Hello, everyone! Welcome to another episode of Q&A about business, innovation, and managing life.
And I see a bunch of, … Questions… Saved up here…
One from Lana. How much of what you do now came from formal education versus self-learning?
Well, I'm an old guy, so the… just… the accumulation of years is vastly on the side of self-learning. But also, I personally…
Well, let's see. I kind of haven't been to kind of a formal class
about anything, pretty much, since I was 16 years old, which is half a century ago now. …
So, I've had an awful lot of time for self-learning, and not so much time for non-self-learning. If I ask myself, what things do I do today that I learned, sort of, from formal classes versus elsewhere.
I would say…
Well, I mean, I kind of learned basic math in formal classes long, long ago, and I use something that is a big tower on top of that today. I did learn writing, like English writing and so on.
from, sort of, formal classes back in the day, and I certainly do that every day. I think
In what I've done with writing, for example, I learned a certain style of writing when I was in school.
And, …
Then in my life, I've adopted a sequence of other styles of writing, like academic papers are one style, like writing software documentation, another style, like writing business communications, another style, like writing, sort of, blog-like writing pieces, another style. Each time, it's sort of been a bit of relearning of how I want to do things that I suppose is self-teaching.
More than anything else. As far as other things are concerned that I kind of formally learnt, I… I mostly learned, for example, science kinds of things myself from books, not from kind of formal classes and so on.
So, at least for me, the vast, vast majority of what I make use of on a daily basis is things that I've learned without not informal classes. How did I learn these things?
I have a decently good memory, and so it helps that, you know, whenever I sort of run into something, so long as I'm kind of primed to be interested in it or remember it, I'll remember it, and still, you know, half a century later, I still remember it, and that's really useful. I think it's useful to have these frameworks into which you can put things that you learn. I think when I start hearing about things where I really have
no framework.
for sort of fitting in the things I'm hearing about. If it's some sort of advanced, abstract area of math about which I know nothing, and somebody's telling me fascinating facts in that area, it's hard for me to fit them into anything, and I probably won't remember them.
If it's something… similarly, if it's about some industry about which I know very little, it'll again be very hard for me to fit that… fit those things into what I'm hearing. But I think, for me, the things that I tend to, to, …
… to… I've tended to try and learn things myself. I learn a lot by doing.
I would say that a large fraction of the things that I know now, I know because I did a project related to those things at some point in the past. For me, that's always a much better way to learn than to sit down and say, you know, I'm going to learn this subject, let me go read books about it, or watch videos about it, or whatever else. I've been, … I don't find that works very well for me.
I kind of have

to have something that I'm sort of already thinking about, some project, some purpose that I have, then I can learn things and fit them into things that I care about.
I think, sort of the vision
that you kind of learn everything you need to learn in school, and then you're sort of free-running for the rest of your life based on what you learned in school. It's certainly not a great vision for modern times. A lot of things in the world change, and so you kind of have to learn new things.
Now, if I ask the question, things that I learned
Back in the day, am I still kind of rerunning those same things that I learned, or not?
When it comes to various technical areas.
For example, I don't know, things in, let's say, math, even things in physics, those fields, they move, but they don't move that much. And the things I learned 50 years ago are still sort of perfectly valid things that I can make use of today.
When it comes to,
things, let's say, with computers, well, the things I learned by the end of the 1970s of, you know, how to type things into a Unix shell, for example, those are things that occasionally I'll still use today.
And there are things where, you know, knowing how to use grep or something is something that, you know, I learned
in the late 1970s, a really long time ago, and it's something I can still learn today. I didn't learn it in a formal class, but I learned it a long time ago. It's still sort of valid today. Many other things about using computers, for example, have changed completely, and sort of one has to, you know, learn them afresh and so on in modern times.
I would say, in terms of, for myself, there are a lot of domains that I've become involved with over the years that I just didn't know anything about when I was, quotes, in school, and I just… I wasn't interested in them at the time, I didn't know anything about them, I've learned them subsequently.
when it comes to, for example, things in business, that's mostly… that's really all learnt by doing. I remember when I was first back, oh gosh, now, 45 years ago, when I was first, my first company, I was kind of like, well, maybe I should learn things about managing companies from books.
at the time, the books… now, admittedly, these were books from a university library, so who knows, maybe they were badly selected, but, the books that I found about management were absurdly bad. And, I'm sure the modern ones are better, I haven't looked at them, but these were, were, sort of, horrifically, you know, not… not useful.
…
And, so I didn't read them, so whatever I learned about doing business, I learned by doing it, so to speak. I mean, for myself, I happened to have experience for a while doing strategy consulting for tech companies, sort of between the time when I did my first company, beginning of the 80s, and the time when I did my current company, starting in the mid to late 80s.
And that… that particular experience of seeing a bunch of different companies and, seeing sort of what was happening in them was… was very useful to me, and I think that, it was,
No doubt, if you go to business school and you do case studies and so on, you might get some of the same exposure. For me, it was kind of a real-life exposure, because it's like, here's a company, here's what it's doing, it's like, what should it do next, type thing, or what, you know, how should you think about the strategy of what it's doing?

I've kind of, in more recent times, I've… I've done a bunch of that. It's one of my sort of hobbies, is doing strategy consulting for tech companies now. It's very… always very interesting. I guess I pick interesting companies, and something that I find very educational, so to speak, but certainly it's not a traditional form of education, so to speak. I think one of the things that people sometimes imagine is
I go to school, I learn what I need to learn, and then I don't need to… you know, I'm finished with learning.
This is obviously a terrible mistake.
And I kind of suspect that anybody who's watching this livestream doesn't have the point of view that they stopped learning at the point where they left school, so to speak. Because I think that not only does the world change, but also one's own interests change, and sort of the opportunities one might have change, and there's always new things to learn.
Let's see… … Sylvia asks, how many patents do you have?
I don't know. That's a good question. I'm sure it's easy to look up on the web. They stretch back into the 1980s. I have to say, I'm not a great
I wouldn't say that… I don't think any patent I have, I've ever gotten any real benefit from.
And, …
I think they're kind of, things where one feels, yeah, you know, I invented this thing, and, you know, it's kind of like a cool thing, and I should get a patent on it. And sometimes I feel like, oh, you know, somebody… it's good to have patents in an area, because at least that establishes you invented it, and if somebody comes along and says, no, no, no, I invented it instead, you can say, no, no, actually.
I got this patent in this area, so don't, don't hassle me about your… your version of what happened in this area. But I have to say, it's, it's been a thing where, generally.
In what we do as a company, it's like we have software, there are some sort of trade secrets about how it works inside, but what you see on the outside, it's not, … there are occasionally things that seem like they're worth patenting, that are
things about the way the software works on the outside. As I say, I don't believe we've ever gotten any substantial benefit, possibly some defensive benefit, from these things, but I don't think we've, we've ever gone and sort of said to somebody, oh, no, no, you're doing something that… that violates our patent. Maybe they never have.
But, that's, it's not been… it's not been something where, we… where a lot of value, has been derived from… from patents. That doesn't mean the patents don't have value, it's just we haven't been able to derive it from them.
Let's see… … Question from Caleb, is memorization so valuable in an age where knowledge is instantly searchable?
Well… It's absolutely useful to know a lot of stuff.
Is it useful to memorize, these are the amino acids, these are the whatever? Yeah, I mean, I think it's still somewhat useful, because the fact is, what you've memorized, you can integrate into your thought process. What you have to go look up is sort of an external thing.
I mean, it's, … we kind of have an example of this, you know, we're slowly seeing this work better and better. LLMs can sort of know a certain amount of stuff intrinsically, and then they go use tools like our tech stack to go figure things out.
And it's sort of a break in the thought
to go and use the tool. It's pretty necessary for an LLM, as for a human, to go use the tool to actually get the right answer. But I think if you have to use that in a very fine-grained level every

moment that you're thinking about things, it's a mistake. The other point is that there's sort of a discovery interface in the computer sense version of, if you know things intrinsically, then you can know what's there to know, so to speak.
And that can be something that you integrate into the way you're thinking about things. If, for example, you know, you're thinking about something and you wonder, you know, is this a thing that is knowable or not? Well, in principle, you'd go search and find out if it's knowable and so on. But if you can immediately know that it isn't knowable, for example, or that is knowable, you can integrate that into your thought process.
I think it's really… knowing things is super important to being able to think well. You know, you might imagine
that, … well, I would say that there are things which are, sort of
principles to understand, and there are facts to know. You might say, learn only the principles, don't learn the facts. I don't think that works. I think that the principles are sort of laid on top of facts, and you really need to know those facts. I mean, if you…
were learning things about, I don't know, biology or biomedicine or something like this, and you just didn't know certain facts.
about, oh, there are these ribosomes, and they work this way, and they deal with these kinds of genetic codes and so on, and there are triples in the, you know, in the base pairs relative to the amino acids, and so on. If you didn't have that sort of framework of knowledge, then
then, you know, even learning some sort of general principle about how things work would… it just wouldn't… just wouldn't work very well. I think… so, I'm a great believer in, you know, there's sort of a base of facts that are worth knowing.
Then on top of that, you can layer, sort of, principles to learn, and then with those principles, you have much more leverage, and you can figure out a lot more things.
and new facts that come in fit in with those principles. But I think it's a mistake. I mean, there have been various theories in the history of education that, oh, facts aren't worth having anymore, everybody can just go look up the facts. I think this is a mistake.
Now, you know, how much is worth memorizing versus sort of cold memorization? I'm… I… I haven't done cold memorization for a long time. It's, … that feels very, painful when one's doing it, but
in many cases, it's awfully useful to have done it. I mean, I don't know, when I've, you know, I've learned… well, I… millions of years ago, so to speak, I, like, learned the Greek alphabet. That was, you know, I learned a bunch of ancient Greek, too, but… but that's, you know, it's a basic thing to know, and I had to sit down and learn it at some point.
And, you know, I, I, … Now, I have to say.
at times when I was a kid, learning, sort of trying to cold learn facts of history, like the date of the, you know, Battle of Hastings or something. Okay, every British school kid knows that, it's 1066. But, you know, I have to say, I had a very hard time cold learning. I'm sure I have a good memory, and I'm sure, you know, I probably
did well in the test of, you know, when was the battle of such and such. …
But I found it a not very useful exercise, and I didn't remember most of those things. In subsequent times, there are things where I've made sort of an effort to remember certain anchor facts and so on, like a good date to know, 1642, the day, the date Newton was born and Galileo died. So, that's a relevant date in the history of science, so to speak.

in a bunch of things that happened in an important period in science. But, you know, that's something that eventually I just sort of learnt one day. I would say that there are things that one sort of learns by osmosis, like
an oldie like me knows American area codes. Like, I know that, you know, area code 212 is New York 312 Chicago, you know, I don't know, 415 San Francisco, things like this. It's sort of useless, but once you know that, it's kind of interesting when somebody gives you their cell phone number, and it has, you know, a, I don't know, an…
an area code 202 prefix or something, you immediately know, oh, they come from, you know, the Washington, D.C. area, or whatever it is. And, that's sort of a mildly interesting thing to know, that knowing that fact lets you, lets you get. But I didn't, you know, I certainly haven't… I certainly never back in the day went and tried to memorize area codes.
It's just that, one hears them enough times that one eventually remembers them.
Let's see… …
Elsie asks about learning. Do you still learn from textbooks? And if so, do you read them linearly, or go directly to the bits of immediate relevance to a current pursuit or project? I am horrible at reading them linearly. I have to say, I tend to try and extract knowledge
where… of things I specifically need. Now, I have discovered that when you're trying to get some general idea about some field.
that finding, sort of, an anchor textbook in that field, typically written by a major practitioner in the field, more so than a pure educator in the field, although there are probably exceptions to that. But find a book by, sort of, a major practitioner and read the introduction, because it's often a good piece of, kind of, orientation for how people think about things in that field.
Because every field has a different way of thinking about things. And sometimes, you know, you jump into these fields, and you start seeing, you know, these are… it's kind of like there's an assumption about, kind of, the types of problems you're solving, and the way you're talking about those problems, that if you don't have that orientation, it's pretty hard to understand what's going on, or why one cares about things, and so on.
But I'm… I tend to be very much a extract information. I'm… I'm not a patient reader from beginning to end. And, you know, occasionally I'll find a case where… where I start reading it, and I realize either I don't know enough, and I have to kind of go from the beginning, or sometimes I'm just like, this is actually a really well-written thing, I'm actually going to try and, Do my best to read through it from the beginning.
Let's see…
Elsie is asking about patents. Do you find value from patent writing as a forcing function of some kind about formalizing and generalizing things? I haven't really found that, partly because I'm often doing patents with other inventors.
the company and so on, and they are usually the ones who are interacting with the attorneys and such like, and so I'm not usually the one who's at the front lines of, crispening up what's being said. Although I can remember a few cases where, sort of.
Being pushed on, well, what kinds of things is this really about?
caused me to have a bunch of new ideas, which probably made it into the patent, but I'm more interested in just the ideas themselves.
Let's see …
Jamie is asking, if you were designing a course called How to Think Computationally, what would the first assignment be?

Well, I have to admit that about a year and a half ago, I started actually writing a book called Introduction to Computational Thinking, and I am about 4 chapters into it, and I really have to get back to it and finish it. But I can tell you that it's actually findable on the web somewhere. not advertised at this point, but it's… it's open out on the web, and probably somebody can find a link to it. …
the way I started, I had sort of an introductory … chapter
that's sort of what… what is the idea of computational thinking? What is this idea of formalizing the world in computational terms? What kind of thing does that involve?
And then I had a chapter that's sort of an example of thinking computationally about things, and … the, …
The example that I ended up using was names of numbers.
So, in English, you know, it's 1, O-N-E, 2, TWO, etc. And in Waltham language, it's… it's trivial to generate all those names of numbers in many different languages, and it's kind of an interesting exercise in sort of data science and what you can compute. Just, for example, plot. The length of the name of every number
for every number up to 100, let's say. And you'll find there are peaks, like, you know, 67 is a big, long peak, but, something like, I don't know, 60, well, that's… that's the… the every 10, it's going to be a dip, and so on. And you can start doing that in more generality. You can look at, you know, let's generate audio for those things, how long is the audio?
Let's, …
compare different languages, which language is sort of most compact at doing these things. It's just an example, but it's an example that allows one to start sort of wondering what kinds of questions can one ask when one kind of thinks about the world computationally, so to speak. I would say one thing that, … I don't know why I picked this question years ago, and I really need to think of others like it.
Sort of a basic, what's the difference between computational thinking and computer science? And, let me give a typical, sort of, computational thinking type of question.
It's a real question that came up for us in building Wolfman Alpha. You're going to make a map. You're given a lat long. You're asked to figure out how big should the map be for that lat long on the surface of the Earth.
If the lat long lands in the middle of the Pacific Ocean, showing, you know, a 10-mile radius around that point is not very interesting. It'll just be blue.
you probably want to show a 1,000 mile or more radius around that point. If that point lands in the center of Manhattan, you probably want to show, you know, a mile or two at most.
Around that point. So the question is, how do you decide what that radius of how far you show, given a point on the Earth, should be?
And you can start thinking, well, how would I compute that? What would I do? You can start thinking, well, let me retrieve the map, try and see how many features there are in the map at a certain scale. One thing you could do, you could ask, you know, how many people live within the radius that you're going to say. I want something where
I'm going to have, some reasonable number of people who live within that region. Or you can think of lots of other criteria for, you know, how you decide what that, what that scale size should be.
It's not a question that is a sort of traditional computer science question where, well, the field of computer science is a complicated one in terms of what it's really meant. I mean, it… back in the day, it meant a bunch of sort of theoretical algorithm kinds of things. How do you do sorting and

searching and building compilers and things like that? That's what computer science as a thing you would do in college meant.
up through, I think, pretty much through the 1980s.
And then there was sort of this separate practical discipline of software engineering that had to do with how do you write code, how do you organize big systems, how do you think about source control, how do you do all these kinds of things to do with sort of managing the development of systems?
And sometime, I think, in the 90s, there started to be a big demand for people who wanted to become software engineers, and so universities dramatically scaled up their computer science operations. Probably they overshot a bit, because at this point, there are, you know, huge computer science operations at many universities, and it's not… and what they are mostly teaching, in the end
Is how to program.
And what is becoming clear, I mean, people like me, for decades, have been trying to automate what computers do to the point where a lot of that detailed, low-level programming is just completely unnecessary.
And that's being re-emphasized as we see, kind of, AI-generated programming, some of which I think is quite unnecessary, because the truth is that the thing that you say, write, okay, LLM, write out this piece of code.
That big blob of code that the LLM might or might not get right is probably just one function in Wolfram language.
we already automated that as a kind of a well-designed thing, rather than you having to say to the LLM, try and reproduce that thing that's roughly this. But in any case, whatever the reasons, kind of the level of automation of programming and the systems for programming is increasing to the point where, sort of, the need for kind of just, oh, I can write
Basic code is going down.
But there's still… I mean, there has been a sort of traditional belief that this is one of the, in the last, probably 20 years or so, this is one of the sort of the career tracks, is become a programmer, go into the tech industry, and, you know, it's a lifelong career type thing, as it would be if you were an accountant, or a lawyer, or a doctor, or whatever else.
I don't know whether that's such a good bet, because I think that the… the amount of automation there is, is making that… there not be as need… need for as much, kind of, manual programming, as, as there has been. And I think,
the, … I mean, that happens for many reasons. I mean, whether it's, that, you know, instead of having to write raw HTML for your website, you can use a good website-building tool.
Or whether you're, you know, doing something where you're programming something algorithmic and it's just one Wolfram language function. Or whether you're creating, you know, a user interface where you use more and more frameworks for doing that, and so on.
I think the, there's this question, then, of what will happen
To this big build-out that's happened in computer science education, for example.
What should it do? What will it do? I mean, I suspect, and I think this has already happened in recent… very recent times, that the… the number of, kind of, jobs for people, sort of, graduating from computer science degrees is going… going down, and, the, …
That will presumably have an effect on the number of people who are going in to study computer science, but yet universities have tended to expand dramatically in terms of the professors and so on that they have in those areas.

I think it is sort of…
well, there's a question of, sort of, what should computer science be, and then what should computational X be, and how does it… is it distinguished from computer science? I mean, I have thought for a number of years that just as one might teach a CS101 class, there really is a CX101 class, which is really about computational thinking.
not about
quotes computer science. Now, when I say quotes computer science, in practice, that often means programming in some low-level language.
That's… well, okay. It often means that at a certain level of university. At a lower level, it can mean learn to create a website, learn to use basic
productivity office-type tools. I mean, that is something people sometimes call computer science. What the academic discipline of computer science
traditionally was, was sort of the theory of how computers… how you can do things with computers, and the theory of algorithms, data structures, things like this. Not sort of the practicalities of engineering programs, and certainly not, sort of, the experience of using application programs as application programs, and so on.
is…
there… there is a strand in most university… high-end university computer science departments, there is a strand of, sort of, theoretical computer science, which is somewhat close to mathematics. There are… it's a fascinating field. It's mostly not that relevant.
to the practicality of programming and software engineering, as I say, I think it's a fascinating field. It's important for the future, it's been important in the past, but it is something quite separate from the practicalities of things.
But what's sort of missing from… then in computer science departments, there are some add-ons that have been sort of inserted over the years. I mean, there's… there's cryptography.
machine learning to some extent, then things like robotics are sometimes sort of an add-on in the computer science department. Sometimes some things closer to sort of the systems engineering side of things, of, things about, sort of, the hardware of computers, things which might be in other places, and sometimes more closer to electrical engineering, and so on.
But, you know, machine learning has thrown a curious kind of, kind of, I don't know, sort of spinning thing into the whole area of computer science, because to
computer science has been becoming less and less mathematical, perhaps for good reason. I mean, it's not really necessary to learn calculus to write code that kind of does some basic IT functionality.
So, and that, it might be more interesting to learn statistics or something like this for that purpose, but it's probably not that important to learn how to sort of figure out an integral algebraically by hand. It's not going to come up in doing basic database programming or something of this kind. So there's been sort of a migration of computer science departments away from these more mathy things.
And now, along comes machine learning. Machine learning is quite mathy. If you try and understand it at a foundational level at all.
It's kind of a great application of multivariate calculus. Now, the next question is, do you need to understand it at that level? It's a weird level, actually, to understand machine learning at.
Because when it comes to the really foundational questions of why does machine learning work. Those are questions nobody knows the answer to. I've worked a bunch on those questions, I have a bunch of scientific things to say about them, but I would say that the kind of science that's

needed to answer those foundational questions is yet a different thing from the kind of thing you get from something like multivariate calculus.
There's a layer of thinking about machine learning where things like multivariate calculus are relevant.
Then there's a layer of, it's just a bunch of black boxes, let's use them. The truth is that the open it up and think about multivariate calculus is becoming less and less relevant in practical machine learning.
It's the same type of thing as saying, well, back in the day, in computer science, everybody would learn how to write a compiler.
But the truth is, the number of people who write compilers in the world is unbelievably small. Even compared to the number of people who use, explicitly or implicitly, compilers. So knowing how to write a compiler is not that important of a thing. Knowing how to create some new activation function with different kinds of layers and back-propagate through it and so on in machine learning is also becoming a deeply specialized thing.
And when it comes to practical systems that are deployed, you're really just dealing with, there's this black box, let me put it to, you know, I've got a big LLM, I'm just gonna do things with the LLM. It's about how do you use that thing as a tool?
So, the idea of machine learning as a kind of an academic-type subject gets pretty weird pretty quickly. There have been a lot of trends, they go up and down as the months go by, of sort of poking at machine learning to try and sort of make it academic, so to speak.
And it's… it's been pretty tough, I would say. I mean, it's… I would say that it's something where It's more reminiscent of, kind of, you know, people are sort of hoping for a theory, just as they might hope for a theory of neuroscience. But the fact is, most of what is done in neuroscience is sort of descriptive science, empirical descriptive science.
And one could do that for machine learning as well. It feels a bit weird to do it in that case, because it's on a thing that is much more, kind of, transitory than our brains, which, you know, have been the same for the time of our species, so far as we can tell, whereas, kind of like, I study this particular LM,
oh, actually, somebody retrained it and made a new LLM. Oops, you know, I've got to rethink or made a new architecture, slightly new architecture for the LLM. I've got to redo all my analysis type thing.
It's, … so…
it's a sort of complicated area that's… that has a weird interaction with computer science, because, you know, there are endless papers people write where they're doing what amount to psychology experiments on things like LLMs, saying, you know, does it manage to do this? How does it think about that?
Some of that, if it's done
well enough will be interesting. It's pretty hard to do it well.
I think… and I think that the, … so, you know, how that really will intersect with computer science is still, I think, a little bit unclear.
But the question then is, well, what about this computational thinking stuff?
Well, that's really important in a zillion different fields, because what that's about is taking the great formalism that we now have in the world, namely thinking about things in terms of computation, and applying it to everything. In the past.
Great success was had by applying mathematics as a formalism to the places where that was successfully applied, particularly in the physical sciences, in a few other areas.

But computation is a much more broadly applicable thing, and the question is, how do you learn how to apply it? If you're… if you're thrown into, I don't know, doing forestry, or doing, archaeology, or doing, …
some… some… lots of different fields. How can you use computational thinking in those fields? How can you use, kind of, this way of structuring your thinking
in computational terms in those fields. And, of course, the great thing about computational thinking is once you've structured your thinking in a computational way, you can immediately get… well, if you have the right tools, at least you can get a computer to sort of amplify your thinking, to work through the consequences of whatever you've formulated. I mean, my big goal with Waltham Language over the last, I don't know, 40 years or so, has been to build that kind of computational notation for talking about the world.
A sort of notation in which you can formalize the things you're thinking about in computational terms
And then the practical system and language to go take that formalization and work out the consequences of it, and be able to sort of deploy what you're doing.
And it's a tremendously powerful thing, which I think is still very… it's understood in corners of the world, but it is not nearly as broadly understood as it should be. And that's, I think, the real power for the future, is sort of formulating things in all areas, computationally.
then…
then getting the computer to assist you in working with that stuff. So now the question is, well, how do you learn to think computationally about things? Well, I think one thing that helps is seeing examples of how you think computationally about a bunch of different domains, whether that's about things about geography, whether that's about things about, sort of optimizing things, whether that's things about
graphical presentation, whether that's things about, sort of, human perception. There are all sorts of different areas, whether they're things about, sort of, data about the world, and so on. There are all sorts of areas which you can think computationally about.
And learning how to do that is… is, I think, an incredibly powerful thing. It's not a thing that's being taught right now.
It's CX101, not CS101, but there isn't yet a CX101.
that's been well-defined. I think it's sort of a… probably a responsibility for me to try and do this, because I've sort of worked on this whole idea of computational thinking for such a long time, and have built sort of the primary tool that I think is powerful to use for that. And so it's probably incumbent on me to write such a thing, and I wish I had more time to do it.
But in any case, the kind of… the goal there is to take whatever you're thinking about.
And know how to take that thing Formalize it computationally.
And then, hopefully, it's pretty easy to go from your computational formalization that you have in your mind to the Wolfram language code and get it run.
the notebook assistant that we have now that's based on a lens and with a whole bunch of, kind of, computational assistance.
is, is a pretty decent way of taking that vague thing you're thinking about and having that turned into, sort of, the closest piece of precise computational formalization in our computational language, in Wharton language.
that sort of the closest approach to that that you can imagine. It does surprisingly well at that.
And I think that's one way to, sort of, one part of the on-ramp to kind of computational thinking.

But there's certainly a lot of basic things to know to help one to think computationally about things. Whether that's where, kind of, the big, sort of, computer science… well, let me make the following point. I mean, you know, if you go to computer science school, so to speak.
and you go to it, to learn how to do programming, that's one thing. Just like you might go to law school to learn how to be a lawyer.
you could go to law school just for interest when you're thinking about becoming an archaeologist or something. Maybe there's legal relevance in doing urban archaeology or something, but… but the point is, people don't generally go to those… to law school, for example, for the purpose of just sort of general interest, to apply, kind of, legal thinking in lots of other areas.
they'll typically go there because they want to be a lawyer. And in computer science schools, so to speak, people right now, I think, are mostly thinking they're doing that because they want to be programmers and so on.
I would say that, it's people, they're… you know, the idea of liberal arts education, for example, is you learn stuff just because it helps you think better about everything, not because you want to be a professional philosopher when you do those philosophy classes, or whatever else.
And I think there is a version of computer science which is more this computational thinking thing, which is more like liberal arts. It's more like
The computational formalization of things is part of the sort of the canon of modern knowledge and modern thinking, and a thing that's worth people getting, even if they're not going to go on and be programmers.
And I think some people sort of get that point and sort of take computer science classes for the sake of sort of understanding something about computers. What they often get in those classes are things that are very inappropriate, I would say, in the wrong direction for getting that broad understanding of the idea of computation and computational thinking. That's something that I think still has to be built in the future.
Maybe some of the kind of exodus of actual programming jobs, for example, will stimulate more of that trend towards sort of teach computational thinking, not computer science and programming, so to speak.
And I'd like to think that the tools we built and, lots of the things around them are really sort of prime, sort of anchor, things to be… to be leading that effort in moving towards computational thinking away from sort of raw computer science and raw programming.
Let's see… …
Boy, so many questions here. Lots of interesting questions.
…
Hilzer is asking about my kids. Do my kids do computational thinking more naturally from osmosis for me, or direct instruction? I would say I have 4 kids. Three of them, I would say, are pretty serious computational thinkers, and … I don't think
… at best, they've learned things by osmosis from me. I would say that, …
It's, … it's interesting, because somehow, …
I don't know whether… I've been surprised by how naturally
they seem to manage to use Wolfram language, for example, and seem to understand fairly complicated thing… abstract things there. I have no idea whether that's an apple falls close to the tree, whether that's an osmosis effect. It is almost never a, let me directly explain to you how to do this type thing.
And I can think of only a very…

Fairly small number of examples of that.
So, I don't know, you can go look up their websites, and you can see their, their various, computational thinking efforts and so on.
Let's see… … Have a very… Long question here from Carlos.
Let's see if I can parse this out.
It says, I was an undergraduate physics student.
Ready to finish my bachelor's degree in 2020.
… And saying they've now been working as a data analyst.
But their true passion is studying physics and becoming a researcher.
And, … wanting to continue their science degree, finishing science career, finishing degrees, and so on.
Let's see. The question is really, if you're 29 years old, can you pivot from being a practical data analyst and so on, to become a science researcher?
Absolutely, yes.
I mean, I think the reason you see less pivoting of people later in life has to do with just the practicalities. If somebody says, well, I've, you know, I'm doing this job, I'm making a good living here, I've got a mortgage, I've, you know, whatever, I can't go back to school and start you know, making no money and just go learn things for a few years. If you're in a position to do that, there is nothing, I think, to be about kind of, oh, you know, you're too old, your brain cells have been… have decayed to the point where you'll never be able to learn these things. It's really often a matter of personal circumstances.
And sometimes a matter of, sort of, personal motivation. I mean, I think it… sometimes, there is, … the best… I will say that the best time to go to school and learn things is when you're really enthusiastic about going to school and learning things.
Unfortunately, most people get to go to school at times when they're not necessarily that enthusiastic about learning things. I mean, the people who take a couple of years off, and then do this, and then do that, and they know why they're going to school, and they're very enthusiastic about what they're learning, because they're there not because they sort of have to be there, or because that's the natural thing that you do when you're 18 years old, or whatever else.
But because they're like, well, I decided I'm gonna do this. And then, you know, they'll have a much better experience than if they're just doing it as a matter of, well, that's just the way the path leads right now.
I mean, I think that's… so I would say that it's a… it's really a win. The only challenge tends to be a matter of personal circumstances, and just having locked into certain aspects of life, and then being like, oh, I can't break out of that to go into a more kind of student-y, less professionally kind of, kind of, direction. But yeah, I think I… I would say that it's a…
It's even a better situation, you know, if you're enthusiastic about studying something, then so long as you can practically do it, that's a great thing to go off and do.
Let's see… Sylvia is asking, should philosophy departments ramp up with regards to ethical risks of AI?
I have been telling a bunch of professional philosopher friends of mine that this is… that these times are a great moment, a great opportunity for philosophy. There are things that are coming up in the world, among them AI ethics.
Where a lot of the traditional questions of philosophy are becoming of practical importance. There are things which one could debate forever about how political philosophy might work, but in a situation where there really can be a change.

Or are those even potentially forced to be a change?
This is the moment when those are things that the people who've studied those things for a long time can really help and make a difference. So I think… I think it's kind of the, you know, the great opportunity hasn't been around for a few hundred years, actually, where sort of philosophy and the way that it's been done
can sort of, I think, make a difference. I think the challenge is, and I have friends who are in the situation of being chair people of philosophy departments and so on, and somebody told them, go hire a bunch of AI ethics people.
Problem is, where do you find them?
Because it's… it's something where… and what kind of background does such a person need to have?
Because it's… the… there probably will be a time…
when, sort of, AI is so… and the things about computation are so taken for granted in our culture that, sort of, having special education in those areas
is not important, not so important for doing philosophy in those areas. Maybe that's true, maybe it isn't true. I certainly know there's been a… for example, in philosophy of Science.
it…
is, I think, increasingly the case, that people really want to know the science to do philosophy of science, which is a good thing.
Although, if you know the science too much.
and you know kind of the contours of how the science works and how people talk about the science, you don't get to break out of it and do the kind of bigger thinking that is what philosophy should lead you to do. So, it's like you have to know enough of the science that you're kind of fluent in understanding what the issues are, but not so much that you're kind of drowning in the way that things are thought about, sort of, in the pure technical science.
And I'm sure the same is true in, kind of, the AI ethics, AI philosophy type area, that if you are immediately thinking, oh, that ethical question, I'm going to turn that into reinforcement learning that uses this particular learning policy and, you know, et cetera, et cetera, et cetera, I think you're going too deep into the weeds too quickly.
On the other hand, if you're, like, … if your view of the AI is it's just a piece of magic.
I don't think that will make you the best philosophy. So, for example, one thing that I think is pretty important in AI ethics is this idea of computational irreducibility that I talk about a lot, that even when you know the rules by which some computational system operates, the actual consequences of those rules you can really only find out by running the rules and seeing what happens.
So, why is that relevant for things like AI ethics? Well, in a sense, in an AI, you have a computational system where you might know the rules, you might not quite precisely know the rules, but let's say you know the rules.
it's still the case that to see what the system is going to do, you have to, like, let it just run and see what happens. If you say, well, I want the system to act ethically according to my particular code of ethics, and I want to constrain the system so it will never do the wrong thing.
then you can't have computational irreducibility. Computational irreducibility means there will always be unexpected things that happen that you can only find out by running the system and seeing what happens. If you insist on saying, I never want to be surprised in this particular kind of way, then you're forcing computational reducibility on the system.

So what? Well, if you force computational reducibility on the system, you're limiting the computational abilities of the system.
So you have this complicated trade-off between, sort of, the idea of… between letting the system do what is the maximum it can computationally do, and making the system act predictably in a certain way to you.
understanding something like that, that kind of trade-off, that does require a bit of, kind of, formal knowledge of… and thinking about things like computational irreducibility, which are getting integrated by now into, sort of, sort of the more science-oriented end of philosophy, I have to say.
But, it's taken 40 years or so, but it is definitely happening. But I think that the, …
the… the… so, you know, I don't think just AI is a magic is not a good start if you're going to do AI philosophy. And I think that it is a challenge right now to see, sort of, who fits in to this space of know enough about the technology
and know enough about the philosophy, and merge those two things together. I think it's a great opportunity for people. I think that there will be AI psychologists, AI philosophers, and so on, and I mean philosophers of AI, psychologists of AI, not AIs acting like psychologists. Well, maybe they'll get somewhere with that. AIs acting like philosophers, I'm not too hopeful about that one.
So, … …
Yes, it's a… it's a… it's a great opportunity for philosophy departments, I think, right now. I mean, one of the things that I find interesting is
philosophy, traditionally, is sort of this way of sort of thinking about things, teaching how to think about things. Learning philosophy is kind of a way of learning how to think in a certain pattern of thinking, of asking questions and kind of thinking what sort of logically follows
And not… not so much based on facts, but based on the structure of arguments and so on. It's a… it's a good kind of discipline of formalizing your thinking.
What I think is quite interesting is that computational thinking is surprisingly close to that. It happens to be a formalism that is as precise as mathematical formalism, but it is a formalism that you can kind of apply it as broadly as you can apply kind of a philosophical formalism, and I think some of the kind of ways to sort of drill down to the essence of things and understand what's going on
Is something quite similar between philosophical thinking and computational thinking. Mathematical thinking also has similarities, but it's in a much narrower area. There's a much narrower set of things to which it can be applied, to which it's successfully applied, and it's something where there's also the actual practical development of these things. The tower of ideas that have developed in mathematics is taller than the tower of ideas that have developed in… well, because it's a formalized tower.
you can kind of develop further than you can, I think, in philosophy, where you can, you know, you're still going back to what did Socrates say type thing, you know, 2,500 years later.
Rather than in mathematics, you're not going back to say, you know, what did Archimedes say? You're building on many layers on top of that. So it's a slightly different thing.
Let's see… …
Caleb asks, is the real future of programming about writing code or about designing the goals for AI systems? I think the real future is doing computational thinking. I think that's what you need to define a meaningful goal. If you just sort of vaguely say, hey, do roughly this thing, you're not going to get the thing you really want.

you have to be… you have to crispen up your thinking to the point where it is sufficiently precise that, yes, you can just use Wolfram language to make the code, you can use some AI to help make that code, but the real focus is on, well, what do you actually want?
And for that, you have to have some structure to formalize this question, what you actually want, and that's the story of computational thinking.
Let's see… … Memes is asking.
What's the best way to get folks with similar ideas to collaborate?
Hmm.
Well… there's in the world at large, and there's within a company, for example.
… I would say, you know, it's a funny thing. In the world at large.
For example, in the intellectual kind of world, sometimes people who are very closely aligned almost repel each other.
And they sort of see themselves as competitors, because they're sort of close, but one's gonna win and one's not gonna win type thing. And sometimes they're like, hey, let's chat about this, let's all be friends type thing.
How does one get to the, let's all be friends, …
in… you know, I think there's a certain amount of the ecosystem around what's going on that affects whether it's the let's all be friends or let's compete with each other type thing. I think if I look at fields that I've known, there have been fields that have been very collaborative, and there have been fields that have been very cutthroat.
And what's the difference between these? I would say the collaborative ones tend to be the ones where it's a very wide-open field.
Where there's… there's a lot to do, and not very many people in it. The cutthroat ones, and it's probably not surprising, are ones where there's perceived to be only a very narrow set of things one can do, and everybody's rushing to get to… to be the one who gets to do that particular thing.
And I think that that's, you know, in general, the… the, kind of,
Having, having the… you know, when people…
Well, when people are going to do things together.
Sometimes they all are going for the same goal.
Sometimes, they'll do a thing together, but they'll each get a different thing out of it.
You know, let's say people are collaborating on some project. Somebody wants to be the person in charge managing the project. Somebody else wants to be the programmer. They all got their various niches, and they're all happily working in those niches.
as… or everybody is, like, so keen together on getting to the final goal that that sort of pulls them… them through. I think…
in, …
You know, I have to say, my own experience with, sort of, collaborative projects, and maybe this is a piece of egotism or something on my part, is somebody always has to be in charge, and usually with things I'm dealing with, I'm the one who ends up being in charge.
Not necessarily because I want to be, but because somehow I'm used to being in charge, and I know a certain amount about being in charge, and I get frustrated with seeing nobody in charge, and so I wind up being in charge, so to speak.
But I think most things end up working better if it has, you know, if there's definite leadership, usually one person, maybe two, but not… not a whole committee of people. You know, I think… in terms of people working on, sort of, related things, I would say that one of my important functions, CEOing our company.

is just pointing out, you know, I'm the node on the tree to which everything else supposedly connects. And so that means that I have some idea what all the different parts of the company are doing. And it's a regular thing. I mean, it happens pretty much every day that I'm saying to somebody.
you know, they're saying, we're doing this, and I'm saying, have you talked to or looked at this other thing that somebody else is doing? Being the connector, so to speak, bringing together these different, these different pieces that are similar. And I would say in science, also.
and in technology, for that matter, I've… because I have a pretty large network of people and things that I know, it's, … I've often found myself in the position of being, oh, you should talk to so-and-so, or you should look at this thing. Being, you know, being the connector, so to speak.
And I would say that, generally, that's a very valuable thing to be. And what ends up happening, not every time when I say, go talk to this person, do they come back saying, yeah, we figured out something great to do together, but I would say.
I don't know if I'm… to put a number on it, I would say it's a solid half the time I'll introduce people, they'll… they'll end up… something definite will end up coming out of it.
And that seems to be, sort of, that's step one in making collaborative things happen. I mean, within a company, you can have this situation where there are three parallel projects running that are all basically the same thing, or should be cooperating, and nobody knows it.
And that's a… that's an issue of management structure. I mean, for example, in our company, something I've been pushing for hard for many years, and we're finally in pretty good shape on it, is this thing we call the Global Project List, which is just a list of the few hundred projects that are project-like things that we're doing.
And as you look at that project list, it becomes fairly clear
That, sort of, there's this thing here and that thing there, and they're really the same thing.
We also have a process list of things that are ongoing. I mean, projects, as far as I'm concerned, are beginning, middle, end.
type activities. Processes are never-ending things that will keep on going, whether it's maintaining a piece of code, doing, you know, doing events every so often, whatever else it is. Those are processes, as opposed to a project where there'll come a moment where it's finished. Tie it in a bow, have the rap party, we're done.
you know, it goes into a… there's a process, perhaps, of maintaining it, but the project is done. I would say an interesting problem in our company is we have lots of groups in the company, and we had previously sort of organized our global project list according to the groups that we have. This turns out not to work perfectly, because a lot of non-trivial projects span many groups, and it's kind of like, who really owns this project? Oh, it's in this group, or you don't even find it, because it's in that subsection of the notebook that's about that particular group.
And we don't even notice it. So we're kind of refactoring that right now, and maybe what will end up happening is, when projects are big enough and turned into processes, they will define new groups. So something that was a cross-group type thing, because it was being sort of put together to make a project, if it's a big enough thing, that sort of cross-group thing will just spawn its own group.
And I think we'll end up seeing that happen a bunch.
Let's see… …
Maybe one or two more.
Questions here…

ABF asks, how do you synthesize a quantitative mindset with the more abstract stresses of life and business?
you know, I think… I…
tend to be a… you know, I've done a lot of analytical thinking, so to speak. I've done a lot of, sort of, quantitative, you know, practical, precise things
And I also am exposed to many things in business, life in general, that, on the face of it, don't seem very analytical-oriented.
things about people and companies, and I was just mentioning, sort of, reorganizing pieces of our company and so on. And, you know, that seems not like, kind of, writing code or something, but it is.
If you… if you choose to think of it that way. I mean, for me, some of these things about, sort of, structuring companies, thinking about processes in companies, all those kinds of things, those things, the more you can think about them formally.
the more… and not by writing down some weird flowchart of this, that, or the other. I'm just thinking about… thinking about them in an organized way, and that will turn into organized documents of this, that, and the other, where things are put in organized buckets, and so on.
But I'm saying that the mindset of thinking about things analytically is super useful.
I mean, there are things where some question will come up, and I will immediately kind of feel like I know the answer. I sort of intuitively know the answer. And that's fine when that works.
But as soon as I don't intuitively know the answer, I have to figure out the answer, and that requires some kind of, in a sense, formalism of figuring it out.
And that's where, kind of, analytical thinking really becomes very, very useful. I would say the same is even true in dealing with people. I'm a person who likes people, I work with lots of talented people, sometimes I work with people idiosyncratic, with idiosyncratic features of many kinds, and it's kind of like, how do you deal with these people and things? And I have found that, sort of.
Thinking about people, at least implicitly, in a rather analytical way, is super helpful.
I mean, partly that sort of pattern matching of, this person reminds me of this person I knew 30 years ago.
And this is how I think about that. But I think… I don't think I have a formalism that I could write down for thinking about people, but I certainly have something that I consider sort of a structured way of thinking about things, and particularly that's things like
here's a person, here are some possible projects, jobs that they might do, are they a fit? You know, what are the attributes of this person that I can kind of
formalize, in some sense, informally formalize, like, this is a person who's really got a short attention span, they're really good for very short projects, don't send them off on a 6-month project, it won't work well. Or this is a person who is, going to be, I don't know, very, very good at dealing with people.
and good at sort of interacting with the world, whereas this is a person who really should be sitting in a cubicle and writing code type thing, and that's what will make them happiest and most productive, and so on. So these are things where you can be sort of informally formalizing these things, and that's a super useful thing to do.
Now, you know, there are situations that are less
sort of where kind of one's structured analytical thinking gets frustrating, and many of those situations I try not to put myself in. So, for example, one is kind of consensus development of various kinds. I mean, in, you know, being on a committee.

is something I… I really avoid doing.
And, because I… I find that that isn't
you know, maybe there's a logic to committees, which I just don't understand. I think there is, I just don't understand it very well. But to me, things happen, and they're just like, this is crazy, and I'm trying to convince people, and they have all sorts of issues, and so on.
And it's like, this is something where I can see this is the right thing to do. You know, if you say, what should we do? It's like, I know what we should do, it's this.
But then it's like, oh, but we all have to discuss it, and we've all got our points of view, and this, that, and the other.
That I find frustrating, and so I tend not to put myself in those kinds of positions if I can help it, and I usually can help it.
Let's see… As a question, sort of related to this, …
from Caleb again, how can computational thinking help avoid decision fatigue in everyday life? I'm not completely certain I know what you mean by decision fatigue, but, you know, for somebody like me, I make many, many decisions every day.
And their decisions about technology, their decisions about business issues and structure and so on, their decisions about strategy for doing things in science, their decisions about what project to work on, and so on.
I don't know, I find that the more decisions I make, the better I get at making decisions.
And also, the bigger the number of decisions I have to make.
the less I agonize and worry about how, oh, if I make this mistake on this one decision, you know, because I've got… if I've got 10 decisions to make, it's like, well, probably you'll get one of them wrong, but we're going to make 10 of them, and so I'm not going to be so wound up on the one I might get wrong, so to speak.
And I think… so, to me, it's… it's kind of, … I don't think there's a sort of an automation
to making decisions that's going to be very good, like, go ask my chatbot, so to speak, what I should do. I don't think that will end well in many cases. Although maybe… maybe if it learnt so much from you, it will be like, well, you should obviously do this, because that's what you did every other time. But you probably know that. You probably don't need a chatbot to validate that.
Maybe… maybe it's psychologically useful.
to have a chatbot validate it, just like it might be psychologically useful to, you know, ask some expert or consultant or whatever, and have them tell you the thing that you already wanted to do.
But it's still, I think, …
I don't see that… so that kind of structure. I think what is… what is useful in making decisions unstressfully, so to speak, is having made a lot of decisions, and being able to feel that… that you can
that you have confidence that you're going to get it right most of the time, because you've always gotten it right most of the time. And also, the fact is, the more that when you hear a question, and the more you can say, oh, I know what we should do.
The more you… it's instant.
the more you feel, I think, confident about it. And the less it's like, well, I have to figure out this and that and the other thing. I mean, sometimes you have to unpack it and know you've got to think about this and this and this. So I suppose in that sense, the fact that I tend to think
in a fairly structured way about… and the structure is mostly, what is the essential point? That's usually the questions being asked. Somebody says, should we do this thing?
And it's like… and nominally, the thing is… I don't know what it might be. Should we…

… do this… make this technology decision. And…
then the first question is, well, what actually is the decision you're talking about? I mean, it's like, you've got a bunch of words, and you're asking me, can you decide, are we going to do A or B, and they're a bunch of words, and I don't even necessarily know what all the words mean. So the first question is, what do you actually mean?
what does this actually mean? And then there are sort of drill-down questions about, okay, what consequences does this have, how does this connect to this, et cetera, et cetera, et cetera.
They're a kind of, … but I think it is a lot easier… maybe I should say this.
To me, one of the reasons that I find it fairly easy to make lots of decisions is because I feel like I'm making these decisions on bedrock, so to speak. That is, I really understand
what the decision is that's being made. Maybe I don't know what its consequence is, maybe there's some computational irreducibility thrown in there, but
the, you know, I really… I'm not sort of floating at the level of people asking me things where I don't really understand what they are, and I'm just sort of, oh yeah, that sounds right and that doesn't sound right. I find myself, sometimes, one thing that happens to me.
is people will show me something, I don't know, a draft of a web page, let's say.
And, or… and I'll sort of look at it, and I'll kind of look at it superficially, and say, looks like it's nicely designed, it looks like this, that, and the other. Kind of the vibes are right, and I'll just be like, yeah, yeah, it looks okay. But I didn't really dig in and really look at it.
And…
for me personally, I tend to, and I, you know, it's something I like to figure out better, I tend to be a bit bimodal. That is, I'm either just sort of glancing at the top and getting the vibe and saying that's okay or not, or I'm diving deep, and I'm reading every word, and I'm thinking about every word. What does this mean? Does this actually mean the right thing, etc. I find it rather difficult to be in between those two things. Maybe everybody does.
But it's something where I think one doesn't want to fool oneself about where one is, so to speak. It's either, like, the light touch, where it's like, you know, do I have an immediate gut reaction that's like, this is a really bad way to do it? And sometimes I do, or am I really digging deep and feeling like I know everything about what's going on.
That, in that second case, when I feel like I'm sort of working on bedrock, I feel much more confident about making that decision, and it's… it's… I guess it's much less stressful to do that, because I'm not just sort of floating freely and trying to sort of…
pick up the wind of this direction or that direction. I'm also not relying, this is another, I suppose, important point. So often, when you make decisions, it'll be somebody tells you this, somebody else tells you that. But… but you're kind of making a decision, like, yeah, what you're saying makes sense to me, let's do it.
Even though you don't really know what that thing is, because that person is the supposed expert who's telling you this or that thing. And for me.
I really insist on understanding what that person is telling me. I really don't accept
the… the kind of, oh yeah, I'm an expert, so I'm telling you this, you should decide to do this. I… I really insist on kind of being able to drill down. And that, for me, is… it really helps in making correct decisions. It helps in… in feeling like it is… it's not…
There's such an arbitrary, people-oriented.
kind of thing. It's… I mean, obviously, you have to trust people at some level, but you've kind of get to know them, and you know what they're saying, and you know how it connects to other things you know.

And I think that's probably… that's probably the real answer for me, is that the, you know, being able to think sort of foundationally about things makes it a lot easier
to sort of solidly make decisions without kind of stressing about those decisions, because you're not worried that there's something that you don't know about the foundations of this decision that is going to upset the whole thing. It's like, yes, I really know what this is anchored on, I really know what I'm deciding. It really, it really helps, I think.
Talking of which, it looks like it's time for me to go back to my day job, and … probably… what am I doing next here? Oh boy, yes, lots of decisions to make. This is a, … I am about to jump into a meeting which will be rife with many, many decisions to be made.
So I get to, take whatever,
I don't know, comments I'm making here, and try and actually apply them in practice, and make sure that they actually work.
Anyway, thanks for lots of interesting questions and comments, and …
Thanks for joining me, and I'll, talk to you another time.
Bye for now.