Miniconda 101

Package, dependency and environment management for any language
(e.g. Python, R, Java, JavaScript, C/ C++, FORTRAN)
https://docs.conda.io/en/latest/miniconda.html

Configuration on Windows 10 for NVU-Lyndon Atmospheric Sciences
(can work off campus and on other operating systems)

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. 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.

Common Python packages applicable to Atmospheric Sciences that can be installed with Miniconda include (but are not limited to): numpy, CDO, WRF-Python, wrfskewt, MetPy, python-awips, jupyter, and QGIS.

Conda as a package manager helps you find and install packages. If you need a package that requires a different version of Python, you do not need to switch to a different environment manager, because conda is also an environment manager. With just a few commands, you can set up a entirely separate environment to run that different version of Python, while continuing to run your usual version of Python in your normal environment.

Miniconda is "mini" because by default, it does not include as many Python packages as Anaconda does. A minimum number of default packages uses less space on hard drives.

Install Miniconda

1. Download the latest Miniconda installer for your platform (Windows, MacOS, Linux) from: https://docs.conda.io/en/latest/miniconda.html The 64-bit version works fine in most cases. This a free installer which contains Miniconda and a few default packages.

2. Double click the installer file to run it. Follow the prompts to install. Ensure **Just Me** is selected and follow the default options. This does not require administrator permissions to install. On Windows, Miniconda3 installs to this folder:

```
C:\Users\yourusername\Miniconda3
```

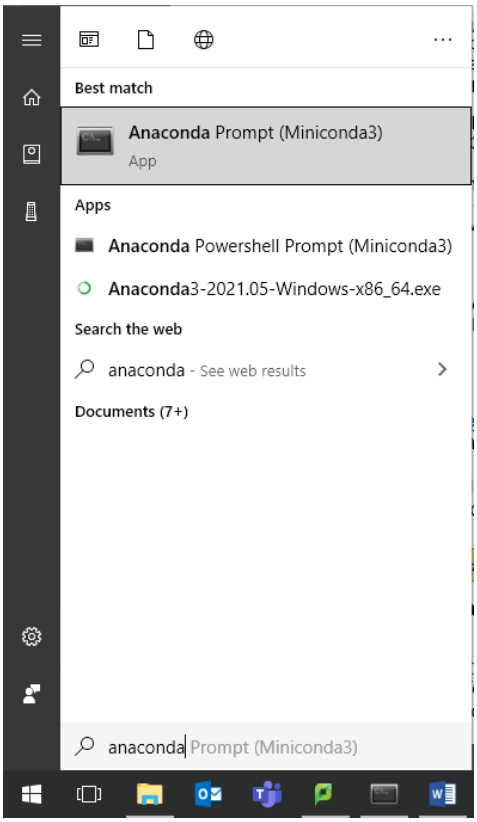
Create an environment from an environment.yml file

1. Download the desired environment.yml file (for example, the unidata environment yml file can be downloaded from: https://github.com/Unidata/python-training/blob/master/environment.yml) Save the yml file to a folder that's easy to find.
   For example, on Windows: C:\Users*yourusername*

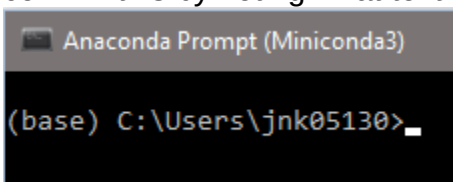
2. *Skip this step if your yml file is already named something other than environment.yml*
   Open the environment.yml file in a text editor (such as Notepad on Windows, or TextEdit on MacOS), to see what the name of the environment is that the yml creates. This is specified in the yml file with a line that says: **name:** Whatever text is after name, this is the name of the environment the yml file creates. Right click on the environment.yml file and rename the file to the name of the environment. This allows you to have several yml files for different

environments, and not inadvertently save over a file that's named environment.yml

3. Click the Start menu (or open Spotlight search in MacOS) and search for: **Anaconda**. Select the **Anaconda Prompt (Miniconda3)** app to open it.



4. By default, the Anaconda prompt will open to your Users folder (on Windows: C:\Users*yourusername*) You can confirm this by noting what text is shown next to (base)



If necessary, change directories to where the yml file is saved. A Windows Command Prompt cheat sheet may be useful.

Create and activate the environment with the yml file

1. To create the unidata environment, type the following and press Enter (if using a different yml file/environment, type in the name of that yml file/environment):

```
conda env create -f unidata.yml
```

An **"EnvironmentFileNotFound"** message will display if one tries to create an environment while not located in the folder where the yml file is saved, or if one specifies an incorrectly spelled yml file. Follow step 4, above, to change directories to where the yml file is saved.

Note about entering conda commands

Windows is not case-sensitive, while MacOS and Linux are always case-sensitive. When in doubt, always use lowercase conda commands on Windows, MacOS or Linux. Additionally, do not create directories or files with spaces in the name, instead use dashes or underscores.

2. To verify that the environment was created, type the following and press Enter:

```
conda env list
```

You should minimally see two environments: *base* and *unidata*
The active environment (the one you're currently working with) is specified with an asterisk when running conda env list, and is always listed in parenthesis to the left of where you type commands.
***Best practice:** create a new environment and install packages there first. **Don't install new packages directly to the base environment.** If you get error messages, you can always delete a new environment and start over. It's more difficult to recreate the base environment.*

3. To activate an environment, type the following and press Enter (in this example, the unidata environment that was created with the yml file; replace unidata with the name of the environment you want to activate):

```
conda activate unidata
```

4. *Optional step.* Another way to install packages (i.e. other than with a yml file) is to directly specify a package (or packages) you want to install. This will also automatically install any additional packages that are needed (called dependencies).

   Best practice: don't repetitively install additional packages on top of each other. *In other words, know which package(s) you need initially, and install those all at once with a single conda install command. Running conda install multiple times can cause conflicts with dependencies, requiring you to delete an environment and start over.*

   To enable Unix commands in Windows whenever your unidata environment is activated, type the following and press Enter:

```
conda install m2-base
```

   Type **Y** and press enter when prompted.

Open Jupyter Lab and complete an assignment

1. At this point, you may want to create a directory for the course by typing the following and pressing Enter:

```
mkdir enterfoldernamehere
```

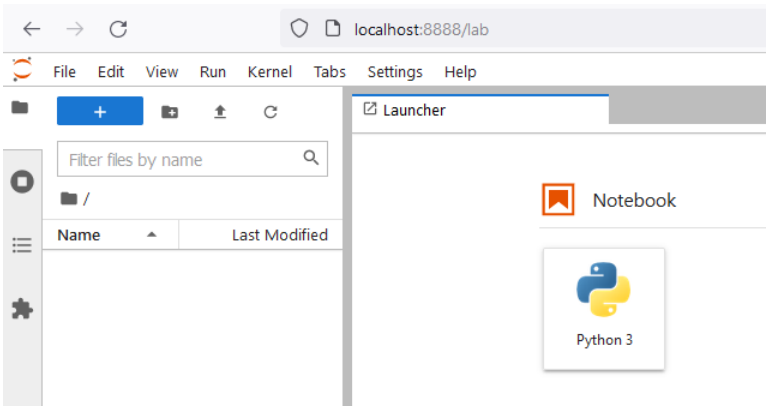
   and change to that directory:

```
cd whateveryounamedyourfolder
```

2. With the unidata environment activated, type the following and press Enter:

```
jupyter lab
```

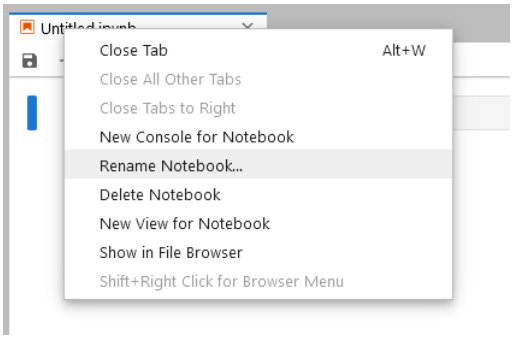
   This should open a web-based interactive development environment. Click the **Python 3** button below Notebook to start a new Jupyter Notebook file (the jupyterlab package needs to be installed in the environment in order to use it).



Questions or trouble, please email: Jason.Kaiser@NorthernVermont.edu

CONDA CHEAT SHEET: https://docs.conda.io/projects/conda/en/latest/_downloads/843d9e0198f2a193a3484886fa28163c/conda-cheatsheet.pdf

3. Right click the tab at the top and click **Rename Notebook…** to rename the notebook file to the desired name (i.e. other than Untitled)



NOTE

The Anaconda Prompt window **must remain open** for jupyter lab to run (if desired, it can be minimized, just not closed).

When finished with a notebook, click **File** and **Shutdown** to close the Jupyter Lab correctly. Then, close the Anaconda Prompt window.

Note

Save notebooks: modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu *File -> Make a copy ...*) and save the modifications on the copy.

Warning

Pay attention to not open the **same** notebook document on **many tabs**: edits on different tabs can overwrite each other! To be safe, make sure you open each notebook document in only one tab. If you accidentally open a notebook twice in two different tabs, just close one of the tabs.

If you don't see the environment you want in the list of environments (conda env list)

1. Create a new environment, and install the desired package(s) to this new environment. Type the following and press Enter:

Main package to install	**Environment creation and package install command (type one of these lines into the prompt)**
SHARPpy (packages for)	`conda create –n sharppyenv pyside=1.2.4 numpy=1.9.3`
cdo	`conda create –n cdoenv -c conda-forge cdo`
wrf-python	`conda create –n wrfpythonenv -c conda-forge jupyter matplotlib cartopy netCDF4 wrf-python`
qgis	`conda create –n qgisenv -c conda-forge qgis`

2. Activate the new environment by typing the following and pressing Enter:

```
conda activate myenv
```

Replace *myenv* with the desired environment name. By default, the active environment -- the one you're currently using -- is shown in parentheses () or brackets [] at the beginning of the command prompt.

3. That's it! You're ready to start working with Python packages installed to your currently-active environment. From here, you can follow the directions for:

    - WRF-Python and Jupyter Notebooks (start at step 5) - https://atmos.northernvermont.edu/ATMOS/wp-content/uploads/2019/03/WRF-PythonJupyter.pdf

    - SHARPpy (start at step 7) - http://atmos.northernvermont.edu/ATMOS/wp-content/uploads/2019/09/SHARPpy.pdf

    - CDO on Linux (start at step 11) - https://atmos.northernvermont.edu/ATMOS/wp-content/uploads/2019/03/CDO.pdf