How to Succeed in CS103

Think about how you learn to program computers. You have to learn a lot of new syntax (for loops, while loops, if statements, etc.) along with exactly how the syntax translates into what the program does (things like what variable assignment actually means, or when the condition in a while loop gets checked). After a while, you realize that most of programming isn't about learning new tools as much as putting together the tools that you already have in new and clever ways. The more you program, the more experience you gain and the more mistakes you learn from. At times, you'll get totally stuck and need to ask for help, which really is a normal part of learning how to program. Over time, if you keep programming, you'll get better and better at writing, testing, and debugging your programs.

Learning proof-based mathematics follows a similar process. You have to learn a lot of new syntax (proof structures, mathematical vocabulary, etc.) along with exactly how the syntax translates into what the proof means (things like what "assume for the sake of contradiction," "without loss of generality," etc. mean). After a while, you realize that most of mathematics isn't about learning new proof techniques as much as putting together the techniques that you already have in new and clever ways. The more practice you get with mathematics, the more experience you gain and the more mistakes you learn from. At times, you'll get totally stuck and need to ask for help, which really is a normal part of learning mathematics. Over time, if you keep doing math, you'll get better and better at solving problems and writing up proofs.

Math is a skill just like any other skill, and you will get a lot better at it if you keep practicing and you're strategic in your approach to learning.

Approaching this Class

Although CS103 is essentially a math class, chances are it's quite different than the math classes you've taken in the past, whether in high school or here at Stanford (Math 19/20/21, Math 51, CME 100, etc.) The most commonly taught and most commonly taken math classes focus on teaching math as *computation*. In algebra, you memorize the quadratic formula and apply it to solve a number of problems. In trigonometry, you memorize formulas like the Law of Sines or Law of Cosines and use them to deduce properties of triangles in all sorts of domains. In calculus, you memorize how to take the derivative of a polynomial, or (gulp) how to integrate tan x, then use those skills to take derivatives or integrals in contexts like biology, economics, or physics.

The mathematical content in CS103 is less about *memorization* and *calculation* and more about *understanding* and *argumentation*. The goal of this class is to teach you how to ask and answer questions of the form "why, exactly, is this result true?" Although there will still be some amount of memorization in this class – there are a number of core terms and definitions that you'll need later in your mathematical adventures – we think you'll pretty quickly realize that this class is a lot less formulaic than what you're probably used to. This is, in our view, a good thing. It shifts the idea of mathematics from "memorize and apply" to "think critically

and try to understand." At the same time, this fundamental shift will likely require you to approach this class in a very different way than what you're used to. Here are a few suggestions of specific areas in which you may want to be critical of your study habits.

Work in Pairs, But Be Strategic with your Learning

Although you are welcome to work in pairs on the problem sets, we ***strongly*** recommend that you attempt all of the problems on your own before coming together as a pair. We also strongly advise against having each member of the pair take just a few of the problems (say, the first half, or the even-numbered problems) and then combining all the answers together at the end. You and your friend are not likely to end up being good cooks if one of you just practices sautéing and the other just practices roasting. You and your friend are not likely to end up being good woodworkers if one of you just practices using a table saw an the other just practices using a miter saw. You and your friend are not likely to end up being good musicians if one of you just practices playing pieces in major scales and the other another just practices playing pieces in minor scales.

To learn to be a good cook, or to get good at woodworking, or to become a good musician, you have to practice a number of different skills in concert with one another. The problem sets in this class are specifically designed to get you to exercise a number of different skills. Some of them hit on specific details of the terms and definitions, others are explorations of pure theory, and others explore applications. If you take the time to work through them and really think about the material, you'll come away with a nuanced understanding of the core concepts from the class and how to apply them. On the other hand, if you just split the work up and do only half of the problems on the problem sets, you'll end up with much less than half the understanding we're expecting you to develop.

Persevere, But Know When to Get Help

There's going to be some point in this course where you find yourself struggling. Chances are it'll be a problem set question that you can't make heads or tails of, but it could also be a proof in lecture that just doesn't make sense or a definition that just seems plain wrong. ***This is a normal part of learning mathematics***, or any skill, for that matter, so when it happens, don't worry! When this happens, the question is what you're supposed to do to get unstuck or to resolve your confusion.

The Greek philosopher Aristotle is famous for a number of things – including, bizarrely enough, the first description of the mouth of a sea urchin – but one that's particularly relevant for this conversation is his idea that every virtue (good thing) has two vices (bad things), one of deficiency (too little of the good thing – like never eating cake) and one of excess (too much of a good thing – like eating a whole cake for breakfast every day). I'd like to talk about the virtue of perseverance: facing adversity and not giving up when things get tough.

Math requires a certain amount of perseverance. When you encounter a new problem and don't see how to tackle it, you need to have enough grit to continue working on the problem, drawing from what you've learned and trying out ideas even if you have no idea whether they'll work or not. The vice of deficiency is immediately giving up and asking for help. If you don't put enough effort into the problems you're trying to solve before you ask for help, then you won't get much practice learning how and where to apply different techniques. Giving up

early means you don't have the opportunity to make mistakes that identify gaps in your understanding. The most common manifestation of this vice would be sitting in office hours and listening to other students ask questions without first attempting any of the problems. We've seen a number of students do this. While it's a great way to *complete* the problem sets, it's a terrible way to *understand* the problem sets or to have the insights that the questions were designed to get you to have. Remember, you won't be able to listen to other people's ideas and TA suggestions when you're taking the exams – and the exams are two-thirds of your grade in this course!

The vice of excess is never for asking for help, no matter how hard you've tried. While it's always admirable to give something a good honest effort, it's easy to end up spinning your wheels pointlessly pushing against a problem without making any progress. The reason we have so many office hours and an active Piazza forum is to ensure that you have the space and opportunity to ask questions. If you've put in a good honest effort on a problem, exhausted all of your options, and still have no idea how to proceed, *please come and ask us questions*! The most heartbreaking cases we see in this class are when people spend twenty or thirty hours working on a problem set because they didn't feel comfortable asking questions. Putting in that many hours is exhausting, stressful, and *less effective* than calling it quits earlier and asking for help. Plus, we have no desire whatsoever for you to put in that amount of time!

It's ultimately up to you to figure out how best to calibrate the amount of perseverance that works best for you (Aristotle would call this the *golden mean* between the two vices). So hop on the Struggle Bus and put in a good honest effort, but know that it's okay to get off it as well.

Engage with Proofs, Techniques, and Definitions

As you'll quickly learn in this class, a lot of math involves learning new definitions and finding ways to prove things about those definitions. Often times, those definitions might seem unusual, complicated, or counterintuitive. Whenever you come across a new definition, it's a great idea to spend some amount of time outside of class playing around with that definition and seeing what it means. For example, let's take one of the first definitions we're going to see: even numbers. We say that an integer n is ***even*** if there is an integer k such that $n = 2k$.

I don't know about you, but when I learned about even numbers, I always learned that they were numbers that end in 0, 2, 4, 6, or 8. How does that relate to the definition given above, which says nothing about the last digit of the number? Well, one way to find out would be to try out a few numbers and see what happens. The number 16 is even because we can write it as $8 \cdot 2$. The number 6 is even because we can write it as $3 \cdot 2$. The number 26 is even because it's $13 \cdot 2$, and the number 36 is even because it's $18 \cdot 2$. Hmmm, that's interesting – seems like numbers that end in 6 are always two times some number that ends in either 3 or 8. How about the numbers 12, 22, and 32? They're $6 \cdot 2$, $11 \cdot 2$, and $16 \cdot 2$. So numbers that end in 2 seem like they're always twice some number ending in 1 or 6. With a bit more exploration – which you totally should do, by the way! – we can get a connection between the formal definition of even numbers and our intuitive definition. Any number n that's twice some integer k will have to end in 0, 2, 4, 6, or 8. So that gives us a bit more of an understanding of where the definition comes from.

What about 0? When I was in elementary school, my fourth grade teacher told me that zero is neither even nor odd. But according to our definition, 0 is an even number, since it's $0 \cdot 2$. Huh.

Looks like zero actually *is* even. The only reason we know that is because we looked at the definition and played around with it – our prior conceptions or notions don't always sync up with how mathematicians define things!

There's some other questions we can ask about this definition. Why does k in the definition have to be an integer? What if we said, instead, that n is even if $n = 2k$ for *any* choice of k, not just one that's an integer? But that would mean that a lot of things that don't "look" even would be even. For example, that would mean that 3 is even, since $3 = 1.5 \cdot 2$. Oops. That also means that 1.6 is even, since it's $0.8 \cdot 2$. In fact, *any* number would be even, since any n can be written as $2 \cdot {}^{n}/_{2}$. So that gives a better understanding of why we said that k had to be an integer – it prevents us from doing things like this.

As you can see, this a pretty lengthy discourse about what, at face value, might look like a simple definition. This exercise isn't meant to be pedantic – it's meant to be instructive. When you come across a new definition, we recommend taking the time to play around with that definition. This gives you a better sense for what the definition captures and might reveal some aspects of that definition – often unusual cases – that aren't apparent at a first glance.

As you're going through CS103, take some time to review your lecture notes (see the next section) and play around with the concepts you encounter. If you have questions on them, you are welcome – and, in fact, encouraged! – to ask them in office hours or on Piazza. Students who ask questions like these overwhelmingly tend to do extremely well in this class.

Take Notes in Lecture

We always post the slides from each lecture up on the course website, and those slides are pretty detailed and have a lot of explanations in them. However, reading posted lecture slides are *not* a substitute for taking your own notes in lecture. Historically, students who take their own notes in class tend to do substantially better in the course than students who don't. Taking notes makes you put the concepts down in your own words and requires you to think a little bit about what it is that you're learning.

Although I try to put as much as possible in the slides, there will always be parts of the course content that aren't in there. People ask good questions in lecture. I'll think of some random tangent and discuss some perspective or thought process that isn't explicitly written down. If you're taking your own notes, you'll end up with a better record of what went on in class.

And Finally, Advice From Students Past

At the end of the Fall 2014 offering of CS103, I asked everyone in the class to offer one piece of advice to future CS103 students. I've compiled and organized their answers here in this handout. The quotes you're seeing here are from actual CS103 students. I hope that it gives you some input about how best to approach CS103 and how to have the best possible experience!

Start problem sets early

> *Start the problem sets early so you have time to think through your proofs.*
>
> *Start all the PSets several days early! Create an independent understanding!*
>
> *Start the homework early so that when you become frustrated with a problem, you can take a break for a day or two; the intuition might strike you when you least expect it.*
>
> ***DO NOT LEAVE YOUR PSETS UNTIL THE NIGHT BEFORE!!!***
>
> *Definitely start the problem sets early. Once you're behind in this class, you're behind forever.*
>
> *Start your psets early and not the night before*
>
> *Start problem sets early! They actually all take a completely reasonable amount of time; most of the stress came from attempting to cram the entire problem set in the night before.*
>
> *Don't rush through problems, they need to stew in your brain.*
>
> *Spend time wrestling with problems; it gives you a sense of confidence.*

This was the most common advice that former CS103 students had to offer. One of the biggest mistakes you can make in this course is putting everything off until the last minute. If you find out at the last minute that you don't actually understand some major concept, you might not be able to get any help. Some of the problems on the problem sets might require some time to work through, and if you start earlier you'll have more time to think over different approaches.

Stay on top of the material

Don't fall behind on material since there's a lot of it

All the material builds on itself, so don't get behind.

Make sure you keep up, and if anything is unclear, work on it right away.

Don't let the material get ahead of you.

The material in CS103 builds on itself. We *strongly* recommend taking some time after each lecture to review what was covered and to play around with the concepts. Sometimes you'll realize that there are concepts you're not comfortable with or that just don't make sense. When that happens, don't worry! It's normal. You might just need to get more practice or to get help and advice from someone else.

Attend lectures in person and take notes

Go to lecture and take good notes.

Go to lecture and stay on top of the material

Pay attention in class and your life will be a lot easier!

Do not fall behind in lecture. Go to lectures.

Pay attention in lecture!

Make sure to be always up-to-date with lectures

Stay on top of lectures!

Make sure to stay on top of lectures--digging a lecture hole is stressful and hurts your learning.

Stay on top of lectures because they build on one another.

The lectures in CS103 are pretty packed with information. Even though we'll post the slides online, it's important that you keep up to speed with the lectures – there's a lot presented that isn't in the slides.

If you're following along with CS103 online – either because you're taking the class through SCPD or because you're an on-campus student and just want to time-shift the lectures – please, please, *please* do not wait until the last minute to binge-watch a bunch of lectures. That is a really easy way to fall far behind. The problem sets in this class take time, and if you wait until the Thursday before they're due to start watching the lectures, you'll make the class extremely hard for yourself.

Do all the problems on the problem sets, even if you're working in a pair

> *Do ALL the work in the homeworks, then compare results with the rest of your pair and fix problems -- splitting the problems in half and trusting your partner to solve them leaves you with a shaky understanding and worse grades.*
>
> *Having each member of the group work through each problem separately before discussing worked very well for my group.*
>
> *Do every problem in the Psets even if you are distributing the work among group members.*
>
> *Do every problem in the PSETs even if you're working in a group so that everyone is familiar with the way of thinking required for different problems.*
>
> *Be careful in working with groups, make sure you pull your weight and understand everything.*
>
> *Find a group, even if you all do your own psets, to talk about the problems with.*
>
> *Don't rely on others for psets. Struggling through them helps a lot with building an intuition.*
>
> *Do every single problem on the pset (don't just divide it up) and give yourself at least two days, because sometimes you really just need to sleep on a problem.*
>
> *Do homework's individually before meeting as a group*
>
> *Do homework by yourself before collaborating - it gives you better understanding*
>
> *Really spend time discussing the PSet and any topics in class with your team members. It's good for your soul.*

From experience, the students who did best in this course were the ones who knew how to work well in a team. We *strongly advise* against splitting up the problems in a problem set across different teammates – you'll end up getting less practice with the material and spending less time discussing important ideas with others.

There are many advantages to having both partners solve every problem. If everyone does the problems individually, then everyone in the group can come together and talk about their solutions. You might find that your approach is not the same as your partner's, which could mean that you've found an alternate solution or that one of your solutions has a flaw in it. Taking the time to figure out which of these cases you're in will give extra practice reading proofs. Plus, having everyone write up their own proof gives everyone in the team a chance to read over everyone else's solutions, which will help you catch typos or logical errors. Oh, and speaking of proofreading your answers...

Proofread your problem sets before submitting

> *Re-read your problem sets before you submit them, you'll catch a lot of silly mistakes that way*
>
> *Make sure you read and re-read EVERY PROOF you write, even it you think it's perfect.*
>
> *Make sure you actually understand everything in detail. Also, pay very close attention to wording!*
>
> *Pay a lot of attention to the wording of the proofs. Copy terms until you feel comfortable. Don't be like me and go cowboy with your proof writing style!*
>
> *Make sure your partners check your answers*

In programming courses, you can test your software before you submit it. You usually have several levels of defense – the compiler will check that you don't have any syntax errors, and testing can reveal bugs that you might not have noticed. In math, there aren't (that many) automated tools to check your answers. Before you submit your answers, be sure to proofread them and make sure that they're clear, easy to read, correct, and make sense. If you're working in a pair, this should be pretty easy: just hand your proofs to your partner and ask them for their honest but polite feedback. If your partner has trouble understanding what you're saying, talk over your reasoning with them and see if you can clear up the confusion. Then, based on your conversation, rewrite your proof so that it's cleaner and easier to understand.

If you're working individually, proofreading is still possible but is a bit trickier. We recommend writing a proof, sitting on it for a day, then reviewing your answer a day later to make sure that you can still understand it. If so, great! If not, consider cleaning up your answer – after all, if you wrote it and have trouble understanding it, what do you think will happen when the TAs read over it? 😊

Get help when you need it

> *Discuss with classmates and TAs at every step of the process! Ask a lot of questions.*
>
> ***don't be afraid to ask for help***
>
> *if you get stuck, ask for help*
>
> *Go to office hours even after you have finished the PSET. It will help you check whether your intuition for concepts is correct and help you gain a deeper understanding of the material.*

It is ***completely normal*** to get stuck and need help at some point in this class. When that happens, you have a lot of options available. If you're working in a pair, you can ask your partner for help or advice. If you haven't already done some of the practice problems from CS103A, you can take a break and do some of those instead. You can also ask questions in office hours or on Piazza if you'd like the staff to weigh in.

Get lots of practice

> ***Practice! Do EVERY problem on your own, and then do them again. It helps so much.***
>
> *Practice makes perfect. The extra problems are very helpful in solidifying your grasp on concepts.*
>
> *Math is not intrinsically hard, the more you do, the better an intuition you build for it, and that's what separates a mathematicians from others.*
>
> ***DO AS MANY PRACTICE PROBLEMS AS POSSIBLE and try to think of everything as a cohesive, big picture.***
>
> ***DO TONS OF PROBLEMS.***

Each week, we'll release a problem set that you'll submit for a grade. We'll also release some extra practice problems in CS103A you can use to play around with the material. If you're shaky on a concept, try working through some of the practice problems first. And, around exam time, we'll release a ton of extra practice problems and practice exams. Work through them – you'd be amazed how effective that can be.

And finally – some general advice from Fall 2014

> *come in with an open mind and eagerness to learn*
>
> *Work. Your. Butt. Off. From the start. You can do it.*
>
> *Let go of the conception that CS is all about writing code and learn to appreciate the deeper logical thinking behind it.*
>
> *If you are stuck on a problem give your mind a break because you can easily lose your calm if you just keep trying to solve one problem for hours without end.*
>
> *One good way to study is to go through the theorems from lecture, cover up the proofs, and write them up yourself.*
>
> *Don't worry if the problem sets are taking you a long time--they are difficult, but if you do them, you'll feel it in your brains.*

Best of luck in CS103 this quarter!