Math 121: Introduction to Computing

Fall 2014

Basic information

Professor: Adam Groce, agroce@reed.edu
Class schedule: Class meets Monday, Wednesday, and Friday. Lab meets Thursday. Make sure you know which section of each you are in and its time and location.
Office hours: I hold office hours (in Library 390) Monday 9-11, Wednesday 3-4, Thursday 10-11, and Friday 10-11. I can also meet with you by appointment if those times are bad or if you would like to discuss something privately. You are also welcome to stop by other times, but I might be busy.
Website: The course website is http://people.reed.edu/~agroce/math121/. Homework, projects, the schedule for required reading, and all handouts will be posted on that website.
Textbook: The textbook is *Composing Programs* by John DeNero. The book is available online at www.composingprograms.com.

Course overview

The goal of this course is to introduce you to the principles of computation. That includes teaching you to write a program, as well as how a computer interprets and runs a that program. Writing a program well can be a difficult skill to master. You must understand the tools available to you, and you must understand the principles of good design that allow those tools to be used in manageable, understandable ways.

Our vehicle for building these skills will be programming in Python, which is a good introductory language for a variety of reasons. First, it uses a reasonably simple syntax, reducing the need to get bogged down in messy details. Second, it is very flexible, allowing me to show you several different ways of writing programs. Finally, it's a widely used language, meaning that knowing it will be potentially quite useful to you after the course.

Given how much time we will spend programming in Python, you could be excused for thinking this was simply a course in how to write Python programs. It is not. There are some areas of programming (like graphics) that are very useful for practical applications, but which do not require any new conceptual understanding. We will largely be ignoring those areas, though the course should make it easy for you to learn those things on your own with some quick internet searches. We will also be covering some topics which do little to aid in the writing of programs, but which are important to the understanding of computation as a whole.

Coursework and grading

Reading: There will frequently be assigned reading, generally from the course textbook. You should make sure to have this complete before the day it's due, because I will assume you have read it. This course is cumulative, meaning that later topics depend on the knowledge of earlier topics. You really don't want to get behind.

Homework: There will be a homework assignment roughly each week. A homework will usually consist of a series of short (though still challenging) exercises. Don't be afraid to come to office hours if you're having trouble (but make sure you at least try things for yourself some first). You will also usually have the entire lab period on Thursdays to work on your homework, with me and a teaching assistant available to help with problems. (In fact, I ask that you not work on anything else during those lab hours unless your current homework is already finished.) Homework will account for roughly a third of your grade.

Projects: During the course there will be three larger projects. These will be similar to the homework except that they will involve larger blocks of code and give you some experience with more involved programming tasks. They will also give you a little room to be creative and decide for yourself exactly how you want your program to behave. These projects can take considerable time, and you will be working on them at the same time you are continuing to do the weekly homeworks, so it is important that you not put them off too long. Projects contribute another third of your grade.

Tests: There will be two tests in total during the course, a midterm during the regular semester and a final. These will be written tests, and they will cover everything being taught in the course. They will be worth roughly a third of your grade.

Other policies

Submitting work: Work, both homeworks and projects, should be submitted in two ways. First, you should upload an electronic version. (The first homework will explain how to do this.) Second, you should print out what you've written and turn it in in class. The graders and I will use the written version to write comments, point to errors, etc., and the electronic version to run and test.

Clarity in programming: When you write programs for homework or projects, you are writing code that needs to be understood by others (me and the graders). That means writing code in understandable ways, and documenting that code well. We will talk in class about how to do this, and your grade will depend on how well it has been done. It is not enough for the program to do the correct thing.

Attendance: I trust you to make decisions regarding attendance for yourself. I think you should attend every class because I think that is important to learning the course material, but it is that learning of the material on which you will be judged, not the attendance directly. I will, however, assume everyone is in class, and if you miss class you should make sure to talk to someone else in the class to find out if you missed any announcements, schedule changes, etc. If you miss a test you will receive no credit unless your absence is excused. Some excuses (such as illness) will require documentation (such as a doctor's note). I expect that if you will be missing class for an excusable but predictable reason (say, a religious holiday) that you inform me before the absence. I will not excuse absences after the fact for reasons that were known about ahead of time.

Academic integrity: Unless otherwise stated, all work you turn in in this class should be yours and yours alone. I take this very seriously and will not hesitate to report violations of this principle. You are allowed to discuss homework with each other and even help a classmate find a bug in their code. You are not allowed to share code or work on each other's code. If you have any question about whether some level of cooperation is acceptable, ask me.

Advice

Don't procrastinate! It is very hard to predict how much time a given program will take to write, even a very simple one. Sometimes what seems easy will turn out to be hard, and what seems hard will turn out to be easy. Often a program feels 99% done, with only a couple little things remaining, and that final 1% ends up taking as long as the first 99% took. You might find that you don't understand something you thought you did and need to ask questions in lab or office hours. Leave extra time. Most of the time you won't need it, but sometimes you will.

Don't get frustrated! Programming and computer science in general require thinking in ways that will be new to many of you. That's part of why this is such a valuable class to take and why it can be so much fun, but it can also make it very difficult, especially at first. Sometimes things will click easily and sometimes they won't. Don't expect everything to work out perfectly the first time. It is entirely normal to run into obstacles along the way.

Have fun! Learning to work with computers can be a fantastic experience. It changes mysterious objects you just take for granted into something you can understand. Whatever you want to do outside computer science, whether it's biology or art or economics, programming can help make it happen. And most importantly, it's just extremely interesting and intellectually rewarding. Don't get so lost in the details the work that you don't enjoy the experience.