Module 12 – External Packages, Setting up Environments, and an Application

Instructions

1) If not already open, from the Anaconda Navigator Home screen, launch Sypder

2) In Spyder, go to “File”, then “Open”, then navigate to the location where you saved: “12_CensusActivity.py”, and select it to open the script file.

3) Overview

In this last module, we’ll go through an example of what to do when you conduct an online search for a Python package that appears to do exactly what you want, but it’s ***not*** part of the Python library or didn’t come along with the Anaconda distribution. This is also a good time to talk about setting up **environments.** Let’s address these together.

4) Introduction to Environments

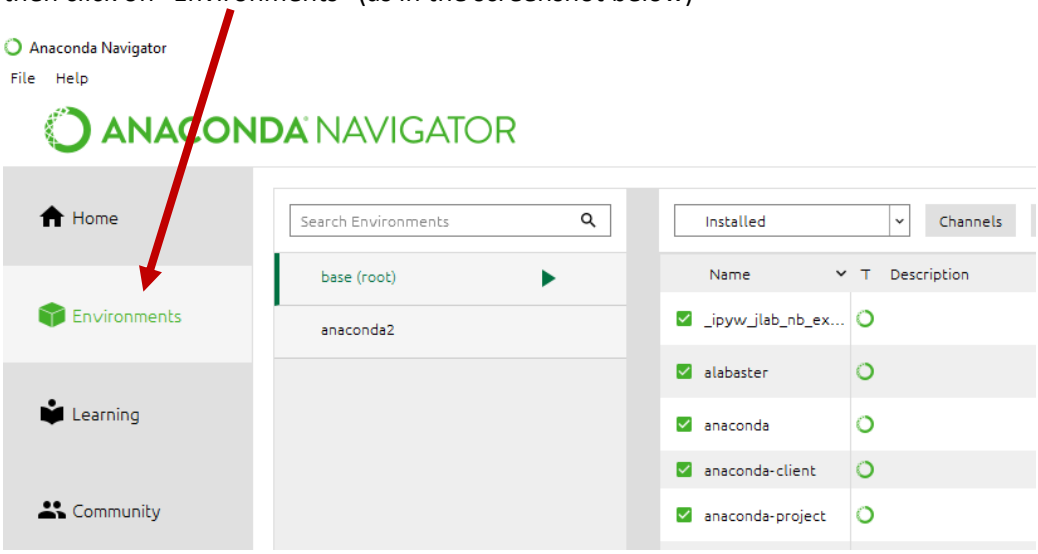
Up to this point, we’ve been working in one **environment**. If you installed the Ananconda Distribution for this first time for this module (which likely applies to most), that means we’ve been working in the **base (root)** environment. As an individual user up to this point, this is fine. But now is a good time to talk about setting up different environments. You’ll want to get comfortable setting up different environments for any/all of the following reasons:

- Certain projects and workflows are set to run in specific versions of Python and/or packages. You typically work with the newest and most updated of each, but this particular procedure is picky. Set up a specific environment for that project
- You have a project with packages that require multiple dependencies, and you know that updating one of them in your base/root directory will throw the complex web of dependencies into chaos
- You have colleagues that work in different versions of Python and/or packages than you, though you collaborate with them frequently. You’d like to keep the versions you work with separate from the versions they work with
- You or someone you work with set up a program that requires Python 2.x. You’d like to set up an environment where that program can run with the requisite dependencies and packages, but definitely keep it separate from your main environment.
- You want to use certain packages that you’ve found elsewhere, but they’re for a pretty specific use and you don’t want them (and their dependencies) mixed in with your “main” environment.

These of course are just a few examples, but they are common ones. This module will cover setting up new environments in Anaconda. As always, there are other ways to do this, but it is pretty straightforward to do so in this platform.

Go through the two tutorials below that show you how to set up two different environments. In each case, you’ll get a chance to install packages, too.

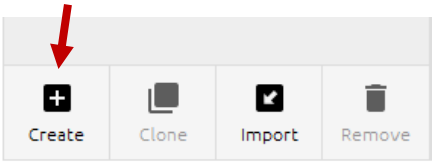
To view your current environment in Anaconda, go back to the main Anaconda Navigator home, and then click on “Environments” (as in the screenshot below)



You’ll see two in my screenshot above. I have an anaconda2 environment set up to work with programs that require 2.7 (programs that require arcpy/ArcGIS Desktop tools, if you’re curious). But the main one in which we have been working throughout this workshop is the (**base (root)**) one, which you can see is active, thanks to the green arrow next to the label at the right, and evidenced by the highlighted green text.

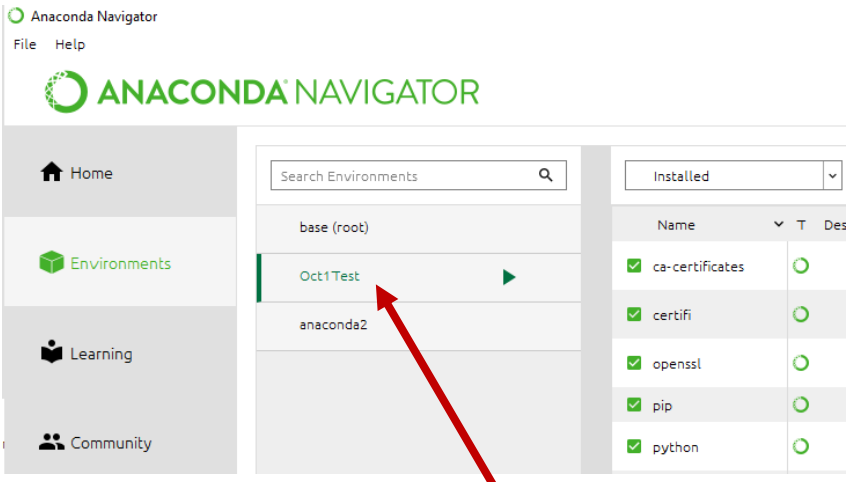
5) Creating New Environments and Installing Packages – Option 1, GUI + Terminal

There are a couple of ways to create new environments. One way is to simply click on the “Create” button at the bottom of the menu. So, let’s walk you through this process.



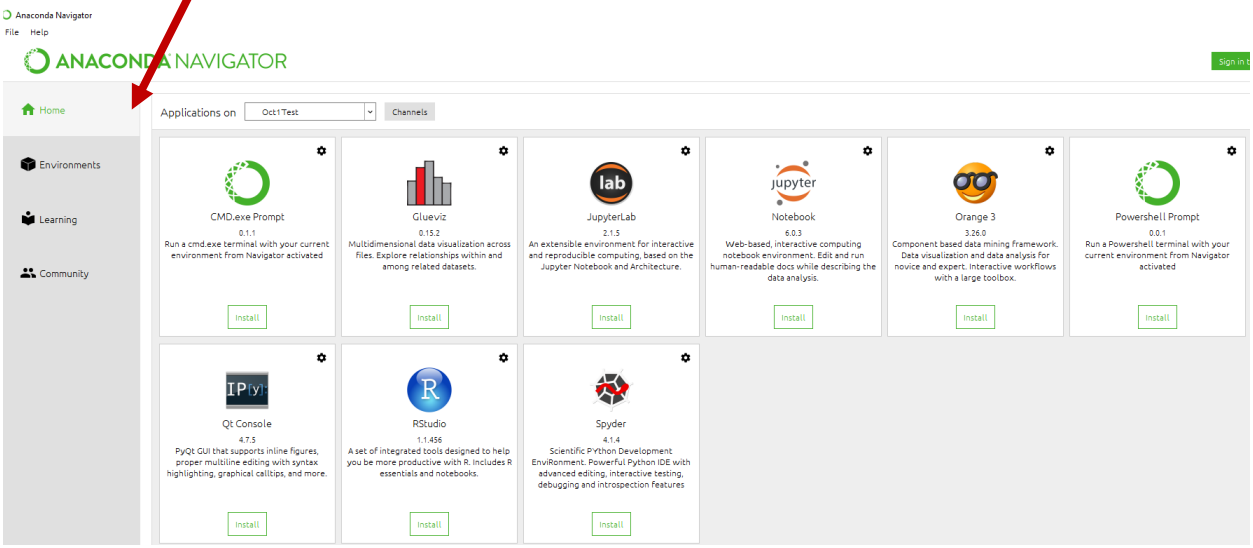
After clicking on “Create”, give your environment a name, indicate the version of Python you want this environment to work with (see above examples for why you would want older versions), and then click create, and then wait a bit as the environment is created.

I called mine “Oct1Test”, and selected Python 3.8. Depending on your previous software installations, that might be the only one visible, or you may see others. Just select Python 3.8 either way. Here is the output (next page) when it finished.



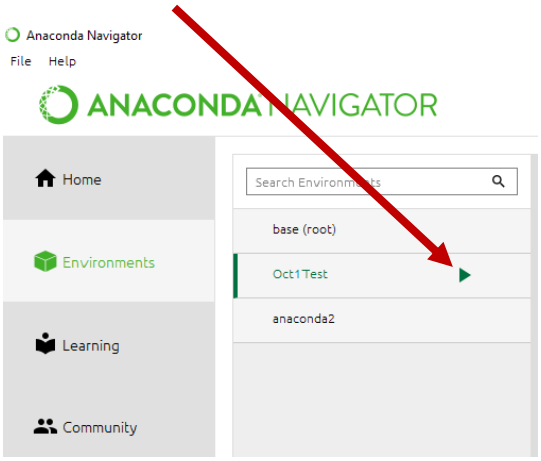
Notice my new environment is now activated. Notice, also, that the number of packages associated with this new environment is quite small in comparison to the base (root) one. Again, environments like this are often helpful for specific application areas, so this gives you the ability to only add what you need.

Now click on the home screen again. Notice that all of the Editors are not installed by default! But you can install and use the ones you want for this environment by clicking on the "Install" button under Spyder.

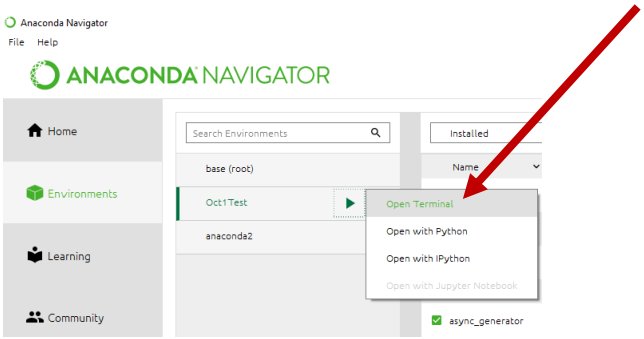


This is something to keep in mind when setting up new environments in Anaconda. You have to specify what editors to install, along with packages. For now, click on "Install" under Spyder to install the Editor for this environment (note: you can see my new environment name in the dropdown menu at the top of the screen). Let it run, which might take a little time. When it loads, you can open the 12_CensusActivity.py file and view it there, which now corresponds with your new environment. Quickly review the code. You'll notice that we're importing two modules in this example: one should look familiar (Line 80), but the other – and the focus for this activity – is new (Line 41). Here's the rub, though: if you try and run Line 80 right now, which has worked smoothly to date, you'll get a message in the output Console saying it can't find the module 'matplotlib.' That means we'll need to install that package in this environment, too!

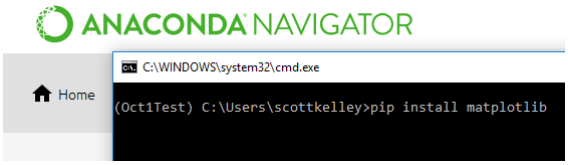
To do so, return to the Anaconda Navigator main screen, and click on the Environments tab again. See the green arrow next to your new environment name, as below? Click on it (a left-click in Windows).



When you do, a dialog as below appears. Select “Open Terminal”

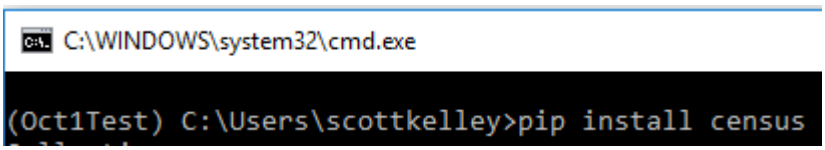


Enter the following command in the terminal, as below, then hit ‘enter’ to initiate it. Let it run, and now this package – and its dependencies – will be installed in this environment.



To be clear, that says: “pip install matplotlib”. ‘pip’ is Python’s built-in package installer, and it is quite helpful for situations like this. Pip is the workhorse for installing packages, so long as they are recognized primarily for use in Python. Any package in Python’s standard library (such as matplotlib) can be accessed and installed using it.

While we’re here, let’s install the **census** package that you’ll need, too (Line 41 in the script). This can also be accessed using pip (as below), by staying the terminal and typing “pip install census.”



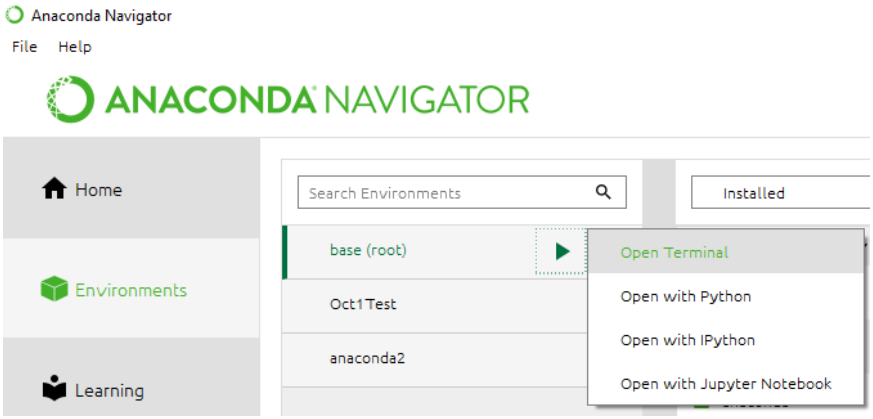
With these two packages installed in this new environment, you are ready to run the script that I have given you.

You can always remove/delete your created environments when you no longer need them. After you complete this module and are done with the workshop – and you no longer want this environment - simply click on the desired environment to activate it, and then click Remove, using the icon that looks like a trash can.

6) Creating New Environments and Installing Packages – Option 2, all Terminal

A more straightforward and powerful way to set up a new environment is to use the terminal to accomplish this, which also helps you install various editors and packages at the same time you create the new environment.

Go back to the Anaconda Navigator dialog, and click on Environments once more. Go back to your base (root) environment. Note it remains at the top of my list, with the environment I just set above in the middle. Click on the green arrow next to the base (root) environment, and select “Open Terminal.”



When terminal opens, type in the following:

conda create -n (yourenvironmentname) spyder matplotlib (or, as below)

```
(base) C:\Users\scottkelley>conda create -n oct1test2 spyder matplotlib
```

Not feeling especially creative today, as evidenced by my environment name of “oct1test.” Enter whatever name you wish, so long as it differs from the name of the environment you created above using the GUI.

Most importantly, though, this one line does the following: 1) creates a new environment, and calls it “oct1test2”, 2) installs Spyder in this environment, and 3) installs the matplotlib package. Far more efficient than the steps in Option 1, to be sure. Let it run for awhile.

If prompted to proceed with a list of packages in various versions, such as this:

```
Proceed ([y]/n)? y
```

enter ‘y’, (as above).

Now wait a few minutes, as this will take some time. When it's finished, you'll get prompted for a response one more time. As instructed in the terminal, enter

conda activate (yourenvironmentname), or as below:

```
(base) C:\Users\scottkelley>conda activate oct1test2

(oct1test2) C:\Users\scottkelley>
```

That should take far less time than creating the environment did. Notice, too, that the terminal is now pointed at my new environment (oct1test2), instead of the base (root) that I used to create this environment.

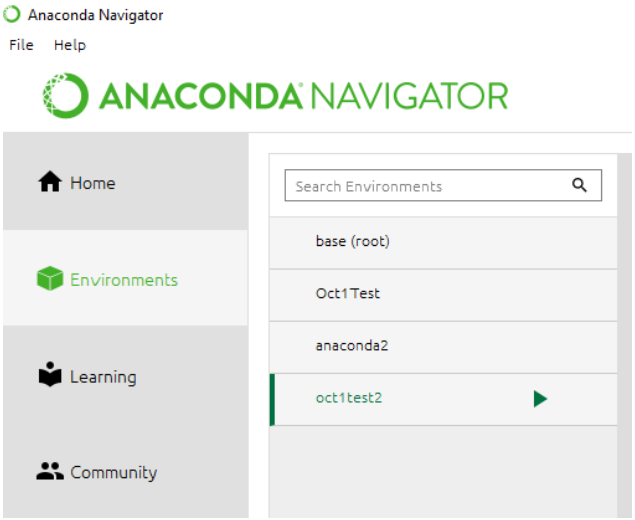
That's actually good, because this new environment, the **census** package is not yet installed: I only did so for the previous environment. Since we're already in the terminal, get some practice installing packages using **pip**. So, enter

pip install census

or, as below:

```
(oct1test2) C:\Users\scottkelley>pip install census
```

Once done, return to your Anaconda Navigator main screen, click on "Environments" (if needed), and see your new environment. Activate it by clicking on it, so that your screen appears as below, but with your environment names instead of mine:



Now, click on the "Home" tab, and notice that Spyder is already installed. Launch Spyder

Open up "12_CensusActivity.py", and try running Line 80 to confirm **matplotlib** installed. If you get no error message from the console after doing so, that confirms this package installed. Do the same for Line 41 to confirm the **census** package installed. If you get no error messages, you can now proceed to the script.

7) An Applied Activity – Accessing and Working with Census Data

We geographers commonly rely on data provided by the U.S. Census at varying spatial scales (national, state, county, census tract, census block group, census block) to understand how all sorts of phenomena are associated with these various spatial subunits. We not alone in this, of course: all kinds of social scientists access and use these data for various purposes.

This activity provides a glimpse at the type of analysis one can do by accessing census data. In this example, I gather estimates from the American Community Survey (2013-2017 5-year estimates) of those that 1) bike to work, and 2) walk to work for each census tract in Washoe County, provide two print statements that report the sum total for those that do each county-wide, and then make a simple histogram that shows a frequency distribution of commuters at the tract level in our county.

You can work through this activity in either of the environments that you created (the one created for Option 1 or Option 2), but it might be easiest to just stay in the second environment you created, since it's already open and the packages you need are installed.

One more thing before you can execute this script: take a look at line 18. In order to complete this activity, you'll need to get a US Census API key. Visit the website indicated in the comments on Line 18, sign up for a key, and when it's emailed to you, **replace the key that I've provided on Line 19 with yours. For those of you who work with Census data – you'll want your own key for later use, so it's worth doing this now. It is a straightforward process.

With your key replacing mine on Line 19, you are now ready to run the script.

You can actually run the full script instead of just a line or a selection, though are of course welcome to run individual lines or selections as you wish. Unlike the previous script files, though, **it is designed to be all run at once**.

Note that I have deliberately included something from just about every module that covered the various data types and fundamental operations. Notice where some of the operators we covered for dictionaries, lists, strings, and numeric data are applied, along with functions and definite loops. This was done to help you see other applied examples of these fundamentals.

Yes, the code could be streamlined and improved (and I encourage you to!), but my advice when learning Python is this: it's best to get something up and running that you can understand before setting about making it flashier.

Modify the script and output as you wish, or store it for later research purposes.