LAB1-PYTHON

OPERATING SYSTEMS-DR. LÉONARD JANER

2020-2021

February 2021

TecnoCampus Escola Superior Politècnica

Centre adscrit a la

Universitat Pompeu Fabra Barcelona

LAB1-PYTHON

Objectives

Part 1: PIP and Python Virtual Environment

Part 2: Processing Ex1 file

Part 3: Processing Ex2 file

Part 4: Processing Ex3 file

Background / Scenario

In this lab, you review Python installation, PIP, and Python virtual environments. Then you will write a Python script to extract information from some configuration and topology files into a PDF file.

Required Resources

- 1 PC with operating system of your choice
- Virtual Box or VMWare
- DEVASC Virtual Machine

Instructions

Part 1: **PIP and Python Virtual Environments**

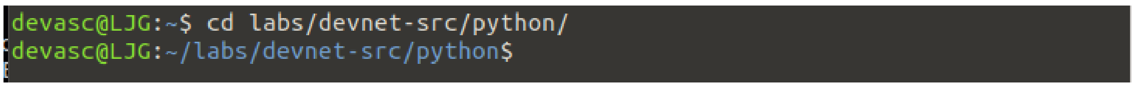
The purpose of this part is just to show you how to work with environments. When delivering the task you must provide the environment, so packages can be imported to the test environment.

PIP stands for **Pip Installs Packages**. Many people first learn about PIP and start using **pip3 install** commands on the system wide Python installation. When you run **pip3 install** command on your system, you might introduce competing dependencies in your system installation that you may or may not want for all Python projects. Therefore, the best practice is to enable a Python virtual environment. Then install only the packages that are needed for the project in that virtual environment. That way, you know exactly which packages are installed in a given setting. You can switch those package dependencies easily when switching to a new virtual environment, and not break or cause problems due to competing versions of software.

To install a Python virtual environment, use the **venv** tool in Python 3 and then activate the virtual environment, as shown in the following steps.

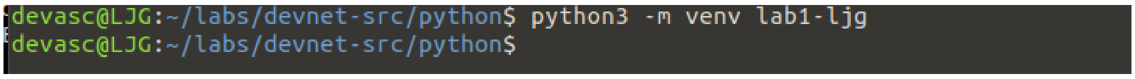
Step 1: Create a Python 3 virtual environment.

Inside the DEVASC VM, change to the **labs/devnet-src/python** directory. This is just an example, to show you how to create your own environment.



```
devasc@LJG:~$ cd labs/devnet-src/python/
devasc@LJG:~/labs/devnet-src/python$
```

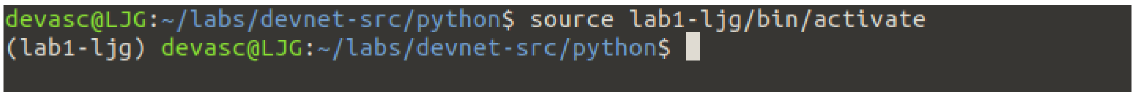
Enter the following command to use the **venv** tool to create a Python 3 virtual environment with the name **lab1-ljg**. The **-m** switch tells Python to run the **venv** module. The name is chosen by the programmer.



```
devasc@LJG:~/labs/devnet-src/python$ python3 -m venv lab1-ljg
devasc@LJG:~/labs/devnet-src/python$
```

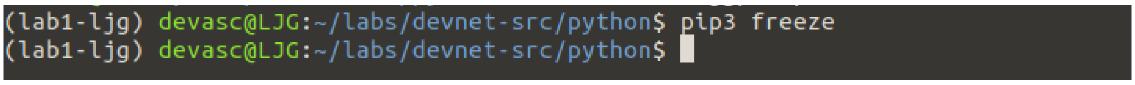
Step 2: Activate and test the Python 3 virtual environment.

Activate the virtual environment. The prompt changes to indicate the name of the environment you are currently working in, which is **lab1-ljg** in this example. Now when you use the **pip3 install** command form here, the system will only install packages for the active virtual environment.



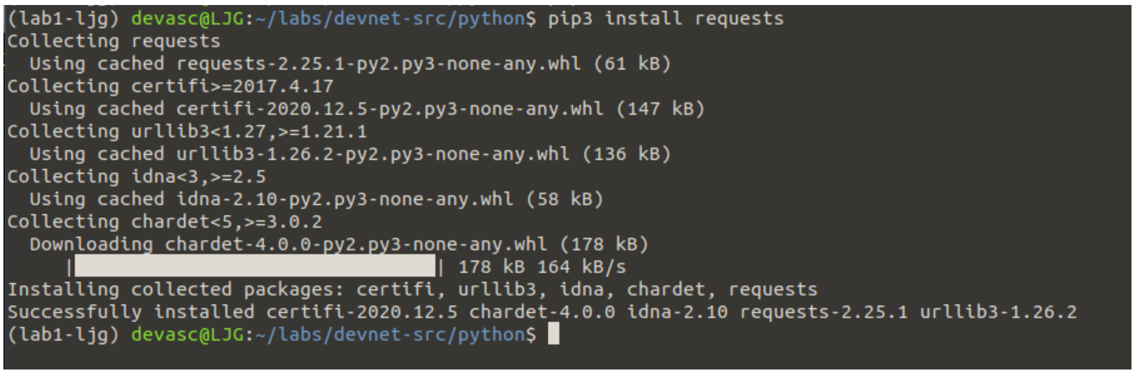
```
devasc@LJG:~/labs/devnet-src/python$ source lab1-ljg/bin/activate
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Run the **pip3 freeze** command to verify that there are no additional Python packages currently installed in the **lab1-ljg** environment.



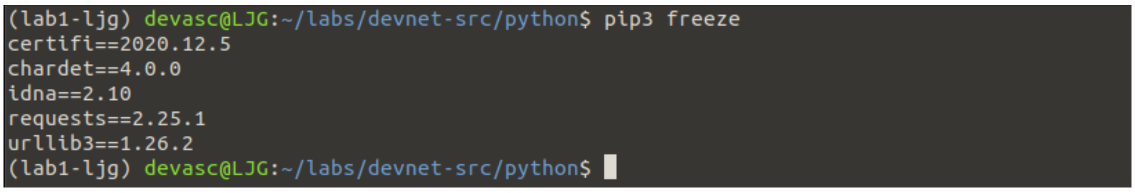
```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Now, you can install the Python **requests** package (this is just an example installing that package; you must install the packages required for your solution) within the **lab1-ljg** environment.



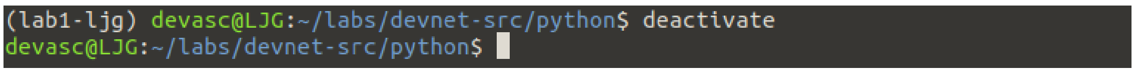
```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 install requests
Collecting requests
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.2-py2.py3-none-any.whl (136 kB)
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting chardet<5,>=3.0.2
  Downloading chardet-4.0.0-py2.py3-none-any.whl (178 kB)
     |████████████████████████████████| 178 kB 164 kB/s
Installing collected packages: certifi, urllib3, idna, chardet, requests
Successfully installed certifi-2020.12.5 chardet-4.0.0 idna-2.10 requests-2.25.1 urllib3-1.26.2
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Re-enter the **pip3 freeze** command to see the packages now installed in the **lab1-ljg** environment.



```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
certifi==2020.12.5
chardet==4.0.0
idna==2.10
requests==2.25.1
urllib3==1.26.2
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

To deactivate the virtual environment and go back to your system, enter the **deactivate** command.

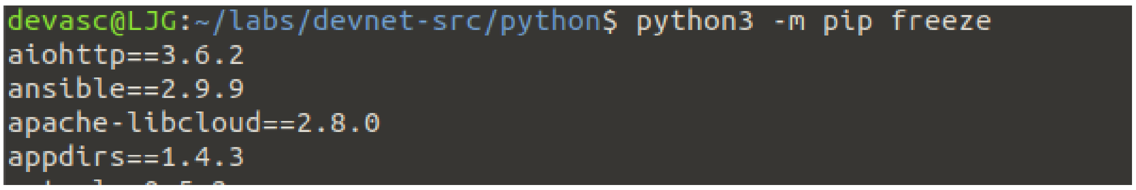


```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ deactivate
devasc@LJG:~/labs/devnet-src/python$ 
```

Step 3: Check the current packages installed in the system environment.

Enter the system wide **python3 -m pip freeze** command to see what packages are installed in the system environment.

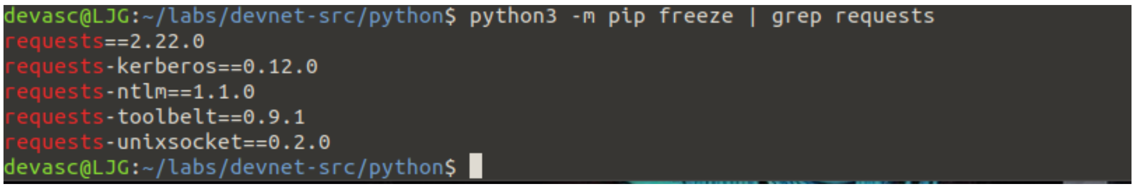
Note: Because Python 3 is invoked with the following command, you only use **pip** instead of **pip3**.



```
devasc@LJG:~/labs/devnet-src/python$ python3 -m pip freeze
aiohttp==3.6.2
ansible==2.9.9
apache-libcloud==2.8.0
appdirs==1.4.3
```

```
     <output omitted>
```

If you want to quickly find the version of a package installed, pipe the output to the **grep** command. Enter the following to see the version of the requests package currently installed.



```
devasc@LJG:~/labs/devnet-src/python$ python3 -m pip freeze | grep requests
requests==2.22.0
requests-kerberos==0.12.0
requests-ntlm==1.1.0
requests-toolbelt==0.9.1
requests-unixsocket==0.2.0
devasc@LJG:~/labs/devnet-src/python$ 
```

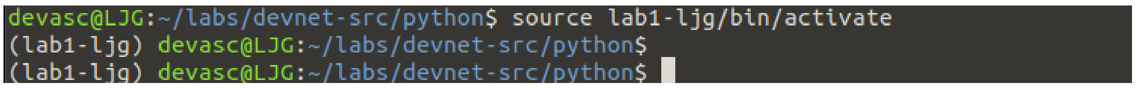
Step 4: Sharing Your Virtual Environment

The output of the **pip3 freeze** command is in a specific format for a reason. You can use all the dependencies listed so that other people who want to work on the same project as you can get the same environment as yours.

A developer can create a requirements file, such as **requirements.txt**, by using the **pip3 freeze > requirements.txt** command. Then another developer can, from another activated virtual environment, use this **pip3 install -r requirements.txt** command to install the packages required by the project.

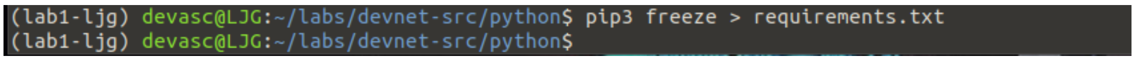
You must always deliver with your Python labs the **requirements.txt** file.

Re-activate the **lab1-ljg** virtual environment.



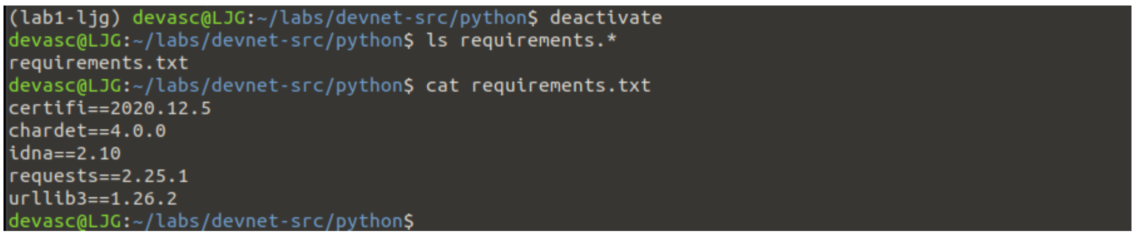
```
devasc@LJG:~/labs/devnet-src/python$ source lab1-ljg/bin/activate
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ 
```

Send the output of the **pip3 freeze** command to a text file called **requirements.txt**.



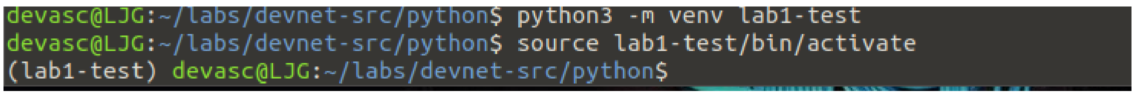
```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze > requirements.txt
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Just to simulate the process, you can deactivate the **lab1-ljg** virtual environment. You can use the **ls** command to see that the **requirements.txt** file is in the **/python** directory.



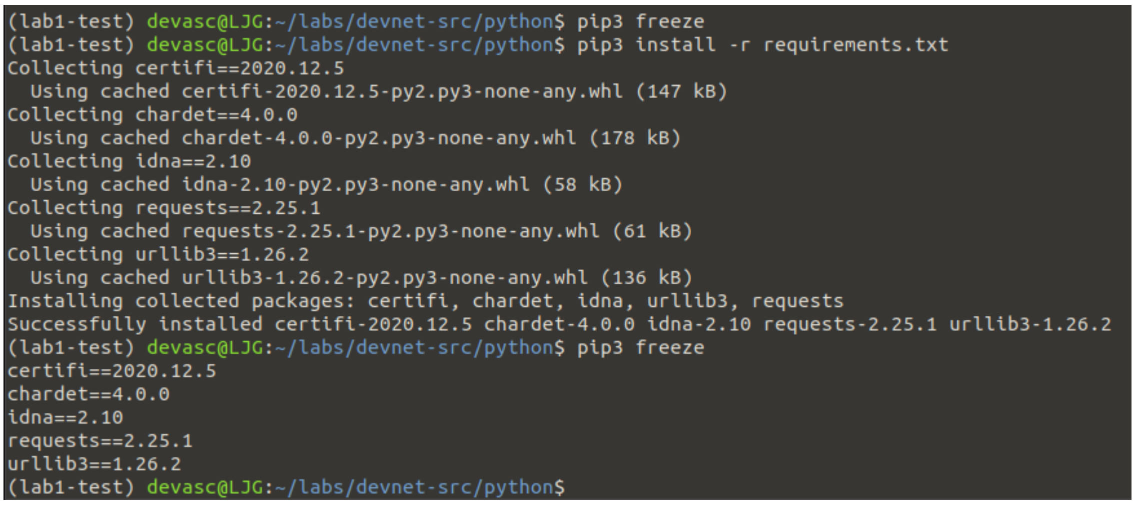
```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ deactivate
devasc@LJG:~/labs/devnet-src/python$ ls requirements.*
requirements.txt
devasc@LJG:~/labs/devnet-src/python$ cat requirements.txt
certifi==2020.12.5
chardet==4.0.0
idna==2.10
requests==2.25.1
urllib3==1.26.2
devasc@LJG:~/labs/devnet-src/python$
```

Now, you can create and activate a new Python virtual environment called **lab1-test**.



```
devasc@LJG:~/labs/devnet-src/python$ python3 -m venv lab1-test
devasc@LJG:~/labs/devnet-src/python$ source lab1-test/bin/activate
(lab1-test) devasc@LJG:~/labs/devnet-src/python$
```

Use the **pip3 install -r requirements.txt** command to install the same packages that are installed in the **lab1-ljg** virtual environment.



```
(lab1-test) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
(lab1-test) devasc@LJG:~/labs/devnet-src/python$ pip3 install -r requirements.txt
Collecting certifi==2020.12.5
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting chardet==4.0.0
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting idna==2.10
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting requests==2.25.1
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting urllib3==1.26.2
  Using cached urllib3-1.26.2-py2.py3-none-any.whl (136 kB)
Installing collected packages: certifi, chardet, idna, urllib3, requests
Successfully installed certifi-2020.12.5 chardet-4.0.0 idna-2.10 requests-2.25.1 urllib3-1.26.2
(lab1-test) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
certifi==2020.12.5
chardet==4.0.0
idna==2.10
requests==2.25.1
urllib3==1.26.2
(lab1-test) devasc@LJG:~/labs/devnet-src/python$
```

This will be the process to test and execute your python programs. Installing the required packages according to your **requirements.txt** file on a new environment.

When delivering your lab, you must deliver an **requirements.txt** file so when evaluating the import (and install) can be done, and the testing environment could be exactly the same as yours, with exactly the same libraries, version number, and so. If the lab works properly on your environment it must work the same way on my imported environment. This is also the way to share environment with your teammate. If the importing process fails then the labs will not be evaluated.

If you want to import/export your environment using **VS Code** you can also.

You have always to write the import process to guarantee that it is clearly written how to import your environment. If it is not written, no import will be done, and so, no evaluation will be undertaken.

Part 2: **Processing Ex1 file**

The purpose of this second part is to write a Python Script to process a file named Ex1 and write a PDF report with the required information. The script must request the file name and generate a PDF file named Ex1.pdf with the same template as the Example

The task report must include:

- Information about how the environment has been saved by the student.
- Information about how the environment must be imported into the instructor system.
- Information about all the packages that must be used on the lab (with a brief description of the package purpose for the lab)
- Information about the structure of the program, and description of the most relevant parts of the code (topology representation and interfaces table must be explained with deep details)
- Screenshots with the execution of the program.

Step 1: Title Page

The title page must include a LOGO, a vertical YELLOW LINE, the bottom BANNER, as static data.

The title page must include a title ("Informació infraestructura de Xarxa TecnoCampus (Ex1)") where everything must be static except for the name of the input file (Ex1), that must be dynamic.

The date must be the MONTH when the file is written by the script and must be dynamic.

Step 2: Header and footer

The document must include for all the pages, except the first one, a header with the name of the title and the LOGO, and a footer with the page number (starting with page number 2).

Step 3: Index

The document must include a dynamic index, as in the example.

Step 4: Introduction

The document must include an introduction section, with the information you have in the example written dynamically.

Step 5: Introduction (topology)

The document must include a dynamic topology of the infrastructure, with the names of the devices (and the type of device), and if possible, the name of the interfaces. It is not necessary to have the kind of topology of the example, but we have to see which devices are interconnected together. The template has been provided by the customer, and is hand-made.

Step 6: Devices configuration

The document must include a configuration section per each device on the topology. For ROUTER/IOSV you must write the last configuration changes date, in the format of the example.

Step 7: Interfaces

The document must include per devices, information about the interfaces of the device.

If the device is a SWITCH that's the information you will have about the device.

If the device is a ROUTER/IOSV you will have further information

If the device is a LAPTOP/PC/SERVER that's the information you will have about the device.

Step 8: Cryptographic information for routers

The document must include information about crypto configuration on the router if available, as in the examples. If the information for the device is not available a message must be written as in the examples.

Step 9: Routing information

The document must include a dynamic information about routing protocol configuration of the ROUTER as in the examples. (for Ex2 and Ex3 if it is not clear how the ROUTING information works and what to write ask the instructor about it).

Step 10: Access Control List configuration

The document must include a dynamic information about access control list of the device, as in the examples (for Ex2 and Ex3 if it is not clear how the ACL works and what to write ask the instructor about it). If the information for the device is not available a message must be written as in the examples.

Step 11: Banners configuration

The document must include a dynamic information about banners of the device, as in the examples.

Step 12: Interfaces configuration

The document must include a dynamic information about all the links of the topology, as in the example, with a list of links.

Step 13: Interfaces configuration

The document must include a dynamic information about all the links of the topology, as in the example, with a table.

Part 3: **Processing Ex2 file**

The purpose of this third part is to process a file named Ex2 and write a PDF report with the required information. The script must request the file name and generate a PDF file named Ex2.pdf.

The task report must include:

- Screenshots with the execution of the program.

Processing Ex3 file

The purpose of this third part is to process a file named Ex3 and write a PDF report with the required information. The script must request the file name and generate a PDF file named Ex3.pdf.

The task report must include:

- Screenshots with the execution of the program.