Python, Anaconda, and Virtual Environments

Morgan McCarty

22 February 2023

1 Installing Python and/or Anaconda

1.1 Download and Install

https://www.anaconda.com/products/distribution
Download corresponding version of Anaconda for your operating system.
Follow the instructions using recommended settings

1.2 Adding to Path

This step will vary significantly for each OS

1.2.1 What is Path?

Path is an environmental variable which allows programs to be run from the command line/terminal without being directly accessed in the current folder (or without a directory being provided). This means that any program or folder of programs added to Path will function with just their command entered into any terminal.

1.2.2 Benefits of Adding Programs to Path

here are many benefits to doing this - notably you can execute programs with considerably less setup and it makes detection of system commands (built by those programs) much easier

1.2.3 Risks of Adding Programs to Path

There are system commands that, if overridden, can severely mess up your computer if they can no longer be executed (this is mainly an issue on Windows, thankfully this is easily reversible)

1.3 MacOS or Linux

Anaconda should add itself to PATH during setup

1.4 Windows

Anaconda recommends against being added to PATH and can function entirely through the Anaconda Prompt, but configuring for the command line is possible (and makes some use cases easier).

1. Find the folder to which Anaconda was installed and copy the file directory name to the ⟨install location⟩/bin/activate

2. Open the environmental variables settings menu and click the edit button. Find the Path variable.
3. Add a new entry and paste the copied directory name.
4. Reboot the command prompt/powershell and run "conda -V" to check if it installed correctly.
5. This should also add Python to path so running "python -V" or "python3 -V" should work as well now

2 Configuring Virtual Environment

2.1 What is a Virtual Environment?

Virtual Environments allow quick and easy changes to the installed packages and python version to enable concise project management.

2.2 Virtual Environment w/ Python

Python natively adds quick management of virtual environments using the "venv" command. This should work for any library added by PyPI (Python Package Index - anything you install using "pip"), but recently many libraries exist under Conda libraries.
https://docs.python.org/3/library/venv.html

2.3 Virtual Environments with Anaconda

Conda, once installed, has its own Virtual Environment tools which are completely compatible with any PyPI library.
You may be familiar with the "conda activate base" command that is run whenever Visual Studio opens into a Conda-based Python environment. (This activates the base Conda environment.) Here is how you create an environment:

1. First open a terminal/command prompt (Powershell on Windows)
2. The command "conda create -n environment" will create an environment named "environment" with the same packages as "base" (including Python version)
3. This environment can then be activated by using "conda activate environment" and exited with "conda deactivate" (Note: you can deactivate the base Conda environment as well if you wish to be inside the system installed Python.)
4. Once inside an environment you can use "conda install ⟨package-name⟩" or "pip install ⟨package-name⟩" and this will only be installed inside of that environment (which reduces potential conflicts).

https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

2.4 A Footnote on Containers

Containers (e.g. Docker https://www.docker.com) allow further abstraction down to the OS level beyond what virtual environments do. These are very useful for creating environments with completely different programs or with a programs set up in a way that is identical to someone else's set up.