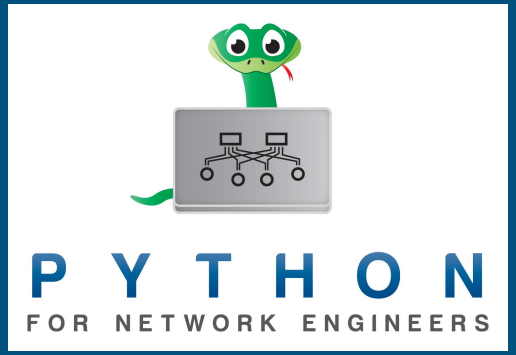
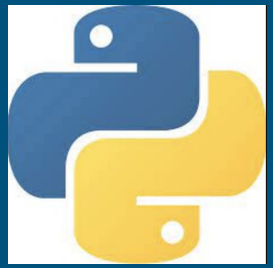




Live Training Session
December 2021

Day2

1. Netmiko (Part2)
2. Python Classes and Objects
3. sys.path and PYTHONPATH
4. Libraries/PIP/Virtualenv/Linting
5. Parsers - TextFSM / Genie (optional)
6. Serialization Protocols: YAML and JSON
7. Handling Complex Data Structures
8. NETCONF
9. Juniper PyEZ Views

Flickr: au_tiger01

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)
print(net_connect.send_command("show interfaces", use_textfsm=True))
net_connect.disconnect()
```

Netmiko and Genie


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

Exercises:
./day2/netmiko/netmiko_ex2.txt

Netmiko Configuration

```
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)
```

Classes and Objects

```
class Server:
    def __init__(self, hostname, username, password):
        self.hostname = hostname
        self.username = username
        self.password = password

    def test_method(self):
        print(f"Device is: {self.hostname}")
        print(f"Username is: {self.username}")


svr1 = Server("test.domain.com", "admin", "passw")
svr1.test_method()
```

1. What is a class?
2. When would you want to create and use a class?
3. How do you create a class?
   a. What is the purpose of dunder-init?
   b. What is the meaning of “self”?
   c. How do you create object attributes?
4. How do you create methods?
5. How do you create objects (instances of the class)?
6. How do you call methods?

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

sys.path and $PYTHONPATH

```
import sys
from rich import print

print(sys.path)
```

```
[
    '/home/ktbyers/pynet-ons-dec21/day2',
    '/home/ktbyers/python-libs',
    '/home/ktbyers/DJANGOX/djproject',
    '/usr/lib64/python37.zip',
    '/usr/lib64/python3.7',
    '/usr/lib64/python3.7/lib-dynload',
    '/home/ktbyers/VENV/py3_venv/lib64/python3.7/site-packages',
    '/home/ktbyers/VENV/py3_venv/lib/python3.7/site-packages'
]
```

How does Python locate other Python Libraries?
Where does Python even look?

```
# Modify PYTHONPATH to get extra libraries
export PYTHONPATH=~/python-libs
export PYTHONPATH=$PYTHONPATH:~/DJANGOX/djproject/
```

Libraries

```
import sys
from rich import print
from netmiko import ConnectHandler
```

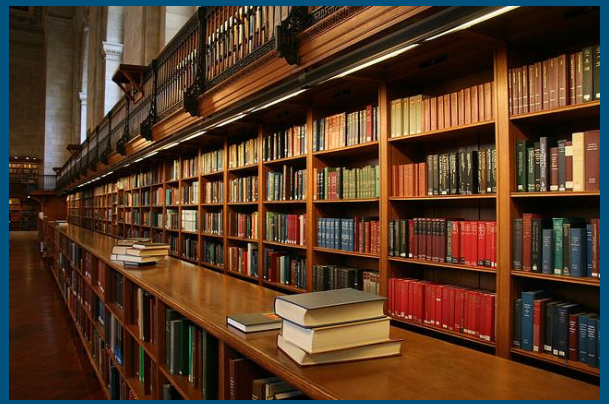


Photo: Viva Vivanista (Flickr)

PIP = Package Installer for Python

$ pip list

$ pip list | grep netmiko

$ pip uninstall netmiko

$ pip install netmiko==3.4.0

$ pip freeze

$ pip install -r ./requirements.txt

$ pip install -e .

pypi = Python Package Index

Exercises:
./day2/virt_env/venv_ex1.txt

Virtualenv

```
kbyers@pydev1 ~/VENV
$ python3.9 -m venv test_venv

kbyers@pydev1 ~/VENV
$ source test_venv/bin/activate
```

```
[test_venv] kbyers@pydev1 ~/VENV
$ pip list
Package    Version
---------- -------
pip        21.3.1
setuptools 56.0.0
```

```
[test_venv] kbyers@pydev1 ~/VENV
$ python --version
Python 3.9.5

[test_venv] kbyers@pydev1 ~/VENV
$ which python
~/VENV/test_venv/bin/python

[test_venv] kbyers@pydev1 ~/VENV
$ python --version
Python 3.9.5

[test_venv] kbyers@pydev1 ~/VENV
$ deactivate

kbyers@pydev1 ~/VENV
$ which python
/usr/bin/python
```

Examples:
{{ github_repo }}/linters

Python Linters

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

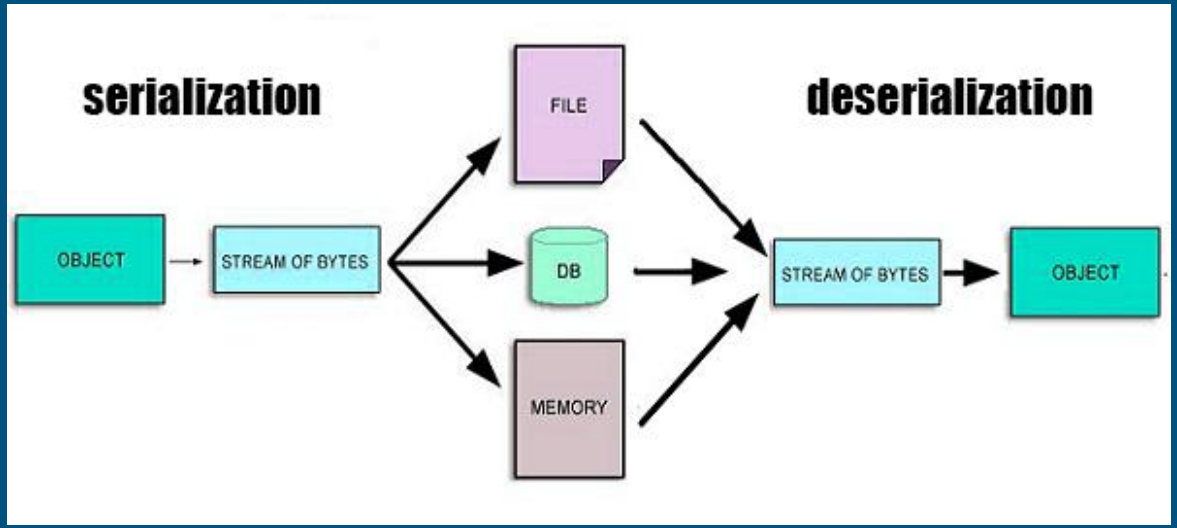
Auto formatting with Python Black

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

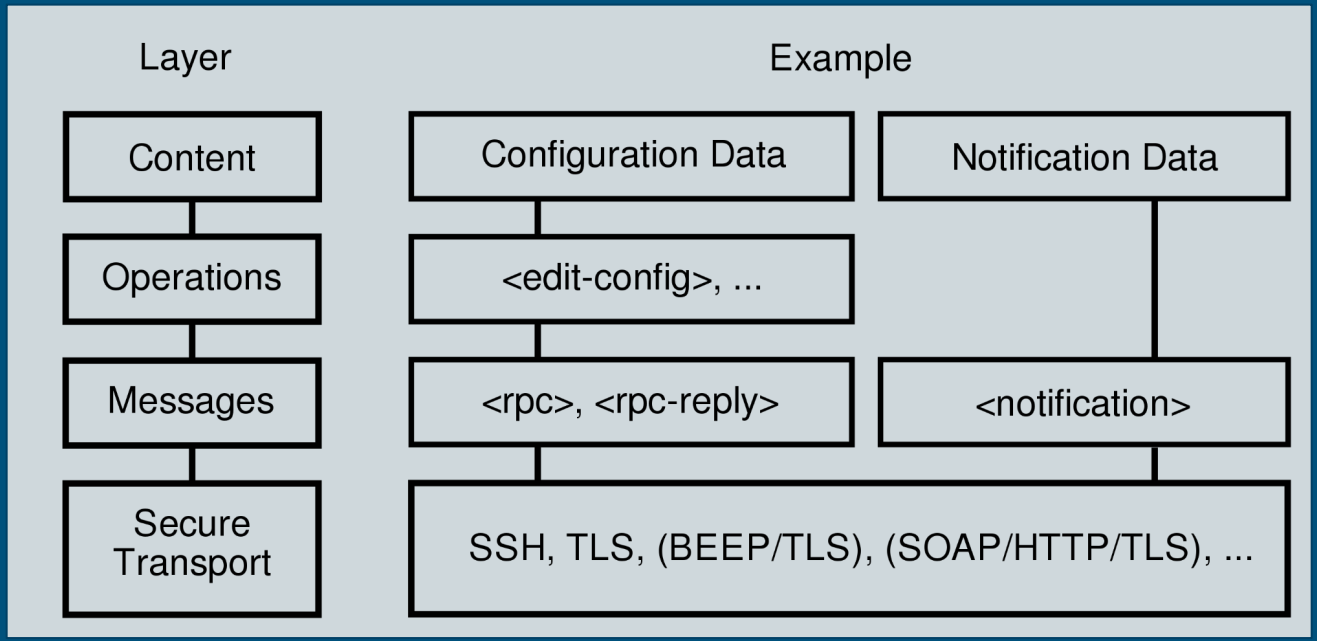
Complex Data Structures

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'])
```

NETCONF

Original RFC: RFC 4741 (2006)
Updated RFC: RFC 6241 (2011)



Juniper and PyEZ

- PyEZ
- PyEZ get operations
- PyEZ config operations

Reference Material in:

{{ github_repo }}/jnpr_examples

PyEZ simple connect / facts

```
from jnpr.junos import Device
from getpass import getpass
from rich import print

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

a_device = Device(**vmx1)
a_device.open()
print(dict(a_device.facts))
```

PYTHON
FOR NETWORK ENGINEERS

JUNIPER NETWORKS

PyEZ table operations

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

```
from jnpr.junos import Device
from jnpr.junos.op.arp import ArpTable
from rich import print
from getpass import getpass
import pdbr  # noqa

a_device = Device(
    host="vmx1.lasthop.io",
    user="pyclass",
    password=getpass()
)
a_device.open()

# pdbr.set_trace()
arp_entries = ArpTable(a_device)
arp_entries.get()
for k, v in arp_entries.items():
    print(k)
    print(v)
```

Reference Material in:
{{ github_repo }}/jnpr_examples

Review Exercise

Exercises:
./day2/exercise_review/review_ex1.txt

1. Load the ~/.netmiko.yml and use this to connect with Netmiko to all of the devices in the lab environment.

   For each device in the ~/.netmiko.yml file use Netmiko to print out the device's prompt.

PYTHON
FOR NETWORK ENGINEERS