Hello everyone, welcome to another episode, Q&A about science and technology for kids and others.
And I see we have a number of questions here.
The fun one from Jill.
If the universe had a bug, How would we notice it?
Well, what is a bug?
a… a… a bug…
It's something where you tried to create something, like a program, you had an objective in mind, you wanted it to do this and this and this.
And you think it does that.
But somehow, somewhere, something goes wrong, and sometimes it doesn't do what you want.
So, for example, let's say you have a program that's supposed to, divide one number by another.
And, you know, for almost every number, it does a great job. It gets all those digits right, it churns out the correct answer. But just one in a trillion times, if you get just the wrong number, it will give the wrong answer.
You might say, that's a… that's impossible. How could something like that happen? Actually, it did happen in the early 1990s. It happened to a microprocessor called the Pentium, made by Intel. It was the so-called FDiv bug, and Intel had made all these chips.
shipped them out, they were in all kinds of devices, et cetera, et cetera, et cetera, and somebody discovered that one, and it wasn't a trillion, it was… it was smaller than a trillion, one in a trillion, trillion, trillion times. If you had the exact right pattern of digits in your number, it would give the wrong answer for that division sum.
So that was an example of a bug where the people who designed that trip thought that they created a trip that would correctly divide numbers, but actually there was a very obscure case
Where it didn't quite work, because it had optimized something, and, you know, bugs often show up in practical programming. There are various common sources of bugs.
One is kind of corner cases, where, oh, I thought about the case, every… everything from, you know, any number 1, 2, 3, 4, 5, but what about the case of 0? I didn't think of that. If you put in 0, something goes wrong.
Or very classic early bugs were things like somebody would type in a password, and the thing would expect, you know, 12 characters or less.
But if they typed in a 40,000 character password, something would go wrong, because that was kind of a corner case that hadn't been thought about when the original program was written.
So, there are many types of bugs like that. There are some
types of bugs, particularly when many programs, pieces of programs are running in parallel, and you have one thing happening, and another thing's happening at the same time, and it's a question. You can have these situations where you get one answer if one of the pieces of the program finishes earlier, another answer if another piece of the program finishes earlier. Those are very insidious, difficult to find bugs, because they depend on, sort of, the
the real-time timing of those particular pieces of program. Oh, this one ran slower because something else was running on the computer, or some network connection was slow, or something like this.
So there are…
There are all kinds of, so there's a typical example. So there are… there are bugs where… so one question is, what is a bug? Like, you write a program, the program does what the program does.

It's a bug when the program does something you didn't want it to do, so to speak. And typically, you'd only call it a bug when the program usually does what you want, but only just sometimes in an obscure corner does the wrong thing.
So this question of, you know, how do you even define a bug? It's sometimes when people say, well, I will write a specification for the program, and if the program doesn't follow that specification, then it's a bug.
That's sort of a crazy thing to say, because the program itself is a specification of what the program should do. So you say, well, I'm going to write a specification for the program, but the program is a specification. So what does it mean to write a specification for the program when the program is a specification of what the program does?
What it typically has meant is the program is written in some low-level language, like C, C++, something like this. And the specification is in some higher-level description language that is closer to the kind of thing that a human can readily understand.
You know, in the earlier days, when things were written in the machine code of computers, or assembly language, it was really pretty hard for humans to understand what was going on.
And the… sort of the idea of specifications can be, oh, it's a high-level thing that is a description of what's going on, that a human can read and understand.
And then the question is, does the computer actually follow what that high-level description said in doing all the bits of what it's supposed to do?
So, a typical example of that would be
you write a program in Wolfram language, which is kind of the highest level programming language that exists, I guess, right now, and you… that program has some very high-level construct. It's doing some complicated thing with some network that's calculating some particular kind of thing. You write that in
kind of using the primitives that exist in Wolfram language.
Which have definitions.
And you say, those are the primitives I want, I want the ones that do what the Wolfram language description says they should do.
And then you get some answer. Now you've got… somebody else says, well, I'm going to rewrite that program in C++ or something, or I'm going to get an LLM to try and rewrite that program. good luck, and in some cases, it would not really be possible. But let's assume that you could do that. Then the question you can ask is.
does that C program do the same thing that that high-level Wolfram language program does?
And that's a question you can ask, and you can say, is there a bug in the C program? Is there something that went slightly wrong in the C program that made it deviate from that high-level description of what it was supposed to do?
Another example of this, when you are dealing with compiled programs, so, okay, for people who don't kind of know how this works, the, you know, a computer intrinsically operates… intrinsically,
performs machine instructions. Those machine instructions are extremely low level. They say things like, move this lump of data from here to there, add this integer to that other integer, do their very low-level kinds of things.
And when a common thing to want to do is to take a program that's written in a language like C,
And to compile it, which means to convert the description of the program in that slightly higher level description that's slightly more human-readable down into raw machine code.
And when you do that.

there's a question of, well, does it produce kind of very… does it produce optimal machine code? If you wanted it to do some operation about sorting numbers or some such other thing, does it just do that in a very silly way, where it's comparing every number with every other number to see what order they should go in? Or is it a more optimized thing, where it knows, oh, if I've compared this with that, I don't need to compare that with that yet? I can keep that in
And weight and compare it in a different way.
Or, for example, even when I'm doing those kinds of things, computers have memory, pieces of memory, that operate at different speeds. They have small amounts of memory closely connected to their CPUs that operates much faster, and then there's sort of a hierarchy of different kinds of memory, and so on. There may be things in some computation where you want to… that thing is going to be used very, very, very repeatedly, so you want it in the
in a register very close to the CPU, and other things where it's used less frequently, and so you can let it kind of drift off
into slower parts of memory, where you store it in some place where it's going to take five times longer to retrieve that data if you need it. So, there's this whole technology of optimizers, where you're taking something like a C program, and you are compiling it into machine code, and the question is, how optimal is that machine code?
And so there are all kinds of tricks.
for making… for doing that optimization was kind of a big thing, I don't know, 30, 40 years ago, of sort of optimizers were climbing up the curve of, could you write code automatically that was better or worse than the code a human could write? I mean, back when I started using computers. Oh, what is it? 50… how long ago? 55 years ago now. 50…
33 years ago. The,
You know, people would say, well, if you really want to program a computer in an efficient way, you've got to program at its level, right down there in the machine codes, telling it exactly what operation to do when.
Well, a few decades went by, and then it got to the point where a compiler can take your code and turn it into machine code, and the machine code it turns it into is better and more efficient than the machine code any human could write. It's something where the computer has done all kinds of transformations on the code.
And so on, and it's now more… more better code than you could write as a human.
And…
there are lots of crazy things that happen in that kind of optimization. For example, a thing that's sometimes done is you have a bunch of machine instructions, and those machine instructions sometimes operate on different pieces of data, and they don't depend on each other, and so on, and sometimes you can literally randomly shuffle the machine instructions, and then you can check
Does it still give the same result?
If the answer is yes, some of those random shufflings, for all kinds of reasons of the detail of how the CPU chip is built and so on, may happen to be faster. So it's inside optimizers, you'll have these things where you're, for example, randomly shuffling instructions. Then the question is, does that actually work?
Or is it… does it usually work, but there's some horrible bug where it doesn't work in some corner case?

So, those are… those are cases where, in that case, you have a definition of what should happen, which is the form of the program as, for example, you wrote it in Wolfram language, in C, whatever else.
And then the question is, does the thing that comes out after all this optimization and this kind of thing, does it actually functionally do the same thing as the program you originally wrote?
Now, determining that, one way you can determine that is to say, let me just run a bunch of test cases. Let me see, does it work for this case, this case, this case?
You'll never cover everything that way.
for example, when Intel had this division bug, they'd tested lots of cases. They just hadn't tested every possible number.
Because, well, in the end, there are an infinite number of possible numbers, but even the numbers that their CPU chip could represent was an absolutely astronomically huge number of numbers, far beyond ones you could just test every one at a time, so to speak.
So, when you do testing, you end up having to sort of guess what might cause a bug, what's likely to be kind of a messy thing that doesn't quite work. So, for example, in our company, we have a software quality assurance team that's been, for the last 40 years, has been building millions and millions of tests and automated tests
For our software that attempts to find out, you know, does the software do what it's supposed to do?
So…
there's two ends to that. One is, how do the tests get created? And the other is, did it do what it was supposed to do? Sometimes there are conditions that should be true about it if it got the right answer. So, for example, if you are… oh, I don't know, if you're trying to,
in some fancy calculus thing, you're trying to compute an integral.
you get this integral, it better be the case that if you differentiate the integral again, you can get something which is equivalent to what you started with. So that's kind of a test where you can say, did I get it right? Or in another example, if you have an integer and you're factoring that integer, it's like you just take the
the factors that you get out from that integer, and you say, if I multiply them all together, do I get the original integer again?
If I do, then it can be the right answer. If I don't, then it can't be.
So, so one thing is, what kinds of tests can you make? One test you clearly want to be true is whatever input I give, the program shouldn't crash.
The program shouldn't start… you know, programs typically crash because they were supposed to be operating on some piece of computer memory, and they tried to access a piece of computer memory that wasn't part of what the operating system had allocated to that program, and then the program would crash.
And, the very common thing is that, you know, you have address space of the program, has all the different… every… every… everything in memory is given an address, a pointer, a number that says, here's
here's a position in memory, go retrieve what was there in memory. You can have, for example, a null pointer, which is zero. Go find what was at position zero in memory. Well, typically, the way operating systems are set up, there's nothing at position zero in memory, and that has to be wrong, and the operating system will typically say, if the program just asked for something at memory position zero.
That program is… has gone wild.

let's just kill that program. And so that's what happens when… and then what happens to the user, what you see is the program crashes.
And the operating system says, sorry, that thing crashed, you know, start another program. Of course, in the end, the operating system can crash.
In which case, the screen of your computer goes black, or blue, or whatever it is, and you have to restart your computer.
But that's… so one type of test is, did it crash? Did the… did the thing that happened cause the program to crash? Another is, did it get an answer which is consistent with whatever con… Consistency condition you can… you can put on?
Now the question is, well, how do you find things that most stress the program?
Well…
A typical thing is you look at all sorts of end conditions, you know, what about the case of zero? What about the case, you know, what happens if you put an infinite number in there? What happens if blah blah blah?
Another thing you do is, what happens if just it's a big blob of data that you're putting in there? Like the password that was supposed to be 12 characters, but you put 40,000 characters in instead? Those kinds of things. Another thing that's pretty common, it's a technique sometimes called fuzzing, where you basically are saying, this was the input you were supposed to give, let's just randomly change that input.
to something no human would ever think to type in. It's something has been randomly scrambled But that's a thing that will exercise the program in a way that was not anticipated.
So there are other tricks you can play. Another one, in terms of the way you can… you can look at the source code of the program, you can look at the actual code of the program, and you can ask, given the tests that I ran, let's say I have 100,000 tests.
I run all those tests, and you can ask the question, did every line of code in the program Get exercised by some tester or another.
Or are there pieces of code in the program that, like, none of the tests that I gave ever went to that piece of code? They never… they never exercise that piece of code. Sometimes, when you do that so-called coverage analysis on code, you find, actually, there's no input I can give that ever reaches this piece of code.
That piece of code was put in there 10 years ago or something, 20 years ago. It's dead code now. Nothing can ever reach it.
That's one kind of thing. Now, coverage analysis gets tricky when there are things like real-time interrupts, when you're running a program and it matters that just at that moment, you press some key… some… something on the keyboard that says, no, change what you're doing. You make an interrupt to the program.
It's very hard to test those kinds of things with coverage analysis, because not only is it given this input to the program, what's the output, but it's also, if this thing happened at some random time while the program is running, what would happen then?
I mean, you know, there's… well, one could talk a lot about software testing, and there's a lot of… okay, the general technique of software testing, a very common technique, is so-called regression testing. You have a version of the software that you think works, you store a bunch of tests based on it working and doing what you wanted it to do then, and you rerun those tests over and over and over again.

Every time, every day, for some of our software, we're running them every hour. Some of the bigger test runs, they run overnight. Some of the other ones are run on lower frequency, but the basic idea is you make a new version of your software product.
and these tests run, sometimes you can have so-called continuous integration, where you are running tests before you add a new piece of code to the code base of your program. Before you commit that code in to the… to the code of your program, you're running the test to see if it worked.
And, by the way, in modern times, LLMs are pretty good at writing tests for certain kinds of code. But,
So, regression testing works well up to a point.
The place where it gets into trouble is when the thing you're testing is, what gets put on the screen? What pixels get put on the screen? Well, actually, you know, something changed in the operating system, something… some detail changed, the particular arrangement of pixels on the screen changed.
But still, the thing looked the same to any reasonable human.
So, for any reasonable human, it passed the test, but if you're just doing pixel-by-pixel comparison, it's like, no, that failed the regression test. So, oh, I don't know, 40 years ago now, we built tooling for our
Our software testing that involves taking images of what came out, and doing all kinds of comparisons of the images, and asking, is the change large enough that you should care about or not?
In modern times, one of the interesting cases is testing LLMs, where you say, you know, you ask the LLM to write an essay, and it wrote one essay. You ask it to write the same essay again, it'll write a different essay.
So the question is, did the LLM, when you make a change to the whole way it's set up, did it succeed in still writing those essays, or did it fall on its face? And kind of the best way of testing seems to be just ask an LLM. You ask another LLM, look at what the thing did before, look at what it did now.
is what it did now reasonable based on what it did before. You can get a little bit more sophisticated than that, but that kind of recursive, have the LLM test the LLM, seems to be the right way to do that.
Well, anyway, I was… this was all in answer to the question, could the universe have a bug in it? So I was trying to explain the nature of bugs
in software systems, and which is where bugs traditionally are found, and this idea, you think you know what it's doing, but actually, it does the wrong thing. The rules that it… the program that it was given doesn't do exactly what you expected.
By the way, in, in,
in terms of the expectation of programs versus what they do, an interesting example of that is a thing that happened, oh gosh, when was it? Nearly 10 years ago now, on the Ethereum blockchain. There was a thing called the DAO, the Distributed Autonomous Organization.
The Dow very proudly said, this is a thing where we're running an investment operation, so to speak, a venture capital thing on this blockchain, and no humans are involved in its governance. It is purely run by code. There's code that defines, you know, if this happens, then people get paid this money, and if that happens, then these people get paid that money, etc, etc, etc.
It's all defined by code. It very proudly says, this is a system defined completely by code. So, it isn't… it isn't the case. There's no human in the loop saying, well, I… according to my judgment,

I think you should get the money or not get the money. It's just, it's all pure code. Okay, what could possibly go wrong?
you might say, well, the… the… what was given was a, you know, was a program that was a specification of what you wanted this DAO thing to do. Okay, well, somebody transferred, I don't remember what it was, $50 million or something to themselves using the system.
And people said, you know, you shouldn't have done that. Said, but look, they followed the code. This thing was defined by its code, and they found what other people said, you found a bug in the code.
And a reasonable argument is, but this thing said it was defined by its code. So what do you mean I found a bug? It was just, I was following the definition of what you said you put up there, which was defined by the code.
So it's meaningless to say there's a bug, because the thing that is what you wanted to have happen, you said, is defined by its code. Now, people said, well, of course we didn't intend it to be the case.
that you could, you know, transfer all this money to yourself. But that's then going beyond… that's a… that's an example of, like, what's the specification? Can you, you know, in a sense, what happened there was there was a bug in the specification. The specification happened to be the actual running code, but there was a bug, as far as people were concerned, there was a bug in the specification. But as far as the specification.
The person followed what the specifications said.
You might wonder what actually happened in that case. What happened was, you know, blockchains are very proud of the fact that whatever transaction happened is recorded for all time and can never change.
Of course, people said, but we want to change this transaction, because it wasn't what we wanted to have happen. So the Ethereum blockchain split in two, I think, Ethereum Classic and Ethereum, I think, were the names.
And on the Ethereum Classic, they're like, we're blockchain purists, the code was the code, and we're going to keep that transaction, even though this person got to walk off with the money that we didn't think they should have walked off with.
And the other… the other branch, the other fork, said, we're going to roll back that transaction. Forget this blockchain purity, we're just going to say, it's as if that transaction didn't happen, and go on from there. And I think the… the branch that, that said that rolled back the transaction quickly became worth a lot more than the other branch. People preferred it.
Which is, you know, one of these statements of… of, sort of, don't let the computers be in charge of everything type thing, because, you know, that's not… that doesn't end up with things that people want. But okay, so that's… that's kind of the idea of bugs, specifications. There's a question of why are bugs
Why is it so hard to write code that doesn't have bugs? I think the answer is that the most interesting code is code that does things that were not obvious to you. In other words, if your code
is doing something where you can say, well, I know what the answer to the code is going to be. Well, why don't you just put down the answer? Why even bother to have the code?
If the… if the result is the code is going to do all this effort, and it's going to, you know, run all these little, you know, do all these computations, and then the answer is you're going to add one to a number or something. It's like, well, just add one to the number by the easiest way you can. Don't go through all that effort to achieve that objective.

So the most interesting code does things which were not readily jumpable to the end and just say, oh, we got the answer.
And as soon as you have that situation where you actually run the code to see what happens, you have this phenomenon of computational irreducibility that I talk about a lot, and you have the feature that
Given that you have to actually run the code to see what happens, there can be surprises. It can be that the result of running the code is something that you didn't expect.
Because if you… if you always knew what to expect, you don't even need to bother to run the code. So, the reason that bugs are so common is that a lot of code that one cares about has computational irreducibility in it.
it has things where it necessarily can do things where you have to run the code to see what happens. It can necessarily produce surprises. What you end up having to do is to say, we want to sort of sculpt this code so that there aren't too many surprises, or aren't surprises we really didn't want to have happen, like the other thing crashes, or some such other thing.
So, it's, it's a… the… the sort of the source of bugs Is computational irreducibility.
You know, when you try and make programs that do things, we sort of understand what they're going to do, but we don't want to know exactly what they're going to do. We're trying to sort of bottle up computational irreducibility
And every so often, it's sort of inevitable, once you have that sort of spark of computational irreducibility inside your program, every so often that spark will kind of jump out, and those are the bugs.
So that's… that's kind of the picture of bugs in programs.
So now, what would it mean for there to be a bug in the universe?
Well…
We don't know the specifications for the universe. I think we're beginning to know them, but we don't know the specifications for the universe.
So, to say, you know, we can't really discuss whether there's a bug in the universe unless we know what the universe was supposed to do.
So you can say, given that we have deduced from observing the universe, we have said… we have deduced these natural laws from the universe. We say, oh, we've always noticed that when you… when you let go of things, they drop to the… to the ground on the surface of the Earth, they're pulled down by gravity. That's a natural law that that happens.
And so, if, you know, if we let go of the thing, and the thing started going up instead of going down, that would be, you know, that's… you could say that's a… that doesn't follow what we thought were the laws. It's, it's kind of like, you know, you might have a helium balloon in your hand, and it starts going up instead of down.
And the question is, what,
You know, we have… we have a… we have deduced an expectation for what should happen in the world, and something might violate that expectation.
Lots of famous discoveries in science have been… have happened because an expectation developed about how things work in the world, and suddenly something was found that didn't work that way. And you could call that a bug, but it's a little bit different from the… the bug
That happens in a program, because in a program, we created the program with certain intention in mind, and then… and then it…
And then it ran, it did things we didn't expect. In the case of the natural world and the universe.

It's like, it's doing what it's doing. It didn't get permission from us to do those things. It just is doing what it's doing, and we are trying to deduce what it's doing. We're trying to kind of reverse engineer what it's doing by doing science and experiments and all those kinds of things.
So, it could be that we just didn't have a complete view of what it was doing from the outside, and therefore it did something unexpected, and we could call that a bug, but it isn't really the same kind of bug. It's not that we started the universe with these rules, with something in mind for the universe, and then the universe went off and did something that we hadn't expected.
By the way, in the world of programs, the,
you know, programs that you wrote yourself, you had an intention in mind. Programs that you have an LLM write for you have a whole other raft of issues associated with, you know, both did the specification get communicated? Did the LLM just sort of spaz out for a moment and write some piece of code that it shouldn't have written? There's a whole chain of other kinds of things that can go wrong.
And this idea of, I have, you know, I have a specification in mind, I have the program, it's a little different in that case. But unless you are sort of the creator of the universe, you… there's… you don't… there's no… nothing has a program in mind for the universe.
Now, in, you know, in past traditions, before the modern period of science, when people said, well, you know, if something goes… happens in the universe that isn't what we expected, then that's a… that's a glitch, a bug in our scientific law, we have to change the scientific law. Before that time, people might have said, you know, it's a miracle.
that this happens. There's a way the universe usually works, and there's a deviation from that, and that's a miracle, kind of a, you know, whatever, depending on how your belief system is set up, you know, that's a direct-from-God intervention, so to speak, in the goings-on of the universe.
And I suppose that that's a, you know, that's kind of a… a… the universe is running according to a program, and… and there's sort of… from outside the universe, somebody reached in and made that little tweak for this particular moment, and made this thing happen differently in the program.
So, you know, I think in the end, this question about, sort of, bugs in the universe, the,
The issue is really how complete are our natural laws, and can we be in a situation where we feel like we've got the natural laws of the universe, and sort of nothing else can happen? Now, you know, my own efforts to understand the fundamental theory of physics.
I think we've gotten a long way in understanding kind of what the machine code of the universe is like, and in realizing that for observers like us, who observe the universe in the ways we do, that it's sort of inevitable that certain laws of physics have to work the way they do.
Now, you could say, well, can there be aspects of the universe that sort of deviate? We think we know how it works. We think we can derive how it works. Could there be things that violate that expectation? The answer is surely yes, because what's going on, a lot of what we observe in the universe
is, is based on sort of how we observe it. Let me give you an example. I seem to go back to an example like this many times.
We got, you know, in the room one's sitting in or whatever, there are all the gas molecules bouncing around. Most of the time, those gas molecules are fairly uniform in density. There's the same density of air here in front of my face as there is on the other side of the room. Pretty much. But it could be the case that all the gas molecules have a conspiracy, and they all bounce around in this and that way, and they all bounce to another corner of the room, leaving me in a vacuum, and I suffocate, or whatever.

Bad, bad scenario. But, you know, it could be the case that there's this weird glitch that happens that causes all the gas molecules to run to one side of the room.
Could happen. It's consistent with the laws of physics.
It just has an infinitesimally low probability. And one might say that something like that is kind of a bug in the universe if it ever happens, even though, in a sense, it's completely consistent with the underlying laws of the universe, it's just not something that normally happens. It is astronomically, incredibly astronomically unlikely that that would happen sort of by pure chance.
So, anyway, a few, a few thoughts of, about, bugs and the universe.
Let's see… Peter asks, why does turning something off and back on again fix so many problems?
You know, that's a question biology might have asked itself back in the day.
You know, why don't we just have one organism that keeps going forever and ever, and a billion years later, it's still the same organism.
And it's kept on, you know, things went wrong. Somebody… somebody poked it, and it had a little hole in it, and then it repaired the hole, and so on. And it just kept on, sort of, sort of repairing the organism for a billion years.
Biology had this bright idea at some point. Let's just make a new organism. Let's have a, you know, let's have the organism have children, and have a new organism.
that's kind of a mechanism, just restart. We've got the same program, more or less the same genetic material, but we just kind of restart the organism and say, you know, let's run it again.
Now, what happens in computer systems,
You have a computer, it's got an operating system, it's a big, complicated program, millions of lines of code, things are happening, you're plugging things into your computer, that's, you know, putting data in this place or that place, things are filling up inside your computer, things are getting full of data that are sort of buffers, that are storing things and so on. And pretty soon, well, after a while, you know, there'll be too much data in some region of the computer. It's some piece of computer memory.
or there will be something which had been supposed to be set up with a very precise collection of instructions, but some of those instructions got modified because something came in and interrupted this at the critical time, or whatever else. A sequence of things start going wrong.
And computers get diseases, much like humans do, that things start going wrong. The memory subsystem starts, or you start getting… memory is getting full up, and you're starting to get more places where the thing has to rearrange things in memory before it can retrieve stuff, or add stuff, or whatever else. The computer starts getting sort of old and tired.
And, you know, kind of, things get very messy inside it. And eventually, something happens that's sort of catastrophic, and the operating system crashes.
It says, I give up, you know, this is… I can't deal with this, I can't fix this. And it's the same thing, more or less, with biological organisms, and I think that, sort of the typical things are that sort of, during the life of the organism, things build up. You know, you get,
I don't know, you have sort of beautiful pieces of cartilage that are… that are perfectly, you know, joints are sliding around on, and then gradually, sort of bits fall off the cartilage, and the joint doesn't slide as nicely, and so on. And it's the same kind of thing with a computer, that, you know, for a while, that area of memory was completely free and empty, and you could put things into memory anytime you want.
But then programs have filled up more and more of that memory, and you had to… so here's a typical example in a computer.
So, has to do with the fragmentation of memory.

So… Your computer has a bunch of memory, gigabytes of memory, let's say.
program comes in, says, hey, I want 200 megabytes of memory.
But it needs those… that piece of memory contiguously. It needs to have… the memory needs to go from the beginning to the end, just in one… in one run. You can't… you can't break it up. So you say, well, okay, you've got 5GB of memory on my computer.
And then, so where can I find 200 megabytes of memory where I've got a big run of memory that's empty, or that I can reuse, that nothing's using anymore, where I can allocate the memory for this new program?
Okay, so you keep doing that a while, and you've allocated… this program got 200 megabytes, that one got 400 megabytes, that one got 50MB, and so on. You keep fitting these things in.
And as you do that, you fit them into the spaces that are left over, and pretty soon you realize, gosh, well, I've actually got quite a bit of memory that is available, but it's in these little, little little…
blocks that are quite small, and when something comes along and says, I need 500 megabytes of memory, it's like, sorry, there's no block. Even though there's 3 gigabytes of memory available, it's all in small blocks that aren't big enough to fit you in.
So, there are all sorts of strategies for dealing with that, and moving memory around, and doing remapping of memory, and so on, but that gives some idea of how, as the computer has run for a while, it's, you know, allocated this piece of memory to that, this piece of memory to that, it's freed this piece of memory, it's given that back, but when it gives it back, it doesn't necessarily, you know, it may not be contiguous blocks.
And that's the kind of thing that happens as you run a computer for a while, and that's the kind of reason that it sort of gets sort of old and crotchety, and eventually when you reboot it, it's like you're clearing everything out of memory, and so it's a fresh start, and you get to start allocating blocks of memory again.
Let's see…
Let's see…
RBS is commenting.
Nothing beats the stress of cycle counting on a Commodore 64 just to get a stable raster interrupt. Modern debugging feels like luxury compared to that.
Yeah, one thing to say about computers.
when… often, when you write a program, the program just does what it does. Start, you give it some input, goes crunch, crunch, crunch, gives you some output.
But some other kinds of programming require sort of real-time behavior. Like, if you're trying to put something on the screen, you know, which people are going to see in some way, it sort of has to… it can't,
Well, let me give you an example from earlier days of programming. I don't know how it works anymore, but on an early Mac, for example.
There was… the screen would get written
And the screen, in those days, it was a cathode ray tube, and cathode ray tubes had an electron beam that scans the screen, scan line after scanline, you know, 500 lines on the screen, and it's just literally, it's turning on and off, black and white, whatever, scans the screen, it scans the screen, maybe, I don't know.
Let's say 15 times or 30 times a second.
Okay, so after it scanned the screen once, there was this weird thing called the vertical retrace interrupt.

And basically what would happen is the computer would say, I'm painting stuff on the screen. Oh, wait a minute, there's… let me give you… let me give another program the chance to do something during the time that the… the sort of the screen is about to start repainting itself again. So, you as a programmer could get this, I forget what the time was,
let's say a hundredth of a second or something, I don't remember the time.
You had that time to do something else with your code before… before your program… before… and, you know, the operating system would run your program for a hundredth of a second, let's say, before it went back to running the program that painted data on the screen. And so that was a case where it's like, are you gonna make it in that hundredth of a second, or not?
And that was, actually in the early versions of Mathematica, we had a weight cursor.
That was a thing where, you know, nowadays you'd have little spinners and things like that. We had a very cool, very early version of that back in 1988, which was a square where a little ball sort of bounced around in that square. That update was done during the vertical retrace interrupt that we could kind of keep. It was as if we were sort of stealing time, but not really slowing the program down.
to do that.
I will say, in much more modern times, when Wolfram Alpha was first powering Siri on the iPhone, there was a different kind of timeout, which was that when you asked a question of Siri, and it went to Wolfram Alpha to get the answer.
The phone had its antenna fully powered up, so it had sent a data request to the server, and the phone's antenna was fully powered up.
and ready to receive a… so I should explain. To power up the antenna took, I don't know, let's say 2 seconds or something, and then it would be ready to start sending data.
And the question was, if you wanted to get the data back, the antenna would stay powered up for some amount of time, then it would power down to save battery life, so maybe it powered down in 5 seconds or something. And then the question was, would you get your answer back
In less than that time, or would the antenna have powered down? In which case it would take another 2 seconds to get the answer back, because you had to power the antenna up again before you could pull in the answer.
So that was a much slower version of the same type of thing, but that kind of thing happens all the time, and in lots of kinds of real-time programming, that's a very difficult thing to debug often.
Let's see… Lois asks, why doesn't Wi-Fi for millions of devices interfere with everything all the time?
well, there are two different issues there. One is, how can these devices be, sort of, sending signals to each other all the time separately, and not interfere with each other? That's question one. And the other question is, back… you go back 100 years.
There was no… there were no radio waves going around the world, and there was no, but nowadays, we're all exposed to radio frequency, to… to, electronic radiation, radio frequency, Radio waves all the time, you know.
does that have an effect? So those are two very different questions.
Maybe I'll answer the second question first.
So, it's been a long-running debate.
whether, for example, a common question is, if you hold a cell phone up to your head, does it, does it cook you? So, just to give some background for that.

a…
cell phones and things like that use microwaves. They use, they're sending signals, radio signals, at a certain frequencies, so typical frequencies.
used to be 2.4 gigahertz, 2.4 billion, sort of cycles per second. Quite often nowadays it's 5 GHz. There are particular, frequencies that are used for cell phones, and there's sort of been a big, adventure over the years that
Well, okay, so I should explain something. In… when you have radio, you are sending electromagnetic waves
a bit like light, but they're lower frequency than light. You're sending these things, and that's the… and what's happening is, you know, electrons are wiggling up and down in an antenna, and the electrons wiggling up and down cause this electromagnetic wave to be produced that then goes off
And then if it encounters another antenna, when the electromagnetic wave hits the other antenna, it causes the electrons in that antenna to wiggle up and down, and that produces an electrical signal that you can then detect.
So, the point is that the way these things work, antennas are tuned to a certain frequency. There's a certain frequency of wiggling that any given antenna will respond to, and other frequencies of wiggling it won't respond to. Kind of like, oh, all kinds of things, like if you,
I don't know, if you have a swing, if you're on a child swing type thing.
and you are, you know, pumping with your legs, and you do that at the frequency that the swing is going back and forth at, you stick your legs out and back in again, at the, you know, every… on every time the swing goes, sort of, to the… to the front and back, then you'll make the swing go further. If you sort of do that randomly, you'll just slow the swing down. So if you are in resonance with the string, with the swing.
If you are putting, sort of, if you are pushing things at the same frequency as the thing naturally is oscillating at.
then you will sort of build up… you'll build something up bigger. And so what happens is that when you have an electromagnetic wave, a radio wave, you… it has a certain frequency, it's wiggling at a certain rate.
And antennas tend to be tuned to deal with waves of a certain frequency. And so, if the wave is of another frequency, the antenna just is like, I don't care, I don't even notice it. If it's at the frequency the antenna is tuned for, then it will produce an electric… it will get amplified, and it will produce an electrical signal.
More or less. Okay, I'm alighting a bunch of details here. So, used to be…
Okay, there's a lot of trickiness about how that amplification works, how you amplify only one frequency. You might have an oscillator, like even a quartz crystal or something, but it's just sort of vibrating at the right frequency or whatever.
In modern times, it's become a little bit more complicated, because one's actually using software radio often, where the raw signal comes in, and you're using, sort of, it's just numbers that align to amplify the thing at a particular frequency. But the basic point is, there are a bunch of different frequencies
of electromagnetic radiation that get used. And by the way, in answer to sort of the first part of this question, why don't all the devices interfere with each other, in a first approximation, it's because they're operating on slightly different frequencies. You'll have a whole band of frequencies, and in a first approximation, I'll say a little bit more about that, in a first approximation, you would say, okay, you've got

all these different devices, and there's a certain band of frequencies, and you've got all these devices, and each one will get allocated one frequency within that band. Back in the early days of cell phones, it used to be the case that every cell in a cell phone region, I think it had 666 channels, and that meant when two cell phones wanted to… well, the cell phones go through cell phone towers, but imagine that
That they wanted to talk to each other.
they kind of had to agree on a channel. We're going to use channel 323. That meant a particular frequency. They're both tuned to that frequency. One can send on that frequency, the other can receive on that frequency. The things that are, you know, frequency channel 270 was a different frequency, and those cell phones had just shrugged their shoulders. They don't even notice that there's something happening at that frequency. It's only at the frequency at which there is, sort of. Resonance with the… with the amplifier, with the oscillator, with the antenna, that it matters. Okay, but in any case, the,
there are bands of frequencies that are allocated to cell phones. Over the years, different frequency bands were allocated to different kinds of things, like some of them were television channels, some of them were AM radio, FM radio, some of them were radio navigation aids for airplanes. You know, if you go through… I haven't done this for a number of years, but if you step through frequencies in the electromagnetic spectrum, it's kind of interesting what you hear. You know, there are… they've pretty much gone away now, but there used to be some frequencies in which you could actually hear speaking, that the… that the frequencies were… That the data for radio stations, for example, was encoded in such a way that, sort of, the raw bits would be, you'd put them together in a sound, you would hear somebody talking. And then you would hear lots of Morse code kinds of things, you'd hear lots of, sort of, beeps and so on from radio navigation aids, you would hear, things, and lots of stuff that just sounded like random noise.
That, to you, sounds like random noise, but if you have the right kind of antenna and decoder, it's like, oh, this is a signal that's relevant to me. But so, as you step through things, you would… you would, you know, every… as you step through different frequencies, you would be, you would kind of, hear different things on the electromagnetic spectrum. I mean, back in the day, oh my gosh, I don't think anybody even
knows this anymore, but you would have a radio where you would tune the radio by turning a dial, and as you turn the dial, it changes, typically, the resistance or inductance of some electric component inside it that changes the frequency at which the antenna or the amplifier resonates, and therefore you would be sensitive to a different frequency. So when I was a kid.
you know, I had a radio where you would turn the dial, and at one frequency, you would be, hearing, you know, some radio transmission from France or something. I was in England. Another, you know, frequency, you'd be hearing some… some local radio station, et cetera, et cetera, et cetera. As you… as you each, at different frequencies, you would be hearing different things, and… and some things were… some frequencies, they…
were frequencies where there were, radio transmissions that were on the opposite side of the Earth.
But the radio transmission was bouncing up and down between the Earth's ionosphere and the surface, and you could kind of get the result of that on the other side of the Earth. But in case, so… so…
And, you know, the way that things have worked in the,
There is a, in any given country.

certain frequency bands are allocated to certain kinds of things. Like, some frequency band will be allocated for this kind of cell phone.
Some frequency band will be allocated for GPS signals.
Some frequency band will be allocated for, well, maybe for radio astronomy, for people, you know, where nobody's supposed to be producing that frequency on the Earth, because one's seeing whether that's produced by stars or whatever. Or some other frequency might be, some, some other kind of thing.
And so there are, in the world,
there are 3 big zones in the world, ITU zones, I think, International Telecommunications Union zones, and the, and in each of those zones, and it's also specific to countries, each part of the frequency spectrum is allocated to something.
And every so often, people say, oh, we don't really care about black and white television signals anymore, and so that band of frequencies is, you know, some government will auction it off.
to somebody who wants to run, you know, I don't know, satellite communications there, or something like this. But in any case, the frequency spectrum has been allocated, and, that, but there are also
So, that's sort of radio transmission, and there's signals in pretty much all those frequency bands. Nobody would buy a frequency band unless they wanted to actually send data in that frequency band. And then, okay, then the question is.
Well, okay, the electromagnetic radiation that's been sent is microwaves, and people know that there are microwave ovens.
So what's a microwave oven relative to the microwaves that your cell phone produces? Well, the answer is it's not that far away, because cell phones… microwave ovens, I think, operate around 2 GHz, and they are producing intense
electromagnetic radiation inside the microwave oven. It has a… it's a… it's a, it's a resonant cavity, basically, that just… is just keeping on pumping, microwaves at a certain frequency.
What does that frequency do? It happens to be a frequency that is tuned to the,
to the water molecules. So, at a particular frequency, that corresponds where the wavelength corresponds roughly to the size of a water molecule. If you… if you, you know.
hit water molecules with that electromagnetic frequency, they'll start spinning around. It's the, rotational band of water where you're… so you, you know, when you put… when your microwave oven is… is pumping microwaves out.
inside this cavity that is the microwave oven, then you, you are taking any water molecules that are free to move, and you're making them spin around, and that sort of generates… that… that… Energy that is being transmitted from the microwaves to the water molecules is what heats stuff up.
And, you know, you can tell that a microwave is just… it's an enclosed cavity for the microwaves. If you… if your microwave oven has a… has a sort of window in front, you'll see that there's this kind of mesh of metal behind the transparent glass or whatever window. And that mesh of metal is… is… the little pieces of the mesh
Are small enough
that the frequency of microwaves doesn't escape, that the wavelength is longer than the size of that mesh, so the microwaves don't escape that mesh. They're being enclosed in this cavity where they're being amplified to cook your food or whatever.

Well, okay, so now the question is, you've got a cell phone, it's making microwaves. Maybe it's even making microwaves that are not far away from that band of water. What happens if you hold the cell phone up to your head? Does it cook the water in your brain?
Reasonable question.
the…
One, people try pretty hard to make sure that doesn't happen, obviously, and one thing that tends to happen is that cell phones will change the power that they use for their antennas. If you hold it up to your head, it will typically go to lower power than if it's somewhere away from there. And there are…
A whole elaborate set of, kind of, engineering
hacks, I suppose you could call them, about sort of what the antenna does when you're holding the cell phone in different ways. I mean, back in the day, when it was like, oh, can I get a cell phone signal? It still happens sometimes, but this doesn't usually work anymore. It's like, oh, let me hold it in this way. If I hold it, it does something different. If I put it this way, it'll do something different, because it's… it had a bunch of things that were changing the antenna power, depending on what you were doing with it.
it.
But the question then arises, well, okay, there's all this electromagnetic radiation, it's mostly rather low power, it's incredibly much lower power than the microwaves inside your microwave oven, but if you are exposed to it all the time, does it
Does it eventually cook you, so to speak, or do something, or produce cancer, or some such other thing?
I think the answer is we just don't really know. And it's, you know, for somebody like me, who's kind of, like.
sort of conservative about these things, you know, I spend all day yakking on the phone or whatever, and I… if I'm gonna wear a headset, I'll typically have one that just has a wire. I won't use Bluetooth.
which is using radio, not very low-power radio, but it's still, like, why do that? If you don't need it, don't do it.
Plus, you know, you get some… some… usually you get some higher quality, audio and things by… by actually having a, you know, a wire that's sending the signals, rather than, oh, it might be interrupted by this or that thing. But it is still sort of an open question whether the sort of… the fact that we're all bathed in so much electromagnetic energy
has had an effect on us. Physiologically. There's some evidence that it has… does have an effect. There's plenty of evidence that doesn't have an effect. There seem to be some people who are so-called electrosensitive, where it has a much bigger effect than other people. It's completely unclear why.
I mean, I have to say, for myself, to say something very silly and crazy about myself, but, if I touch a,
an intense magnet, like a rare earth magnet. My fingers tingle.
I have no idea why. And at various times in the past, my children tried to do sort of tests on me to, you know, is this effect real, or is this something not true? And I would say the experiments were a little bit…
Not completely definitive, either way. But, you know, what could cause that? Well, you know, you have a magnet, if you move the magnet around, it will induce an electric current.

And, you know, electric currents are what nerves use to transmit, you know, to transmit sensations, and so it's not completely out of the realm of possibility that a high-power magnet wheeled around the nerves in the tips of your fingers could have an effect. I don't know. These other kinds of things, it's really unclear.
And it's unclear why
as I say, certain… some people seem to have this kind of sensitivity to electromagnetic radiation that other people just don't seem to have. You know, maybe we all have it to some extent, some people have it more than others, maybe some of us don't have it at all, maybe the whole effect isn't, you know, is more imagined than real. I don't know. My guess is there is some effect there for some people.
And, we don't really know what its cause is.
I will tell you another thing. Back in the day, long ago.
oh gosh, when I was doing particle physics in the late 1970s, in particle physics, you often are trying to do things with high-energy particles, and you have these incredibly strong magnets. You have these huge magnets. You could walk inside the magnet.
And it would have a very strong magnetic field, which was intended for bending high-energy particles.
And one of the questions was, if you walk into this magnet, and the magnet is on, do you even notice?
And, people said back in those days that, anecdotally.
people got funny tastes in their mouths when they were, for example, working inside the magnet when the magnet was on. And that's not, again, a completely crazy thing, because there's electrochemical processes that happen in, you know, in taste and so on, that could be affected by a magnetic field.
I mean, this is a,
you can ask the question, you know, if you get your… an MRI, and you have, you know, a, I don't know, a typical MRI, I think has a 1.5 Tesla magnetic field. There are fancier ones that have three Tesla magnetic fields, maybe even some six Tesla ones now. Those are… the Earth's magnetic field, by the way, is one 10 millionth of a Tesla, so 10 to the minus 5, 1 100th thousandth.
of a Tesla, of magnetic field intensity. Whereas… so, a… the magnetic field of your average MRI is, 100,000 times more intense than the Earth's magnetic field.
And so, you know, a question is, does that have any effect? Do you… do you have, you know, if you have an MRI for an hour or something, do you, you know, is there some sensation you get? I don't think so. I don't know. You know, the times I've…
I've done that, getting my, you know, whole body MRI scan, why not look inside oneself type thing. You know, I have never noticed that.
And… but it doesn't mean to say there are no effects. Just to comment on magnetic fields and effects, I mean, so there's… one question is, sort of.
continuous magnetic fields, like the Earth's magnetic field, or like… well, the magnetic fields in MRIs aren't quite continuous, but they're… they're closer to that. Or electromagnetic radiation, which is at
Billion times a second, you know, frequencies and so on.
When it comes to magnetic fields, there's sort of a long-running question of how sensitive are we to magnetic fields at all. There's pretty good evidence that pigeons use the Earth's magnetic field as part of their navigation, that they have… they have metal atoms in their brains, in little cages,

molecular cages, and the metal atoms respond to the Earth's magnetic field, and the pigeon can sense that.
there's sort of a whole effort in so-called quantum biology where people are concerned about things like this, and whether there are effects on the Earth's magnetic field, and so on. And I think the experiments on that are still pretty undetermined.
I think, let's see, so the, the, I think one branch of this question was.
is all the electromagnetic radiation in the world causing us trouble? And the answer, I think, is we don't really know. And the other question is, all these devices, how do they arrange not to get confused about which device is being talked to? One of the… the more primitive technique is just use different frequencies. Two devices will kind of meet at a hailing frequency, and sort of negotiate, oh, I'm going to use that frequency rather than that frequency. We're both going to use channel number such and such. They both open up that channel, and they're communicating through that channel.
There are fancier techniques.
The, as you go to…
In cell phones, for example, the 2G, to 3G to 4G, to 5G, and so on.
These, these generations of cell phone technology have slightly different techniques. So one thing that was used in 3G was the thing called CDMA, Code Division Multiplexing. So, okay, there are different techniques. So you can segregate by frequency. That's one thing you can do. You can also do time division multiplexing. That means every, every piece of speech
That lasts, let's say, a hundredth of a second.
you say, I'm going to take that piece of speech that lasts a hundredth of a second, I'm going to take
the data from it, and I'm going to compress that so I can send that in a 10 thousandth of a second. And you can do that because as we speak.
We're not, the sort of…
the motion of our vocal tract or whatever, it's slowly moving up and down. By slowly, I mean it takes, you know, a thousandth of a second to say that vowel or something. And in that thousandth of a second, our vocal tract is doing something quite predictable, for at least a thousandth of a second before we get to the next thing, and so on. And so you can use that predictability to be able to compress the data associated with speech
That's what, well, for example, MP3, MP4, and so on, these, standards, they are compressing by using the fact that once you can sort of say, well, the vocal tract is roughly doing this, you immediately know, well, I'm going to fill it in with these particular frequency wiggles.
So, one method for, sort of, sharing the frequencies is time division multiplexing, where you say, okay, you're gonna break every hundredth of a second into, 100 pieces.
And each, and each…
conversation is going to occupy piece number 7… a particular conversation is going to occupy piece number 17 out of those hundred slots that we gave. And at that precise time… time moment, every hundredth of a second, hundredth of a second plus
17 ten thousandths of a second, let's say, you get 1 ten thousandth of a second to say your piece for conversation number 17, and then it goes to conversation 18, and so on. So that's another way to kind of share bandwidth, to share the… is to slice it by time. That's a pretty common technique used for lots of kinds of things.
Another technique that's a little bit more mathematically sophisticated is code division multiplexing, where what you're doing is you take every signal.

and you kind of, you take every bit in the signal. Let's say you've turned the signal into a sequence of ones and zeros, and then what you do is you take kind of a template of ones and zeros that
you kind of… you combine with the data that you had, and so you typically do a… mathematically, you do an XOR, exclusive OR. You take… every time you have two 1s, you turn it into 0. Every time you have a 1 and a 0, you turn it into one. 0, 1, you turn it into 0, 0, etc.
So, anyway, the way that operation works
If you have your original data, and you have this kind of key, this template that you are imposing on it.
you can… you impose that, you get something… you get a result. If you want to go back to what you had originally, you just have to use the same key again on the result of that, and you get back to the original signal you had.
So, the way it works is that in co-division multiplexing, you're using so-called pseudo-noise sequences, which are things which look for all the world like they're just random sequences of bits, but they're not. They're made with things called linear feedback shift registers, and they have mathematical properties that cause them to… let's say you can make… just give me
you know, 256 sequences that are all pseudo-noise sequences. You can do that in such a way that each
Every one of those sequences
It's kind of… it looks random relative to the other sequences, and it looks… if you just look at them, they look random.
But then if you encode a signal using one of those, you kind of apply that as the key, you get something that kind of seems random for everybody else, but if you were to apply that same key again, you would retrieve your original signal, and you get the thing back again. So what happens is, in that case.
it's typical in 3G communications, is you're… you're taking all those different conversations, you're combining them with the different pseudo-noise sequences, you're getting out something, and the ones… each one
Kind of looks like noise to the other conversations, and then you decode them at the end.
When you get up to things like 5G, there are other techniques that are used that have to do with trying to just not have the electromagnetic waves go in all directions. They're put in the direction that you actually want them to go to get to the target that you want to get to, rather than just having them spread out in all directions.
So that's, so there are these different techniques. I mean, you know, one of the things that's sort of fun to think about is if
you know, if you're imagining the Earth viewed from space, you know, the alien intelligences are watching the Earth, and they're listening to its radio transmissions. Well, there was a period of time when all radio was just sort of… you want to transmit, you have a radio antenna, it's transmitting the radio everywhere.
And then things… and so at that time, some period of 50 years or something, it's like there's all kinds of radio just spewing out into space.
But increasingly, radio is more like, no, no, we don't want to do that. That wastes a bunch of energy to have the radio sort of spew out into space. We want the radio to be concentrated in the exact direction from the original sender to the receiver that wants to receive that particular thing.
And so the Earth is sort of getting quieter from the point of view of the outside world, because more and more radio energy is being directed in that way. And that's kind of another way that

you avoid sort of one thing interfering with another, is it's just the radio waves just aren't going in the same direction.
Okay, there's questions here.
Wow, okay, there's one from Jakes. What does AM and FM mean?
Okay, so that has to do with how you encode signals for radio waves. AM is amplitude modulation, FM is frequency modulation.
So here's roughly how it works.
So… A particular antenna, Or the oscillator behind that antenna is tuned to a particular frequency.
So… and the… when you have a… this is sort of old-fashioned radio story. It works differently now with everything being digital, but this is what happened in the past.
So, there is… a given radio station, let's say, is, you know, we are 90.7, you know.
Megahertz or something, they'll say when they announce their radio station.
And that means that they are producing a signal where the main part of the signal is a 90.7 MHz wiggle.
and the antenna is tuned, is resonating at that frequency to receive that signal. Okay, well, it's all very good to send the carrier wave of that frequency, but that contains no information.
It's just, that's the carrier wave. Great. And if you're sending Morse code or something, maybe that's enough. You just say, is there a carrier wave or isn't there a carrier wave? But if you want to send something like voice, you have to sort of add something into that carrier wave to, to transmit the voice.
So amplitude modulation just means you take the carrier wave, it's wiggling up and down, you know, 90 million times a second, or whatever else, a little bit less, actually, for typical AM.
Let's see, and I'm trying to remember.
Radio stations of old.
they used to quote them in wavelength as well. Well, okay, but anyway, let's say it's, I don't know,
10 million times a second or something. It's wiggling up and down. Okay, amplitude modulation is…
Okay, when we speak, for example, there's pressure waves that we're generating that correspond to the sound waves, and those pressure waves kind of are… there's high pressure, there's low pressure, there's high pressure. If we say, for example, the,
middle C on a piano is typically tuned to 256 Hz. That means, it goes from compression to… it's higher pressure, lower pressure, higher pressure, lower pressure, 256 times per second.
So, when we speak, we have… we produce sound waves that are wiggling high pressure, low pressure, high pressure, at frequencies of order hundreds of times per second.
Okay, so how does AM work? You take those wiggles that correspond to our speech sounds, and you simply multiply the carrier by those
by the intensity of those wiggles. So, if you're making a very loud sound or whatever, that… then the carrier, then the… what's transmitted in amplitude modifi… modulation is a… a higher… a higher
a higher amplitude, a higher energy of carrier wave, and when you're being very quiet, it's a lower energy of carrier wave. So amplitude… modulation is changing the intensity of the radio energy that's being put into that particular frequency.
Okay, so that was the earliest technique from the early part of the 20th century for radio, was amplitude modulation.

sometime in the 1940s, I guess it was invented, I think it came to use in the 50s, maybe 60s, was frequency modulation. So in frequency modulation… okay, one problem… okay.
let me explain what frequency modulation is, and then why we would care about it. In frequency modulation, you have that carrier wave that's operating at, let's say, 10 million cycles per second, 10 million vibrations per second, or whatever. In frequency modulation, you're slightly modifying that frequency. Instead of it being 10 million, it's 10 million plus 100.
cycles per second, or something. And you're taking your signal, and you're using that to modulate the frequency of the wave.
Why is that a good idea? The problem with amplitude modulation is many things can change the radio energy that you get. You know, if you're… if you're receiving this radio signal, and a tree, you know, the wind blows the leaves of a tree in front of you, it can reduce the amount of radio energy that you get.
So that can… that can cause you to… to get the wrong signal, so to speak. In amplitude modulation, it's… it's much less likely that something like that will happen, that will affect the… sorry, in frequency modulation, much less likely that will happen, and so if you change the frequency, very few things will… will… very few sort of natural things in the world will do that, and so it's easier to get the data through.
So…
That's, so that's kind of the idea of frequency modulation, and, there, there are other techniques, But later, and I think it's,
Probably time for me to go to something I need to do for my day job here. But, well, we got through at least two questions today. Thanks for asking those interesting questions, and
Bugs and electromagnetic waves. Two very different kinds of areas, and
It was fun to chat about them, thanks for joining me, and I look forward to, chatting with you all again soon.
Bye for now.