PYTHON
FOR NETWORK ENGINEERS

Onsite Training Session
June 2019

$ whoami

Kirk Byers
Network Engineer:
CCIE #6243 (emeritus)

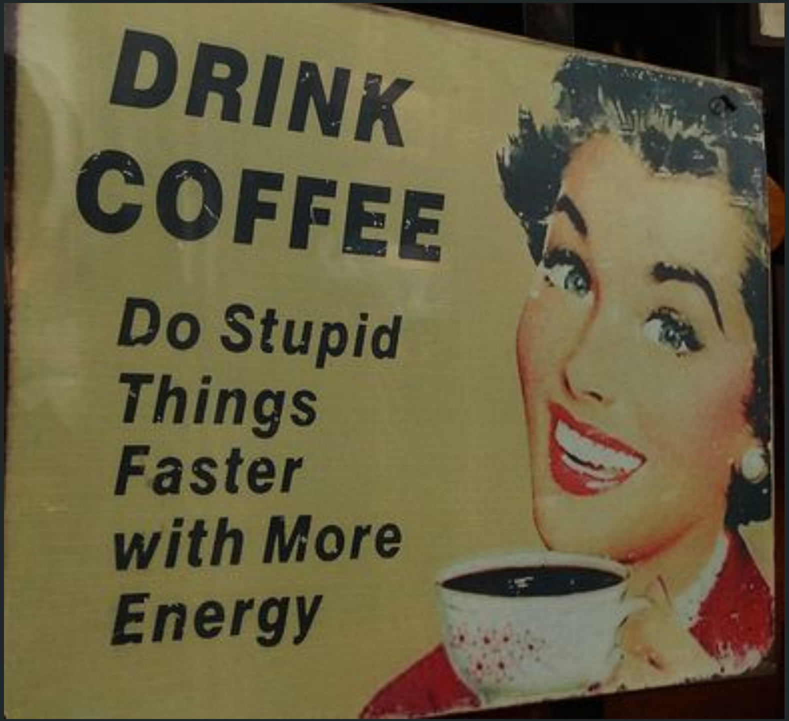
Programmer:
Netmiko
NAPALM
Nornir

Teach Python, Ansible, Nornir in a Network Automation context

General:

June 20, 8:00AM - 5:00AM
June 27, 8:00AM - 5:00AM
<July 4 - break week/holiday>
July 11, 8:00AM - 5:00AM
July 18, 8:00AM - 5:00AM

Focused/Minimize Distractions
Exercises and Examples
Examples in the Python Shell
Try not to fall behind on day1 & 2



Flickr: Ben Sutherland

Day1 Schedule

Course introduction

Working with Git

Why Python?

Python3 versus Python2

Python Fundamentals (Part1)

- REPL / dir / help
- Variable naming / indented blocks
- Strings
- Files
- Lists
- Conditionals
- Loops
- Dictionaries
- Exceptions

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 diff *# what changed*

Git Push & Pull

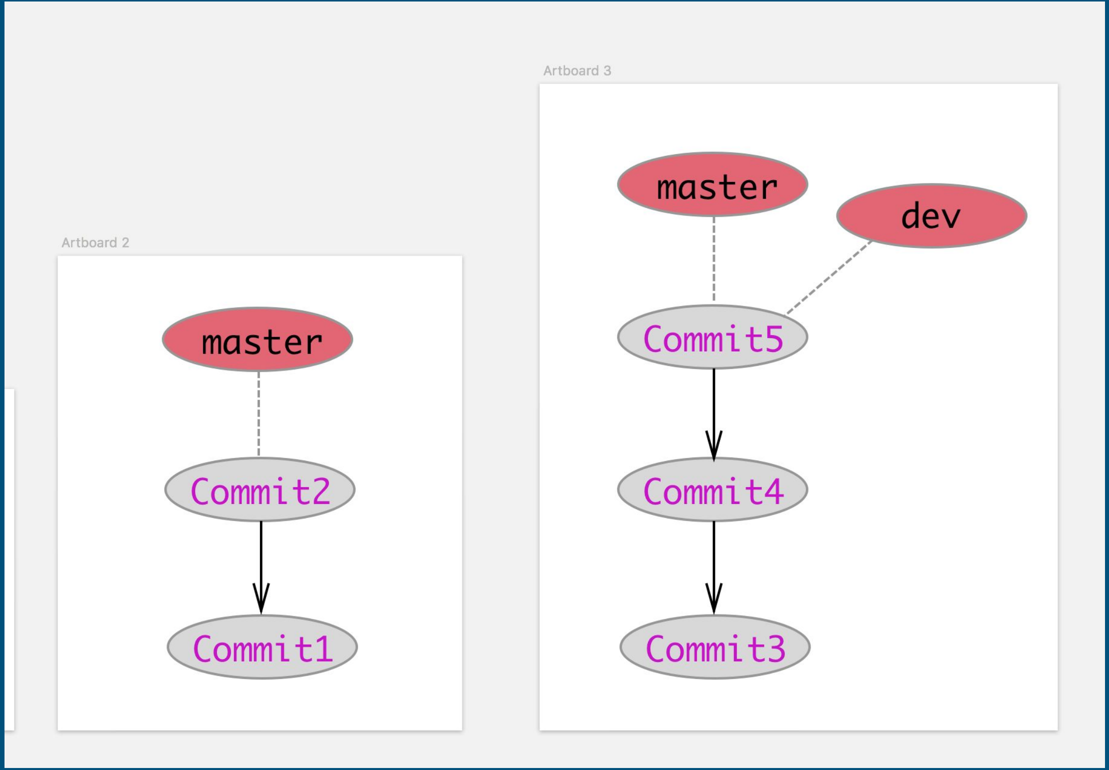
Changes have been committed locally, but haven't been pushed up to GitHub

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

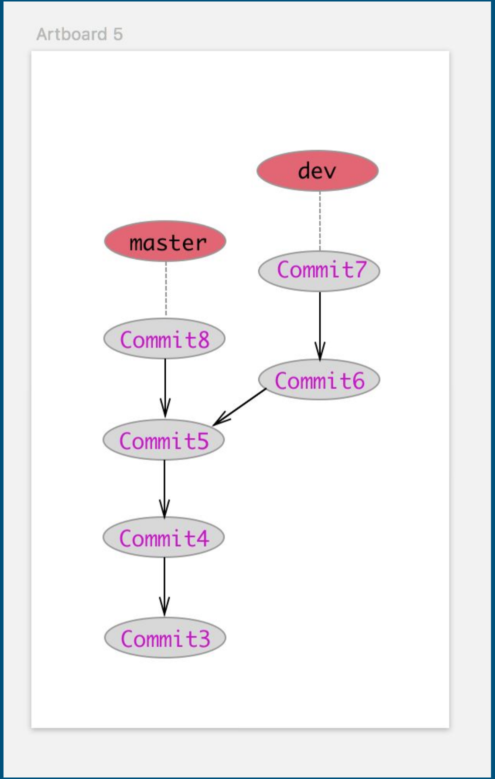
Reference Commands:
{{ github_repo }}/git_notes/git_commands.md

Exercises:
./day1/git_ex1.txt

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_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?

- 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 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()

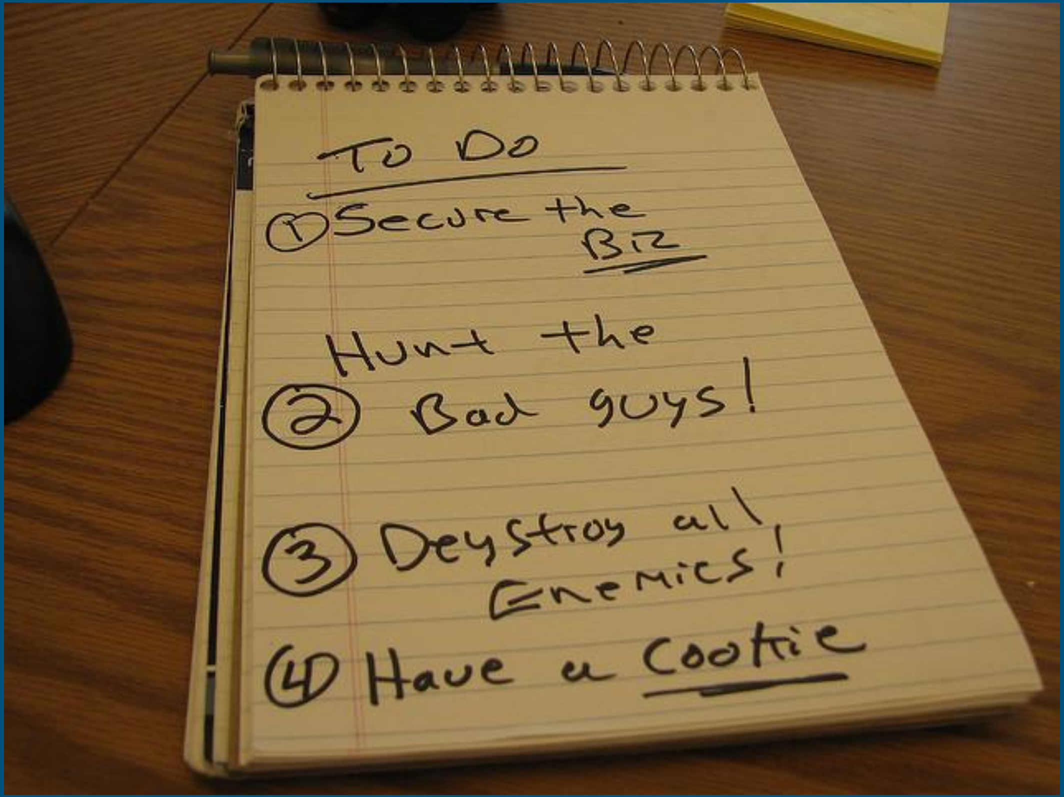
.pop()

.join()

List slices

Tuple

Copying a list



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

Photo: Purple Slog (Flickr)

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

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

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.

Day2

1. Functions
2. Regular Expressions
3. Python Classes and Objects
4. Modules
5. Packages
6. Namespaces

Flickr: au_tiger01

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.