Introduction to programming and computation CS 1: Winter 2015

About the course

CS 1 is the header course for the Computer Science major and minor, and it is one option as the header course for the Digital Arts and Computational Methods minors. CS 1 fulfills the TLA distributive requirement, so you get both your Technology or Applied Science (TAS) credit and your lab credit. Pretty sweet!

You might think that this course is about computer programming. True, you will do a lot of programming. But that is not what this course is about.

Some day you will be a leader, a decision maker. As computer technology becomes more prevalent and pervasive in society, decision makers increasingly need to understand not just how to use the technology, but also the principles behind it. In this course, you will begin to learn about the principles behind computer technology. You'll start to understand why computers do the things they do.

More broadly, in CS 1 you'll learn to think about problems the way a computer scientist thinks. This skill is valuable in any field—yes, even in the humanities.

Professor

Tom Cormen
Office: 204 Sudikoff
thc@cs.dartmouth.edu
646-2417

Office hours

I have 7 office hours per week, in 204 Sudikoff, January 5 through March 12:

- Tuesday, 2:00–4:00 pm (3:00–5:00 pm on January 6)
- Thursday, 10:30 am–12:00 noon
- Thursday, 4:00–6:00 pm (except January 15)
- Friday, 10:30 am–12:00 noon

My office hours are *always* first come, first served; I do not schedule individual appointments during office hours. If you'd like to meet with me outside of office hours, please contact me and we'll set up a time.

Additionally, if my office door is open and I'm not obviously engaged in something, feel free to knock and drop in. If I'm busy, I'll let you know nicely. I try to be highly available to students.

Course staff

Teaching assistant

Keith Carlson
Office: 108 Sudikoff
Email: Keith.E.Carlson.GR@Dartmouth.edu
Office hours:

- Tuesday, 1:00–2:30 pm
- Thursday, 2:00–3:00 pm

Phone: 646-3297

Undergraduate section leaders

Mazen Ammar '16

David Aspinall '15

Nicole Boyd '15

Lizzie Brissie '17

James Burton '17

Derek DeWitt '15

Lota Ezenwa '17

Nick Fiacco '17

Max Gibson '16

Dani Gnibus '17

Emily Greene '17

Nicole Hedley '15

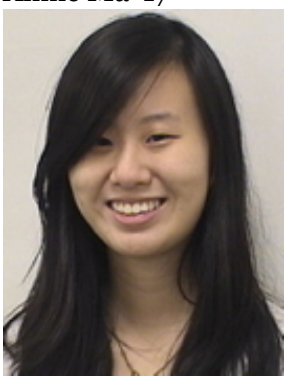



Emily Holt '16

Young Kang '17

Justine Lee '16

Annie Ma '17



You

In order for you to take this course, you **must** fulfill the following three ironclad prerequisites:

- You have to know how to read email.
- You have to know how to send email.
- You have to know how to read web pages.

If you have these prerequisites, then you are prepared to take CS 1. **Notice that we do not assume that you have prior programming experience.** Indeed, historically, most students in CS 1 (and its predecessor, CS 5) came in with no prior programming experience.

In addition, we'll be using a small amount of math. Do you know what it means to square a number? (Yes, of course you recall that it means to multiply the number by itself.) Do you know what a logarithm is? Don't worry about it if you don't remember logarithms, or even if they make you uncomfortable. The way we use them in this course, you'll understand them.

Questionnaire

I need to know a little about you. Please fill out this questionnaire and send it to me as soon as possible.

Lectures

- **Location**: LSC 100 (Oopik Auditorium)
- **Times**: MWF 1:45–2:50 pm. x-hour is Thursday, 1:00-1:50 pm.

x-hours

We will use two other x-hours for review sessions before exams, on January 29 and February 19. And we'll use some other x-hours for lectures. So keep your x-hours available for this course!

Questions

I like it when you ask questions in lecture. I really do. I understand that it can be intimidating to raise your hand for a question in a large lecture, but please do not hesitate to ask questions. Don't worry about what anyone thinks of you for asking questions. I'll think more of you if you understand the material, and if you need to ask questions, that's fine. And don't worry about what the other students might think. Chances are that if you have a question, then at least one other student—and possibly many more—has the same question. You're doing the other students a favor by asking!

If you still are reluctant to ask a question in lecture, even after all the encouragement in the last paragraph, feel free to write it down and ask me after class, by email, or in office hours, or ask a member of the course staff in lab hours. We want to help you!!

Occasionally, I might elect to answer a question later or after lecture. That will usually be because the answer to the question is tangential to the material we're covering, or simply because I am pressed for time. But please feel free to ask; I'll let you know nicely if I cannot answer the question then and there.

Laptops and phones

If you bring a mobile phone, please make sure that it does not audibly ring during lecture. If it does, *I* will answer it, which will embarrass you greatly and suprise your caller. If your phone is a smartphone, you may not use it for any purpose during class.

Although you are not required to bring your computer to class, I realize that you might find it useful to read the programs or lecture notes on your laptop during lecture. I'm OK with you doing so. But I am not OK with you using your laptop for email, Facebook, Twitter, etc. during lecture. If you are that bored during class, please leave the room so that your online activities do not distract those around you. Really. If you're going to screw around on your laptop, please leave.

Recitation section

You are required to attend a one-hour recitation section meeting each week. Attendance is mandatory; section leaders will actually take attendance!

We will do some exercises in the sections, and you will find that these exercises help you write code for both the programming assignments and the exams.

Download this form, and follow the instructions. You must fill it out and email it to Professor Cormen by 1:00 pm on Thursday, January 8. We expect to determine the recitation section schedule by Thursday evening, and your section assignment will be on the Recitation Sections page by then. This page will be accessible only through the Canvas site. You are responsible for checking this page and attending your recitation section starting on January 12, 13, or 14.

If you say you are available during a one-hour slot, I expect you to honor this commitment. Please take this expectation into account as you make other commitments on and off campus this term.

We recognize that occasionally (which means "not on a regular basis"), you might not be able to attend the recitation section to which you have been assigned. In that event, please find another section to attend that week. Make sure that your regular section leader knows that you'll be attending another section, and make sure you get permission *in advance* from the section leader whose section you'll be visiting.

Grading

- Short assignments and recitation sections: 15%
- Lab assignments (5): 45%
- Exams: 40%

There will be two midterm exams, on Thursday, January 29 and Thursday, February 19, 6:00 pm to 9:00 pm. If a scheduling conflict prevents you from taking one or both of the midterm exams at the scheduled time, you must let me know by 5:00 pm on the preceding Monday (January 26 and February 16). *No exceptions*.

The final exam is scheduled by the Registrar and will be offered at one time, and one time only: Friday, March 13, 11:30 am to 2:30 pm. If you have plans that prevent you from taking the final exam on March 13 at 11:30 am, you have three choices: change your plans, do not take this course, or get a 0 on the final exam. *The only way that you may take the final exam at another time is if you have a documented illness that prevented you from taking the exam at the scheduled time*.

Before each exam, I will post a web page to help you review. There will also be review sessions during the x-hours the days of the first two exams. We will also hold a review session before the final exam, time and location to be determined.

The three exams count toward 40% of your final grade, but they will be weighted unevenly, to your advantage. The two that you do best on will count 15% each, and the one on which you do the worst will count only 10%.

There is no fixed scale for assigning letter grades. I expect the median grade for the course to be B+ or B, *but I make no guarantees in advance*. The median grades depend on *you*. If your grade is borderline, I may take your attendance at recitation sections into account when deciding which way to go.

Short assignments

Short assignments are relatively brief exercises that you will turn in at the next Monday, Wednesday, or Friday class. In some cases, you might have two classes to work on a short assignment. Short assignments will usually consist of one or two short programs to help you understand the concept being covered.

You will submit short assignments via Canvas. They will have a timestamp, and so we will know when you submitted them. They are due at the start of class on the day they are due. The start of class is 1:45 pm *sharp*. If your short assignment comes in at 1:46 pm, it is late. **Short assignments handed in late will receive no credit.**

Your section leader will grade your short assignments on a scale of 0, 1, or 2.

- 2 means that you essentially got it right, though your section leader might suggest some ways that you could improve your work.
- 1 means that you got it partially right, but not enough to say that you got it "essentially" right.
- 0 means that either you did not submit the short assignment, or that what you submitted was a very poor effort.

You may resubmit up to five short assignments on which you get a 1. If you resubmit a short assignment, you may resubmit it one time, and you have to resubmit by the start of the next Monday/Wednesday/Friday class after you got back your original assignment. When you resubmit a short assignment, we ask that you do a couple of things as a courtesy to your section leader. First, please make it clear that it is a resubmission. Second, please include

your original submission along with the resubmission, so that your section leader can see your progress.

Lab assignments

In-depth lab assignments will be due on certain Mondays throughout the course; you will have one week to work on each lab. You are not required to complete the lab assignment in the lab. You may use your own computer wherever you like, just as you do for short assignments.

Lab assignments require a significant time commitment. Start early, and if you need to, get advice from me or the course staff.

Like short assignments, you will submit lab assignments via Canvas. They are due at the start of class, 1:45 pm, on the day they are due. If your lab assignment comes in at 1:46 pm, it is late. Each lab assignment will also have a "checkpoint," which is a piece of the lab assignment that we separate out and is due at 1:45 pm two days after the lab assignment officially goes out. We do not accept checkpoints late.

Each lab assignment is worth 40 points. Of the 40 points, 5 points are for the checkpoint. If you hand in a lab assignment late but by 5:00 pm the day it's due, we will deduct 4 points (10%). If you hand it in between 5:00 pm the day it's due and 5:00 pm the next day, we will deduct 16 points (40%). We will not accept it after 5:00 pm the next day, and you will get 0.

There are two ways that you can get an exemption from this lateness policy. First, if something happens beyond your control that prevents you from doing your assignment on time, then we can excuse a late assignment. Second, you may be able to arrange **in advance** to submit your assignment late. You will need a Darned Good Reason to do so. Note that you need to ask **in advance**. The day the assignment is due does not count as "in advance." **Only Professor Cormen is authorized to excuse late assignments.** Nobody else, including any other member of the course staff, is authorized to excuse late assignments.

Yes, this lateness policy is harsh. Why? Because in the past, those who have fallen behind have had a devil of a time catching up. So we are trying to prevent you from falling behind. Yes, 40% off for a lab assignment handed in at 5:01 pm sucks. In the past, students have complained that they could have handed in something substandard on time and gotten more points than if they had handed in something really good a little late. Unfortunately, the Real World works this way as well. Imagine showing the World's Best Software...a week after the trade show. **It is up to you to plan your time carefully and get your work in on time!**

I am, however, not a complete ogre. We will give you a discount on *one* unexcused late lab assignment submission. There are three possibilities:

- If you submit all your lab assignments on time, you don't need a discount.
- If you submit some lab assignments late, but whenever you submit a lab assignment late it's in by 5:00 pm the day it's due, then we will waive the 10% penalty for one of them.
- If you submit any lab assignments between 5:01 pm on the due date and 5:00 pm the next day, we will reduce the late penalty from 40% to 20% (8 points) for one of them. That will be the only discount you get; you won't get any discount for lab assignments submitted late but by 5:00 pm on the due date.

Extra credit

Some of the assignments may suggest extra credit work. Extra credit in this course will be tallied separately from regular scores. If you end up on a borderline between two grades at

the end of the course (or are being considered for a citation), extra credit will count in your favor. Failure to do extra credit will never be counted against you, however. You should do extra credit work if you find it interesting and think that it might teach you something. It never pays to skimp on the regular assignments in order to do extra credit problems.

One corollary of the way in which we count extra credit is that if you get a 30 on a lab assignment and also 5 points of extra credit, that is *not* the same as getting a 35 with no extra credit. The latter—35 points with no extra credit—is far better.

How to get help

There are many ways for you to get help. You can visit me during office hours, you can go to lab help hours, or you can make an appointment with me, Keith, or your section leader. You should also feel free to ask for help by email. You can email a specific person, including me. For a faster response, you can get an answer from the first member of the course staff (again including me) to read your message by emailing to

cs1help@cs.dartmouth.edu

This account is a mailing list consisting of me, Keith, and all section leaders. (Note that you have to put cs. after the @ in the address.) We will try to respond as soon as possible.

If you visit me or any of the course staff for help with a program that you are writing, make sure that you can get to an electronic copy of your program. That way, we can try to compile and run it. It's nice to have a printed version of your program, but we cannot compile and run a printed version.

When you email a program to one of us, you just need to add your program as an enclosure to your email, along with additional information as to what problems you have observed. When you email a question, please be as specific as possible, and tell us what you've done to try to figure out a solution for yourself. We reserve the right to not answer emails that say nothing more than "My program doesn't work; where is the problem?" Make sure you send your program as an enclosure; *do not copy and paste your program into the message*.

The Tutor Clearinghouse is another source of help. They will have private, one-on-one tutors available for this class. The tutors are recruited on the basis that they have done well in the subject, and they are trained by the Academic Skills Center. If you are on financial aid, the College will pay for three hours a week of tutoring. To get a tutor, go to 301 Collis and fill out an application.

Lab help hours

Several days per week, there will be course staff in LSC 200 to lend a hand with assignments or anything else you would like help with.

- Tuesdays, 7:00 pm to 10:00 pm
- Wednesdays, 6:00 to 9:00 pm
- Thursdays, 7:00 pm to 10:00 pm
- Sundays, 3:00 pm to 6:00 pm
- Sundays, 7:00 pm to 10:00 pm

There is one change that we know about in advance: on January 28, lab help hours will be in LSC 201 instead of LSC 200.

The LSC rooms do not have their own computers, so make sure to bring your own laptop with you. If you need to use one of the Computer Science department's Macs, you can use the ones in 002 Sudikoff.

You might find it useful to do some of the assignments during the lab help hours. Many of the problems that will stump you and waste your time if you are working alone can be cleared up in moments by the course staff. Then again, you might not find it useful to work in the lab, depending on how many of your fellow students are competing with you for the attention of the course staff.

Please prepare before going to lab; course staff can be much more helpful if you've already made a solid effort at solving the problem you are working on.

LSC 200 and 002 Sudikoff are locked at all times, but if you're registered for CS 1, you should be able to get in with your Dartmouth ID. The entrances to LSC and Sudikoff are locked after 6:00 pm, but if you're registered for CS 1, you should be able to get into the building with your Dartmouth ID. The second floor of Sudikoff also requires card access after 6:00 pm, and again, your Dartmouth ID should get you in, if you're registered for CS 1.

Note that there are several restrictions that come with access to LSC and Sudikoff. Among them are that you are not to lend your Dartmouth ID to anyone else in order to allow them into the building or labs. Also, *you must not bring food into LSC 200 or 002 Sudikoff*.

A note about lab help hours

In the past, some CS 1 students have used lab hours to ask the course staff to show them how to solve an assignment step by step. That is not what lab help hours are for. The job of the course staff is *not* to write your program for you. The course staff will help you with conceptual questions or if you're stuck on a problem. They can often help you find errors, though in the past some CS 1 students have had some extremely subtle errors that eluded the section leaders.

But if you think that you can show up to lab help hours unprepared and repeatedly have the course staff show you how to write the next few lines of code, think again. Moreover, if you need that much hand-holding to get through the programming assignments, you will struggle on the exams.

Textbook

The textbook is *Think Python: How to Think Like a Computer Scientist* by Allen B. Downey. You'll like this part: the textbook is not required, and it's free! You can find it at this site, where you can either view it on the web or download it as a PDF.

This textbook is purely for your reference. I won't assign reading from it. You'll find that the online lecture notes suffice for almost all purposes.

Software

We will be using PyCharm, a free Integrated Development Environment (IDE) that runs on both Windows and Mac OS X. You will install this software on your own computer as part of the first short assignment.

Unfortunately for Windows users (and, yes, we know that many Dartmouth students have opted for Windows), the Mac will be the "primary" machine supported. 002 Sudikoff contains only Macs. I use a MacBook Pro. Some of the course staff have experience with Windows, however, so you may find it a good idea to ask them your Windows-specific questions. Because I don't run Windows, I will be unable to help you with Windows-specific problems. Indeed, one great asset of our course staff is that many of them can help you with Windows issues.

Note that you can always email your program to yourself. So even if you normally run

Windows but you want to work on a Mac in the lab, you'll be able to work on your own code.

The Web

All documents, class examples, lab assignments, short assignments, and sample solutions related to the course will be on the web. You can get at them through the course Canvas site, or you can dial direct and go to http://www.cs.dartmouth.edu/~cs1/. Presumably you know how to find the web pages, since you're reading this one. The first short assignment tells you how to find the CS 1 information on the web.

Honor Principle

On exams, all work must be your own. You may work on short assignments individually or in groups. Programs that you turn in, however, should be created, typed, documented, and output generated, yourself. For the lab assignments, you may consult freely with instructors and classmates during the phase of designing solutions, but you should then work individually when creating your programs—typing, documenting, and generating output. During the debugging stage you may discuss your problems with others in the class, but you should not copy code to "fix" bugs. To do otherwise is a violation of the Academic Honor Principle. If you work with a classmate on any assignment, you should tell us who you worked with in a comment at the beginning of your program.

You should attribute the proper source in any code that you submit that you did not write yourself. This includes code that you take from outside references—for example a book other than the course text. And it even includes code that you take from class examples, a book, or the assignments. (I agree that may be tedious to attribute the source in code that we have given you, but we want you to be in the habit of attributing your sources.)

If you resubmit a short assignment and use code from the published solution, you should attribute that. Note also that proper respect for copyright laws as it applies to printed and software products is part of the Computing Code of Ethics.

Whenever we ask you to turn in sample runs of your program, the runs you turn in must be the result of actually running your program. It is a violation of the Academic Honor Principle to falsely represent output as coming from your program if it did not. If you change your program, make sure to generate output from the version of the program that you hand in. It's amazing how a seemingly minor change to the code can cause a big change to the output of a program. Also, make sure that when you are running a program, that it is your program; it is easier than you might think on a public Mac to run a program that someone else had left on the machine.

In the past, we have had a few incidents in which students turned in output that did not come from the program handed in. In each case, it turned out that the student had made a foolish mistake (in not rerunning the program or handing in an old version of the program or the output) and had not intended to misrepresent the work. Yet it caused many an uncomfortable moment for the student and also for the student's section leader and for the professor as well. So please—pretty please with sugar on top—endeavor to verify that you're handing in output that comes from the very program you're handing in.

It is not easy to come up with good homework problems that help you learn a concept, are interesting, and require an appropriate amount of work. Over the years we have developed and refined a number of homework problems, and I plan to reuse some of them for this class. You should not look at any solutions to homeworks assigned in previous terms, including sample solutions, or solutions written by other students.

We have had some uncomfortable situations occur in the past, and I want to make it clear what the policy is. Two students, Alice and Ralph, discuss an assignment and design their

code together. That is fine. But then they decide that since their programs would be so similar, they might as well have Alice type in the code, have Ralph make his own copy of the file containing the code, and then have Ralph make his own minor changes. This is a violation of the Academic Honor Principle. Although you may discuss and design with others, the code you hand in must be entirely your own.

Here's another situation that occurred. Trixie and Ed start working independently on a program. Trixie finishes and has a working version. Ed has trouble with his. Trixie helps Ed debug. That is fine. But then Trixie realizes that Ed has a section of code that is all wrong and the program she wrote has just the right code for that section. She shows Ed her program. Or worse, she gives Ed an electronic copy of her program so that he can just paste in the correct code. Either action is a violation of the Academic Honor Principle.

I realize that it can be hard to decide when you might be violating the Academic Honor Principle when we let you collaborate to a limited extent. Here is a good rule of thumb. If you are talking in normal English (or Chinese or German or some other natural language) you are probably OK. If you find yourself talking in Python code, you have crossed the line. So saying, "Your program runs forever because you have the wrong condition in the while loop" is OK. But saying, "Change while x == 0: to while x >= 0:" is not.

If you have any question about whether what you're doing is within the Academic Honor Principle, do not hesitate to check with me. If it's late and you can't find me, you're better off erring on the side of caution.

Most violations of the Academic Honor Principle come down to failure to cite work that is not yours. If you copy any portion of your program from your friend Elvira and represent it as your work, then you either intended to deceive or were careless about citing. Either case is a violation of the Academic Honor Principle. If you copy your entire program from Elvira but include the comment, "This code was copied in its entirety from Elvira," then you cited properly, though you didn't actually do the work. In this latter case, I would not report a violation of the Academic Honor Principle, though your grade on the assignment would be 0. But that would be far preferable to a COS hearing.

The same goes for code that you find in some other book or on the Internet. You are in violation of the Academic Honor Principle if you fail to attribute your sources.

You don't need to cite if you wrote the code yourself. You don't need to cite just because you're using a construct you saw elsewhere. For example, you need not cite for using `print "Hello"`, even though it was in the class examples. That would be like citing "printing press" in an essay! Nor do you have to cite just because you use a while-loop, even though you saw a while-loop in a class example. It's when you lift several lines of code from elsewhere that you need to cite.

To cite, include in a comment—near the top of your file is fine—stating where you got the code from:

```
# Based on the say_introduction function in the January 5 lecture.
```

Please do not cheat. Cheaters—whether or not they are caught—bring dishonor upon themselves and upon everyone else at Dartmouth. To do that, for just a few lousy points in a course, is [insert your favorite strong adjective meaning "stupid" here]. Furthermore, if you cannot solve the short assignments and lab assignments on your own, then you will be nailed on the exams.

I have served on the COS, and I have seen what happens to students who are caught cheating: they take an involuntary vacation from Dartmouth for several terms. I have

brought several Academic Honor Principle cases to the COS, and I hate having to do so. It is by far the worst part of my job. Please don't be the one that makes me contact the Undergraduate Judicial Affairs Officer. If you need help, come see me! If you need to cheat to do the assignments, then you are going to fail the exams, anyway.

Special note for when you work on a computer that anyone else might use

If you are working on a computer that is not yours—especially a Mac in 002 Sudikoff—or that someone else in the course might use, you should be very careful to remove your code from the computer when you are all done. You should probably email your code to yourself before you remove the code. Why do we tell you to remove your code when you're done? Because if you leave your code on a computer, and someone else can see it, then they can copy it and hand it in. If that happens, then we have a bad situation involving you (the copy-ee) and the other person (the copy-er), and it's difficult—if not impossible—to tell who was the copy-ee and who was the copy-er. By removing your code from the computer when you're done, you can avoid getting yourself into that situation.

To remove your code, you'll want to delete it from the PyCharm workspace. And you'll also want to move any other copies on the computer to the Trash (or the Recycling Bin) and empty it.

Religious observances

Some students may wish to take part in religious observances that occur during this academic term. If you have a religious observance that conflicts with your participation in the course, please meet with me before the end of the second week of the term to discuss appropriate accommodations.

Disabilities

I encourage students with disabilities, including "invisible" disabilities such as chronic diseases and learning disabilities, to discuss with me after class or during my office hours appropriate accommodations that might be helpful to you.

Students with disabilities enrolled in this course and who may need disability-related classroom accommodations are encouraged to make an appointment to see me before the end of the second week of the term. All discussions will remain confidential, although the Student Accessibility Services office may be consulted to discuss appropriate implementation of any accommodation requested.

Acknowledgments

This course was designed with the help of many students, including Sucharita Jayanti, Siddhartha Jayanti, Zachary Marois, Aaron Watanabe, Peter Johnson, and Weifu Wang. Many of the web pages and lectures are based on web pages written by Professors Devin Balkcom and Hany Farid.

Advice

Read the material I ask you to read, and follow the instructions. Many students have wasted a lot of their valuable time, and their grades have suffered, simply because they did not follow directions. Start all assignments early. With few exceptions, at the time you receive an assignment, you'll know everything you need to do it. There is no reason to procrastinate.

At the same time, do not be afraid to get help. The purpose of this course is not to waste your time. If you are spinning and not making progress on a problem, please see me, Keith, or a section leader. We can point you in the right direction without giving away the store.

Three final pieces of advice

1. Don't fall behind in this course.
2. *Don't fall behind in this course.*
3. **DON'T FALL BEHIND IN THIS COURSE.**

The material in CS 1 builds on itself, and the pace is fast. As a result, it's easy to fall behind in this course, and if you do it's very difficult to recover.