Using and Installing Python

**Versions**

- **Ocelote**
- **ElGato**

**Python Packages**
- Installation & Package Policy
- Installing Python Packages Using virtualenv
- Accessing Custom Packages from a Jupyter Session

You can always check the current python modules available on either cluster with the command `module avail python`.

### Ocelote

There are four versions of Python available on Ocelote. The naming convention is different from the older clusters to support version 3. Python version 3 requires the `python3` command or `pip3 list` to differentiate. It is very different from Python version 2, so do not assume that Python 3 will work for you or that all older modules will work with version 3.

<table>
<thead>
<tr>
<th>Version</th>
<th>Accessibility</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python 2.6.6</td>
<td>system version (no module)</td>
<td>Accessible as <code>python</code> when no python 2.7.14 is not loaded</td>
</tr>
<tr>
<td>Python 2.7.14</td>
<td><code>module load python/2</code> or <code>module load python/2.7</code></td>
<td>Overrides system python 2.6.6</td>
</tr>
<tr>
<td>Python 3.5.5</td>
<td><code>module load python/3</code> or <code>module load python/3.5</code></td>
<td></td>
</tr>
<tr>
<td>Python 3.6.5</td>
<td><code>module load python/3.6</code></td>
<td>Loaded by default with <code>module load python</code>. This version contains many of the machine learning packages like Tensorflow that can be utilized on the Centos 7 / GPU nodes.</td>
</tr>
</tbody>
</table>

### ElGato

There are three versions of Python available on ElGato.

<table>
<thead>
<tr>
<th>Version</th>
<th>Accessibility</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python 2.7.5</td>
<td>system version (no module)</td>
<td>Accessible as <code>python</code></td>
</tr>
</tbody>
</table>
**Python Packages**

### Installation & Package Policy

We maintain a two-tiered approach to Python packages:

- **Tier 1**: We install the basic Python packages that are required by most users (these are mostly libraries rather than packages, such as numpy and scipy). This is done for the versions of Python that we install as modules. Adding some packages might force an upgrade of numpy for example, which might break a user’s environment that was dependent on the prior version.

- **Tier 2**: For packages that we do not provide we STRONGLY recommend the use of `virtualenv`, which is detailed below and provides a custom and easy to use Python environment.

### Installing Python Packages Using `virtualenv`

Useful overview of virtualenv and `venv`<br>InfoWorld Article: Python virtualenv and venv do's and don'ts

One of the best things about Python is the number of packages provided by the user community. On a personal machine, the most popular method today for managing these packages is the use of a package manager, like `pip`. Unfortunately, these require root access and are not a viable solution on the clusters.

There is an easy solution. You can use `virtualenv` to create a personal python environment that will persist for each time you log in. There is no risk of packages being updated under you for another user.

1. Set up your virtual environment. This is done one time only and will be good for all future uses of your Python environment.<br>   a. `module load python`, for example `python2` or `python3` on Ocelote<br>   b. `virtualenv --system-site-packages path` where path is where you want your python environment. You can use `~` for your home directory.

2. Implement the change<br>   a. `source path/bin/activate` where path is the same as step 1.b<br>   b. Append this line to your `.bashrc` if you want it to take effect on login. And add the `module load python` like in step 1.a Be aware that this python profile will load every time you log in so you are likely to get errors if you try to use a different python version for something else.

3. At this point you can pip install a package to your new environment for example `pip install pycurl`

4. When you log out and return later, your installed package will still be present. If you do not do step 2.b you will have to do 2.a each time you log in.

### Accessing Custom Packages from a Jupyter Session

An OOD Jupyter session is a different environment from working directly on the command line. This means to access and install custom packages, you will need a slightly different approach. To execute system commands, precede them with a `!`

If you need to create a new virtual environment, you may do so with the following (substituting in your own name and path):

```
!virtualenv --system-site-packages /path/to/virtualenv
```

Next, to install your custom package you’ll need to activate your environment and perform the install on the same line. This is necessary because system commands do not carry over from one line to the next.

```
!source /path/to/virtualenv/bin/activate && pip install pycurl && pip show pycurl
```
The `pip show` command is used to get the location of where the package was installed. You'll need this because the install will not automatically allow access to the package, meaning if you try to import it in the next step you'll get an import/module not found error. The next step will allow you to add the location to your python system path.

```
import sys
sys.path.append("/path/to/virtualenv/lib/python3.5/site-packages") # where pycurl is located in this example
import pycurl # success!
```

You can always use `pip show` with any virtual environments in your account allowing you to find and import anything you've previously installed. Just start with the source command above and exclude the `pip install` (you may want to double-check that the python virtual environment and the Jupyter version match before using/importing packages).

Another option to get a path automatically added to your environment is to include an export `PYTHONPATH` statement in your `~/.bashrc`. Be aware, though, that this adds the path to your environment every time you log in, so you may want to proceed with caution if you're switching between python versions:

```
export PYTHONPATH=/path/to/virtualenv/lib/python3.5/site-packages:$PYTHONPATH
```