Python environment overview

Python files

There are two basic file types for working with python code:

- Plain text file with extension `*.py`
- Jupyter notebook file with extension `*.ipynb`

Integrated Development Environment (IDE)

Although it is entirely possbile to edit and run python code using only a simple text editor and your command line, an IDE can make coding a more pleasent experience.

A few recommended IDEs:

- Jupyter Lab
- Visual Studio Code (VSCode)
- PyCharm
- Spyder
- Google Colab

Package managers

In this course we will primarily use `conda` and `pip` to manage python packages. `conda` also allows you to manage `python` as a package, and for each `python` version that you install you can still use `pip` as well if you want.

- `conda` (e.g., miniconda) <-- Also manages `python` and `pip` as packages in conda.
- `pip`: Python Package Index (PyPI) <-- Tied to a particular installation of `python`.

There are other package managers out there, but `conda` and `pip` are probably the most widely used.

Python Environments

You can think of a python environment as a folder that contains a particular version of python and a set of python packages that you installed into the environment. The key point is that you can have multiple python environments on your computer and each environment is completely separate from all other environments. Thus, each environment can have a different set of installed packages and even a completely different version of python.

Why should you almost always use a separate python environment for each project?

1. Different projects may require packages that are incompatible with each other.
2. The more packages you install into a single environment, the more you risk having a package dependency error. It is not a question of *if* this will happen, it is a question of *when*. Thus, you are asking for trouble if you keep installing all of the packages that you need for all of your projects into the same environment. If dependencies in an environment become sufficiently convoluted, you may

  need to simply delete the environment and remake it from scratch. If you have a separate environment for each project, this won't affect all your other projects.

In this course we'll use `conda` to manage our python environments:

- conda environments (e.g., neu365 as in the setup instructions)

There are other python environment managers out there such as venv, but `conda` can manage more than just python packages (e.g., python itself), and thus is a good all-purpose option. Blog on python package/environment tools with nice Venn diagrams.