



**Virtual Environments**

**Python Environments**

In the next part of the lesson, you'll be given a workspace where you can upload files into a Python package and pip install the package. If you decide to install your package on your local computer, you'll want to create a virtual environment. A virtual environment is a silo-ed Python installation apart from your main Python installation. That way you can install packages and delete the virtual environment without affecting your main Python installation

Let's talk about two different Python environment managers: conda and venv. You can create virtual environments with either one. Below you'll read about each of these environment managers including some advantages and disadvantages. If you've taken other data science, machine learning or artificial intelligence courses at Udacity, you're probably already familiar with **[conda](https://conda.io/docs/" \t "_blank)**.

**Conda**

Conda does two things: manages packages and manages environments.

As a package manager, conda makes it easy to install Python packages especially for data science. For instance, typing conda install numpy will install the numpy package.

As an environment manager, conda allows you to create silo-ed Python installations. With an environment manager, you can install packages on your computer without affecting your main Python installation.

The command line code looks something like this:

conda **create** *--name environmentname*

**source** **activate** environmentname

conda **install** numpy

**Pip and Venv**

There are other environmental managers and package managers besides conda. For example, venv is an environment manager that comes pre-installed with Python 3. Pip is a package manager.

Pip can only manage Python packages whereas conda is a language agnostic package manager. In fact, conda was invented because pip could not handle data science packages that depended on libraries outside of Python. If you look at the [**history of conda**](https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/#Myth-#5:-conda-doesn't-work-with-virtualenv,-so-it's-useless-for-my-workflow), you'll find that the software engineers behind conda needed a way to manage data science packages (NumPy, Matplotlib, etc.) that relied on libraries outside of Python.

Conda manages environments AND packages. Pip only manages packages.

To use venv and pip, the commands look something like this:

python3 -m venv environmentname

**source** environmentname/bin/activate

pip install numpy

**Which to Choose**

Whether you choose to create environments with venv or conda will depend on your use case. Conda is very helpful for data science projects, but conda can make generic Python software development a bit more confusing; that's the case for this project.

If you create a conda environment, activate the environment, and then pip install the distributions package, you'll find that the system installs your package [**globally rather than in your local conda environment**](https://github.com/ContinuumIO/anaconda-issues/issues/1429). However, if you create the conda environment and install pip simultaneously, you'll find that pip behaves as expected installing packages into your local environment:

conda **create** *--name environmentname pip*

On the other hand, using pip with venv works as expected. Pip and venv tend to be used for generic software development projects including web development. For this lesson on creating packages, you can use conda or venv if you want to develop locally on your computer and install your package.

The video below shows how to use venv, which is what we recommend for this project.

**Instructions for venv**

Here are instructions about how to set up virtual environments on a macOS, Linux, or Windows machine using the terminal: [**instructions link**](https://packaging.python.org/guides/installing-using-pip-and-virtualenv/).

These are a few notes for understanding the tutorial:

* If you are using Python 2.7.9 or later (including Python 3), the Python installation should already come with the Python package manager called pip. There is no need to install it.
* env is the name of the environment you want to create. You can call env anything you want.
* Python 3 comes with a virtual environment package pre-installed. So instead of typing python3 -m virtualenv env, you can type python3 -m venv env to create a virtual environment.

Once you've activated a virtual environment, you can then use terminal commands to go into the directory where your Python library is stored. And then you can run pip install .. In the next section, you can practice pip installing and/or creating virtual environments in the classroom workspace. You'll see that creating a virtual environment actually creates a new folder containing a Python installation. Deleting this folder will remove the virtual environment.

Note that if you install packages on the workspace and run into issues, you can always reset the workspace; however, you will lose all of your work. So be sure to download any files you want to keep before resetting a workspace.

# From the exercise on virtual environments

Following the instructions from the previous video, convert the modularized code into a Python package.

You can put your code into the 3a\_python\_package folder in the workspace. Inside the 3a\_python\_package folder, you'll need to create a few folders and files:

\* a setup.py file, which is required in order to use pip install

\* a folder called 'distributions', which is the name of the Python package

\* inside the 'distributions' folder, you'll need the Gaussiandistribution.py file, Generaldistribution.py and an \_\_init\_\_.py file.

Once everything is set up, open a new terminal window in the workspace by clicking 'NEW TERMINAL'. Then type:

cd 3a\_python\_package

pip install .

If everything is set up correctly, pip will install the distributions package into the workspace. You can then start the python interpreter from the terminal typing:

python

Then within the Python interpreter, you can use the distributions package:

from distributions import Gaussian

gaussian\_one = Gaussian(25, 2)

gaussian\_one.mean

gaussian\_one + gaussian\_one

etcetera...In other words, you can import and use the Gaussian class because the distributions package is now officially installed as part of your Python installation.

If you get stuck, there's a solution in the 3b\_answer\_python\_package folder.

If you want to install the Python package locally to your computer, you might want to set up a virtual environment first. A virtual environment is a siloed Python installation apart from your main Python installation. That way you can easily delete the virtual enviornment without affecting your Python installation.

If you want to try using virtual environments in this workspace first, here is how to do it:

- There is an issue with the Ubuntu operating system and Python3 where the venv package isn't installed correctly. In the workspace, one way to fix this is by running this command in the workspace terminal: `conda update python` See: https://stackoverflow.com/questions/26215790/venv-doesnt-create-activate-script-python3 Then type `y` when prompted. It might take a couple of minutes for the workspace to update. If you are not using anaconda on your local computer, you can skip this first step.

- now, type this command to create a virtual environment `python -m venv venv\_name` where venv\_name is the name you want to give to your virtual environment. You'll see a new folder appear with the Python installation named venv\_name

- In the terminal, type `source venv\_name/bin/activate`. You'll notice that the command line now shows (venv\_name) at the beginning of the line to indicate you are using the venv\_name virtual environment

- Now, you can type `pip install python\_package/.` That should install your distributions Python package.

- Try using the package in a program to see if everything works!

================================/////////////////////////////////=============================

# Contributing to a GitHub project

Here are a few links about how to contribute to a github project:

* [**Beginner's Guide to Contributing to a Github Project**](https://akrabat.com/the-beginners-guide-to-contributing-to-a-github-project/)
* [**Contributing to a Github Project**](https://github.com/MarcDiethelm/contributing/blob/master/README.md)