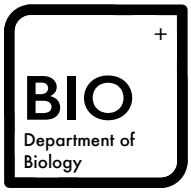

Bi 410/510: Introduction to Programming for Biologists

Winter 2015

Biologists who collect and analyze data often develop a complicated "workflow" that involves several different applications. But in order to get data from one program to another, they find themselves reading through large text files, copying and pasting from one document to another, checking each item to see if it can be used, and often reformatting the data for the next application. Boring! That's what computers are for!

This class is a hands-on introduction to computer programming, with an emphasis on developing the practical skills required to automate data analysis workflows. The course begins with basic concepts in Python programming: breaking complex tasks into manageable pieces, reading and writing data files, iteration and other control structures. We will see how to find code libraries and use software written by others, in particular modules written specifically for biology applications. If time permits, we will also learn about databases, statistical analysis using a language named R, and how to write "scripts" (programs that control other programs) so we can develop fully automated workflows.

This is a project-oriented course. Grades will be based solely on completed projects; there will be no exams.

Software. Plan on installing quite a bit of new software on your computer. Everything we do this term can be accomplished on a laptop or small desktop machine (as long as it's new enough and has up to date systems software). Details about which software to install will be posted on the class web site.

Create Your Own Project. We will provide data sets and predefined project descriptions, but if you would like to define your own projects send me e-mail.

Instructor:	John Conery 313 Pacific `conery@uoregon.edu`
Lectures:	MW 4:00 – 5:20, 123 LLC
Office Hours:	MTW 3:00 – 4:00 (tentative)
Textbook:	*Practical Computing for Biologists*, by Steven Haddock and Casey Dunn (`http://practicalcomputing.org`)
Prerequisites:	None (no prior computer programming experience is necessary)
Web Site:	`http://piazza.com/uoregon/winter2015/bi410510/home`

Learning Outcomes

I = introduce
D = develop
M = master

(1) Know how to use a terminal emulator and command line interface [D/M]

(2) Know how to implement simple functions in Python [D]

(3) Understand basic data structures in Python, how they are used [D]

(4) Learn general techniques for developing, testing, and debugging software [I/D]

(5) Be able to find, install, and use Python modules [I]

(6) Use BioPython for basic sequence analysis and other tasks [D]

(7) Know how to use an SQLite database to store project data [I/D]

(8) Know how to use R for analyzing and plotting data [I/D]

Students should know about the command line interface for several reasons: it provides useful background for writing Python programs that read and write files, it's required for writing scripts (programs that control other applications, *e.g.* as part of an analysis pipeline), and it's necessary for running jobs via queuing systems on high performance computers.

The "Python Literacy" goal for this course is learning about the fundamental data structures – strings, lists, and associative arrays (called "dictionaries" in Python) – and how to use them. Students will write a few simple programs that use these objects.

One of the most important messages to convey is that good programmers are lazy. Instead of writing a new program they check to see if it's already been done. Is there a module in the standard Python library, or has someone contributed a library to the user community through PyPI (Python Package Index)? Students will learn how to find and install packages, create data objects defined in the package, and write programs that use these objects.

In the same vein, we'll talk about searching Stack Overflow and other "social media for programmers" to find solutions for common problems.

To reinforce the idea of using modules students will work on projects that use BioPython (or maybe SciKit Bio, a newer alternative) and NumPy or SciPy.

SQL and R are both valuable for biology projects. They will be introduced in the middle of the term and used along with Python for projects in the second half.