Hello, everyone. Welcome to another Q&A about history of science and technology. We haven't had one of these for a little while. Happy to talk about
either things I might know about history of science and technology, or personal experiences of it, and so on.
Well… Let's see… truly asks, When did the idea of computation first begin? It's a good question. It's… what do we mean by computation? For me, the abstract idea of computation is you specify rules, then you work out the consequences of those rules.
implementing that on electronics and things like that is a different story. Building this sort of giant tower of technology we have on top of the idea of
implement certain simple rules, see what they do, that's, again, a separate thing. But if we ask the question, sort of, when did the notion that you could specify rules for things and then build something from them, when did that originate? The answer is definitely there in antiquity.
So, we know, you know, whether it's, you have this way of making ornament that is this, you know, sort of precise algorithm, in effect for making ornament, that would be one thing, or doing, you know, doing some architectural plan
Whether it's the way of specifying, you know, what formation your armies should follow.
Or whether it's the way, and this is something people certainly recognized in antiquity, of saying how words are put together into language through grammar, the idea that there are sort of rules for constructing language, that was, again, known in antiquity.
Now, also in antiquity, there were sort of rules known for calculating things, whether it's with math or otherwise.
And sort of the one data point we have of something very much more like a computer from antiquity is the Antikythera device from some time between 1st century BC, 1st century AD,
It was a… it's a device, it's about that big, and maybe a little… yeah, about that big. And it's,
A, it's a thing with a bunch of cogs.
And, it has, I don't know how many, some number of tens of cogs, and it was, it was originally found, the object was found around 1900,
It was in the 1950s, it became clear it was a device of some kind. Then, in more recent times, last 20 years or so.
It's been… had CT scanning and all that kind of thing. It's been kind of decoded what it is. What it is, is a cog-based computer that calculates certain astronomical kinds of things, like the Saros cycle for eclipses and things like this. So it's a thing which, pretty convincingly, is kind of a clockwork computer.
For computing the specific… for doing that specific computation about astronomy.
Okay, so if we look at, sort of, the history of things, then…
actual, sort of, clockwork computers don't really appear until the mid-1600s, when a guy called Shecard built one out of wood for his friend Kepler, and then a little bit later, Pascal, Blaise Pascal, I think it was 1640 or so, 1642,
maybe built one out of brass, which still survives, and those were addition calculators that had cogs, kind of like an odometer. You make the cog go around enough times, and it then picks up the next cog and has that increment by 1, so after you get to the nines, then you carry over to the 1,0 and things like this. You could use that for
a, Kind of a…
a mechanical calculator. And mechanical calculator technology advanced, you know, Leibniz built one that was sort of nominally a four-function calculator. Lots of mechanical problems. When you get 99999 and you have to move those cogs to get the 1000, you know, that was a

mechanically difficult thing to do, to both have the things work easily for small numbers and have that… those carries work well.
But so, the idea of, kind of, there being sort of a machine that could calculate things, very specific things, arithmetic and so on, that was a known thing by the 1600s, and that technology advanced a lot over the following 300 years.
Now, when it comes to, kind of, is there a kind of formal way of setting up rules and looking at their consequences, as I say, there's specific cases where people had done that for building ornament, for making pieces of rope, for doing, you know, textile designs, all these kinds of things.
the idea of thinking about that abstractly has a little bit more of a complicated history. I mean, logic.
was something that was certainly kind of, described by Aristotle, 4th century BC, 1st century BC. The, you know, Aristotle was thinking, there are all these forms of argument, we can have this way of describing arguments work like this. They have a precedent, an antecedent of this that follows with that, and so on.
So, Aristotle had this idea of syllogistic logic, these patterns of argument that could be thought of as sort of rules for specifying how things work.
That was one kind of thing. Then,
it, but… but, you know, that's… that's… is that like a computation? Well, not… not really clear yet. Then…
There were, sort of, things where you were take… trying to take
sort of things from the world and formalize them in that way. I think the I Ching can be thought of as doing something a bit like that, as sort of taking these… these base two numbers, hexagrams and so on, and trying to sort of take things in the world and kind of reduce them to this kind of symbolic, numerical.
thing.
Later on, Ramon Lull in the 1200s had this idea of
being able to sort of figure out what would happen in the world by sort of just doing the combinatorics of different kinds of propositions. That's, again, a sort of pre-computational kind of thing, that there could be an abstract
combinatorics of things that could be done at the level of those abstract rules, independent of the particulars of what propositions you were dealing with. Much like Aristotle had thought about, well, there are these abstract rules of logic that can talk about many different kinds of topics.
Well… Then, I suppose the next…
figure in this whole thing was probably Leibniz in the late 1600s. Leibniz had a somewhat more general idea of computation.
he had… Well, let's see, he…
he, you know, his PhD thesis was about, you know, Solving legal cases using logic.
Trying to… he really had this idea throughout his life of, can you formalize things in the world in terms of something like logic?
Can you make something where there are abstract rules whose consequences you can then work out? And I suspect that's how he thought about calculus, his version of calculus, in the same way.
Now, one thing Leibniz, well, didn't yet have, and that took a long…
time further to develop is the idea of a function. The idea that you could have a thing, like an F, and you could say F of X, X is kind of anything of a certain category of numbers or whatever

else, and F is something that does an operation on that thing, and then you could sort of talk about F itself.
independent of the actual operation of F. So, for example, Leibniz certainly understood polynomials and things like that. He could write down, you know, X cubed minus 2x plus 3 or something, but the question of what… of a function, f of x.
Whose operation is…
you know, cube and add whatever I said, two times the thing plus 1, that sort of pure function that extracted just the function without kind of the expression that you think you're computing, that did not really exist in Leibniz's time. What did begin to exist with predicate calculus, the ability to say, rather than just
You know, if it's, if… if you drop the thing, it will fall on the floor.
To say something like, for everything that you might drop.
There exists a way of doing this, that, and the other, of having these quantifiers that say, for all, or there exists, that deal with, kind of, infinite collections of things, say, for all
People, you know, all men are mortal.
et cetera, et cetera, et cetera. That was already a syllogism. That structure
was, the overall structure already existed in Aristotle's time, but the notion of sort of quantifying over people, of saying, for all of something, for all numbers, such and such will be true, and so on, that was a thing that was emerging into existence by Leibniz's time in the 1600s.
Okay, so then…
we go a whole bunch further into the 1800s, and this sort of push towards abstraction really jumps in around the 1830s, with several different kinds of abstraction coming in. So there was Boolean algebra.
Where the idea that you could take, so the operations of logic
and make them really like an algebra, something where you could say, well, OR is like plus, AND is like times, and well is a little bit of a glitch, because if true is 1 and false is 0, 1 plus 1, 1 or 1, true or true is just true.
Whereas 1 plus 1 is normally 2, but so Boole had to sort of adjust that and say, well, in Boolean algebra, 1 plus 1 is really just 1. But that was, again, a formalization of these ideas of logic in something that was much more abstract.
There were, at the same time, a couple of other forms of abstraction. Another one was Lobachevsky, and the idea of curved space, the idea that you could have a geometry that was not attached to the actual geometry that we experience in the world, normally, of Euclidean geometry and so on, but that it was sort of abstracted from that.
Then there was the idea of group theory and Galois, bringing that in, where there's something where you can have, sort of, elements that can be multiplied and so on, but the elements aren't really numbers, they're abstract kinds of things that have certain properties.
And then the third one of those was Hamilton and quaternions. Again, a form of… so the complex numbers have been introduced around 1500, and the idea that one could have another type of thing, which was a number-like thing, but had sort of more abstract rules.
I mean, this notion of, well, I suppose a precursor to all of this had been algebra from around the 1400s, where you could have the idea of just the X representing whatever it represents, rather than having to just talk about the specifics of particular numbers.
There'd been a precursor to that, even with Euclid, because Euclid had done things like labeling angles in a figure with letters, and sort of this letter represents this angle. However, whatever the measure of that angle is, is represented by that letter.

So there have been precursors of that, but this sort of, you know, emergence of kind of symbolic representation of things, the abstraction of things, the… it's a… it's kind of an algebraic structure, but it isn't just in terms of numbers, it's an arbitrary algebraic structure. That was really a thing of the… of the early part of the 1800s.
In the later part of the 1800s, there were all these efforts to really say, you know, all this stuff that we know, like mathematics, what is it really based on? How could we really… what are its real foundations? Could its foundations just be logic?
Could you derive all of mathematics from logic? Gottlab Frager was very big on that idea. And then…
then there was sort of this push towards formalization. Can you write down official axioms for mathematics from which you can sort of mechanically derive the truth of mathematics? Piano, Giuseppe Piano was a big person in that in the 1880s, and so on.
So there was sort of a gathering interest in sort of the formalization of things.
At that point, the notion of a function, this just the thing, the function, you could pick it up, you could do things with that function F,
Normally, it was like, well, what is the function… what is the value of the function? Is it X squared plus 3, or something like this? We can do something with the polynomial X squared plus 3, but if you just hand me the function on its own, I don't know what to do with it.
So, as that push towards abstraction, through Frege, through piano, and so on.
as that sort of advanced, it got more to the point where you could just sort of pick up a function and do something with it. So functions became sort of things that had structure of their own, not just sort of the value of the function is this concrete thing like a polynomial.
So then, around 1900 and so on, David Hilbert had this Hilbert program.
Which was, sort of, can one
Can one mechanize mathematics, in a sense? Can one turn it into something where you could write down axioms and then just sort of grind out the results? It's beginning to seem a lot like computation at that point. It's beginning to seem like, can you write down rules for mathematics and then just grind out all the theorems of mathematics from that? Something that sort of has the general character of computation.
Well, then Whitehead and Russell did their big, sort of show-off exercise around 1910 of producing these books, Principia Mathematica, which tried to sort of do that from the machine code of mathematics that they viewed as being essentially logic. Could you build up
Kind of to, you know, 1 plus 1 equals 2, they took them 100 pages or something.
to do that very, very show-offly, so to speak.
that, it was kind of from the machine code, kind of like a modern proof assistant system, like Lean or Coq or whatever else, where you're building up from the axioms to get to 1 plus 1 equals 2, and it's a similar amount of effort, maybe even more effort, in a modern proof assistant than it was in Whitehead and Russell's Principia Mathematica.
But by this point, sort of, there's a higher level of abstraction. You can kind of take functions, you can take operations you're doing, you can sort of pick up that thing that represents a collection of operations and abstractly talk about it.
Well…
Then, I suppose the next big step… so there was a question of, well, is there raw material from which we can build all the operations of mathematics? We have plus, and we have times, and so on. We have the construction of a function.
We have things like mathematical induction, which allows you to deduce this from that.

But is there some way… is there some sort of raw material from which any function that shows up in mathematics can be built from sort of raw primitive elements? So the original idea was to use this concept of primitive recursion.
The idea that if you have a function f of n, you can define it in terms of, let's say, f of n minus 1 and previous values of F. It's something where, given this function, you can just always sort of roll down and define it in terms of things that you sort of already could have looked at. Same idea of mathematical induction.
Given that you have a certain sequence of propositions proved, can you prove the next one? You always get to the next one and keep going forever. Well…
That was a… people thought in the beginning of the 20th century that sort of all reasonable mathematical functions could be built using primitive recursion. Around 1920, a student of Hilbert's named Wilhelm Ackermann.
showed that there was at least one function, the so-called Ackermann function, which can't be built that way, but it seems like a reasonable mathematical function. It's kind of the function that, with one parameter of 0, it gives just adding one to things. Parameter 1, it says X plus X plus whatever, X plus Y. Parameter 2, it's X times Y. Parameter 3, it's X to the power y.
parameter 4, it's titration, X to the X to the X to the XY times, and so on. As you keep on going along those lines, you can show that there's no way to derive the values of that function purely by this look-back primitive recursion type technique.
So… That was kind of a, okay, so perimeter recursion is not the raw material for all of mathematics.
Then, 1931 rolls around, and that's when Kirk Godel proved his incompleteness theorem. The technical thing that he did was to define so-called general recursive functions, which are functions which, instead of just always looking back, they have a thing where they can just say, just keep testing this until it's true.
so-called mu operator in what Godel did. It's just keep going until something happens. And it could go as long as you want.
So Godel constructed that. Then his big result
was that you could encode that and a lot of other kinds of things that let you do, sort of, talking about mathematics, you can encode those things in terms of actually just arithmetic. You could write down mathematical equations where the solving of the equations was equivalent
to the doing of all these operations and recursion and all this kind of thing. And that's how Godel showed that, in the end.
there would be that this Hilbert program, where you could just mechanically work out all the theorems of mathematics, that wasn't going to work, because there were theorems that were sort of arbitrarily far away, and there were… there were only… there were theorems that you just couldn't reach
From any given set of axioms.
So, implicit in what Godel did was showing that you could compile all sorts of mathematical statements, including the statement, this statement is unprovable, that metamathematical statement, you could compile those things into statements about arithmetic. So he built, in a sense, the first compiler
in a very obscure way, and he made use of things like the general recursive functions idea to do that. But that was a place where he was taking, kind of, something which was a piece of mathematics.

like… arbitrary kind of statements in mathematics, and he was grinding it down into raw material that
was… that was… that was sort of a universal raw material. So a case where you're kind of taking… now that we… now that the idea of functions, abstract functions existed, it's like, well, what can you make those out of? What can you make the things that you do mathematics out of? So… That was sort of that version. 1936 comes around, Alan Turing…
kind of tries to do something related to that, actually tried to solve the Enscheiden's problem, the Hilbert's decision problem, which was essentially a mechanization and mathematics problem, and comes up with the idea of Turing machines, which are a sort of much more concrete and mechanical
way of thinking about, sort of, the applying of rules to do a computation. I mean, Turing was basing what he was doing on how bank clerks would kind of get data from some bank some sort of… well, he called it a tape, but some… some sort of piece of paper or something, and they would do operations, and then they would write that back onto another piece of paper, put that back in the file cabinet or whatever, and keep going. That was sort of the thing that Alan Turing was idealizing in his Turing machine construct.
once… so the Turing machine was very much more directly, you can take
things which are sort of operating according to rules, just like people had sort of told their armies to do, go into this, the tortoise formation, or whatever it was, and, you know, operate according to rules, do this kind of thing. But he was doing that, giving sort of an arbitrary sequence of possible rules to do things with.
I should have… I should have, added in two other pieces to this… this tale. So, okay, so then, let me just finish on the… the Turing strand.
So, once one got to Turing machines, there was a notion, then, of abstract computation.
It wasn't well understood, but that notion did exist by that point.
I'm going to tell you about a couple of precursors to that, which weren't part of the main strand of history, but which were real precursors to that. But given that.
Meanwhile, the technology of mechanical calculators had advanced a lot. There were electromechanical calculators, and then, by the 1940s, there started to be, we'll just use electronics to build, sort of, things that can calculate.
And then there was the idea of the stored program computer, the thing which is, sort of, it's just…
something where the specification of what operations you do is stored just like the numbers you're operating on. So that's where the sort of code equals data comes in. Although, code equals data is already a thing in Turing machines, it's critical to Turing machines, that sort of code and data are the same kind of thing. But as a practical matter, that sort of emerged in electronic computers
in the 19… by the… by the… well, 1946 was the ENIAC,
By the end of the 1940s, beginning of the 1950s, that was sort of a practical thing done with engineering in electronic computers. Those things were a little bit more brought together by John von Neumann, who was a mathematical logic sort of enthusiast, who actually worked on practical computers, kind of helped bring those strands together. But by that point, there were practical computers
using the fact that you could store programs as data and do things with them, and there were theoretical computers like Turing machines that also worked that way. So the concept of computation was something that had, I think, firmly emerged by the 1950s.

Now, there are two precursors I'll mention. One of them is Charles Babbage and Ada Lovelace, that's from the 1840s, 1830s, and 1840s.
The, Charles Babbage was kind of following the tradition of mechanical calculators and wanted to make very systematic, large-scale mechanical calculators that would be able to sort of, for example, compute tables of logarithms and so on. So he built
His difference engine that essentially computed, polynomials using cogs and so on, and,
That was a… that was the thing he built.
He then imagined the idea that you could make a computer, like a calculator, where the specification of what it was computing was kind of a soft specification.
Where you would have a punch card that would have little holes in it that would say, do this operation now, do that operation now, do the other operation now.
He got that idea from Jacquard looms, which had emerged around 1800, where there were weaving patterns that were specified by punch cards, and some little
sort of probe would go into the punch card and determine how to move the needle that was doing the weaving to make pictures of, you know, birds and flowers, to quote Ada Lovelace.
Okay, so meanwhile, Ada Lovelace got in the picture, and she originally saw herself as mostly an expositor of what Babbage had done with his analytical engine. Babbage was kind of a person who made a big fuss about things and didn't necessarily effectively get projects done.
Ada saw herself as more of a poetess of science and an expositor of science, but what ended up happening was she got the point of what this analytical engine was going to be able to do. And as she rather charmingly says, you know, the analytical engine
will weave algebraical patterns as a Jacquard loom weaves patterns of birds and flowers. So she had the idea that
this was a thing. She had this sort of abstraction of computation idea that this was a way of sort of weaving algebraical patterns, and she thought about taking that abstract notion of computation, applying it to all kinds of things, whether it was to computing the motion of planets, or to composing music.
Or whatever else. She'd written down several of these different directions. So she, I think, really got the idea of sort of the abstraction of computation, the notion that you'd have a machine, a mechanical device that could do this abstract thing of computation and apply it in all these different places. There had been precursors. You know, Leibniz had talked about applying logic likes of things. Ramon Lal had talked about his kind of combinatorics doing that.
But I think Ada Lovelace was the person who first, with some degree of clarity, saw kind of this computation as a universal thing applicable to many different areas, and the notion, implicitly, that there would be a certain set of operations that would be sufficient, that you just need those operations, and you could sort of make
anything
anything computable out of them. She didn't completely have that idea. That idea didn't really emerge until, basically, Turing.
in a completely clear form, but, she had that basic idea. Now, there are two other precursors. One is Moses Schoenfinkinkel from 1920, and the other is Emil Post from 1921.
So Moses Schoenfrenkel was trying to generalize logic. So it had been discovered around 1900 that logic thought about in terms… Booles thought about logic in terms of AND and OR and NOTs and so on.
Then, Scheffer and Peirce and several other people, at about the same time had discovered that you didn't need AND and OR NOT. You could have a single operation that we now call NAND,

which is not a band, and you can make every other logic operation out of combinations of NANDs. And so, that had been sort of another one of these, what can you make out of what? Like, Whitehead and Russell, in the second edition of their Principia Mathematica, were very big on, you can make everything out of NANDs. They wanted to make everything out of as small a primitive as possible.
So that was the thing that was known.
Moses Schonfenkel was a student of David Hilbert's, and he wanted to do the same thing for predicate logic, for logic with… for alls and there exists, as… as people had done for propositional logic to stand and or not with the NAND operation.
So he came up with these things he called combinators. Actually, did he call them combinators? I… yes, he did. That,
S and K, these two operations that are these very weird symbolic operations. I've written a whole book about them. They… and by making combinations of S's and K's and S and Ks and so on. You can effectively encode any computation.
And Schon Finkel understood that.
And he knew that, for example, well, you could construct something like numbers by just saying the number 3 is S of S of S of K, and, you know, etc. And you could then have a representation of the plus operation that's just an S of K of S of S of K, of k of s, whatever it is. I don't know what the plus operation is offhand.
But that you could grind everything in mathematics down into combinations of these symbolic constructs, S and K, and making expressions out of those. What you get is very powerful, very universal, and very obscure. Very hard for us humans to wrap our brains around.
But Moses Schonfenkel
Figured that stuff out, wrote a pretty clear paper about this in 1920, then more or less disappeared. I've written a whole bunch about what happened to Moses Joan Finkel, but That…
Combinators sort of died off. Haskell Curry picked them up a few years later and did a bit with them, but, wasn't… I mean, that was… that was kind of a… a thread that sort of led into, well. Okay, it led into another piece, which we'll talk about in a moment. Another branch was Emil Post.
And we'll post…
was looking at string rewriting systems. Take strings of symbols, and you say, oh, we… every time you see BAA, replace it by BBACA or something.
Many people had taken the Whitehead and Russell Principia Mathematica idea and essentially wanted to go further, instead of to have even simpler primitives for setting up mathematics. That's, Sean Finkel mentions that, Godel was all about that.
Turing has a slightly different branch, but still is thinking about that.
Post really wanted to take it, just… you've got strings of characters, just have replacement rules for strings of characters. Very quintessentially computational.
In 1921, Emil Post thought, if I can just show that all of Principia Mathematica is reducible to transformation rules for strings, and then I can solve the problem of transformation rules for strings, I've solved mathematics. That's what he thought in the summer of 1921.
But then he had managed to grind down Prohibia Mathematica to a very specific problem, about a string of symbols, and you remove the three symbols at the beginning, you look at the first one, you say if it's a 1, you add 1101. If it's a 0, you add…

Is it 000 at the end? Something like that. Very simple kind of description of this sort of string rewriting operation.
Okay, and then he spent a couple of months trying to figure out, so how does this work?
He discovered it was really complicated.
And, in fact, I wrote a piece about this in 2021, the 100th anniversary of that event. And we still don't know what happens to Post's so-called tag system. We still don't know whether it goes on replacing symbols forever or eventually stops.
So, post,
It depends on what it started with, whether it always stops, or whether it can go on forever. It can go on for a really long time, but can it go on forever?
Well, Post, sort of, at that point, kind of gave up. He said, I'm not going to be able to solve mathematics this way. He had kind of gotten precursors of Godel's theorem and, and sort of the Turing ideas, but didn't quite get there.
One more piece to this puzzle. So, so POST's kind of stringer writing systems reappear in early text editors a bit later, but really didn't become a thing for how you think about computation until a lot later.
I mean, my own efforts in transformation rules for symbolic expressions, I was certainly aware of posts, transformation rules for strings, and thinking about that kind of thing.
Okay, there's one more piece to this puzzle. Early 1930s, Alonzo Church.
Taking ideas from Schonfinkinkel invents lambda calculus.
actually, a thing like lambda calculus that existed in Prokibia Mathematica, what Church called lambda, I believe, Whitet and Russell called IOTA. But a lambda is sort of this pure function.
Lambda of X, X plus 1, means
whatever you feed me, replace the X with what you're feeding me, and evaluate the body X plus 1.
So Church had this idea of… of having sort of disembodied functions, and showed that then it turns out that that was equivalent to Turing machines, et cetera, et cetera, et cetera. That was the beginning of the… all these different models of computation are going to be equivalent.
So, that was this, this notion that computation would be a, you know, there were these abstract models of it, there were practical computers.
There was a… you know, people were understanding more and more that you could do with computation. Could you use computation to approximate physics? Well, you could take the equations of physics, and you could try and discretize them to fit them on a computer. That was all happening.
Well, what wasn't happening was thinking about just how broad is this idea of computation.
For example, if you'd asked, even in the beginning of the 1980s, Essentially, all physicists.
is…
the universe, physical universe, computational, they would have said no. They would have said, computer's about discrete things, integers, bits, and so on. The universe is about continuous real numbers. About, you know, you can be at this position or one arbitrarily small distance away. They just didn't think computation was relevant to physics.
Meanwhile, Godel, for example, had very much thought that his pre-computational ideas about, sort of, this were not relevant to minds. He has this… he kind of thought there's a way that minds will, while
the things he's describing sort of work in terms of integers, and sort of the things you can do with integers. Godel had this idea that it was some sort of… some sort of… maybe some kind of

ramification into the transfinite that is done by minds, sort of every day, all the time, that wasn't a thing that he could talk about in terms of the abstract structures he was dealing with.
So, by the 1980s, it really wasn't a thing yet.
that computation was broad enough to encompass the mental world, the physical world, and so on. People had thought back in the 1950s, oh, we'll make sort of brain-like things out of computers, that was the origin of AI in 1956. And, you know, things had happened, but it didn't work that well, it wasn't going gangbusters, and people still thought there's more to minds Than we know in computers.
Well, then my own efforts in mid-1980s, I kind of started studying, sort of, the universe of possible simple programs, and started studying something which really had not been studied before, which you can think of as kind of computation in the wild.
People had imagined constructing programs, or the mathematical logic equivalent of programs, to do particular things.
And the idea that you could just pick a program at random, or start enumerating all possible programs, and say, what do all of these do? People had imagined enumerating all possible programs as a theoretical matter, but what do they actually do?
You know, when you run them, what do they look like? What do they… what actual behavior do they have? People had done a few things along those lines in the early 1960s, hadn't found anything terribly interesting, and gave up.
So that was… that was sort of my big break in 1981 or so, was starting to study simple programs and what they actually do.
And the answer is they do amazingly complicated things. The intuition that you need a complicated program to get complicated things to happen is just not a correct intuition. Even very simple programs can do very complicated things. And so that, for me, was a big tip-off that when we look around at the natural world, and we see complicated things going on, well, those could come from programs, even very simple programs, but from programs. And so we really should think about, maybe the whole world can be thought about computationally.
And that's kind of a thing that I initiated at that time and came up with ideas like computational irreducibility and so on. The sort of physical version of Church's thesis was kind of my construction from around 1984 or so.
But was, at the time very much not, not a thing that physicists would think about. Sort of in a parallel track, David Deutsch, around the same time, was thinking about quantum mechanics, which was sort of already known to be discrete, and kind of coming up with, sort of, the physicalization of the idea of computation in the context of quantum mechanics. Somewhat different branch.
But, the, the thing that, so, I mean, the sort of universality of
the significance of computation was something I would say that, for me, emerged in the 1980s. I think other people sort of gradually got that idea. Even when I published my book, A New Kind of Science, in 2002, plenty of people were saying, well, it's all very nice that you can have these programs do these complicated things, but of course we know, when it comes to physics, for example, we're writing down differential equations and things like that.
In the 20 years that followed that, I would say there's been a pretty thorough transition to the point where people know models that you make of things are often done with programs.
Implicitly, therefore, these… sort of computation is really the thing we're dealing with, not sort of continuous mathematics and real numbers and things of that kind.
And of course, with our physics project from 5 years ago now.

The realization that, really, the physical universe is presumably computational all the way down. That, really, we can think about everything that happens in the universe as the result of a computation. That's kind of the next level of really understanding that computation is a significant thing. That was a long description of the idea of computation.
Let's see…
Sylvia comments, didn't Conrad Zuza in 1967 propose the universe is running on a cellular automaton, a discrete computational grid?
Conrad Zuzer was a funny fish. I mean, I exchanged letters with him in the early 1980s, and I think some of the later things he wrote were kind of based on things I'd said, and it all got very confused.
In the 1940s, Conrad Zouza
had worked on early computers. There's sort of a big mystery of what happened to the computer that he built during World War II that was in a house that was bombed, and where is it, and urban archaeology, and I can tell you a long, elaborate story about our efforts to kind of raise Conrad Zouza's computer.
And, and what really… and sort of the scurrilousness of why that doesn't… why people don't want that to happen, perhaps, all those kinds of things.
But Karnat Zuza then started a computer company in Germany,
I certainly talked to Conrad Zuza's son about, sort of, what did Zuza know when about touring, about things like this, during the war and after the war, and it's all a bit muddy, a bit murky. what he knew, and I think it will be fair to say, and I have to say, I've run across a number of people who are not
Zusza enthusiasts in terms of sort of what was said versus what was real, and so on. It became a complicated issue because, particularly in the 1990s, there was sort of a very nationalistic effort, I would say, particularly in Germany, to say, we invented the computer.
Which, I don't think is really correct.
You know, I think the computer is actually, if… in its electronic form, is really an American construct.
But in any case, the,
I think that, in, so…
Zusa had this idea of what he called calculating space.
It was the idea that you could sort of make a discrete grid of cells in space, and you could then compute on that, you could… and then that might represent physics, and he had the idea of having real numbers.
That idea was a very old idea. That idea was the idea of discretizing a partial differential equation. That idea was known, certainly by the time of World War II, that was an idea that was known, I think it was used, the sort of
early versions of that idea with human computers were used famously in the design of the aerodynamics of the Spitfire plane.
But, the notion that you could discretize space and sort of have real numbers at every point as an approximation to physics equations was a well-known thing. Now, something that actually just occurs to me as I'm telling you this story is something I had known only quite recently, is that Werner Heisenberg, along with most other physicists in the early part of the 20th century.
Had believed that space was discrete, and had tried to kind of construct models of the universe based on discrete space.

So, I know by 1931 or so, Heisenberg had kind of… around that, 1932 maybe, had… had really pushed this idea of discrete space and building things on discrete space, but couldn't make it work, couldn't make it consistent with relativity.
And that was what led him to define the S matrix and what's now modern approach to quantum mechanics. The question that I have is, did Werner Heisenberg know Conrad Zouza? I think the answer is almost definitely yes.
So that's a link I'd never thought about before, because Heisenberg actually died in 1976 or 1977. I, could have met him if I'd been less of a…
if I'd been a more, I don't know, if I'd been… had more of a respect for history when I was 16 years old. But in any case, the,
But Heisenberg was absolutely around, and I think must have interacted with Zeuser. It's not that big a, you know, that stratum of, kind of, scientific Germany
particularly post, post-war, was, you know, was… was… had to be thin enough. They have to have interacted.
So that's kind of interesting, isn't it? I hadn't thought of that before, but Heisenberg absolutely had that idea, as did other… as did many other people at the same time, that space would be discrete, you'd sort of build up the fields of electromagnetic fields, or whatever it was, on space.
So I think that's a… so I don't think Karnazuza added much.
And, I certainly didn't learn anything from his calculating space piece. It seemed rather, rather… Frankly, naive to me.
And it's sort of a little frustrating, because… because, you know, at various times people say, what about Karazzuza? It's like, I don't think there's a whatabot there. You know, he did what he did, he built a computer company, et cetera, et cetera, et cetera, but I don't think this was a thing.
Let's see…
So, question here…
So we're commenting about, Ada Lovelace.
Okay, there's a…
Okay, Jammy asks, do you think if Babbage and Lovelace had actually built the analytical engine, the computer revolution would have happened a century earlier? I think a bunch of things would have happened earlier, yes.
I think, you know, they were… had this friend, Charles Whetstone, who was an electric… electricity guy. I think they would have made the analytical engine not powered by steam, but by electricity.
They thought it would be the size of a large locomotive. It would have been cloud computing in the 1840s. And, you know, they really had the idea that they would, you know, somebody would send in, oh, I want to run this batch job of computing nautical tables, or I want to make these insurance tables, or whatever else. I think there would have been a lot of practical computing, and I think that would have probably
Built from that. Well, I mean, the theoretical side of it
you know, Ada Lovelace wasn't that far away from those ideas. I mean, it was… the abstraction that came in the late 1800s would have been important to kind of help build more scaffolding for that, but I think the path was already laid down. I mean, if Ada Lovelace hadn't gotten cancer
And, and died young.
The history might have been very different in that regard. Charles Babbage was not really on that path. He was much more of an engineer, you know, figuring out how the cogs would interact and so on, than thinking about the big picture of what was going on.

Graham asks, do you think the history of computation is moving towards a point when the line between natural processes and design computation disappears? Well, I've certainly thought that for a long time. I mean, I was very struck years ago when I visited
Leibniz's archive, seeing his brass computer, and realizing in Leibniz's day, that was kind of the only computer he'd seen. Now we've got billions of computers.
But the thing, you know, what… to us, now, looking back at Leibniz, it's like, oh my gosh, that was the only computer he'd seen. He had no intuition about computers.
My guess is that in the future, everything will be made of computers. That every piece of what is now just a material, like a piece of plastic or something, will be something which, like biological tissue, is really computing things.
And I think that, yes, the, you know, as we get into molecular-scale computing, which is what biology succeeds in doing, as we manage to harness that as a piece of technology, we'll see much more of, yes, all natural objects, all objects in the world are computing all the way down to the level of their molecules and so on.
Rather than just… I'm just sitting here as a piece of solid… solid stuff.
It'll be a different kind of thing, and probably the particular place where that's important is in the interface to us, because we are molecular computers in our biology, so to speak, and interfacing, kind of, the world, the technological world, as a molecular computational world.
to us is surely going to be an important thing. So yes, I do think that what is… I mean, in my own view of how the natural world works, we should think about it computationally, and that's been a powerful idea. There's much further for that idea to run, but that's different from, sort of, the technologically constructed computation.
Let's see
Josh, there are lots of interesting questions here, right? Let me pick a few, a few here.
Graham is asking, in your own career, you've seen multiple eras of computation, from mainframes, PCs, internet, cloud, AI.
What transition surprised you the most?
I have to say, nothing has been that surprising so far. I mean, it's kind of like, from the first time I had access to a computer and it was very big, I realized it's going to get smaller. And, you know, from the first time that I saw a… well, okay.
I don't think I completely imagined bitmap displays.
I think, you know, I'd seen character-based displays, I'd seen sort of pen-plotter-type displays. I don't think I internalized that. I think the main thing I haven't internalized is when you have a piece of raw technology, what happens when you build a giant tower on top of that technology? It's the same thing, you know, if you imagine combinators from 1920, nobody would have imagined that you could build, you know, a video conferencing system on top of what is computationally the same as combinators. You know, all these use cases are very surprising, and the extent to which you can build these very tall towers and where the towers go is often surprising.
I mean, I think that, for me, kind of, I would say by the early 1980s, sort of my own experience of computers has not been that different since that time. I mean, many details, you know, I could run a program on my watch now, I can do, you know, all kinds of things like that, but it's not qualitatively different. You know, this… this notion, this very robust notion of computation, I think, as a practical matter.

for much of what I've done already existed at that time. Even, you know, even the fact that there was sort of this network out there where you could get information and so on, that already existed, albeit in an attenuated form relative to the web and so on.
Let's see, John asks,
Do you think people in the 1940s and 50s worried about computers taking over the way people talk about AI today? Yeah, absolutely. I mean, giant electronic brains, we're going to have giant electronic brains that out-think us just as bulldozers out…
muscle us, so to speak. Absolutely, that was a thought.
From, from certainly the 1950s.
And, you know, people imagine that AI would take off in the early 1960s, and it would be kind of… people were writing in the early 1960s, you could… you could take what they wrote and transport it to today with a few changes of societal commentary, and it would seem completely modern.
It was all about how maybe we're building the artificial intelligences that will be the next stage in the evolution of, sort of intelligence on our planet type thing.
It's… it's, very much the same thing, and that showed up in science fiction, it showed up, all over the place as, as sort of a common
theme of, yeah, the automation is going to take over. It's, in the end, it's a weird thing, because automation on its own doesn't know what to do.
I mean, nature is automated. What happens in the natural world just happens. Nobody is sort of making it happen, like we try to make technology do things. Nature just runs. And so, so too will any, you know, ultimate automation just runs like nature.
The question of what, you know, whether the things we build as technology, almost by definition, are things where we're defining the goals, and then it's trying to automate those goals.
So I think, Let's see…
Question here from Sylvia. Insight into how Ed Fredkin interacted with Dick Feynman around the idea that computation could be a substrate of reality.
Well, I knew both Ed Fredkin and Dick Feynman well, and I talked to both of them about each other, and I think I know that in a decent level of detail. I mean.
the… Dick Feynman.
I don't… would…
Was still a believer in very much physics as physics was thought to be, with quantum field theory and all that kind of thing.
he did pay a certain amount of lip service to, maybe we should think about this computationally, but he didn't really take that that seriously. I mean, when I was starting to work on simple programs and things, and showing him the results from that,
I would say we didn't talk much about how this was relevant to physics. It was mostly about the thing in itself. I think that connection was not really being made there.
Ed Fredken had a very, kind of, I would say.
Sort of a high school level physics kind of way of thinking about physics, but he knew a lot about computation.
Or practical computers. He's like, let's… and he knew a certain amount about how fairly simple programs can do somewhat complicated things, but the things he wanted the programs to do were, like, make electrons.

And it was all a bit muddled, what it means to make an electron. And for him, kind of the things that were the laws of physics that describe fairly… in fairly straightforward ways what electrons do was the things he was trying to get from simple programs and so on.
was sort of a different play, but for Ed Fredkin, the idea that the universe would ultimately be sort of made of discrete cells and like a computer was absolutely what he believed.
And I… I must say that I didn't believe that, and I always got a bit frustrated with Ed starting in the early 1980s, when he would sort of say, but all this is relevant to physics, and then he would show this, you know, collection of three cells on a screen moving across the screen, and say, that's like an electron.
And I'd be like, but it's not like an electron. That's not how… you know, we know a lot about the physics of electrons, and this does not capture that.
And so, it was, for me, sort of the realization that
simple programs could be relevant to fundamental physics, kind of came through a lot of understanding about what simple programs can really do, the really complicated things that simple programs can do, which is not something that Ed was really thinking about. Ed wasn't interested in the complicated stuff, Ed wanted the stuff
Where you could interpret it using, sort of, the simple physics of electrons goes from here to there.
Let's see…
Double triple asks, have you read Max Son's Master from 1894, a story about a robot that kills its master and takes over? I have not. I mean, I have to say that the term automaton, which sounds very modern, actually dates from the 1600s.
people were talking about automata, particularly toys, that would be, you know, a music box-like thing with little dancing figures on the top and so on, or a… or a duck. There was a famous duck that was kind of a mechanical duck that did all kinds of things, like, including quacking and walking and so on. And, that was… those were automata, usually for toys, or for displays of various kinds.
And they were a… they were a thing from the 1600s, so this idea that you could kind of… you could mechanize a duck-like thing or a human-like thing was very much a known thing. Then there was the Mechanical Turk.
Of, you know, the box that supposedly played chess, that people…
you know, found it plausible that there could be a box that played chess. In fact, it was a small person inside it playing the chess. But it was still, you know, this idea that you could sort of make automated humans
was… is an old idea, and, you know, that shows up in, well, in early science fiction and things like this, in the, what was it called? The Rossum's Universal Robots, that's from 1920.
Oh my gosh.
1920s, CAPEC.
Carol Kaeperck, I think, a filmmaker.
And, you know, hence the robot… robots, which is from, ultimately from Polish for worker, was, you know, came from that time. But I think… so… so that idea that, you know, that there could be robots, that's a very old idea.
Let's see…
Elsie is asking for and thinks about computation in this very inclusive sense. Anything the universe does is framed that way. How do you keep the useful, kind of, symbolic manipulation distinction? Maybe that's a little bit off-topic for today, but I think the thing to say is.

sort of… There's computation happening in the wild.
There's what we can understand in our minds.
And there's the bridge between those things, which is basically what science does.
Science is trying to take what goes on in the natural world and make a representation of it that we can play in our minds. And that requires this kind of compression that's associated with having sort of symbols that represent things. And so when we talk about what happens in the world, we'll do it in terms of things like symbols that are our representation that's useful for us thinking about things, writing programs, and so on.
Let's see, truly asks, how did the history of physics influence the history of computation, and vice versa?
Physics and computation were pretty separate.
I mean, there were physicists who needed to compute things.
You know, Kelvin had a tide calculator in the late 1800s, which was a mechanical calculator. Hartree had a thing made out of Meccano that was a way that he used to solve equations with analog computers. So, physicists have been using computers of all kinds. You know, Kepler was a friend of Kepler's who made the first modern sort of cog-based computer in the 1600s. So, physicists have been using
sort of com… Computers to particularly do things with numbers for a long time.
The idea that
computation would be a conceptually useful thing for thinking about physics. I'm pretty sure that I was the main person who kind of introduced that idea. I think that the notion that, you know, ideas from mathematical logic might really relate to physics
that was not a thing. I mean, that was a thing that probably some of the stuff I wrote early 1980s kind of made that point.
But, as I say, most physicists wouldn't have believed it, Godel wouldn't have believed it, Turing wouldn't have believed it. I mean, when Turing was working on biology late in his life, in 1954 or so.
You know, he immediately thought, if I'm going to model reactions and diffusions in biology, I'm going to do differential equations. I'm not going to even think about Turing machines for that. I mean, there were other people, like Aristot Lindenmeyer, who I met when he was quite an old chap, who was a botanist.
had this kind of idea of L systems, which were kind of systems or rules for representing the growth of plants and so on. He had more of the Turing machine-like idea, in, I think, by the… I think that was the 1970s, early 1980s.
For thinking about that. But that was, you know, those were… that was… it was not in physics. Remember, that was biology. And biology, by the way, had had the kind of digital shock of discovering that DNA was digital in 1953.
Physics had not had a digital shock.
you know, it had had one when atoms were discovered, and when quantization was discovered in the early part of the 20th century, and it had, you know, there'd been a thought in the early part of the 20th century, physics will all go discrete, but that hadn't worked out. So by the 1970s and things, physics was just this thing that was about equations and so on, and it really was not… You know, it was not conceptually connected to computation.
Now, computation versus physics, that's an interesting question, too. There's… when people were thinking about, oh, we're just going to calculate things with computers and make electronics to do this, people were not thinking about that in terms of space and time and so on.

They were thinking about the mechanics of computers, they were thinking about things like noise levels and so on. That was a physics injection, but mostly they were just thinking about calculating things without any regard for the structure of space and time and so on.
By about 1970, people were starting to think about, well, just how fast does that algorithm run? How much memory does it take? And then people were thinking by that time, as microprocessors were coming online, it started to matter how long is it going to take the signal to go from here to there? There was sort of a physicalization of thinking about, sort of abstract computation that started to happen.
with computational complexity classes, about time complexity, space complexity, thinking about, sort of, the actual layout of microprocessors and so on. That was a thing of the 1970s, of sort of a beginning of injecting a little bit of light physics into a light conceptual physics, into thinking about the abstract computation, but it didn't really take off. And I would say that, I mean, it had its…
That is a thing in computation theory, but a general injection of ideas from physics into computation has not really been a thing. I mean, that's something that is now, with our physics project, and seeing that physics is sort of deeply computational, you can start taking physics ideas and importing them into theory of computation. In fact, I have a big project doing exactly that, which we'll see how it works out.
But it's kind of a physicalization of theoretical computation that I think has the potential to be really interesting and to address things like the P versus NP problem and so on in a very different way. But that's… I'm afraid that's… that's a thing of the 2020s, I haven't gotten there yet.
Let's see… Weasel is asking, what's the oldest automaton that I have actually seen with my own eyes?
Otherwise seen. Well… I'm not sure. I think I've seen some of these clocks.
And, N…
Switzerland, maybe? I'm not sure, that are these very ornate kind of… kind of things. I don't know, it's a good question. I saw, actually, there was an exhibit of automata
Hot.
a museum that I saw fairly recently. I don't know the answer to that. I should know the answer to that.
Okay.
Let's see…
Are there relations between Cantor sets and Turing machines? asks M. Rudeau. And there was a previous question here from KD about, can you tell us the story of Gorg Cantor and the continuum hypothesis?
Yeah, that sort of relates to, to some of what I've been talking about. Georg Cantor was… was 1870s, I guess, and,
Originally, he was doing very practical things about looking at trigonometric series. Sine 2x plus 3 sine 3x plus whatever plus whatever, infinite series. What do they converge to?
And he discovered they can be incredibly wild.
And that led him to several constructs. One was the Cantor set, which is kind of an early fractal. And the other is set theory. Thinking about, sort of, abstractly, what are sets of points that could be the limits of these trigonometric series.
From that, he also came to transfinite numbers, and really was one of the people who was driving the sort of abstraction of things in the 1870s, 1880s.
He,

his work on transfinite numbers and so on sort of went off into the mathosphere, so to speak, and hasn't really been brought back to applications. I've… I've recently been thinking about some applications of transfinite numbers to essentially classify infinite computational data structures, but generally they've been the thing that's very much just in math and never been applied.
set theory became sort of the standard for, you know, this is what we're going to base our math on. It's actually very weird that people think that, because set theory is based on axioms which are quite unintuitive.
And they're talking about infinite sets, we don't really have intuition about that. Why is it set theory rather than some other form of set theory with other axioms? Nobody can really say that. There's no ground truth. It's not like physics is based on set theory. It's not. Set theory is a way of describing an abstract system, and the set theory lives inside of universal computation.
Physics plugs into universal computation as well.
I mean, you can say, well, how far does set theory get in kind of formalizing things? You can actually make up computational systems like Turing machines. You could say, okay, I'm going to have this Turing machine. I want to answer a question about the Turing machine. Is this Turing machine never going to halt?
Well, you can make a proof of that, but you… the proof relies on axioms, and sometimes the axioms of, let's say, arithmetic will be enough to make the proof. You can also have a Turing machine that's so wild that the axioms of set theory are needed, or even so wild that the axioms of set theory aren't enough.
There's no way, you know, the Turing machine is just doing this infinite collection of computations, and in effect, what it's doing is, it's something I actually only understood very recently.
is in that infinite computation, it has sort of a microcosm of the whole of set theory. As it goes and does its infinite computation, it is, in effect, enumerating every theorem of set theory.
And so, to ask whether the thing will ever halt.
is to essentially ask the bigger picture of set theory, like, is set theory consistent? Which is something you can't answer from within set theory, and that's kind of why you escape those proofs. Anyway, the,
Turing machines versus Kanto sets, and so on. Kanto sets are, well, the classic way of doing it, you have a… you have interval 0 to 1, you say, cut out the middle third of that interval.
Okay, then with each… within each third that's left, cut out the middle third of that, keep going down. You get this kind of tree-like construction, where you've got thirds of thirds of thirds of thirds of thirds, and then the set of numbers that are still left are kind of the leaves of that infinite tree.
And so is that relevant to things like Turing machines? Well, yes, because the topology of that infinite tree is like a sequence of digits.
So you can imagine, you know, you have binary numbers, there's a… you go 1 goes right, 0 goes left. You can make a tree, every binary number corresponds to a path down a binary tree. And so you can think about the raw material that Turing machines deal with on the infinite tapes of… which have zeros and ones on them.
as being elements of Kanto sets. They are points in the Kanto set.
So in that sense, the topology of a Turing machine is like the topology of the Kanto set, and you can do a certain amount of analysis about, well, for example, cellular automata are continuous maps on the Kanto set.

And you can, you know, the geometrization of a Turing machine is a little bit like thinking about a Kanto set. I have never quite figured out how to get more mileage out of that. I have worked on that at some length, actually, in the 1980s. But I don't have a great thing to say about that.
Let's see, maybe a couple more questions, and then I should wrap up.
well, I'll, question here…
Boy, I'm just going to make a brief comment on this, but maybe I'll talk about this more some other time, from Rebel. How will scientists carve their names in history and future? Till now, it's like, you know, getting a Nobel Prize or a Fields Medal or something, or having their name attached to something. You know, I think the thing to realize
Is that… You know, in the end, ideas are what survive.
And whether… A name is attached to an idea, is…
is a completely hit-and-miss thing. It's often misattached, it's often not attached when it should be, it is attached when it shouldn't be, et cetera, et cetera, et cetera, but it's the ideas that survive. You know, the humans disappear, and perhaps their stories are known. Often the stories told about scientists are very idealized stories, that don't really reflect the intellectual
Sort of path that those people took.
Let's see… Oh, gosh, there are a lot of interesting questions here, my gosh.
I think I, I went through, let's see… So, question here… Saying, correctly, that,
Where is this?
It's a question.
Somebody asked, commenting.
Oh, boy. Origami arsena, I'm gonna do this, talk about this another time. How do you determine what is a fact or fiction in history? How do we know who's real and who is just a person in a story or symbolic? Yeah, I mean, you've got to go look at the original documents. That's the only way to tell. And if you're dealing with, you know.
places where, like India, where there's been largely an oral tradition going back a long way, it's really hard to tell what happened.
You know, it's, it's… I could have mentioned in the story of the formalization of things, Panini, a grammarian of Sanskrit from ancient India.
When did he live? We don't know, within centuries.
What do we know about him and how he came to write his grammar of Sanskrit? We don't know, it wasn't written down until the 1600s. The only thing that's claimed to be known about Panini is that he met his end being eaten by a lion.
May or may not be true.
It's, you know, we really know nothing about the person there, and that's typical of oral traditions. We know the ideas, we don't know the person.
Robot is asking, you mentioned some upcoming travel. Will you be exploring any interesting history or historical places? You know, it's good you remind me about that. I am going…
to Europe for a few weeks, briefly to Italy.
Switzerland, and for a while to the UK, and actually, I had it on my list to go check out Jamie Maxwell, James Clerk Maxwell's estate in Scotland, because I will be there briefly. I need to do that. I had forgotten. I hope I'll be driving right by, in which case I'll definitely
sort of drop in. I don't know if there's anything there, but, it's… by the time
James Clerk Maxwell spent a number of years on this kind of country estate that his family had had in Scotland, and he wrote a lovely paper called On Hills and Dales, which was, I think, based on his daily walk around trying to work out, you know, how many peaks, how many troughs do

you have in the surface? It's kind of an early version of Morse theory, and I kind of have to see if I can recapitulate his walk.
I think I'll, I'll…
drop in at things like the Science Museum in London, I mostly have an intense, ridiculously intense schedule of talks and meetings and things that I've been kind of saving up for probably 15 years or something, and finally, okay, I'm doing this trip, so let me… let me do these things now. It will be absurdly intense. But, I think I did arrange to drop in at the Science Museum in London.
I have to say, I was looking at the floor plan of the Science Museum, and was realizing that it's very similar to what it was like 60 years ago when I was first visiting there.
And I suppose if you have, you know, the original steam engine and things, it's still the original steam engine, and people are still interested in seeing it. I'll be interested to see if my grandfather's bicycle
which was, strangely, ended up in that museum. He lent it to somebody, and that person failed to return the bicycle, but then the bicycle ended up in a museum instead.
whether that's still on display there. It certainly had been for many decades. It was a notable bicycle because it was a bicycle that had no chain. It was… it was driven by some kind of, rotating thing. My grandfather was just a purchaser of such a bicycle, not, but, I think,
It, somehow it wound up in the Science Museum. But in any case, the,
Yeah, you, you guys are reminding me that I should, I should check out my, what historical sites should I see. I'll, I'll be,
I'm trying to… think,
yeah, it's a good, good prompt for me. I need to… I always like seeing these historical kinds of things, and somehow.
often these little tiny museums that exist in particular places are, you know, have very interesting content that hasn't sort of made it out onto the broader web and so on. I think,
I don't know, Kelvin is, William Thompson, Lord Kelvin, is, somebody, another person who I, I've studied quite a bit recently, and maybe I should look for… I bet there's a… he was pretty famous in his day, you know, he was buried in Westminster Abbey and all those kinds of things, which is always a sign of fame in one's day, and I don't… I don't know what there is from him. I think,
it always used to be the case, yeah, I'm… I'm now, I'm thinking about, unfortunately, I will not have very much spare time to go, go look at interesting,
interesting historical sites, although I'd like to do that. And thank you for prompting me on this. Great.
Well, thanks for joining me. I think I should go back to my day job here. I probably will be able to do another livestream this Friday, and then I won't be able to for a while after that.
Unless maybe I wind up in some really wonderful place with an extra hour, at the right time of day in the UK, and I'll be able to… maybe I should take that as a… as a, hey, that would be cool to be able to do one of these from some interesting historical site. I doubt it, but we shall see.
All right, anyway, thanks for joining me today. Lots of interesting questions, lots of questions I'd love to address more in the future.
So, bye for now.