Programming Methodology-Lecture27

Instructor (Mehran Sahami): So welcome back. Wow. That's a little loud. To our last week of cs106a. Of course, it is another fun filled exciting day despite it being our last week. We're getting down to the end. We have class today, there's class on Wednesday, there's no class on Friday. So next time will be our last day. But a few announcements. There's actually just a load of announcements because we're so close to the end of the quarter. First announcement, there's one handout, which is your section handout for this week. There are still sections this week, so despite the fact that we don't have class on Friday, still go to your sections this week.

There's a couple problems on the section handout, as well as the sectional will just be a general review for the final exam in case you have any questions. That's a good place to ask them. Also it would be a good place if you want to ask some questions say, about [inaudible] for example, the last assignment. Just wondering, how many have started assignment number seven? Wow, good to see. Anyone done with assignment number seven? A couple of folks. That's good to know. I might talk to you afterwards as to how much time it actually took you, but hopefully, it wasn't too painful. The graphics contest was due last Friday. The winners will be announced in next class, so Ben and I actually took a first pass already over all the contest entries. There was actually some very impressive entries in the contest. Things you were just kind of jaw dropping, like, go and show them to other faculty in the department because they're just that cool. But this afternoon we are having our staff meeting with all the section leaders and they will actually be the ones voting and deciding on the winners in each category. So we'll give them the short list and they'll make the final determination, and then on Wednesday, I'll announce it to and I'll check with the winners of the contest to see if they're okay demo'ing it. But if they're okay demo'ing it, then I'll show you the winning contest entries as well, plus on Wednesday we'll have the random drawing to give away the last sort of grand prize. But if you didn't happen to win, just for entering you still get an entry into the grand prize or you could just get a free 100 percent on any assignment in the class including the final exam. So assignment number seven, we just talked a little bit about. Since it's due the last day of the quarter on Friday, but we don't have class that day, it's just electronic submission. So if you're wondering about what do I do with the hard copy, you don't need to do anything with the hard copy. We just need electronic submissions.

For the other assignments in this class, we requested that you turn in a hard copy because when you did interactive grading or if you're a section leader to write comments on, we actually had something that they could mark up. For assignment number seven, because it's due the last day of the quarter, there will be no interactive grading for the last assignment so the down side is there's no interactive grading for it, the plus side of that is that you don't need to turn in a hard copy because we can just take a look at your online submission to figure out functionality and other kinds of things. And just as a reminder, even though it wasn't clear that was on the handout, like, right on the front the last couple days, no late days on assignment number seven. Just in case you're wondering. So final exam, it's time to start thinking about the final exam. Finals are next week. You probably

know that, but just in case you didn't, the final exam is just like the mid-term. It's open book and open notes, so you can bring in your text book for the class. Feel free to bring in all print outs of all your programs, all the notes you've taken in the class, all your handouts. That stuffs all open, but just like the mid-term, it is a closed computer exam. So if you have a laptop or a PDA or whatever, you can't use that during the exam. Same rules basically applies to the mid-term. And we'll talk a little bit more about the final when we actually do review for it next class.

The regular final is scheduled for Thursday of finals week. That's December 13th, 12:15 to 3:15 P.M. in Kresge Auditorium, which is the same place we had the mid-term. One of the only rooms large enough to actually accommodate us, and the alternate has been scheduled. So alternate final is December 12th, that's Wednesday of finals week, 3:30 - 6:30 in the afternoon, also in Kresge Aud. So this time seemed like a relatively unpopular time for other final exams. And you're free to take either one. So you don't need to send me an email saying you have a conflict with the regular exam or whatever. If you just want to get done with your finals earlier and you want to take the alternate exam, you're just welcome to take the alternate exam. But only take one exam. So you can pick one, and just one. That's just life in the city. Okay. And if you're an SEPD student, I announced this last time, but I'll announce it again. I've already gotten email from one of you, which is a good thing. To email me by 5:00 P.M. December 5th, that's Wednesday if you're planning on taking the exam at your site. If you're gonna come in for the exam, you don't need to email me. You can, feel free to email and say, "Hey, man, I'm gonna come into campus and take the exam." And you're welcome to take it at either one of these times as well as if you're out in SITN. But if you plan on taking it at your site, send me an email. Also, let me know the name and email address of your site administrator so I can send the site administrator your exam to administer to you. So that was just a load of announcements.

Any questions about anything before we delve into our next great topic? All right. You're feeling okay. Good. So a lot of today's class is actually about life after this class because we're getting pretty close to the end of this class. So one of the things I want you to just kind of know about and so you can think about it, are what are some of the options that are available to you afterwards. Whether or not you're just thinking about declaring a major or if you've already declared a major or you just want to get sort of a lay of the land of what's this whole computer science thing all about. Because probably the biggest thing I would stress, despite the fact that you just spent the last nine weeks programming, is computer science is not computer programming. Okay? A lot of time the two get equated, but if it was called computer programming, this class wouldn't be called programming methodology, we'd just call it something like programs that work, right, and we wouldn't worry about style and all this other stuff, and good software engineering principles, and at the same time, computer science wouldn't be called computer science, it would called something like programming. Right? And it's, like, "Oh, what did you major in?" "Oh, I majored in programming," and that's like, when you say, "Oh, I'm sorry. I think you can get shots for that kind of thing now." Because it's not just about programming. There is programming in computer science, but there's actually a science to the field and there's a lot of things that go on outside of programming and that's what

it's important to, in some sense, appreciate. So if we think about life after this class, let's first kind of deal with some of the short-term logistical kinds of things. Like, you just took this class; you might think, well, there's probably a couple things you think. You think, "Hey, Miron, that was kind of interesting, I might consider taking 106b." You might consider, "Hey, Miron, that was interesting, I might actually considering minoring or majoring in computer science." And you might say, "Hey, Miron, that was interesting, in the same way, for example, that dropping a brick on my head is interesting, and I'm gonna run screaming."

And if you're thinking the third option, I apologize, because that was not the point of this class, but here's a few things that you can potentially think about, even if you're in the third option and definitely some things to think about for the first two options. And I guess there was also that option of the, "Oh, I got the general education requirements out of the way and now I will go on figuring out what to do with the rest of my life," and if that's the case, you should pay attention as well. So what happens after 106a? So here's cs106a, this is where we're all sort of happy, and we're scrappy and we're making social networks, and life after this kind of, you know, your next immediate step is actually pretty clear. There's a class cs106b, that's called Programming Abstractions, which is the next class to take. And that class is on a language called C++, so you'll learn a whole new language, although, you'll realize when you actually CC++, that a whole bunch of things in it are just the same as Java. Whole notions of parameter passing and methods, and decomposition and objects, all those same things exist in here. Okay. But you also will get with this class called Programming Abstractions because, so far, what we've done is used a lot of extractions. There you get into a whole bunch of tradeoffs with how you can make things run more quickly versus perhaps using more memory versus different kinds of programming techniques that actually come up. There's also some really cool ideas that come up in here, which are just sort of mind blowing ideas, which is the notion – for example, one of them is called recursion, which is so far we have methods and methods call other methods and they call other methods. What if a method called itself? That's kind of weird, Miron. Why would a method call itself? Because some functions are defined in terms of themselves. Right?

If you kind of think about the factorial function – anyone remember this function? The "N" function. Right? This is N factorial. And all this really is – sorry if I just shattered your ear drums, is N times N minus 1 times N minus 2 times….times 1. You just multiply everything together. That's where [inaudible] all about. You can define a function in terms of itself. And it turns out, yeah, a factorial, that's kind of a simple way to understand it. It turns out that this is a hugely powerful concept that allows you to do all kinds of things, and this is kind of another cool thing you get in cs106b. Okay. Now, you might say, "Okay, Miron, that's still sounding like programming to me, even though I'm learning these cool concepts, isn't that just a programming class," and in some sense, yeah, this is a programming class. There's other options that are also available to you now that kind of fall into the category of being part of the CS major or the CS minor. A set of classes called cs103. And cs103 come in a different couple different flavors, like, vanilla, grape and pork – no, they come in those in a, b sequence and there's [inaudible] – I can't think of anything in the world that would come in those three flavors.

And this is really a class that in some sense is about discrete math. And you might say, "Oh, gee, Miron, besides your class, I'm taking calculus and that's about as much fun as sliding down a 50 foot razor blade. Why would I want to do that again? Not on the sharp side, right, just imagine the other side, like, the flat side of the razor blade and it's been made slick and it's like a big slide. It's fun. Wait until all my friends in the math department see that. Anyway, why would I care about this discrete math thing? Well, first of all, this is an operative word here, which means this little symbol that you have grown to know and love, our friend the integral, just nod around, right, this is all discrete, this is, like, "Hey, you know what, what we want to think about our some things that are useful to us in a computer science context," and computers at the end of day are digital objects. Right. They have ones and zeros, which means there's a whole bunch of things, like, sets for example and logic that come up in these things. But there's also interesting ideas that come up in here, like, computability. In these classes, you get exposed to some things like some of the biggest open problems in computer science.

Now, there isn't time to go into what some of the biggest open problems in computer science are, but there's a problem called the P = NP problem. Right. And this is a big question mark. Basically, we just don't know if these two things, one of them named P and the other one named NP are equal to each other or not. And you'll find out what those are in the class and you might say, "Okay, Miron, why do I care about that?" Because it turns out this little problem here, has a $1 million prize associated with it. And it's simple enough to explain that after having had 106a when you take these next two classes you'll actually get exposed to this problem It's one of things that's, like, a minute to understand, a lifetime to master. And no one's mastered it yet. But in some sense, this is also a problem that's only about 35 years old. Maybe just slightly older than 35 years old. So it's not like this problem that's existed for, like, hundreds of thousands of years and, you know, cave people were writing does P = NP on stone tablets. This problem actually came to the floor and people realized it was an important problem in the 70s, which means it's possible that it'll be solved in your lifetime, and it's possible that you may be the one, presumably, solving it in your lifetime because it would be difficult to solve it if it wasn't in your lifetime.

So even if programming, by itself, doesn't necessarily turn you on, but you think, you know, programming is interesting, is there also some deeper science or some mathematics because for a lot of people, they didn't necessarily get exposed to computer science earlier on, but they did get exposed to mathematics, this might be the kind of thing that really turns you on. Now, you might say, "Okay, Miron, math doesn't necessarily turn me on, programming turns me on." Besides that 106 class, what other options are there? There's two other classes, cs107 and 108. And these classes, basically, look at building, in some sense, larger scale systems, so this involves object oriented systems and in some sense, building larger applications. So you build some things here which are outside the scope of a one or two-week project, like, you might spend four weeks on a large project in this class by the end and actually build a fairly substantial application, and 107 looks at a whole bunch of issues, that in some sense, we like to think of as lower level kinds of issues, but it involves a lot of programming and it gets into the nuts and bolts as to how does the software sit on top of the hardware of your machine and

how do these things interact and getting into understanding memory better and whole bunch of other things.

And if you think about this set of class, like 106a, b, 103a, b and 107, 108, if you were to take that set of classes and add to it 2cs electives, that's the minor. Okay. So the minor is basically these six core classes. You need to take math up to math51 I should say as a little side note. Just in case you're wondering. That's just something, that, you know, we're not responsible for that, it's just kind of required. And then two cs electives beyond this kind of stuff and then you're getting a minor. Okay. So if you want to kick it up a notch beyond a minor and think about the major – actually, I'll just leave this up. Two cs electives, you sort of add that all together and it equals the cs minor, which is kind of fun. Okay. Now, if you want to think about besides just a cs minor, potentially, actually majoring in cs, you might want to think about, "Okay, first of all, what are some other things that I can do in computer science beyond the introductory classes," and there's a whole bunch of things. There's something that we call artificial intelligence, or just AI, for short. And there's a whole bunch of aspects of artificial intelligence. That's sort of the highest level.

It's the notion of trying to make your computer work more intelligently, and in some sense, appear to be more intelligent, sort of on the order of the intelligence of a person. But really this has a whole bunch of sub fields to it, for example, robotics and various other things such as computational biology, there's a lot of computational biology that's ground in artificial intelligence. Data analysis and I'll show you some examples of these as we go along. And this is today, and there's a whole bunch of people in the world who are wondering what happens tomorrow. And if you can do slightly better than 50 percent predicting what happens tomorrow based on analyzing all the data from today and before, you make tons of money. Okay. And if you wondered is this really the case? Yeah, in fact, anyone heard of a company called D.E. Shaw? Yeah. Anyone. A few folks. Yeah. It's David Shaw. He was actually a grad student at Stanford in computer science. And this whole – I wouldn't say he started this whole thing, this actually existed long before that, but there's whole companies whose entire business is based on the notation of quantitative analysis and guess who are a bunch of people that they employ? Computer scientists who go and do the data analysis and actually figure this out. Okay.

So the application and understanding what are all the variables that you care about and the information that exists in the stock market that you can extract and model with different kinds of algorithms to make your prediction, is all part of what computer science is all about. Besides, AI, there's various other kinds of little areas. I'll show you some more pictures, like, robotics. Anyone heard of Darpa Grand Challenge or a little robot called Junior or Stanley? Yeah. Oh, Junior, he's so cute. Because it's a robot, that in some sense, is a car. Right. And there's no reason why a car can't be a robot. Just think if Carol had wheels on it, and instead of move…you had move at 60 miles an hour, you'd be doing the same thing, except you'd be doing it in a simulation. This is Stanford's car, Junior, and this is a car that's basically a robot. It doesn't have a human driver, at least most of the time. Right? It has things like various kinds of sensors on it, various sorts of radar and other kinds of laser range finding that sense what's going on in the world and

then it makes decisions. Okay. And so let me show you a little example of that. So here's a little video of Junior actually involved – the joy of software.

That's another thing you can do as a computer scientist. You can fix other people's bugs. Here's what's actually going inside Junior when it's actually running along. It's sensing a bunch of things about its environment, and you can actually see it's driving along – this is where it has some uncertainty or some distribution over where it thinks it is, where it thinks different lane markers in the road are and it's doing all this by actually taking pictures of the road, analyzing them in real time and then making various kinds of decisions about where to steer and where to go. And this is all happening in real time. Right. This isn't, like, "Oh, we had to load all this data and figure it out on some super computer," there's just a little bank of computers inside of Junior that is actually figuring this out as he goes along. It figures out certain places to stop or how it's going to re-maneuver itself. Let me show you the set of computers that are actually doing this. They're just sitting in the back of the computer. Yeah, there's a few different machines in here, but it's sort of computational power on par with what you're gonna have in your dorm room by the time you graduate basically. Let me show you one more quick example of Junior actually parking. Okay.

So these little red marks over here are actually cars and it's basically sensing that these areas are blocked and what it wants to do is get to a parking spot that's between two cars right here. So it plans this little path and it looks like it's gonna rear end this other car over here, but really all it's doing is repositioning itself so it can re-plan to be able to back up and then pull into the parking spot. Right. And if you think about all of the dynamics that need to be going on to do this, all the low level stuff to sense where things are, the high level planning to figure out how sharp of a turn it can make and now it's gonna back out and drive off. All of this stuff is basically just software. It's a computer science problem. And that's how the junior team actually views this robotic car. They view it as there's a bunch of sensors in the car and there's some actuators, like, they can hook up computers to the steering wheel to turn it and really the whole problem is solved in software. How to do the planning, when to turn the steering wheel, by how much, when to figure out if lanes are blocked, stuff like that. Okay. So that's a little bit of AI. Let me show you a few other fields. Okay. So besides AI, and there's a class related to this, cs121 or 221, you sort of have your choice.

This is kind of a survey of artificial intelligence and this is kind of, in some sense, modern techniques for artificial intelligence. If you really want to go and build robots, I sort of suggest you take 221. If you want to get a lay of the land, of what's in artificial intelligence, you can take cs121. Okay. Some other things that you take along the way are a class like cs140, Operating Systems. Right. And if you've ever wondered about things, like, "Hey, I have my Mac, how does my Mac actually do all this stuff for me, how does it take care of a file system for me, how does it take care of the fact that there's multiple things running at the same time, how does it deal with the fact that I may actually be running more applications than I actually have real ram in my computer?" There's a notion of virtual memory, for example, where it uses your disk for part of memory. That's all stuff that's covered in Operating Systems class, and if you're interested in

systems kinds of things, there's just a ton of things that you'll in here that you can kind of build on. Right? Graphics is a big area that's in cs, and it turns out, interestingly enough, of our graphics faculty, Pat Hanrahan is one of the faculty here. He actually has, not one, but two academy awards. All right. Interestingly enough, he's actually got Oscars. Right. And you might wonder, "Why does he have Oscars, Miron?"

Well, because guess what, there's all these animated movies these days, there's a system called Render Man that was actually responsible for being able to do a lot of the rendering for original computer graphic movies. He was on the team that built that system. And he's done a bunch of other stuff since then, which is why they gave him a second one in 2004. Okay. There's a guy named Ron Fedkiw, and I'll show you a little animation that his group developed. So here's what looks like a lighthouse and water, and here is basically, a realistic computer animated waveform crashing over the lighthouse. Right. This was all done. This wasn't like scanned over some real lighthouse when there was flooding. This is all basically done as a computer simulation. All right. That's the kind of stuff his group does. And as a matter of fact, for doing stuff like this, it doesn't just show up in little animations to show in 106a, if you happen to see Star Wars 3, he was in the credits for it, if you happen to see – what were some of the other movies he was in – anyone see Terminator 3? Horrible movie. Don't see it. But he was in the screen credits for that as well. Evan Almighty, yeah, so there's serious movies that involve major computer graphics where the stuff that's being done here is actually at the cutting edge of that to be able to figure out new ways of actually doing things with computer graphics and actually doing the animation. But there's other things you can do. Like, here's a mis-focused camera, you just bring the picture into focus automatically. Here's a really blurry one. Awe – pretty hardcore. And here's focusing through a splash of water. So it doesn't just have to be a picture of some solid object. I hope you can actually see that re-focusing while it's happening.

Then we get into the audio part. I won't share the audio part. It's kind of more of the same. But that's the basic idea. They're actually starting a company around this idea of light field photography where you have a camera and just the way the lenses is constructed and the amount of light that you sample at various kinds depths of fields allows you to take this image and then be able to refocus on different parts of it later or clean things up or whatever. That's just another thing that's kind of based on graphics that you wouldn't necessarily think of right, but photography really is taking some sample of the world, turning it into a graphical image and then doing manipulations on that image.

So a lot of the things that happen in graphics, apply directly to photography as well. Okay. So besides graphics and robotics, we talked a little bit about those. There's folks that worry about stuff like databases, like, handling large volumes of data on streaming data, on different kinds of things you could do with data and I was kind of thinking about this and I was, like, what's a demo I could show having to do with large volumes of data because that's not something you can actually draw a picture of real easily. And then I just thought I'd show you this. Because Google came out of Stanford. It came out of a group of folks who did things like understanding data structures and the algorithms

associated with them and who understand how to keep track of large volumes of data and be able to do manipulations on that kind of data. And in the early, early days, most people don't know this now, but if you went to google.stanford.edu was the web address for Google. Okay. And it turned out at some point this was actually eating up so much of the entire bandwidth on campus that some folks said, "You really need to go and move this somewhere else," and then they actually created the company Google, which is based on a misspelling.

Right. The actual – does anyone know what a Google is, which is the correct spelling of Google, is ten to the hundredth power, it's 10 with a hundred zero's after it. And so Larry Page and Sergey Brin were grad students here and they wanted to think of same name that captured the largeness of Google or of the web search that they were doing so they went off and registered Google because that's how they thought it was spelled, or at least one of them, and I won't tell you which one thought that. When they were grad students, and then when the other one of them came back to the room and looked at it he said, "You misspelled it," but two things transpired. One was that this .com was already taken, and the second one was when you're a grad student and at the time it was, like, $50 or $70 to actually register the name, that's kind of spendy when you're living on Ramen. So that's what it was. Okay. But it just shows you the kinds of things that get done by taking basic ideas in computer science and building them to a larger scale. Other things that go on. I'll just give you a brief sampling.

Cryptography, which is big for web security. Right? It turns out a lot of the web is actually pretty insecure. Much more less secure than you would actually imagine. Anyone ever had a credit card number stolen? A few folks. Yeah. When you get your credit number stolen, then you think twice about a lot of the transactions you make. I had it happen, actually, a couple times and I still, like, you know, Christmas time rolls around, I'm just like online shopping until the cows come home. But it's important to actually think about what's secure and what's not secure. And there's actually a group that deals with cryptography, especially security in the context of the web.

Other kinds of things that go on. We talked about AI, and sort of a sub field of AI, which is growing into a whole area of its own, is machine learning, and I talked a little bit about things like biology or predictive data analysis. There's actually also machine learning that affects your life on a daily basis, whether or not you know it. How many people have a spam filter on their email? Anyone? Yeah, did you know that chances are probably in all likelihood that your spam filter is actually based on machine learning? It's seen a whole bunch of email, some mail that was spam, some mail that wasn't spam. And it learned, no one told it what was spam and what wasn't spam. It learned to figure out how to distinguish between what's spam and what's not spam. Now, it's not perfect. Right? People aren't perfect either, so sometimes you get messages in your inbox that are spam, and every once in a while, rarely, but it happens, someone sends you a message and you never hear about it, and they're like hey, I sent you this email and the you go check your spam folder and it shows up in there. But spam filtering is another one of these things that in the last oh, ten years or so, is another something we take for granted and don't think

about the fact there's actually a bunch of science under the hood as to how to do this and people continue to do research how to improve it.

And there's a bunch of other things based on this, like, robot and navigation. Some of the stuff you just saw with Junior, is actually based on learning landmarks of the road or learning where lanes are on the road or what obstacles actually are. There's a ton of other things. I'm just giving you a sampling. Now, if any of this has interested you at all, there's a guy you need to go see. One guy you could go see is me, and I'd be happy to talk to you about any of this, but there's another guy you can go see whose name is Dave Koslow. And Dave, is what we refer to as the CS course advisor. I'll just put the CS advisor up here. He's in G160. He's the guy you see when you want to declare a computer science minor or major. Not that I'd be putting in a plug, but he's an interesting guy to talk to about some of the different possibilities in the field, but open invitation. So this class is gonna end, like, after Wednesday or after the end of the week or after you take the final depending on how you look at it. Don't be a stranger. Right. Come on by. If you want to talk computer science, if you want to talk about what's possible to do in the field, come by. My office hours will be on the web or send me an email to set up a time to talk, and I'd be more than happy to take you through a bunch of this stuff. So besides, Dave, there's me.

Now, last but not least, and I shouldn't say, last but not least, you might say, Miron, computer science is kind of interesting, but are there other related majors that I should consider. So in the sense of full disclosure on fair play, there's computer science, there's some other possibilities. There's electrical engineering if you're more interested in the hardware side of things. There's math and computational science. And math and computational science is more if you're interested in the mathematical side of computing. You'll still get a lot of math if you do computer science major, but if you sort of are really kind of immersed on the mathematical side, math computational science is something to consider. And there's also a major called symbolic systems or just sym sis. And sym sis is also a fun major. It's actually a combination of linguistics, computer science, philosophy and psychology. I always forget the last one. Except, it's always different every time which one I forget. And the basic idea here is to think of both humans and machines are symbol processors, right? People are symbol processors, in some sense, because they take in symbols of the world, namely, language or visual [inaudible] that they actually see and they make some sense of it, and then they act in the world.

But now you might say, okay, that's interesting. You've told me about all these fields, but you told me that computer science was more than just programming and so far, it's unclear what I might be doing other than programming all these cool application you're showing me. So let me tell you about a few of them. This is the one I refer to as kind of the peanut butter cup version of computer science, which is you can take computer science and a whole bunch of other things and mix them together and they're just two great tastes that taste great together, and I'll show a lot of examples of that. So there's CS and business. Okay. If you're interested in sort of the business side of thing, product management is a whole field or a whole area that people go into, especially in high tech

product management which are people who don't necessarily program, but they have technical backgrounds to be able to define what products are going to do and how people are gonna interact with them. So if you look at a lot of high tech companies, people who are product managers, who are taking more of a managerial role and defining a role for product, many of them, in some sense, I'd actually say most of them, probably have a technical background. In a lot of cases, it's computer science even though they do know programming. They do product definition.

Beyond that, and this is kind of a popular one around here. Entrepreneurship. Yeah. That's good enough. I always get nervous writing that. And that's the whole notion of you think about people who are doing startup companies. There's been a ton of startup companies. I can't name them all because over the last few years, there's been over 2,500 companies that have come out of Stanford. Some of them are big and you know about, like, Google and Yahoo and Cisco and Sun and HP and all these other ones, and there's a whole bunch of smaller ones out there that also did pretty well. Anyone ever remember Evite? Anyone ever send an Evite? Yeah, that was started by a guy I lived next door to many years ago. And they did pretty well. It got acquired eventually, but life was all good. And the whole notion here of thinking about startup companies – now, one thing that's interesting is a lot of people think, "Oh, well, if I want to do entrepreneurship, I should go do business, right?" Well, what I'd actually challenge you to do, if you think that, is go find out about the backgrounds of people who are things like successful venture capitalists and see what they did when they started.

And one of the things that you'll actually find, which is surprising, is most of these people didn't start as business people, they started as technical people who actually went and did interesting technical work and at a certain point, realized there was a need and then moved on into the business realm. Tons of examples of that. I'll just give you a quick one. Eric Schmidt, who happens to be the CEO of Google, PhD in computer science. Right, now an MBA. And that's not to say an MBA is a bad route. It's just to say that, realistically, if you look at what a lot of people have done, the route to actually getting there, in many cases, actually, flows through a technical area. Okay. There's also finance, in the sense of computational finance. All right. Again, not only in predicting the stock market, but there's a whole bunch of people that what they do is they worry about different kinds of modeling algorithms or managing different kinds of funds, basically, by thinking of financial markets as a computational problem that they model with different kinds of data structures and different sorts of algorithms to potentially make predications on or just to get insight into. If you're interested in this kind of stuff, there's actually a program called the Mayfield Fellow's Program. If you do a search for Mayfield Fellow's Stanford, in your favorite search engine, you can find out more about it, which is actually a program that you learn about entrepreneurship. You go into an internship with a startup company to learn more about it, but you actually get immersed in thinking about the different issues of starting a company. We'll just leave the CS up here.

And biology. This has become a hugely popular area these days. Okay. So there's a whole bunch of things like bioinformatics, and bioinformatics – there's kind of different flavors of CS and biology, is thinking about the information systems that keep track of

biological data, or they keep track of medical data. Right. So if you think about if there's a whole bunch of medical data that's being kept on, like, your medical records and results for tests and a whole bunch of things that I want to be able to slice and dice in different ways, or understand how, for example, symptoms that you have might be related to some other symptoms or some other diagnosis that happened in the past. These are the kinds of information systems that deal with that, and we have a whole program here called the BMI program, Biomedical Informatics that just deals with that. But beyond that, there's also fun things, like, genomics and proteomics, and doing things like being able to look at gene expression data and DNA and be able to determine what kind of diseases do you hereditarily have more of a disposition to because of your genetic makeup. And if you're interested in this kind of stuff, there's actually a program also on bioengineering. They don't, right now, have an undergraduate program. They have a graduate program. They're gonna form an undergraduate program. That's something you could be interested in, or it's something we actually have sort of a sub areas of the computer science major that you can also do this sort of stuff in.

Now, one thing that's kind of interesting, which also sometimes surprises people is they say, "Oh, I want to be a patent lawyer. I want to go and deal with all these issues like making sure that file copying of music is legal for everyone, so I'm gonna go and be a political science major cause that's what I should do to go to law school." Right. Turns out that if you actually want to be an intellectual property lawyer, you need to have a technical background. There's a list of approved areas that you could've done for your undergraduate degree that allow you to become an intellectual property or copyright lawyer. Computer science got added to that list about 15 years ago. Political science, not on that list. Okay. So it's something you should probably know now. If this is the area that you're thinking about going to, you need to understand the technology to understand how intellectual property and copyright issues apply. You need to understand what an algorithm is. What parts of an algorithm are obvious versus what parts of an algorithm are not obvious? That's what allows people to do this work. Okay.

And then last, but not least, CS plus CS. So you can just do – you don't have to mix computer science with something else. You can just do computer science, and obviously, programming is part of this. There's a lot of people who are very happy being software engineers and there's lots of jobs in software engineering and life is good. But there's also people who go into engineering management. Most managers, in computer science, are not professional managers. They are people who at one time were programmers or engineers, and worked their way up through the ranks and eventually became managers and became senior managers and became VPs and the whole deal. Right. So it started by having a technical background. It didn't start by saying, "Hey, I want to be a manager," and having someone hire you to be a manager. Okay. And there's also, and this is near and dear to my heart, so I'm just gonna sort of wrap up quickly, teaching. Right. So you could think about computer science as a field that you go into because you want to teach it to other people, in addition, to perhaps doing some stuff in it yourself because you find it interesting, but if teaching at all is something that's interesting to you or, like, when you were in your section, you were, like, "Hey, section leading is kind of cool, this is

something I might consider," there's the cs198 program. And this is a program that I've talked about in the past, but I just want to spend a little bit more time talking about.

And what you need to go into cs198 is you need cs106a and b, or you could've taken X, but at this point, it's kind of too late to take X. So what you really need is cs106b, after one more class, you're eligible to become a section leader. And being a section leader, you might say, "Oh, well, what does that involve?" And it turns out to involve a whole bunch of things. One is that you actually teach a section, which is kind of cool in itself because you get to learn the material a whole lot better when you teach it to someone else. There's always some new little nuance about something that you learn somewhere. So you learn the material that much better by teaching the section. You also get to know other section leaders. So there's kind of a social aspect to it, and especially if you want to go into computer science, this is a great way to meet project partners and other people who you know are really interested in computer science and motivated. You also get to meet faculty. So when it comes time for getting letters of recommendations, which is something that people don't necessarily think about early on in their program, but then later on how many people are thinking about letters of recommendations now, like, it's getting to that grad school application time, and how many people wish you had thought about it earlier. Yeah. Mostly the same hands.

This is a good way to do that. Is to actually get to know people who are involved in teaching and we have regular staff meetings, and it's a good way to sort of think about that. And at the same time, and one last side point I would put in, is that there's a huge network of people who went through this program who are out there. So the cs198 program is actually a program that's not just known at Stanford, but it's actually known nationally. Like, if I go to other companies or something like that, there's people that come to visit, for example, from Microsoft and they're, like, "Yeah, tell me where the 198 meeting is, and what's going on there," and they come and recruit from that group of people, and this happens for a whole bunch of companies across the board. So, with that kind of said, hopefully this has given you a little bit of a taste for what computer science can be about. Not just thinking about programming per se, which is what we've done a lot of in this class, but programming is really just the first step that opens up a whole bunch of other venues. And hopefully you got a sense of some of the other classes you can take that will broaden your horizons even more and some of the different areas you can go into potentially with a computer science or related major that can open up all these possibilities. Any questions about any of that? You're all set? All right. Then I will see you on Wednesday.

[End of Audio]

Duration: 47 minutes