## Lab 04: Explore Python Development Tools

### Case Study

A leading telecommunications company in Ireland provides broadband and fiber services to millions of customers across major cities. With a workforce of over 1,000 professionals, the organization is committed to modernizing its network and internal systems through automation and advanced development practices. As digital expectations rise, the company recognized the need to streamline its development processes and strengthen its network automation capabilities.

### Business Challenge

The company faced several business challenges related to inconsistent Python environments, manual network configuration tasks, and time-consuming troubleshooting. Development teams often struggled with conflicting Python package versions, which delayed testing and slowed down automation deployments. Network engineers, on the other hand, relied heavily on manual steps to back up router configurations and apply changes to network devices, resulting in operational inefficiencies and an increased risk of error.

To address this challenge, the company hired you—an individual certified as a Cisco DevNet Associate—to automate the backup process and modernize the network configuration workflow.

### Solution

The organization implemented a standardized solution using Python virtual environments to isolate dependencies for each project. This ensured consistent and conflict-free setups across development machines, making project environments easier to reproduce, manage, and share.

1. Review the Python Installation
2. PIP and Python Virtual Environments
3. Sharing Your Virtual Environment

|  |
| --- |
| **// Review the Python Installation**  1. Double-click on a **Terminal** icon to open it.    2. Verify the Python installation on your DEVASC virtual machine. To begin, check the installed Python version by running the command **python3 -V**. This confirms which version is currently available in this case, Python 3.8.2.    3. Locate where Python is installed on the system, using the **which python3** command. This returns the path to the Python binary, typically **/usr/bin/python3,** indicating the system-level Python interpreter. |

|  |
| --- |
| **// PIP and Python Virtual Environments**  1. PIP, short for "Pip Installs Packages," is used to install and manage Python packages. Using pip3 install directly on the system-wide Python setup can cause conflicts between different projects due to incompatible dependencies. To avoid this, Python developers commonly use virtual environments, which isolate dependencies on a per-project basis.  To set up a virtual environment in the DEVASC VM, first navigate to your project directory using **cd labs/devnet-src/python**/.    2. Create a virtual environment named **devfun** with the command **python3 -m venv devfun.** Once created, activate it using source devfun/bin/activate, and notice the shell prompt reflects the environment name.    3. Once created, activate it using **source devfun/bin/activate**, and notice the shell prompt reflects the environment name.    4. Run the **pip3 freeze** command to verify that there are no additional Python packages currently installed in the devfun environment.    5. Using **pip3 install requests**, and afterward, running pip3 freeze will list the installed packages, such as certifi, idna, urllib3, etc.    6. Re-enter the **pip3 freeze** command to see the packages now installed in the devfun environment.    7. To deactivate the virtual environment and go back to your system, enter the **deactivate** command.    8. Enter the system-wide **python3 -m pip freeze** command to see what packages are installed in the system environment.    9. To find the version of the **requests** package currently installed, run the command **python3 -m pip freeze | grep requests.** This command lists all installed Python packages and filters the output to show only the version of the requests package.    **// Sharing Virtual Environment**  1.  Reactivate the **devfun** virtual environment using the command **source devfun/bin/activate**.    2. Send the output of the **pip3 freeze** command to a text file called **requirements.txt**.    3. **Deactivate** the devfun virtual environment. Use the **ls** command to see that the requirements.txt file is in the /python directory.    4. Create and activate a new Python virtual environment called **devnew**. By using the command **python3 -m venv devnew** and then, **source devnew/bin/activate.**    5. Use the **pip3 install -r requirements.txt** command to install the same packages that are installed in the devfun virtual environment.    6. When entering **pip3 freeze** in the devnew environment, you should see the following output.    7. **Deactivate** the devnew virtual environment |