02-601: Programming for Scientists

Phillip Compeau
Fall 2017

1. Course Information

1.1 Contact information

Instructor: Prof. Phillip Compeau
E-mail: pcompeau@cs.cmu.edu
Website: http://compeau.cbd.cmu.edu
Office: Gates-Hillman Center (GHC) 7403
Office Hours: Wednesdays 10:30 - 11:30 AM (or by appointment)

TA: Amir Alavi
E-mail: aalavi@andrew.cmu.edu
Office Hours: Tuesdays 12:00 - 1:00 PM and Fridays 11:30 - 12:30 PM (see Canvas for locations)

1.2 Class location

Lecture: Tuesdays and Thursdays 3:30 - 4:20 PM in Wean Hall (WEH) 5409
Recitation: Fridays 10:30 - 11:20 AM in Gates-Hillman (GHC) 4102

1.3 Course description

This course provides a practical introduction to programming for students with little or no previous 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. After an introduction to computational thinking (as it pertains to science), 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. Other assignments may be given in other languages (Java, Python, etc.) to highlight the commonalities and differences between languages.

1.4 Course philosophy

Our first goal for this course is to make you comfortable writing your own programs and have a better understanding of how computing works, at multiple levels.

Our second goal is to convince you how much fun programming is! Writing a program is like solving a sudoku puzzle — programming tests (and builds!) your powers of concentration and logical thinking. But programming is more rewarding than sudoku because it equips you with a transferable skill instead of the ability to fill in a square of numbers.

Our third goal is to help you understand some fundamental scientific (in particular, biological) algorithms on a high-level.

Finally, we want you to gain independence to hack your own scientific problem by planning and executing a longer programming assignment of your own choice, so that you will be capable of programming independently in the future.

This can be a challenging course for some because "thinking like a programmer" might be new to you. Rest assured: the difficulty of the course and the material we cover is well within that of a typical introductory programming course.

1.5 Pre-requisites

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

1.6 Course Details

Course Homepage The homepage for the course will be hosted on Canvas, where I will post assignments, lecture notes, and general information. You should automatically be enrolled in the course, but the course homepage can be found at `https://canvas.cmu.edu/courses/625`.

Submitting Assignments. The programming assignments will be submitted via Autolab (see Canvas/Piazza for link), and will be graded in conjunction with the policy outlined below.

Discussion Forum. An online forum is provided on Piazza 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, assignment specifics should not be discussed — any hints will be provided by the teaching staff. Piazza is integrated into Canvas, and you can find the class at `piazza.com/cmu/fall2017/02601`.

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 only look at a single language. Think of it as "comparative computational linguistics" — we may have a few assignments in a couple of other languages. The best way to understand programming concepts is to think about them in terms of pseudocode as well as see how they are realized in more than one language.

2. Tentative Course 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 large programs using top-down design
5. Basic execution and memory model (Von Neumann architecture)

- **Basic data structures and algorithm design techniques:** Several data structures and algorithms will be introduced.

6. Linear data structures: arrays, lists, stacks, queues; binary search
7. Dictionary data structures: binary search trees including tree traversals (DFS, BFS, pre-, in-, post-order); hash tables.
8. Divide and conquer, recursion

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

9. Assertions, preconditions, postconditions
10. Code documentation
11. Unit tests — testing small sections of code
12. Debugging — strategies, common errors
13. Profiling — figuring out what's taking so long

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

14. Bigger units of code: Modules, namespaces, packages
15. Type interfaces and user-defined types
16. Object-oriented programming
17. Design patterns

- **Parallelism:** Using Go's parallel features to let your program do more than 1 thing at a time.

18. Goroutines
19. Channels

- **Computability and the limits of computers.** Are there problems computers fundamentally *cannot* solve?

20. Turing machines
21. Halting problem and uncomputable functions
22. NP and NP-complete problems

- **Fundamental scientific algorithms.** We will also encounter a handful of fundamental scientific algorithms, especially those taken from biology. Examples are below:

23. Graph-based algorithms for genome assembly
24. Evolutionary tree construction with UPGMA
25. Applying dynamic programming to compare biological sequences
26. Local search for protein folding
27. Simulating large galaxy models with the Barnes-Hut algorithm
28. Applying game theory to understand the dynamics of cooperation

3. Coursework

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

Programming assignments. (30% of your grade) Approximately ten assignments designed to give you familiarity with a particular language feature, programming idea, common programming task, or algorithm.

Examinations. (40% of your grade) The midterms and final exam will tests your knowledge of the material from the class, and your ability to read and design programs. The midterms will be held in class and the final held during the university's scheduled time. The midterm dates are:

- Midterm 1 (10% of your grade): Friday, September 29 (in recitation)
- Midterm 2 (10% of your grade): Friday, November 3 (in recitation)
- Final (20% of your grade): Time and location TBD (will be posted when set by university)

The midterms will not be cumulative: midterm 2 will cover material encountered after midterm 1. That having been said, later material in the class builds upon the earlier material.

The final will cover all the material from the class.

Project. (15% of your grade) The homework assignments in this course are designed to help reinforce the introductory concepts of programming. However, because they are weekly assignments, they are unable to delve very deeply into a single idea. The purpose of this project is for you to "take the next steps" as a scientific programmer and have the opportunity to design a larger project on your own. An ancillary purpose is for you to become a better scientific writer; to be able to clearly communicate the key aspects of a scientific issue and explain the technical ideas used to address that issue. After all, a large part of being a successful scientist is about clearly and effectively communicating ideas.

As a result, you will write a Go program of your choosing and apply this program to a practical dataset, then write an explanation of the key technical ideas presented in that paper.

Finally, you will describe your project in a short in-class presentation during the final week of class.

We will have in-class presentations at the end of the course. I will send around a separate assignment describing expectations for the project.

Pre-class exercises. (7.5% of your grade) Learning to program is a lot like learning to speak a foreign language; it is far better to practice a little each day than to practice a lot once a week. Accordingly, you will perform much better in the course if you don't wait until the last minute to complete a week's worth of homework. Sleep is often the best way to debug! To encourage you to space out your work, I will frequently assign short exercises due the evening before class. Each assignment will receive a binary grade of "pass" or "incomplete".

You are allowed three dropped pre-class exercises without penalty. These can be used for any purpose.

Attendance and participation (7.5% of your grade) Attendance will be taken, and we will have occasional in-class exercises that serve to reinforce the concepts we have covered. These

exercises will not be graded, but participation will be expected in order to receive a complete grade for that day.

You are allowed three "dropped" attendance grades without penalty. These can be used for any purpose.

Programming assignments must be completed on your own (unless noted) and turned in to the autograder by a given deadline. **No late assignments will be accepted.** All programming assignments will be graded by the autograder, which will check that your programs are outputting the expected results. In addition, a fraction of your grade for every programming assignment will be based on following appropriate programming style. The autograder will give you an estimated grade immediately, excluding the style portion of your grade. The TAs will subsequently go over your code and assign the final grade, taking into account coding 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 allowed only for following along with in-class examples. The use of phones or other electronic devices during class is forbidden and will result in a zero discussion grade for the day.**

Excused absences: Students claiming an excused absence for an in-class exam 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 and programming assignments 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 at the following link: `http://www.cmu.edu/academic-integrity/`. 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.

6. Accommodations for Students with Disabilities

If you have a disability and have an accommodations letter from the Disability Resources office, I encourage you to discuss your accommodations and needs with me as early in the semester as possible. I will work with you to ensure that accommodations are provided as appropriate. If you suspect that you may have a disability and would benefit from accommodations but are not yet registered with the Office of Disability Resources, I encourage you to contact them at `access@andrew.cmu.edu`.

7. Provost's Statement on Student Well-Being

Take care of yourself. Do your best to maintain a healthy lifestyle this semester by eating well, exercising, avoiding drugs and alcohol, getting enough sleep and taking some time to relax. This will help you achieve your goals and cope with stress.

All of us benefit from support during times of struggle. You are not alone. There are many helpful resources available on campus and an important part of the college experience is learning how to ask for help. Asking for support sooner rather than later is often helpful.

If you or anyone you know experiences any academic stress, difficult life events, or feelings like anxiety or depression, we strongly encourage you to seek support. Counseling and Psychological Services (CaPS) is here to help: call 412-268-2922 and visit their website at `http://www.cmu.edu/counseling/`. Consider reaching out to a friend, faculty or family member you trust for help getting connected to the support that can help.

If you or someone you know is feeling suicidal or in danger of self-harm, call someone immediately, day or night:

CaPS: 412-268-2922

Re:solve Crisis Network: 888-796-8226

If the situation is life threatening, call the police:

On campus: CMU Police: 412-268-2323

Off campus: 911

If you have questions about this or your coursework, please let me know.