Python - Create and manage your own Python environment

At NYUAD we maintain a centralised python distribution that is very lean without almost any module. The reason for that is that we want to avoid conflicts and cross-contamination from different users working on the same python distribution. So, how someone can use a specific set of modules then? We have two different approaches for that:

a) Users load the Python environment but install all the modules in a specific predefined location, ie: in their $HOME dir
b) Load Python environments using Anaconda

How to install Python modules in a personalised environment

NOTE: The localisation is based on Python "--user" user-based installation method.

WARNING: We recommend to use the same Python version to use localised Python modules (i.e. module load python/xxx).

1. To localise Python 2.7.11 add the following lines to your ~/.bashrc file

```bash
export PYTHON_VERSION=2.7
export PYTHONLOCALIZED=~/.python$PYTHON_VERSION
export PYTHONUSERBASE=$PYTHONLOCALIZED
export PYTHONPATH=$PYTHONLOCALIZED/lib:$PYTHONLOCALIZED/lib/python$PYTHON_VERSION:$PYTHONLOCALIZED/lib/python$PYTHON_VERSION/site-packages
export PATH=$PYTHONLOCALIZED/bin:$PATH
if [ ! -e $PYTHONLOCALIZED ]; then
    mkdir $PYTHONLOCALIZED
fi
```

2. Then, source the .bashrc file doing the following:

   `$> source ~/.bashrc`

   If this is the first time you do it, then you will see a new directory in your $HOME named python2.7
   This is where all your local installations will reside.

3. You are ready to install your modules

Manage your Personal Python Environment

NOTE for Python 2.7.11 on (including version 3)

Starting with Python 2.7.9 the module management tool "pip" is included in the central Python environment. So there is no need to install pip anymore and the usage is a bit different:

The following command will install the latest version of a module and its dependencies from the Python Packaging Index:

   `python -m pip install --user SomePackage`

Refer to the "pip" documentation web page for more information. Here's the list of available PyPI modules you can install using pip.
Installing Python Modules

If the module you need is not part of PyPI

```
$> python setup.py install --user
```

If the module is found in PyPI

```
> pip install --user Mako
Downloading/unpacking Mako
  Downloading Mako-0.9.1-py2.py3-none-any.whl (75kB): 75kB downloaded
Downloading/unpacking MarkupSafe>=0.9.2 (from Mako)
  Downloading MarkupSafe-0.18.tar.gz
  Running setup.py (path:/tmp/pip_build_benoit/MarkupSafe/setup.py) egg_info for package MarkupSafe
Installing collected packages: Mako, MarkupSafe
  Running setup.py install for MarkupSafe
building 'markupsafe._speedups' extension
  gcc -pthread -fno-strict-aliasing -g -O2 -DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/share/apps2/NEWAPPS/python/2.7.6/include/python2.7 -c markupsafe/_speedups.c -o build/temp.linux-x86_64-2.7/markupsafe/_speedups.o
  gcc -pthread -shared build/temp.linux-x86_64-2.7/markupsafe/_speedups.o -L/share/apps2/NEWAPPS/python/2.7.6/lib -lpython2.7 -o build/lib.linux-x86_64-2.7/markupsafe/_speedups.so
Successfully installed Mako MarkupSafe
Cleaning up...
```

Sometimes the PyPI module is hosted on a different host and you'll require more "typing" to install (you'll see the following message: “Some externally hosted files were ignored (use --allow-external getargs to allow).”)

```
> pip install --user --allow-external getargs --allow-unverified getargs getargs
```

Removing Python Modules
Uninstalling Mako:
/home/benoit/python/bin/mako-render
/home/benoit/python/lib/python2.7/site-packages/Mako-0.9.1.dist-info/DESCRIPTION.rst
/home/benoit/python/lib/python2.7/site-packages/Mako-0.9.1.dist-info/METADATA
(...)
Proceed (y/n)? y
Successfully uninstalled Mako

**Updating Python Modules**

> pip --upgrade getargs

**Listing Python Modules**
> pip search eigen
minieigen                 - Wrap parts of Eigen3, c++ library for basic math and
gometry.
slepc                     - SLEPc: Scalable Library for Eigenvalue Problem
Computations
sppy                      - A sparse matrix package based on Eigen.
Ceygen                    - Cython helper for linear algebra with typed memoryviews
built atop the Eigen C++
library
OpenModes                 - An eigenmode solver for open electromagnetic resonators
using the method of
moments
PyEigen                   - PyEigen is a Python wrapper for the C++ linear algebra
library Eigen.

> pip list
Mako (0.9.1)
MarkupSafe (0.18)
pip (1.5.2)
setuptools (2.2)

wsgiref (0.1.2)

> pip list --outdated

> pip show setuptools
---
Name: setuptools
Version: 2.2
Location: /share/apps/NYUAD/python/gcc_4.9.1/2.7.9/lib/python2.7/site-packages
Requires:

Related articles

- Use GPU nodes in Dalma
- Access Dalma
- Mount $WORK with SSHFS
- Python - Create Your Own Environment using Anaconda
- Enable X11 Forwarding on Mac OS X