Homework 0 - Git and Python

When preparing your homework submissions, don't forget to `git fetch, git status, git pull` *before you issue any other commands in Git – this is to guard against you changing a document that someone else is working on in the same directory*[1].

Don't forget to `git commit` *(with* `-m` *comments) frequently as you work. This allows other users to see how your work progressed and it automatically backs up your work as you produce. Thus, you're less likely to lose any of your work and/or so you can revert to earlier versions of your work as needed.*

Remember to comment your code carefully with your initials before every comment (as in # JCR `I just wrote a Python comment to document a change.` *Remember to provide an informative header for* **every** *function that you write.*

Homework

1. In your working directory of the class repository, create a `homework/` (or similarly obvious) directory for your homework submissions. Also create a directory for `HW0/`. Write your submission in this directory.

2. Write a Python function that is passed the values m and b in the expression for a straight line $y = mx + b$, and; (**i**) generates 10 floating point numbers in the rage $0 - 10$ at random along the x-axis *(hint: look at the* `numpy.random.random` *function)*; (**ii**) uses $y = mx + b$ to recover the appropriate y values; and (**iii**) scatters each of the 10 points in the $\pm y$ direction according to a random offset drawn from a Gaussian of standard deviation 0.5 centered on a given y value (*hint: look at the* `numpy.random.normal` *function*). Your function should return an array of the x values, an array of the y values, and an array of the uncertainty on y (this y_{uncert} is always 0.5).

3. Use `numpy.polyfit` to fit a straight line to your generated x, y, y_{uncert} data and recover the best fitting values of m_2 and b_2 (which may differ from your original m and b).

4. Plot your data, the original line ($y = mx + b$) from which your data were drawn, and the best-fitting line ($y = m_2x + b_2$) that `numpy.polyfit` derived, to the screen.

5. Use `matplotlib` to create a hardcopy of your plot in PNG (PNG is suitable for displaying on webpages). Try to make your plot look as professional as possible. Consider whether the x and y ranges displayed in the plot are appropriate. Use color and style to distinguish lines and points. Use an appropriate symbol to display your points and don't forget to display the error bar. Consider adding labels. Think about font choice and size. Make sure your characters and lines are of appropriate thickness to display on a webpage *(hint: for some examples, see e.g. publication-quality plots in Molina et al., 2019, arXiv:1905.02166.)*.

6. Finally, create a function that links steps $2 - 5$ together to create a single Python module that is passed a value of m and a value of b at the UNIX command line, generates some mock x, y, y_{uncert} data, fits a line to the mock data, plots everything to the screen and

[1] this shouldn't be a big deal unless we're working collaboratively, but you should get into the habit *now*.

makes a hardcopy in PNG format. Use the `time.time()` function to add a timer to your code and print useful comments and times to the screen after each of the steps $2-5$ in this master procedure *hint: consider the* `if _name_ == ''_main_''`: *command listed on the ASTR8080 links page).*

7. Create a README file to submit with your homework that describes how to run your HW0 assignment, plus any other necessary information.