Hello, everyone. Welcome to another Q. And a about business innovation and managing life. We didn't have these for a little while, because I was at our annual summer programs, our summer school for grownups and summer research program for high school students. I have to say. I think both of them went really well this year. Lots of very interesting projects got done, and you can see all the projects out on on the web.
if you go to the home pages of those programs, there's a big scrolling array of all those. All the projects shingles for each of the projects.
It's an interesting process. We had. Well, this year. We had, I think, 49 grownups and 75 high school students.
And the challenge is to find a great project for each person to do. And I'm the main one who has responsibility for doing that. And it's kind of an interesting thing. That's sort of a a generalization of what I've had to do in management for many, many years, which is given a person
what is the best project or activity for them to fit into doing here. It's meet students for not that many minutes each, and find a little bit about them, and then
try and work out with mentors who've chatted with them some more, which project out of perhaps a long list that I've come up with, or ones that I haven't come up with are the right ones to suggest to these students. Anyway, I was really happy this year to see how well that all came out. I mean our main summer school for grownups. This is the 23rd year we've been doing that for high school students. It's the 14th year. So in all that time it's not. One would hope that we would have developed some kind of expertise in doing it, and I feel like this year we did particularly well. And I was excited about that. Anyway, I'm now back able to do these live streams again.
and I'm happy to try and answer questions people have here.
So let's see, D. 0 is asking, what AI tools do you use for personal tasks?
Well, very mundane ones like, you know.
siri, for simple commands, dictating things things like that.
I tend to use.
I'm finding I'm about 1 3rd of the time. I don't want a web search. I want a web summarization. I use Kaggy for web search, and it has. There's this sort of trick that if you end a thing you're asking with a question mark. It will try and generate a summary, and it allows you to go and make a more elaborate
query to an assistant, and you can kind of connect it to all the different kinds of Llms that are out there. I think about a 3rd of the time I find myself looking at those summaries. I think the typical workflow is. I'll look at the summary.
I might look at the references to the summary. Well, the summary itself might be enough, and you know I I don't necessarily trust the AI entirely, but if it seems reasonable to me and it's I'll tend to believe it.
Then. In terms of other AI tools, while I use our notebook assistant. In with some frequency, I would say I use it probably
couple of times a day at least. I use it kind of for enhanced documentation search
sort of thematic documentation search. And I use it sometimes when I have kind of an idea what I want to do.
And I want to just see some Wolfram language code that represents some guess that the AI has about what I mean when I say that particular thing. It's a good way to kind of get started in writing code for something. And the nice thing about Wolfram. Language is very succinct, and so you can easily see what the code does. And you can easily tell. Did the AI get it right

and produce the thing I wanted? Or do I have to change it? To get the thing I actually want and specify it precisely in in Wolfram language, computational language.
Where else am I using AI tools? I've been trying to use them
for kind of thematic searching of things like the scientific literature.
So I'm particularly interested right now in things like experimental implications of our physics project.
or some questions about foundations of biology. I'm about to launch into some questions about foundations of economics.
I think it's I find it. I'm sort of finding it useful as a way to trawl millions of papers, but not in cases where I know keywords where I know. Oh, people call that phenomenon in economics, let's say by this name. I don't know the name, so I have to do kind of a thematic search of the academic literature. And that's the thing that I'm finding reasonably useful with Llms.
Other kinds of things. I mean I tend to use when it comes to sort of thinking of different words for things and making up names for things I have to say. I tend to use more of a thesaurus like word search than a fancy AI system. I found the fancy AI systems have a habit of just telling me things that are kind of banal and
and and not really not really that useful. I find it better to just have a simple thesaurus. But occasionally I've used Llms for that purpose.
Where else? I mean, there are a zillion places where one is implicitly using sort of AI like methods. Whether it's sort of the the lots of mobile apps.
you know, navigation and lots of other things. I mean, I think that some level of machine learning is to be found all sorts of places, I mean in Wolfram language. For example, we've been using machine learning like methods for a very, very long time, particularly for the heuristics for meta algorithms.
So for example, if you want to solve some problem like solve a differential equation, there may be 25 different algorithms that could be used, some much more efficient and maybe more accurate than others, and the thing that we've worked on for at least 3 decades is to have kind of a heuristic methods, often using kind of machine learning ideas
to decide for the particular problem you've presented to the system. Which of those 25 underlying algorithms. Should it use, any one of them can give the right answer, it's just some of them will be much faster than others. And so that's the thing that we've we've often needed to use.
and but that's a very sort of under the hood. Implicit use of of AI trying to think, what other things have I use? Well, I guess I use transcription
a fair amount of you know, zoom sessions. I'm doing occasionally other kinds of things. I haven't yet found sort of the great sort of go, from what I'm saying to a to do list about what I should do haven't yet found a sort of great version of of how to do that. I think that
the transcripts that I get I have to say I'm I'm not finding them
super useful. I find them.
sometimes they're useful as something that I can search in. They're not so useful as something that I can summarize. What I want to do with.
I would say, yeah, that's a
no, I mean, there are some other very simple ones, like one routinely uses language translation. And I found that the sort of modern Llms are really it's become language translation, for most languages has become really good. I haven't yet

had much serious experience of trying to do real time translation, talking to people who are speaking other languages where I'm speaking English. I've had a couple of experiences which have been less than perfect along those lines, but I fully expect that to be a thing that will emerge as something that really works before too long.
Let's see, Brian asks
general question, but looking at what made windows 95 so successful and massively adopted, what will be the windows? 95 of AI.
I have to say, I think. Kind of chat. Gpt sort of was the windows. 95 of AI in the sense that it had a very simple interface kind of like
the simple interface that search engines like Google ended up having windows. 95 sort of had simplified the process and had made more elegant. The process of using computers had taken off some of the sort of
crazy restrictions and detail that had to be done. But I think that's sort of the thing that already happened in in the world of kind of Llm. Like AI, with the kind of Chatbot interface.
I'm not sure it's kind of the it's the minimal interface. It's the kind of interface that we we saw in search engines. We see more from Alpha. We see with with chat. We're seeing something where it's a conversational kind of thing rather than the kind of the one shot thing. But still the important point is that it's a very simple interface. And I think that's kind of the the big thing that helps mass adoption of something.
Let's see, is a question from Brady. Have you made economic simulations beyond what you describe what I've described in my book? New kind of science
which is now from 23 years ago. The answer is a bit.
Some people at our summer school various times have done additional simulations, and so on. In fact, just this year a couple of people did rather interesting work on something that I've been meaning to study in more detail for about 30 years, which is kind of how, when you're doing games between agents
and the agents have certain kinds of computational capabilities, how do those games come out. I mean, there is a in the economic literature. It's there are some famous examples where people say, Oh, with choosing between these
dozen strategies that game theorists invented. This is what happens if you have an iterated game where you're you're playing one side against the other at rock paper scissors, or something like that over and over again. The
I've always found that a rather unsatisfactory way to approach this, because in the kinds of things I've done it's very easy to sort of enumerate strategies, enumerate all possible strategies consistent with some kind of level of computational capability.
And so that's what a couple of people did this year, looking at different kinds of simple computational systems, where you can just look at all possibilities, all possible strategies, including ones that are really dumb. But looking at them all and seeing how they compete against each other. That's an example of a sort of a tiny piece of something that leads towards economic simulations.
I am really getting ready. I hope to start making a serious assault effort to understand the foundations of economics from kind of a computational point of view. I'm kind of suspect that a bunch of things that I've studied in our physics project in the things I've been doing recently in the foundations of biology.
I think these things have a lot of relevance to what one can study in economics. And I'm looking forward to studying this. I've just got a rather large number of things to do on my plate right now,

and I've been hoping to get to this for a few years, but I think soon I will finally be able to do something on it.
Let's see, Jesseo asks, what's a skill you think more people should learn in 2025.
Well, I mean, I think one very important overall skill is what one might call foundational thinking
how to sort of think deeply and in foundational terms about things, how to not sort of skate on the surface of things, and sort of know the the key words that the experts say, but rather actually try to understand what one is doing from a at a foundational level. I think that's a really important skill, and that skill sort of segues into questions like
like as well as knowing how to answer a question, knowing how to figure out what question to ask
for me. The there's sort of a foundational approach to foundational thinking. That is just ways of thinking about things.
But at some level one can do some of that as sort of pure thought.
And then one has to start sort of formalizing what one is thinking about. And the big methodology for formalizing what one's thinking about is is computational language and sort of the big example of that that we have is Wolfram language which I've been working on building for the last 4 decades or so, and kind of the notion. The concept of Wolfram language is to find a way to sort of take the things one wants to talk about in the world
and make them computational, make them kind of precise and well defined, so that one can think about them in these much more precise and well-defined terms. But also one can delegate one's thinking to a computer, to the actual wolf language system to go figure out a lot of things. I think the methodology, knowing computational language and being able to fluently use it.
Knowing Wolfram language, knowing how to how to take one's ideas and immediately put them into Wolfram language form. This is super useful. I mean, I just saw it. Dramatic examples of that
at our summer school and High School summer research program. Just a lot of people who have gotten quite fluent in using Wolfram language, and are really able to take ideas you just sort of wonder about and immediately sort of formalize those ideas and explore them. Computationally, I think this is just an incredibly powerful skill to have.
But sort of behind. That is this idea of foundational thinking, the idea of can you drill down and know sort of at a bedrock level what's going on in the thing you're thinking about?
Are you? Are you able to really sort of always
go to the go to the Primitives, go to the foundations
if you can, then you can feel very, very certain it's a very solid thing to kind of build up to whatever conclusion you want to come to. If you can't do that, you're always kind of floating. At least that's the way. I see it with all sorts of well, people say this, and and you know that's the thing that's commonly said by experts about this thing. I don't really understand why it's true. but that's what people say. And I'm going to sort of build my my tower on that kind of floating foundation.
I think that's much less satisfactory than being able to sort of drill down and really understand things. It's become easier to really understand things. It's easy to get all that information from the web. It's easy now to also be able to take the understanding. You think you have, formalize it in computational terms, see its consequences. Kind of that often shows you unexpected things that change your understanding.
And that's those are. Those are sort of skills of modern times that I think are are really important.

Now, having said that, I think a foundation of many of those things is just knowing facts about lots of different areas and having some feeling for the way that one thinks about things in those different areas.
It's sort of an interesting challenge. How do people learn foundational thinking because they don't really learn it. Typically sort of in school, in school. It's more about sort of the mechanics of answering questions of particular kinds. It's not sort of saying things like, well, how do we know that that is true, or what questions should we be asking those kinds of things? I think it's a real hole in sort of the education that's available in the world today
to to kind of be able to communicate to people sort of how to do foundational thinking, I think, to some extent, and I've I've done a fair amount of this myself
to some extent. It's a bit of a Socratic kind of thing for each individual person. You have to kind of have a discussion to kind of. Let them see what it means to actually think foundationally about things. I'm not sure that it's kind of a thing where it's going to be successful, to just sort of say for foundational thinking, you do things 1, 2, and 3. I think it's something where you have to sort of feel it and experience it yourself to really internalize and understand how to do it.
Let's see.
It's a question from day here. Do you think language can persist outside of life?
Well, I mean language?
That's a complicated question, because it has to do with what we really mean by language, is it? Is it?
If we're dealing with computational systems, kind of communicating with each other? Here's sort of the story within a system, within some computational system, it could be a brain, it could be an Llm. It could be some, some program, whatever else there may be some internal representation of kind of what it's dealing with.
The question is, can that internal representation be communicated to another computational thing or another brain.
even something which is sort of a copy of the 1st thing in terms of the in terms of the code that it's using. But a different instance of it. What does it take to kind of communicate things from one computational system to another?
Well, I think what it takes is to sort of package up
those things into some kind of standardized language
for brains. We have all these different nerve firings going on in our in our brains at all times, 100 billion neurons doing their thing, lots of individual nerve firings, you know, a billion at a time or something. But the details of those are not what we communicate, because the details of those don't really connect with another brain that has different nerve firings going on.
So what we have to do in order to sort of communicate brain to brain is to package up our detailed neuron firings into things which are kind of standardized, which can be transmitted to another brain and then unpacked there.
And that's, I think, the key role that language plays is as the standardized way of representing some kind of version of what all those nerve firings, what all those internal thoughts are representing. It corresponds to a sort of standardized representation of some aspect of those thoughts in a form that can be communicated, for example, to another brain.
Well, you see the same kind of thing needed. If you're communicating from one program to another, there has to be some standardized format that those programs are communicating with each other using.

And that's true whether one is doing sort of interprocess communication, whether one's doing, interacting with a server. All these kinds of things there needs to be some sort of standardization of what's sent. That is not all details of what's going on inside that computer system.
But just those aspects that can be are readily defined and can be can be sort of delivered to another system.
So I think there are
plenty of examples where it's sort of computer to computer communication. I mean, one that's that's happening right now is with things like Llms. Some of the communication can be done in natural language like English.
But it's becoming increasingly clear that one needs sort of a more precise level of communication. And Wolfram. Language is a great way to do that, to have the Llm. Generate peace of Wolfram language, and then have that be what is transmitted to another Llm. Which can kind of unpack it and make use of it. So yes, I think that language as this concept of how you sort of standardize thoughts
is is something that absolutely can exist, and already does exist outside of actual living systems and systems with actual biological brains, and so on.
Let's see.
let's see, Jacob is commenting while historically, algorithms were engineered by humans. Now, more often than not, they're learned from data.
Where do you think this will get us, will we discover new algorithms for things like mathematical optimization or other computationally critical tasks?
Well.
I've been interested in mining the computational universe for algorithms for about 4 decades.
And I've done a lot of it.
And with machine learning, we are typically
trying to sort of incrementally figure out, we're trying to progressively adapt until we have something which can serve as a useful algorithm for things.
I've more often in the past done more exhaustive kinds of searches in the computational universe. Both these methods are useful. Exhaustive searches in the computational universe tend to get you much more unexpected and often much more minimal results than something where you're sort of doing progressive adaptation to get to the thing you want
in out. I've done many examples, actually.
where sort of you can exhaustively search a trillion possibilities, and you find 10 that are very bizarre and very, very useful. They're not things a human would ever have come up with. They're things that just happen to exist out in the computational universe and are very useful for doing particular operations.
That's so. That's sort of one thing. You can a different end of things is, you can try to engineer things sort of step by step yourself. So you always understand every step of what you're building that tends to give you things that are much less efficient.
although they're understandable by humans. And sort of an intermediate between those 2 is the thing where you're sort of doing adaptive evolution to get to the point where you can to get the thing you want. I think that's more relevant than well, to say that
there are
neural nets are a convenient sort of parameterization of computation. There's a lot of kinds of shallow computation, at least, that can be represented by neural nets.

They're just like you can represent some mathematical function by some power series of powers of X or something. So you can also represent a mathematical function by the successive layers that correspond to what's in a neural net, and you can represent more than just simple mathematical functions of one variable or something. You can represent things that depend on sort of all the pixel values in an image, or all the characters in some long piece of text, and so on. And those are things where you can use a neural net as kind of your model to figure out what you what you want to get from that particular input.
Now, when it comes to making kind of algorithms, I think that sort of the the what what you're more interested in is using the neural net as the actual algorithm for something rather than training the neural net and then feeding in inputs to see what happens with those inputs. So you're really trying to make a neural net. That sort of implements, a particular kind of thing that you might think of as an algorithm.
And what has been the big surprise is that the kinds of things that we humans find easy to do like tell cats from dogs and pictures, and so on, are also things that neural nets find it fairly easy to do.
There are also things that we humans have always found it hard to do, like sort of elaborate mathematical computations or things where they're just many steps of computation that need to be done. Those are not things that Llms find easy to do, that neural nets find easy to do. I think the the idea that one's going to sort of find well, there are there are. This is this is a complicated question, because there are certain kinds of algorithms that sort of seem to be a good fit for things that can be to which things like neural nets can be applied. And there are other kinds of algorithms which are terrible fits to neural nets. I think a big dividing line is kind of how much computational irreducibility there is in the problem one's trying to solve. To what extent do you have to do all those computational operations to solve the problem? Or is it the case that the problem really has a quite reduced amount of computation that's needed? But you don't know what that computation is.
So, for example, when you see something which is, let's say, solving an equation, solving a differential equation for some sort of physical kind of thing like, I don't know something about fluid dynamics, or something about molecular structure, or something of this kind.
When you look at those solutions, one of the things you notice immediately about them as a human is. They're kind of boring. They're kind of oh, it's just a smooth variation from here to there, and so on. It's not. Oh, there's a detailed thing that was produced. It's a detailed number that has 25 digits, and you got to get every digit right? Rather. It's just sort of this smooth structure that you say that's pretty boring.
Well, those pretty boring things are things that I think neural nets can do a good job potentially at reproducing. Certainly we've we've been beginning to find that.
And that's a place where, in a sense, there probably is at some level sort of a better way to solve the equation. We don't know it, and neural nets are a good way to sort of approximate that another version of that very different kind of thing is when you say, work out this thing that's going to come out as an exact formula.
Well, if the exact formula is like going to be something where it's like. Here's the formula, just believe me, that's a very bad situation for things like neural nets. But when it's a question of, here's a formula, and there's a way to check this formula. There's a way to tweak the formula, to to really nail it down and get all the all the details accurate.

Those are cases that are much better for neural nets. So, for example, let's say you're trying to solve a differential equation
symbolically.
Well, one of the things that's important is even to know roughly what kind of functions could appear in the solution to that differential equation.
That's a place where neural nets can do quite well. It's kind of a fuzzy problem. And if it says, Well, it could be this function or this function or this function, and not all the functions are relevant. But given, you know what kind of functions they are. You can go in and try and figure out well what detailed coefficients and parameters, and so on, might appear in those functions. That's a thing that's a sort of hard computation task.
You can go in and do that. And you've used the the neural net to kind of give you that sort of heuristic intuitive idea of what kinds of functions might appear.
But if you didn't have a way to check the answer, a precise way to check the answer. It won't be useful for doing that.
So general thing to say, sort of mining the computational universe for algorithms. Very thing to do. I've been doing it for years. You find all sorts of things where you look at it. And sometimes you say that's really clever. I never would have thought of that. Often you say, well, I can see that it works. I can even prove that it works, but I can't explain why it works. I don't have a good sort of human narrative explanation, for why it works.
The intermediate case of kind of adapting to the point where you have a good algorithm for something. I think it works best when the algorithm is kind of something that that has to be sort of. It's it's trying to do something approximate. It's trying to do something where there's there's not a lot of detail that needs to be got right.
And yes, there are places where that's that's important. And we've been using such things for a long time, I would say, and things like image processing. That's a place where there's sort of immediate use for that. It's not a question there's no sense in which it's like is every
for most kinds of image processing. It's not like is every pixel where that pixel should be? It's does the image look roughly right? Did it pick out roughly the correct set of pixels that are the boundaries of that particular biological cell on an image, or or some such other kind of thing.
I think the things to say about when you do generate
kind of algorithms in this automatic way in this, what's the difference between what comes out and what comes out when you generate algorithms by sort of human engineering, I think one of the things that I've noticed that's different, and is that in human engineering there tend to be modular pieces where every piece can be understood on its own.
particularly when you're finding algorithms by just searching the computational universe, you tend to have just this one sort of blob of computation that happens to do what you want.
But it isn't something that can be readily separated into pieces where you can explain this piece, does this, that piece does that, and so on.
The adaptive evolution case seems to be a bit intermediate between those 2. Still trying to understand this. It's very relevant to questions about the foundations of both biology and machine learning itself how that actually works. But I think the picture
that well, the the picture that has tended to emerge for me
for machine learning in general is that you're kind of getting these kind of lumps of irreducible computation, and just sort of roughly putting them together to achieve the objectives you want. If the objective you want

is coarse enough, that's a thing that can work. If there's a lot of fine detail, then it tends not to work. This is really the distinction between computational irreducibility and computational reducibility. But you often don't know. You have computational reducibility and a good way to sort of find it is to generically use a neural net, use adaptive evolution, and so on to get there.
Paulo asks here, how is Wolfram research preparing for the time when AI agents will do most software development? And when do you think this will happen.
You know I have to say I think I and we have been sort of living the AI dream at our company for many decades now, because kind of the thing we've been trying to do is just automate everything we can.
What is called AI has evolved over the years much of what we do and have done for a long time with our kind of symbolic language, symbolic programming, and so on, is what would have been called AI in a past period, and might be called AI again in the future. The kind of neural net AI is something that we've used a bit.
and we can make use of when we need to. It's not been the core of what we've done. But the idea, the whole concept of go from an idea to sort of automatically develop that idea. That's absolutely what we've done and what we've been building for for 4 decades. Now.
now, when people say, Oh, I'm going to get an AI to write that code for me.
That's a very weird thing at some level. It is the thing that I found very useful, for example, with our notebook assistant is, you can be lazy and and vague in what you say you want the code to do, and the notebook assistant will produce something. A piece of code. Now is that piece of code what you actually wanted to do? You've got to read it to see. That's something that's easy in Wolfram language. It's very difficult in a low, level, programming language.
because you just get a big blob of code, and it's hard to tell you have to sort of run the code in your mind to understand what it's doing in Wolfram language. The primitives are very high level, so you get to kind of have a small number of chunks of kind, of of sort of functionality that you have to be able to understand, to see what the code is going to do.
But the thing, the idea that you just sort of vaguely say to a computer I want roughly, this kind of thing in the end, if you want to build any tower of functionality, you're going to have to have something precise done at certain levels in that tower. And if you want something precise done, just vaguely saying what you want.
it's never going to work unless what you want is pure boilerplate, unless what you want is something that is just the same as everybody's done a Zillion times. If it's the same as everybody's done a Zillion times. Well, then, you can write that boilerplate piece of python code or something to do it. But guess what if we did our job in language design properly for Wolfram language? There's probably just one function.
That is a built-in function in Wolfram language that does that. You don't have to write the big blow of code. You just use that one function to achieve the objective you want.
and that's that's the thing that allows you to both do it with a much more automation and have a thing where you can understand what it's doing.
So.
as far as I'm concerned, watching what we've done, watching how we built Wolfram language itself has been a very recursive process, making use of what we've already built in Wolfram language. And we've been building kind of higher and higher level structures that allow us to sort of automate more and more. And and that's been a thing that sort of the the. It's it's kind of it's kind of surfing the AI dream, although not with the particulars of neural net ais, and so on.

So you know, the thing that to me is is just bizarre is that if you look at things that you can write in big blobs of low level code. You can write them in tiny amounts of orphan language code. but often people don't do that. They sort of feel more comfortable writing a thousand lines of code full of bugs rather than writing 3 lines of orphan language code. That's not true of sort of the most sophisticated folk in my experience, I think what happens is that people say, well, I'm a programmer. What do I do? I churn out lines of code.
If somebody says, Well, you should figure out what the the sort of next step of what that code should do, they said, that's not really my job. My job is just to churn out lines of code.
So sort of the most sophisticated folk who are running tech companies or or figuring out sort of frontline things in science.
Their big thing is to sort of figure out the next idea about what to do and for them using Wolfram language, as as many of them do, to sort of go from that idea to the development of that idea. That's a very powerful thing to do. And it's a thing that fits in with how they want to work. They want to have an idea and very quickly be able to implement that idea. Then go on and have another idea.
But that's not the rhythm of software engineering programming, so to speak, the rhythm of programming is you're given a specification. You don't make up the specification.
then your job is to grind out lines of code to implement that. That's a very bizarre activity. It's in some sense, a very low, level activity. It's an activity that I know we've successfully automated for a great many things. And if you look at the productivity that our company has had over the years. It's sort of spectacular
given. We only have 800 people or something. If you look at what we've been able to implement. It's really very impressive. And the reason for that is because we've automated all these layers of what might have been called programming. We're not writing those thousands of lines of code. We're writing the 3 lines of orphan language code that allow us to implement things to another level.
So my
I think it's very nice that people are sort of having a wake up call with AI agents, hey? This way of doing sort of programming and software engineering that we've been doing for years doesn't necessarily make sense. I would hope that more of them will understand that the things we built in sort of our very high, level, computational language. That's a great direction to go rather than just like I'm going to bash the Llm. Until it finally does what I want
worth remembering that you. The only way you can know that the Llm. Did. What you want is to read the code.
You could maybe say to the Llm. Here's some code. Tell me what the code does. Oh, tell me what else the code does. I don't think that's a very efficient way to work much better, that you have code that you can actually read. That's high level enough. That's humanized enough that you can actually understand what it's doing that happens to be what our technology stack that we've been building for the last 40 years does.
And it, I think, will be a good thing if there's sort of a contraction in the in the number of people that are doing sort of low, level sort of churn out code type programming.
I'd like to see an expansion in the number of people using the computational language that we built, because I think that will allow a great deal of better productivity, but also better ability to explore new ideas, better ability to take ideas that one just sort of has one day and see their consequences. If every idea you have, you have to feed it to a programmer who's going to come back in a month

and give you some some result that might be where you've probably forgotten what you asked. But come back with some result. We say, well, that's not actually what I wanted. Let's iterate again for a month.
That's that's a very
bad way to kind of develop ideas. Much better is to have something where you can hopefully with your own fingers, so to speak, type in that piece of orphan language code and get a result in 15 min, or an hour, or something about what the consequences of that idea are. This is the methodology that I've used for years.
both in building technology at our company, but also in doing science. And it's what's kind of allowed me to make lots of kinds of pieces of progress in science that might seem kind of might seem. How could you possibly do that? It's sort of too big a step. But it's really because I'm able to have lots and lots of the smaller steps just completely automated for me.
Let's see.
Array asks, do you ever procrastinate when you feel pressure on yourself from other people?
I do procrastinate. I like to think of myself as a rational procrastinator.
In the following sense.
There are certain kinds of things where the longer you leave it before you start it, the more likely you are to do the right thing.
I mean.
it's it's like you can say, well, I'm going to prepare this talk a week in advance. I don't really know what the audience is, and I might have forgotten what I was going to say by that time. But I'm not going to procrastinate. I'm going to prepare it.
What I've found is that the if I I'm afraid many talks I give are I don't end up preparing at all. but when I do, the closer I can get to actually giving the talk, you know, if I can like see the audience, if I can get a sense of what kinds of things they're interested in the likelihood of preparing a good talk is vastly higher. In fact, for me, the calculation is, it's better for me to do a completely unprepared talk where I can see the audience and get a sense of of what they want. That's better than preparing something and having it miss what the audience wants. So the the sort of don't prepare too far in advance thing. In a case like that I view as being a piece of sort of rational procrastination.
The same is true with things like, Oh, I don't know. Marketing, product, marketing kinds of things. It's always good thing, at the very beginning of building a product, to say, well, what would I say about this product to market this product? That's a really useful thing to define what the product is supposed to be. If you can't explain what the product is at sort of a marketing level when you start building it. That's bad news.
But then you actually build it, and things will evolve in ways you didn't expect. There'll be things you that turn out to be easy to build. There'll be things that don't when you actually try them. They don't really make sense, and so on. So to me. I tend to leave
until I leave kind of the final description of what is this product, and why should one care about it until the very end, until one knows as much as possible about what the product actually is? And that's another kind of form of what I would consider sort of rational procrastination
when it's the thing of other people say you should do this, I'm probably more immune to that particular dynamic than most people. I think I'm I'm sort of a strong willed CEO type, and so I tend to be more in the nature of. I'm going to do what I think I should do rather than I'm going to sort of take input from lots of other people about what I should do so. I'm probably not not the best person to talk about that particular dynamic.

I mean, I think a thing that I see quite a bit within our company, for example, is people sort of encouraging, saying to other people, Oh, can you do this thing, or whatever? And sometimes they say, Oh, the people that we asked to do this aren't doing it. They're not responding to our email, or whatever else
the number. One problem in my experience with that is not
sort of obstinacy on the part of the people to whom the email is sent. The number one problem is, the email wasn't crisp enough. It didn't. It didn't give indications that if the person who's receiving it actually did what was asked. Well, 1st of all, they may not know what was really being asked. And second of all, they need to feel that what's being asked is something where the person who's asking can actually make use of what's what's produced.
So you know, often when it's other people are saying, oh, you should do this
well, what exactly should I do? And if I do it, what will you do with what I've done with it? For example, these are things that I think sometimes the asker is is as much at fault as the person being asked in terms of whether the particular thing gets done. And it's sort of the crisper the ask can be, the more likely it is that the person being asked
will will know what to do, and will realistically sort of psychologically be prepared to do it.
I mean, I think when you feel that the thing you're being asked to do is like, Oh, yeah, I can just do that, you know. It'll take me a short amount of time to do it, or the thing that I'm being asked. I understand why I'm being asked. I'm motivated. It sounds interesting. Sure. I'll jump in and do it rather than I don't understand what was asked. I'm going to try and go back and figure it out and unscramble it, and I can't be bothered to start that.
You know, I have to say, I see that with email that I get
from lots and lots of people when it's sort of a crisp ask, and I immediately understand it. And there's something definite that I can do with it.
That's much more likely to get answered quickly than the thing where it's a long kind of rambly thing. And it's like, Well, what's the actual? Ask, what is what? What kind of a thing could I do that would respond to this email in a useful way.
Let's see it.
I itched itch, asks, would you consider yourself to be creative?
They say I feel there is a common discourse of those who are more mathematical, scientific, and lack creativity.
I think I'm fairly creative. I've invented a lot of kinds of things.
I think the dichotomy that you're seeing in part is people who learn sort of how to technically do things.
How can have
a situation where they think that's the important thing to do. Maybe when they were kids they were always asking different questions and were quite creative, and were coming up with different kinds of things. But then they went through the education process and they were taught. This is what you do. This is the this is the mechanism by which you get things done. This is the kind of mechanics of what has to be done.
and they kind of lost that kind of childlike creativity. One might call it and instead, that sort of what was emphasized was the more mechanical kinds of things about figuring stuff out working out sort of the the scientific answer, or whatever else.
I think that the at sort of the the leaders of
pretty much every field have a certain degree of kind of creativity and strategic creativity, knowing kind of inventing strategies for what to do.

and as well as just the mechanics of of doing the particular thing well.
There's certainly plenty of room in the world for people who just do a particular thing. Well, but that's a different. It's a different kind of problem from the sort of strategic creativity to figure out new things to do, and so on.
I would say that that for me
I'm I'm always coming up with new ideas about things. The the challenge is more to channel those ideas so that they're about things where I can actually execute on those ideas.
And I suppose over the years I've expanded the set of things where I can sort of execute on the ideas, not least through building technology, building organizations and so on, to make it possible to execute on those ideas I mean, like in the basic science side, our new Wolfram Institute is a good sort of channel for executing on certain kinds of basic science ideas. Wolfram language and our company, Wolfram research is another kind of big mechanism for executing on ideas and sort of the thing for me is well, I'm always coming up with new ideas about things. How do I come up with those ideas?
Usually it really has to do with this foundational thinking kind of approach.
It's like, well, what's the basic point here, you know. And then what's the obvious question to ask? Given the point that, I understand is the basic point.
It's it's kind of almost one has to be kind of. I suppose one has to be very pedestrian, very childlike, very sort of stupid about things to to ask the obvious questions.
because most of the most, the best creative things I've come up with have been sort of obvious questions. Those tend to be the things that are more things on which lot a lot can be built is the obvious questions, you know. Can you do this would. What about building this kind of thing?
I think that tends to be the approach. And yes, that creativity can be kind of crushed out of people by the need, the feeling that they need to be sort of doing this mechanical stuff running very fast to do the mechanical things, no time to sort of think foundationally about things. No time to sort of think creatively.
That's if if you want to do things that are kind of really at the leading edge. It's a mistake. That's a mistake to be sort of so concerned with sort of the mechanics of what's being done.
and I kind of feel like I think I get pretty good grades for for sort of creativity in the kinds of things that I come up with, and and more than that the kinds of things that I have built, the the wherewithal to actually execute on it's 1 thing to just have a sort of an idea. It's another thing to have an idea that's grounded in something you can actually execute with.
let's see.
Katie asks how important is international trade for innovation in different sectors will scaling back international trade result in slower innovation.
In some places, perhaps. I mean, I think the Us, you know, remains the largest world's largest economy and just innovating, for things being done in the Us. Is is a big driver for innovation, and I think plenty big enough for many purposes.
The challenge often is, if you're doing something innovative and you're in tiny Country X, the the place where the the being able to sort of sell that innovation.
There's only going to be some small percentage of people who might care about that innovation. If you can only sell that innovation within tiny country X, there just won't be a critical mass necessary to support the effort to do that innovation.
You know, one sees this all the time with technology products and software in particular, you know. Can you make it in the Us market? Is it something where you can get from the country where you where you build it to actually sell it in the Us.

And and and that's a challenging thing. I mean, there's. And sometimes I think the issue is that in the Us. There's a certain kind of set of cultural expectations about how software works and how things are sold and all those kinds of things. And sometimes people have a challenging time coming from from other countries, and sort of understanding enough about the Us.
To sort of be able to to present and and sell products successfully there, I think.
in terms of. If you look at our company, for example, the Us. Is the number one market for us. Europe is the number 2, Japan, other places in Asia, and so on. They follow. I will say that it has become increasingly difficult to sell software in lots of countries around the world. They're very high piracy rates in many places. So you just don't get to sell very much
in some cases. If you have sort of server based technologies. You know, you have to have some locally owned company, and there's all kinds of complicated government regulation, and so on. Sometimes you can put up the software, but you can't get payment for it. In those countries
it's become increasingly difficult to do business in different countries around the world, in an area like software. I mean, I would say that when we started selling software in my current company in 1988,
the the big thing that was that happened at that time was resellers in different countries, people who would sort of take the software that we built, you know, in the Us. Or wherever, and being able to sort of present it to the local market.
we still have plenty of resellers around the world, but I would say that that's not such a big thing, because back in 1988,
you were kind of dealing with sort of handwritten registration cards sent through the mail and people telephoning to to order things, and so on with with the web that all became very internationalized.
And it seems as if anybody from anywhere in the world should be able to buy software, for example, from anywhere in the world. But it isn't true, because there are all these complexities about things like payment mechanisms kind of whether, if you make a version of your software that's localized for a particular country, whether that actually makes any sense at all, or whether there's, you know, 97% piracy rate in that country, which means you never really sell them a thing.
And and you could say, Well, you know, maybe it's the wrong model of selling things, and maybe it is, and maybe one should do the thing where
when? Sort of giving it away and making money from the ads, or making money from service, or something like this. I don't think that works either for a whole variety of different reasons, I mean. So I think that the idea of sort of the the full globalization of software is is not not as real as one might have hoped. And I think it's getting less real over time. Now, how much does that affect
innovation that's done, for example, in the Us. Probably not so much because the us market is big enough on its own to make it sort of worth doing the innovation without these other markets. Now, there's certain kinds of innovation
that other markets specifically need. I mean, whether it's I don't know. Right to left. you know whether it's figuring out how to do the kinds of things we do with math typesetting in a situation where there's, you know, right to left characters and so on, whether it's doing things that are specific to ideographic languages or something like this.
Those are things where, yeah, if you can't make a market in those countries, then it doesn't make sense to do innovation around those particular kinds of things. But certainly in our business, for example, of of making software for general computation, so to speak, that's something that

mostly it makes sense to do the innovation just for the Us. Market. It doesn't make so much difference.
Whether I mean we certainly I mean for us. I don't know something about 2 thirds of our revenue, for example, as a country, as a company comes from outside the Us. In aggregate. But it's not like the Us. Market is is tiny for us.
Hybris is asking with all the content you produce. Don't you
fear the possibility of becoming predictable in the future, and therefore easier to manipulate by AI.
I think making an AI version of me becomes more easier and easier as as I yak on and on on live streams and things, and there's more and more more and more
tens of millions of words out there that I've said, and that one could train an AI on. I'm not sure the question of whether it makes it easier to manipulate me. Because there's sort of more that the AI can know about me. It's an interesting question. I
I I think perhaps.
Well, it's a thing where, if you really know nothing about a person you have to use if you want to, quotes, manipulate them. If that's what you're choosing to do, you might have to use sort of generic human hacks to try and do that like it's free. It's whatever. There's many, many kinds of hacks. That sort of a fairly generic, although, won't work in all cases.
Will it be easier for the AI to figure out kind of what exactly to say to you, to persuade you to do something if more is known about you? The answer is surely yes, I mean, that's the whole idea of most Internet advertising is the more you know about the person, the better you can target the ads the better. You can sort of persuade the person to do the thing that the ad is trying to get them to do so. Yes, I'm sure that's true. I mean in a more positive light
the more the AI knows about you, the easier it is for the AI to tell you just that one thing you need to know to get you to understand a particular thing, or if you're building something like an AI tutor, the more the AI tutor knows the student the more it's going to be able to explain things in the terms that that student can understand.
So I think it's like so many things. It's kind of a double edged sword the better the AI understands you the better the AI can help you, perhaps the better. The AI. Well, certainly, the better. The AI can figure out what ad to feed you, and perhaps the better somebody can sort of work through the AI to provide the things that that are needed. To quote manipulate you. I think that's surely the case for a lot of bot
kinds of activity on social media where the bot is, is presumably customizing what it says to particular types of people who show up, who have certain kinds of feeds and and all those kinds of things.
It sort of reminds me of the whole basilisk idea of oh, you know things we do now. The ais of the future will
will come and and be be negative about us, and be mean to us in the future.
I have to say I find it very interesting that a lot of the kinds of tropes of religion
seem to reappear in these kinds of technology areas.
I think it's kind of a you know, the there will be a thing in the future, and there will be a day of judgment in the future. It is, you know, that's both a religious trope, and it also has become kind of an AI technology trope. And I find it kind of interesting that there's sort of enough commonality. We're talking about some human hacks. There's enough commonality in the ways that people think that these are ideas that emerge

both in religion in often thousands of years ago, and in sort of modern thinking about AI. I think it's sort of an unconscious thing for people that they just come up with these ideas. And it's like, well, actually, you know, that idea has been talked about in religion for 2,000 years, or whatever 4,000 years, whatever it is a thousand years, whatever
it's it's sort of interesting that these ideas sort of recur. They're part of the kind of generic human feeling about about things and about what might happen in the future, and so on.
Let's see, maybe a couple of other questions.
0 G is asking, what's my opinion about using? Automated theorem provers and dependent type languages to put provable constraints on AI, so that its generated code can be proved to meet certain logical requirements.
Well, I think it's really hard. I think there are things where, for example, the structure of the code in terms of its syntax, things like that. I absolutely can see that being something that you can systematically get the AI to only write
things where the syntactic structure is correct.
When it comes to, does it do the right thing?
Well, the question is always well, what is the right thing? Somehow you have to define what the right thing is.
and that's where things get difficult, because at some level. When you write a piece of code that code defines. What the thing you're trying to do is
maybe the code. You had something in your mind. That was I wanted the code to do this. And the code you actually write doesn't quite do that. It has a bug that leads it to do something else in certain cases, but at some level. The code is the definition of what you wanted the code to do.
Now, when you say, let's add this validation step.
Well, you have to know that you can add validation that actually represents what you want the validation to do. You might say, well, I want this AI, to satisfy these requirements to do these particular things. It's kind of like Asimov's laws of robotics, you know. You say, I want this, this robot, this AI, to satisfy these particular laws and never do anything different?
Well, the problem is that those laws probably don't capture what you actually want.
and in. In so, in other words, you've got the code on one side. That is one representation of what you want to have happen, and then you've got the validation code on the other side saying what you want to have happen. They're both saying what you want to have happen.
Now, there are some cases where you can say, Well, I don't want bad syntax or something, or I don't want something which is going to in a low, level language, you know, scribble all over memory and things like this. Those are things you can ask for, but beyond that it gets really difficult. And really the validator is kind of expressing what you want. But that was what the original code you built
was supposed to do as well. So it really doesn't end up gaining you something to do that now, when it comes to saying, you know, you've got some big question like, Oh, I've got a cryptocurrency I'm building. I want to make sure that tokens can't be spelt spent twice.
It's really hard to validate that what happens is any piece of code that is sort of trivial in its operation at some level. Yes, you can validate that any piece of code that sort of digs down into computational irreducibility, undecidability. Things like this becomes arbitrarily hard to validate.
And so, for example, technically, things like dependent type theory, dependent type theory is undecidable.
So that means there are things where you might say, let me prove it, I can prove it in principle.
Well, the problem is, the length of the proof could be arbitrarily long. It could be arbitrarily

difficult. You can't just say send in the dependent type system and all we well, so to speak, it's really just a and here's another. There's a here's a sort of second opinion about what the code is supposed to do. See if that works as well. I think that's the kind of approach one has to take there. I don't think
that patching it up afterwards is a realistic thing to do. I mean, for example, in our notebook assistant, we do do a certain amount of patching up obvious Llm. Mistakes in the code we generate. It's not that satisfactory. Most of what we can do is detect that there was a mistake rather than being able to say how to make it. The thing that the Llm. Should have said, because we don't really know what that is.
Well, let's see, I think I should wrap up here because it is time for something for my day job. But thanks for joining me.
Thanks for lots of interesting questions. I see many more here that I would like to address and look forward to doing that another time.
but for now see another time bye, for now.