AST 3100 – Mini-course: Coding (2018)

Marten van Kerkwijk

March 14, 2018

1 Syllabus

Lectures Friday March 16, 23 and April 6, 13 (last one TBC); on each day, a more lecture-style meeting from 9:30 to 10:30 in AB88, and two hands-on sessions from 11-12 (lounge? TBD) and 12-1 (AB 88; latter mostly for those taking cosmology);

Lecturer Marten van Kerkwijk, MP 1203B, 416-946-7288, mhvk@astro.utoronto.ca

Office hours Drop by my office, or by appointment

Web page `http://www.astro.utoronto.ca/~mhvk/CODINGMINI/`

notes pdf

1.1 Synopsis

Almost all of us use code, be it to analyze data, design an instrument, simulate something, or do semi-analytical analysis. This course's goal is to help acquire improved tools with which to tackle programming needs. Since this only really works if you have a purpose for those tools, the focus will be on putting them to practice, i.e., on how to write well-documented and well-tested code **for your work**; there will only be a bit on background (e.g., on how data is stored and implications for writing efficient code).

Specific topics

- Basic principles: breaking into small pieces, clear naming, consistent style; nothing is interactive more than once, tool everything;
- Version control: track history automatically; make small incremental changes;

- Automated testing: test modules, assert code expectations, bugs are new tests;
- Profiling: optimize what matters;
- Documentation: embed where possible, auto-generate, auto-test code examples;

Course links

- A general description of how to optimize programming by Wilson et al. (2014, PLoS Biol 12(1); effectively the course text);
- The zen of python;
- Astropy's developer documentation, in particular on how to contribute, but also including style, documentation, and testing;
- Numpy's developer documentation and work flow suggestions.
- Astropy's description of how to create and maintain a package using their template, including automatic testing and documentation building;
- Setting up Emacs for automatic checks or as a full-fledged IDE.

Prerequisites

- Familiarity with basic python, numpy, and astropy;
- A github account and a laptop set up for development (if this is the first time you do this, start with astropy and follow their instructions on getting the development version before the first class; you'll see that there is a choice for authentication in github; I recommend using SSH keys). If you encounter difficulties, write them down: you have just found a possible first contribution!

Evaluation

- A contribution to someone else's open-source packge (30%). I can help most with astropy and numpy, which each label issues that are suitable for new contributors (astropy, numpy), but other packages are fine too (e.g., Jo wrote me with possible galpy extensions; I have suitable issues in baseband; Hanno can surely tell you about REBOUND; or check out astropy affiliated packages).

- A piece of **your own** code fully tested and documented (70%). I strongly recommend using astropy's template, as it makes it very easy to automate continuous integration and generation of documentation.