Practical Computing for Scientists
Spring 2013

Instructors: Gabriel Ehrlich, Rex Garland, Julian Kates-Harbeck, Ian Tenney, Jason Chaves

Course Goals:

This course teaches essential computer skills for researchers in the natural sciences. The goal is to provide students with the essential and most powerful tools used in modern research environments. The course will be taught primarily using the UNIX operating system and the Python programming language, but with an eye toward the different computing environments used in research situations.

By the end of this course, you should be a self-sufficient programmer and software user. This means that you will be able to:

1. Navigate the UNIX operating system and use many of its powerful utilities, including shell scripting, version control, and distributed filesystems
2. Be confident using the Python program programming language, including advanced data structures, object-oriented programming, functional programming, and debugging tools.
3. Use scientific analysis and plotting libraries and integrate Python with the operating system and data workflow
4. Plot and present data in an effective and informative manner
5. Autonomously find, incorporate and learn to use the best external libraries for the task at hand
6. Write fast, efficient and optimized code for time-critical applications using the C programming language and the Cython extension to python

While examples will be drawn primarily from physics, these skills are highly useful in any scientific discipline involving quantitative analysis and are generally considered very desirable qualifications for research positions.

Reading List:

Readings will consist of occasional preparation for lecture topics. This will include

- Online Software Documentation
  - Python Standard Library (http://docs.python.org/library/)
  - Numpy, Scipy, Matplotlib (http://www.scipy.org/)
  - Software Carpentry: (http://software-carpentry.org/)
  - Cython documentation (http://cython.org/)
- Tutorials and Videos
  ● CS1U lectures (http://www.stanford.edu/class/cs1u/)
  ● Interactive vim tutorial (http://www.openvim.com/tutorial.html)

**Lecture/Lab policy:**

- Course meets Tuesday, Thursday 4:15-6:05 in 160-325 (Wallenberg Hall)

- 2 combination lecture/lab sessions per week, with mandatory attendance
  ○ Tuesday lectures introduce concepts, lab consists of small conceptual exercises
  ○ Thursday lectures focus on applications, longer lab with more involved problems

- Office hours TBA

**Assignments and Grading:**

Grades are based on adequate attendance of lectures and lab sessions as well as submissions of assigned lab work. The attendance of all sessions is required, unless there are special circumstances.

- Assignments are completed in the lab sections

- Code submission at the end of lab to receive credit

- In order to receive course credit, students must attend all 20 sessions (lecture + lab counts as one session). The permission of the instructor must be obtained for missing a session and missed lab work must be completed and submitted.

**Syllabus:**

**Week 1, Unix Introduction:**


  b. Advanced utilities, mercurial, shell scripts, piping

**Week 2, Python Introduction:**
a. Language Basics
   i) python interpreter
   ii) python scripts
   iii) control (if, while for)
   iv) functions
b. ADT’s (advanced data structures)
   i) lists, tuples
   ii) dictionaries
   iii) sets, stacks queues
   iv) arrays

Week 3, More ADT’s and debugging:

   a. More ADT’s
   b. Debugging
      i) pdb
      ii) unittest

Week 4, Scientific Python:

   a. numpy, matplotlib
   b. more matplotlib, scipy, sympy (if time)

Week 5, Advanced Python:

   a. Object Oriented programming
      i) classes, modules, scope, and namespaces
      ii) operator overloading and special functions
      iii) python data hierarchy: code as data!
   b. Functional Python
      i) list comprehensions, mappings
      ii) lambda functionals
      iii) passing functions as arguments

Week 6, Python Capstone Project:

   a. How to find, incorporate and learn new libraries
   b. Autonomous project
Week 7, Other languages:
NOTE: Week 7 and following weeks are tentative—subjects likely to change as per student interest.

a. Integrated packages
   i) Mathematica
   ii) Matlab
   iii) Comsol
   iv) ROOT

b. Other big languages
   i) Java
      - Idea of JIT’s and PyPy project
   ii) C++
   iii) C & Fortran

c. Deciphering unfamiliar code

c. Useful utilities
   i) Advanced UNIX: pipes and command-line utilities
   ii) Managing and converting data

Week 8, Speed:

a. Intro to optimization
   i) Runtime analysis
   ii) Loops
   iii) CSE
   iv) Vectorization

b. The C programming language (and why it’s fast)

Week 9, More Speed:

a. Cython

b. Profiling

Week 10, Advanced Topics (tentative):

- Parallel Programming
- Why parallel programming?
  - Types of problems that lend themselves to parallelism
  - GPUs vs. CPUs
- Threading
- MPI (mpi4py)
- OpenMP (pyopenmp)
- CUDA (pyCuda)

- Moving on from here
  i) Trends in scientific computing
  ii) Other important topics to learn about
  ii) Courses at Stanford