1-2015

BIOB 491.02: Programming for Genomics

John P. McCutcheon
University of Montana - Missoula, john.mccutcheon@umontana.edu

Programming for Genomics, BIOB 491 02, 3 credits, CRN 32925
Tuesdays and Thursdays, 2:10pm-3:30pm, Health Sciences 114

Instructors: John McCutcheon, john.mccutcheon@umontana.edu
Tom Brekke, thomas.brekke@umconnect.umt.edu

Office Hours: By appointment, email John or Tom.

Textbooks: Required: *Practical Computing for Biologists*, Haddock & Dunn, 2011.
Optional: *Processing*, Reas & Fry, 2007.

Overview: Cheap, easily accessible DNA sequencing has transformed biology. For a couple of thousand dollars, individual research groups can generate more data in a week than the entire Human Genome Project generated in ten years. However, while the generation of these large data sets is routine, the analyses of these data are not. This course is aimed at teaching students the skills required to manage, analyze, and display large genomic data sets, but the skills you learn in this course will be useful in any aspect of biology (and in many aspects of life).

Most of the course will be taught in the UNIX environment using the programming language python. Students will write on average one new program a week, using problem areas and data sets from genomics as examples. We will cover the use of regular expressions (pattern matching), data types and structures, program control using logic and loops, reading and writing files, and python modules for biology. In the last few weeks of the course, we will explore large data visualization using Processing, a programming language originally designed for visual artists.

This course is intended for advanced undergraduates and graduate students. The prerequisites are either completion of BIOB 486, Genomics, or consent of instructor. Students already working with large data sets—whether genomic in nature or not—are encouraged to bring these data to the course. Because this course will be taken by students with varying levels of experience, needs, and expectations, I reserve the right to deviate substantially from this syllabus if I feel that certain topics need more or less time. If this makes you uncomfortable, I encourage you to drop the course sooner rather than later.

Expected Outcomes: We want you to learn the fundamentals of computer programming. Programming is a field all on its own, with rich traditions and deep theoretical underpinnings, and so we cannot hope to cover even a tiny fraction of it in a semester-long course for non-specialists. The goals of the course are for you to understand the basics of the python programming language, regular expressions, basic data types, program logic and control, reading and writing files, python modules, and rudimentary visualization of large genomic datasets using Processing. In general, you will learn this content by **doing**—that is, actually working out problems and writing your own code. We will lecture as little as possible, and as such this course will be more like a lab course than a didactic course.

Grading: 60% Approximately 8 programming exercises
20% Final programming project
10% Two midterm exams (5% each)
(The midterms will be on 24 Feb and 26 Mar)
10% A few homework assignments, mostly at the start of the course

Final grades will be based on your total points as a percentage of the total points possible. Pluses (+) and minuses (–) will be used (A, A–, B+, B, B–, C+, C, C–, D+, D, and D–) in the assignment of letter grades will be determined by the distribution of total scores, following these guidelines:

>90% of points (540): A- or better
>80% of points (480): B- or better
>70% of points (420): C- or better
>60% of points (360): D- or better

These cutoffs may be adjusted downward (in favor of the student).

Accommodations: to ensure accessibility of students with disabilities will be gladly made, but to qualify you must be registered with Disability Services for Students (DSS). Arrangements for accommodations on exams must be made through DSS.

Late work policy: This class will cover a lot of ground, and will require you to keep up with the assigned reading and assignments. If you have a problem understanding the material, or with turning an assignment in on time, we strongly encourage you to speak with us as early as possible. In general we won't accept late work, but we are sympathetic and reasonable if you deal with us in an upfront and honest manner and do not wait until the last minute to explain your situation.

Academic misconduct will be reported and handled as described in the University of Montana Student Conduct Code. All students must practice academic honesty. Academic misconduct is subject to an academic penalty by the course instructor and/or a disciplinary sanction by the University. All students need to be familiar with the Student Conduct Code: http://life.umt.edu/vpsa/student_conduct.php

The work you turn in should be your own. You free to discuss any aspect of the course with us or your classmates, including questions on the homework and programming assignments. But at the point when you begin formulating your answer on the computer, the work must become completely your own. Every individual develops a programming style, and it is surprisingly easy for us to see cheating when looking at code. As you will see from the very first assignment, there are usually several ways to get a solution when programming, and if we see solutions that are too similar we will ask the involved students about the incident; if no obvious explanation exists we will treat the matter extremely harshly. This may include receiving a failing grade for the entire course and filing a report with the Provost & Vice President for Academic Affairs. We don't expect this to be an issue with this course, but we do want you to know that

we take plagiarism very seriously. If you are unsure about any of this, we urge you to ask us before turning something in.

Dropping course or changing grading status will strictly follow the University policies and procedures, which are described in the catalog. Please note that dropping the course or changing the grading status (to CR/NCR) is not automatically approved after the 30th day of the semester. These may be requested by petition, but the petition must be accompanied by documentation of extenuating circumstances. Requests to drop the course or change the grading status simply to benefit a student's grade point average will not be approved.

Course outline (this *will* change, so I've only outlined the first few weeks):

Week 1. *Course intro and regular expressions.*
H&D Chapters 1-3.
27 Jan, Lecture 1, Plain text files and editors
29 Jan, Lecture 2, Regular Expressions
Homework 1, Regular expressions, due 5 Feb.

Week 2. *Intro to the UNIX shell and command line BLAST.*
H&D Chapter 4-5.
3 Feb, Lab 1, Regular expressions.
5 Feb, Lecture 3, The UNIX shell and BLAST
Homework 2, UNIX and BLAST, due 12 Feb.

Week 3. *Basic data types and structures.*
H&D Chapter 7.
10 Feb, Lab 2, The UNIX shell and BLAST
12 Feb, Lecture 3, Variable types and behaviors
Homework 3, manipulation of variables

Week 4. *Variables and exam review.*
H&D Chapter 7.
17 Feb, Lab 3, manipulation of variables
19 Feb, grep and exam review

Week 5. *Program control.*
H&D Chapter 8.
24 Feb, Exam 1
26 Feb, Lecture 4, Program control

Week 6. *Reading and writing files.* Week 7. *Modules and libraries.*
Week 8. *Programming efficiency.* Week 9. *Interacting with databases and web servers.*
Week 10. *Assessing the quality of a genome assembly.* Week 11. *Introduction to Processing.* Week 12. *Shape primitives.* Week 13. *Working with color.* Week 14. *Independent python/Processing coding project.*