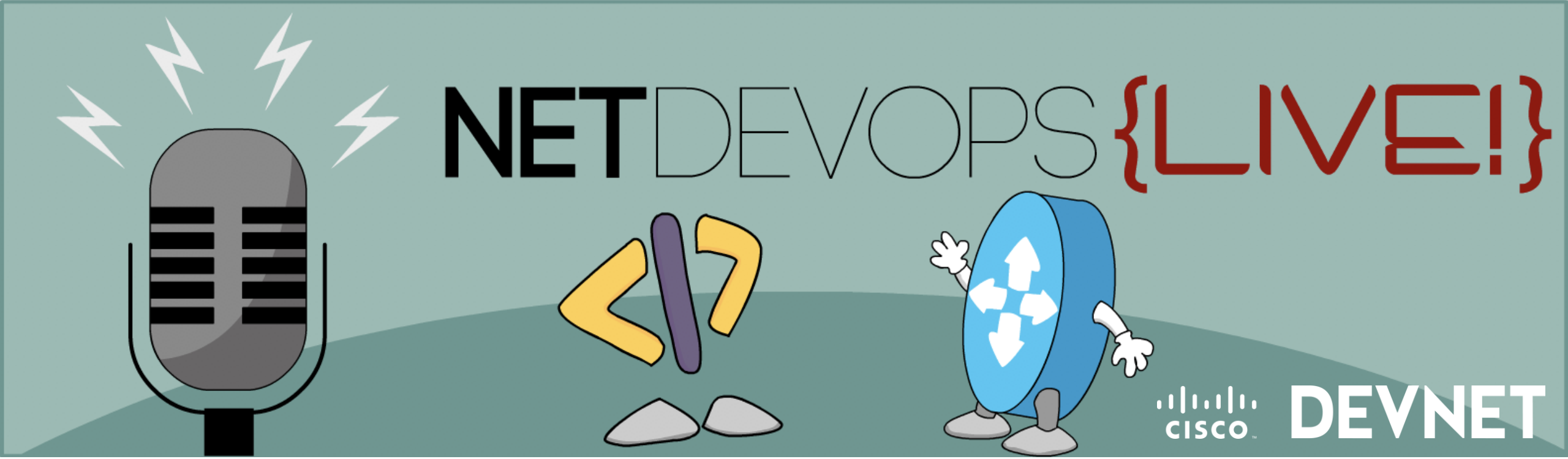


Python Skills and Techniques for Network Engineers, Part 1

Hank Preston, ccie 38336 R/S

NetDevOps Engineer, DevNet Sandbox

Twitter: @hfpreston

Season 2, Talk 1

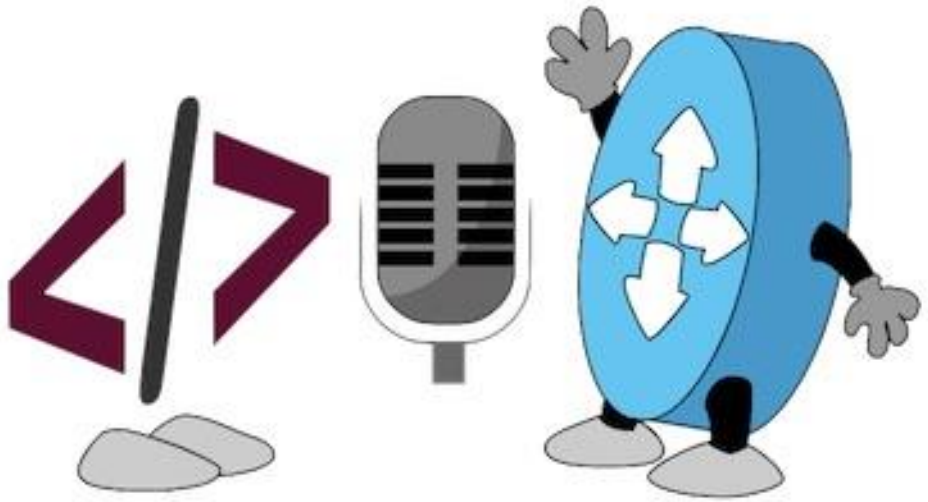
https://developer.cisco.com/netdevops/live

http://cs.co/ndl

Help us track NetDevOps Live Interest!

What are we going to talk about?

- Better Code Style / Sharing
  - Sh-Bang, Documents and Requirements, Linting, Licensing
- More Reusable Code
  - Functions, Modules, & Packages, Oh My!
- More Robust Code
  - Trying, Testing, CLI Tools, and the Environment



Code Samples Available On Code Exchange

python_code_tips directory

Before we start…

- ✓ Not all my code follows every suggestion
- ✓ Some examples are for specific concepts
- ✓ Some times I just didn't get to it yet
- ✓ Code is always evolving, working code first!

Better Code Style / Sharing

The Sh-Bang Line – Make Your Code Executable

- First line in scripts
- Identify the interpreter to run the script file
- Use /usr/bin/env python
  - Leverage the active Python environment
- Versus /usr/local/bin/python
  - Hard code specific version
- Also need to **chmod +x** the file

https://en.wikipedia.org/wiki/Shebang_(Unix)

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

import os


def say_hello(name):
    """Function that will say hello to someone.
    """
    # Print out a hello message to the name given
    print("Hello there {name}.".format(name = name))


if __name__ == "__main__":
    # If executed as a script, run this block.

    # List of names, and say hello to them
    names = ["Hank", "Eric", "Stuart", "Bryan"]
    for name in names:
        say_hello(name)
```

```
$ ./example1.py
Current directory is /Users/hapresto/code/python_networking/python_code_tips
The user id is 501
The user is a member of the following groups:
20,12,61,79,80,81,98,501,33,100,204,250,395,398,399,701
```

Code File

Doc Strings – What's this for anyway…

- Information about files, functions, objects, classes, etc
- Always first statement
- Surrounded with triple double quotes
- Become value of **__doc__** variable
- Often used in **help()** output

```
#! /usr/bin/env python
"""Example Python script.

Copyright (c) 2018 Cisco and/or its affiliates.
"""

import os


def say_hello(name):
    """Function that will say hello to someone.
    """
    # Print out a hello message to the name given
    print("Hello there {name}.".format(name = name))
```

https://www.python.org/dev/peps/pep-0257/

Code File

Requirements – So you want to run my code?

- Always include a **requirements.txt** file listing all Python dependencies
- Generate automatically with
  pip freeze > requirements.txt
- Versions of modules matter
  - Code changes, lock in known working version
  - ==, >=, <= helpful operators

```
$ cat requirements.txt

ansible==2.6.1
genie==3.0.0
ipython==6.5.0
napalm==2.3.1
ncclient==0.6.0
netmiko==2.2.2
pyang==1.7.5
pyats==4.1.0
pysnmp==4.4.4
pyyaml>=4.2b1
requests>=2.20.0
urllib3==1.23
virlutils==0.8.4
xmltodict==0.11.0
```

https://pip.pypa.io/en/stable/reference/pip_freeze/

The guidelines provided here are intended to improve the readability of code and make it consistent across the wide spectrum of Python code.

From PEP 8 -- Style Guide for Python Code
https://www.python.org/dev/peps/pep-0008/#id15

Linting - Do you code with Style?

- Linters check code for syntax errors, bugs, style, and other stuff
- Flake8 reviews code for PEP8
- End lines with **# noqa** to bypass linting

```
$ flake8 example1.py

example1.py:9:1: E302 expected 2 blank lines, found 1

example1.py:13:67: E251 unexpected spaces around keyword / parameter equals

example1.py:13:69: E251 unexpected spaces around keyword / parameter equals

example1.py:15:1: E302 expected 2 blank lines, found 1

example1.py:31:1: E305 expected 2 blank lines after class or function definition, found 1
```

https://pypi.org/project/flake8/
https://www.python.org/dev/peps/pep-0008/#id15
https://en.wikipedia.org/wiki/Lint_(software)

Code File

Black and White – Auto Code Formatting

- Manually fixing Flake errors tedious and repetitive
- Code formatters automatically fix style errors
- **Black**: “the uncompromising Python code formatter.”
- **White**: “Black, but White instead (PEP8 line-lengths)”

```
$ white example1.py

reformatted example1.py

All done! ✨ 🍰 ✨

1 file reformatted.

-----------------------------

$ flake8 example1.py


# No errors, no output!
```

https://pypi.org/project/black/
https://pypi.org/project/white/

Code File

Licenses – Sharing the Open Source Way

- *Always consult proper legal representatives for advice, and abide by corporate policies.*
- Be explicit when you share code, include a license
  - Typical practice name file **LICENSE**
- Good information at https://choosealicense.com

"For your repository to truly be open source, you'll need to license it so that others are free to use, change, and distribute the software."

https://help.github.com/en/articles/licensing-a-repository

More Reusable Code

Functions – Stop Repeating that Code!

- Often you'll find yourself repeating processes in code
- Seems quick and simple at first
- Quickly bloats to redundant and repetitive code
- Use functions instead

Before

```
# URL for Host Calls
url = "https://{}/api/v1/host".format(dnac["host"])
```

```
# Get Host Infomration for Source
source_url = url + "?" + "&hostIp={}".format(source_ip)


# Make API request and return the response body
response = requests.request("GET", source_url,
                            headers=headers, verify=False)
source_host = response.json()["response"][0]
```

```
# Get Host Infomration for Destination
destination_url = url + "?" + "&hostIp={}".format(destination_ip)


# Make API request and return the response body
response = requests.request("GET", destination_url,
                            headers=headers, verify=False)
destination_host = response.json()["response"][0]
```

Code File

https://docs.python.org/3/tutorial/controlflow.html#defining-functions

Functions – Stop Repeating that Code!

- Build a function with the programming logic
- Call function each time needed

After

```
def host_list(dnac, ticket, ip=None):
    url = "https://{}/api/v1/host?hostIp={}".format(dnac, ip)
    headers["x-auth-token"] = ticket
    filters = []

    # Make API request and return the response body
    response = requests.request("GET", url,
                                headers=headers, verify=False)
    return response.json()["response"]
```

```
# Retrieve Host Details from dnac
source_host = host_list(dnac["host"], token, ip=source_ip)
destination_host = host_list(dnac["host"], token, ip=destination_ip)
```

Code

https://docs.python.org/3/tutorial/controlflow.html#defining-functions

Refactor into Modules and Import for reuse!

- Python Module = File
- Functions and static variables often needed across many scripts
- Examples:
  - Login functions
  - Device Details
- Put common resources into a module(s) and import

```
# Import  and functions
from dnac_resources import dnac
from dnac_functions import (
    dnac_login,
    host_list,
    verify_single_host,
    print_host_details,
    network_device_list,
    interface_details,
    print_network_device_details,
    print_interface_details
)
```

https://docs.python.org/3/tutorial/modules.html

Code

Packages – Easier than you think

- Python Packages not much more than folders
- Leverage them to organize your code
- Requires a __init__.py file in folder
  - Import elements from package for easy reference

```
$ ls -l
total 8
drwxr-xr-x  dnac
-rw-r--r--  host_troubleshooting.py

$ ls -l dnac/
total 40
-rw-r--r-- __init__.py
-rw-r--r-- dnac_functions.py
-rw-r--r-- dnac_resources.py
```

```
"""Basic package for interacting with Cisco DNA Center"""

from .dnac_resources import dnac
from .dnac_functions import (
    dnac_login,
    host_list,
    verify_single_host,
    print_host_details,
    network_device_list,
    interface_details,
    print_network_device_details,
    print_interface_details
)
```

https://docs.python.org/3/tutorial/modules.html#packages

Code

Packages – Easier than you think

- Python Packages not much more than folders
- Leverage them to organize your code
- Requires a **__init__.py** file in folder
  - Import elements from package for easy reference

Using Package in your Scripts

```
# Import  and functions
from dnac import (
    dnac,
    dnac_login,
    host_list,
    verify_single_host,
    print_host_details,
    network_device_list,
    interface_details,
    print_network_device_details,
    print_interface_details
)
```

https://docs.python.org/3/tutorial/modules.html#packages

Code

Classes/Objects – Packaging up your code (a bit more advanced)

Bonus!

- Objects combine methods and properties together
- Simplify use of code
  - *Though it may not be immediately obvious…*
- ***With object, no need to repeat the host and token with each call***

Without Object:

```
# Log into the dnac Controller to get Ticket
token = dnac_login(
    dnac["host"], dnac["port"],
    dnac["username"], dnac["password"]
)

# Retrieve Host Details from dnac
source_host = host_list(dnac["host"], token, ip=source_ip)
destin_host = host_list(dnac["host"], token, ip=destin_ip)
```

With Object:

```
# Initialize Cisco DNA Center Object
dnac = DNAC(
    address=args.dnac,
    port=args.port,
    username=args.username,
    password=args.password,
)

# Retrieve Host Details from dnac
source_host = dnac.host_list(ip=source_ip)
destin_host = dnac.host_list(ip=destin_ip)
```

https://docs.python.org/3/tutorial/classes.html

Code

More Robust Code

Try… Except… – Because it doesn't always work

- Even working code, sometimes doesn't
- API calls, file operations, user input – All points where unexpected errors can come in
- Reasons to catch errors
  - The error's and messages from Python often not clear for users
  - Or you might be able to recover from errors

```
def dnac_login(dnac, username, password):
    """
    Use the REST API to Log into an DNA Center and retrieve ticket
    """
    url = "https://{}/dna/system/api/v1/auth/token".format(dnac)

    # Make Login request and return the response body
    response = requests.post(url, auth=(username, password),
                             headers=headers, verify=False
    )
    return response.json()["Token"]
```

```
$ python without_try.py sandboxdnac2.cisco.com \
                 devnetuser \
                 'Cisco123'     # Wrong Password

Traceback (most recent call last):
  File "without_try.py", line 44, in <module>
    token = dnac_login(dnac, username, password)
  File "without_try.py", line 22, in dnac_login
    return response.json()["Token"]
KeyError: 'Token'
```

Code

https://docs.python.org/3/tutorial/errors.html#handling-exceptions

Try… Except… - Because it doesn't always work

```
def dnac_login(dnac, username, password):
    """
    Use the REST API to Log into an DNA Center and retrieve ticket
    """
    url = "https://{}/dna/system/api/v1/auth/token".format(dnac)


    # Make Login request and return the response body
    try:
        response = requests.request(url, auth=(username, password),
                                    headers=headers, verify=False
        )
    except requests.exceptions.ConnectionError:
        print("Unable to connect to address https://{}".format(dnac))
        exit(1)


    # Return the Token
    try:
        return response.json()["Token"]
    except KeyError:
        print("No token found in authentication response.")
        print("Response body: ")
        print(response.text)
        exit(1)
```

https://docs.python.org/3/tutorial/errors.html#handling-exceptions

Try/Except Block

- Before "risky" code – try
- After except Exception
- exit(1) to indicate error after printing message.

```
$ python with_try.py sandboxdnac2.cisco.com \
                     devnetuser \
                     'Cisco123'

No token found in authentication response.
Response body:

{"error":"Authentication has failed. Please provide valid credentials."} '
```

Code

Test returned data and status early

- It's very bad practice to "assume" details in code
- "Trust but verify" is a great motto.
- Great things to check
  - API Status Codes
    - HTTP 200 OK
    - NETCONF <ok />
  - Key exists in dictionary/JSON
  - File does(n't) exist
- exit() early if something isn't right
  - Pass non-0 value to indicate error

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

```
def dnac_login(dnac, username, password):
    url = "https://{}/dna/system/api/v1/auth/token".format(dnac)

    # Make Login request and return the response body
    try:
        response = requests.request(
            "POST",
            url,
            auth=(username, password),
            headers=headers,
            verify=False,
        )
    except requests.exceptions.ConnectionError:
        print("Unable to connect to address https://{}".format(dnac))
        exit(1)

    if response.status_code != 200:
        print(
            "Login request failed.  Status Code {}".format(
                response.status_code
            )
        )
        print("Response body: ")
        print(response.text)
        exit(1)
```

Code

Simple Command Line Tools with argparse

- Argparse is part of core Python
- Make scripts more flexible with arguments
- Positional, optional, defaults, etc
- Automatic "-h" help options
- For more advanced CLI tools look at Click

```
# Script Entry Point
if __name__ == "__main__":
    # Use Arg Parse to retrieve device details
    import argparse
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--host", help="Host address for network device", required=True
    )
    parser.add_argument(
        "--port", help="Override default NETCONF port of 830", default=830
    )
    parser.add_argument("--username", help="Device username", required=True)
    parser.add_argument("--password", help="Device password", required=True)
    args = parser.parse_args()

    print("Getting route list from device...")
    print("")

    # Get route list
    routes = get_ipv4_default_rib(
        host=args.host,
        port=args.port,
        username=args.username,
        password=args.password,
    )
```

```
$ ./argparse_example.py --host ios-xe-mgmt.cisco.com --port 10000 \
                --username root --password 'D_Vay!_10&'
```

https://docs.python.org/3/howto/argparse.html

Code

Make use of Environment Variables

- Available in Linux, macOS, Windows (basically everywhere)
- Programs dynamically update
- Part of 12 Factor App
- Great for "secrets"
  - Security tokens & Keys
  - IP Addresses
  - Username/Passwords
- Use "source files"

```
import os

# If Username or Password not provided as arguments, check OS ENV
if username is None:
    username = os.getenv("USERNAME")
if password is None:
    password = os.getenv("PASSWORD")


if username is None or password is None:
    print(
        "You must provide a username and password as a command argument,"
    )
    print("or as Environment Variables of USERNAME or PASSWORD")
    exit(1)
```

```
$ export USERNAME=root
$ export PASSWORD='D_Vay!_10&'

$ echo $PASSWORD
D_Vay!_10&

$ ./argparse_example.py --host ios-xe-mgmt.cisco.com --port 10000
```

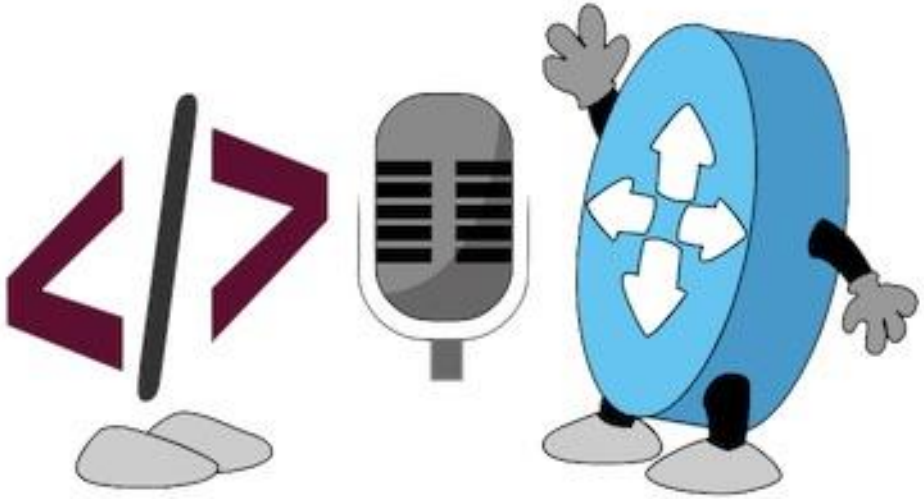
https://docs.python.org/3/library/os.html?highlight=os.getenv#os.getenv
https://en.wikipedia.org/wiki/Environment_variable

Code

Summing up

What did we talk about?

- Better Code Style / Sharing
  - Sh-Bang, Documents and Requirements, Linting, Licensing
- More Reusable Code
  - Functions, Modules, & Packages, Oh My!
- More Robust Code
  - Trying, Testing, CLI Tools, and the Environment



Webinar Resource List

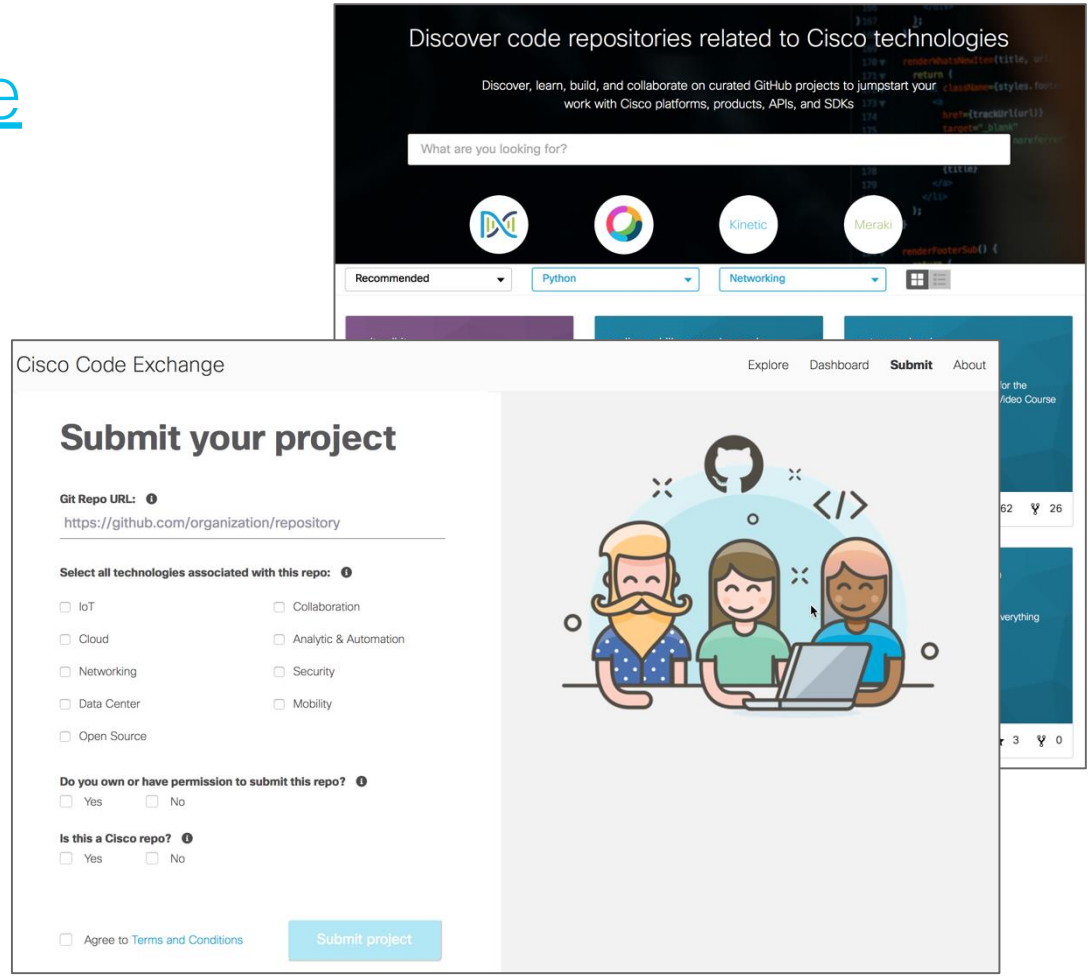
- Docs and Links
  - https://developer.cisco.com/python
- Learning Labs
  - Laptop Setup http://cs.co/lab-dev-setup
  - Coding Fundamentals http://cs.co/lab-coding-fundamentals
  - Model Driven Programmability http://cs.co/lab-mdp
- DevNet Sandboxes
  - Cisco DNA Center Always On http://cs.co/sbx-dnac-ao
  - IOS Always On http://cs.co/sbx-iosxe
- Code Samples
  - http://cs.co/code-python-networking

NetDevOps Live! Code Exchange Challenge

developer.cisco.com/codeexchange

Leverage one or more of the suggestions shown in an active network automation project of yours and submit to Code Exchange!

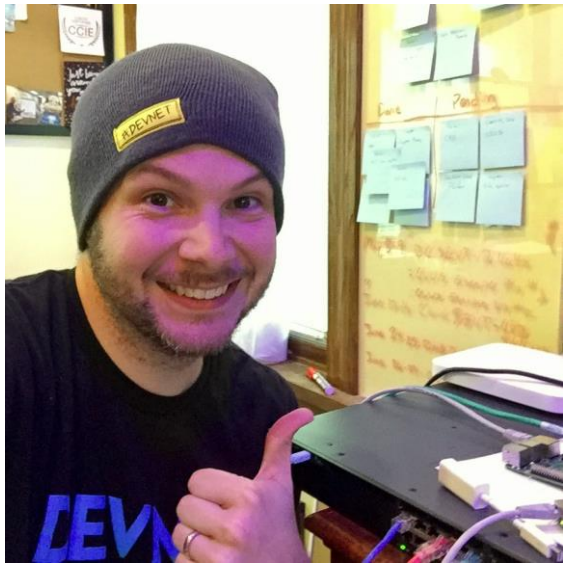
Example: Move your functions into modules (ie other files) and import them into main script for better modularity.



Looking for more about NetDevOps?

- NetDevOps on DevNet
  developer.cisco.com/netdevops
- NetDevOps Live!
  developer.cisco.com/netdevops/live
- NetDevOps Blogs
  blogs.cisco.com/tag/netdevops
- Network Programmability Basics Video Course
  developer.cisco.com/video/net-prog-basics/

Got more questions? Stay in touch!



Hank Preston

hapresto@cisco.com
@hfpreston
http://github.com/hpreston

CISCO DEVNET

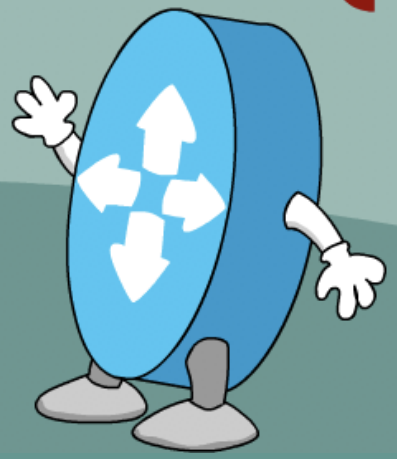
LEARN CODE INSPIRE CONNECT

developer.cisco.com

@CiscoDevNet
facebook.com/ciscodevnet/
http://github.com/CiscoDevNet

NETDEVOPS {LIVE!}

CISCO DEVNET



https://developer.cisco.com/netdevops/live

@netdevopslive