



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

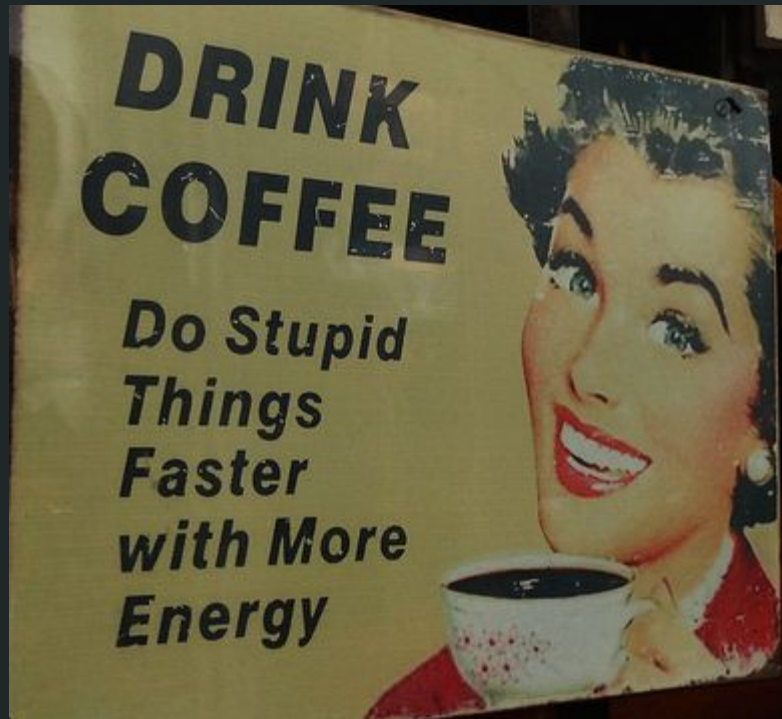
Focused

Minimize Distractions

Exercises and Examples

Examples in the Python Shell

Try not to fall behind on day1 & 2



Day1 Schedule

1. GIT Basics

1a. 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

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

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 great 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

To exit

:wq - saves file and exits

:q! - exits WITHOUT saving

REMEMBER:

<esc> is your friend

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

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`

`./day1/files_ex2.txt`

Lists

Zero-based indices

.append()

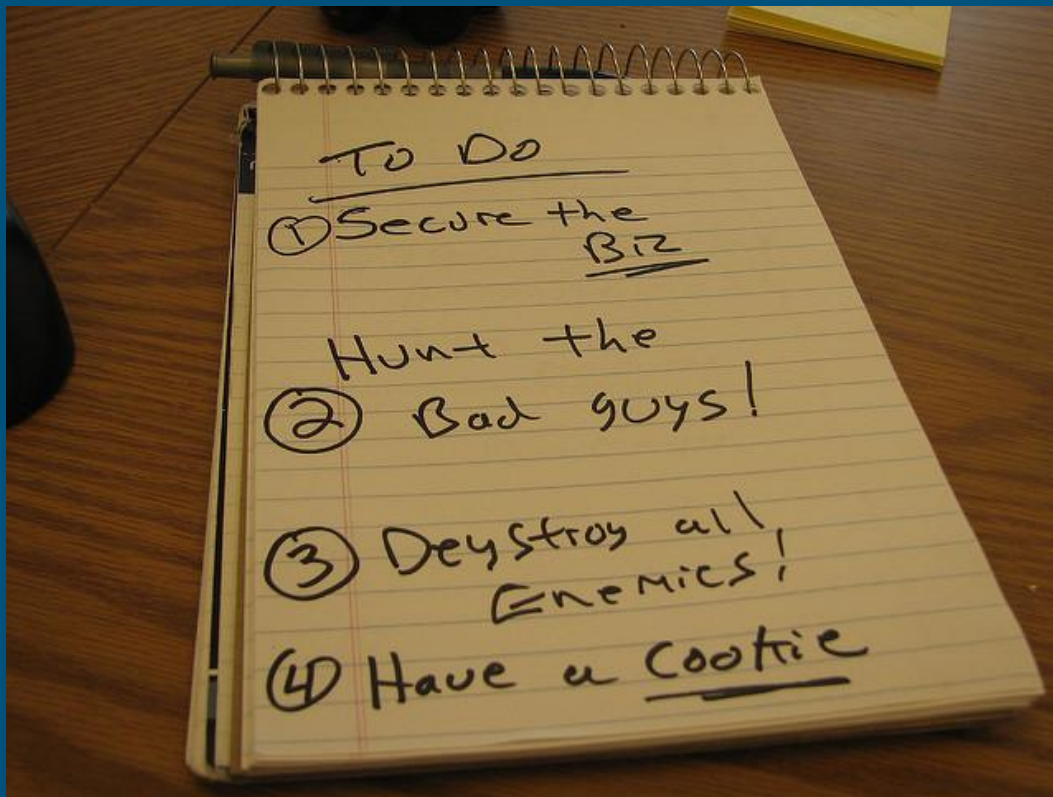
.pop()

.join()

List slices

Tuple

Copying a list



Exercises:

./day1/lists_ex1.txt

./day1/lists_ex2.txt

Photo: Purple Slog (Flickr)

Booleans and None

Boolean operators (and, or, not)

is

Truthy

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)

Day2

1. Exceptions
2. Regular Expressions
3. Functions
4. Python Code Structure
5. Linters
6. Classes and Objects (HERE)
7. Managing Python Libraries
8. Modules and Packages



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:
./day2/except_dict_ex1.txt

Python Regular Expressions

import re

Other re methods

re.split()

re.sub()

re.findall()

Exercises:

`./day2/regex_ex1.txt`

`./day2/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.*)

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:

`./day2/func_ex1.txt`

`./day2/func_ex2.txt`

`./day2/func_ex3.txt`

`./day2/func_ex4.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`

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
```

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](#)

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`

Exercises:

`./day2/reuse_ex2.txt`

`./day2/reuse_ex3.txt`

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

Day3 Schedule

1. Writing reusable code
2. Virtualenv
3. Namespaces

Python Applied to Network Engineering

4. Python + SNMP
5. Sending Email Notifications
6. CiscoConfParse
7. Python and SSH
8. netmiko-tools



Flickr: Pierre-Olivier Carles

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==1.4.3
```

```
pip install pycodestyle
```

Exercises:
./day3/virtualenv_ex1.txt

Namespaces (very briefly)

General rules:

1. Look in function first.
2. Look in global.
3. Look in builtins.

`global my_var` `# generally avoid doing this`

Python + SNMP

Using PySNMP library

Using helper library I created, see:

`~/python-libs/snmp_helper.py`

Reference Material in `{{ github_repo }}/snmp_example`

Exercises:

`./day3/snmp_ex1.txt`

`./day3/snmp_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

Reference Material in:

{{ github_repo }}/confparse_example

```
#!/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
```

Paramiko & 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 ICX/FastIron
Brocade MLX/NetIron
Cisco WLC
Dell-Force10 DNOS9
Dell PowerConnect
Huawei
Mellanox
Palo Alto PAN-OS
Pluribus
Vyatta VyOS

Experimental

A10
Accedian
Alcatel-Lucent SR-OS
Aruba
Ciena SAOS
Cisco Telepresence
CheckPoint Gaia
Enterasys
Extreme EXOS
Extreme Wing
F5 LTM
Fortinet
MRV Communications OptiSwitch

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 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()
```

Reference Material in:

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

Exercises:

```
./day4/netmiko_ex1.txt
./day4/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

Day4 Schedule

1. Serialization: JSON and YAML
2. Concurrency: Threads and Processes
3. Arista eAPI
4. Juniper, NETCONF, and PyEZ

Data Serialization

Why do we need data serialization?

Characteristics of JSON

Characteristics of YAML

Reference Material in:

`{{ github_repo }}/json_yaml`

Exercises:

`./day4/yaml_ex1.txt`

`./day4/yaml_ex2.txt`

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`

Arista eAPI

Reference Material in:

{{ github_repo }}/arista_pyeapi_example

```
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://{username}:{password}@{ip}:443/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")
```

Exercises:

`./day4/arista_ex1.txt`

`./day4/arista_ex2.txt`

~/.eapi.conf file contains connection definition information

Juniper, NETCONF, and PyEZ

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

NETCONF

NETCONF by Example

<https://trac.ietf.org/trac/edu/raw-attachment/wiki/IETF94/94-module-3-netconf.pdf>

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  
  <get/>  
</rpc>
```

```
<rpc-reply message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  
  <data>  
    <!-- ... entire set of data returned ... -->  
  </data>  
</rpc-reply>
```

NETCONF Operations

The base protocol includes the following protocol operations:

- o get
- o get-config
- o edit-config
- o copy-config
- o delete-config
- o lock
- o unlock
- o close-session
- o kill-session

*From RFC6241

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)
```

Reference Material in:

`{{ github_repo }}/juniper_example`

Exercises:

`./day4/juniper_ex1.txt`

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()
```

Reference Material in:

`{{ github_repo }}/juniper_example`

PyEZ config operations

Reference Material in:

`{{ github_repo }}/juniper_example`

```
#!/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)
```

Exercises:

`./day4/juniper_cfg_ex1.txt`

```
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()
```

PyEZ RPC operations

Reference Material in:

{{ github_repo }}/juniper_example

show version | display xml rpc

```
pyclass@juniper1> show version | display xml rpc
```

```
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/15.1F4/junos">
```

```
<rpc>
```

```
<get-software-information>
```

```
</get-software-information>
```

```
</rpc>
```

```
...
```

```
</rpc-reply>
```

```
show_version = a_device.rpc.get_software_information()
```

Exercises:

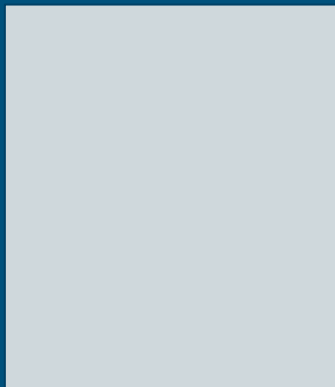
./day4/juniper_ex2.txt

Day5 Schedule

1. Jinja2 Templating
2. Integrating to a Database
3. NAPALM
4. Pulling data from a CSV file
5. Unit Testing
6. Continuous Integration

Jinja2

Configuration Template



Variables



Output Files



Jinja2 Template - the double curly-brace

```
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
{{ github_repo }}/jinja2_example/jinja2_bgp2.py
```

Jinja2 Template - Loading Template from a File

```
import jinja2
```

```
template_file = 'juniper_bgp.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:

`./day5/jinja_ex1.txt`

Jinja2 Template - 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 Template - 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

Exercise

On juniper1.twb-tech.com and juniper2.twb-tech.com configure an IP address on the ge-0/0/2 interface. Additionally configure eBGP peering between the two devices.

Use the ./day5/build_bgp.j2 template to accomplish this.

At the end of the configuration, you should be able to ping across the ge-0/0/2 links and BGP should be in the established state.

Integrating to a DB

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

Reference notes in:

`{{ github_repo }}/django/django_notes.txt`

Defining the Database Fields (models.py)

```
class NetworkDevice(models.Model):  
    device_name      = models.CharField(primary_key=True, max_length=80)  
    device_type      = models.CharField(max_length=50)  
    ip_address       = models.GenericIPAddressField()  
    port             = models.IntegerField()  
    vendor           = models.CharField(max_length=50, blank=True, null=True)  
    model            = models.CharField(max_length=50, blank=True, null=True)  
    os_version       = models.CharField(max_length=100, blank=True, null=True)  
    serial_number    = models.CharField(max_length=50, blank=True, null=True)  
    uptime_seconds   = models.IntegerField(blank=True, null=True)  
    credentials      = models.ForeignKey(Credentials, blank=True, null=True)
```

Initializing the DB

```
cd ~/DJANGO/djproject
```

```
$ python manage.py makemigrations
```

Migrations for 'net_system':

0001_initial.py:

- Create model Credentials
- Create model NetworkDevice

```
$ python manage.py migrate
```

...

Exercises:

Initialize your Django Database

See:

[./day5/db_ex1a_solution.txt](#)

Create/Delete Objects

```
cd ~/DJANGOX/djproject/  
$ python manage.py shell  
...  
>>> from net_system.models import NetworkDevice  
>>> pynet_sw2 = NetworkDevice(  
...     device_name='pynet-sw2',  
...     device_type='arista_eos',  
...     ip_address='184.105.247.73',  
...     port=22,  
... )  
>>> pynet_sw2.save()  
>>> pynet_sw2.delete()  
>>> pynet_sw2 = NetworkDevice.objects.get_or_create(...)
```

Load Data into the DB

```
$ cd ~/DJANGOX/djproject/net_system
```

```
$ python load_devices.py
```

```
(<NetworkDevice: NetworkDevice object>, True)
```

```
(<NetworkDevice: NetworkDevice object>, True)
```

```
(<NetworkDevice: NetworkDevice object>, True)
```

```
(<NetworkDevice: NetworkDevice object>, True)
```

```
(<NetworkDevice: NetworkDevice object>, True)
```

```
(<NetworkDevice: NetworkDevice object>, True)
```

```
$ python load_credentials.py
```

```
(<Credentials: Credentials object>, True)
```

```
(<Credentials: Credentials object>, True)
```

Exercises:

Load your data.

See:

[./day5/db_ex1b_solution.txt](#)

Query the Database

Exercises:
Manually query the database

```
$ python manage.py shell
```

```
...
```

```
>>> from net_system.models import NetworkDevice
```

```
>>> all_devices = NetworkDevice.objects.all()
```

```
>>> all_devices
```

```
[<NetworkDevice: pynet-rtr1>, <NetworkDevice: pynet-rtr2>, <NetworkDevice: pynet-sw1>,  
<NetworkDevice: pynet-sw2>, <NetworkDevice: pynet-sw3>, <NetworkDevice: pynet-sw4>,  
<NetworkDevice: juniper-srx>]
```

```
>>> all_devices[0]
```

```
<NetworkDevice: pynet-rtr1>
```

```
>>> all_devices[0].ip_address
```

```
'184.105.247.70'
```

See:

[./day5/db_ex1c.txt](#)

Link to credentials

```
>>> NetworkDevice.objects.get(ip_address='184.105.247.72')
```

```
<NetworkDevice: pynet-sw1>
```

```
>>> arista1 = NetworkDevice.objects.get(ip_address='184.105.247.72')
```

```
>>> from net_system.models import Credentials
```

```
>>> creds = Credentials.objects.all()
```

```
>>> creds
```

```
[<Credentials: pyclass>, <Credentials: admin1>]
```

```
>>> arista_creds = creds[1]
```

```
>>> arista1.credentials = arista_creds
```

```
>>> arista1.save()
```

Exercises:

`./day5/db_ex1d.txt`

Solution:

`./day5/db_ex1d_solution.txt`

`./day5/db_ex1d.py`

Retrieving all objects using a given credential

```
>>> arista_creds  
<Credentials: admin1>
```

```
>>> arista_creds.networkdevice_set.all()  
[<NetworkDevice: pynet-sw1>, <NetworkDevice: pynet-sw2>]
```

Exercises:

[./day5/db_ex2.txt](#)

[./day5/db_ex3.txt](#)

[./day5/db_ex4.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

COMMUNITY

Fortinet Fortios

Mikrotik RouterOS

Palo Alto NOS

Pluribus

VyOS

NAPALM Getters

Exercises:

`./day5/napalm_ex1.txt`

`./day5/napalm_ex2.txt`

`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()`

Exercises:
[./day5/napalm_ex3.txt](#)
[./day5/napalm_ex4.txt](#)

The end...

Questions?

ktbyers@twb-tech.com

Twitter: @kirkbyers