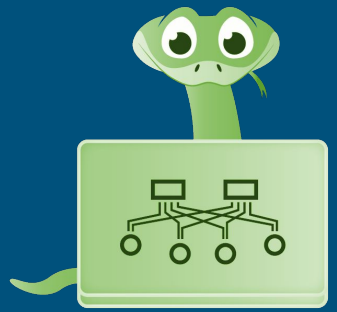


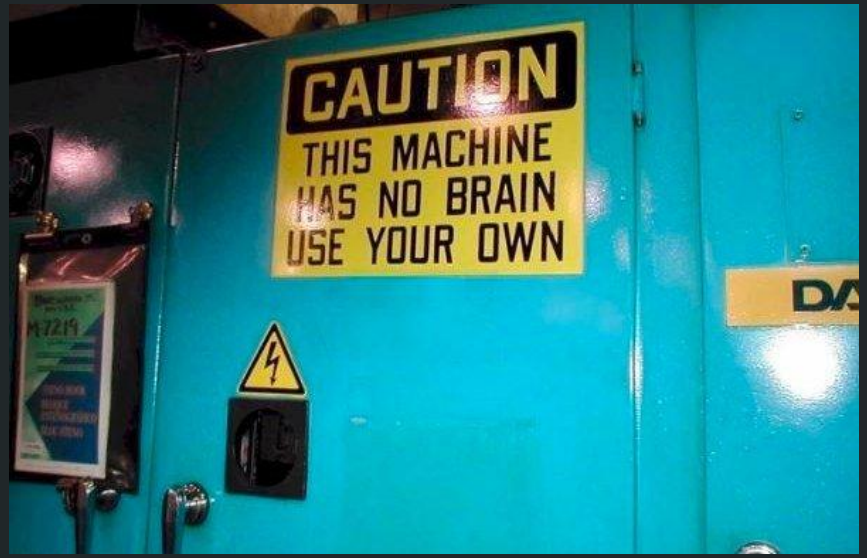
PYTHON

FOR NETWORK ENGINEERS

Onsite Training Session
June 2019

Day3 Schedule

- Review
- Pdb - The Python Debugger
- Basic Netmiko for SSH Management
- Serialization: JSON and YAML
- Requests and using a REST-API (Slack)
- Handling complex data structures
- Integrating to the operating system with subprocess (optional)



Flickr: Pierre-Olivier Carles

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)
```

Pdb - The Python Debugger

```
python -m pdb my_script.py

import pdb
pdb.set_trace()
```

Pdb Commands

```
help (h)
list (l)
list 1          # list starting at line1
list 1, 25      # list lines 1 - 25

next (n)        # Step one line at a time; don't descend
step (s)        # Step one line at a time descend into callables
break 16 (b 16) # Set a breakpoint at line 16
continue (c)    # Continue execution
```

Pdb - The Python Debugger

./day3/pdb_ex1.txt

Pdb Commands

```
down (d)        # Move down the stack
up (u)          # Move up the stack

p foo           # Print out variable foo
pp foo          # Pretty print out variable foo

!print(“hello”)  # Exclamation point can prefix generic Python code

quit (q)        # Abort the current Pdb session and program
```

Netmiko

(p)expect → Paramiko → Netmiko

Paramiko is the standard Python SSH library.

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

https://github.com/ktbyers/netmiko

Netmiko Vendors

- Currently (very) roughly 60 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

NETMIKO

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

```
#!/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

NETMIKO

Netmiko and TextFSM

Reference Material in:

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

$ python netmiko_textfsm.py

```
Protocol  Address         Age (min)  Hardware Addr   Type   Interface
Internet  172.31.255.255        -   c800.84b2.e9c4  ARPA   Vlan3967
Internet  172.31.255.254      134   8478.acae.c196  ARPA   Vlan3967
```

[{'protocol': 'Internet', 'address': '172.31.255.255', 'age': '-', 'mac': 'c800.84b2.e9c4', 'type': 'ARPA', 'interface': 'Vlan3967'}, {'protocol': 'Internet', 'address': '172.31.255.254', 'age': '134', 'mac': '8478.acae.c196', 'type': 'ARPA', 'interface': 'Vlan3967'}]

Exercises:
./day3/netmiko_ex1.txt
./day3/netmiko_ex2.txt

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

REST API

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

Apps | SDN Live Seminar... | View pynet Discus... | Python Net Engr | Web Design | Weekly Zag Briefi... | Read Later | Network Engineeri... | https://github.co... | Email software | Tools | Engines | Blog

NetBox

API Root

API Root

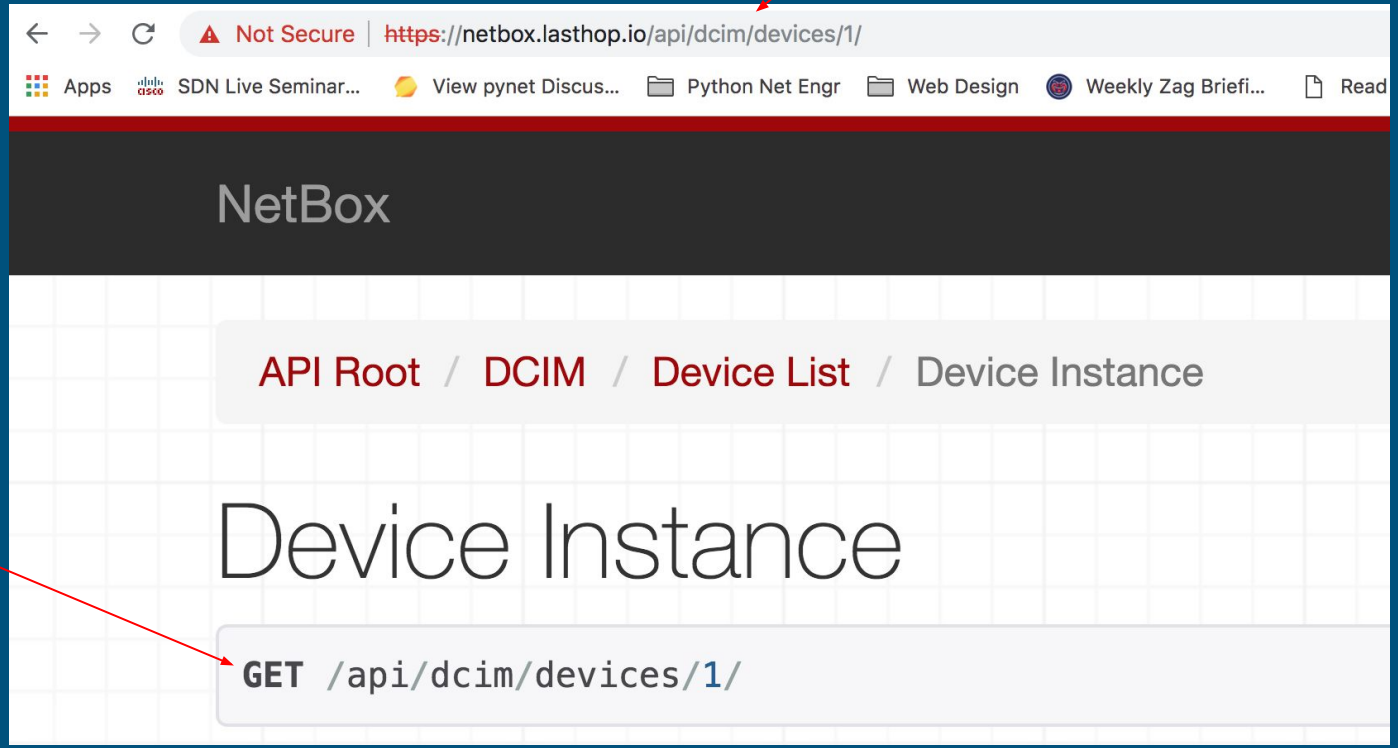
GET /api/

```
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

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
- Replace - HTTP Put
- Update - HTTP Patch
- Delete - HTTP Delete

Remember: Not all APIs are the same!

HTTP Methods Ref: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

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

Reference Material in:
{{ github_repo }}/rest_api

```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_ex1.txt
./day3/restapi_ex2.txt

- Simple Auth
- Token Based
- OAuth

```
import requests
from pprint import pprint

from urllib3.exceptions import InsecureRequestWarning

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


if __name__ == "__main__":

    token = "1234123412341234123412341341341134123433"
    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()
```

Tokens, Tokens Everywhere! & REST API - Basic Requests POST

Exercises:
./day3/restapi_ex3.txt

- Tokens can be included in multiple locations:
  - Headers
  - Encoded in URL
  - In a payload

- POST - expects a “payload”

```python
def main():
    # Get list of channels; authenticate with token in the header
    headers = {"Authorization": f"Bearer {SLACK_TOKEN}"}
    resp = requests.get(f"{SLACK_BASE_URL}/channels.list", headers=headers)
    pprint(resp.json())

    print()

    # Get list of channels; authenticate with token encoded in url
    resp = requests.get(f"{SLACK_BASE_URL}/channels.list?token={SLACK_TOKEN}")
    pprint(resp.json())

    print()

    # Get list of channels; authenticate with token encoded in url
    data = {"token": SLACK_TOKEN}
    resp = requests.post(f"{SLACK_BASE_URL}/channels.list", data=data)
    pprint(resp.json())

    print()
```

REST API

1. Determine if there is an existing Python library available.
    a. If library exists, does it do everything you need? If not: requests
    b. If no library: requests
2. Determine how to accomplish authentication.
    a. Library should abstract much of this for you, if no library...
    b. Token based auth common for non public web services
    c. OAuth common for public web services
3. Gain an understanding of the object model if API is truly RESTful.
    a. Determine how to do information retrieval.
        i. If truly RESTful should be a a GET method, some/older style APIs may require POST where payload requests object to GET
    b. Determine how to create and modify objects.
        i. PUT/POST/PATH
4. Start building up abstractions to accomplish your goals.

Complex Data Structures

Exercises:
./day3/struct_ex1.txt
./day3/struct_ex2.txt

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'])
```

Subprocess - *Integrating to the System Operating System*

```python
import os

print()
print("Current working directory")
start_dir = os.getcwd()
print(os.getcwd())

print()
print("Path of module we are executing")
print(os.path.realpath(__file__))

print()
print("Is this a file?")
print(os.path.isfile(__file__))

print()
print("Change directory into /tmp")
os.chdir("/tmp")
print(os.getcwd())
```

PYTHON
FOR NETWORK ENGINEERS

Subprocess - *Integrating to the System Operating System*

```python
import subprocess


def subprocess_wrapper(cmd_list):
    """Wrapper to execute subprocess including byte to UTF-8 conversion."""
    proc = subprocess.Popen(cmd_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    std_out, std_err = proc.communicate()
    (std_out, std_err) = [x.decode("utf-8") for x in (std_out, std_err)]

    return (std_out, std_err, proc.returncode)


cmd_list = ["ls", "-a", "-l"]
std_out, std_err, return_code = subprocess_wrapper(cmd_list)
print()
print(f"Return Code: {return_code}")
print(std_out)
print()
```

Reference Material in:

{{ github_repo }}/subprocess_example

PYTHON
FOR NETWORK ENGINEERS

The end...

Questions?

ktbyers@twb-tech.com

carl@twb-tech.com