Parametric Thoughts

Using Virtual Environments in Jupyter Notebook and Python

02 Feb 2019

Are you working with Jupyter Notebook and Python? Do you also want to benefit from virtual environments? In this tutorial you will see how to do just that with Anaconda or Virtualenv/venv.

Before we start, what is a virtual environment and why do you need it? A virtual environment is an isolated working copy of Python. This means that each environment can have its own dependencies or even its own Python versions. This is useful if you need different versions of Python or packages for different projects. This also keeps things tidy when testing packages and making sure your main Python installation stays healthy.

Create Virtual Environment with Virtualenv/venv

A commonly used tool for virtual environments in Python is virtualenv. Since Python 3.3, a subset of virtualenv has been integrated in the Python standard library under the venv module. If you are using Python 2, you can install virtualenv with:

```
pip install --user virtualenv
```

Now, you can create a virtual environment with:

```
virtualenv myenv
```

where `myenv` can be replaced with the name you want for your virtual environment. The virtual environment can be found in the `myenv` folder. For Python >= 3.3, you can create a virtual environment with:

```
python -m venv myenv
```

After you have created your virtual environment, you can activate the virtual environment with:

```
source myenv/bin/activate
```

To deactivate the virtual environment, you can run `deactivate`. To delete the virtual environment you just need to remove the folder with the virtual environment (e.g. `rm -r myenv`). For further information, have a read in the virtualenv documentation or venv documentation.

Create Virtual Environment with Anaconda

Let's have a look how to create an virtual environment with Anaconda. Anaconda is a Python (and R) distribution that has the goal to simplify package management and deployment for scientific computing. After the installation you can create the conda virtual environment with:

```
conda create -n myenv
```

where `myenv` is the name of your new environment. If you want a specific Python version that is not your current version, you can type:

```
conda create -n myenv python=3.6
```

The environment is then stored in the `envs` folder in your Anaconda directory. After you have created the enviroment, you can activate it by typing:

```
conda activate myenv
```

If you now run `python` , you'll see that you are in your freshly created virtual environment. To deactivate the environment you can type `conda deactivate` and you can list all the available environments on your machine with `conda env list` . To remove an enviroment you can type:

```
conda env remove -n myenv
```

After creating your environment, you can install the packages you need besides the one already installed by conda. You can find more information on how to manage conda environments in this user guide.

Add Virtual Environment to Jupyter Notebook

Jupyter Notebook makes sure that the IPython kernel is available, but you have to manually add a kernel with a different version of Python or a virtual environment. First, make sure your environment is activated with `conda activate myenv` . Next, install ipykernel which provides the IPython kernel for Jupyter:

```
pip install --user ipykernel
```

Next you can add your virtual environment to Jupyter by typing:

```
python -m ipykernel install --user --name=myenv
```

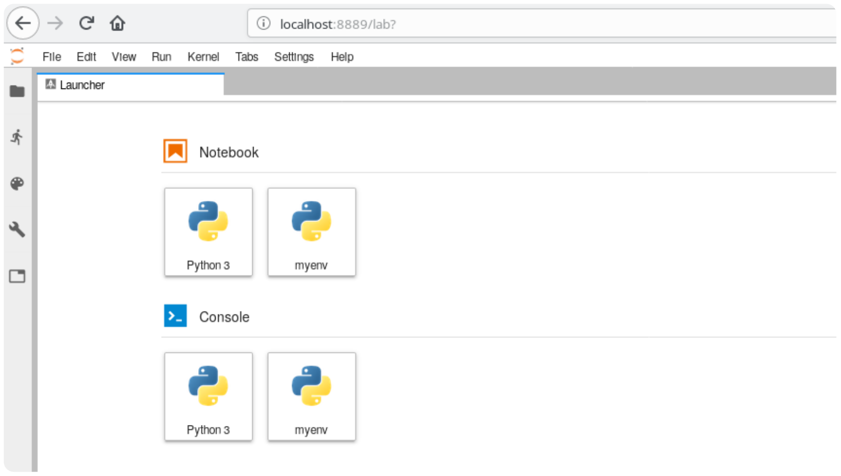
This should print the following:

```
Installed kernelspec myenv in /home/user/.local/share/jupyter/kernels/
```

In this folder you will find a `kernel.json` file which should look the following way if you did everything correctly:

```
{
 "argv": [
  "/home/user/anaconda3/envs/myenv/bin/python",
  "-m",
  "ipykernel_launcher",
  "-f",
  "{connection_file}"
 ],
 "display_name": "myenv",
 "language": "python"
}
```

That's all to it! Now you are able to choose the conda environment as a kernel in Jupyter. Here is what that would look like in JupyterLab:



Remove Virtual Environment from Jupyter Notebook

After you deleted your virtual environment, you'll want to remove it also from Jupyter. Let's first see which kernels are available. You can list them with:

```
jupyter kernelspec list
```

This should return something like:

```
Available kernels:
  myenv      /home/user/.local/share/jupyter/kernels/myenv
  python3    /usr/local/share/jupyter/kernels/python3
```

Now, to uninstall the kernel, you can type:

```
jupyter kernelspec uninstall myenv
```

Image from Wikimedia Commons

Login

Recommend 3 Tweet Share

Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Michael Ershov • 2 years ago • edited

Is it okay, that
pip install --user ipykernel
leaks out of anaconda environment
(C:\Anaconda3\envs\myenv)?

It stores files in Application Data folders
c:\Users\nameless\AppData\Roaming\Python\
c:\Users\nameless\AppData\Roaming\jupyter\

1 ^ | ˅ • Reply • Share ›

rhombus ➦ Michael Ershov • a year ago • edited

Agreed. Installing packages from inside the environment with 'pip install --user' defeats the purpose of having the virtual environment, the install leaks into the local user's environment.

^ | ˅ • Reply • Share ›

therden ➦ rhombus • a year ago

You're right.

If you know how install the ipkernel within a specified virtual environment, can you please describe or provide a link to it?

^ | ˅ • Reply • Share ›

Stephen • 10 months ago

Very nice tutorial.

In anaconda, I think `conda install ipykernel` is

In anaconda, I think `conda install ipykernel` is probably better.

1 ^ | ⌄ 1 • Reply • Share ›

Isaac Bang ➦ Stephen • 6 months ago

Thx

^ | ⌄ • Reply • Share ›

mushi • 7 months ago

It be nice if you shared the GUI approach as well. I don't think you have to connect the kernel if you do that. And isn't there another benefit to a virtual environment that is just share the whole environment and tell people the version of python, they won't have to install anything on their own.

^ | ⌄ • Reply • Share ›

Related Posts

How to Create Your Data Science Blog with Pelican and Jupyter Notebooks

26 Jul 2019