# A Installing and Managing Python

**Lab Objective:** *One of the great advantages of Python is its lack of overhead: it is relatively easy to download, install, start up, and execute. This appendix introduces tools for installing and updating specific packages and gives an overview of possible environments for working efficiently in Python.*

## Installing Python via Anaconda

A Python *distribution* is a single download containing everything needed to install and run Python, together with some common packages. For this curriculum, we **strongly** recommend using the *Anaconda* distribution to install Python. Anaconda includes IPython, a few other tools for developing in Python, and a large selection of packages that are common to applied mathematics, numerical computing, and data science. Anaconda is free and available for Windows, Mac, and Linux.

Follow these steps to install Anaconda.

1. Go to `http://www.continuum.io/downloads`.
2. Download the Python 3.6 graphical installer specific to your machine.
3. Open the downloaded file and proceed with the default configurations.

Achtung!

This curriculum uses Python 3.6, **not** Python 2.7. With the wrong version of Python, some example code within the labs may not execute as intended or result in an error.

For help with installation of Anaconda, visit `https://docs.continuum.io/anaconda/install/`. This page contains links to detailed step by step installation instructions for each operating system, as well as information for updating and uninstalling Anaconda.

## Managing Packages

A *package manager* is a tool for installing or updating Python packages, which involves downloading the right source code files, placing those files in the correct location on the machine, and linking the files to the Python interpreter. **Never** try to install a package without using a package manager.

## Conda

Some packages are not included in the default Anaconda download but can still be installed using `conda`, Anaconda's package manager. See `http://docs.continuum.io/anaconda/pkg-docs.html` for the complete list. When you need to update or install a package, **always** try using `conda` first.

| Command | Description |
|---|---|
| `conda install package-name` | Install the specified package. |
| `conda update package-name` | Update the specified package. |
| `conda update conda` | Update `conda` itself. |
| `conda update anaconda` | Update **all** packages included in Anaconda. |
| `conda --help` | Display the documentation for `conda`. |

For example, the following commands attempt to install and update `matplotlib`.

```
$ conda update conda                # Make sure that conda is up to date.
$ conda install matplotlib          # Attempt to install matplotlib.
$ conda update matplotlib           # Attempt to update matplotlib.
```

See `https://conda.io/docs/using/pkgs.html` for more detailed examples.

NOTE

The best way to ensure a package has been installed correctly is to try importing it in IPython.

ACHTUNG!

Be careful not to attempt to update a Python package while it is in use. It is safest to update packages while the Python interpreter is not running.

## Pip

The most generic Python package manager is called `pip`. While it has a larger package list, `conda` is the cleaner and safer option. Only use `pip` to manage packages that are not available through `conda`.

| Command | Description |
|---|---|
| `pip install package-name` | Install the specified package. |
| `pip install --upgrade package-name` | Update the specified package. |
| `pip freeze` | Display the version number on all installed packages. |
| `pip --help` | Display the documentation for `pip`. |

See `https://pip.pypa.io/en/stable/user_guide/` for more complete documentation.

## Workflows

There are several different ways to write and execute programs in Python. Try a variety of workflows to find what works best for you.

### Text editor + Terminal

The most basic way of developing in Python is to write the program in a text editor and then run it using either the Python or IPython interpreter in the terminal.

There are many different text editors available for code development. Many text editors are designed specifically for computer programming which contain features such as syntax highlighting and error detection, and are highly customizable.

Some popular editors include:

- Atom (`https://atom.io/`)
- Sublime Text (`https://www.sublimetext.com/`)
- Notepad++ (Windows)
- TextWrangler (Mac)
- Geany (Linux)
- Vim
- Emacs

Once Python code has been written in a text editor, it can be executed in the terminal or command line. This can be done directly by running

```
$ python <filename>  # Do not include < >
```

or through the Python or IPython interpreter. The Python or IPython interpreters are opened by running

```
$ python
```

or

```
$ ipython
```

in the terminal respectively. In these environments you can directly execute Python expressions individually, or run entire Python files.

IPython is an enhanced version of Python that is more user-friendly and interactive. It has many features that cater to productivity such as tab completion and object introspection.

Note

While Mac and Linux computers come with a bash terminal built in, Windows computers do not. Windows does come with *Powershell* which works similarly. Python and IPython both work in Powershell, along with commands such as `conda` and `pip`.However, some commands in Powershell are different than their bash analogs, and some programs (such as git) do not work in Powershell. Windows users who want a bash terminal can download git bash at `https://git-for-windows.github.io/`.

## Integrated Development Environments

Integrated Development Environments (IDEs) provide a comprehensive environment with all the tools necessary for development combined into a single application. Most IDEs have many tightly integrated tools that are easily accessible.

### Eclipse with PyDev

Eclipse is a general purpose IDE that supports many languages. The PyDev plugin for Eclipse contains all the tools needed to start working in Python. It includes a built-in debugger, and has a very nice code editor. Eclipse with PyDev is available for Windows, Linux, and Mac OSX.

To download Eclipse, visit `http://www.eclipse.org/`. The PyDev pluggin can be installed through the Eclipse application or downloaded from the web. For installation instructions, see `http://www.pydev.org/manual_101_install.html`.

### Spyder

Spyder: `http://code.google.com/p/spyderlib/` Spyder is a Python IDE that comes included with Anaconda. Spyder comes with built in Python and IPython consoles, as well as interactive testing and debugging features.

## Jupyter Notebook

The Jupyter Notebook (also known as IPython Notebook) is a browser-based interface for Python that comes included as part of the Anaconda Python Distribution. It has an interface similar to the IPython interpreter, except that input is stored in cells and can be modified and re-evaluated as desired.

To begin using Jupyter Notebook, run the following command into the terminal:

```
$ jupyter notebook
```

This will open your file system in a web browser in the Jupyter framework. To create a Jupyter Notebook, click the *New* drop down menu and choose “Python 3" under the heading *Notebooks*. A new tab will open with a new Jupyter Notebook.

Jupyter Notebooks differ from different forms of Python development in that notebook files contain not only the raw Python code, but also formatting information. As such, Juptyer notebook files cannot be run in any other development environment. They also have a different file extention (`.ipynb`) rather than the standard Python extension (`.py`)

Jupyter Notebooks also supports Markdown, which is a text formatting engine, LaTeX formatting, and embedding images. This makes Jupyter Notebooks ideal for presenting code.