes”

```
> show version | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/12.1X44/junos">
    <software-information>
        <host-name>pynet-jnpr-srx1</host-name>
        <product-model fake_attribute=’whatever’>srx100h2</product-model>
        <product-name>srx100h2</product-name>
        <jsr/>
        <package-information>
            <name>junos</name>
            <comment>JUNOS Software Release [12.1X44-D35.5]</comment>
        </package-information>
    </software-information>
    <cli>
        <banner></banner>
    </cli>
</rpc-reply>
```

XML Attribute “Nodes” (ElementTree/lxml Perspective)

Treat the Attribute as an attribute of the Element Node



This is not what the DOM does?

In the DOM (document object model):

The following are nodes (and other things are also nodes):

- Element Nodes
- Text Nodes
- Attribute Nodes

Implications of this when using Python

Terminology: Element

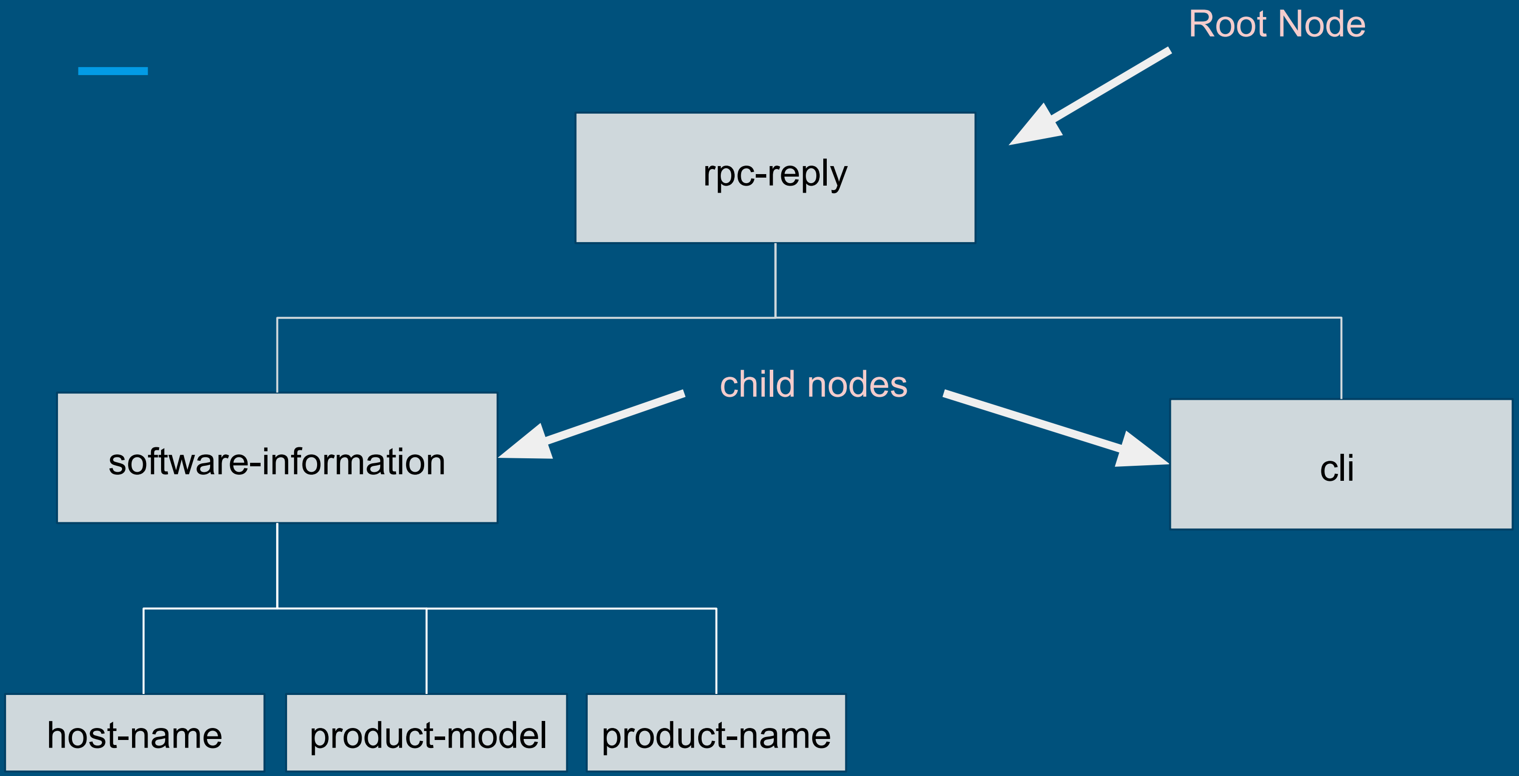
```
> show version | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/12.1X44/junos">
    <software-information>
        <host-name>pynet-jnpr-srx1</host-name>
        <product-model>srx100h2</product-model>
        <product-name>srx100h2</product-name>
        <jsr/>
        <package-information>
            <name>junos</name>
            <comment>JUNOS Software Release [12.1X44-D35.5]</comment>
        </package-information>
    </software-information>
    <cli>
        <banner></banner>
    </cli>
</rpc-reply>
```

Terminology:

Child Nodes
Parent Nodes
Sibling Nodes
Ancestor Nodes
Descendant Nodes

Namespaces - a way to uniquely identify the names of nodes.

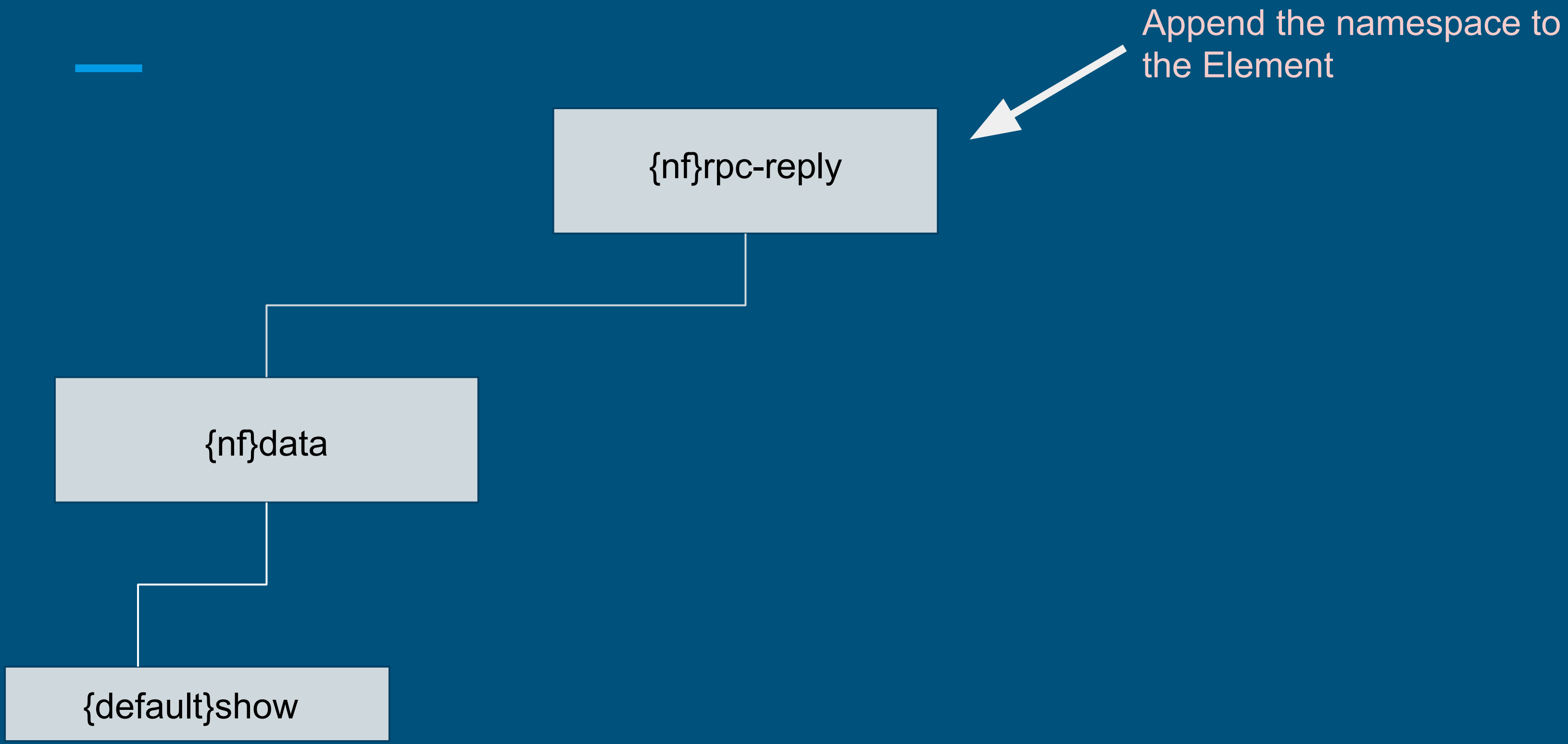
XML


Root Node
rpc-reply
child nodes
software-information
cli
host-name
product-model
product-name

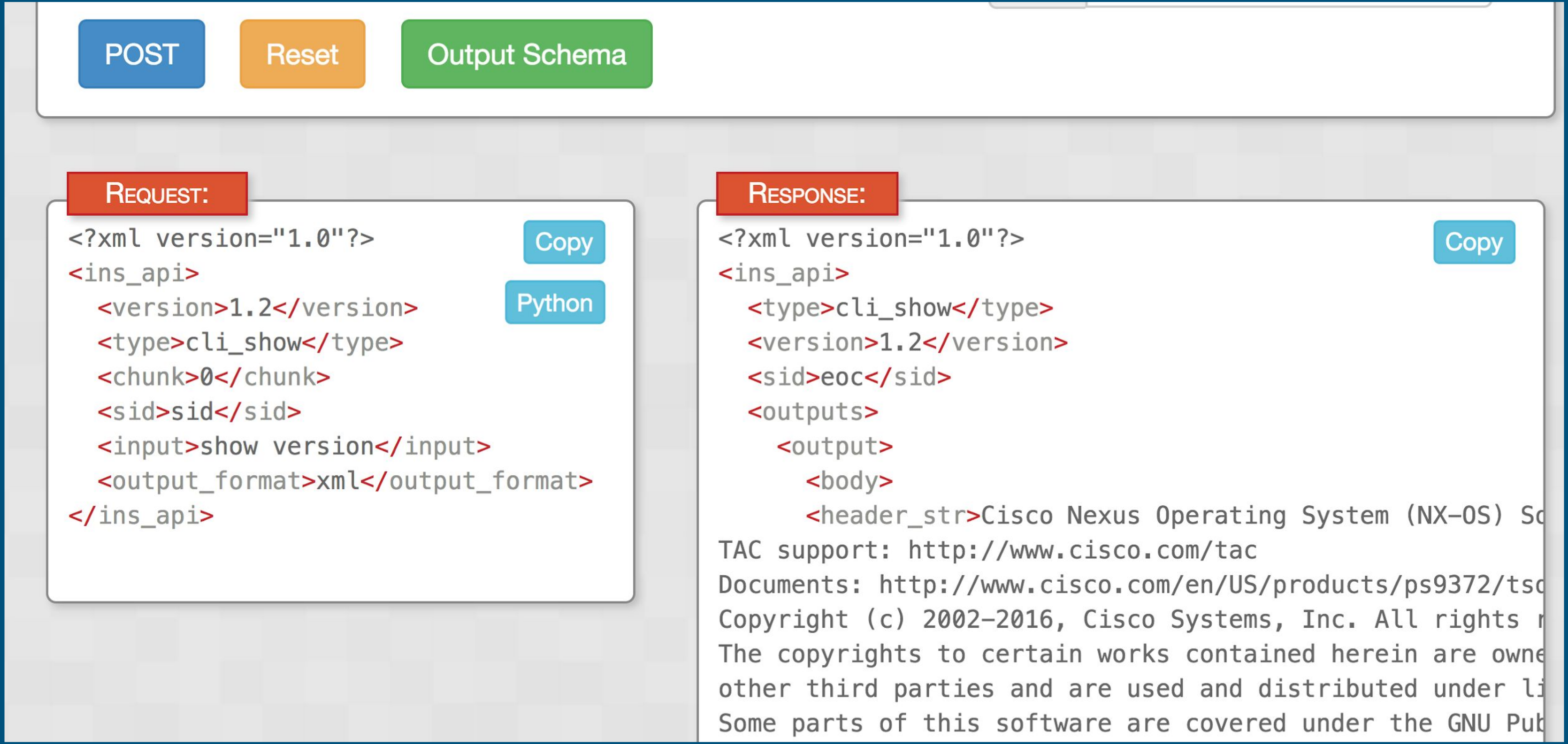
Namespaces

```
nxos1# show version | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:sysmgrcli">
 <nf:data>
  <show>
   <version>
       ...
   </version>
  </show>
 </nf:data>
</nf:rpc-reply>
```

XML with Namespaces



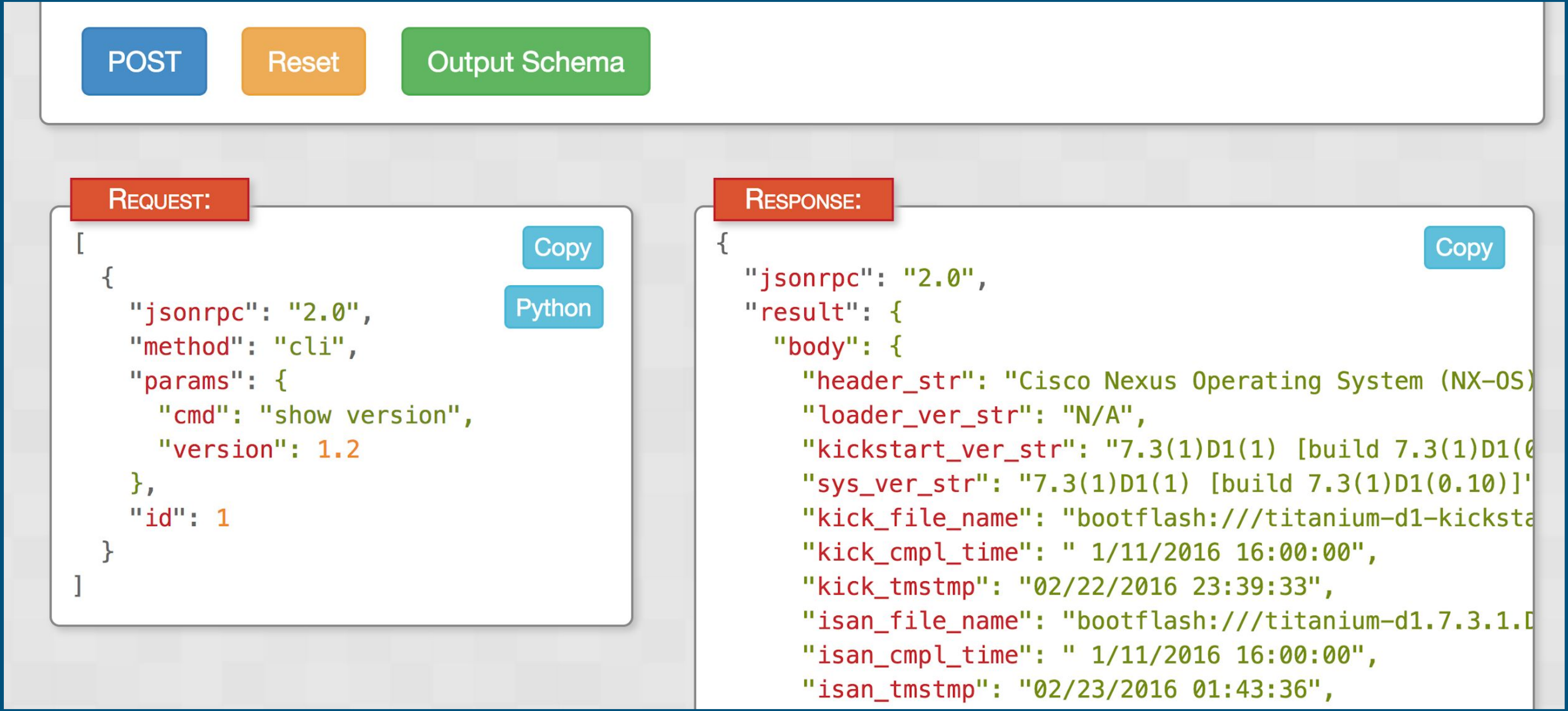
Cisco NX-OS and NX-API



```
<?xml version="1.0"?>
<ins_api>
  <version>1.2</version>
  <type>cli_show</type>
  <chunk>0</chunk>
  <sid>sid</sid>
  <input>show version</input>
  <output_format>xml</output_format>
</ins_api>
```

```
<?xml version="1.0"?>
<ins_api>
  <type>cli_show</type>
  <version>1.2</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body>
      <header_str>Cisco Nexus Operating System (NX-OS) So
TAC support: http://www.cisco.com/tac
Documents: http://www.cisco.com/en/US/products/ps9372/tsd
Copyright (c) 2002-2016, Cisco Systems, Inc. All rights r
The copyrights to certain works contained herein are owne
other third parties and are used and distributed under li
Some parts of this software are covered under the GNU Pub
```

Cisco NX-OS and NX-API (JSON-RPC)



```
[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show version",
      "version": 1.2
    },
    "id": 1
  }
]
```

```
{
  "jsonrpc": "2.0",
  "result": {
    "body": {
      "header_str": "Cisco Nexus Operating System (NX-OS)
      "loader_ver_str": "N/A",
      "kickstart_ver_str": "7.3(1)D1(1) [build 7.3(1)D1(0
      "sys_ver_str": "7.3(1)D1(1) [build 7.3(1)D1(0.10)]"
      "kick_file_name": "bootflash:///titanium-d1-kicksta
      "kick_cmpl_time": " 1/11/2016 16:00:00",
      "kick_tmstmp": "02/22/2016 23:39:33",
      "isan_file_name": "bootflash:///titanium-d1.7.3.1.D
      "isan_cmpl_time": " 1/11/2016 16:00:00",
      "isan_tmstmp": "02/23/2016 01:43:36",
```

Cisco NX-OS and NX-API

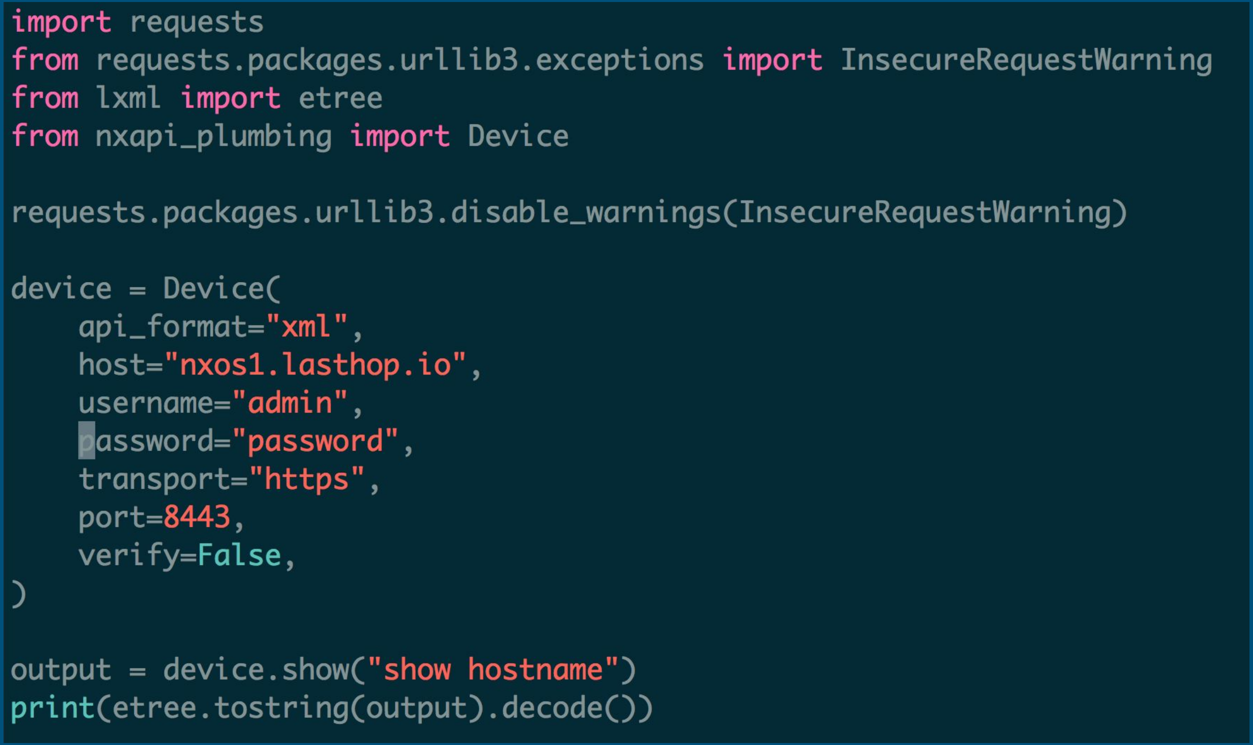
- Uses HTTP/HTTPS transport
- XML or JSON-RPC payload
- Python Libraries: nxapi-plumbing and pynxos
- NAPALM

Cisco NX-OS and NX-API

```python
import requests
from requests.packages.urllib3.exceptions import InsecureRequestWarning
from pprint import pprint
from nxapi_plumbing import Device

requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

device = Device(
    api_format="jsonrpc",
    host="nxos1.lasthop.io",
    username="admin",
    password="password",
    transport="https",
    port=8443,
    verify=False,
)

output = device.show("show hostname")
print(output)
```

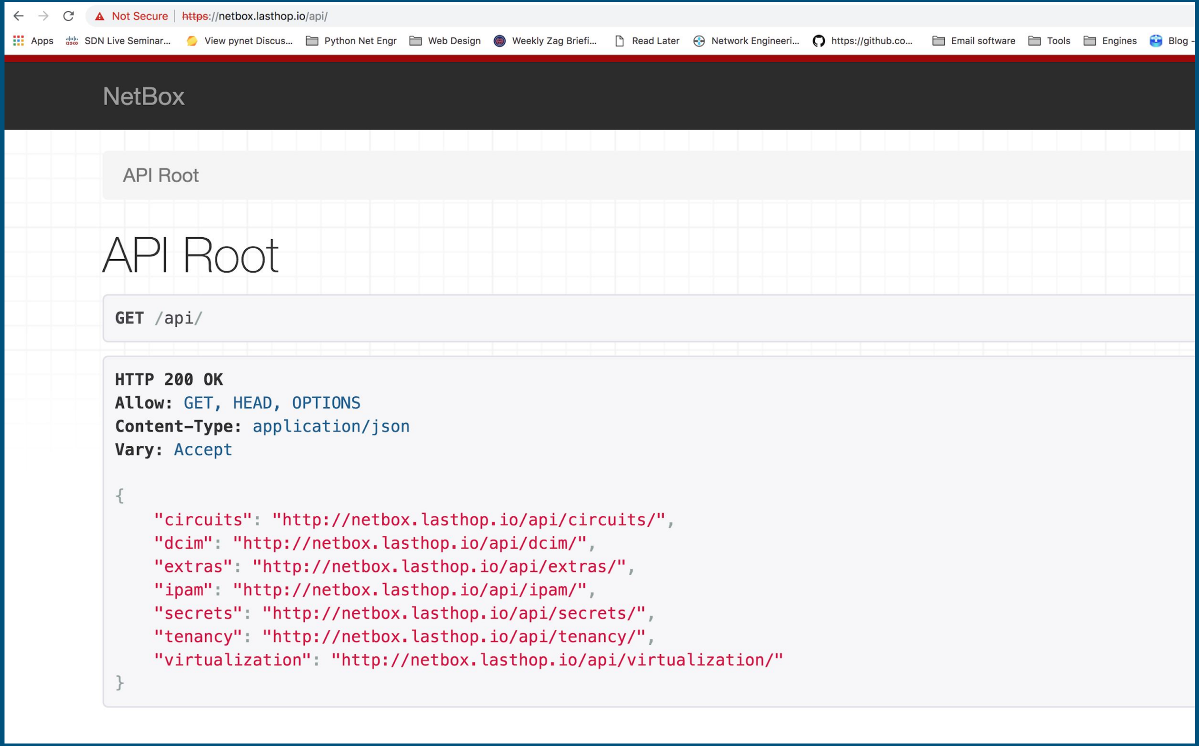
Cisco NX-OS and NX-API

Exercises:
./day3/nxapi_ex1.txt



```
import requests
from requests.packages.urllib3.exceptions import InsecureRequestWarning
from lxml import etree
from nxapi_plumbing import Device

requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

device = Device(
    api_format="xml",
    host="nxos1.lasthop.io",
    username="admin",
    password="password",
    transport="https",
    port=8443,
    verify=False,
)

output = device.show("show hostname")
print(etree.tostring(output).decode())
```

REST API



Not Secure | https://netbox.lasthop.io/api/

API Root

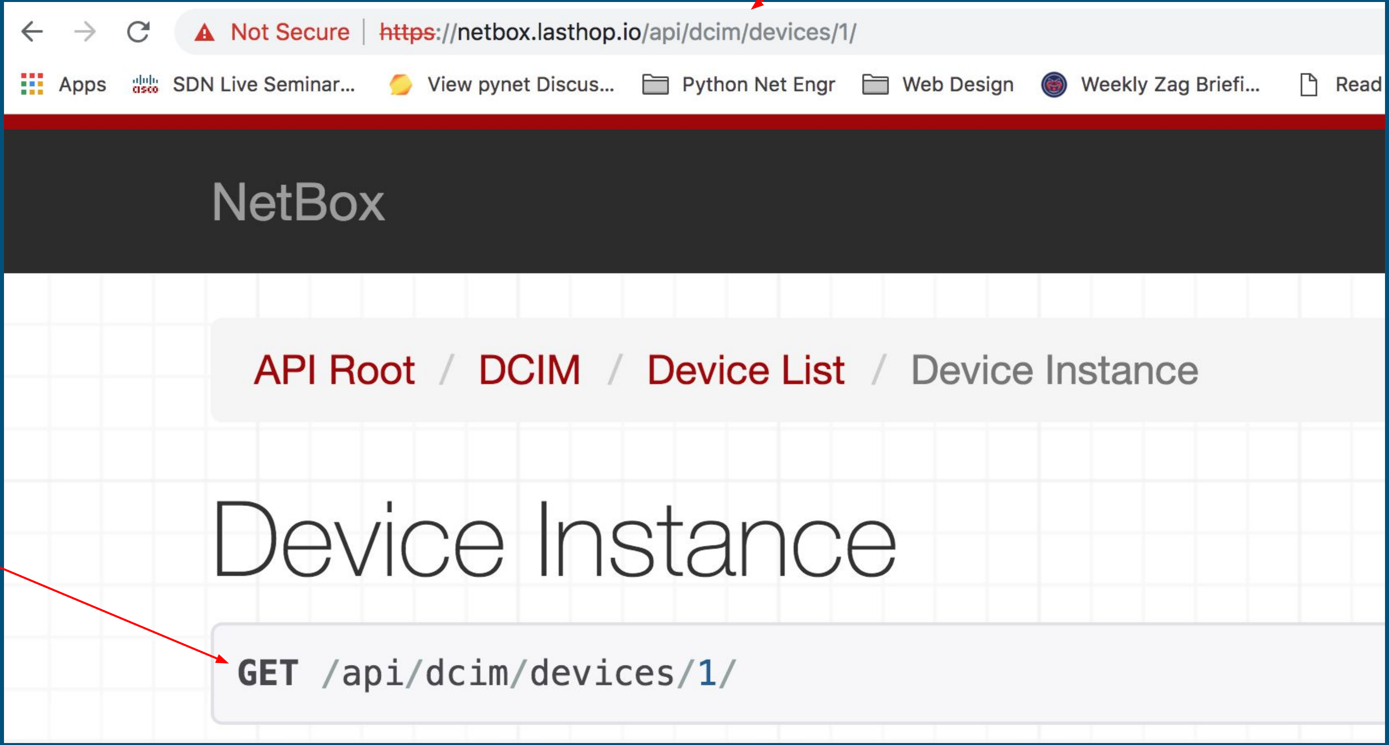
```
GET /api/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "circuits": "http://netbox.lasthop.io/api/circuits/",
    "dcim": "http://netbox.lasthop.io/api/dcim/",
    "extras": "http://netbox.lasthop.io/api/extras/",
    "ipam": "http://netbox.lasthop.io/api/ipam/",
    "secrets": "http://netbox.lasthop.io/api/secrets/",
    "tenancy": "http://netbox.lasthop.io/api/tenancy/",
    "virtualization": "http://netbox.lasthop.io/api/virtualization/"
}
```

REST API - Characteristics

URL - the object I am accessing.

HTTP Method



Not Secure | ~~https~~://netbox.lasthop.io/api/dcim/devices/1/

Apps SDN Live Seminar... View pynet Discus... Python Net Engr Web Design Weekly Zag Briefi... Read

NetBox

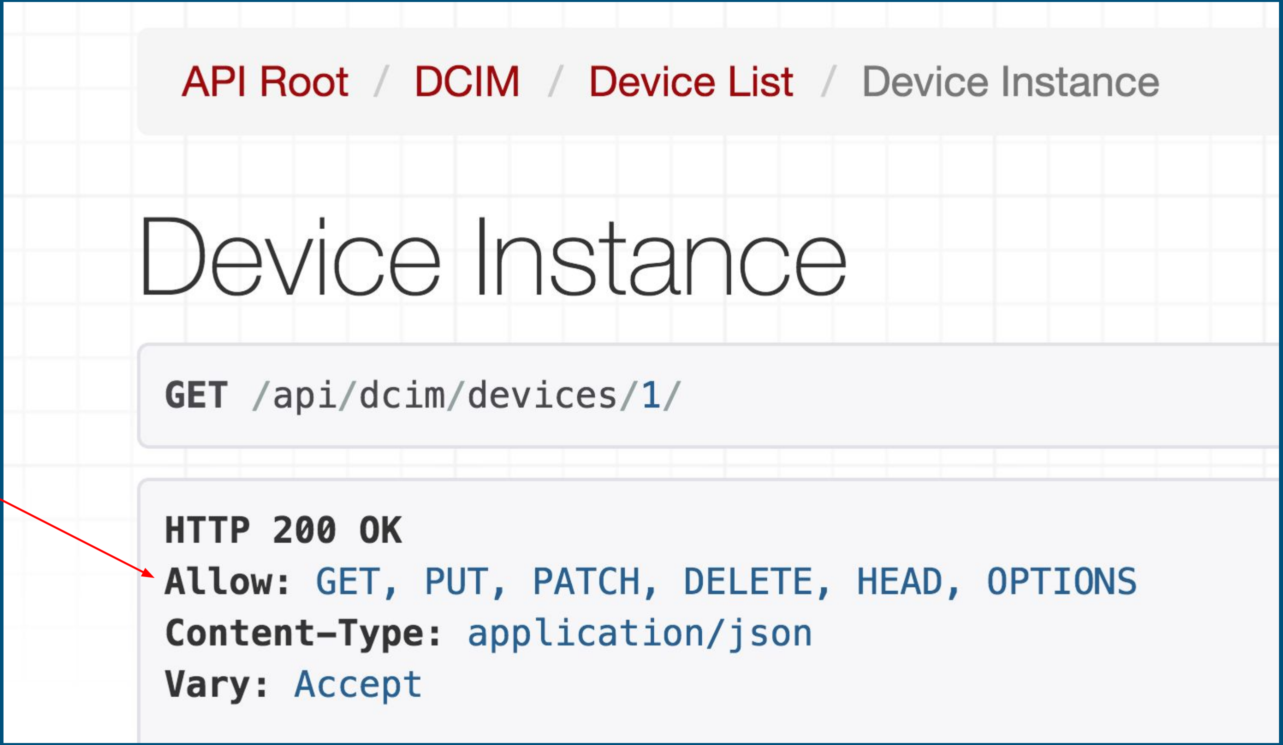
API Root / DCIM / Device List / Device Instance

Device Instance

```
GET /api/dcim/devices/1/
```

REST API - Other HTTP Methods

Available HTTP Methods



API Root / DCIM / Device List / Device Instance

Device Instance

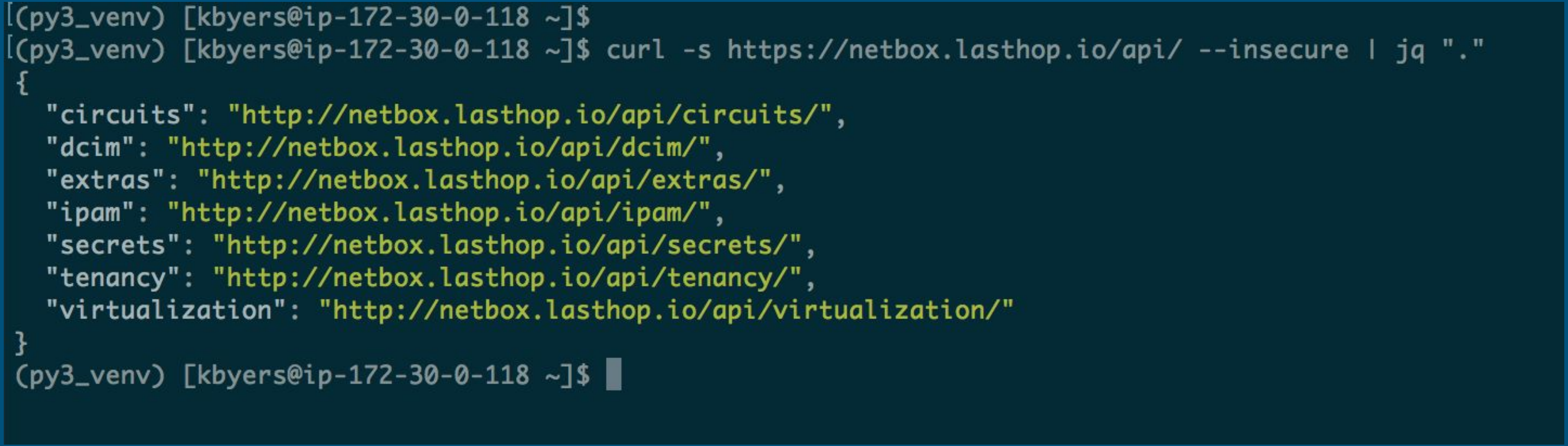
```
GET /api/dcim/devices/1/
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```

REST API - CRUD

- Create - HTTP Post
- Read - HTTP Get
- Update - HTTP Put / HTTP Patch
- Delete - HTTP Delete

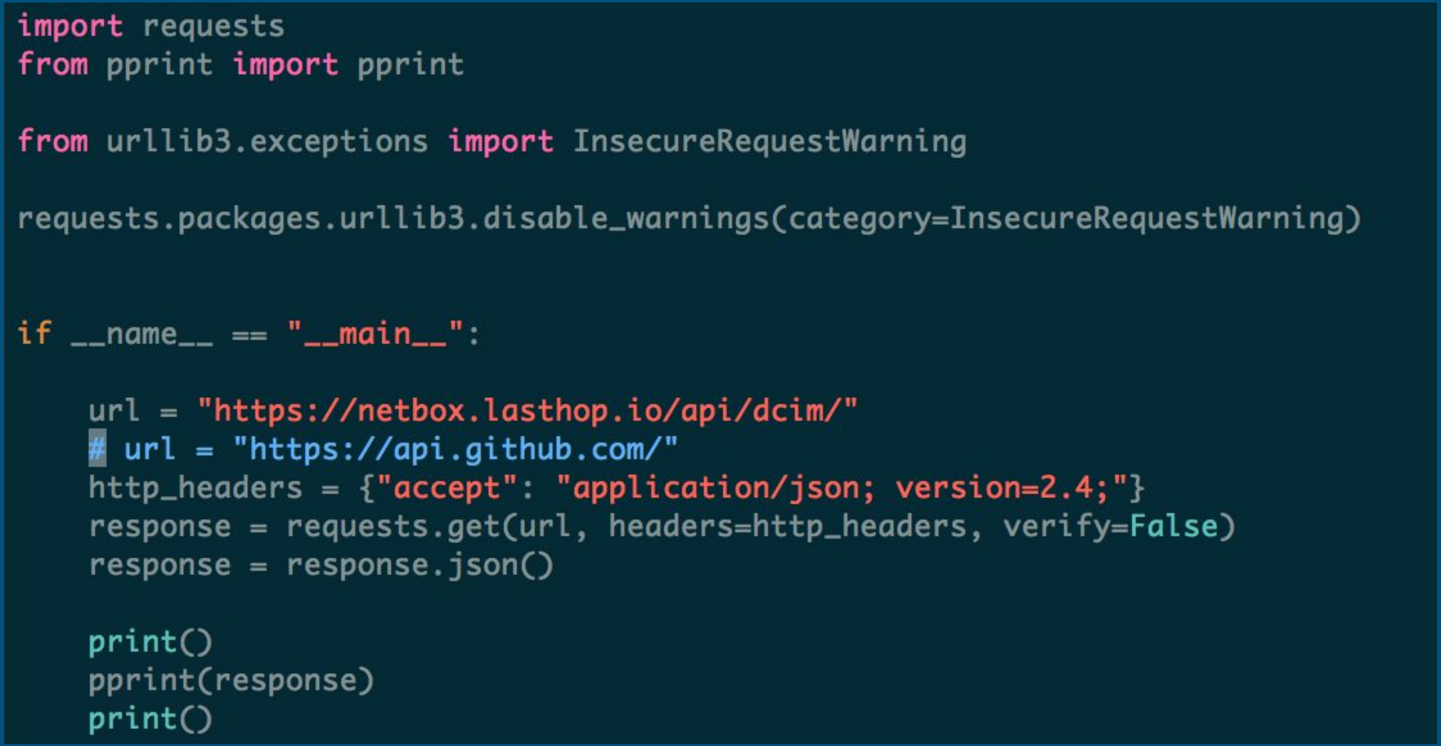
REST API - Accessing API via Browser + CLI



```
[(py3_venv) [kbyers@ip-172-30-0-118 ~]$
[(py3_venv) [kbyers@ip-172-30-0-118 ~]$ curl -s https://netbox.lasthop.io/api/ --insecure | jq "."
{
  "circuits": "http://netbox.lasthop.io/api/circuits/",
  "dcim": "http://netbox.lasthop.io/api/dcim/",
  "extras": "http://netbox.lasthop.io/api/extras/",
  "ipam": "http://netbox.lasthop.io/api/ipam/",
  "secrets": "http://netbox.lasthop.io/api/secrets/",
  "tenancy": "http://netbox.lasthop.io/api/tenancy/",
  "virtualization": "http://netbox.lasthop.io/api/virtualization/"
}
(py3_venv) [kbyers@ip-172-30-0-118 ~]$
```

REST API - Basic Requests Get

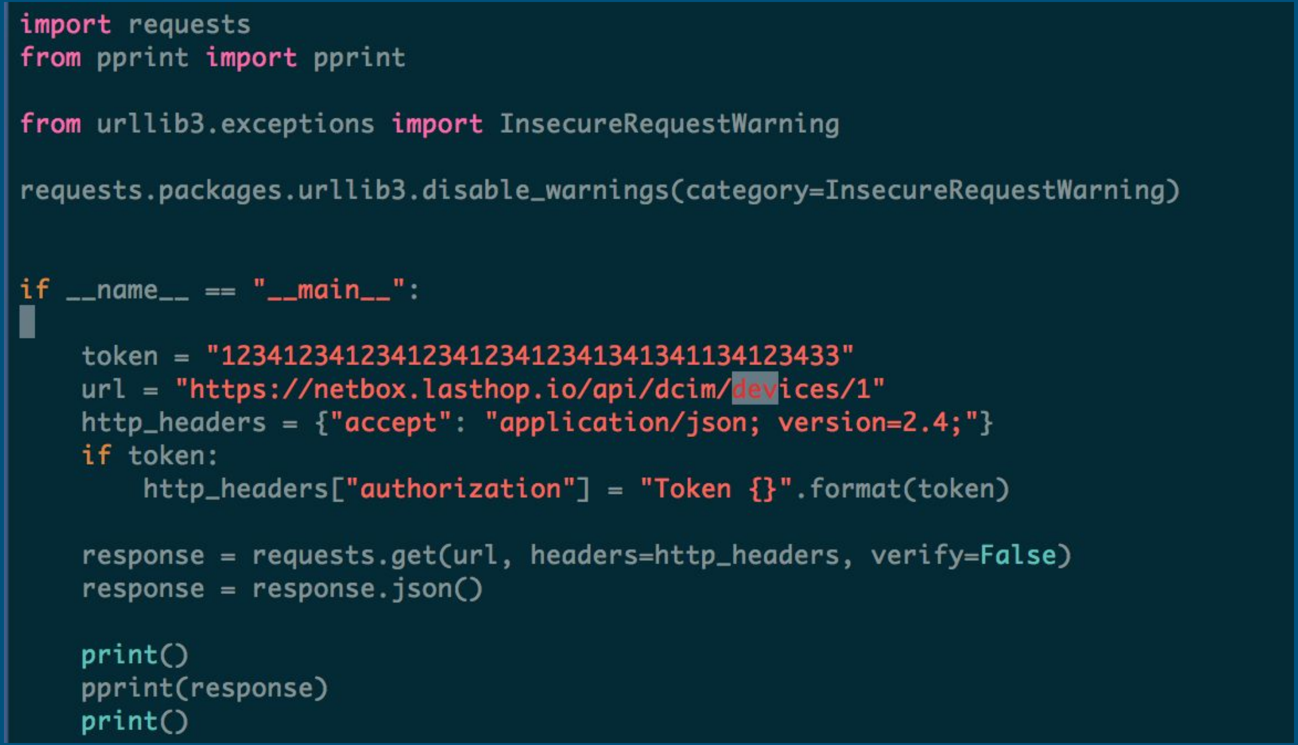
Exercises:
./day3/restapi_ex1.txt



```python
import requests
from pprint import pprint

from urllib3.exceptions import InsecureRequestWarning

requests.packages.urllib3.disable_warnings(category=InsecureRequestWarning)


if __name__ == "__main__":

    url = "https://netbox.lasthop.io/api/dcim/"
    # url = "https://api.github.com/"
    http_headers = {"accept": "application/json; version=2.4;"}
    response = requests.get(url, headers=http_headers, verify=False)
    response = response.json()

    print()
    pprint(response)
    print()
```

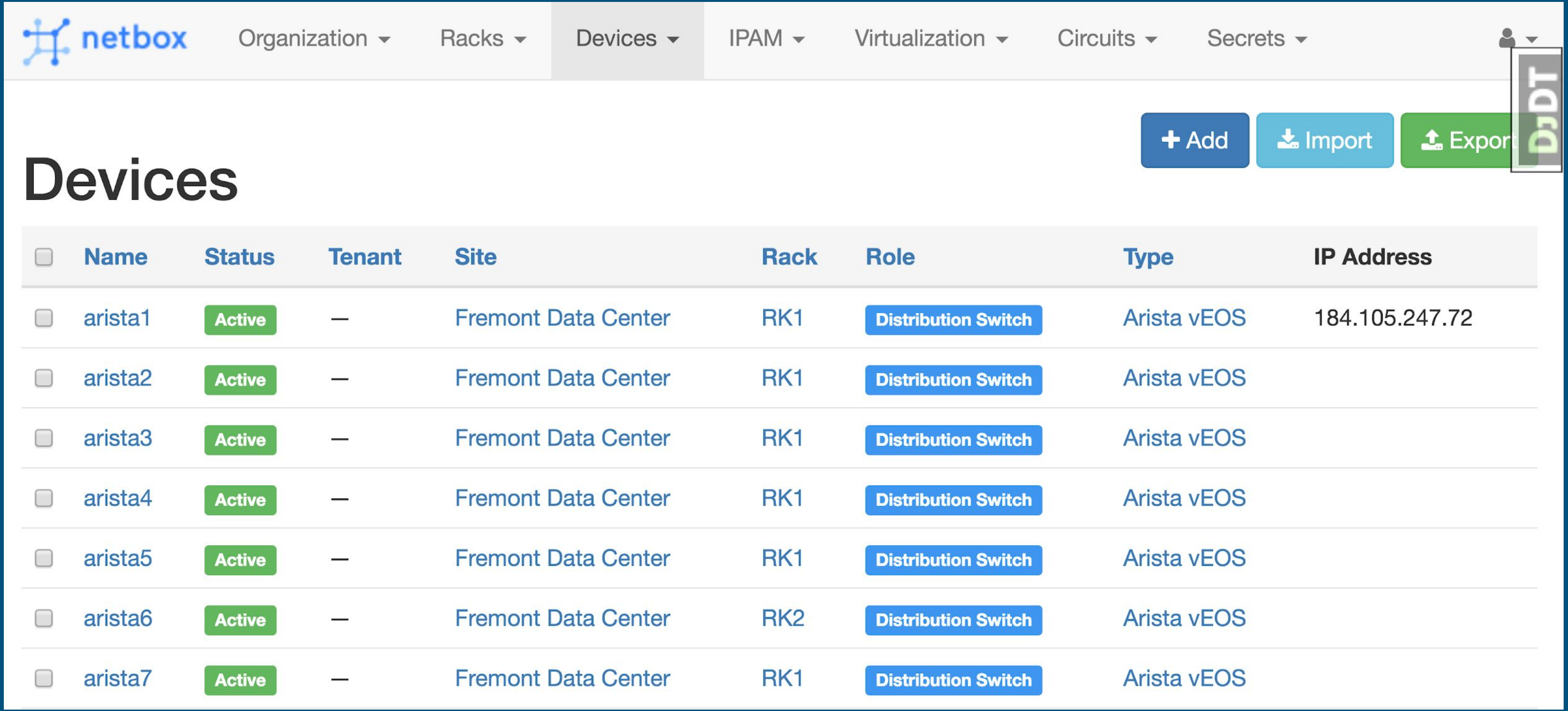
Authentication

Exercises:
./day3/restapi_ex2.txt



```
import requests
from pprint import pprint

from urllib3.exceptions import InsecureRequestWarning

requests.packages.urllib3.disable_warnings(category=InsecureRequestWarning)


if __name__ == "__main__":

    token = "12341234123412341234123413413411341234 33"
    url = "https://netbox.lasthop.io/api/dcim/devices/1"
    http_headers = {"accept": "application/json; version=2.4;"}
    if token:
        http_headers["authorization"] = "Token {}".format(token)

    response = requests.get(url, headers=http_headers, verify=False)
    response = response.json()

    print()
    pprint(response)
    print()
```

REST API - Adding a device using HTTP POST



Devices

Name	Status	Tenant	Site	Rack	Role	Type	IP Address
arista1	Active	—	Fremont Data Center	RK1	Distribution Switch	Arista vEOS	184.105.247.72
arista2	Active	—	Fremont Data Center	RK1	Distribution Switch	Arista vEOS	
arista3	Active	—	Fremont Data Center	RK1	Distribution Switch	Arista vEOS	
arista4	Active	—	Fremont Data Center	RK1	Distribution Switch	Arista vEOS	
arista5	Active	—	Fremont Data Center	RK1	Distribution Switch	Arista vEOS	
arista6	Active	—	Fremont Data Center	RK2	Distribution Switch	Arista vEOS	
arista7	Active	—	Fremont Data Center	RK1	Distribution Switch	Arista vEOS	

REST API - Modify (put) and Delete

```
response = requests.put(
    url, headers=http_headers, data=json.dumps(arista6), verify=False
)
```

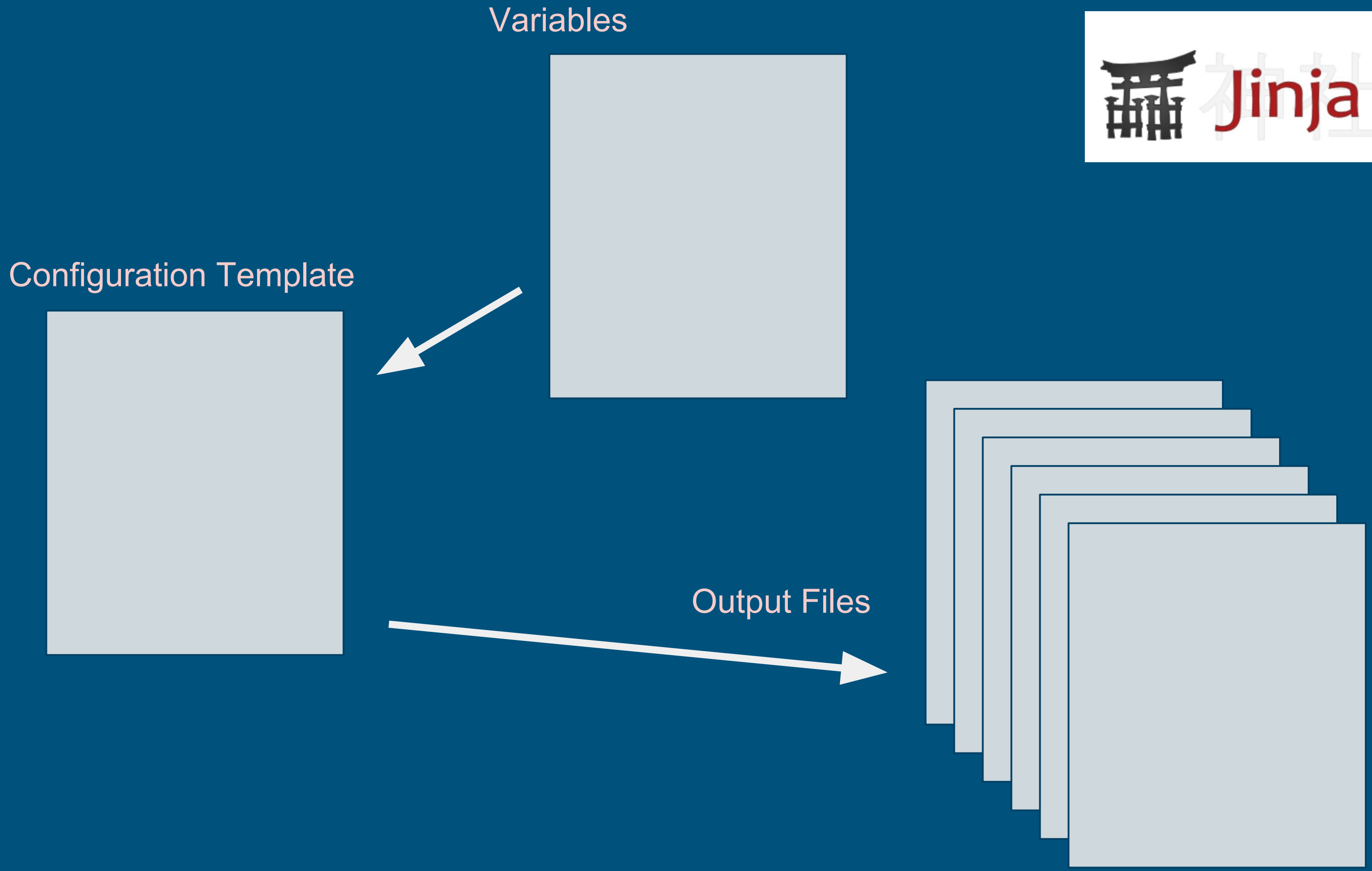
```
response = requests.delete(url, headers=http_headers, verify=False)
```

REST API

1. Determine if there is an existing Python library available.
2. Determine how to accomplish authentication.
3. Determine how to do information retrieval.
4. Determine how to create and modify objects.
5. Start building up abstractions to accomplish your goals.

Day4 Schedule

- Jinja2 Templating
- Pulling data from a CSV file
- NAPALM
- Writing Reusable Code
- TextFSM
- Concurrency: Threads and Processes
- Introduction to Nornir
- Unit Testing


Variables
Configuration Template
Output Files

Jinja2 Templating

```
import jinja2

my_dict = {'a': 'whatever'}

my_template = '''
Some text
of something
{{ a }}
something
'''

t = jinja2.Template(my_template)
print(t.render(my_dict))
```

Reference Material in:

{{ github_repo }}/jinja2_example/jinja2_simple.py

{{ github_repo }}/jinja2_example/jinja2_bgp.py

Jinja2 Templating - Loading Template from a File

```
import jinja2

template_file = 'bgp_config.j2'
with open(template_file) as f:
    bgp_template = f.read()

my_vars = {
    'peer_as': '22',
    'neighbor1': '10.10.10.2',
    'neighbor2': '10.10.10.99',
    'neighbor3': '10.10.10.220',
}

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

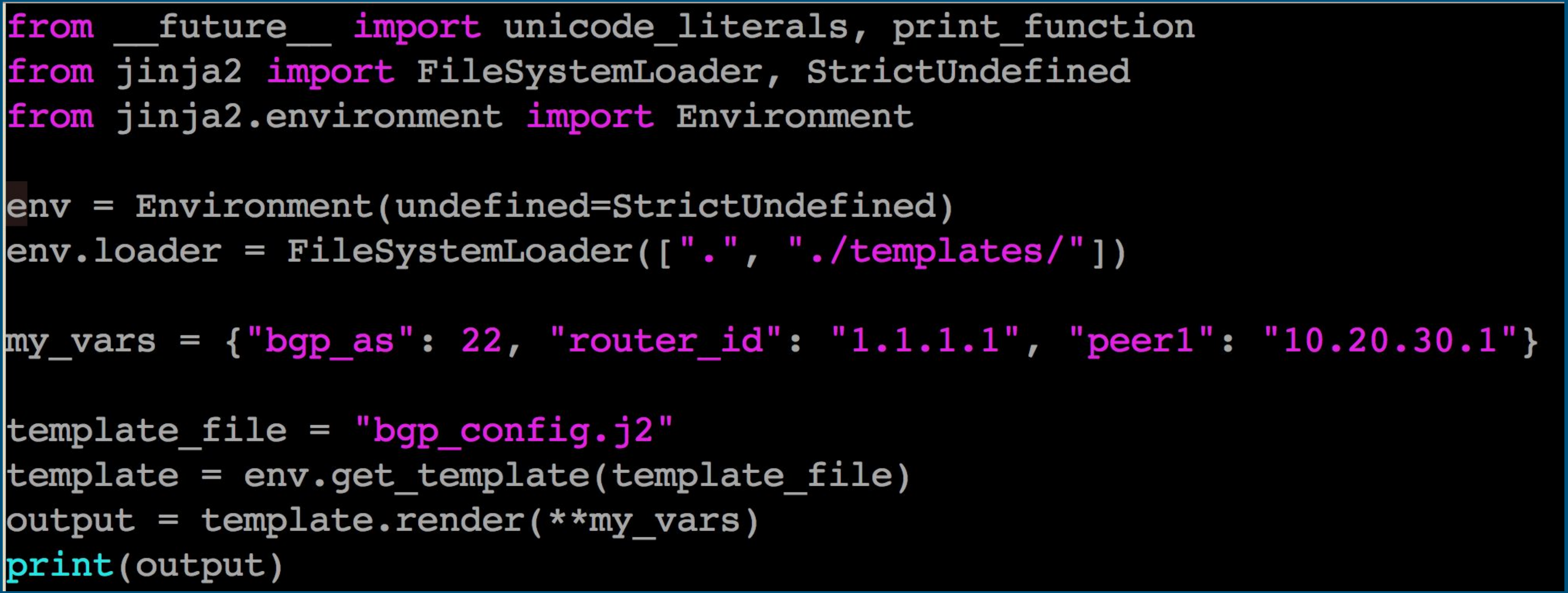
Reference Material in:
    {{ github_repo }}/jinja2_example/jinja2_bgp_file.py

Exercises:
./day4/jinja2_ex1.txt

Reference Material in:
{{ github_repo }}/jinja2_example/jinja2_env.py

Jinja2 Template - Environment

Exercises:
./day4/jinja2_ex2.txt



```
from __future__ import unicode_literals, print_function
from jinja2 import FileSystemLoader, StrictUndefined
from jinja2.environment import Environment

env = Environment(undefined=StrictUndefined)
env.loader = FileSystemLoader([".", "./templates/"])

my_vars = {"bgp_as": 22, "router_id": "1.1.1.1", "peer1": "10.20.30.1"}

template_file = "bgp_config.j2"
template = env.get_template(template_file)
output = template.render(**my_vars)
print(output)
```

Exercises:
./day4/jinja2_ex3.txt

Jinja2 Templating - Conditionals

```
{% if SNMPv3 %}
access-list 98 remark *** SNMP ***
access-list 98 permit any
!
snmp-server view VIEWSTD iso included
snmp-server group READONLY v3 priv read VIEWSTD access 98
snmp-server user pysnmp READONLY v3 auth sha auth_key priv aes 128
encrypt_key
{% endif %}
```

Jinja2 Templating - Loops

```
protocols {
    bgp {
        group external-peers {
            type external;
            {% for neighbor_ip, neighbor_as in my_list %}
                neighbor {{ neighbor_ip }} {
                    peer-as {{ neighbor_as }};
                }
            {% endfor %}
        }
    }
}
```

Reference Material in:
{{ github_repo }}/jinja2_example/jinja2_bgp_loop.py

Jinja2 - Other Topics

- Jinja2 Whitespace Stripping
- Jinja2 Create Variables
- Jinja2 Filters
- Jinja2 Macros
- Jinja2 Includes / Hierarchy

CSV Examples

```
device_name,device_type,host,username,password
pynet-rtr1,cisco_ios,184.105.247.70,pyclass,my_pass
pynet-rtr2,cisco_ios,184.105.247.71,pyclass,my_pass
------------------------------------
file_name = 'test_net_devices.csv'
with open(file_name) as f:
    read_csv = csv.DictReader(f)
    for entry in read_csv:
        print(entry)
```

Reference Material in:
{{ github_repo }}/csv_example

Exercises:
./day3/csv_ex1.txt

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.

Reference Material in:
{{ github_repo }}/napalm_example

NAPALM Vendors

CORE
Arista EOS
Cisco IOS
Cisco IOS-XR
Cisco NX-OS
Juniper Junos

NAPALM Community Drivers

https://github.com/napalm-automation-community

NAPALM-Salt Integration

NAPALM-Ansible Integration

NAPALM Getters

Exercises:
./day4/napalm_ex1.txt
./day4/napalm_ex2.txt

https://napalm.readthedocs.io/en/latest/support/#getters-support-matrix

get_arp_table
get_bgp_config
get_bgp_neighbors
get_bgp_neighbors_detail
get_config
get_environment
get_facts
get_interfaces
get_interfaces_counters
get_interfaces_ip
get_ipv6_neighbors_table
get_lldp_neighbors
get_lldp_neighbors_detail
get_mac_address_table
get_network_instances
get_ntp_peers
get_ntp_servers
get_ntp_stats
get_optics
get_probes_config
get_probes_results
get_route_to
get_snmp_information
get_users
is_alive
ping
traceroute

NAPALM Config Operations

device.load_merge_candidate()
device.load_replace_candidate()

device.compare_config()
device.discard_config()

device.commit_config()

device.rollback()

Exercises:
./day4/napalm_ex3.txt

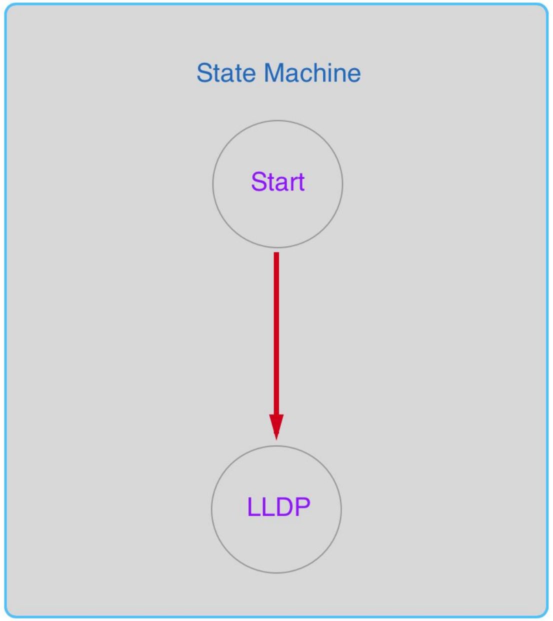
Writing Reusable Code

- Functions/Classes
- Code Structure
- Linting Tools
- Unit Testing
- Systems Testing
- CI-CD

TextFSM - The Problem

```
$ cat show_ip_bgp.txt
BGP table version is 17889841, local router ID is 128.223.51.103
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *   1.0.0.0/24       208.74.64.40                           0 19214 174 13335 i
 *                    162.251.163.2                          0 53767 13335 i
 *                    94.142.247.3             0             0 8283 13335 i
 *                    212.66.96.126                          0 20912 13335 i
```

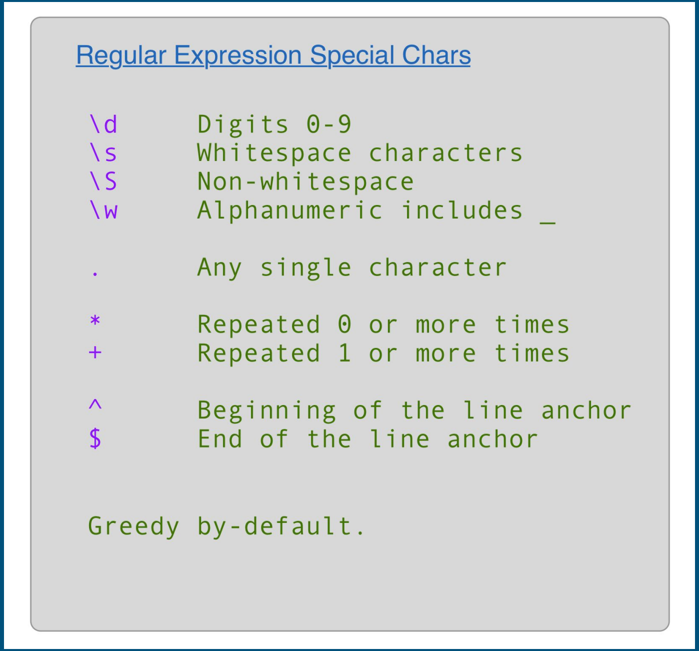
TextFSM



TextFSM File

```
# Define your fields to extract
Value VAR_NAME (regex_pattern)
Value VAR_NAME (regex_pattern)
Value VAR_NAME (regex_pattern)


# Start of the FSM
Start
  ^Device.*ID -> LLDP

LLDP
  ^${VAR_NAME}.* -> Record

# Implicit EOF and Record
# EOF
```

TextFSM - A minimum set of regular expressions



Regular Expression Special Chars

```
\d     Digits 0-9
\s     Whitespace characters
\S     Non-whitespace
\w     Alphanumeric includes _

.      Any single character

*      Repeated 0 or more times
+      Repeated 1 or more times

^      Beginning of the line anchor
$      End of the line anchor


Greedy by-default.
```

TextFSM - Example

Reference Material in:
{{ github_repo }}/textfsm

- Variables > Start > State Transition
- Implicit EOF
- Installing TextFSM
- Installing ntc-templates
- Coupling TextFSM with Netmiko

Threads/Processes

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

Reference Material in:
{{ github_repo }}/threads_procs

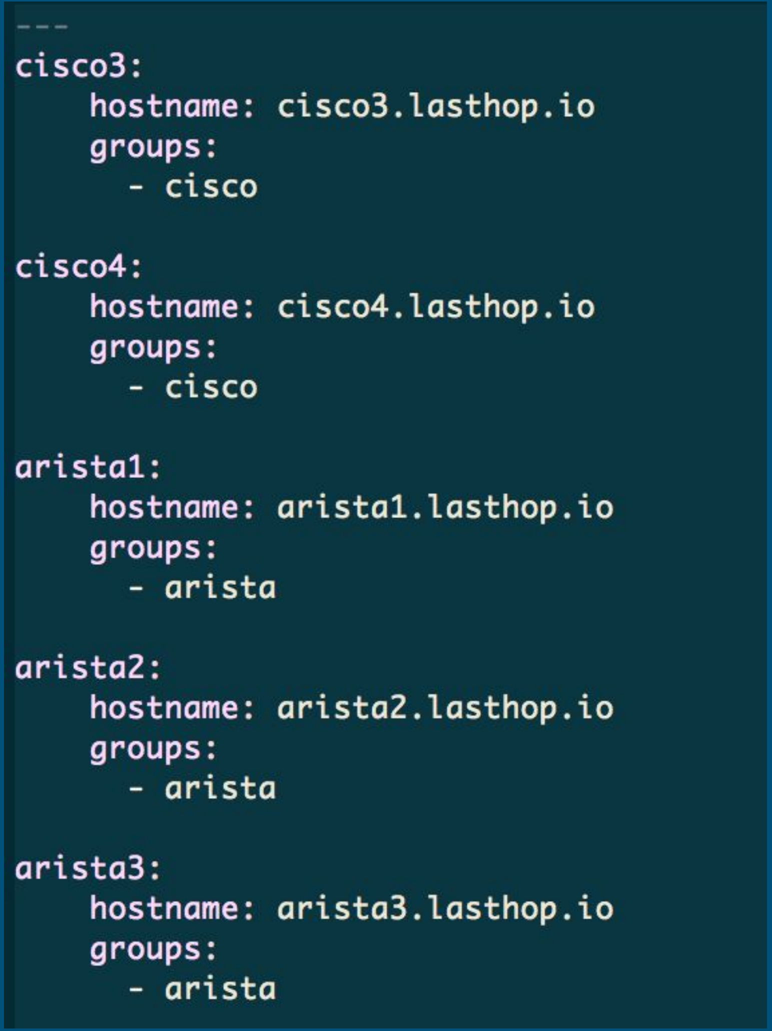
Exercises:
./day4/threads_ex1.txt

Introduction to Nornir - Why?

1. Systematically handle inventory management inventory in a modular way.
2. Handle concurrency.

Nornir - Simple Inventory

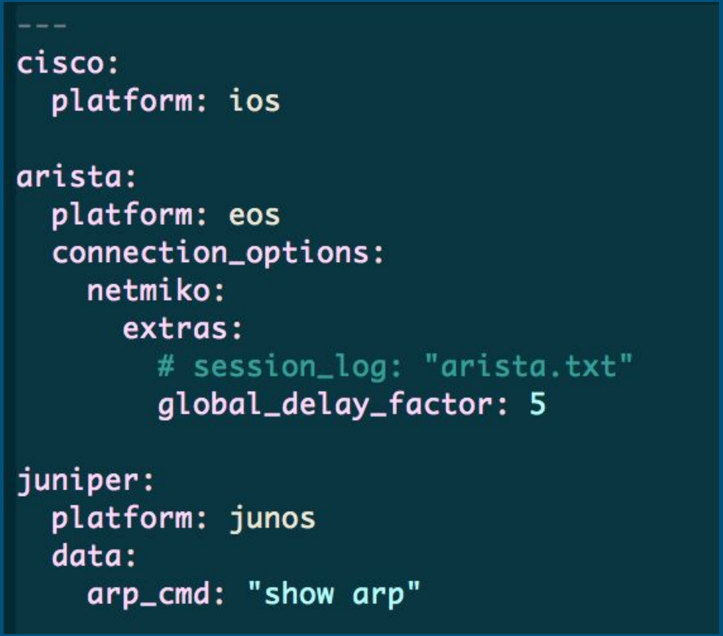
hosts.yaml file



```
---
cisco3:
    hostname: cisco3.lasthop.io
    groups:
      - cisco

cisco4:
    hostname: cisco4.lasthop.io
    groups:
      - cisco

arista1:
    hostname: arista1.lasthop.io
    groups:
      - arista

arista2:
    hostname: arista2.lasthop.io
    groups:
      - arista

arista3:
    hostname: arista3.lasthop.io
    groups:
      - arista
```

Nornir - Simple Inventory

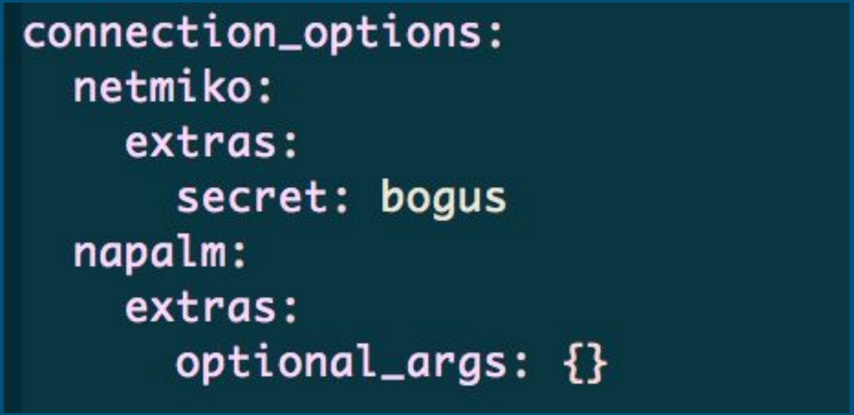
groups.yaml file



```yaml
---
cisco:
  platform: ios

arista:
  platform: eos
  connection_options:
    netmiko:
      extras:
        # session_log: "arista.txt"
        global_delay_factor: 5

juniper:
  platform: junos
  data:
    arp_cmd: "show arp"
```

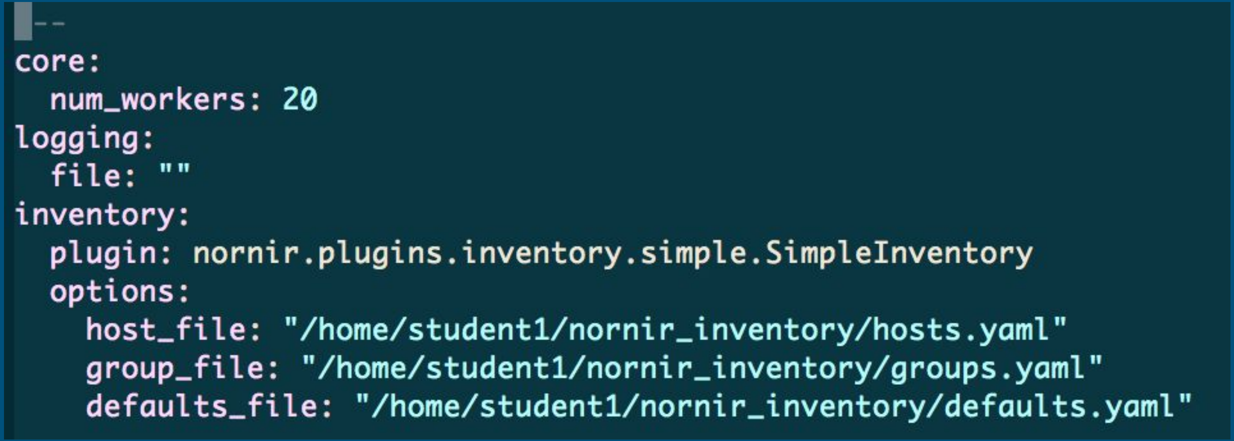
Nornir - Simple Inventory

defaults.yaml file



```
connection_options:
  netmiko:
    extras:
      secret: bogus
  napalm:
    extras:
      optional_args: {}
```

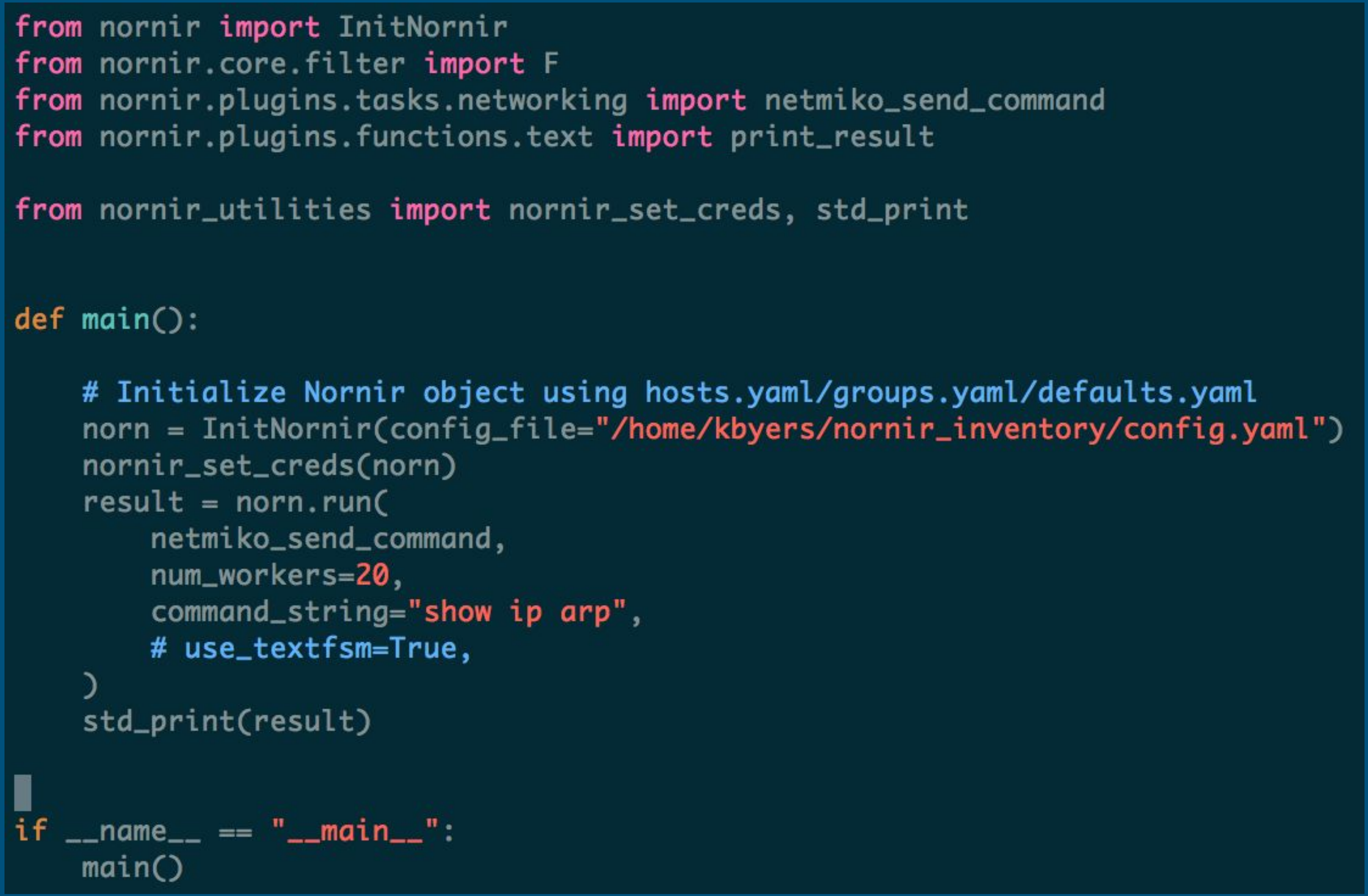
Nornir - Config File



```
---
core:
  num_workers: 20
logging:
  file: ""
inventory:
  plugin: nornir.plugins.inventory.simple.SimpleInventory
  options:
    host_file: "/home/student1/nornir_inventory/hosts.yaml"
    group_file: "/home/student1/nornir_inventory/groups.yaml"
    defaults_file: "/home/student1/nornir_inventory/defaults.yaml"
```

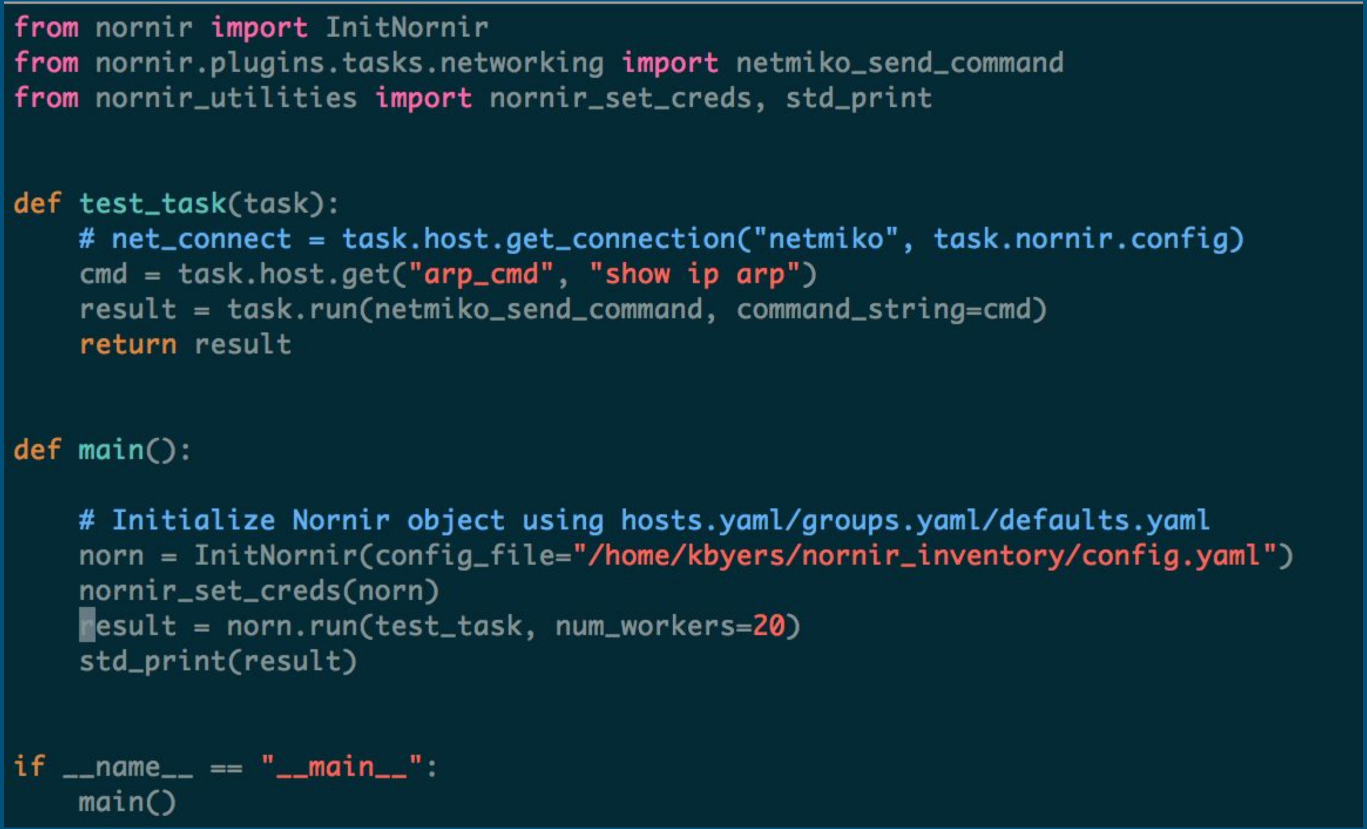
config.yaml file

Nornir - Example



```python
from nornir import InitNornir
from nornir.core.filter import F
from nornir.plugins.tasks.networking import netmiko_send_command
from nornir.plugins.functions.text import print_result

from nornir_utilities import nornir_set_creds, std_print


def main():

    # Initialize Nornir object using hosts.yaml/groups.yaml/defaults.yaml
    norn = InitNornir(config_file="/home/kbyers/nornir_inventory/config.yaml")
    nornir_set_creds(norn)
    result = norn.run(
        netmiko_send_command,
        num_workers=20,
        command_string="show ip arp",
        # use_textfsm=True,
    )
    std_print(result)


if __name__ == "__main__":
    main()
```

Nornir - Subtask



```
from nornir import InitNornir
from nornir.plugins.tasks.networking import netmiko_send_command
from nornir_utilities import nornir_set_creds, std_print


def test_task(task):
    # net_connect = task.host.get_connection("netmiko", task.nornir.config)
    cmd = task.host.get("arp_cmd", "show ip arp")
    result = task.run(netmiko_send_command, command_string=cmd)
    return result


def main():

    # Initialize Nornir object using hosts.yaml/groups.yaml/defaults.yaml
    norn = InitNornir(config_file="/home/kbyers/nornir_inventory/config.yaml")
    nornir_set_creds(norn)
    result = norn.run(test_task, num_workers=20)
    std_print(result)


if __name__ == "__main__":
    main()
```

Unit Testing

```
import pytest

# Functions
def func(x):
    return x + 1

# Tests
def test_answer():
    assert func(3) == 4
```

Reference Material in:
{{ github_repo }}/unittest_example

Unit Testing

```
py.test -s -v test_simple.py
============================= test session starts ==============================
platform linux -- Python 3.5.1, pytest-3.2.3, py-1.4.34, pluggy-0.4.0 --
/home/kbyers/VENV/py35_venv/bin/python35
cachedir: .cache
rootdir: /home/kbyers/pynet-ons-oct17/unittest_example, inifile:
plugins: pylama-7.4.3
collected 1 item

test_simple.py::test_answer PASSED
```

Creating a fixture

```
@pytest.fixture(scope="module")
def netmiko_connect():
    cisco1 = {
        'device_type': 'cisco_ios',
        'ip':   '184.105.247.70',
        'username': 'pyclass',
        'password': getpass()
    }
    return ConnectHandler(**cisco1)
```

Using a fixture

Exercises:
./day4/unittest_ex1.txt
./day4/unittest_ex2.txt

```
def test_prompt(netmiko_connect):
    print(netmiko_connect.find_prompt())
    assert netmiko_connect.find_prompt() == 'pynet-rtr1#'

def test_show_version(netmiko_connect):
    output = netmiko_connect.send_command("show version")
    assert 'Configuration register is 0x2102' in output
```

Continuous Integration using Travis CI

Define a .travis.yml file in your repository.

Link Travis-CI to GitHub account

Add linting

Add automated testing

```
---
language: python
python:
  - "2.7"
  - "3.5"
  - "3.6"
install:
  - pip install -r requirements.txt
script:
  - pylama travis_test/
  - py.test tests/
```

The end...

Questions?

ktbyers@twb-tech.com
Twitter: @kirkbyers