PYTHON

FOR NETWORK ENGINEERS

Onsite Training Session
July 2020

Day2

1. Netmiko
2. pytest Fixtures
3. Libraries
4. sys.path and PYTHONPATH
5. pip and virtual environments
6. Data Serialization: YAML and JSON
7. Handling Complex Data Structures.
8. Juniper PyEZ Views

Flickr: au_tiger01

Netmiko

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

https://github.com/ktbyers/netmiko

Netmiko Vendors

- Currently (very) roughly 74 different platforms supported by Netmiko.

- Three different categories of supported platform (regularly tested, limited testing, experimental).

https://ktbyers.github.io/netmiko/PLATFORMS.html

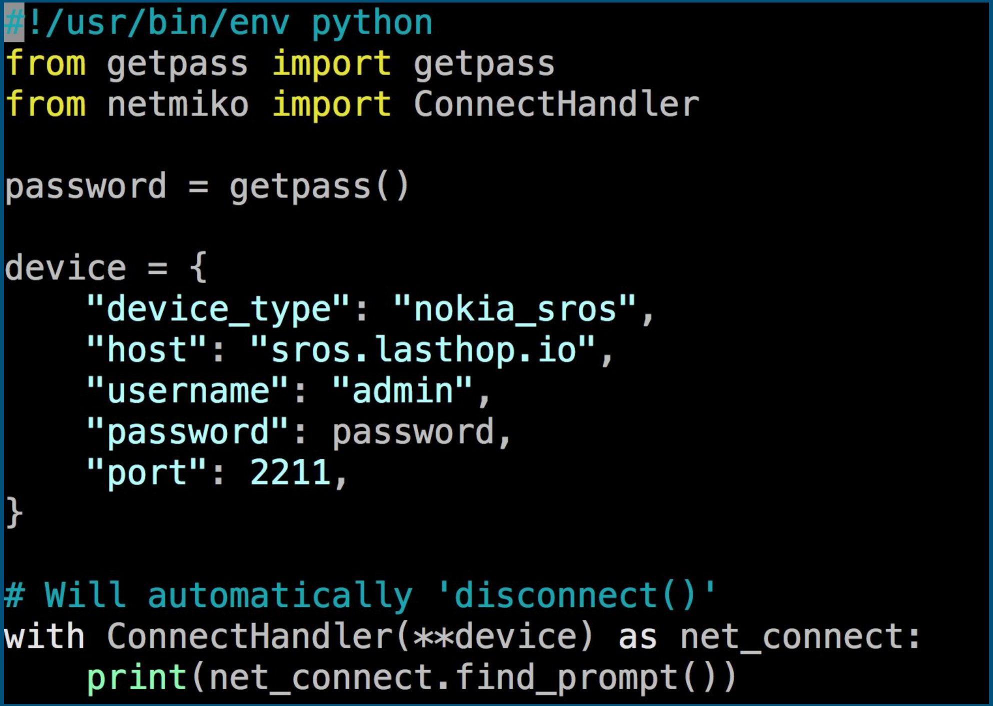
Regularly tested
Arista vEOS
Cisco ASA
Cisco IOS
Cisco IOS-XE
Cisco IOS-XR

Regularly tested
Cisco NX-OS
Cisco SG300
HP ProCurve
Juniper Junos
Linux

Key Netmiko Methods

.send_command()	Send command, use pattern matching to know when “done”
.send_command_timing()	Send command, use timing to know when “done”
.send_config_set()	Send list of configuration commands
.send_config_from_file()	Send configuration commands from a file
.save_config()	… save the config
.commit()	Commit configuration (for specific platforms)
.enable()	Enter “enable”/privilege mode
.disconnect()	Close connection
.write_channel()	Write to channel directly (bypass Netmiko prompt searching/timing)
.read_channel()	Read directly from channel (bypass Netmiko prompt searching/timing)
FileTransfer Class	SCP files to/from devices

Netmiko example



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

password = getpass()

device = {
    "device_type": "nokia_sros",
    "host": "sros.lasthop.io",
    "username": "admin",
    "password": password,
    "port": 2211,
}

# Will automatically 'disconnect()'
with ConnectHandler(**device) as net_connect:
    print(net_connect.find_prompt())
```

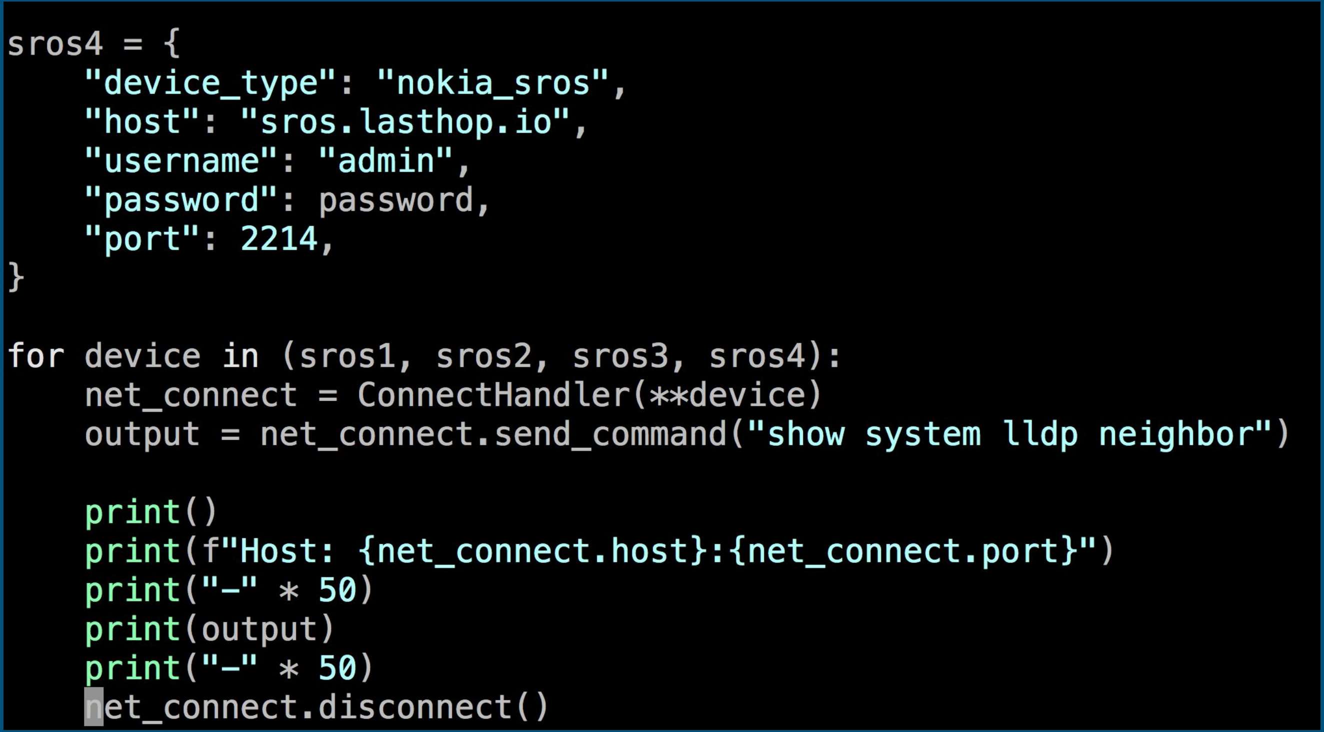
Reference Material in:

{{ github_repo }}/netmiko_example

Netmiko ‘show’ command

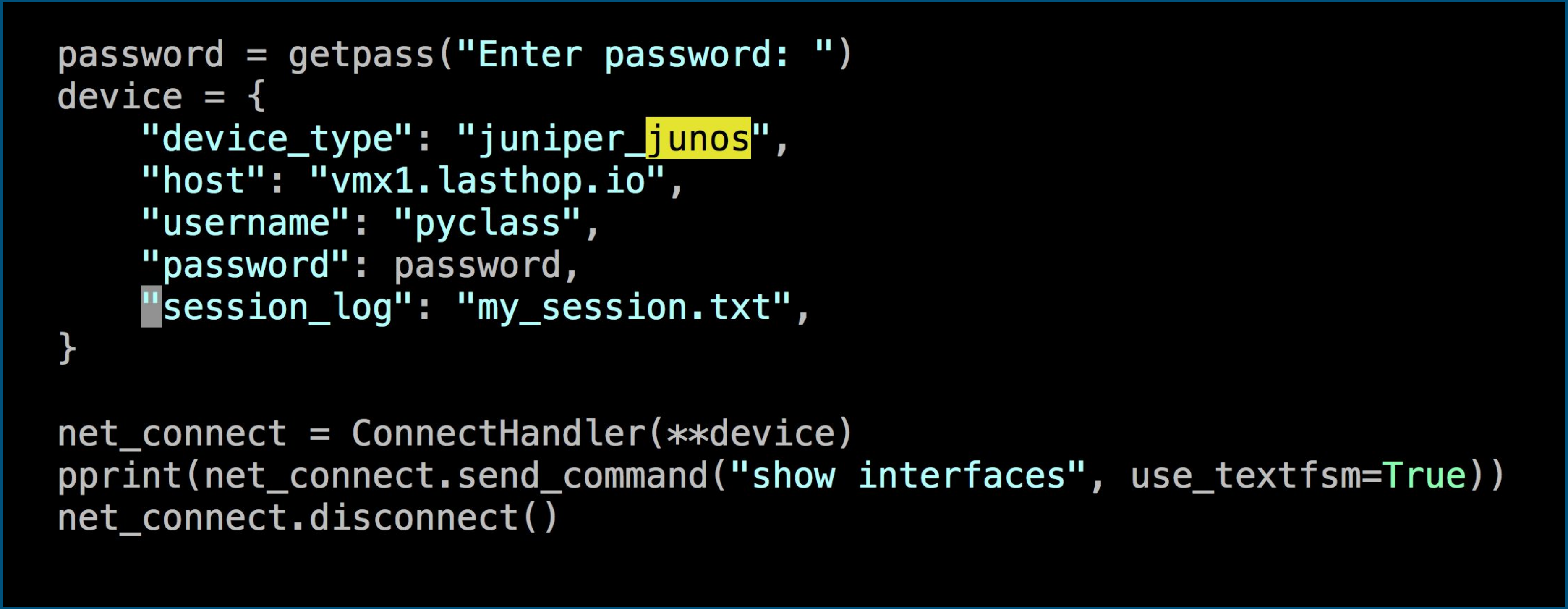
```
net_connect = ConnectHandler(**device)
output = net_connect.send_command("show system lldp neighbor")
net_connect.disconnect()

print("-" * 50)
print(output)
print("-" * 50)
```

Netmiko multiple devices



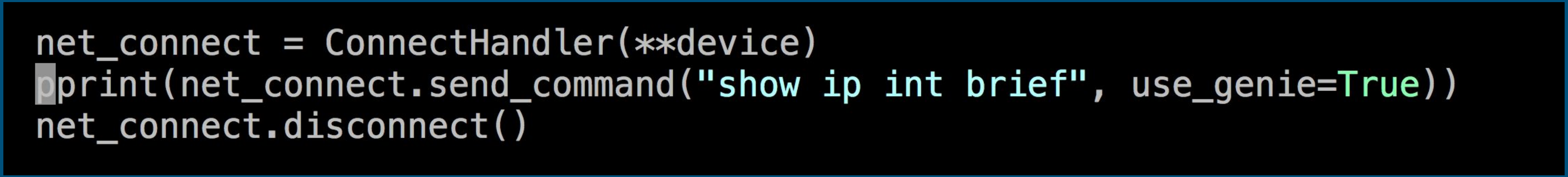
```
sros4 = {
    "device_type": "nokia_sros",
    "host": "sros.lasthop.io",
    "username": "admin",
    "password": password,
    "port": 2214,
}

for device in (sros1, sros2, sros3, sros4):
    net_connect = ConnectHandler(**device)
    output = net_connect.send_command("show system lldp neighbor")

    print()
    print(f"Host: {net_connect.host}:{net_connect.port}")
    print("-" * 50)
    print(output)
    print("-" * 50)
    net_connect.disconnect()
```

Netmiko and TextFSM



```
password = getpass("Enter password: ")
device = {
    "device_type": "juniper_junos",
    "host": "vmx1.lasthop.io",
    "username": "pyclass",
    "password": password,
    "session_log": "my_session.txt",
}

net_connect = ConnectHandler(**device)
pprint(net_connect.send_command("show interfaces", use_textfsm=True))
net_connect.disconnect()
```

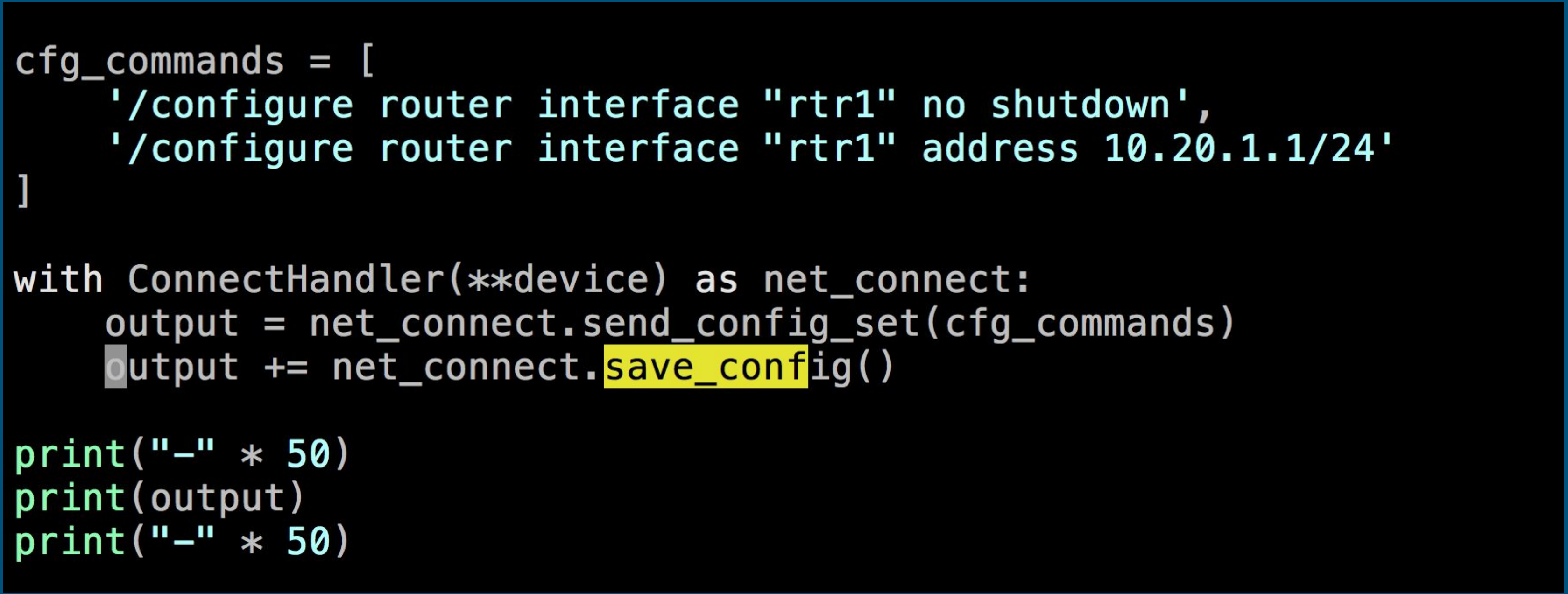
Netmiko and Genie



```
net_connect = ConnectHandler(**device)
pprint(net_connect.send_command("show ip int brief", use_genie=True))
net_connect.disconnect()
```

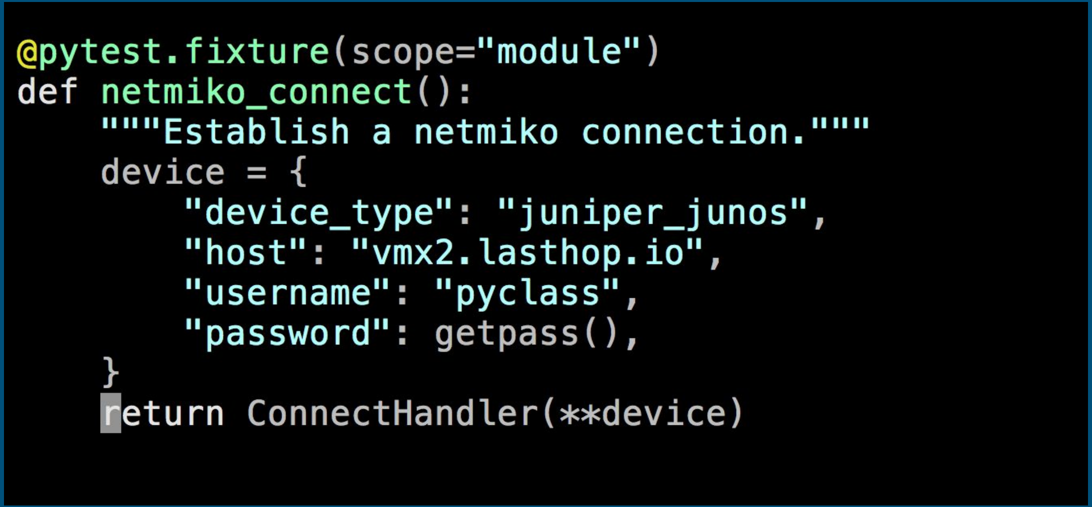
Netmiko Configuration

Exercises:
./day2/netmiko/netmiko_ex1.txt
./day2/netmiko/netmiko_ex1_test.txt
./day2/netmiko/netmiko_ex2.txt



```
cfg_commands = [
    '/configure router interface "rtr1" no shutdown',
    '/configure router interface "rtr1" address 10.20.1.1/24'
]

with ConnectHandler(**device) as net_connect:
    output = net_connect.send_config_set(cfg_commands)
    output += net_connect.save_config()

print("-" * 50)
print(output)
print("-" * 50)
```

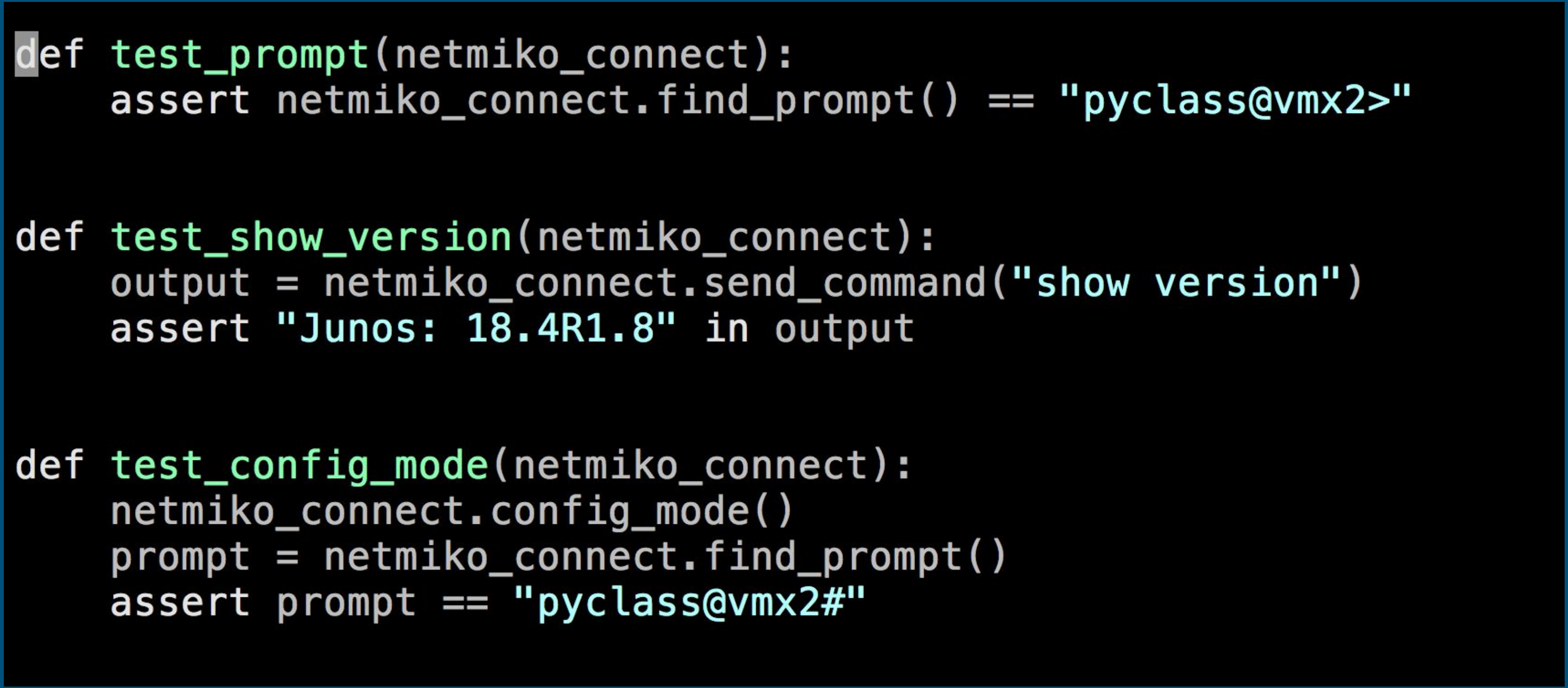
Creating a fixture



```
@pytest.fixture(scope="module")
def netmiko_connect():
    """Establish a netmiko connection."""
    device = {
        "device_type": "juniper_junos",
        "host": "vmx2.lasthop.io",
        "username": "pyclass",
        "password": getpass(),
    }
    return ConnectHandler(**device)
```

Reference Material in:
{{ github_repo }}/unittest_example/separate_fixture

Using a fixture



```
def test_prompt(netmiko_connect):
    assert netmiko_connect.find_prompt() == "pyclass@vmx2>"


def test_show_version(netmiko_connect):
    output = netmiko_connect.send_command("show version")
    assert "Junos: 18.4R1.8" in output


def test_config_mode(netmiko_connect):
    netmiko_connect.config_mode()
    prompt = netmiko_connect.find_prompt()
    assert prompt == "pyclass@vmx2#"
```

Libraries

import x

from x import y

sys.path

PYTHONPATH

Installing packages (pip)

Virtual Environments

Photo: Viva Vivanista (Flickr)

Exercises:
./day2/virt_env/venv_ex1.txt

Virtualenv

```
$ python36 -m venv test_venv

$ source test_venv/bin/activate
$ which python

~/VENV/test_venv/bin/python

$ pip list
Package    Version
---------- -------
pip        20.1.1
setuptools 40.6.2
```

```
$ deactivate

$ which python
/usr/bin/python
```

Data Serialization

Why do we need data serialization?

Characteristics of JSON

Characteristics of YAML

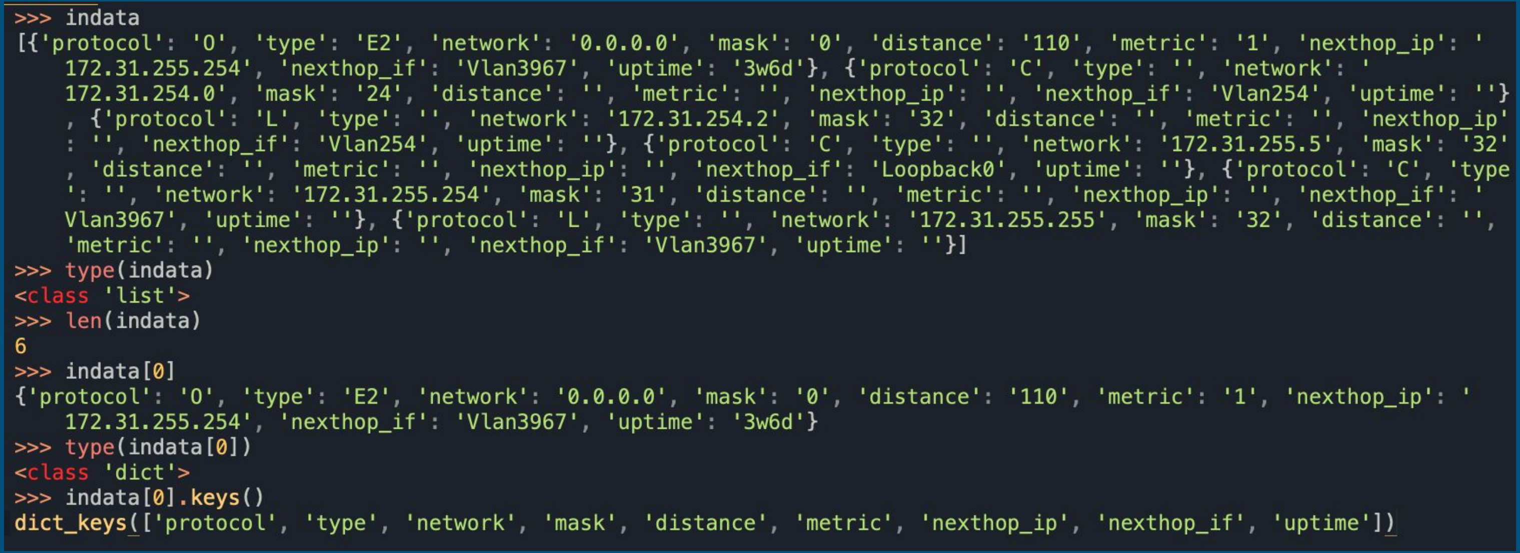
Reference Material in:
{{ github_repo }}/json_yaml

Exercises:
./day2/yaml/yaml_ex1.txt
./day2/yaml/yaml_ex2.txt

Exercises:
./day2/complex_data_struct/struct_ex1.txt

Complex Data Structures

1. Investigate layer by layer
2. Determine object type (list, dict, or ?)
3. Single or multiple elements?



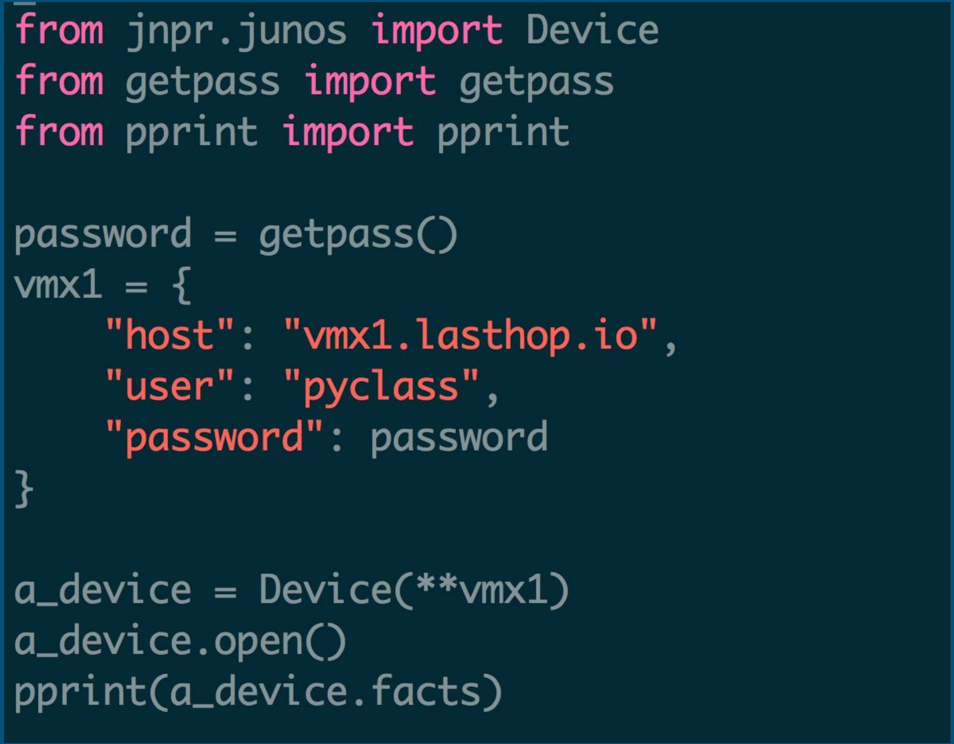
```
>>> indata
[{'protocol': 'O', 'type': 'E2', 'network': '0.0.0.0', 'mask': '0', 'distance': '110', 'metric': '1', 'nexthop_ip': '
    172.31.255.254', 'nexthop_if': 'Vlan3967', 'uptime': '3w6d'}, {'protocol': 'C', 'type': '', 'network': '
    172.31.254.0', 'mask': '24', 'distance': '', 'metric': '', 'nexthop_ip': '', 'nexthop_if': 'Vlan254', 'uptime': ''}
    , {'protocol': 'L', 'type': '', 'network': '172.31.254.2', 'mask': '32', 'distance': '', 'metric': '', 'nexthop_ip'
    : '', 'nexthop_if': 'Vlan254', 'uptime': ''}, {'protocol': 'C', 'type': '', 'network': '172.31.255.5', 'mask': '32'
    , 'distance': '', 'metric': '', 'nexthop_ip': '', 'nexthop_if': 'Loopback0', 'uptime': ''}, {'protocol': 'C', 'type
    ': '', 'network': '172.31.255.254', 'mask': '31', 'distance': '', 'metric': '', 'nexthop_ip': '', 'nexthop_if': '
    Vlan3967', 'uptime': ''}, {'protocol': 'L', 'type': '', 'network': '172.31.255.255', 'mask': '32', 'distance': '',
    'metric': '', 'nexthop_ip': '', 'nexthop_if': 'Vlan3967', 'uptime': ''}]
>>> type(indata)
<class 'list'>
>>> len(indata)
6
>>> indata[0]
{'protocol': 'O', 'type': 'E2', 'network': '0.0.0.0', 'mask': '0', 'distance': '110', 'metric': '1', 'nexthop_ip': '
    172.31.255.254', 'nexthop_if': 'Vlan3967', 'uptime': '3w6d'}
>>> type(indata[0])
<class 'dict'>
>>> indata[0].keys()
dict_keys(['protocol', 'type', 'network', 'mask', 'distance', 'metric', 'nexthop_ip', 'nexthop_if', 'uptime'])
```

Juniper, NETCONF, and PyEZ

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

Reference Material in:
{{ github_repo }}/jnpr_examples

PyEZ simple connect / facts

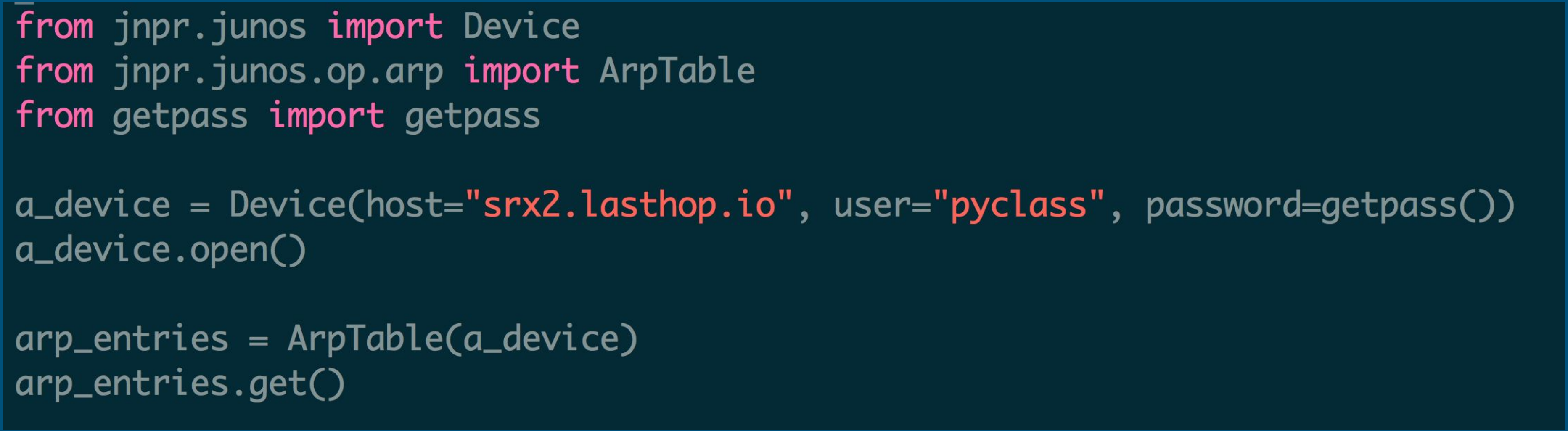


```python
from jnpr.junos import Device
from getpass import getpass
from pprint import pprint

password = getpass()
vmx1 = {
    "host": "vmx1.lasthop.io",
    "user": "pyclass",
    "password": password
}

a_device = Device(**vmx1)
a_device.open()
pprint(a_device.facts)
```

PYTHON
FOR NETWORK ENGINEERS

PyEZ table operations

Exercises:
./day2/jnpr/ex1.txt
./day2/jnpr/ex1_test.txt
./day2/jnpr/ex2.txt
./day2/jnpr/ex2_test.txt



```
from jnpr.junos import Device
from jnpr.junos.op.arp import ArpTable
from getpass import getpass

a_device = Device(host="srx2.lasthop.io", user="pyclass", password=getpass())
a_device.open()

arp_entries = ArpTable(a_device)
arp_entries.get()
```

Reference Material in:

{{ github_repo }}/jnpr_examples