Creating Python Virtual Environment

Unix Based System (Mac OS)

Dr. Saad Laouadi

Econometrics and Data Science Academy

Outline

Introduction

Virtual Environment Definition

Virtual environments can be described as isolated installation directories. This isolation allows you to localized the installation of your project's dependencies, without forcing you to install them system-wide. [1]

Why Would You Ever Need a Virtual Environment? Do you want to have a computer for each project?
Of course not, with virtual environment you can have a specific environment for each project. Here are some benefits of virtual environments:

1. Having different projects or applications, and each project with its needs of packages of specific versions.
2. Having multiple environments, with multiple sets of packages, without conflicts among them.
3. Working with different projects at the same time without the fear of breaking projects when upgrading the base system packages.
4. Releasing projects with their dependent dependencies.

[1] source

Creating Virtual Environment I

- To create a **virtual environment** you can use **venv** module, which comes by default with python standard library.

The standard steps to create a **virtual environment on windows** are as follows:

1. Create a new working directory for your project. I will call this NumAnalysis for numerical analysis project:

```
root$ mkdir NumAnalysis
```

Creating Virtual Environment II

2. Change the working directory:

```
root$ cd NumAnalysis
```

Print the working directory

```
root$ pwd
```

Creating Virtual Environment III

3. Create a new virtual environment in this current directory, and give it a name like **myvenv or venv** or anything you like. I will call mine venvnum.

```
root$ python -m venv venvnum
```

4. Check your new virtual environment is created (not necessary, just to make sure).

```
root$ ls
```

Creating Virtual Environment IV

5. Check where python is globally installed, in other words, the main python on your system(not necessary as well).

```
root$ which python
```

you will have something like this:

```
/usr/local/bin/python3
```

Creating Virtual Environment V

6. Activate the virtual environment. You can use a path as well where the **venv** is created.

```
root$ source NumAnlysis/venvnum/bin/activate
```

You will see the name of your **virtual environment** before the prompt on your command line interpreter.

```
(venvnum) root NumAnalysis $
```

Check where python is now:

Creating Virtual Environment VI

```
root$ which python
```

You will see a path like this:

```
/Users/UserName/dirname/venvnum/bin/python
```

Creating Virtual Environment VII

7. List the installed packages in this environment.

```
root$          py -m pip list
```

```
Package    Version
---------- -------
pip        21.2.3
setuptools 57.4.0
```

Creating Virtual Environment VIII

8. Install new python packages. The packages will be available only on this environment and won't affect the main python environment. I will install numpy and scipy.

```
root$ python -m pip install numpy scipy
```

You can check again the installed packages in this environment.

```
root$ python -m pip list
```

Creating Virtual Environment IX

```
Package    Version
---------- -------
numpy      1.22.3
pip        21.2.3
scipy      1.8.0
setuptools 57.4.0
```

Creating Virtual Environment X

9. Deactivate the environment using **deactivate command**. Note, if you close off the command line interpreter, the virtual environment will be deactivated automatically.

```
root$ deactivate
```

You will see the base in active now

```
(base) root/NumAnalysis $
```

Reactivating The Virtual Environment

- You can activate the previously create **venv** from anywhere in the command line. It's better to change your directory where your project is at then acitvate. But not necessary.

```
root$ source venvnum/bin/acitvate
```

- If you are at another project (not the one it has **venv**) and you want to use this **venv**, you can activate it using the relative or absolute path:

```
root$ source dirname/venvnum/bin/activate
```

The dirname is your directory name of your.

Creating Source files

- When you create a **venv** in the project you are working on, you should create the source files in the project folder, **not inside** the **venv**.
- If you are working with a control version, you should ignore the **venv**.

You can create a python file in the **NumAnalysis** folder like this:

```
root$ touch test.py
root$ ls
```

```
test.py
venvnum
```

Creating a File from Virtual Environment

- If you intend to use the same environment elsewhere, with the same packages with their versions, or use it on git or share it with a colleague so they can set up an environment similar to yours, you can export that to a **requirements file** using **freeze** command in the active **venv**.

```
root$ python -m pip freeze > requirements.txt
```

Creating a Virtual Environment from a File

- Usually, you have a point to start from, that is you already a requirements file. If that is the case, then you can install the packages in active environment using this file.

```
root$ source dirname/venvnum/bin/activate
root$ python -m pip install -r requirements.txt
```

Creating Virtual Environment Accessing The Base System Packages

- There is a chance that you want the virtual environment to have access to all the packages installed in the base system
- If you install new packages in the **venv**, this won't affect the base system.

```
root$ python -m venv envTobase --system-site-packages
root$ source venvTobase/bin/activate
root$ python -m pip list
```

you will see all the packages installed in the base system are listed.

All Steps Example

```
root$ mkdir NeuralNets
root$ cd NeuralNets
root$ python -m venv venv
root$ pip list
root$ source venv/bin/activate
root$ python -m pip list
root$ python -m pip install -r requirements.txt
root$ pip list
root$ pip freeze > requirements.txt
root$ deactivate
```

Virtual Environment Help

```
root$ python -m  venv -h
```

The head help page looks like this:

```
usage: venv [-h] [--system-site-packages] [--symlinks | --copies] [--clear]
    [--upgrade]
[--without-pip] [--prompt PROMPT] [--upgrade-deps]
ENV_DIR [ENV_DIR ...]
Creates virtual Python environments in one or more target directories.
positional arguments:
ENV_DIR               A directory to create the environment in.
optional arguments:
-h, --help            show this help message and exit
--system-site-packages
```

Deleting Environment

- In case you want to delete an environment, you can delete the directory containing that environment using the next command.

```
root$ rm -rf dirname
```

Copyright Information