

1.1: Preliminaries

1.1.1 Installing Python and Scipy

If you don't have Scipy installed yet, there are plenty of installation options, detailed here.

- If you are using GNU/Linux, Python is probably already installed, so just install Scipy using your distribution's package manager (e.g. `apt-get install python3-scipy` for Debian or Ubuntu).
- If you are using Windows or Mac OS, the easiest installation method is to use the Anaconda distribution, which bundles Python with Scipy and other packages you might need. Pick the 64-bit Python 3.5 version.

From now on, I'll assume that you have installed Python 3, which is the newest version of the Python programming language. The old version, Python 2, also supports Scipy, but it brings along lots of little differences, too many and annoying to enumerate. All new (non-legacy) Python code ought to be written in Python 3.

1.1.2 Verify the Installation

If you are using GNU/Linux, open up a text terminal and type `python`. If you are using Windows, launch the program `Python 3.3 → IDLE (Python GUI)`. In each case, this will open up a text terminal with contents like this:

```
Python 3.3.3 (default, Nov 26 2013, 13:33:18)
[GCC 4.8.2] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
```

The `>>>` part is a command prompt. Type the following:

```
>>> from scipy import *
```

After pressing Enter, there should be a brief pause, after which you get back to the prompt. (If you see a message like `ImportError: No module named 'scipy'`, then Scipy was not installed correctly.) Next, type

```
>>> import matplotlib.pyplot as plt
```

Again, there should be no error message. These two commands initialize the Scipy scientific computing module, and the Matplotlib plotting module, so that they are now available for use in Python. Note: in the future, you don't have to type these lines in by hand when starting up Python; we'll do all the necessary "importing" commands in our program source code.

Now let's do a simple plot of $y = \sin(x)$:

```
>>> x = linspace(0, 10, 100)
>>> y = sin(x)
>>> plt.plot(x,y)
>>> plt.show()
```

This should pop up a graph showing a sine function, in a window titled "Figure 1". Here's what these four lines of code did:

1. Create an array (a sequence of numbers), consisting of 100 numbers between 0 and 10, inclusive; then give this array the name `x`.
2. Create an array whose elements are the sines of the elements in `x`; i.e., a sequence of 100 numbers, the first of which is `sin(0)` and the last of which is `sin(10)`. Then, give this array the name `y`.
3. Set up an $x - y$ plot, using the `x` array as the set of x coordinates, and the `y` array as the set of y coordinates.
4. Show the plot on-screen.

If you don't understand *why* the above lines do what they do, don't worry. Let's just keep going for now.

This page titled [1.1: Preliminaries](#) is shared under a [CC BY-SA 4.0](#) license and was authored, remixed, and/or curated by [Y. D. Chong](#) via [source content](#) that was edited to the style and standards of the LibreTexts platform.