In this class, we will use Python, and in particular, we will use Jupyter notebooks for assignments. You will gain exposure to this in Homework #1.

The following descriptions will guide you through the installation of `Python` 3. You are welcome to do the assignments in `Python` 2.7. The guide will also show you how to set up a virtual environment, which enables you to install and do work without adversely affecting your global installation of `Python`. You are also welcome to handle `Python` package management on your own. Many students last year preferred to install `anaconda` and subsequently install any remaining packages later on (i.e., through `pip install` or `conda install`). If you choose to use your own global installation instead of virtual environments, it is not possible for us to help debug your unique installation, as any prior packages you may have installed before this class may interact negatively. This is why we walk you through virtual environments in this handout; we will provide a file that will install all the packages you need for each assignment.

Installing `Python` for OS X

To install Python 3+ on Mac OS X, first install `homebrew` via the following link:

`https://brew.sh/`

`homebrew` is a package manager for OS X. It mirrors `sudo apt-get install` for Ubuntu, but on Mac OS X.

To install `Python3`, call:

```
brew install python3
```

This should install `python3` on your machine, which is the version of `Python` we will use in this class. If you have an installation of `python`, this ought not interfere with the installation. Calling `python` will still run Python 2.7, while calling `python3` will run `Python` 3. You are welcome to use Python 2.7, however, and we will do our best to ensure the code runs smoothly on both. Bear with us as we assess compatibility.

Note: If you installed `homebrew` previously and then updated your OS to High Sierra, you may have to uninstall and reinstall `homebrew`.

Using `virtualenv`

In this class, we recommend you use `virtualenv`. Documentation can be found here:

`https://virtualenv.pypa.io/en/stable/`

At a high-level, a virtual environment creates an environment where you may install packages, etc. that aren't installed on your main system `Python`. This is useful for separating out projects. Further, if you accidentally break your `Python` during some package installation, but you were in a virtual environment, you can simply delete the virtual environment and start anew.

To install the `virtualenv` package, use `pip`:

```
sudo pip install virtualenv
```

To create and use a virtual environment, go to the directory you'd like your project to be in:

```
cd path_to_hw1/
virtualenv -p python3 .env     # This creates the virtual environment
source .env/bin/activate     # This activates the virtual environment
# Do some work...
deactivate     # This deactivates the virtual environment
```

If you'd like to return to your virtual environment, simply go back to the directory where you started the virtual environemnt and run `source .env/bin/activate` and you'll be where you left off.

Installing packages for an assignment

For each assignment, there will be some packages you need to install. These include, e.g., `jupyter notebook` and `numpy`. Instead of having to list these out manually each time, each assignment will come with a `requirements.txt` file with all the packages to be installed. These packages can be installed via:

```
pip install -r requirements.txt
```

To be clear, when you start up the virtual environment for the first time, you should install the packages in the `requirements.txt`. i.e., you should run these commands.

```
cd path_to_hw1/
virtualenv -p python3 .env     # This creates the virtual environment
source .env/bin/activate     # This activates the virtual environment
pip install -r requirements.txt     # This installs all the packages
# Do some work...
deactivate     # This deactivates the virtual environment
```

Once you've installed all the packages you need in a virtual environment, you're done and the next time you load `virtualenv`, you can continue where you left off.

Installing `Python` and `virtualenv` for Windows

Download `Python` package from `https://www.python.org/downloads/release/python-364/` and install it. Run cmd as administrator and run these commands:

```
pip install virtualenv
cd path_to_hw1/
virtualenv -p path_to_python3/python.exe .env     # This creates the virtual environment
.env\Scripts\activate.bat     # This activates the virtual environment
pip install -r requirements.txt     # This installs all the packages
# Do some work...
.env\Scripts\deactivate.bat     # This deactivates the virtual environment
```

Launching `jupyter` notebook

After finishing installing packages for assignment, the `jupyter` notebook will be ready to use. Run these commands:

```
cd path_to_hw1/
jupyter notebook
```

This will launch a new browser window (or a new tab) showing the notebook dashboard. Or you can visit `http://localhost:8888/` to open the dashboard.

When started, the `jupyter` notebook can access only files within its start-up folder (including any sub-folder). Make sure your relevant files are on the desired path.

Do I have to use `virtualenv`?

As stated above, no, you don't. However, if you choose not to use virtual environment, it is up to you to make sure all dependencies in the code are installed globally on your machine. This is not difficult to do. But if things break, it is not possible for us to help debug each student's unique installation, as each computer setup is different, and the bug may be any package you have previously used or installed interacting negatively. That is why we prefer to have you use virtual environments and have provided you the `requirements.txt` file for each assignment.

If you don't want to use `virtualenv` and prefer a standard `Python` install, a popular installation is the `anaconda` distribution, which comes with a large collections of packages.