Getting Started With Python

Katharin Jensen

Introduction

If you've never used Python before, this guide is for you. It will walk you through installation, how to learn the basics, and a few other topics that are necessary for understanding the analysis code. If you already know Python and are just here to learn how the animation works, jump to Using matplotlib – Animation.

Table of Contents

Installation

This section goes over everything required to get Python installed and running.

Installing Eclipse

The first thing to do is install an IDE (Interactive Development Environment). The code was written using the Eclipse IDE, so that's what these instructions are for.

Linux Users

If you have the software manager, just open it and search for Eclipse. It'll install everything properly for you.

Everyone Else

Before you download Eclipse, you'll need to check whether you have 32-bit or 64-bit Java. To do that, open the command line and type "java -version". Look at what it says right after "Java HotSpot<TM>". If it says something like "64-bit Server VM" you have 64-bit Java. If it says something like "Client VM" you have 32-bit.

Now, go to https://eclipse.org/downloads/ and download the one labeled "Eclipse IDE for Java EE Developers." Get the bit version that matches your version of Java. This is very important! Downloading 64-bit Eclipse if you're running 32-bit Java or vice versa will cause Eclipse to crash whenever it tries to run.

After you've downloaded it, install it how you'd install any other program in your particular OS.

Installing Python

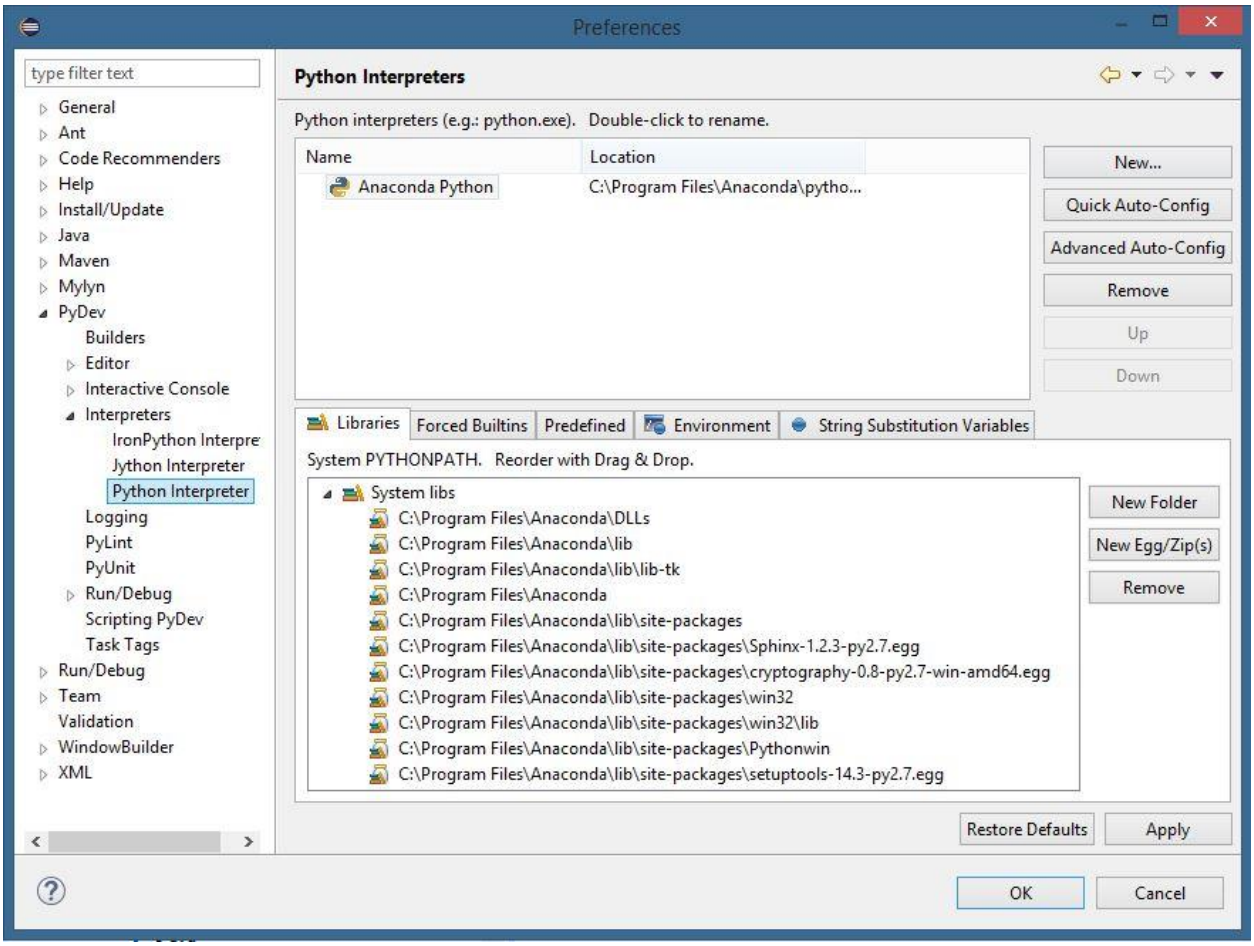
The code was written using the Anaconda Python distribution, so that's what these instructions are for. However, if for some reason you can't download Anaconda, any distribution that includes numpy and matplotlib should work just fine.

First, go to https://store.continuum.io/cshop/anaconda/. Click the large blue "Download Anaconda" button in the upper right corner of the page. Enter your email, then follow the instructions on the download page to install it. Make sure to get the one for Python 2.7.

Making Python Work with Eclipse

The first thing you have to do to be able to use Python with Eclipse is install PyDev. Detailed instructions for this, with pictures, can be found on the PyDev website at http://pydev.org/manual_101_install.html.

Once PyDev is installed, Anaconda has to be added as an interpreter. To do this, first open Eclipse. Go up to the top toolbar and click "Window," then "Preferences." In the window that opened, expand "PyDev" and "Interpreter," then select "Python Interpreter."



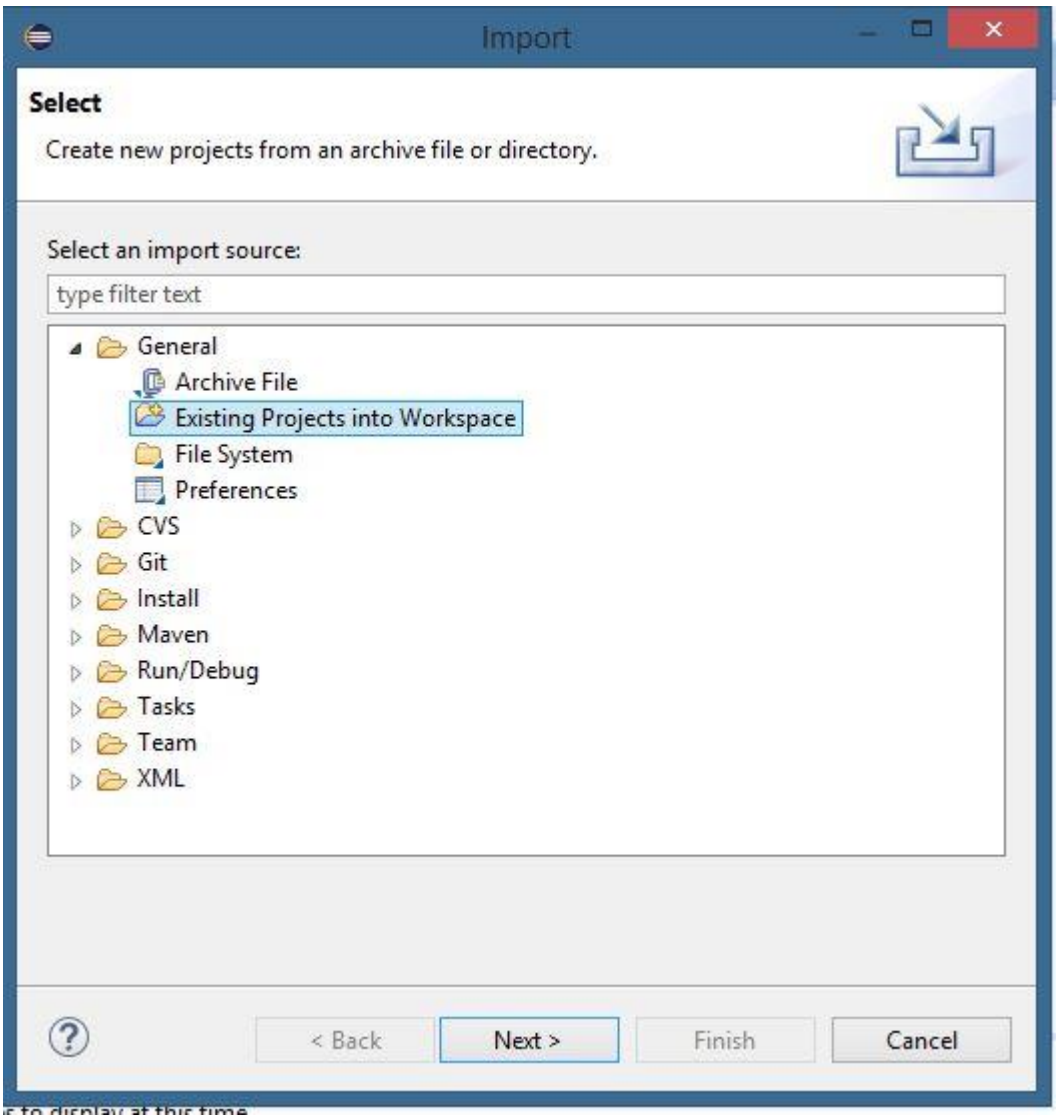
Click "New," then type in a name for your interpreter (it doesn't really matter what this is, just go with "Anaconda Python"). Then browse for the Python executable file. It'll be located in the folder Anaconda was installed into, and will be named "python.exe." If it isn't in the main Anaconda folder, see if there's a bin folder. If there isn't one, or it isn't in there, try doing a file search for it. Click "OK," then click "OK" on the next window to add all the required folders. Now Anaconda should be listed as an interpreter, and can be used with Eclipse.

Installing Imagemagick

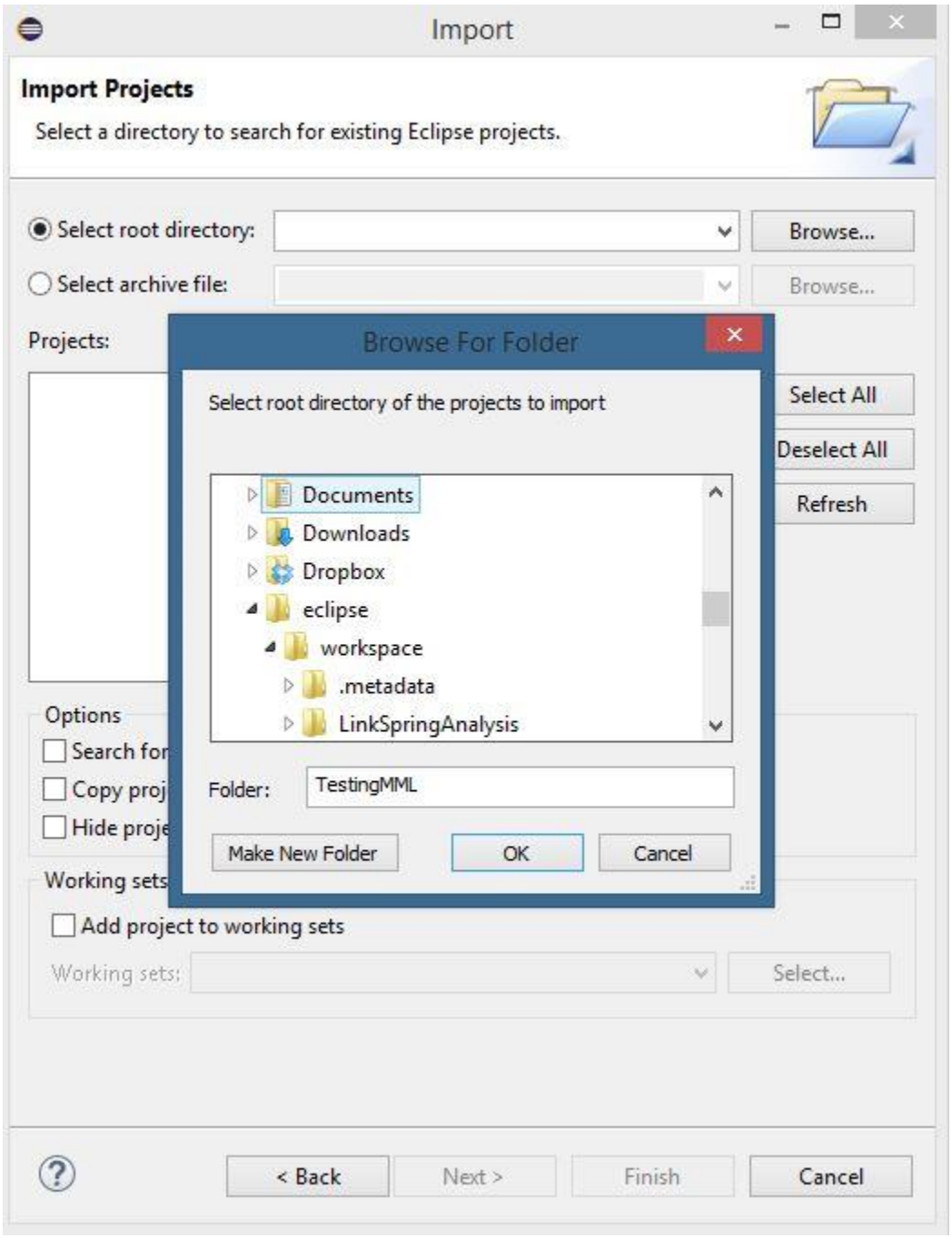
In order to save animations from the code as GIFs, an additional program called ImageMagick is required. To install it, go to http://www.imagemagick.org/script/binary-releases.php. Near the top of the page, find and click the link for your OS, then follow the installation instructions on the page.

Importing the Code

The last step is to import the code into Eclipse as a PyDev project. First, go to the folder where Eclipse stores projects (under users/yourname/eclipse/workspace, unless you told it to save somewhere else). Copy the folder "LinkSpringAnalysis" into the workspace folder. Next, open Eclipse, and go to File – Import… In the window that opens, expand "General" and click "Existing Projects into Workspace."



Check the “Select Root Directory” bubble, then click browse. Find the “LinkSpringAnalysis” folder and select it. Click “Finish.”



Now double check that the interpreter is set to Anaconda. To do this, go to the toolbar at the top and select Project – Properties. Click "PyDev – Interpreter/Grammar." Make sure the interpreter is set to Anaconda Python (or whatever you named your Anaconda interpreter). Now you're done! You should have everything ready to run the code.

Learning Python

Now comes the fun part – learning how to program in Python! The best place to start is at http://www.codecademy.com/. Sign up for a free account and find the Python tutorial. It covers everything you need to know to do basic programming in Python. It also covers some more general programming concepts, so you can learn even if you've never programmed before. After you finish the Codecademy tutorial, you'll know pretty much everything you need to understand all the code except the plotting and animations.

One important thing to remember is that if you can't figure out how to code something, or how something works, or why something doesn't work, Google and http://stackoverflow.com/ are your best friends.

Using matplotlib

The main thing the code uses that isn't covered in the Codecademy tutorial is matplotlib. It uses it for two things: plotting force vs. displacement curves and animation. The full documentation and some examples can be found at its official website, http://matplotlib.org/, but the parts that are actually used by the code are covered here as well.

Plotting

All plotting is done using *matplotlib.pyplot*. It is imported as *plt*. To make a basic plot, just type *plt.plot(*stuff*)*. It behaves very similarly to MATLAB's plotting commands, so if you're familiar with MATLAB it's easy to use. If you're not, http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot has all the arguments for the plotting function.

plt.axis(*[*xmin,xmax,ymin,ymax*]*) allows you to set the axis limits. *plt.xlabel(“*xlabel*”)*, *plt.ylabel(“*ylabel*”)*, and *plt.title(“*title*”)* allow you to add labels to the x- and y- axes and add a title. *plt.legend([“*line 1*”,”*line 2*”])* adds a legend, with the labels referring to the lines in the order in which they were plotted (e.g. line 1 would be associated with the first line plotted, and line 2 with the second line). In order to get the plot to show up, you have to add *plt.show()* at the end of all the plotting instructions. Once it shows the plot, any code after *plt.show()* will not run until you close the plot.

Animation

The animation uses *matplotlib.animation* imported as *animation*, *matplotlib.patches* imported as *patches*, and *matplotlib.path* imported as *Path*. Animating with matplotlib can be tricky. A tutorial on the basics of animation can be found here https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/. Some basics on drawing and animating shapes are here https://nickcharlton.net/posts/drawing-animating-shapes-matplotlib.html, and a tutorial on how to draw a patch with a path is here http://matplotlib.org/users/path_tutorial.html. The best way to really understand how animation works is to try those tutorials, and play around with them a bit.

The one big difference between the animation in the animating shapes tutorial and animating a PathPatch, like the LinkSpringAnalysis code does, is that, unfortunately, you can't just change values in the PathPatch object. You have to make a new patch for every frame.