1B_Coding_Environments

May 12, 2021

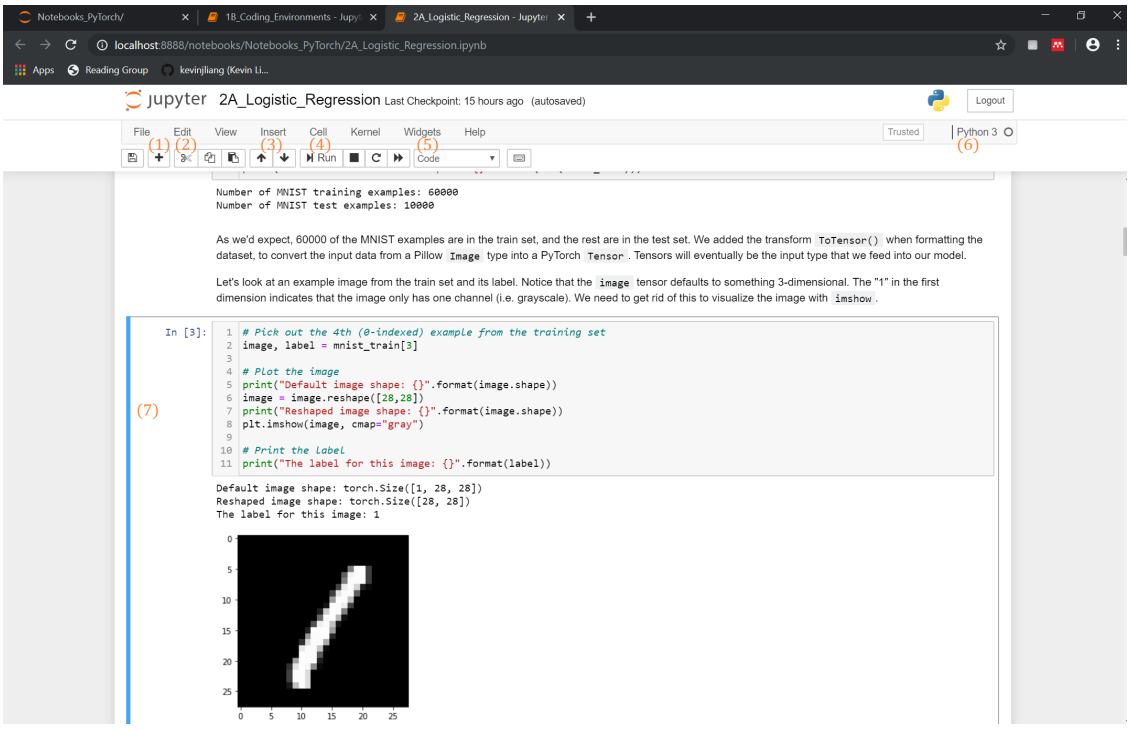
1 Coding Environments

You are welcome to use any coding environment that you like for this tutorial, so if you have prior experience or a workflow set-up already, feel free to continue to use it. If not, a few suggestions are listed here:

1.1 Jupyter Notebook (IPython): *(Recommended for this class)*

IPython is an interactive version of Python that allows users to use a shell to run chunks of Python code, as opposed to an entire Python script or program. For those familiar with MATLAB or R, it is very similar to using their console/command line, but with Python code.

Project Jupyter is a spinoff project of IPython that provides a modern, browser interface for IPython:



In a Jupyter Notebook, we can mix working IPython code, markdown text, graphics, magic commands (such as lines of bash), and much more. You'll notice that many of lecture materials were prepared using Jupyter notebooks.

1.1.1 Installation

If you followed the PyTorch installation instructions in Notebook 1A, congrats, you already have Jupyter installed! You can skip to the Launching section below.

If you *didn't install Jupyter* but have Anaconda, Jupyter is easily conda installable:

```
# Activate your PyTorch environment, if you haven't already:
# conda activate pytorch
conda install jupyter
```

If you *didn't install Jupyter* and choose not to use Anaconda, Jupyter is also easily pip installable:

```
pip install jupyter
```

1.1.2 Launching

To open, switch to your working directory and type the following into your terminal/command prompt:

```
jupyter notebook
```

Depending on your OS, Jupyter will either automatically open a tab with Jupyter running, or provide a local host address for you to paste into your favorite browser.

In your browser, you should see your working directory appear. IPython notebooks will have a book symbol before their names and can be opened by clicking on them. In this course, you'll mostly be working through pre-existing notebooks, but should you want to create a new noteboook, new notebooks can be created by clicking on the "New" dropdown menu in the upper right corner.

1.1.3 Use

Jupyter notebooks are composed of a collection of cells. New cells can be created with the "+" button (1), or the `A`(bove) or `B`(elow) key while in Command mode. Move a selected cell relative to other cells with the arrow buttons (3). Delete a cell by either clicking the cut button (2) (Note: this is "Cut", so it will also copy the contents), or typing `d` twice in Command mode. Cells can be run by clicking on the play button (4), or with `Ctrl`+`Enter`. Various options for running multiple or all cells are also listed under Cell > and Kernel >. For this class, you'll mostly use cells containing "Code" or "Markdown", as indicated by the dropdown menu in the ribbon at the top (5).

Keyboard shortcuts can be used to perform many common tasks in Jupyter. These can be found at Help > Keyboard Shortcuts. Many of these hotkeys require a cell to be in Command mode, which is signified by a blue stripe on the left side of the cell (7) (as opposed to the green stripe for editing mode). Command mode can be entered with either the `Esc` key, or by clicking on the side bar (7). Editing mode can be entered by clicking on the appropriate cell, or hitting `Enter`.

Code cells Cells that run code in the kernel set during the creation of the notebook. This should default to Python 3 for you, and you can verify the kernel by looking in the upper right corner (6). Code cells can contain import statements, multiple lines of Python scripts, function definitions, etc., and outputs (or errors) are shown below the cell after they are run.

A note about IPython cells: the results of previously defined variables persist, *even if you delete the cells that originally defined them*; this is important to remember, as it can be quite startling the first time you experience a variable you thought you deleted coming back from the dead. Since cells usually contain fragments of a program and can be run in any order, you need to be sure to run any pre-requisite cells first. Normally this is as simple as finding the cells that haven't been run yet, running them, and then trying again. It is good practice to put your cells in the order they should be run so that other people (or you) opening your notebook can just "Cell > Run all cells" to replicate your results.

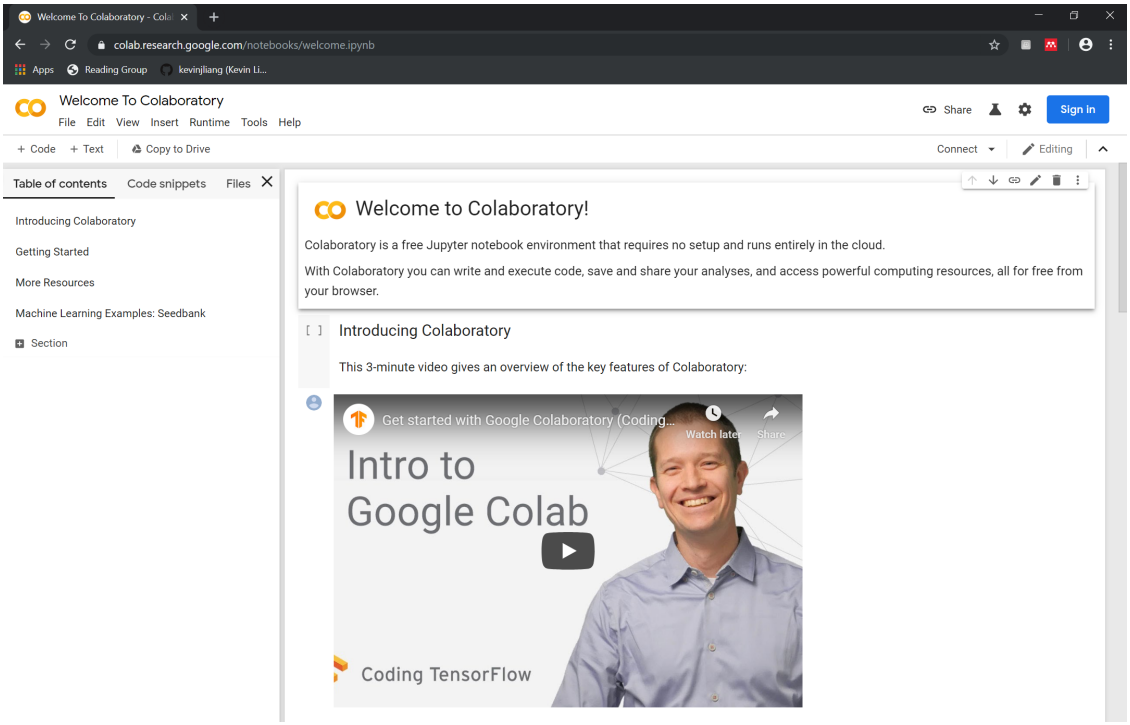
Markdown cells Cells for inserting text. These are not lines of code, but are formatted nicely when run. All of the instructions in these notebooks are written in Markdown. It's easier seen than explained, so if you're curious, double-click on any cell consisting of text (including this one) to see how they were writen.

1.1.4 Additional Resources

More info on Jupyter here: http://jupyter-notebook.readthedocs.io/en/latest/

1.2 Google Colaboratory

If you do not want to install PyTorch and Jupyter locally on your machine, Google Colaboratory (Colab) is a great alternative which is also free! If you're familiar with Jupyter IPython notebooks, you'll feel right at home using Colab as well; many of the core features and much of the usage feel the same.



Colab is cloud-hosted service, so your browser serves as a connection to one of Google's data centers, which does the computation for you before returning the results. As a result, unlike Jupyter, which runs locally, you'll need an internet connection to use Colab. You can upload IPython notebooks from your Google Drive, or try out notebooks written by others. For example, you can download the notebooks for this course, and open them up in Colab to run.

One of the best parts about Colab is that it has **free GPUs (and even TPUs) available to run code**! This is actually an incredible offering, as GPUs greatly accelerate neural network computation and are usually fairly expensive to buy, set-up, and maintain. GPUs allow you to run more complex neural networks than what we will show in this course, as we try to make everything runnable on just CPUs. Keep in mind though that these free GPUs are primarily meant for education and research. For other purposes (or heavier computation), Colab will likely direct you to Google Cloud Platform, for a paid service.

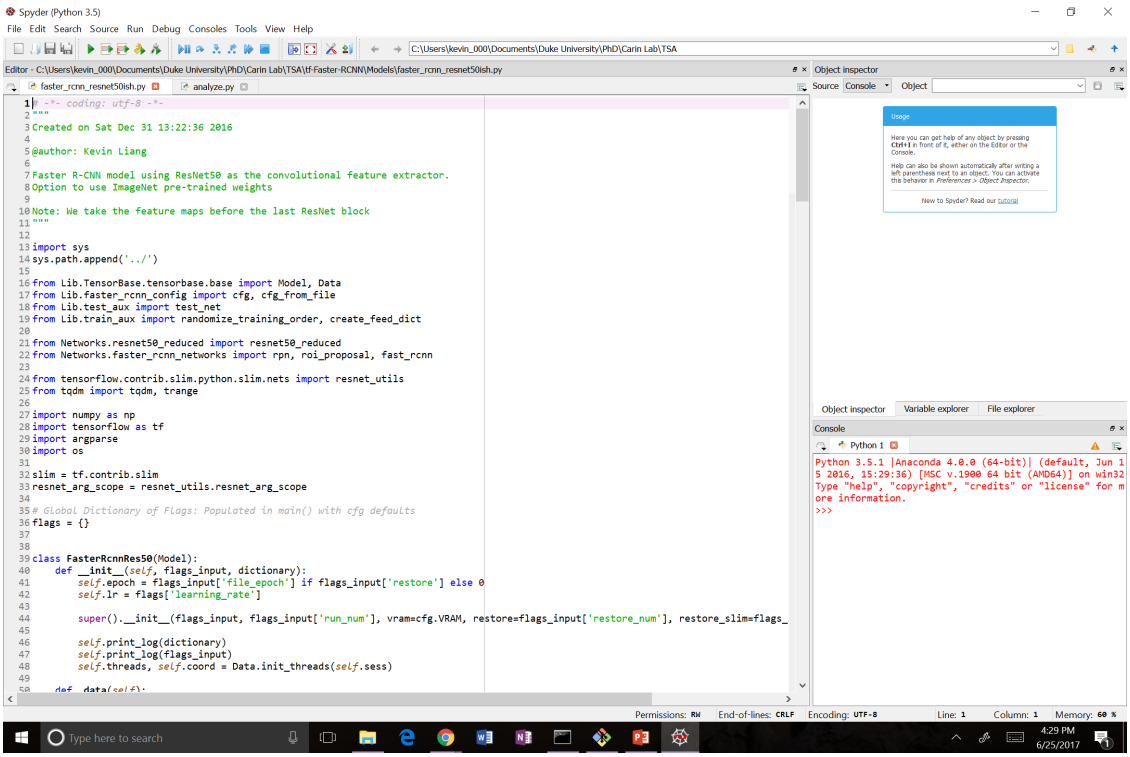
1.2.1 Installation and Launching

No installation required (besides a web browser)! Just go to https://colab.research.google.com/notebooks/welcome.ipynb and sign in with a Google account. It's free, as long as you stick to the basic functionalities. The `welcome.ipynb` notebook above serves as an introduction/tutorial, if you want to learn more!

1.3 Spyder

Spyder is an open-source integrated development environment for Python. Spyder is a bit more traditional of an IDE compared to Jupyter or Colab; it's primarily used to write full Python (`.py`)

files to be executed as scripts or programs, though an IPython console (bottom right, in the figure below) is also included. Anyone who has coded in MATLAB's IDE or RStudio should feel right at home. It looks like this:



1.3.1 Installation

Again, if you followed Notebook 1A's installation instructions and installed Anaconda as your Python distribution, you already have Spyder! Spyder is included in the Anaconda distribution. However, you may need to make sure that Spyder is installed in the same environment as PyTorch in order to use it:

```
# Activate your PyTorch environment, if you haven't already:
# conda activate pytorch
conda install spyder
```

Alternatively, you can go to "Tools>PYTHONPATH Manager", click "Add Path", and enter in the path to your Anaconda environment.

If you *didn't install Anaconda* but already have Python, Spyder is also easily pip installable:

```
# NOTICE: Only do this if you didn't already install Anaconda
pip3 install --upgrade pip
pip3 install spyder
```

1.3.2 Launching

If using Anaconda, before launching, make sure to activate your environment so Spyder can find your packages. To open, type the following into your terminal/command prompt:

```
spyder
```

1.3.3 Additional Resources

More info on Spyder here: https://docs.spyder-ide.org/

1.4 Text editors: Vim, Emacs, Sublime, Notepad++, gedit, nano, etc.

There's nothing forcing you to use an IDE specialized for Python. Python files are just text with a `.py` extension. This means that if you prefer something more lightweight, you can do all you coding in your favorite text editor, and when you're ready to run your code, just run the following in your shell:

```
python [your_python_file].py
```

This will run `[your_python_file].py` with your default Python version.

If this is your default preference, I'm guessing you probably don't need help with installation or usage.