Last updated: 9/1/20

Math 260: Getting started
Fall 2020

Foreword:

A first step in learning programming is to setup your work environment. This means (i) getting python configured correctly for your computer and (ii) setting up a space for coding (an editor) and (iii) getting acquainted with your setup. You may want to experiment and try a few things - Having a virtual space you like helps to improve your coding experience, and everyone has their own preference.

If you run into any technical issues, feel free to ask for help, either through a Piazza post or by contacting me (you may also find searching for the error online helps, but just be careful, as it's possible to end up installing the 'wrong' python features/configuration that you would need to undo later. You can ask me if you are unsure).

Step 1: Install python

You must have Python 3.7 or 3.8 installed.[1] Check by typing `python` into your terminal/-console (it should load the console and state its version) or

```
python --version
```

To install python (if needed), you have several options. Note that the path you select matters for later steps, so you are choosing here how you want to manage python.

A) **Use a distribution (suggested!):** Anaconda (https://www.anaconda.com/) is a distribution of python that ships with a variety of features designed for scientific computing. As with any computing platform, it has the advantage of convenience and streamlined management, and the disadvantage that work outside its bounds may take a bit more effort. Fortunately, it does exactly the sort of work we do in the class!

B) **The direct way:** Go to https://www.python.org, select the version for your operating system and follow the download instructions (for OS X/Windows).

- For Linux, I suggest using your package manager to install/update (e.g. with `apt`). Make sure to select the correct version (the latest stable release package you can find).

[1] The current stable version of python is Python 3.8, released in 2019. Some packaged distributions may have 'outdated' versions (anaconda, for instance, uses Python 3.7). There are only minor additions from 3.7 to 3.8, so it does not matter which of the two you choose.

Step 2: Install scipy

The main packages required for the course are `numpy` (for fundamental numerical methods) and `scipy`, the scientific computing package for python that is built on numpy. The `scipy` package provides a powerful and convenient framework (like Matlab) for scientific computing in python. In addition, we use the `scipy.matplotlib` package for creating plots - installed as part of `scipy`, but worth noting.

To check whether a package is installed, just try to import it in the console:

```
import scipy
```

which returns an error if scipy is not installed. Some distributions of python have it by default (e.g. Anaconda). If needed, install using the **appropriate package manager**; in all cases see https://www.scipy.org/install.html for details.

- The native python manager is `pip`; follow the instructions at the URL above to install scipy and its dependencies via `pip`.
- If using Anaconda, scipy should be installed by default. For other packages you might need, use Anaconda's native package manager, `conda` (see https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html).

An example python program is posted to Piazza; either use that or find some other code and run to test your installation. To run from the command line, simply write

```
python myprogram.py
```

which will run the program and output to your console. Otherwise, open it in your editor and run it from there. You can also play around with the python console (`python`) one command at a time to make sure the packages are all working.

Step 3: Install jupyter

In addition, you will need `jupyter-lab`, which provides functionality for creating 'notebooks' containing text and code (some lecture notes/exampls will be jupyter notebooks).

Go to https://jupyter.org/install and follow the instructions for Anaconda or non-Anaconda installs. To check the install, grab the example notebooks posted to Piazza (one requiring `scipy`, one that does not) and see if they run. This ensures that the notebook can find the `scipy` package and that plotting works properly.

Step 4: Get an IDE

Technically, you could write your code the 'hard way' in a text editor, running from the command line. It's recommended to find an IDE (integrated development environment) that provides all the tools (write, run, manage, debug) in one place. Note that for the relatively simple programs of this course, you may want a more 'lightweight' IDE with less features (simplicity can be a good thing!).

Here are a few options that I have personally used and liked:

- The simplest is IDLE, which is included in many python distributions.
- `spyder` is designed for scientific computing and ships with Anaconda. It has a nice set of features, plus convenient plotting display. computing).
- PyCharm is another full-featured IDE, not specifically forscientific computing.

In each case, you'll need to tell the IDE which python distribution to use. If it came with the distribution (e.g. spyder with Anaconda), it should be pre-configured. Otherwise, you may need to give your IDE the path to the right python (e.g. `usr/bin/python38` or whatever it may be) in the appropriate settings. Note that you can figure out the right path by typing `which python` into the terminal in Linux/Mac (otherwise your installation instructions from Step 1 should have this information).

As noted at the start, The most important feature of an IDE is that it **feels convenient** after you have gotten used to it. This is a matter of personal preference, and can take a bit of testing and configuration. If it works for you, then use it.

Extra step: A quick LaTeX refresher

Formatting math is essential to be able to communicate. We will be using LaTeX for:

- Writing descriptive text in `jupyter` notebooks (maybe)
- Writing the discussion component of the project (also maybe)
- Sharing calculations/notes/work and posting to Piazza (definitely)

You can get by without LaTeX for the most part, but it's *very* useful when discussing math. A good starting point is the tutorial at
https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes