CS106X Course Information

Instructor: Jerry Cain  
E-Mail: jerry@cs.stanford.edu  
Office: Gates 192  
Cell Phone: 415-205-2242  
Office hours: Wednesdays and Thursdays, 2:45 – 5:00 p.m., and by appointment

Instructor: Truman Cranor  
E-Mail: truman@cs.stanford.edu  
Office: Gates 190  
Cell Phone: 541-852-0361  
Office hours: Tuesdays, 12:30 – 2:30 p.m.  
Fridays, 11:00 a.m. – 1:00 p.m., and by appointment

Website: http://cs106x.stanford.edu

Prerequisites: CS106X is the more advanced of the two courses teaching introductory programming abstractions and algorithms. CS106X is designed as an alternative to the more sensibly paced CS106B, because some students—self-taught programmers, exceptionally strong CS106A students, and AP Java graduates—prefer a more intense treatment in the company of other aficionados.

AP Java and CS106A are all about basic programming practices—expressions, control idioms, decomposition, algorithmic thinking, class design, object orientation, simple inheritance, and basic client use of arrays, lists, and maps. CS106X teaches advanced abstraction techniques, worrying first about C++ language mechanics and eventually focusing on topics such as recursion, inheritance, networking, event-driven programming, C++ lists, sets, and maps, and the implementation techniques used to build custom data structures.

Lectures: MWF 1:30 – 2:20 p.m.  
Building 200, Room 002

Our CS106X lectures are (and are intended to be) very conversational, feel-good and informal, working through material at an intense but manageable pace. We go through a good mix of examples—some drawn verbatim from the reader, but most are my own. We often stop mid-topic at 2:20 one lecture and pick up as if we never stopped talking two or three days later at 1:30. Jerry prefers whiteboard work where he can give the illusion that we’re collaboratively discovering new material, and Truman will often rely on
prepared slide decks to animate the material behind him while he explains key concepts.

Sections: In addition to the three lectures every week, you’ll also participate in a 50-minute discussion section (beginning the week of January 11th). The three weekly lectures are optional, in that you don’t need to attend them if you’re able to keep up with the material. However, discussion section attendance is required, and your presence and willingness to work on the exercises in good faith contributes to your final grade. The sections are mostly whiteboard discussion, sometimes with a little bit of coding, and the problems you’ll be discussing will be posted well in advance. Those with laptops should bring them to section, but those without laptops shouldn’t worry, as we’ll be pairing everyone up for the coding portion.

There are several discussion section times to choose from, and those times will be published to [http://cs198.stanford.edu/section](http://cs198.stanford.edu/section) by Thursday, January 7th at 5:00 p.m. You’ll have between Thursday at 5:00 p.m. and Sunday, January 10th at 5:00 p.m. to view the options and state your preferences. In the past, we’ve been able to assign the vast majority of students to their first choices, and virtually all to one of their top two. If after you’ve been scheduled to a lab time you find that you can’t regularly attend, you can contact me if we failed to reasonably accommodate your schedule, or you can just return to the CS198 web site and switch sections.

If you plan to complete your assignments with a partner (see the Assignments segment below), then you and your partner should ideally submit the same discussion section preferences and try to get into the same discussion section (although we won’t require it for CS106X this quarter).

Social Good: In addition to your normal discussion section, you may optionally attend a special discussion section dedicated to the intersection of computer science and social good. The section time has already been set for Wednesdays from 3:30pm until 5:20pm, and it meets in Building 160, Room 323. Those interested should visit [http://bit.ly/cssgsection](http://bit.ly/cssgsection) starting at 9:00pm this evening to sign up. The CS+SG section begins on Wednesday, January 13th.

Readings: The class textbook is Programming Abstractions in C++ by Eric Roberts, which should be available at the Stanford Bookstore. If you’d prefer, you can download a PDF of the reader from the course website and read from that.

In addition to the reader, we distribute a good number of handouts and slide decks, chock full of additional material and examples. All of the handouts are posted online to the course web site in PDF format, and it’s our expectation that you read the handouts online, printing them out yourself if that suits you
better. We will provide hardcopies of some handouts—discussion section handouts, assignments, and practice exams—when it’s clear that having a paper copy available is unambiguously better.

**Software:** Programming assignments can be written on either Macintosh or Windows PC computers using a development environment called QtCreator. More information about QtCreator will be provided online by this Wednesday’s lecture, when your first assignment goes out.

**Mailing List:** All students enrolled in CS106X are automatically subscribed to the cs106x-win1516-students@lists mailing list. The list server is in touch with Axess, so if you’ve signed up for the course, you’re probably on the mailing list already. Please make it a point to register for CS106X as soon as possible, since we tend to broadcast a good number of announcements during the first two weeks, and we don’t want any of you to miss them.

**Assignments:** There are six or seven programming assignments, and it’s possible we’ll throw in a written problem set as well. The assignments are serious projects, and they get more difficult as we cover more advanced material. The only way to learn programming is to work at it, so expect to spend lots of time in front of a computer. Your assignments are graded interactively in a one-on-one session with your section leader. In general, your section leader will meet with you and return an assignment within one week of the day you submit it.

We require—without exception—that you work solo for Assignment 1. However, you may (though you’re not required to) work with a partner for all other assignments. When you work with a partner, the expectation is that you each contribute an equal amount of work to the final submission, and our strong preference is that you pair program.

**Exams:** There will be two examinations

- **First Exam:** Thursday, February 4th 7:00 p.m. – 10:00 p.m.
- **Second Exam:** Thursday, March 3rd 7:00 p.m. – 10:00 p.m.

The first exam will cover the first four weeks of the course, and the second exam will cover the first eight weeks of the course, focusing on the material not covered on the first. Each exam could be administered in two hours, but we’ve scheduled three hours so as to do our share to remove whatever time pressure might otherwise be present.

There will be no final exam, which would have normally been held on the morning of Monday, March 14th from 8:30 – 11:30 a.m. Instead, your final assignment will be due at 11:30 a.m. on March 14th at the exact time when your final exam would have ended.
Grading: Your final grade will be computed as follows:

<table>
<thead>
<tr>
<th></th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments</td>
<td>50 %</td>
</tr>
<tr>
<td>Section Participation</td>
<td>5 %</td>
</tr>
<tr>
<td>First Exam</td>
<td>20 %</td>
</tr>
<tr>
<td>Second Exam</td>
<td>25 %</td>
</tr>
</tbody>
</table>

Assignments are graded on a bucket system, as we want to de-emphasize the letter grade and focus on the feedback. In the interest of transparency, however, here is a clear description of the various buckets and the numbers they correspond to.

+  Given to an exceptionally strong submission that not only meets the requirements, but exceeds them in some significant, algorithmically interesting way. In general, we see less than 5% of assignments getting +’s. The + is ultimately recorded as a 100 in the spreadsheet, since it’s clearly A+ work.

√+ Given to a solid submission that gets the job done and contains at most a very small number of trivial errors. In general, 35-40% of assignment submissions get the √+, which maps to a 95.

√  Given to a good submission that gets most of the job done and contains one or more major errors, or a significant number of minor ones. In general, about 45-50% of assignment submissions get a √, which maps to an 88 come spreadsheet time. This is the most controversial grade, because Stanford students don’t like getting B+’s on assignments. However, when we give them, it’s because the program wasn’t as good as it could have been and there were more impressive submissions.

√-  Given to a submission that does much of the work, but contains so many obvious problems that even a √ isn’t warranted. The √- maps to an 80 come spreadsheet time.

-  Given to a submission that clearly fails to solve the assigned problem or problems adequately. CS106X students generally don’t get these grades, but if you get one it’s because something clearly didn’t go well. The - maps to a 70 when we work all of my spreadsheet magic at the end of the quarter.

For each assignment, we also issue a companion style grade evaluating your overall design, decomposition, and code clarity. While issuing grades, we’re very open to different approaches, and penalties are imposed only when there are clear arguments that you overcomplicated something or your general coding style is subpar. Style grades are also bucketed, but we only issue √’s, √+’s, and √-’s. Functionality counts twice as much as style.

The class median on the first exam tends to be high—typically above 80
percent, while the median on the second exam tends to be between 70 and 80. When an exam median is 80 or above, your raw exam score contributes as is to your final average. When the exam median is below an 80, we curve the highest grade to a 100, the median grade to an 80, and everything else is scaled proportionately.

Those with a 90.0+ average (around a third of you, typically) at the end get some form of an A. Those with 80.0+ averages who don’t make it to 90.0 (all but a handful of you) typically get some form of a B, and so forth.

**Fair Access**

Students who may need an academic accommodation based on the impact of a disability must initiate the request with the Student Disability Resource Center (SDRC) located within the Office of Accessible Education (OAE). SDRC staff will evaluate the request with required documentation, recommend reasonable accommodations, and prepare an Accommodation Letter for faculty dated in the current quarter in which the request is being made. Students should contact the SDRC as soon as possible since timely notice is needed to coordinate accommodations. The OAE is located at 563 Salvatierra Walk (phone: 723-1066).

**Late policy:**

The pace of this course makes it difficult for students to catch up once they have fallen behind, so we encourage you to submit all of your assignments on time. Of course, we’re all busy people, so we understand when you can’t meet each and every deadline we put before you.

Here’s how we handle lateness: You get three free late days, and you consume one late day any time you hand in work between one second and one class period after the original deadline. Once you consume your three free late days, you can still hand in late work, but your late days are no longer free. For each additional late day, we subtract 2% from your overall homework average. In general, it’s wiser to take an extra late day unless you already think you’re in √ territory, in which case it’s probably not worth it. Of course, you should still work to complete the assignment and/or figure out what’s preventing it from working.

You may never hand in an assignment more than two class periods late, as it impedes your section leader’s ability to deliver feedback in a timely manner. And you may not use any late days on the final assignment due during final exam week.

**Incompletes:**

We only grant incompletes to those who complete all work due prior to the course withdrawal deadline, and only because of a severe illness or a family emergency. Understand that an incomplete is not a giant reset button you get to press to start over. Rather, it’s a courtesy extension on some end-of-quarter
deadlines to help mitigate a very poorly timed personal crisis. In general, all work must be completed before spring quarter begins.

**Honor Code:** Although you are encouraged to discuss ideas with others, your programs are to be completed independently and should be original work. Whenever you obtain help (from other students, the section leaders, students in other classes) you should acknowledge this in your program write-up, e.g. "The idea to use insertion sort instead of quicksort to alphabetize the list of names was actually my section leader’s idea." Even if you get help from others, the work you submit should uncontroversially be viewed as original work.

To be even more specific, you are not allowed to collaborate while actively coding, nor are you allowed to copy programs or parts of programs from other students. The following four activities are among the many considered to be Honor Code violations in this course:

1. Looking at another student’s code.
2. Showing another student your code, or making your code public so that it’s searchable and easily discovered online or elsewhere.
3. Discussing assignments in such detail that you duplicate a portion of someone else’s code in your own program.
4. Uploading your code to a public repository (e.g. github.com or bitbucket.com) so that others can easily discover it via word of mouth or search engines. If you’d like to upload your code to a private repository, you can do so on bitbucket or some other hosting service that provides free-of-charge private hosting.

Unfortunately, the CS department sees more than its fair share of Honor Code violations. Because it’s important that all cases of academic dishonesty are identified for the sake of those playing by the rules, we use software tools to compare your submissions against those of all other current and past CS106 students. While we certainly don’t want to create some Big Brother environment, we do need to be clear how far we’ll go to make sure the consistently honest feel their honesty is valued.

If the thought of copying code has never crossed your mind, then you needn’t worry, because I’ve never seen a false accusation go beyond a heated conversation. But if you’re ever tempted to share code—whether it’s because you don’t understand the material, or you do understand but just don’t have enough time to get the work done—then you need to remember these paragraphs are here.