



DEVNET

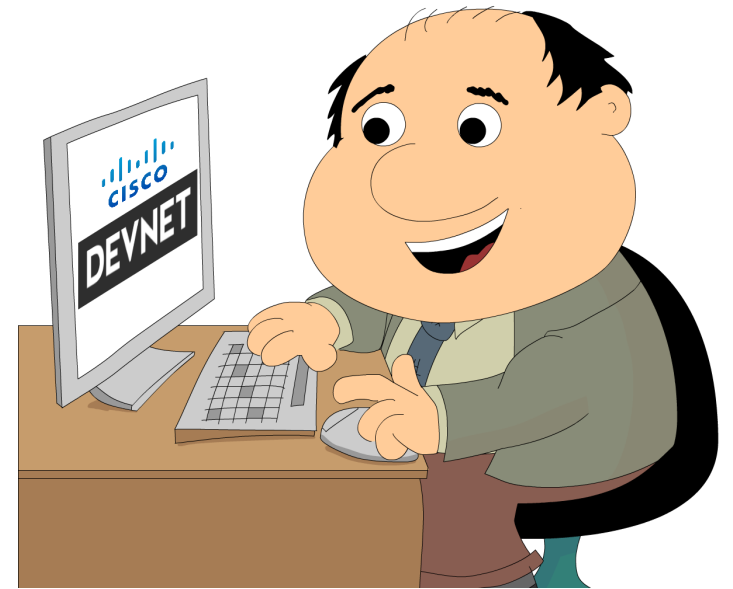
Python Part 1: Python Language and Script Basics

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

Module: Programming Fundamentals

- 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

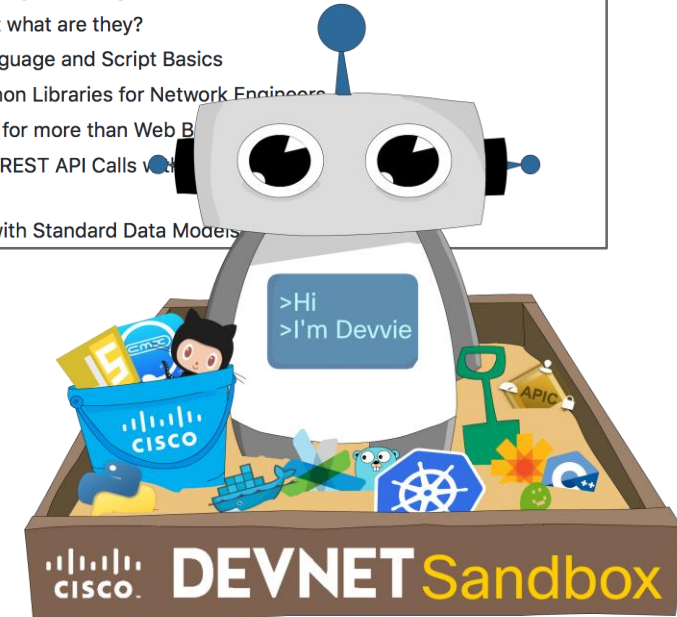
- Get the Code!
 - github.com/CiscoDevNet/netprog_basics
- Setup Lab Prerequisites
 - Each lab includes a README with details
- Access to Infrastructure
 - [DevNet Sandbox](#)
 - Specifics in lab README

Network Programmability Basics

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

Table of Contents

- **Programming Fundamentals**
 - Data Formats: Understanding and using JSON, XML and YAML
 - APIs are Everywhere... but what are they?
 - Python Part 1: Python Language and Script Basics
 - Python Part 2: Useful Python Libraries for Network Engineers
 - REST APIs Part 1: HTTP is for more than Web Browsers
 - REST APIs Part 2: Making REST API Calls with Python
- **Network Device APIs**
 - Getting the "YANG" of it with Standard Data Models



Topics to Cover

- Why Python and How to get it?
- Breaking Down Python Code
- Interacting with Python Live

Why Python and
How to get it?

Python is powerful... and fast;
plays well with others;
runs everywhere;
is friendly & easy to learn;
is Open.



Source: <http://www.python.org/about>

The Zen of Python

by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one -- and preferably only one -- obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

The Zen of Python

by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

Why Python for Network Engineers

- Readable and easy to learn
- Widely available and Open Source
 - Windows, Mac, Linux
 - Routers & Switches
- Many relevant code samples
- Lots of training resources



How to get Python?

- You might already have it
- <http://python.org/downloads>
- Package Management Tools
 - Example: Homebrew for Mac
 - brew install python2
 - brew install python3

```
DevNet$ python -version  
Python 2.7.12
```

```
DevNet$ python  
Python 2.7.12 (default, Oct 11 2016, 05:20:59)  
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-  
800.0.38)] on darwin  
Type "help", "copyright", "credits" or "license"  
for more information.  
>>>
```

Breaking Down Python Code

example1.py

- Script Structure and Format
- Importing and using packages
- Variable declaration and usage
- Function creations and usage
- Basic Error Handling

```
#!/usr/bin/env python
"""
Learning Series: Network Programmability Basics
Module: Programming Fundamentals
Lesson: Python Part 1
Author: Hank Preston <hapresto@cisco.com>
```

example1.py

- Script Structure and Format
- Importing and using packages
- Variable declaration and usage
- Function creations and usage
- Basic Error Handling

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

```
"""
```

```
Learning Series: Network Programmability Basics
```

```
Module: Programming Fundamentals
```

```
Lesson: Python Part 1
```

```
Author: Hank Preston <hapresto@cisco.com>
```

```
example1.py
```

```
Illustrate the following concepts:
```

- Script Structure and Format
- Importing and using packages
- Variable declaration and usage
- Function creations and usage
- Basic Error Handling

```
"""
```

```
__author__ = "Hank Preston"
```

```
__author_email__ = "hapresto@cisco.com"
```

```
__copyright__ = "Copyright (c) 2016 Cisco Systems, Inc."
```

```
__license__ = "MIT"
```

```
import sys
```

example1.py

- Script Structure and Format
- Importing and using packages
- Variable declaration and usage
- Function creations and usage
- Basic Error Handling

```
def doubler(number):  
    """  
    Given a number, double it and return the value  
    """  
    result = number * 2  
    return result  
  
# Entry point for program  
if __name__ == '__main__':  
    # Retrieve command line input  
    try:  
        input = float(sys.argv[1])  
    except (IndexError, ValueError) as e:  
        # Indicates no command line parameter was provided  
        print("You must provide a number as a parameter to this script")  
        print("Example: ")  
        print("  python example1.py 12")  
        sys.exit(1)  
  
    # Double the provided number and print output  
    answer = doubler(input)  
    print(answer)
```

example1.py – Testing the Code

```
DevNet$ python example1.py
You must provide a number as a parameter to this script
Example:
    python example1.py 12

DevNet$ python example1.py 5
10.0

DevNet$ python example1.py 45.4
90.8
```


example2.py

- Reading from and writing to files
- The “with” statement
- Requesting interactive user input
- Writing to the command line

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

__author__ = "Hank Preston"
__author_email__ = "hapresto@cisco.com"
copyright = "Copyright (c) 2016 Cisco Systems, Inc."
```

example2.py

- Reading from and writing to files
- The “with” statement
- Requesting interactive user input
- Writing to the command line

```
from datetime import datetime

log_file = "example2.log"

def read_log(log):
    """
    Open the logfile and print contents to the terminal
    """
    with open(log, "r") as f:
        print(f.read())

def write_log(log, name):
    """
    Add new logfile entry with timestamp
    """
    # Get current date and time
    log_time = str(datetime.now())
    with open(log, "a") as f:
        f.writelines("Entry logged at: {} by {}\n".format(log_time, name))
```

example2.py

- Reading from and writing to files
- The “with” statement
- Requesting interactive user input
- Writing to the command line

```
# Entry point for program
if __name__ == '__main__':
    # Get users name
    name = input("What is your name? ")

    # Add entry to log file
    print("Adding new log entry")
    write_log(log_file, name)
    print("")

    # Access Starting Log File
    print("Log File Contents")
    print("-----")
    read_log(log_file)
```

example2.py – Testing the Code

```
DevNet$ python example2.py  
What is your name? Hank  
Adding new log entry
```

```
Log File Contents
```

```
-----
```

```
Entry logged at: 2017-07-25 23:11:16.168003 by Hank  
Entry logged at: 2017-07-26 00:17:03.199335 by Hank
```

example3.py

- Creating and using dictionaries
- Creating and using lists
- Working with for loops
- Conditional statements

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

__author__ = "Hank Preston"
__author_email__ = "hapresto@cisco.com"
__copyright__ = "Copyright (c) 2016 Cisco Systems, Inc."
__license__ = "MIT"
```

example3.py

- Creating and using dictionaries
- Creating and using lists
- Working with for loops
- Conditional statements

```
# Example Dictionary
author = {"name": "Hank", "color": "green", "shape": "circle"}

# A list of colors
colors = ["blue", "green", "red"]

# A list of dictionaries
favorite_colors = [
    {
        "student": "Mary",
        "color": "red"
    },
    {
        "student": "John",
        "color": "blue"
    }
]

# Entry point for program
if __name__ == '__main__':
    print("The author's name is {}".format(author["name"]))
    print("His favorite color is {}".format(author["color"]))
    print("")
```

example3.py

- Creating and using dictionaries
- Creating and using lists
- Working with for loops
- Conditional statements

```
# Entry point for program
if __name__ == '__main__':
    print("The author's name is {}".format(author["name"]))
    print("His favorite color is {}".format(author["color"]))
    print("")

    print("The current colors are:")
    for color in colors:
        print(color)
    print("")

    # Ask user for favorite color and compare to author's color
    new_color = input("What is your favorite color? ")
    if new_color == author["color"]:
        print("You have the same favorite as {}".format(author["name"]))
        print("")

    # See if this is a new color for the list
    if new_color not in colors:
        print("That's a new color, adding it to the list!")
        colors.append(new_color)
        # Print update message about the new colors list
        message = ("There are now {} colors in the list. ".format(len(colors)))
        message += "The color you added was {}".format(colors[3])
        print(message)
    else:
        pass
```

example3.py – Testing the Code part 1

```
DevNet$ python example3.py
The author's name is Hank.
His favorite color is green.

The current colors are:
blue
green
red

What is your favorite color? green
You have the same favorite as Hank.
```


example3.py – Testing the Code part 2

```
DevNet$ python example3.py  
The author's name is Hank.  
His favorite color is green.
```

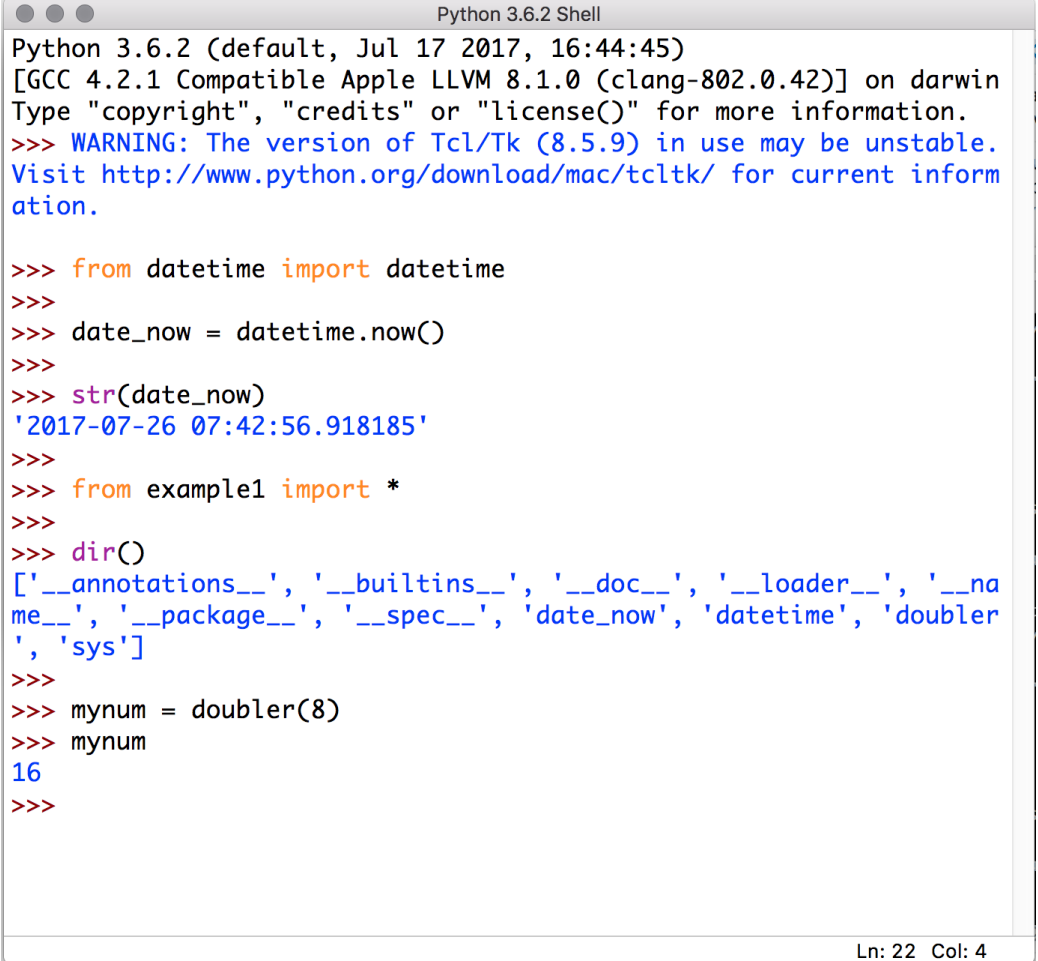
```
The current colors are:  
blue  
green  
red
```

```
What is your favorite color? orange  
That's a new color, adding it to the list!  
There are now 4 colors in the list. The color you added was orange.
```

Interacting with Python Live

What is and Why use Interactive Python

- Real-time enter code and see results
- Experiment and learning
- Debug scripts that aren't working
- Build on the output of a script



```
Python 3.6.2 Shell
Python 3.6.2 (default, Jul 17 2017, 16:44:45)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.

>>> from datetime import datetime
>>>
>>> date_now = datetime.now()
>>>
>>> str(date_now)
'2017-07-26 07:42:56.918185'
>>>
>>> from example1 import *
>>>
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'date_now', 'datetime', 'doubler', 'sys']
>>>
>>> mynum = doubler(8)
>>> mynum
16
>>>
```

Ln: 22 Col: 4

Accessing Interactive Python

- `python -i`
 - Basic command line
- `python -i script.py`
 - Run a script and leave session active
- `idle`
 - Basic interpreter client written in Python
 - *Virtual Environment*
`python -m idlelib.idle`

```
DevNet$ python -i
Python 3.6.2 (default, Jul 17 2017, 16:44:45)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-
802.0.42)] on darwin
Type "help", "copyright", "credits" or "license"
for more information.
>>> from pprint import pprint
>>> pprint("Hello World")
'Hello World'
>>> exit()

DevNet$ python -i example1.py 21
42.0
>>> doubler(3)
6
>>> exit()

DevNet$ idle
```

Helpful Interactive Python Commands

- `dir()`
 - Return all variable, classes, objects (collectively called “names”) available
- `dir(name)`
 - Return attributes for an object
- `help(name)`
 - Built-in help system. Displays docs and info for object

```
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']
>>> from datetime import datetime
>>> import example1
>>>
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'datetime', 'example1']
>>> dir(example1)
['__author__', '__author_email__', '__builtins__', '__cached__', '__copyright__', '__doc__', '__file__', '__license__', '__loader__', '__name__', '__package__', '__spec__', 'doubler', 'sys']
>>>
>>> print(example1.__author__)
Hank Preston
>>>
>>> help(example1.doubler)
Help on function doubler in module example1:

doubler(number)
    Given a number, double it and return the value

>>>
>>> mynum = example1.doubler(4)
>>> mynum
8
>>> |
```

Demo Time!



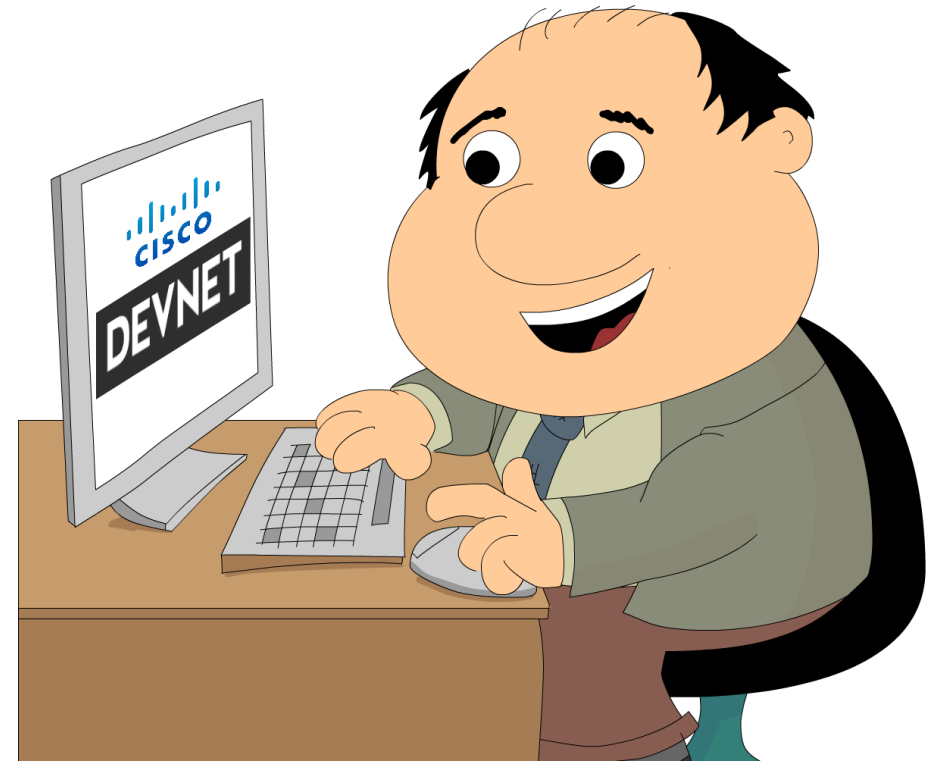
Summing up

Review

- Why Python and How to get it?
- Breaking Down Python Code
 - Script structure, imports, variables, functions & error handling
 - Working with files and command line input
 - Using lists and dictionaries, loops, and conditional statements
- Interacting with Python Live
 - How to access the Python interpreter
 - Using `dir()` and `help()` to explore and learn

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](https://twitter.com/hfpreston)

 <http://github.com/hpreston>

 [@CiscoDevNet](https://twitter.com/CiscoDevNet)

 facebook.com/ciscocodevnet/

 <http://github.com/CiscoDevNet>





DEVNET