Setup start guide

The following instructions should help you get setup with all necessary softwares quickly and ready to run your first job on your local machine (we'll get to cloud later). This workshop will include the installation instructions of following tools and best practices to build your project:

- Anaconda -- python environment and package manager
- Unix Shell -- interface to communicate with your machine
- PyCharm - an Integrated Developement Environment (IDE)
- Git -- team collaboration system (time permitting)

Note that some of the lectures and/or workshops after this may use Collaboratory for examples, and it's a fine approach to getting started or playing around with an idea, but for projects that will grow beyond small we strongly recommend switching to these these tools sooner rather than later.

Anaconda (Miniconda)

Installation

Anaconda is a package manager, an environment manager, and Python distribution that contains a collection of many open source packages. Anaconda makes it straightforward to quickly set up new python environments, making it much easier to set up the standad python scientific stack (numpy, scipy, pandas, etc).

We recommend Miniconda, which just includes the package and environment manager, rather than the full Anaconda distribution - it's substantially smaller, and you can easily install anything Anaconda provides using the conda package manager.

1. Go to the https://docs.conda.io/en/latest/miniconda.html and choose a Python 3.x (recommended version in this course) graphical installer.
2. Download the appropriate Python 3.7 bash installer
3. From a terminal, cd to your downloads directory (usually just `cd ~/Downloads`) and run `sh Miniconda3*.sh` In a new terminal window type `conda env list`.

You should see something similar to

```
$ conda env list
# conda environments:
#
base                     /Users/marc/anaconda3
...
```

Note: If all of the commands in step (3) are new or confusing, it might be worth working through a tutorial or two in order to get acclimatised to a UNIX command line. Here are a few:

https://www.datacamp.com/community/tutorials/shell-commands-data-scientist

https://ryanstutorials.net/linuxtutorial/

https://www.learnenough.com/command-line-tutorial/basics might be good places to begin.

Creating environments

1. In a terminal window enter

   ```
   $ conda create --name my_environment python=3.6 ipython numpy pandas
   ```

   The first part, `conda`, is the conda package/environment manager, `create` is the command, `--name` is an option followed by the name of your new environment, and everything after that is a package version. You could just do `conda create --name deleteme`, but it's convenient to add a few common things right away, and it's useful to specify the Python version.

2. Activate your environment simply with `conda activate my_environment`

3. Your prompt will now look something like

   ```
   (my_environment) [marc:~] $
   ```

   where your normal prompt is now prepended with the loaded environment name, `(my_environment)`. If you run `python` or `ipython` or `pip`, everything is now happening *within* the current environment i.e. you install a package with `pip` or `conda`, that package will be installed into `my_environment` (try typing `which python` to see this)

4. You can deactivate or unload the environment with `conda deactivate`, or by activating a different environment

Install TensorFlow 2.0 in your conda environment

1. `pip install tensorflow==2.0.0-beta1`
2. That's it! Try opening `ipython` and doing an `import tensorflow` to test it out.

More info

https://conda.io/projects/conda/en/latest/user-guide/getting-started.html

PyCharm

PyCharm is a popular Python IDE from JetBrains (other popular IDEs they make include IntelliJ IDEA and PhpStorm).

PyCharm has a good code editor, language aware code completion, strong syntax error catching, a powerful debugger with a graphical interface, version control integration, an extensive plugin community, and a million other things.

1. Download and install the Student license of PyCharm (which is really the Professional version except free): https://www.jetbrains.com/student/. We recommend against using Community Edition because it's missing the remote interpreter feature, which means you can't do live debugging of code running on the cloud
2. Start and create a new project with PyCharm.
3. Start the project using "New environment using" and select Conda from the dropdown menu. This will create a fresh conda environment that you can also access from the terminal as described in the earlier section.
4. Alternatively, if you already have a Conda environment you want to use for the project, you can select Existing Interpreter. Click the "...", select Conda environement on the left, click the "..." again, and find the python executable (from a terminal with your desired conda environment activated, type `which python` and copy the output)
5. (Note that you can easily swap interpreters on an existing project later)

Beyond this, PyCharm is huge and it's worth familiarizing yourself with its documentation, https://www.jetbrains.com/pycharm/documentation/

Git

Git is the most widely used version control system in the open source world. A version control system is tracks and manages changes to files over time in a project. Working with version control is a critical skill to have in basically any coding context. Some of the benefits of learning Git include:

- Nothing version controlled using Git is ever lost if you committed it, so you can always see older versions of code.
- Version control allows for collaboration and conflict handling (multiple people making different changes to the same lines of code)

Going into depth on git usage might be out of the scope of this workshop, so we recommend you take some time to learn it. Here's a good guide for getting started here.