CISCO DEVNET

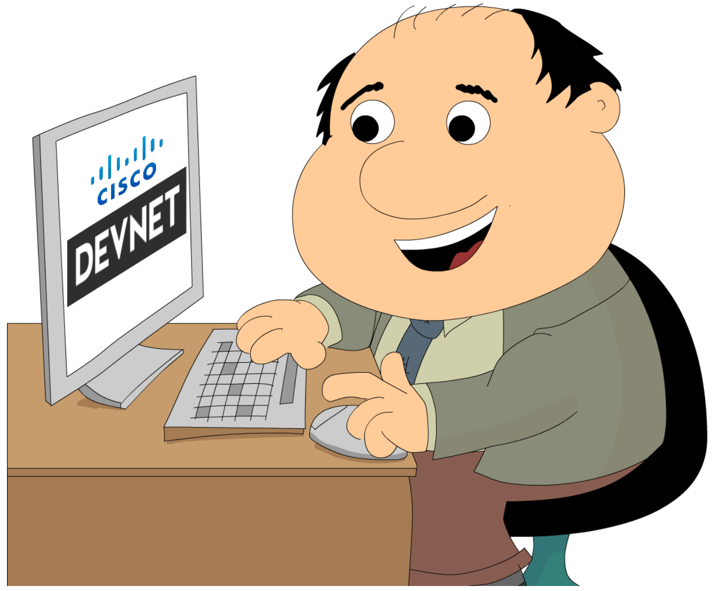
Python Part 2: Working with Libraries and Virtual Environments

A Network Programmability Basics Presentation

Hank Preston, ccie 38336
Developer Evangelist
@hfpreston

Network Programmability Basics Modules

- Introduction: How to be a Network Engineer in a Programmable Age
- **Programming Fundamentals**
- Network Device APIs
- Network Controllers
- Application Hosting and the Network
- NetDevOps



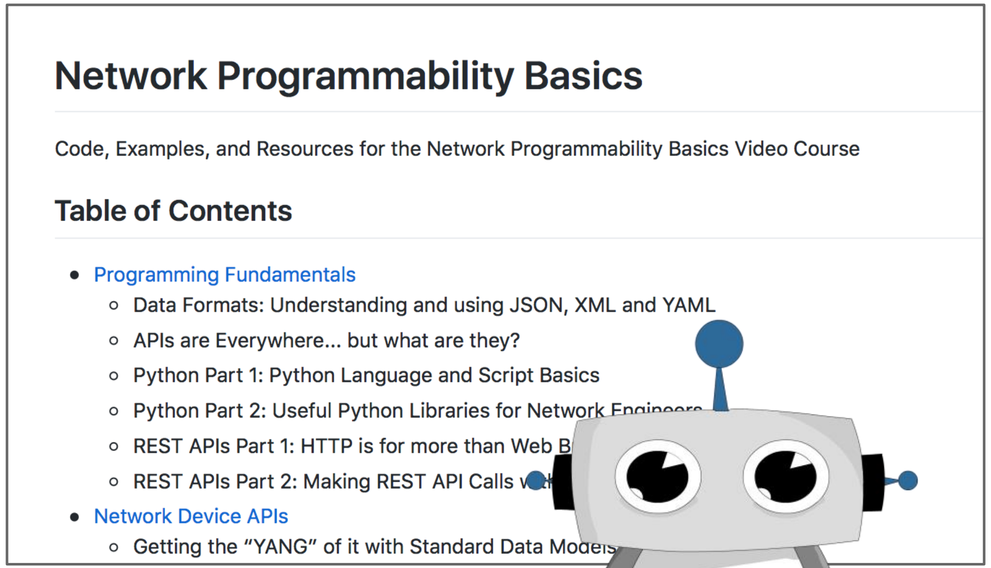
Network Programmability Basics: The Lessons

<u>Module: Programming Fundamentals</u>

- Data Formats: Understanding and using JSON, XML and YAML
- APIs are Everywhere… but what are they?
- REST APIs Part 1: HTTP is for more than Web Browsing
- REST APIs Part 2: Making REST API Calls with Postman
- Python Part 1: Python Language and Script Basics
- **Python Part 2: Working with Libraries and Virtual Environments**
- Python Part 3: Useful Python Libraries for Network Engineers

Code and Develop Along

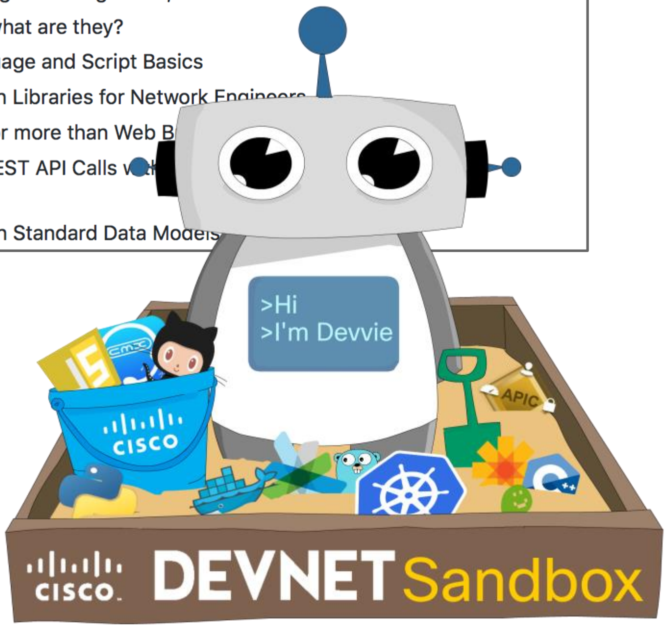
- Get the Code!
  - github.com/CiscoDevNet/netprog_basics
- Setup Lab Prerequisites
  - Each lab includes a README with details
- Access to Infrastructure
  - DevNet Sandbox
  - Specifics in lab README



Network Programmability Basics

Code, Examples, and Resources for the Network Programmability Basics Video Course

Table of Contents



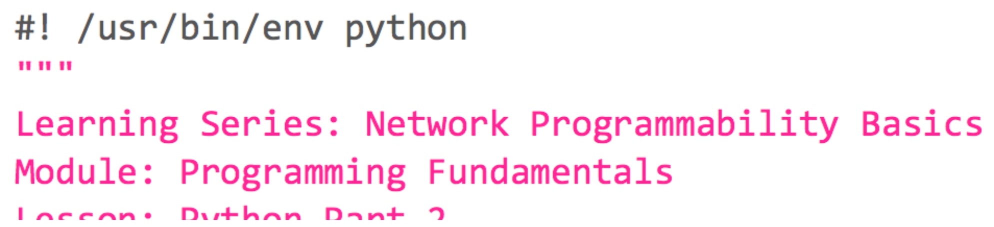
Topics to Cover

- What are Libraries and How to Use Them
- Using pip to Install Libraries
- Virtual Environments
- Foundational Libraries

What are Libraries and How to Use Them

Python Libraries (Modules, Applications, etc)

- Any Python code outside of your script you want to use
- Provide some capability or data you need
- Included with statements
  - `from` **`library`** `import` **`name`**
  - `import` **`library`**



```
#! /usr/bin/env python
"""
Learning Series: Network Programmability Basics
Module: Programming Fundamentals
Lesson: Python Part 2
```

netprog_basics/programming_fundamentals/python_part_2/example1.py

Python Libraries (Modules, Applications, etc)

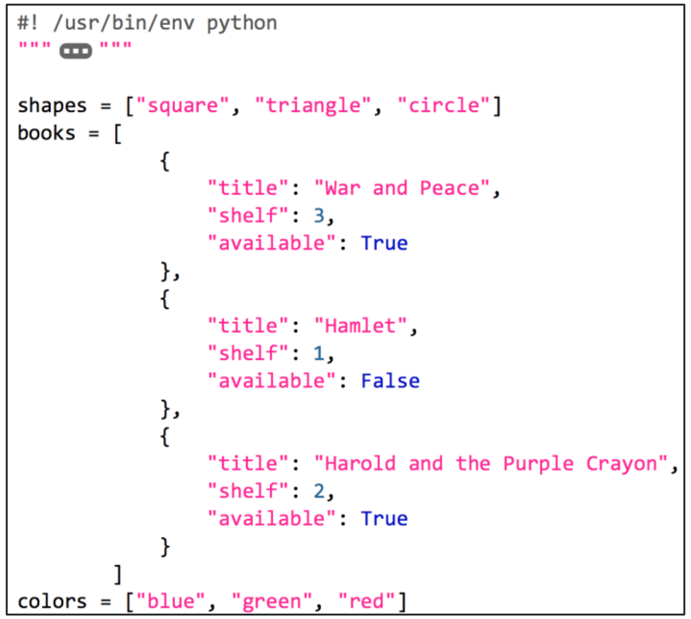
- Any Python code outside of your script you want to use
- Provide some capability or data you need
- Included with statements
  - `from` **`library`** `import` **`name`**
  - `import` **`library`**

```
# Import data from another script
from common_vars import shapes
# Import a library that offers date-time capabilities
import datetime

print("The shapes are:")
for shape in shapes:
    print(shape)
print("")

# Get Current Date and Time
date_now = datetime.datetime.now()
print("It is currently {}.".format(str(date_now)))

# Add 1000 minutes to Current Date and Time
new_date = date_now + datetime.timedelta(minutes=1000)
print("In 1000 minutes it will be {}.".format(str(new_date)))
```

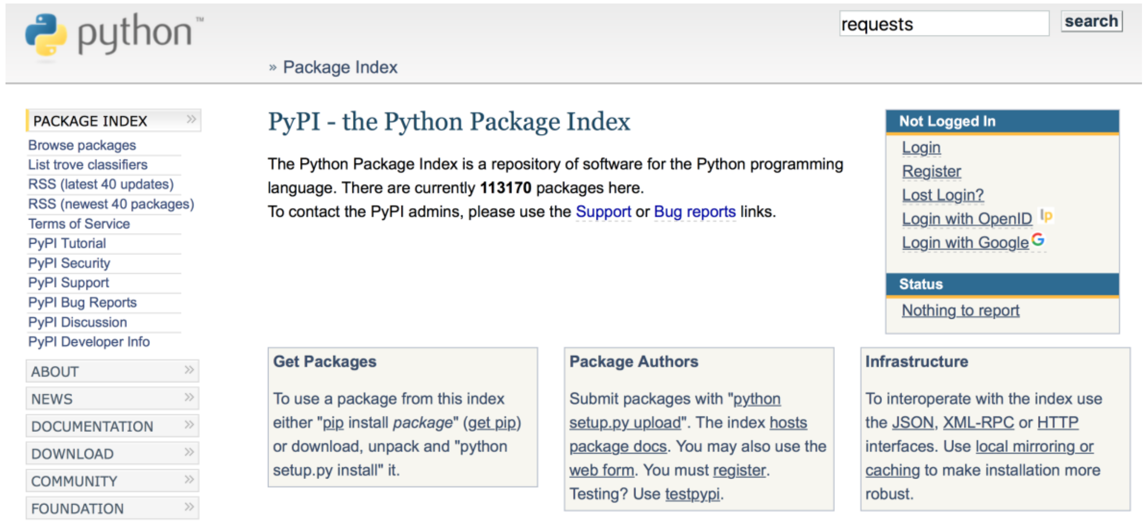
netprog_basics/programming_fundamentals/python_part_2/example1.py

Where to get Libraries

- Write them yourself
  - Example: `common_vars`
- Included with Python itself
  - Example: `datetime, os, sys, json`
- From Python Package Index (PyPI)
  - Example: `pip install requests`
- Download and install manually
  - Example: ACI Toolkit from GitHub



```
#! /usr/bin/env python
""" ... """

shapes = ["square", "triangle", "circle"]
books = [
            {
                "title": "War and Peace",
                "shelf": 3,
                "available": True
            },
            {
                "title": "Hamlet",
                "shelf": 1,
                "available": False
            },
            {
                "title": "Harold and the Purple Crayon",
                "shelf": 2,
                "available": True
            }
        ]
colors = ["blue", "green", "red"]
```



python™ requests search

» Package Index

PACKAGE INDEX
Browse packages
List trove classifiers
RSS (latest 40 updates)
RSS (newest 40 packages)
Terms of Service
PyPI Tutorial
PyPI Security
PyPI Support
PyPI Bug Reports
PyPI Discussion
PyPI Developer Info

ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
COMMUNITY
FOUNDATION

PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **113170** packages here.
To contact the PyPI admins, please use the Support or Bug reports links.

Not Logged In
Login
Register
Lost Login?
Login with OpenID
Login with Google

Status
Nothing to report

Get Packages
To use a package from this index either "pip install *package*" (get pip) or download, unpack and "python setup.py install" it.

Package Authors
Submit packages with "python setup.py upload". The index hosts package docs. You may also use the web form. You must register. Testing? Use testpypi.

Infrastructure
To interoperate with the index use the JSON, XML-RPC or HTTP interfaces. Use local mirroring or caching to make installation more robust.

netprog_basics/programming_fundamentals/python_part_2/common_vars.py

Using pip to Install Libraries

Pip, The Python Package Installer

- Python 2.7.6 and Python 3.4 or greater includes by default
- Integrates with PyPI for packages
- Install, Upgrade, and Uninstall packages
- "`requirements.txt`" in projects provide input to pip

```
DevNet$ pip --version
pip 9.0.1

DevNet$ pip install pyang
Collecting pyang
  Downloading pyang-1.7.3-py2.py3-none-any.whl (326kB)
    100% |████████████████████████████████| 327kB
1.3MB/s
Installing collected packages: pyang
Successfully installed pyang-1.7.3

DevNet$ pip install -r requirements.txt
Collecting requests (from -r requirements.txt (line 1))
  Downloading requests-2.18.2-py2.py3-none-any.whl
(88kB)
    100% |████████████████████████████████| 92kB
662kB/s
.
{OUTPUT TRUNCATED}
.
Successfully installed requests-2.18.2 six-1.10.0
urllib3-1.22
```

netprog_basics/programming_fundamentals/python_part_2/requirements.txt

pip Commands to Know

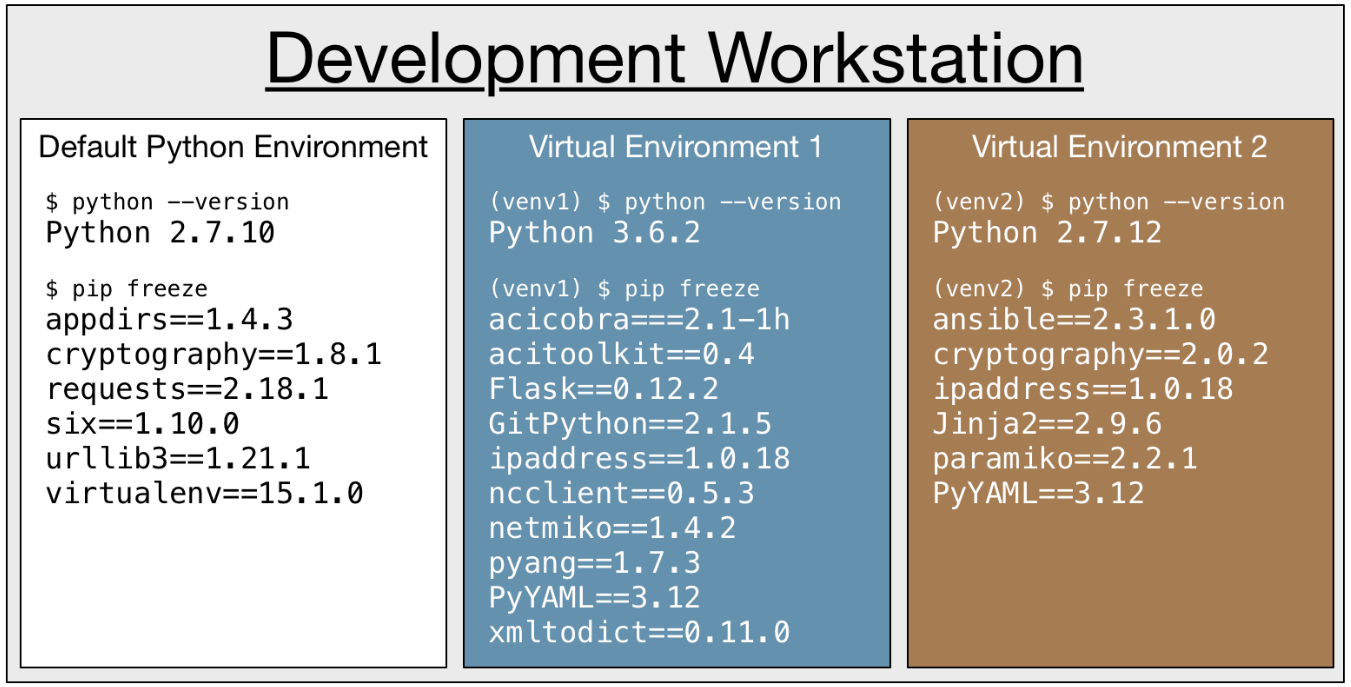
- Install a package
  - `pip install` **`package`**
- Upgrade a package
  - `pip install --upgrade` **`package`**
- Uninstall a package
  - `pip uninstall` **`package`**
- View all packages installed
  - `pip freeze`
- Install "`requirements.txt`"
  - `pip install –r requirements.txt`

```
DevNet$ pip freeze

asn1crypto==0.22.0
bcrypt==3.1.3
certifi==2017.4.17
cffi==1.10.0
chardet==3.0.4
cryptography==2.0
flake8==3.3.0
idna==2.5
lxml==3.8.0
mccabe==0.6.1
ncclient==0.5.3
paramiko==2.2.1
pyang==1.7.3
pyasn1==0.2.3
pycodestyle==2.3.1
pycparser==2.18
pyflakes==1.5.0
PyNaCl==1.1.2
requests==2.18.2
six==1.10.0
urllib3==1.22
```

Virtual Environments

What is a Virtual Environment (venv)

- Build isolated, fully functional Python environments on a single workstation
- Virtual Environments can
  - Run different versions of Python
  - Have different libraries installed
  - Have different versions of libraries installed



Development Workstation

Default Python Environment

```
$ python --version
Python 2.7.10

$ pip freeze
appdirs==1.4.3
cryptography==1.8.1
requests==2.18.1
six==1.10.0
urllib3==1.21.1
virtualenv==15.1.0
```

Virtual Environment 1

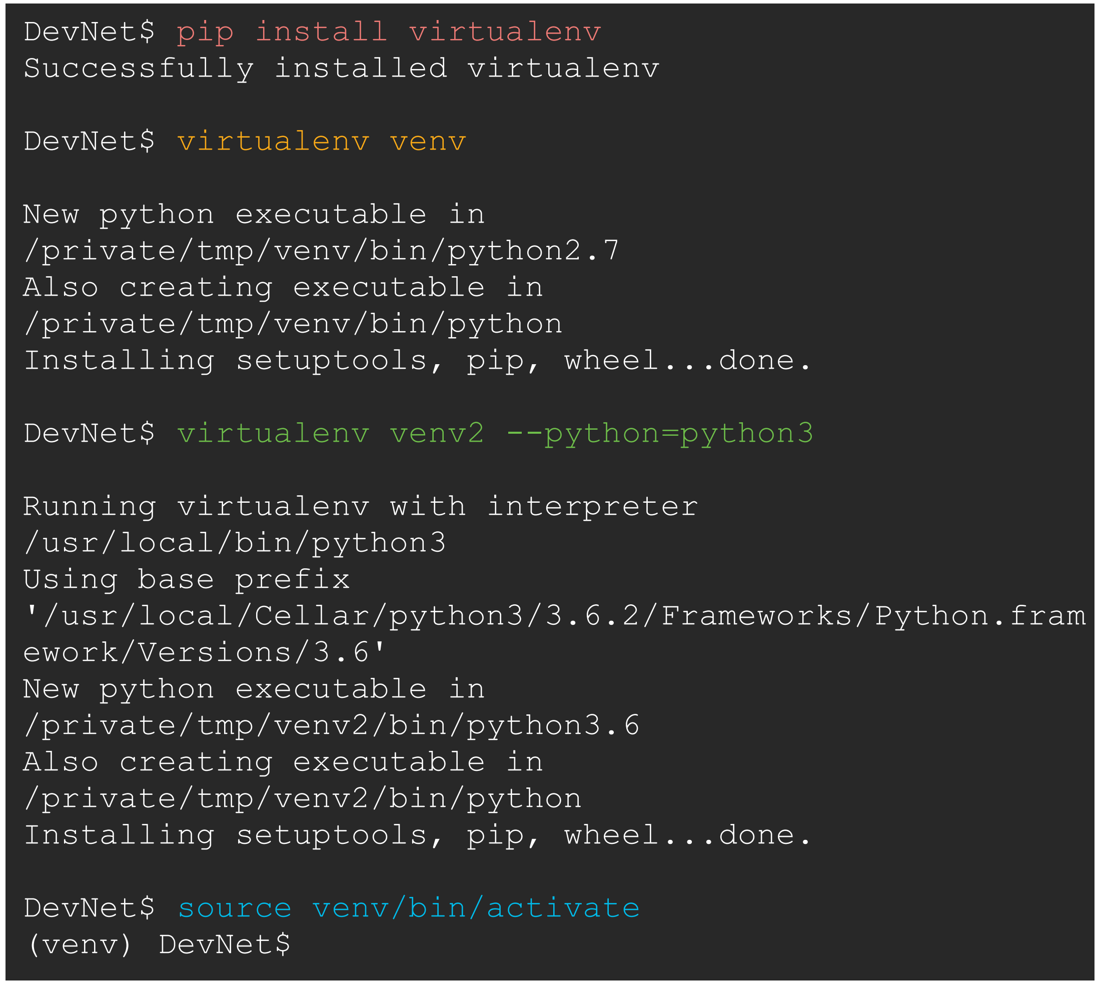
```
(venv1) $ python --version
Python 3.6.2

(venv1) $ pip freeze
acicobra===2.1-1h
acitoolkit==0.4
Flask==0.12.2
GitPython==2.1.5
ipaddress==1.0.18
ncclient==0.5.3
netmiko==1.4.2
pyang==1.7.3
PyYAML==3.12
xmltodict==0.11.0
```

Virtual Environment 2

```
(venv2) $ python --version
Python 2.7.12

(venv2) $ pip freeze
ansible==2.3.1.0
cryptography==2.0.2
ipaddress==1.0.18
Jinja2==2.9.6
paramiko==2.2.1
PyYAML==3.12
```

Setting Up a Virtual Environment

- Install the virtualenv library
  - `pip install virtualenv`
- Create the Virtual Environment
  - `virtualenv` **`name`**
- Specify Python Version
  - `virtualenv` **`name`** `--python=python3`
- Activate Virtual Environment
  - `source` **`name`**`/bin/activate*`
- Deactivate Virtual Environment
  - `deactivate`



```
DevNet$ pip install virtualenv
Successfully installed virtualenv

DevNet$ virtualenv venv

New python executable in
/private/tmp/venv/bin/python2.7
Also creating executable in
/private/tmp/venv/bin/python
Installing setuptools, pip, wheel...done.

DevNet$ virtualenv venv2 --python=python3

Running virtualenv with interpreter
/usr/local/bin/python3
Using base prefix
'/usr/local/Cellar/python3/3.6.2/Frameworks/Python.fram
ework/Versions/3.6'
New python executable in
/private/tmp/venv2/bin/python3.6
Also creating executable in
/private/tmp/venv2/bin/python
Installing setuptools, pip, wheel...done.

DevNet$ source venv/bin/activate
(venv) DevNet$
```

** Commands on Windows Platforms slightly different*

Installing Python Libraries in Virtual Environments

- Once activated, no different

```
(venv) DevNet$ pip install pyang
Collecting pyang
  Downloading pyang-1.7.3-py2.py3-none-any.whl (326kB)
    100% |████████████████████████████████| 327kB
1.3MB/s
Installing collected packages: pyang
Successfully installed pyang-1.7.3

(venv) DevNet$ pip install -r requirements.txt
Collecting requests (from -r requirements.txt (line 1))
  Downloading requests-2.18.2-py2.py3-none-any.whl
(88kB)
    100% |████████████████████████████████| 92kB
662kB/s
.
{OUTPUT TRUNCATED}
.
Successfully installed requests-2.18.2 six-1.10.0
urllib3-1.22
```

Foundational Libraries

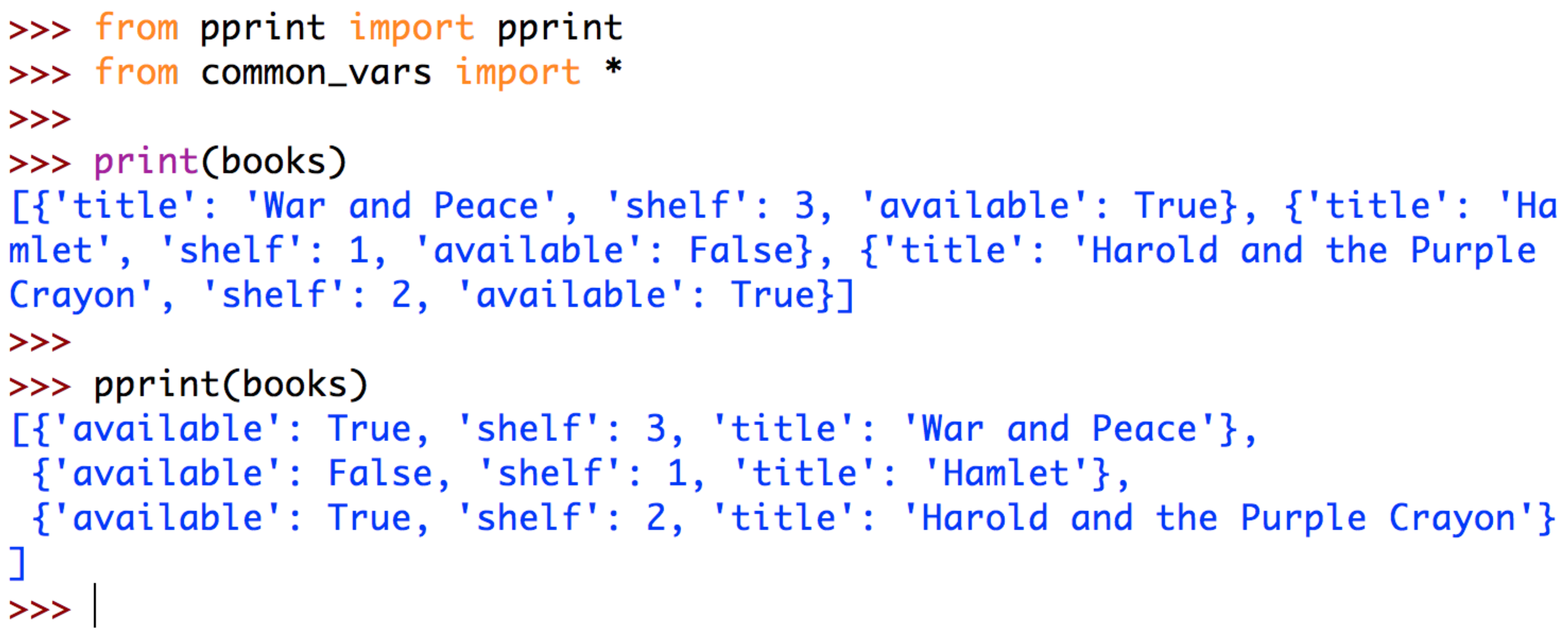
Core Python Libraries to Know and Love

- Pretty Print
  - `from pprint import pprint`
- Python Interpreter Utilities
  - `import sys`
- Operating System Interfaces
  - `import os`
- Date and Time Utilities
  - `import datetime`

* Many libraries included with core Python, see docs

Prettier Printing with pprint

- Better formatting than default print() function



```
>>> from pprint import pprint
>>> from common_vars import *
>>>
>>> print(books)
[{'title': 'War and Peace', 'shelf': 3, 'available': True}, {'title': 'Ha
mlet', 'shelf': 1, 'available': False}, {'title': 'Harold and the Purple
Crayon', 'shelf': 2, 'available': True}]
>>>
>>> pprint(books)
[{'available': True, 'shelf': 3, 'title': 'War and Peace'},
 {'available': False, 'shelf': 1, 'title': 'Hamlet'},
 {'available': True, 'shelf': 2, 'title': 'Harold and the Purple Crayon'}
]
>>> |
```

https://docs.python.org/3/library/pprint.html

Access Details about Python with sys

- Access to some details/variables concerning running state
  - Access command line arguments with `sys.argv[]`
- Access to functions that interact with the interpreter
  - Exit Python with specific error message `sys.exit("Message")`

```
DevNet$ python -i common_vars.py "CLI Arg 1" "CLI Arg 2"
>>>
>>> import sys
>>>
>>> sys.argv[1]
'CLI Arg 1'
>>> sys.argv[2]
'CLI Arg 2'
>>>
>>> sys.exit("Error Occurred")
Error Occurred
```

https://docs.python.org/3/library/sys.html

Interact with Files, Paths, and Environment with os

- Access and manipulate directories and files
  - Note: Opening files can be done with `open(`**`filename`**`)`
- Access and Manipulate Environment Variables
  - `os.environ[`**`var_name`**`]`

```
>>> import os
>>> os.getcwd()
'/Users/hapresto/coding'
>>> os.chdir("../")
>>> os.getcwd()
'/Users/hapresto'
>>>
>>> os.environ["USER"]
'hapresto'
>>> os.environ["VAR_FROM_PYTHON"]
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    os.environ["VAR_FROM_PYTHON"]
  File "/Users/hapresto/coding/netprog_basics/venv/bin/
line 669, in __getitem__
    raise KeyError(key) from None
KeyError: 'VAR_FROM_PYTHON'
>>> os.environ["VAR_FROM_PYTHON"] = "Set from Python"
>>> os.environ["VAR_FROM_PYTHON"]
'Set from Python'
```

https://docs.python.org/3/library/os.html

Get your Date and Time Correct with datetime

- Create, format, and manipulate dates and times
- Time arithmetic!
- Work with timestamps and other representations

```
>>> import datetime

>>> right_now = datetime.datetime.now()

>>> four_weeks_from_now = right_now + datetime.timedelta(weeks=4)

>>> date_display_format = "%I:%m %p on %B %w, %Y"

>>> right_now.strftime(date_display_format)
'11:07 AM on July 3, 2017'
>>> four_weeks_from_now.strftime(date_display_format)
'11:08 AM on August 3, 2017'
```

https://docs.python.org/3/library/datetime.html

Demo Time!

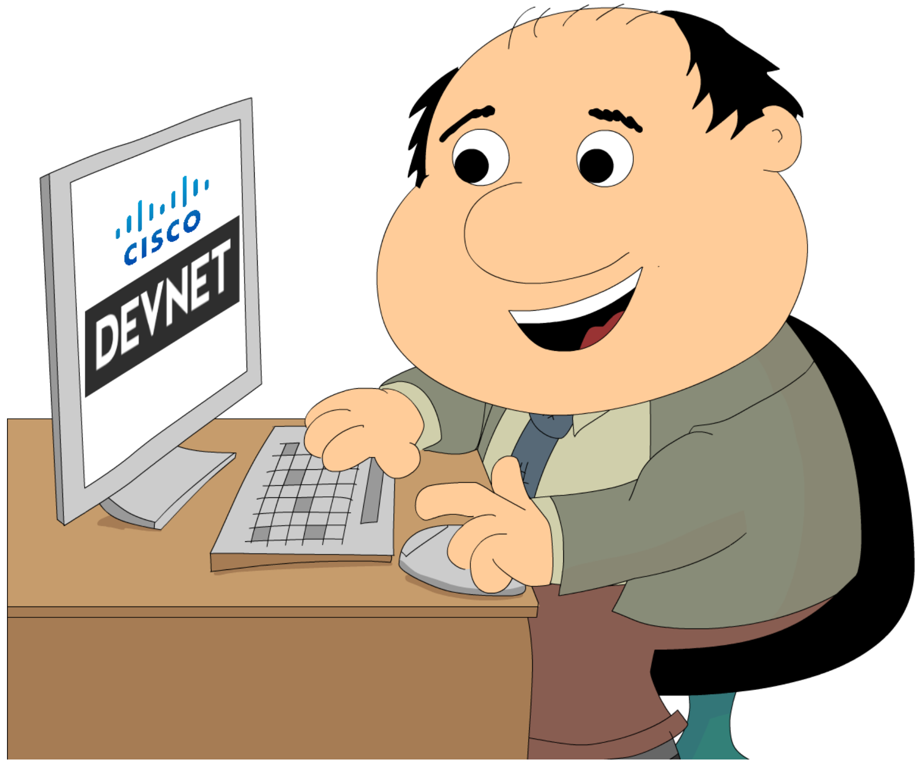
Summing up

Review

- Understand what Python libraries are and how to use them
- Looked at Python Virtual Environments, why and how to use them
- Explored core Python libraries for displaying data, managing running scripts, and working with the operating system

Call to Action!

- Complete the full **Network Programmability Basics** Course
- Run the examples and exercises yourself!
  - Bonus Examples!
- Join DevNet for so much more!
  - Learning Labs
  - Development Sandboxes
  - Code Samples and API Guides



Got more questions? Come find me!

hapresto@cisco.com

@hfpreston

http://github.com/hpreston

@CiscoDevNet

facebook.com/ciscodevnet/

http://github.com/CiscoDevNet