Module 0. Set Up Your Environment

(This has been modified to for the Federal SE Q3FY17 KSO)

The goal of this module is to prepare your laptop for the *DevNet Express for DNA* learning track.

Objectives

Completion Time: 60 minutes

- Ensure Python is installed
- Ensure network connectivity
- Register for accounts (such as Cisco Spark)
- Make sure required tools are available

Prerequisites

None

Step 1. Decide on the Scenario You Want to Set Up

There are 5 possible scenarios. The scenario that you prepare for depends on the lab environment and development environment you want to use and whether you are attending a hands-on event.

Decide which scenario makes the most sense for your situation:

Scenario	Setup Description	Minimum Prep Required
#1	Attend a hands-on event, use dev tools in dCloud	• Install either: • a browser that supports HTML5 like Chrome or Firefox, -or- • install AnyConnect and RDP client • Obtain CCO account
#2	Attend a hands-on event, use dev tools on your local machine	• Install AnyConnect • Obtain a CCO account • Run the connectivity test • Install required tools on your local machine
#3	Work from home, use dev tools in dCloud	• Install a browser that supports HTML5 like Chrome or Firefox -or- • install AnyConnect and RDP client • Obtain CCO account

Scenario	Setup Description	Minimum Prep Required
		• Run the connectivity test
#4	Work from home, use dev tools on your local machine	• Install AnyConnect • Run the connectivity test • Install required tools on your local machine
#5	Work from home, use your local machine and your own lab environment	• Install required tools • provide your own routers, controllers, and instrastructure

The following pages explain in more detail the steps you need to prepare each scenario.

Should You Use Dev Tools On Your Local Machine or on the Remote dCloud Workstations?

Local Machine

Why use your local machine? You can choose to run the development tools (such as Python and Postman) on your own local machine (such as a laptop). You might do this if you want to have all the dev tools available after the class. This could take an hour or so to set up depending on how many of the tools (such as Python and Postman) you already have on your machine. This can be slightly challenging if you have never installed any of these tools before.

In this case, code will be developed and run on your local machine. To have the local code 'talk' to the remote infrastructure in dCloud (like APIC-EM or routers), a VPN connection is required.

Step 2. Install Additional Tools

The tools listed on this page are useful for completing the labs. Depending on your setup you might already have some of these tools.

Cisco Spark

(Required) Download and install the Cisco Spark client for your platform. Native clients are available for Windows and Mac (and Android and iOS). If on Linux (or if you don't want to install the native client) you can use the Web client.

Chrome

(Optional) Chrome is an HTML5-enabled browser. Install Chrome if you want to use it to access the dCloud pod through remote desktop.

Postman

(Required) Postman is **the** tool if it comes to testing and exploring REST APIs. Either install a stand-alone version or install it as an extension into Chrome.

RDP Client

(Optional) Install a remote desktop client. Various choices for all platforms are available. This is required if you use the dCloud pod and can not or do not want to use the browser-based RDP session.

Text Editor

(Required) You will need a text editor to edit source code files. This varies greatly across platforms and also personal preferences. So this should be considered just as a recommendation. Just make sure that your text editor of choice saves text files and not some binary format.

- **Windows:** Atom, Sublime Text or Notepad++ are some choices to consider. NotePad++ is a lightweight and free option to consider if you don't have any preference at all.
- **Mac OS X:** Again, Atom and Sublime Text are powerful choices. Another option is TextWrangler.
- **Linux** Again, Atom and Sublime Text and plenty of other options like gedit or Kate.

Git Client

(Required) A Git source code management client is also required to clone the code samples and other software. There's a variety of options for graphical front ends and, of course, you could stick with CLI-only as well. Start at the git-scm.com downloads site which has a free client for all major platforms from git-scm.com and also lists plenty of alternative / third party GUI clients for all kinds of platforms.

Alternatively, you can also install the GUI version from https://desktop.github.com.

Make sure to include the Git tools into your path during installation. Also note that changes to your path / environment are not immediately effective. You need to open a new shell or maybe even restart your computer.

```
C:\Users\userid>echo %PATH%
C:\Python35\;C:\Python35\Scripts\;C:\Python27\Scripts;C:\Windows\system32;C:\
WinndowsPowerShell\v1.0\;C:\Program Files (x86)\PuTTY\;C:\Program Files
(x86)\WindoFiles\Git\cmd
```

```
C:\Users\userid>
```

The above output shows Git included in the path (the last entry)

SSH / Scp

(Required) If you don't have it already (most Linux and Mac OS X users should be good to go) then we also recommend to install a SSH suite. Putty is a popular choice on Windows.

Another powerful Windows tool is WinSCP which integrates nicely into Windows and allows for easy and secure file transfers.

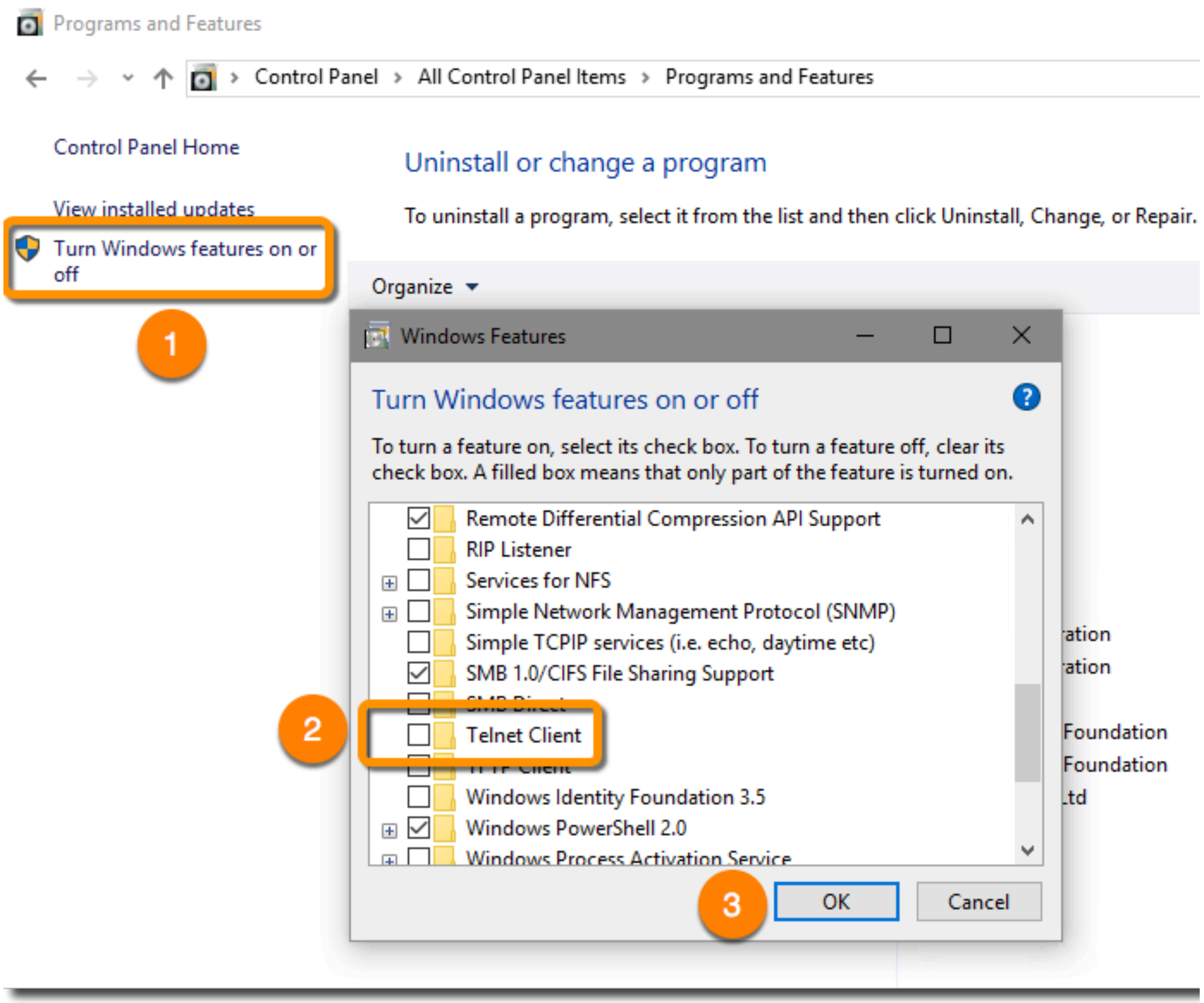
ZIP tools

(Recommended) Install a ZIP tool of your choice. 7-Zip or Winzip are popular choices on Windows. Mac users have one installed by default (`zip` and `unzip`). Linux users might need to install those tools manually:

```
$ apt-cache search zip | grep -e '^zip -' -e '^unzip -'
unzip - De-archiver for .zip files
zip - Archiver for .zip files
$
```

Telnet

(Optional) Yes, you heard it right... Telnet is still a useful tool to test connectivity to services on remote computers. It's **the** minimalistic 'bare-bones' browser and connectivity tester. Mac OS X has it installed per default. On Windows you'd have to install it from the Windows installation media:


Programs and Features

Control Panel > All Control Panel Items > Programs and Features

Control Panel Home

View installed updates

Turn Windows features on or off

Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

Organize ▾

Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- [✓] Remote Differential Compression API Support
- [ ] RIP Listener
- [ ] Services for NFS
- [ ] Simple Network Management Protocol (SNMP)
- [ ] Simple TCPIP services (i.e. echo, daytime etc)
- [✓] SMB 1.0/CIFS File Sharing Support
- [ ] SMB Direct
- [ ] Telnet Client
- [ ] TFTP Client
- [ ] Windows Identity Foundation 3.5
- [✓] Windows PowerShell 2.0
- [ ] Windows Process Activation Service

Also, on Linux it often is not installed as part of a default installation and you have to install it manually:

```
$ apt-cache search telnet | grep '^telnet -'
telnet - The telnet client
$ sudo apt-get install -y telnet
```

Cygwin

(Optional) Cygwin *is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows.* While entirely optional, Cygwin provides a lot of powerful CLI tools which can make your life as a programmer a lot easier if you are eager to pick up and learn a bit of *nix know-how.

Wireshark

(Recommended) Some of the labs require a packet capture tool. You can either use tcpdump (on a Mac or on Linux) or use the full-blown GUI tool which is Wireshark. Wireshark is available on all platforms and can be downloaded from https://www.wireshark.org/

Step 3. Install Dev Tools (Python, PIP, etc.) On Your Local Machine

Depending on your OS, you might already have a Python environment installed. Mac OS X has Python already installed. Most Linux distributions also have Python installed. The examples in the labs are compatible with Python versions 2.7 and 3.x.

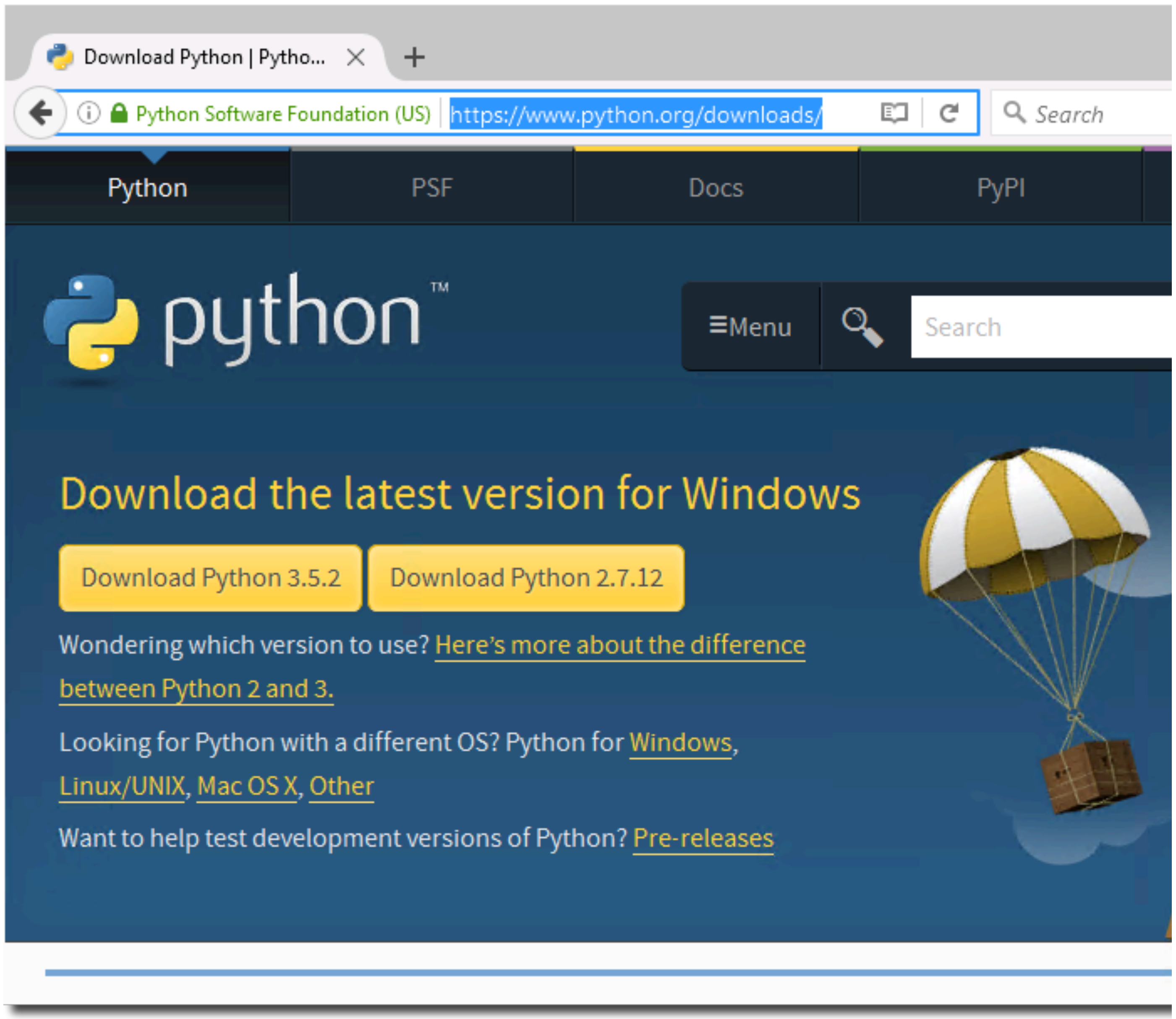
If you do not have Python installed, follow the instructions below to install it.

After you have ensured that Python is on your machine, read through the **PIP** and **Virtual Environments** sections further below. Those sections explain how to install the required Python packages and how to create a virtual Python environment. Having a virtual Python environment is considered a best practice because it helps to prevent *poisoning* your global Python environment.

Get The Python Installer

Download the Python installer:

1. Go to https://python.org
2. Navigate to the download page


Download Python | Pytho...
Python Software Foundation (US)
https://www.python.org/downloads/
Search
Python
PSF
Docs
PyPI
python™
≡Menu
Search

Download the latest version for Windows

Download Python 3.5.2 Download Python 2.7.12

Wondering which version to use? Here's more about the difference between Python 2 and 3.

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases

Where to install?

There are several options:

- install it natively on your computer
- install it inside of a container (if you already have e.g. a Docker environment)
- install it inside of a VM (Vagrant is a nice tool to get you up and running fast)
- install it inside of a VM that you install and maintain manually, for example using VirtualBox or VMware Workstation / Fusion

All of these approaches have pros and cons. If in doubt, ask a colleague or instructor what the best approach would be for your machine.

Installing on Windows OS

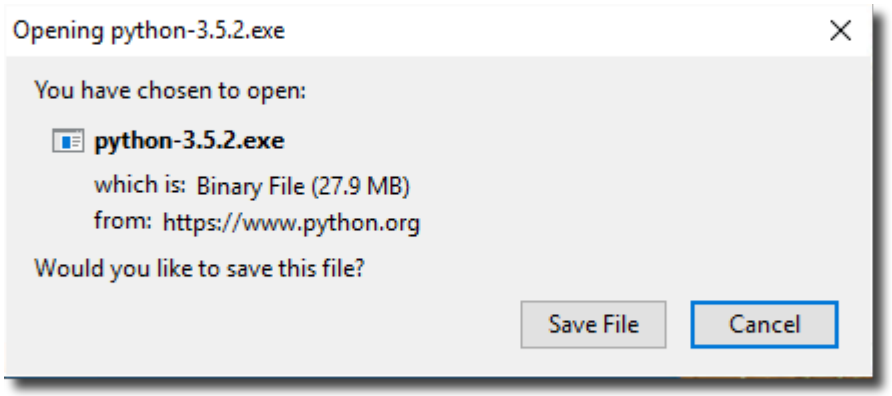
Install the version that matches your operating system version. For example, if you are running a 64-bit version of Windows then install the 64-bit version of Python.

If you plan to run multiple Python environments (for example running versions 2.7 and 3.5 in parallel), then installing in the root of your computer's harddrive makes adapting paths easier. For example, install to directories such as `C:\Python27` and `C:\Python35`.

You can change the directory name in the installer by selecting *'Customize installation (Choose location and features)'* in the first dialog shown by the installer.

Make sure to mark the 'Add Python to the environment' checkbox during the customization sequence before clicking 'Install'.

1. Download and run the installer



Opening python-3.5.2.exe

You have chosen to open:

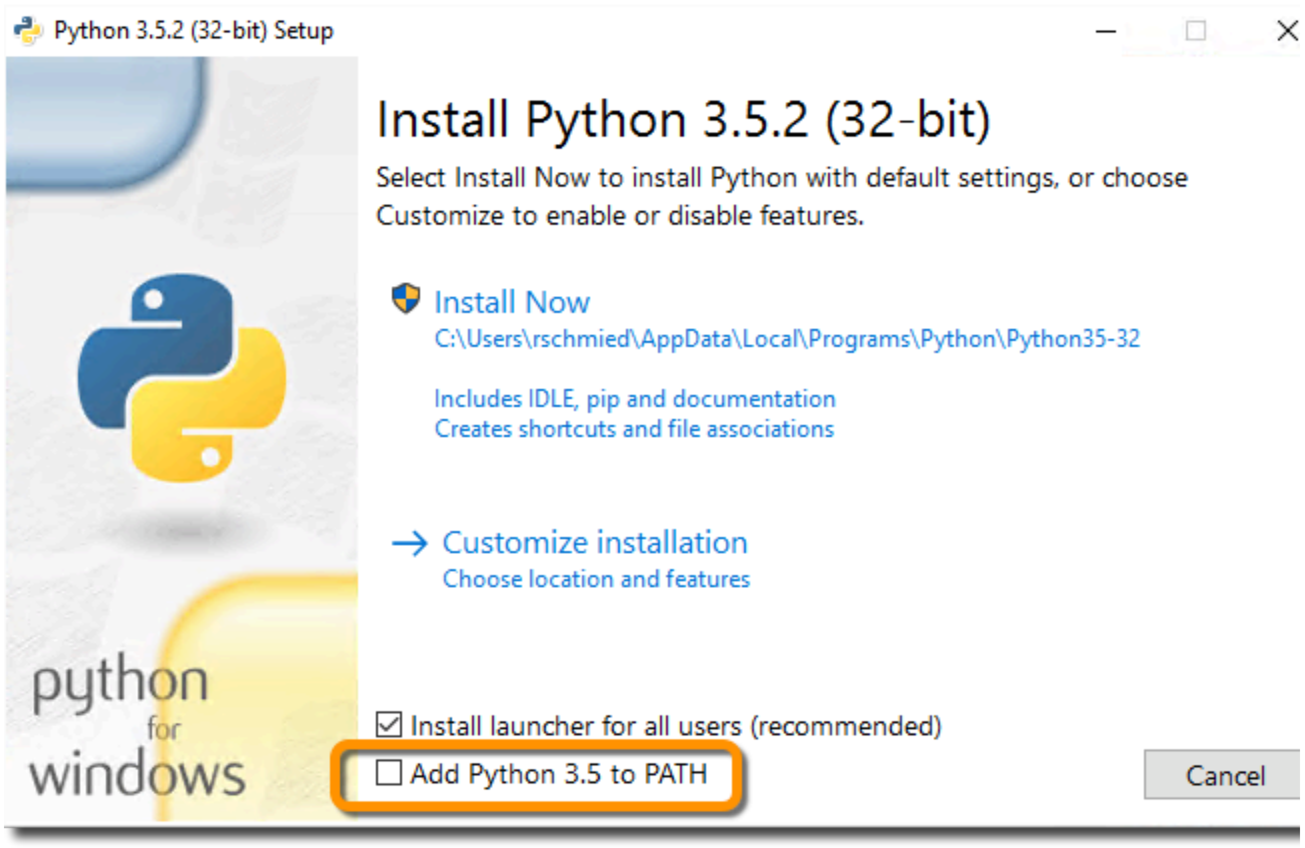
python-3.5.2.exe

which is: Binary File (27.9 MB)

from: https://www.python.org

Would you like to save this file?

2. Mark the *Add Python 3.5 to PATH* box



Python 3.5.2 (32-bit) Setup

Install Python 3.5.2 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

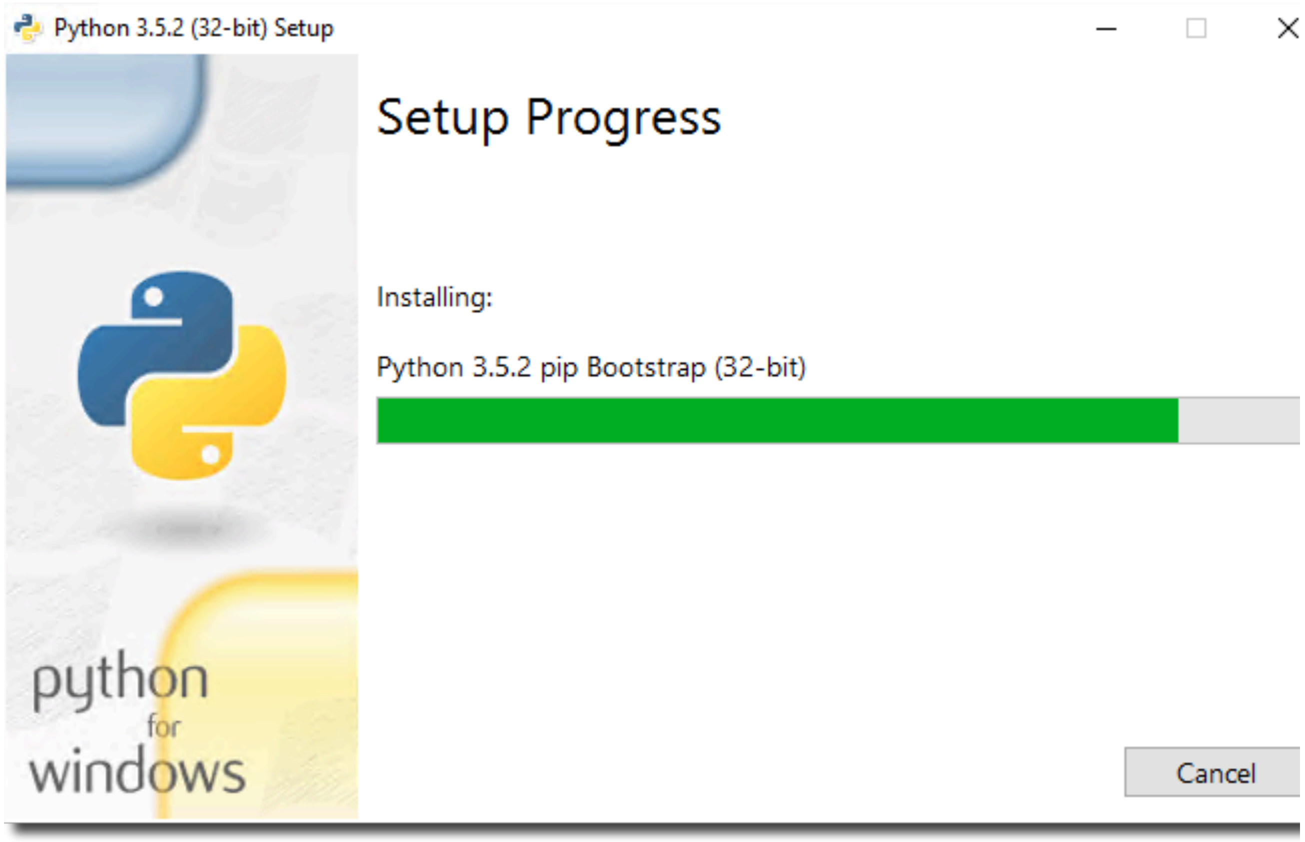
Install Now
C:\Users\rschmied\AppData\Local\Programs\Python\Python35-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[ ] Add Python 3.5 to PATH

3. Click 'Install Now', and proceed through the installation

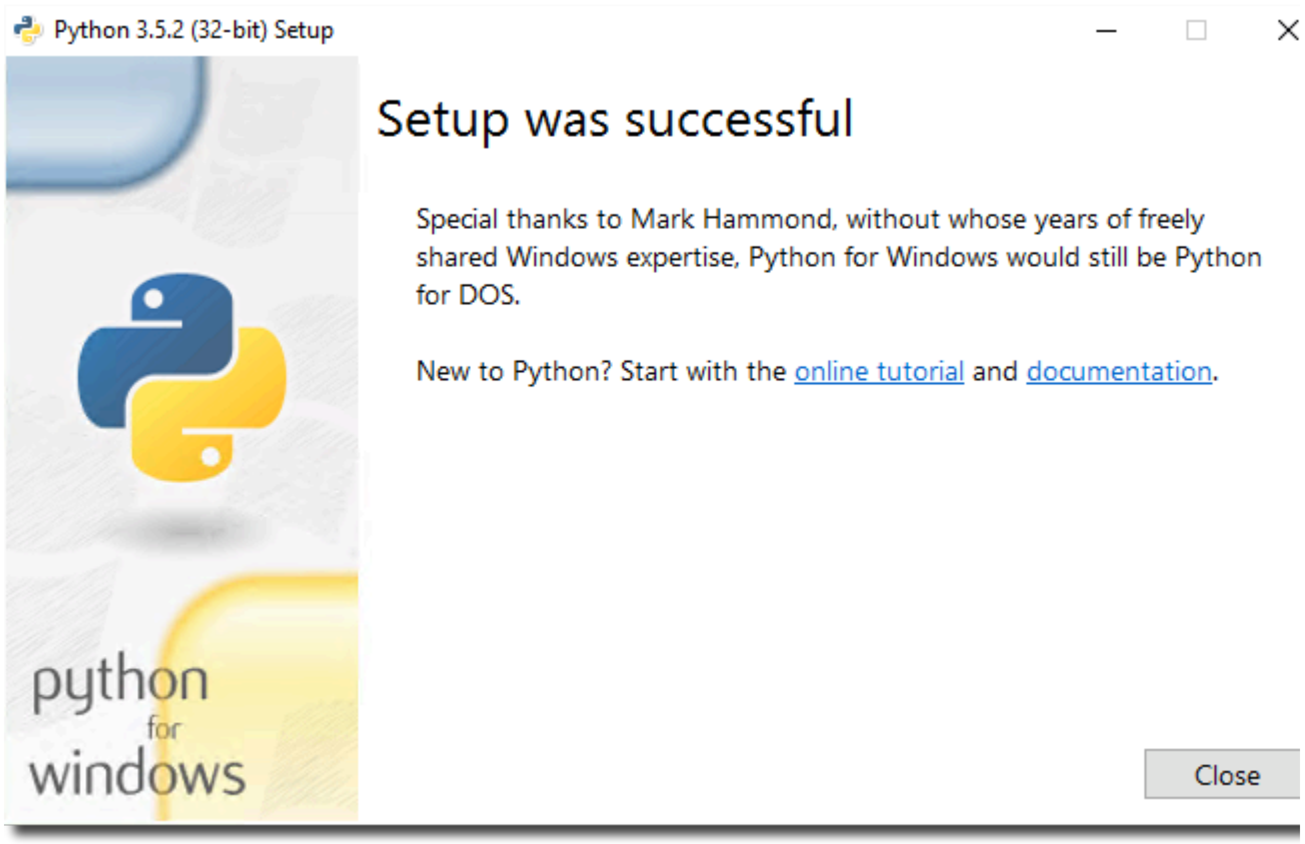

Python 3.5.2 (32-bit) Setup

Setup Progress

Installing:

Python 3.5.2 pip Bootstrap (32-bit)

4. Close the installer.

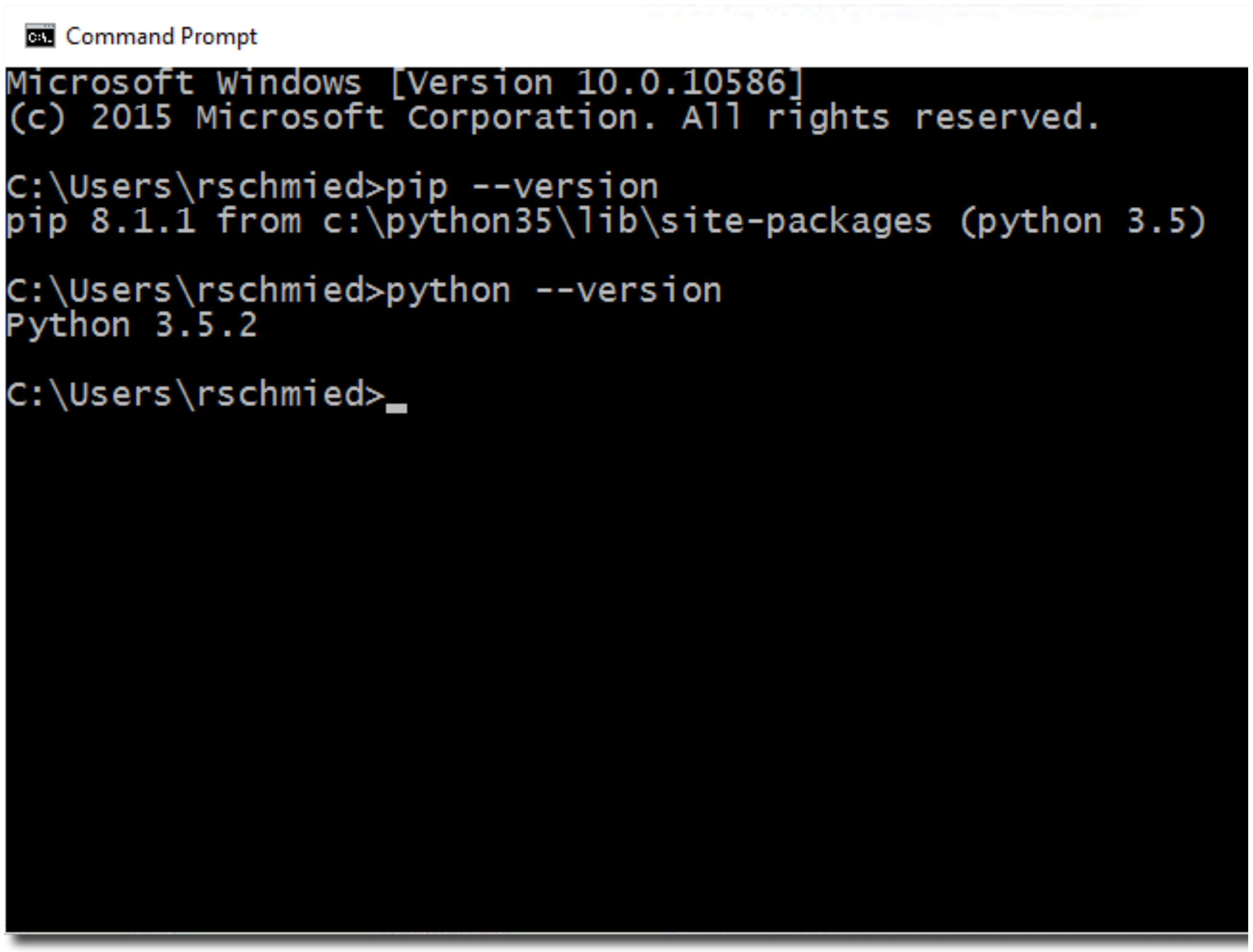


Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

5. Open a command window (`cmd.exe`)
6. On the command line, verify your PIP installation by typing `pip --version`
7. On the command line, verify the Python installation by typing `python --version`.



```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\rschmied>pip --version
pip 8.1.1 from c:\python35\lib\site-packages (python 3.5)

C:\Users\rschmied>python --version
Python 3.5.2

C:\Users\rschmied>_
```

Installing on Mac and Linux

Mac OS X and Linux typically come with the Python environment already installed. Both platforms come with Python version 2.7 as the default. If you are using on Mac OS X, and you want to upgrade to Python 3.5, go to https://python.org. Download the installation package.

If you use Linux, use your package manager to find the 'python3' package. The https://python.org site has no binary releases for *nix platforms as they typically are included with the platform package manager (like `apt` on Ubuntu or `yum` / `dnf` on RPM based systems).

Verify that your Python installation is properly installed:

1. Open a command line window (terminal)
2. Type: `python --version`

The following example shows the typical output with Python 2.7 installed on Mac OS X:

```
$ python --version
Python 2.7.10
```

```
$
```

The exact version depends on the source of your installation:

- Python that came with OS X,
- Homebrew or MacPorts -or-
- a binary package from https://python.org.

If you installed Python 3.5, the output will look like this:

```
$ python --version
Python 3.5.2
$
```

If multiple Python versions are installed at the same time then make sure to modify your path accordingly to ensure you use the Python version you want to use. You might also reference this document.

All Platforms

After you have Python installed on your local machine, install additional packages into the Python environment as described below.

The Uniq Package

Uniq is a Python API client library for Cisco's Application Policy Infrastructure Controller Enterprise Module (APIC-EM) Northbound APIs. Before July 2016, APIC-EM required Python version 2.7.x in order to work with the Uniq package. The package is now also available for Python version 3.5.x.

Install PIP

The `pip` tool is the Python package manager that interfaces with the Python Package index PyPI. We recommend that you use it to manage the installation of Python packages. A Python package is a collection of Python modules. Importing modules can be useful because they provide additional functionality to the Python environment.

PIP on Windows OS

If you installed Python on Windows, `pip` has already been installed as part of the installation (assuming you didn't disable it during installation). You can test its availability by typing in a command window:

```
pip --version
```

PIP on Mac OS X

On Mac OS X `pip` might not be installed. Check to see if PIP is installed:

1. On a command line (Terminal window), type: `pip --version`

```
2. $ python --version
3. Python 2.7.10
4. $ pip --version
5. -bash: pip: command not found
6. $
```

7. If the command is not found, you need to install PIP:

```
8. $ sudo easy_install pip
9. Password:
10.  Searching for pip
11.  Reading https://pypi.python.org/simple/pip/
12.  Best match: pip 8.1.2
13.  Downloading
   https://pypi.python.org/packages/e7/a8/7556133689add8d1a54c0b14aeff0acb
   03c64707ce100ecd53934da1aa13/pip-
   8.1.2.tar.gz#md5=87083c0b9867963b29f7aba3613e8f4a
14.  Processing pip-8.1.2.tar.gz
15.  Writing /tmp/easy_install-mEZelf/pip-8.1.2/setup.cfg
16.  Running pip-8.1.2/setup.py -q bdist_egg --dist-dir /tmp/easy_install-
   mEZelf/pip-8.1.2/egg-dist-tmp-VZMCVd
17.  warning: no previously-included files found matching '.coveragerc'
18.  warning: no previously-included files found matching '.mailmap'
19.  warning: no previously-included files found matching '.travis.yml'
20.  warning: no previously-included files found matching '.landscape.yml'
21.  warning: no previously-included files found matching
   'pip/_vendor/Makefile'
22.  warning: no previously-included files found matching 'tox.ini'
23.  warning: no previously-included files found matching 'dev-
   requirements.txt'
24.  warning: no previously-included files found matching 'appveyor.yml'
25.  no previously-included directories found matching '.github'
26.  no previously-included directories found matching '.travis'
27.  no previously-included directories found matching 'docs/_build'
28.  no previously-included directories found matching 'contrib'
29.  no previously-included directories found matching 'tasks'
30.  no previously-included directories found matching 'tests'
31.  Adding pip 8.1.2 to easy-install.pth file
32.  Installing pip script to /usr/local/bin
33.  Installing pip2.7 script to /usr/local/bin
34.  Installing pip2 script to /usr/local/bin
35.
36.  Installed /Library/Python/2.7/site-packages/pip-8.1.2-py2.7.egg
37.  Processing dependencies for pip
38.  Finished processing dependencies for pip
39.  $ pip --version
40.  pip 8.1.2 from /Library/Python/2.7/site-packages/pip-8.1.2-py2.7.egg
   (python 2.7)
41.  $
```

The `pip` tool is now available and can be used to install additional packages.

PIP on Linux

On Linux distributions, use the package manager to install the `pip` package. For example, on Ubuntu the package is called `python-pip`:

```
$ apt-cache search python-pip
python-pip - alternative Python package installer
```

Verification (All Platforms)

Verify that `pip` has been installed by looking at the packages that are available in your global Python environment.

1. Open a command line window
2. Type: `pip list`

The following example shows the typical expected output:

```
$ pip list
altgraph (0.10.2)
bdist-mpkg (0.5.0)
bonjour-py (0.3)
macholib (1.5.1)
matplotlib (1.3.1)
modulegraph (0.10.4)
numpy (1.8.0rc1)
pip (8.1.2)
py2app (0.7.3)
pyobjc-core (2.5.1)
...
pyobjc-framework-WebKit (2.5.1)
pyOpenSSL (0.13.1)
pyparsing (2.0.1)
python-dateutil (1.5)
pytz (2013.7)
scipy (0.13.0b1)
setuptools (1.1.6)
six (1.4.1)
xattr (0.6.4)
zope.interface (4.1.1)
$
```

The above example output shows output from pip installed on Mac OS X.

The following example shows the packages installed of a fresh Python installation on Windows:

```
C:\Users\userid\Code>pip list
pip (8.1.1)
setuptools (20.10.1)
You are using pip version 8.1.1, however version 8.1.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.

C:\Users\userid\Code>
```

Note that in this case Windows recommends to upgrade to a newer version of `pip` which can be done via the command given in the command line window.

With `pip` installed, you can now do things like search for a package, install or uninstall a package or update packages. We will need this capability to install the package `virtualenv` which is needed in the next section of this lab module:

```
$ pip search virtualenv | grep '^virtualenv ('
virtualenv (15.0.3)      - Virtual Python Environment builder
$
```

Virtual Environments

With `pip` installed, we can now go ahead and install one important package into the global Python environment. This is optional but considered a best practice to keep your global Python environment *clean*. To achieve this, we are using the Python package `virtualenv`. It is a virtual Python Environment builder.

A Virtual Environment is a tool to keep the dependencies required by different projects in separate places, by creating virtual Python environments for them.

For more information, continue to read here.

With an activated virtual environment, only a few packages inside of this Python environment are available. If we need anything in addition, it has to be installed separately. But this way we can keep a local / specialized Python environment inside of a directory without poisoning our global Python environment.

You also don't have to run `pip` with elevated privileges on *nix platforms. Python packages are simply installed with the normal user rights, no `sudo` required -- which is another benefit.

The following steps illustrate how to install `virtualenv`, how to create and activate a virtual environment and how to install additional packages into such an activated virtual environment. As we need the `requests` package going forward, we are using it to illustrate the procedure.

Code Directory

We suggest to create a directory where all the sample code, virtual environments and your own code goes.

Virtual Environments on Mac / Linux

- Install `virtualenv` using `pip` (this is only needed once if you haven't installed `virtualenv` before):

```
$ sudo -H pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-15.0.3-py2.py3-none-any.whl (3.5MB)
    100% |████████████████████████████████| 3.5MB 335kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.0.3
$
```

> **Note:** We have to run this with elevated privileges (`sudo`) because we install the package globally. Also, the use of `-H` is recommended to set the correct home directory for the root user.

- For every virtual environment needed we have to set it up in a directory of our choice (`mycode` is the name of the directory / project we want to use):

```
$ virtualenv mycode
New python executable in mycode/bin/python
Installing setuptools, pip...done.
$
```

- Activate the virtual environment

```
$ cd mycode
$ source bin/activate
(mycode)$
```

- Install the requests package into the virtual environment:

```
$ pip search requests | grep '^requests ('
requests (2.11.0)      - Python HTTP for Humans.
$ (mycode) $ pip install requests

Collecting requests
  Downloading requests-2.11.0-py2.py3-none-any.whl (514kB)
    100% |████████████████████████████████| 522kB 1.4MB/s
Installing collected packages: requests
Successfully installed requests-2.11.0
(mycode) $
```

- Install additional packages into the virtual environment:

```
(mycode) $ pip install netaddr argparse pyang
Collecting netaddr
  Using cached netaddr-0.7.18-py2.py3-none-any.whl
Collecting argparse
  Using cached argparse-1.4.0-py2.py3-none-any.whl
Collecting pyang
  Using cached pyang-1.7-py2.py3-none-any.whl
Collecting uniq
  Using cached uniq-1.2.1.28-py3-none-any.whl
Installing collected packages: netaddr, argparse, pyang, uniq
Successfully installed argparse-1.4.0 netaddr-0.7.18 pyang-1.7 uniq-
1.2.1.28
```

- `(mycode) $`
- Install the ncclient package into the virtual environment (This is required for the labs that use NETCONF):
- `(mycode) $ pip install ncclient`
- `.`
- `.`
- `.`
- `Successfully installed ncclient paramiko lxml six cryptography pyasn1 idna cffi pycparser`
- `Cleaning up...`
-
- `(mycode) $`

Virtual Environments on Windows

On Windows OS, install the virtual environment and look for the `Scripts` directory. Typically, `Scripts\activate` can be called to activate the virtual environment.

Similarly, additional tools and commands are stored in the `Scripts` directory. Either add the directory into your path or run the command using `python Scripts\sometool` from within the directory of your virtual environment.

- Install `virtualenv` using `pip` (this is only needed once if you haven't installed `virtualenv` before):
- `C:\Users\userid\Code>pip install virtualenv`
- `Collecting virtualenv`
- `  Downloading virtualenv-15.0.3-py2.py3-none-any.whl (3.5MB)`
- `    100% |################################| 3.5MB 273kB/s`
- `Installing collected packages: virtualenv`
- `Successfully installed virtualenv-15.0.3`
- `You are using pip version 8.1.1, however version 8.1.2 is available.`
- `You should consider upgrading via the 'python -m pip install --upgrade pip' command.`
-
- `C:\Users\userid\Code>virtualenv --version`
- `15.0.3`
- Create a virtual environment called `mycode`
- `C:\Users\userid\Code>virtualenv mycode`
- `Using base prefix 'c:\\python35'`
- `New python executable in C:\Users\userid\mycode\Scripts\python.exe`
- `Installing setuptools, pip, wheel...done.`
-
- `C:\Users\userid\Code>`
- Activate the virtual environment
- `C:\Users\userid\Code>cd mycode`
-
- `C:\Users\userid\Code\mycode>Scripts\activate.bat`
-

```
(mycode) C:\Users\userid\Code\mycode>pip list

pip (8.1.2)
setuptools (25.1.6)
wheel (0.29.0)

(mycode) C:\Users\userid\Code\mycode>
```

> **Note**: Activating depends on the shell in use. If you use `cmd.exe` then `activate.bat` should be used. When using Cygwin and / or Bash then `activate` (which is a bash script) should be used.
>
> You'll notice that it worked when the prompt changed.

- Install additional packages into the virtual environment

```
(mycode) C:\Users\userid\Code\mycode>pip install requests netaddr
argparse pyang uniq
Collecting requests
  Using cached requests-2.11.1-py2.py3-none-any.whl
Collecting netaddr
  Using cached netaddr-0.7.18-py2.py3-none-any.whl
Collecting argparse
  Using cached argparse-1.4.0-py2.py3-none-any.whl
Collecting pyang
  Using cached pyang-1.7-py2.py3-none-any.whl
Collecting uniq
  Using cached uniq-1.2.1.28-py3-none-any.whl
Installing collected packages: requests, netaddr, argparse, pyang, uniq
Successfully installed argparse-1.4.0 netaddr-0.7.18 pyang-1.7
requests-2.11.1 uniq-1.2.1.28

(mycode) C:\Users\userid\Code\mycode>
```

- Install the ncclient package into the virtual environment (This is required for the labs that use NETCONF):

```
(mycode) C:\Users\userid\Code\mycode>pip install ncclient
.
.
.
(mycode) C:\Users\userid\Code\mycode>
```

> **Note:** Ideally, the above is all that needed to install ncclient. However, you'll likely see error messages about missing compilers or UniCodeDecodeErrors.
>
> Please see the next section about installing ncclient on Windows. It has some specific instructions how to install it as there are a couple of installation issues around ncclient on Windows.

Installing ncclient on Windows with Python 3.5

As noted above, it is not straight forward to install ncclient on Windows with Python 3.5. As of the time of this writing (October 2016) there are two major obstacles that prevent this:

1. **lxml dependency**. ncclient depends on lxml. To 'pip install' lxml, the machine needs a working C development environment. However, this results in a lot of installation tasks of Visual C, .Net runtime and related libraries. All those installs amounts to more than 4GB worth of diskspace.

   The easier approach in this case is to install a binary wheel as described here from http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml.

2. **UniCode encoding error**. When installing ncclient using pip install it might throw an error like this *UnicodeDecodeError: 'charmap' codec can't decode byte 0x90 in position 4336: character maps to <undefined>.*

An installation from source helps.

To avoid the above issues follow the steps below to install ncclient:

1. Download lxml for your Python environment. Python version (3.5) and architecture (64-bit) must match. Check your Python installation to verify. Download the matching version from http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml. It will be named `lxml-3.6.4-cp35-cp35m-win_amd64.whl` (or very similar). The important thing is that the filename must include the platform ('win_amd64') and the Python version ('cp35'). Otherwise pip will refuse to install the wheel.
2. In your activated Virtual Environment, install the wheel:
3. `(mycode) C:\Users\userid\Code\mycode>pip install ..\Downloads\lxml-3.6.4-cp35-cp35m-win_amd64.whl`
4. `Processing c:\users\userid\downloads\lxml-3.6.4-cp35-cp35m-win_amd64.whl`
5. `Installing collected packages: lxml`
6. `Successfully installed lxml-3.6.4`
7.
8. `(mycode) C:\Users\userid\Code\mycode>`
9. Clone the source of ncclient from GitHub. This is currently only needed if Python 3.5 is used. Python 2.7 is much more at ease when it comes to character encoding. When using Python 2.7 then a simple 'pip install ncclient' should be all that's needed at this point.
10. `(mycode) C:\Users\userid\Code\mycode>git clone https://github.com/ncclient/ncclient.git`
11. `Cloning into 'ncclient'...`
12. `remote: Counting objects: 3734, done.`
13. `remote: Compressing objects: 100% (2/2), done.`
14. `Receivinremote: Total 3734 (delta 0), reused 0 (delta 0), pack-reused 3732`
15. `Receiving objects: 100% (3734/3734), 2.68 MiB | 0 bytes/s, done.`
16. `Resolving deltas: 100% (2086/2086), done.`
17. `Checking connectivity... done.`
18.
19. `(mycode) C:\Users\userid\Code\mycode>`
20. Change the working directory to the ncclient directory (use the cd command)

21. `(mycode) C:\Users\userid\Code\mycode>cd ncclient`
22. Install requirements and prerequisites for ncclient
23. `(mycode) C:\Users\userid\mycode\ncclient>pip install -r requirements.txt`
24. `.`
25. `.`
26. `.`
27. `(mycode) C:\Users\userid\mycode\ncclient>`
28. Install ncclient
29. `(mycode) C:\Users\userid\mycode\ncclient>py setup.py install`

If you successfully completed the above steps, the ncclient package and its dependencies are installed.

Verify The Package Installations (all platforms)

Your Python installation in the `mycode` directory should now have the following packages installed. Keep in mind that on a Windows OS the output and prompts might look slightly different, but packages should be the same.

```
(mycode) $ pip list
cffi (1.8.3)
cryptography (1.5.2)
enum34 (1.1.6)
idna (2.1)
ipaddress (1.0.17)
lxml (3.6.4)
ncclient (0.5.2)
netaddr (0.7.18)
paramiko (2.0.2)
pip (8.1.2)
pyang (1.7)
pyasn1 (0.1.9)
pycparser (2.14)
requests (2.11.1)
setuptools (28.3.0)
six (1.10.0)
uniq (1.2.1.28)
wheel (0.30.0a0)
(mycode) $
```

To verify that your Python environment is functional, follow these steps by running the following commands in a terminal window:

1. Open a command line window (terminal window)
2. Change the working directory to where your virtualenv resides. For example if you installed it in a directory called mycode, type:

   ```
   cd mycode
   ```

3. Activate the virtualenv by typing:

```
source bin/activate
```

4. Open an interactive Python session by typing:

   ```
   python
   ```

5. Within the Python session, import the installed Python packages by typing:
6. `import ncclient, netaddr, pyang, requests, ipaddress, uniq`

The following example shows the expected output of the above steps:

```
$ cd mycode
$ source bin/activate
(mycode)$ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import ncclient, netaddr, pyang, requests, ipaddress
>>> exit()
(mycode) $
```

Congratulations! You now have a functional Python environment including the `pip` package manager and a few additional packages which we installed from the PyPI package repositories off the Internet. You are now also able to use the `virtualenv` command to create and activate customized virtual Python environments.

Note: The version output above shows a specific Python version (here: the native Python 2.7 that comes with Mac OS X 10.11). Your Python version banner might look different, especially if you installed Python 3.5. The important thing is the `import` statement. If that works for you without any error then all is well!

Your local machine is ready to continue your programming adventures with Python.

Good Job!

Note: Step 4 and 5 are omitted because they require dCloud.

Step 6. Sign Up For Accounts

We recommend that you obtain the following accounts.

Get a Cisco Spark Account

(Required) Throughout the learning track you will use Cisco's instant messaging tool called Cisco Spark. To use Cisco Spark you need a Spark account. Register with Cisco Spark if you do not already have a Spark account. It's a free service. You only need an email address to register.

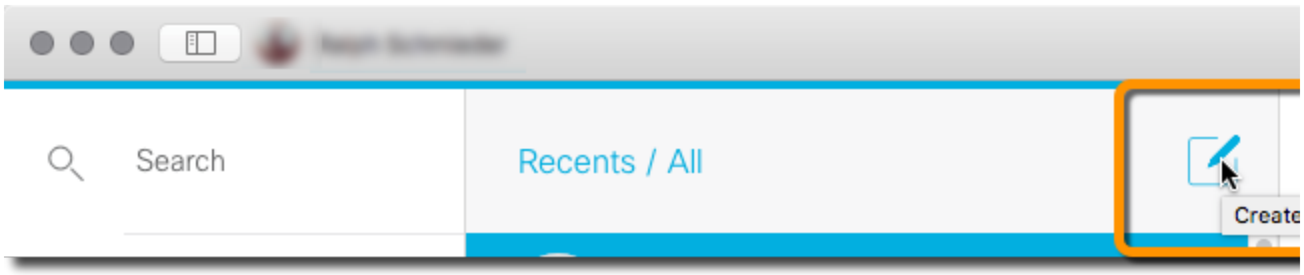
1. Go to https://www.ciscospark.com
2. Fill out the registration info

Create a Spark Room

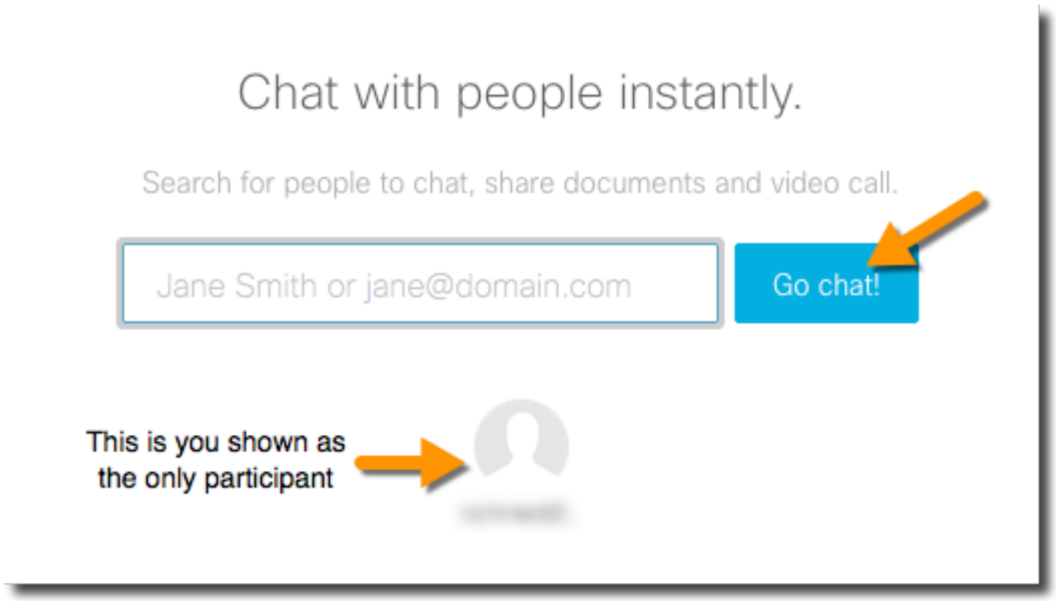
Create a test room in Spark where you can send yourself messages (unless told otherwise, in a hands-on event you will be provided with a specific event room name where everybody posts messages).

To create a Spark room:

1. Open your Spark client or the Spark Web client
2. Create a new room



3. Add your email address, followed by 'Return'


Chat with people instantly.
Search for people to chat, share documents and video call.
Jane Smith or jane@domain.com
Go chat!
This is you shown as the only participant

4. Click 'Go chat!' to create the room
5. The room will only be started when you send a message to it.
6. Type a message. The room shows as 'Untitled'
7. Click the title and change the room name to something memorable such as 'LetMySPARKRoomFly'

Congratulations, you have a Spark room! You can use this room later when you perform tests against the Spark API.

Get a Cisco CCO Account

(Required) We recommended that you register for a CCO account on the Cisco.com web site. A CCO account is required to access certain lab environments and resources.

1. Go to https://tools.cisco.com/RPF/register/register.do
2. Fill out the CCO registration info

Register on DevNet

(Optional) We strongly encourage you to register with DevNet. There's lots of cool benefits like access to free Learning Labs and the DevNet Sandbox.

Get a GitHub Account

(Optional) While this is not scrictly required, we recommend that you join GitHub.

1. Go to https://github.com/join
2. Fill out registration info

You are not required to have a Github account. The repository that we use in this track is a public repository, which does not require that you have a Github account.

However, a Github account can be useful to you because you will be able to create your own code repositories and save your personal code there. For example, you'll be able to store the source code that you create during the labs to your GitHub account for your use later.

Continue

Continue to the next and final step in the preparation module, *Troubleshooting Your Pre-event Setup*.

Step 7. Troubleshooting Your Pre-event Setup

Did you have a problem completing any of the previous steps?

To be able to proceed to the lab activities in this track, it's important to have (at a minimum) the following requirements completed:

- Python environment properly installed (if you are using your local machine to run the dev tools)
- Client for remote connection with either:
  - VPN installed (AnyConnect or OpenConnect), or
  - an HTML5-enabled browser
- CCO account
- Connection to the lab environment tested

Please be aware that there is not much time for troubleshooting during a hands-on event. This is why you need to make sure that Python and AnyConnect are working properly **before** the event.

If you plan to attend a hands-on event and do not have the above items working, we **strongly** suggest that you use the web-based remote desktop session during the hands-on event. The web-based remote desktop will give you access to a complete lab environment without having to install Python on your local machine.

Getting Support

If your Python environment does **NOT** work, you can look for answers to general questions on http://stackoverflow.com.

You can ask questions about the setup in the DevNet Support Spark room. The DevNet team will answer your questions directly in that room.

This concludes the pre-event setup. You're ready to start the next module on DevNet resources!