Introduction to the Conda Package Manager

Learn how to create and manage Python environments

Fall 2024

Vincent F. Scalfani
Chemistry and Informatics Librarian
vfscalfani@ua.edu

Avery Fernandez
Data and Informatics Student Assistant
amfernandez7@crimson.ua.edu

Outline

Today, we will learn how to:

1. Create conda environments
2. Install packages in conda environments
3. Share and reproduce conda environments
4. Run code from conda environments

Slides

Download a copy of these slides here:

https://github.com/ualibweb/UALIB_Workshops

What is Conda?

Quoting from the docs:

“Conda is an open source package management system and environment management system that runs on Windows, macOS, Linux and z/OS. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.” [1]

[1] https://docs.conda.io/en/latest/

Conda and Miniforge

Conda is the package manager

Miniforge is a community packaged distribution of conda [1].

See also Anaconda or Miniconda [2-4]:

[1] https://github.com/conda-forge/miniforge
[2] https://docs.conda.io/en/latest/miniconda.html
[3] https://docs.anaconda.com/distro-or-miniconda/
[4] https://stackoverflow.com/questions/45421163/anaconda-vs-miniconda

What about venv/pip?

See “Understanding Conda and Pip” [1].

A few take-home messages from this reference:

1. Pip is generally for installing/managing Python software. If software is not all Python, you may need to manage some dependencies outside of python packages yourself.
2. Conda was designed to package and manage Python plus other software (e.g., can be common in some fields to have a mix of Python and shared C++ or other libraries)
3. Conda creates isolated environments
4. Pip can be used in isolated environments as well, but needs to be combined with other software like venv (virtual environments)
5. Pip can also be used within conda environments, so if a package is not available through conda-forge, you can install via pip.

[1] https://www.anaconda.com/blog/understanding-conda-and-pip

Why should I use Conda?

1. It’s great to be able to create separate non-dependent environments for projects (separate from your system Python installation too...)

2. Conda automatically manages dependencies for you (e.g., If you install the Python pandas package, conda automatically installs dependencies such as NumPy).

3. It’s easy to re-create environments and share your environment settings.

4. Some software may only be packaged thorough conda, so it’s convenient to use.

5. The conda-forge channel is community driven.

Install Miniforge

Download and install instructions: https://github.com/conda-forge/miniforge

Getting Help

Conda Managing Environments Docs:
https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#

Conda Managing Packages:
https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html

From within terminal:

```
conda --help

conda create --help
conda install --help
conda update --help
```

Creating Environments

After installing conda, launch a terminal:

Here is the general workflow:

# first create an environment

```
    conda create --name my_env
```

# activate the environment

```
    conda activate my_env
```

# install some packages

```
    conda install -c conda-forge package1 package2 ...
```

Creating Environments

Let’s create our first environment with Jupyter Lab, Pandas, and Matplotlib:

Tip: It’s best to install everything you need at the same time [1]

```
conda create --name my_env1
conda activate my_env1
conda install -c conda-forge jupyterlab pandas matplotlib
```

A convenient command is `list`, which shows all packages installed in the environment

```
conda list
```

To deactivate an environment: `conda deactivate`

[1] https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html

Creating Environments with More Specificity

Let's create a new environment with Jupyter Lab, Pandas, and Matplotlib, and a specific (older) version of pandas:

```
conda deactivate


conda create --name my_env2
conda activate my_env2
conda install -c conda-forge jupyterlab pandas=1.4.2 matplotlib
conda list
```

Conda and Pip [1]

Let’s create a new environment with Jupyter Lab, pandas, matplotlib, and then a package, rxn4chemistry, installed via pip.

install pip and any conda packages first, then install pip packages

```
conda deactivate

conda create --name my_env3
conda activate my_env3
conda install -c conda-forge jupyterlab pandas=1.4.2 matplotlib pip
pip install rxn4chemistry
conda list
```

[1] https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html

Updating Packages in Environments

Use conda update (e.g.):

```
conda activate my_env2
conda update -c conda-forge pandas
```

or for all installed packages in the environment:

```
conda activate my_env2
conda update -c conda-forge --all
```

Note: Not sure if there is an official best practice, but Vin typically creates a new environment with the updated packages, rather than updating an existing environment. This way, if something breaks, at least we have the old working environment!

Saving Conda Environments

One approach is to export your environment as a .yml file.

```
conda activate my_env3
cd Desktop
conda env export > my_env3_packages.yml
```

For compatibility across operating systems, use the --from-history flag [1]:

```
conda env export --from-history > my_env3_packages_hist.yml
```

Good idea to export both ways to make sure you capture version numbers.

[1] https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#sharing-an-environment

Re-Create the Saved Conda Environment

If you have an active environment, first:

```
conda deactivate
```

Then...

```
conda env create --file my_env3_packages.yml --name my_env3_new
conda info --envs
conda activate my_env3_new
conda list
```

Try Out Your Environments with Python

From terminal...

```
conda activate my_env3_new
ipython
```

```
Or from your IDE > Select conda env.
```

Thanks!

Questions?

Additional reading of interest:

Conda: Myths and Misconceptions by Jake VanderPlas
https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/

Vincent F. Scalfani
Chemistry and Informatics Librarian
vfscalfani@ua.edu

Avery Fernandez
Data and Informatics Student Assistant
amfernandez7@crimson.ua.edu