Creating Python Virtual Environment

Windows System

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:

```
C:\Users\Name> mkdir NumAnalysis
```

Creating Virtual Environment II

2. Change the working directory:

```
C:\Users\Name> cd NumAnalysis
```

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.

```
C:\Users\Name> py -m venv venvnum
```

Creating Virtual Environment III

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

```
C:\Users\Name> dir
```

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

```
C:\Users\Name> where python
```

you will have something like this:

```
C:\Users\Name\AppData\Local\Programs\Python\Python310\python.exe
```

Creating Virtual Environment IV

6. Activate the virtual environment. Pay attention to the leading dot(.) means this current directory. You can use a path as well where the **venv** is created.

```
C:\Users\Name> .\venvnum\Scripts\activate
```

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

```
(venvnum) C:\Users\Name\NumAnalysis>
```

Check where python is now:

Creating Virtual Environment V

```
root$ where python
```

The active path will show first as follows:

```
C:\Users\Name\NumAnalysis\venvnum\Scipts\python.exe>
C:\Users\Name\AppData\Local\Programs\Python\Python310\python.exe
```

Creating Virtual Environment VI

7. List the installed packages in this environment.

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

```
Package          Version
----------       ----------
pip              22.0.4
setuptools       58.1.0
```

Creating Virtual Environment VII

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$ py -m pip install numpy scipy
```

You can check again the installed packages in this environment.

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

Creating Virtual Environment VIII

```
Package                   Version
----------                ----------
numpy                     1.22.3
pip                       22.0.4
scipy                     1.8.0
setuptools                58.1.0
```

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
```

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$ .\venvnum\Scripts\activate
```

- 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$ dirname1\dirname2\venvnum\Scripts\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$ echo > test.py
root$ dir /b
```

```
test.py
venvnum
```

Creating a File from Virtual Environment

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

```
root$ py -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$ .\venvnum\Scripts\activate
root$ py -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$ py -m venv envTobase --system-site-packages
root$ .\venTobase\Scripts\activate
root$ py -m pip list
```

you will see all the packages listed.

All Steps

```
root$ mkdir NeuralNets
root$ cd NeuralNets
root$ py -m venv venv
root$ pip list
root$ source .\Scripts\activate
root$ py -m pip list
root$ py -m pip install -r requirements.txt
root$ pip list
root$ pip freeze > requirements.txt
root$ deactivate
```

Virtual Environment Help

```
root$ py -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:
Give the virtual environment access to the system site-packages dir.
```

Deleting Environment

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

```
root$ rmdir  dirname /s
```

Copyright Information