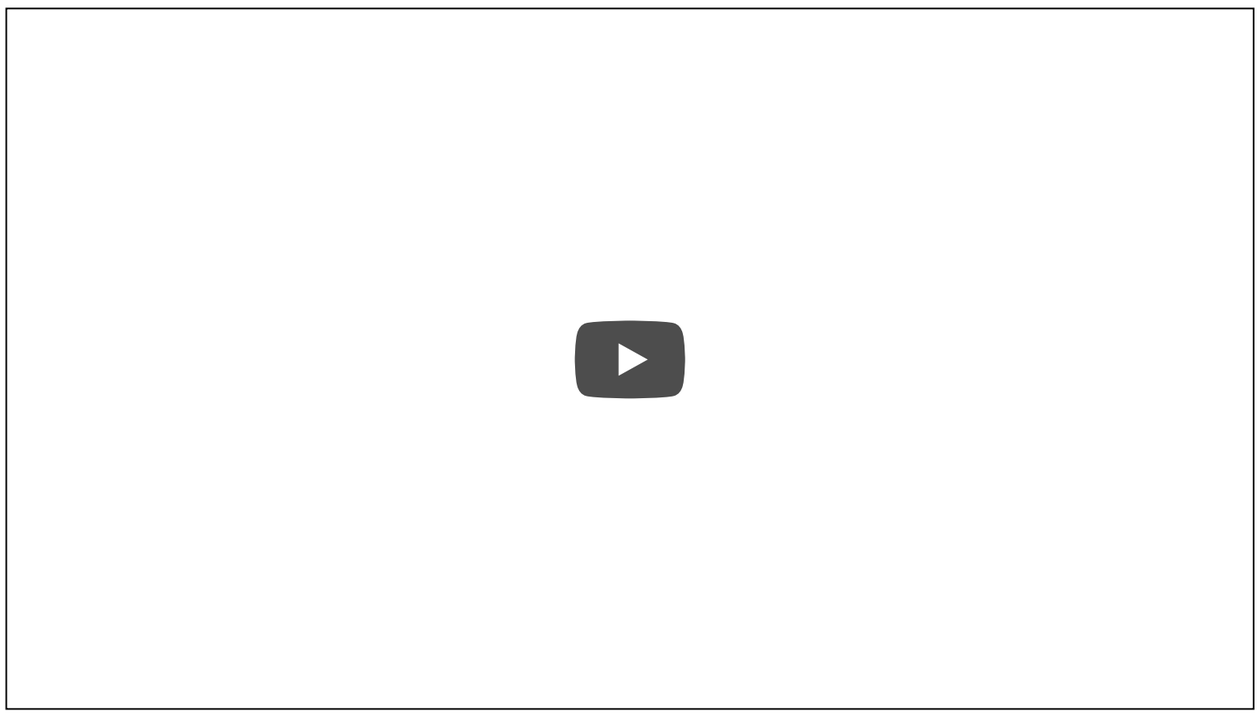
Making a Package

In the previous section, the Distribution and Gaussian code was refactored into individual modules. A Python module is just a Python file containing code.

In this next section, you'll convert the Distributions code into a Python package. A package is a collection of Python modules. Although the previous code might already seem like it was a Python package because it contained multiple files, a Python package also needs an `__init__.py` file. In this section, you'll learn how to create this `__init__.py` file and then pip install the package into your local Python installation.



What is pip?

Pip is a **Python package manager** that helps with installing and uninstalling Python packages. You might have used pip to install packages using the command line: `pip install numpy`. When you execute a command like `pip install numpy`, pip will download the package from a Python package repository called **PyPi**. However, for this next exercise, you'll use pip to install a Python package from a local folder on your computer. The last part of the lesson will focus on uploading packages to PyPi so that you can share your package with the world.

You can complete this entire lesson within the classroom using the provided workspaces; however, if you want to develop a package locally on your computer, you should consider setting up a virtual environment. That way if you install your package on your computer, the package won't install into your main Python installation. Before starting the next exercise, the next part of the lesson will discuss what virtual environments are and how to use them.

Object-Oriented Programming and Python Packages

A Python package does not need to use object-oriented programming. You could simply have a Python module with a set of functions. However, most if not all of the popular Python packages take advantage of object-oriented programming for a few reasons:

1. Object-oriented programs are relatively easy to expand especially because of inheritance
2. Object-oriented programs obscure functionality from the user. Consider scipy packages. You don't need to know how the actual code works in order to use its classes and methods.

NEXT