

Anaconda: The crème de la crème of Python distros

Summary

In this post I will briefly talk about a beautiful all-in-one Python distro, [Anaconda](#), and how to use it to create two virtual environments with Python 2.7 and Python 3.4 for all your Pythonistic needs.

The Past

For years, I was a hardcore advocate of the [Enthought Python Distribution \(EPD\)](#) but ever since Enthought decided to migrate things to [Enthought Canopy](#), I've found myself yearning a less... commercial, i.e. free, solution. Don't get me wrong the EPD was a god-send for those who didn't want to spend hours trying to get libraries like NumPy and SciPy onto their Windows boxes but I just found Canopy to be heavy, clunky, and altogether unnecessary. Enough bashing however, what doesn't work for some works for others and vice-versa.

The Present and, hopefully, the Future

Some time ago however, I stumbled across [Anaconda](#) by Continuum Analytics which answered all my Pythonistic needs.

In a nutshell, the foremost perks of this distro are:

- [Anaconda](#) is based on the [conda package manager](#) which IMHO is the best package manager to ever hit the Python scene. Don't take my word for but if [Travis Oliphant says so](#) then just jump on the bandwagon.
- Not only does [conda](#) access repos with a lot of the nastier (to install) Python packages, e.g. NumPy, SciPy, VTK, etc., but it makes creating virtual environments and installing whatever packages you want in them a breeze.
- I shouldn't have to say it but yes! It is freer than the air you breathe (there's taxes and junk on that)!
- Not only is the 'basic version' free which means it contains pretty much everything you would most likely need, but the good ol' folks at Continuum have created a lil' addon called [Anaconda Accelerate](#) which... well let me just quote what it does: *"Accelerate includes two packages that can be added to your Python installation: NumbaPro and MKL Optimizations. MKL Optimizations makes linear algebra, random number generation, Fourier transforms, and many other operations run faster and in parallel. NumbaPro builds fast GPU and multi-core machine code from easy-to-read Python and NumPy code with a Python-to-GPU compiler."* If you are an academic then you're in luck cause they actually give [Accelerate for free to most academic institutions](#). I mean, how AWESOME IT THAT?!

Personally, the ability to install all those nasty packages was what sold me. I mean, if I want to spend hours battling with esoteric package compilations then I'll go back to C++ thank you very much. Python is supposed

to be about high-productivity-minimal-nonsense coding.

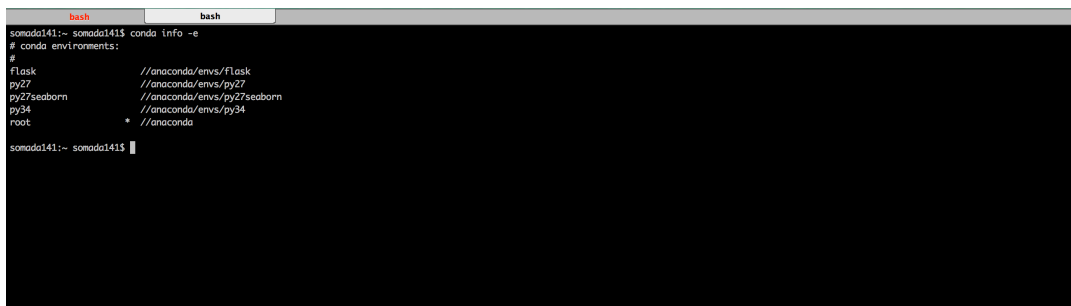
Creating the Environments

So if by this point somehow I've convinced you to go for Anaconda you'll find yourself wanting me to make good on my promise about helping you set up two virtual environments containing Python 2.7 and Python 3.4.

Now I am NOT going to join the holy war over whether you should migrate to Python 3 or not or rehash any of the arguments. Personally, I believe we will all have to eventually move onto Python 3 but for the time being Python 2 has the larger code-base. Period. Thus, its useful to have both on your system and be able to easily switch between them as needs and situations warrant.

After the Anaconda installation completes, open a command-prompt or terminal (I recommend [iTerm2](#) for Mac and [ConsoleZ](#), [PyCMD](#) or, if you can afford it, [TakeCommand](#) for Windows).

Enter `conda info -e`. The `conda info` command gives basic information on your Anaconda installation. The `-e` flag provides info regarding the environments you have on your machine. The output should look something like the figure below.



```
somado141:~ somado141$ conda info -e
# conda environments:
#
flask          //anaconda/envs/flask
py27           //anaconda/envs/py27
py27seaborn    //anaconda/envs/py27seaborn
py34           //anaconda/envs/py34
root           * //anaconda

somado141:~ somado141$
```

Output of the `conda info -e` command on my system

Naturally, in your system you should only be able to see the `root` environment but that's about to change. Lets first create a Python 2.7 environment with the following command:

```
conda create -n py27 python=2.7 anaconda
```

What this command does is `create` and environment named `py27` with an Anaconda distro of `Python 2.7`. Allow conda to perform all necessary changes and keep in mind that it will most likely download a truckload of packages so be patient.

Upon completion you can now enter the next command:

```
conda create -n py34 python=3.4 anaconda
```

Which, similarly to above, will create a `Python 3.4` environment named `py34`. Upon completion, re-running `conda info -e` should give you an output similar to the one in the figure below.

```
bash bash
(root)somada141:~ somada141$ conda info -e
# conda environments:
#
py27      //anaconda/envs/py27
py34      //anaconda/envs/py34
root      * //anaconda

(root)somada141:~ somada141$
```

Output of the `conda info -e` command after creation of the `py27` and `py34` environments

Switching between Environments

At this point the two additional environments should have been created and all should be well with the world. In order to enable or 'activate' these environments, you need to use the `source activate` (on Mac) or `activate` (on Windows) commands.

By default, the `root` environment would be the one active in any new session (which you can of course check through `conda info -e`). By entering `source activate` (on Mac) followed by the name of the environment you want to enable, e.g., `source activate py27` or `activate py27`, you will be able to switch to that environment.

You can find a lot more info on how to use conda under [here](#) and there's a wealth of information around on how to install and use Anaconda (or its minimalistic version [Miniconda](#)) on any of its supported platforms including CI systems like [Travis CI](#).

Thanks for reading!