Python Virtual Environments

What is a Python virtual environment and why you should use it?

One big change when passing from Python 2 to Python 3 is the use of virtual environments (venv). The idea is simple: in a virtual environment, you may install a set of libraries exclusive to this environment. Venvs avoid compatibility issues (i.e. less problems with python), and help a lot all support operation (faster resolution of problems).

Divide to Reign

When you you work in a venv, you are limiting the number of packages you are working with. Listing (`pip list`) and Checking (`pip check`) is just far easier and faster with a smaller number of packages.

For support, you can create an ID card of your python environment, which is simply the list of packages installed. The smaller the list is, the faster your dear support will be able to be in the exact state as you are on a different machine.

How to install a virtual environment?

Prerequisites

First you have to install Python 3, you can find how to do it on https://www.python.org/downloads/. Make sure you are using the right version of Python with the command 'which python'

python -m venv

The command to install a virtual environment is

```
python -m venv -awesome_new_environement-
```

Worked example

Here is an example with the venv **totoro** created in a folder **python_venvs** at the root level:

```
mylogin@mymachine:~>python -m venv ~/python_venvs/totoro
mylogin@mymachine:~>ls totoro
bin/    include/    lib/    pyvenv.cfg
```

This command will create a folder `totoro` with inside three folders and a file: `bin/`, `include/`, `lib/` and `pyvenv.cfg`. Now that you have created your environment, you have to activate it before installing the libraries you need on it.

To activate your virtual environment, use the `activate` shell script inside the venv:

```
mylogin@mymachine:~>source ~/totoro/bin/activate
(totoro) mylogin@mymachine:~>
```

You should now see at the beginning of the command line the name of your virtual environment.

Now you can install every library you need:

```
(totoro) mylogin@mymachine:~>pip install numpy
(totoro) mylogin@mymachine:~>pip install scipy
```

You can quit your virtual environment and go back to the default one with **deactivate**:

```
(totoro) mylogin@mymachine:~>deactivate
mylogin@mymachine:~>
```

The packages available in **totoro** are no more visible. Try `pip list` to check.

Common troubleshoots

Virtual environments can experience troubles.
Here are some good practices.

As soon as you create your virtual environment, **update its pip and setuptools**:

```
(totoro) mylogin@mymachine:~>pip install --upgrade pip
(totoro) mylogin@mymachine:~>pip install --upgrade setuptools
```

These two libraries changes fast, keep them as updated as possible to limit troubles.

Never use the `--user` option of pip. This will put some of your dependencies in a user-defined place. In case of support, this trick will make you loose hours.

Take **care of third party actions impacting your `$PYTHONPATH`**. This variable must be handle by your venv only. Common trouble makers are:

1. `module load ****` The modules often changes envirionment variables. A module `module load python/3.8.2` done while in a venv is a poison bullet.
2. Shell scripts, especially profile shell scripts, often edit your environment variables. If one action deals with python tools, it probably update `$PYTHONPATH`.

Installing package properly

Once you have your virtual environment setup, you might end up in the need to install packages to which you have access through git for example and want to install it as well as its various dependencies. The way to do it properly is to avoid using `python setup.py install` and `python setup.py develop`, those two commands might lead you to dependencies not installed in your virtual environment.
A better ans safer way to do it and avoid doing the `pip install -dependency1- -dependency2-` and so on is to directly used the `pip install` command in the repository folder.

To do so, make sure you are in the same folder as the `setup.py` of the package you're currently trying to install and use one of these two commands :

- `pip install .` if you want to install the python package in install mode, this will first look for the dependencies and collect the files from pypi and then install everything to have a safe running package, it's only after collecting all the files that the command `python setup.py install` is executed;
- `pip install -e .` if you want to install the python package in develop mode.

A collection of virtual environment tutorials and motivators

Virtual environment is not a COOP invention, but "the recommended way to master your python framework". It can be related to your first contact with the UNIX environment variables (`.bashrc` , `module load` , etc...). See the following link for non-COOP venvs explanations:

- python.org: the absolute reference about venvs.
- realpython : a more accessible tutorial.
- geeksforgeeks an even shorter primer on venvs.
- [towardsdatascience](https://towardsdatascience.com/why-you-should-use-a-v irtual-environment-for-every-python-project-c17dab3b0fd0) a **non-technical** blog on WHY you should use venvs