02-201/02-601: Programming for Scientists

Carl Kingsford

1. Course Information

1.1 Course description

Provides a practical introduction to programming for students with little or no prior programming experience. Extensive programming assignments will illustrate programming concepts, languages, and tools. Programming assignments will be based on analytical tasks that might be faced by scientists and will typically include parsing, statistical analysis, simulation, and optimization. Principles of good software engineering will also be stressed. Most programming assignments will be done in the Go programming language, an industry-supported, modern programming language, the syntax of which will be covered in depth. Several assignments will be given in Python, Java, and C++ to highlight the commonalities and differences between languages. No prior programming experience is assumed, and no biology background is needed. Analytical skills and mathematical maturity are required.

1.2 Pre-requisites

- Some analytical skills and mathematical background.
- No biology or programming knowledge is assumed.

1.3 Course Details

Course Format. There will be two lectures each week. We will also have 3 "optional recitation / help sessions" per week.

Submitting Assignments. The programming assignments (homeworks and projects) will be submitted via Autolab, and will be graded in conjunction with the policy outlined below.

Discussion Forum. An online forum is provided on BlackBoard as an area for discussion and questions. The forum will be moderated by the course staff who will respond to questions, but students are encouraged to help each other via discussion. However, specifics of assignment or project solutions should not be discussed — any hints will be provided by the teaching staff.

Programming Expectations. You are expected to produce clean, readable, and well-documented code. The coding practices should be consistent with what is taught in the lectures (e.g. consistent naming conventions, descriptive variable names, and short functions). A component of each assignment's grade will be based on coding style.

Programming Languages. Most of the course will be taught in the Go programming language. This is a relatively new language that has industry backing and that is simpler than many languages, but not too simple (it still contains important concepts like types, pointers, and type interfaces). However, unlike most introductory programming courses, we will not focus on a single language. Think of it as "comparative computational linguistics" — we may have assignments in C, C++, Python, Java, and perhaps a couple of other languages. The best way to understand programming concepts is to see how they are realized in a few different languages.

2. Tentative Topics

We hope to cover the following topics. We won't necessarily cover them in this order, however.

- **The machine and how programming languages abstract it.** The connection between programming language constructs and the underlying machine will be discussed. The syntax for the programming language "Go" will be covered in depth, and comparison between common programming language syntaxes will be given.

    1. Imperative programming constructs: functions, if-statements, loops (for, while), switch-statements, expressions,
    2. Basic data structuring constructs: variables, arrays, strings, structs, types, and pointers
    3. Reading and writing files
    4. How to build Go large programs
    5. Basic execution and memory model (Von Neumann architecture)
    6. Execution paradigms: how the computer turns your program into something it can run (interpretation, native compilation, bytecode compilation)

- **The tools of programming:** Throughout the course, we will cover important tools for good software engineering practice.

    7. Assertions, preconditions, postconditions
    8. Code documentation
    9. Version control (likely SVN and git)
    10. Unit tests — testing small sections of code
    11. Debugging — strategies, debuggers, common errors
    12. Profiling — figuring out what's taking so long
    13. Make — automating compilation

- **Basic data structures and algorithm design techniques:** Sophisticated data structures and algorithms will be introduced, along with more difficult programming assignments.

    14. Linear data structures: arrays, lists, stacks, queues; binary search
    15. Dictionary data structures: binary search trees including tree traversals (DFS, BFS, pre-, in-, post-order); hash tables.
    16. Heaps, heapsort
    17. Graphs; MST
    18. Divide and conquer, recursion
    19. Dynamic programming

- **Abstraction and modularization:** How we control complexity through well-defined interfaces.

    20. Bigger units of code: Modules, namespaces, packages
    21. Type interfaces and user-defined types
    22. Object-oriented programming

23. Design patterns

- **Advanced programming constructs:** Some advanced abstraction mechanisms will be covered to help code re-use and clarity.

    24. Anonymous functions, functional programming
    25. Higher-order functions, function objects
    26. Generic programming, Generics in Java and C++, STL

- **Parallelism:** Using Go's parallel features to make use of multiple cores on a machine.

    27. Work queues
    28. Parallel loops
    29. Locks
    30. Futures & promises
    31. Memory safety, data parallelism, instruction parallelism, resource contention, deadlock

3. Coursework

Coursework will consist of extensive programming plus a midterm and final exam:

Programming Projects. (70% of your grade) Approximately 8 – 12 assignments designed to give you familiarity with a particular language feature, programming idea, common programming task, or algorithm. You will typically be given about 1 week to complete these assignments. These assignments will generally be independent of each other.

Midterm and final. (30% of your grade) The midterm and final exams will tests your knowledge of the material from the class, and your ability to read and design programs. The midterm will be held in class and the final held during the university's scheduled time.

Programming assignments must be completed on your own (unless noted) and turned in to the autograder by a given deadline. Late assignments will be accepted for 24 hours past the deadline at a penalty of 25%. No assignments will be accepted more than 24 hours late. All programming assignments will be graded by the autograder, which will check that your programs are outputting the expected results. In addition, 25% of your grade for every programming assignment will be based on following appropriate programming style.

4. Collaboration Policy

You may discuss programming assignments with classmates. **However, you must not share or show or see the code of your classmates.** You must write your own code entirely. You can post general coding questions (with code snippets) on the discussion board.

You must write all programming assignments on your own and cannot share code with other students or use code obtained from other students. In addition to manual inspection, we use an automatic system for detecting programming assignments that are significantly similar.

You may *never* use, look at, study, or copy any answers from previous semesters of this course.

5. Other policies

Classroom etiquette: To minimize disruptions and in consideration of your classmates, I ask that you please arrive on time and do not leave early. If you must do either, please do so quietly. Laptop use is *encouraged* so long as you are trying examples or following along with the lecture slides. Use of laptops for other purposes is *strongly* discouraged.

Excused absences: Students claiming an excused absence for an in-class exam or midterm must supply documentation (such as a doctor's note) justifying the absence. Absences for religious observances must be submitted by email to the instructor during the first two weeks of the semester.

Academic honesty: All class work should be done independently unless explicitly indicated on the assignment handout. You may *discuss* homework problems with classmates, but must write your solution by yourself. If you do discuss assignments with other classmates, you must supply their names at the top of your homework / source code. No excuses will be accepted for copying others' work (from the current or past semesters), and violations will be dealt with harshly. (Getting a bad grade is much preferable to cheating.)

The university's policy on academic integrity can be found here: `http://www.cmu.edu/policies/documents/AcademicIntegrity.htm`. In part it reads "Unauthorized assistance refers to the use of sources of support that have not been specifically authorized in this policy statement or by the course instructor(s) in the completion of academic work to be graded. Such sources of support may include but are not limited to advice or help provided by another individual, published or unpublished written sources, and electronic sources." You should be familiar with the policy in its entirety.

In particular: use of a previous semester's answer keys or online solution manuals for graded work is absolutely forbidden. Any use of such material will be dealt with as an academic integrity violation.