Hello, everyone! Welcome to another episode of Q&A about science and technology for kids and others. I think it's been a little while since I've done one of this particular type of livestream, so let's see what questions are saved up here.
So, question here from Kathy.
What's the difference between information and data?
Is there a difference?
It's an interesting question. People are often confused about this particular thing.
So… One of the things that is sort of a remarkable sort of…
The thing that's emerged as a result of computers and the ideas of computation is that lots of different kinds of things that we care about
can be represented in a fairly uniform way. So, images, sounds, text, Each of those, as humans. we kind of consume them in different ways, you know, text, we read it with a, you know, scanning word by word. You know, images, maybe there's some picture, and we kind of might look around it in different places. Sound, we hear it with our ears, and so on. These things kind of play differently to us as humans.
But one of the big things that comes up with computers and computation and so on is that all those different, sort of, forms of human experience can be, in the end, turned into bits for a computer.
So the ways they're turned into bits are different in those different cases. So, for example, for text.
You know, every letter, you know, A through Z, whatever, every, every, every kind of character that you can put in a piece of text.
every so-called glyph that might be a letter, a number, it might be a Chinese character, it might be some mathematical symbol, all these kinds of things, there's a standard called Unicode that tells you how to convert each of these into a number.
So, I think A is a 65, if I remember correctly. That probably is a… uppercase A, I think.
For just the alphabet and ordinary numbers, there's a slightly simpler specification called ASCII that tells one how to take the 26 letters of the alphabet, upper and lower case, plus a bunch of other characters, and turns them into numbers
Between, less than 128.
When you're trying to represent, kind of, all the characters used by all the languages and so on of the world, the, the basic kind of collection of those is about 65,000 possibilities, and these days, that's been extended because everybody needs emoji and things like this.
And it's been extended to several times that number. But in any case, the main point is, when you're dealing with text, every character in a piece of text can be converted to a number. And in the end, for the computer, you're converting that number to a series of bits
And you're doing that by saying, so in a base 10 number, you know, number 127 or something, one's got, you know, the 7 is in the units column, the 2 is in the tens column, the 1 is in the hundreds column. You can do the same thing
For… for binary, where instead of being powers of… instead of being 1, 10, 100, 1,000, etc, it's…
1, 2, 2 squared, which is 4, 2 cubed, which is 8, and so on. Each
Each position represents a power of 2, and you can represent any number as 1 or 0 times each power of 2, and so you can convert any number into a sequence of ones and zeros, and that's ultimately how computers represent
Represent data in the memory of the computer.

So…
For example, in the case of a piece of text, what will happen is the text will be, well.
In the general case will be converted into a sequence of numbers that can be up to a few hundred thousand, and that sequence of numbers is then written in binary, and then the data associated with that text becomes a series of bits.
So that's how text, how one sort of takes text and converts it into sort of raw data.
Same kind of thing with images. Like, a typical image as recorded by a camera might be, oh, I don't know, 10 million pixels, so that would be a, let's say a
Let's… let's say it's 16 million pixels, just because it's a convenient number, because then that means it's 4,000 pixels across by 4,000 pixels down, and each
Pixel represents a light level in that region of the image, a brightness of that region of the image, and then that brightness is broken into a red component, green component, blue component, because all colors can be represented as… all colors that we humans can see can be represented as, more or less, at least, as a certain amount of red, a certain amount of green, a certain amount blue. So once again, we're turning this image that we are used to perceiving as a bunch of different colors and light levels and so on in different places into a big collection of numbers. So it might be 16 million numbers.
well, actually, it's 16 million times 3 numbers, where the times 3 is because we're representing the red, green, and blue, and we might say it's, I don't know, it's,
the red, green, and blue are intensities, and they're often represented using, using 24 bits each of, to say what the intensity of the red, green, and blue is. So anyway, bottom line.
the image gets turned into a big, long list of numbers. It might be, in the case of the thing that I'm saying, it's… oh boy, I've got to do the math here. It's 48 million numbers, and, in terms of bits, it would be,
Oh boy, this is, my math is failing me here. It's going to be about,
48 million times,
This is why I built tools that do math, for one, is because I was never very good at doing this myself.
But let's see, it's, I'm trying to work out,
16 times 3 is… let's, let's say, 50 times, 4 is, is, a thousand…
So that's a… A thousand million bits.
And that's 100 million…
bytes, which are 8-bit chunks, so, that would be… as a pure representation with raw numbers, that would be, I think, 100 million bytes for that whole image.
So, for a typical, piece of text.
You know, a, a…
Page of text is perhaps, Let's see, maybe, 20,000…
Characters, so that's a… a page of text is a much smaller number of bits of data to represent.
So, you know, you can go on and talk about other kinds of data, like sound, for example.
Typically, sound is a sequence of intensities that are sampled maybe 48… 44,000 times a second, and so 44,000 times a second, you're giving a number that represents the actual intensity of the sound wave.
And you're putting that together to represent sound as A sequence of numbers.
So, okay, what… what is… so you have all these different sources of input, and you can turn them all into sequences of bits.

And that's kind of the raw data associated with each of those kinds of things, whether it's text, image, sound, whatever. The raw data is just that sequence of bits.
Now, the fact is that the… all the details of that sequence of bits aren't things we particularly care about. Knowing whether in this image there is, this particular pixel has
You know, is, is, .961 of red versus point… 952 of red or something.
Humans will… the human eyes is barely able to distinguish that.
And when it comes to, does it matter, whether when we're looking at this image, does every pixel matter in the image? The answer is, every pixel doesn't matter. I mean, if you made little changes to some of the pixels, it wouldn't… the perception of the image would change not at all.
And if you ask a question like, is there a cat or a dog in this image, you can change an awful lot of pixels and not change the answer to that question.
So, you know, when it comes to what is data, at the simplest level, it's the way we take things that are kind of sensory input to us, with our eyes, ears, reading, whatever, and we turn it into something that a computer can represent.
And the big surprise, the big interesting thing, is that all these different forms of input can ultimately be turned into sequences of numbers that can be represented as bits in a computer. They all kind of turn into the same sort of thing.
And you can measure, for example, how much… how many bits it takes to represent, let's say, a piece of text, an image, whatever. The image will be very big, the audio will be a bit smaller, the text will be vastly smaller than that.
In terms of, sort of, typical amounts of stuff that we might absorb in a second or something, in terms of text, the number of bits needed to represent it is much, much smaller than for an image or for sound.
The,
So then, the question is, well, first question is, do you actually need to use that number of bits to represent
kind of the content of what you're describing, and as I was saying, for images, a lot of the bits in that image are really quite redundant.
You don't need them in detail. You could just take some much more, kind of, digested version of the image, and that's what matters to us as humans looking at the image.
And that's a thing that's much used in practical computing, because images are very rarely represented in their raw form as the actual color levels of every pixel. More often, they're represented in a compressed form. They're formats like
JPEG and GIF and PNG, and then for videos, things like MP4 and so on. These are all ways of going from the raw collection of pixels with all their values, down to something which is a much more compressed representation that contains the information that we need as humans to perceive that image.
But it's not a faithful representation of what our digital camera recorded in every pixel.
And so, often that will be a hundred times, or even a thousand times smaller number of bits needed than the raw number of bits that was needed. And how does that compression work?
So, for an image, a sort of typical thing that's done is to say, doesn't matter what every pixel value is, it only matters what this sort of, what this region of the image, what the pixel values in this region of the image were like.
And so you can think about that, well, there's… there's a variety of particular ways that those… those compression methods work. A very simple one is so-called run length encoding, used to be used by fax machines back in the day, where what you do is you say, I've got a sequence of

pixels, I'm gonna… I'm gonna scan my image one line at a time, and I'm going to say, what's the sequence of pixel values as I scan the image?
well, if you were sending a fax, for example, that would be a, you know, typically a black and white image of a, you know, piece of printed text or something, and there are large regions of that where it's just white pixel, white, white, white, white, white, for a long time. So run-length encoding works by saying, instead of giving every pixel value, you just say, I've got to run 1,000 white pixels. So you're just saying 1,000 times a white pixel. Or you may have a little region of black for being the inside of a letter or something. It might be 10 black pixels.
So that's an example of how you can compress the sort of raw data of that image, in that case of a piece of printed material or something, how you can compress that to something a bit smaller.
Well… The… the real… the original question was sort of the distinction between data and information.
And ultimately, the distinction is that data is kind of the raw representation of whatever came from the sensory device, whether it's a camera, a microphone, whatever else, whatever came from the sensory device, represented in sort of raw form for a computer in terms of bits.
That's kind of what one is talking about when one's talking about data.
When one's talking about information, one is typically wanting to talk about the aspects of that that matter to a human.
So, it doesn't matter what every pixel value is, what matters is the thing that you will perceive from the image.
In the case of text.
Well, there's a certain amount of redundancy in text. So, for example, typical thing you can do is, if you have a piece of text, you can start just sort of blanking out letters, and you can say, can I still read this piece of text? And what you'll find for English
is that, on average, about… you can blank out about half the letters in a piece of text, and if you kind of look at it real quickly and so on, you pretty much just read the original text as it was there. You wouldn't have to know, you know, when you see the word elephant, if you've blanked out a, you know, an H in the middle and an A in some place, you're still pretty sure that's the word elephant. It's got enough stuff there
to be, to make it seem like the word elephant. And as I say, in English, you typically can blank out about half the letters before you start to be very confused about what the text actually says.
So, in practice, when people store text for computers using formats like zip, for example, the gzip.gz and things like this, they often will do some compression on the text, and the compression, if it's just a generic piece of text, is usually about a factor of 2. The compression for images, where you only sort of care about what humans perceive.
might be a factor of 1,000. The compression for audio, I think it's… Hundreds of times, at least.
And… and then, so that's… that's kind of the… the compressed version is closer to what one might mean by information, by the thing that matters to humans coming out of what was given as input.
Now, actually, one can go a lot further than that, because in the end, if the,
Let's see, what would be a good example? If you've got,
If you know that you're going to be…
It's a good example of a long sequence, I don't know.
sit back, relax, and enjoy the flight type thing. It's kind of like, once you… once you have the beginning of that, and you're on a plane or something, you kind of know what the end of it is going to be. And…

in a sense, you could almost… it's kind of like telling jokes by numbers. If you… if you sort of know enough about what's… the context of what's being talked about, you can be just saying, that's a number 3.
And that could turn into a big, long.
description of, you know, a shaggy dog story about the beaver crossing the road, or something like this. Oh, that's just joke number 3. All I need to tell you is, it's number 3, and you know all of that underlying raw data about what all the characters are going to be, and all the words, and so on, and so on and so on, but I just have to tell you it's number 3.
So, when one thinks about, sort of, information and measuring, sort of, the quantity of information, one has to be, sort of, aware of the fact that there is, it's… what matters is what you need to tell
to a person, or maybe a machine, to have them reconstruct the thing that you were talking about. And that may be vastly smaller than the actual, sort of, every bit of the, sort of, raw data that represented the thing that's being transmitted.
So, a very extreme version of this is when the thing you transmit is just a little tiny program, and the receiver is supposed to run that program, like, on a computer, and see all the output it produces.
And in fact, I've spent quite a bit of my life studying cases where there are incredibly tiny programs that sort of… you can write the program in one line or whatever else, but yet you can run the program to generate huge sequences of bits and complicated patterns and all this kind of thing.
That's a place where, sort of, the amount of true information
that is being provided, it's just saying, there's this little tiny program, but the amount of data that comes out from that can be very large.
And another example of that is in something like math, where you can say the number pi. Okay, I just tell you it's pi.
Well, if you say, well, what's the data associated with pi? It's the whole sequence of digits, 3.14159, et cetera, et cetera, et cetera, goes on infinitely, actually.
So, in a sense, there's pi, the kind of… the word pi kind of gives you, sort of, in some sense, an infinite amount of data about what… there's an infinite amount of data packaged up in that one piece of information that I'm talking about, pi.
So sometimes
People, there's a certain amount of confusion around all of this, because sometimes people's meaning for these words is a bit different.
It… there's this whole field called information theory that originated in the 1940s, that had to do with kind of knowing, in part.
how much compression you can make, given that you're trying to transmit this amount of data, how much can you compress that? And that depends on, sort of, how much redundancy is there in that data. How much do you not need to say, because you can deduce it from the stuff you did say?
That's, that's one thing. Another piece of, sort of, information theory is when there's a certain amount of noise, when some of the bits you're trying to transmit don't come through the way you intended them to come through, how much can you then reconstruct? That's sort of the other big piece of information theory.

And those are very related things. How much redundancy is there? How much noise can you withstand? I mean, when you hear a person talking in a loud room or something like this, there are pieces of what they say that you don't hear because there was, you know, a loud
A loud noise right when they were saying that particular syllable of that particular word.
But the fact is, you can usually reconstruct what they were saying. You can often reconstruct what they were saying. If you know kind of what you thought they were going to say, you have a much better chance to reconstruct what they were saying when there's a certain amount of noise. If they're telling you, you know, names you've never heard before, and there's all kinds of background noise, you'll keep on saying, what did you say? What did you say?
and so on. But if they're telling you things where, really, you only have to hear one word out of three, because you kind of know what they're… roughly what they're saying.
then you can withstand a quite high level of noise. Information theory gives you a sort of more mathematical way to talk about that phenomenon. But the thing about information theory is people often sort of say, well, information theory tells you how much true information there is in a message.
It's not really correct.
Because the mathematical structure of information theory, tends to be about particular ways of doing compression. It's a little confusing, because the original, sort of, idea of information theory was to say, sort of, how… to think about, sort of, what are all the possible messages you might receive.
What, and that's kind of the true information is I'm receiving one of these whole messages.
And… but then there's the question, how is the message actually represented? Well, it's represented as bits, and then there's sort of the way of kind of joining up the data part of the representation of the message with the true information that's being transmitted. Which, out of the thousand messages you might transmit, which one are you actually transmitting?
Well, a lot of what was done in information theory, and it's
first, well, in traditional information theory, is that one is using particular ways of compressing and encoding the data. Usually, you take a block of
of bits, and you say, well, it's sort of redundant if we know that if it started 110, it's always going to continue 001. So we don't have to give the 001 at the end, we can just give the beginning.
And…
there's this way of doing, kind of, block encoding, which lets you specify, kind of a certain amount of compression. It's not all the compression.
For example, for the digits of pi, you can do… you can say, well, there's this block of digits, there's that block of digits, but in the end, all… you'll only be able to compress it a bit by saying… actually, with pi, not very much at all, by saying what blocks of digits occur. You can't compress it very much at all, because the digits of pi contain its… so far as people guess, every possible block of digits appears somewhere
in the digits of pi. We had a nice thing a few years ago about find your birthday in the digits of pi. That's possible because any sequence of digits appears somewhere in the digits of pi.
Well…
The, but nevertheless, when you're sort of counting amount of information, that means if you're just looking at digits, you're going to say there's tons and tons of information in pi. But actually, sort of looked at from the outside with the context of what you're talking about, it's just… we just have to say pi, and that encodes an infinite amount of data.

but also an infinite amount of information in the sense that people often use the term information in information theory, where you're talking about, kind of, compressing things by saying which blocks occur. So you can go a bit further.
And go to what's called algorithmic information theory, in which you're saying not, well, what blocks of digits occur, but what
what size of program do you need to reconstruct the, the data that you got? So in the case of pi, you need a fairly small program that's just a pi-digit generating program that can go off, and if you run it for long enough, it'll generate all the dig… you know, just keep generating digits of pi forever.
So it seems like a really good idea, algorithmic information theory, so it seems like a really good idea as a way of capturing the true amount of information in something. How big was the smallest program you need to reproduce the thing?
There's a bit of a bug, though, because there's a question of, well, okay, you have a program, but how long did it take to go from the program to the actual thing you were generating?
And the problem is, and this is a sort of a deep problem in the theory of computation, the problem is, in general, you can have a program, it can keep running, it can keep running, it can keep running, and it's like, is it going to generate what I want, or is it just going to flap around forever and just be in a loop and never come to a conclusion?
The answer is, in general, you can't know.
It's related to a phenomenon I've studied a lot that I call computational irreducibility. When you run a program, in general, you can't expect to jump ahead and say what that program will do. You're always, in the end, stuck with having to go step by step through the program to see what happens. Some programs, you can jump ahead. If the program is just repeatedly saying 1, 0, 1, 0, 1, 0, and you know it's just going to loop like that forever.
It's easy to jump ahead. But in general, you can't do that. And so algorithmic information theory, while it gives you, in principle, a way to say what's the true information content, the minimum program needed to specify this data, in practice, you won't, in general, be able to do that, because given if I give you this program and say, hey, I claim this program generates this data.
You could have to spend an arbitrarily long time waiting to see whether that actually is true.
So there are kind of versions of that that are kind of, say, well, you know, what's the smallest program that you only have to run for some number of steps that generates this piece of data?
In… in general, it's,
The… the question of, kind of, what, how much.
true information is there in a thing depends very much on who is looking at the thing, what outside context do they have? What level of compression can they effectively do with whatever resources, whether it's, sort of human brains, or whether it's computers, whatever they have.
So anyway, the…
The sort of bottom line here is data is really about, kind of, the raw representation of things as they come from whatever sensory device is picking up that input.
And information is more what is the… what is the thing you need for a perceiver of that information to determine what is being said, what it is that's being said. You don't need all the bits. You only need something which allows you to either reconstruct the bits or get what you need from the bits to go on and come up with the next conclusion.
That's a more subtle answer than I was expecting.
Let's see… Oh, boy.
the,

All right, a follow-on.
From talking about bits,
Well, okay, I'm gonna do two things here. So, Klepto comments, does compression not have lossy and lossless versions?
Yes, that is true. I was slightly eliding that point. Let me explain what that's about.
So… If you have an image, for example, you might have millions of pixels in the image.
And when I was talking about run length encoding, where you say it's a sequence of white pixels, there's a thousand white pixels, and then there's black pixels, and so on.
Run length encoding, if you say there's a thousand white pixels, even though it's much shorter to say a thousand times white plus 10 times black and so on, than it is to go white, white, white, white, white a thousand times.
Even though it's much shorter, if you're given that specification of a thousand times white plus 10 times black and so on, you can take that shortened specification, and you can reconstruct exactly the original pixels that you had.
That's how, for example, well, some aspects of TIFF compression work that way. To do lossless compression of images. Usually, you don't need or want to do lossless compression of images, because the details of every pixel don't matter.
And so, for example, JPEG, PNG, GIF,
They all do lossy compression of images, which means that you're… you're just sort of keeping… so, so, okay.
With specificity, what happens is,
In, in, for example, JPEG, a typical thing that's done is you say.
let's represent the image in terms of what are called Fourier components. And what that is, is you're saying, across the image, it's like you imagine you have kind of a wave that goes, you know, black to white, black to white, black to white.
And that wave can be very long wavelengths, so across the whole image, it's just going from black to white once. Or it could be a wave with very high, high frequency, small wavelength, where it goes black, white, black, black, black, white, every few pixels.
The point is, you can represent the image by adding up this kind of, you know, lazy, long waves with very frenetic short waves, but
To us humans, to our eyes, the details of all those frenetic short waves don't matter very much. The specific variation from one pixel to the next door pixel, unless it's on the edge of an object in the image, doesn't really matter very much.
So you can drop those very high frequency, short wavelength pieces, and you can just be left with the long, lazy, kind of slow variation kinds of things.
And so, in JPEG compression, what you typically are doing is to say, let's decompose the image into the kind of slow-changing, slowly changing, big blobs of identical color, and these very fast-changing things. And in some places, like at edges, you might have to keep all the fast-moving stuff, fast-changing stuff, but most of the time you don't. And that allows you… that's sort of a core idea in compression in that case.
In the case of things like text.
One of the core ideas is so-called dictionary compression, where essentially you say, well, there are a certain number of words in the dictionary, and in some first approximation, the common words get short code words, like probably, you know, something like,

oh, I don't know, the word there, or something, which has, you know, a bunch of letters, but it's pretty common, you know, might get a very short code word. That might be word number 23, and the word elephant might be word number 4000.
And, you're… you're trying to encode short words, more common words with short code words, and the idea is that's a… that's a lossless form of compression, because you can always go back and go back and say, what was word 27 in the dictionary? Okay, it was this. It's exactly this.
So that… that's… and by the way, in the case of audio, it's typical. MP3, for example, does lossy compression. There are other kinds of files that do lossless compression. I think,
Black files, I think, do lossless compression. The… what happens is the original audio signal is a series of numbers that represent the intensity of sound at every tiny fraction of a second. Because what's happening, you know, sound is this sequence of compression and rarefaction waves in air, and so what's happening is, you know, the loudspeaker cone is moving, and it's pushing the air, and it's making a compression piece, or whatever else, and, you know, the… if you… the frequency of the sound depends on how fast the loudspeaker cone is going in and out. So, for example, middle C on a piano is roughly 256 times per second of the loudspeaker cone going in and out. And so what's typically done
Is to sample sound 44,000 times a second.
So that means, gosh, I have to do my… some math again. That's, every,
every 1 44th of a thousandth of a second. So, that means if the loudspeaker cone is going in and out 256 times a second for middle C, that means within a single middle C, one's dealing with about 1,600, different samples of, how much… what was the compression of the, of the air in that time, what was the
profession of the air in that time, 1,600 times in a single loudspeaker cone wiggling to make a middle C, note.
But so, what happens in sound, again, is most of the time, as in that middle C note, most of those 1600 samples in every wiggle of the loudspeaker cone, they'll be kind of more or less the same. So you don't have to take account of every single one of those.
Actually, the, the way that, so, that was,
That's, so that's why you can do compression. You can do lossy compression, because to our ears, it doesn't really matter whether you had, you know, 1500 of this intensity followed by 7 of that slightly different intensity. None of that is anything we could distinguish.
So you can do compression for sound, by using this kind of frequency idea, by saying what is the mix of frequencies, a little bit like what I described in,
in the case of images, actually, in the original case of MP3, the thing that was done was something a little bit more interesting and elaborate.
Which was, it was originally intended to be a compression method for dealing with human, speech.
And human speech, we know human speech is generated by our human vocal tract. It's generated by changing the shape of the vocal tract and the mouth and so on, and having our vocal cords vibrate and produce sounds.
And in a sense, the kind of… the set of sounds that can be produced by humans is determined by the kind of shapes that humans can make in their vocal tracts. Something like birds, for instance, have a different construction of their vocal tracts. They have syrinxes.
And, for example, they have the possibility of making… we have vocal cords that can sort of make one frequency of sound at a time.

Where, more or less.
Whereas birds can make multiple ones, and so their kind of twitterings are somehow qualitatively different from human ones. And you can see that if you look at spectrograms of bird songs versus human speech, for human speech, we have these kind of dominant frequencies, so-called formants, that we make, where there's sort of a dominant frequency of the sound that we're making at every given moment in time.
that frequency changes over time, and that change of the frequency over time is what gives us, for example, different vowel sounds and so on. But so what was done in MP3 was to make essentially a model of the human vocal tract
And to say, well, wait a minute, you're making an E sound. Okay, if you're making an E sound, that's determined by the shape of your vocal tract, and there are not very many different numbers you have to specify to say it's a vocal tract with this shape rather than that shape.
And so, the idea was more or less to say, well, what were the numbers that you have to use to describe the human vocal tract?
to reproduce what sounds it was making, let's just send those numbers, let's just include those numbers as a way to represent… as represent sounds. Ironically enough, even though the whole setup was originally built for human voices, it was immediately applied to music.
Where the sounds being generated are of a bit of a different character, but it turned out those methods worked, worked well in that case as well.
But so, MP3 and so on is a lossy form of compression. When you're compressing videos.
for example, with MP4, another big thing about videos is that
that from one frame to the next to the video, the video might have a new frame every, you know, 60 times a second, or something like this, or 30 times a second, but most of those frames, it's kind of like the same thing's happening in the next frame as the previous frame. So the raw data from all these frames is huge.
But the compressed version of it is much smaller, because among other things, from frame to frame, you don't have to store all the values of all the pixels in every frame, you can just sort of say, hey, this pixel's the same for a thousand frames.
And typically, and that kind of compression is also lossy, because we humans just absolutely don't notice these very tiny changes that happen. Now, occasionally, you'll have situations where where, kind of, lossy compression gets you into trouble. I've had that situation because I often generate these, images that are computationally generated, where sort of every pixel is computed by a program, and every pixel is different.
If you have a photograph, usually the pixels have some smooth variation in most of the photograph. In the kinds of things I've often generated, it's like every pixel is a separate computation.
And so, if one tries to do lossy compression on that kind of image, you'll get… you often get very terrible things happening, and you often get a lot of very weird effects that, are perceivable to humans and aren't really anything to do with what's in the underlying image.
But so, lossy compression has this character that is supposed to give you the stuff that matters to us humans, and sort of ignores the stuff that doesn't matter, at the… and the benefit of that is, by ignoring the stuff… by ignoring all that stuff, you get to compress the data a lot.
Whereas lossless compression, you're supposed to be able to reproduce the original data perfectly by… when you receive it.
So, there's a question here from memes.
about, the Landauer limit.

And, well, let me try and talk a little bit about that.
So… One of the questions about,
When you have a computer, and the computer's doing all these things with bits.
And it, like, might be doing, sort of, at the… in the end, a modern computer is made of silicon and semiconductors and so on. A modern computer, the CPU of a modern computer, is billions of gates
that, do the operation of NAND, actually NOR, usually, which is kind of equivalent to NAND.
What does that mean? It means, little pieces of current
There's a detail about whether it's current or voltage, let's not worry about that. Basically, a little piece of electrical… a little electrical signal comes in, and every gate has two inputs and one output.
Two inputs come in. Each input is either there's a bit there, or there isn't a bit there. It's either a 1 or a 0.
And it's either a true, there's a bit there, or false, there's no bit there.
And every inside a computer, it's a very elaborate circuit that's made of a huge number, a billion or more, of these little NAND gates. And every NAND gate takes, takes two inputs, gives one output.
the, what does the NAND gate do? Let me explain a simpler case of an AND gate. An AND gate says.
are… is… are both the inputs true, or one, or on? Are both the inputs true, or not?
If both the inputs are true, then the output is going to be true as well. If either one or both of the inputs was not true, then the output will be false, zero.
So, that's what an AND gate does. It's kind of like… well, it's, yeah, it's… if you think about the inputs as 1 and 0, an AND gate is basically multiplying the two inputs together and giving the output.
Okay, a NAND gate does the NOT version of that. So whenever the AND gate would have given a 0 as output, a NAND gate gives a 1 as output, and vice versa. So it just turns out that you can make any logical operation as a combination of NAND operations.
So, for instance… oh boy, can I do any of these? They get a little complicated. But basically, a NAND of P and a P, and then a P, NAND of Q comma.
R or something.
will be, and I can't do this in real time, sorry, that might be an OR, for example, I'm not sure if it is. But you can combine NANDs together to make ANDs and ORs and NOTs and any other logical operation you want to put together, and that's why… that's why those things are used as the raw material out of which you make computers and CPUs, is because by arranging those NANDs in all the right ways, you can make all the things you need in a CPU
Thank you for adding numbers and multiplying numbers and jumping to this place and that place, and so on.
So… Okay, that's… that's the raw material of a computer, isn't… is NAND gates.
Well, one feature of NAND gates is they always have two inputs and one output. So we're talking a little bit about lossy and lossless compression and so on. One thing is, if you've got the input.
You got, let's say, two bits of input, but you only have one bit of output. So there's no way you can uniquely reconstruct the input given the output.
You got… because you could have, for example, in the case of a, let's say, an AND gate.

If you have 10 as the input, it'll give 0. If you give… if you have 01 as the input, it'll still give 0. So just knowing the output was 0 doesn't tell you whether the input was 1001, or 00. Could be any of those three.
If the output was 1, it so happens that for an AND gate, you then know that the input was 11, but in general, you can't know. There's an ambiguity. On average, there's sort of twice as many things that could have been there in the input than you can know from the output. Okay, so that's the typical situation.
Well… The… there's… next, there's the question of Okay, when…
Let's see how to explain this. So, let me tell you where I'm going, and then we'll talk about how we get there. Essentially, when you have
two bits coming in, and only one bit coming out. What has to happen is a certain amount of energy has to be dissipated to go from those two input bits. Think about it as two bundles of electrons coming in, but only one bundle of electrons is going to come out.
And somehow, the energy associated with those bundles of electrons coming in, something has to happen to that energy in order for one to only have one bundle of electrons coming out.
I'm cheating a little bit in the way I'm describing this, but let's go with this kind of way of talking about it for now. Well, what has to happen to get only one bundle out is you've got to dissipate half the energy associated with what came in.
that… how do you dissipate energy? Well, you have to turn it… what was sort of electrical energy, you have to turn it into something that, you know, that… it's sort of a generic thing that can flow out of your computer, and that's basically heat.
So what is originally sort of the organized bundle of electrons that represents one of those bits. As you dissipate the energy associated with that, it turns into just this incoherent,
bunch of little motions of the atoms inside your piece of silicon or whatever else. That's what heat is. It's this sort of incoherent motion of lots of microscopic molecules or atoms in a material.
So what happens is, two inputs come in.
And, you're dissipating the heat associated with one… dissipating energy associated with one of them, it's dissipated as heat. What happens to that heat? It heats up your computer. And that's what, in… if you go to a data center, for example, that has racks and racks of computers.
you'll find there's an awful lot of heat being generated there. Usually, a data center has these sort of cold aisles and hot aisles, where all the computers are spewing their heat out into this hot aisle. Where… where it's quite hot compared to the other side, so to speak.
And fancy data centers have all kinds of ways to get rid of that heat. It's one of the things that is… it seems very disappointing that all the power that's going into your computer, you know, much of it is being dissipated as heat.
And the reason that's happening is that you have this, this phenomenon where you have to go from, sort of, two inputs to one output. You've sort of got to get rid of the energy associated with one of those inputs. I'm cheating a little bit in the way that I'm saying that, but it's more or less, more or less the idea. The,
the thing, so it was believed when computers were young, particularly John von Neumann, a sort of pioneer of thinking about computers, just believed it was inevitably the case that every time you kind of did one of these gates.
You would, have to dissipate a certain amount of energy.
And, Ralph Landauer, who I knew, was the person who sort of codified that idea, but then, it became clear that, and this is happening in the, 1970s, it, it became clear that it wasn't actually necessary

To dissipate heat
and do computation. You could have it be the case that instead of having a NAND gate that has two inputs and one output, you could have a gate that had two inputs and two outputs, so that in principle, you could always go back
from the output to the input, you didn't have to get rid of any energy, you could just have that in the output as well as the input.
And the question is, was that practical? Because the problem is that often in a computation, you say, hey, I've got this, this question, it's going to turn into all these intermediate pieces of the computation, and in the end, I just want to know, is it yes or no?
And somehow, to get down to the is it yes or no, you have to get rid of all this extra… extra bits and so on that you were preserving because you had every gate be a 2 to 2 gate.
And so there's sort of an elaborate way of thinking about it. You do a computation, you generate a lot of garbage bits, and you throw the garbage bits away. Sort of the discarding of the garbage bits, in some sense, is going to… is going to dissipate heat.
But…
But you can perhaps do that in a more controlled way if you just, you know, if you do most of the computation with all this back and forth, all this back and forth going on in a way where you're never losing any bits. And then, at the end, you've got a bunch of bits that are going to be the garbage bits, and you dump them all at once. You can end up with a lot less heat being dissipated when you do it that way.
Now, a place where this all got very relevant is in quantum computing, when you think about… the standard formalism of quantum mechanics assumes that no bits are lost, no information is lost. The standard formalism of quantum mechanics says that every
gate that's happening in a quantum circuit must be two inputs, two outputs, or one input, one output. Same number of inputs and outputs, so that it's always a perfectly… it's always something you can reverse. You can always say, I've got the output, I can go back to the input.
So, in quantum mechanics and quantum computing and so on, one of the big problems is going from
this, oh, we're preserving all the bits, to a, oh, I just want to know about the output and the specific, a specific one of these bits. That's usually called quantum measurement, and it's a big can of worms, which I actually think is not properly being thought about and hasn't been for a very long time.
Actually, interestingly enough, John von Neumann was the person who also came up with the standard mathematical formalism that's used to think about quantum measurement and so on.
But in any case, so quantum computing and thinking about quantum computing has made one more interested in this kind of idea of reversible computing, computing where you can feed back the output and get the input back from it. Now, actually, there have been, for many years, sort of a trickle of attempts
To,
to be more serious about, about reversal computing. There's a company now called VAIR that I've been a bit involved with that is trying to make, sort of a serious
attempt to do reversible computing. I mean, they're… they're kind of edging into it by… by doing some things and making semiconductors work in slightly different ways. It isn't the full story of, we're going to, be able to fully reverse every computation, but they're kind of making use of sort of the energy that's being generated and used in every
In every operation.

And, sort of, it's an interesting question, how you can formulate thinking about computations in a reversible way, so that, sort of, no energy has to be lost in doing the computation. It's something that's still, I would say, not well developed.
I mean, one can certainly imagine a whole sort of programming language that's all reversible, and where you only sort of have to throw away the garbage at the end, and you could always reverse it at every step.
There are all sorts of interesting reasons you might want to do that. In addition to the fact that you might be able to save lots of dissipation of heat. You might want it to be the case that you're doing a computation, and you sort of speculatively try and do something, and oh, it didn't work out. Let's back it out. Let's literally reverse the computation and see what happens. So, for example, the thing you might want to do
is you're doing a computation, and something goes wrong. There's a bug of some kind. It produces… does something you didn't want it to do.
Imagine if you just say, well, I got to this state, let me just literally back it out. We didn't have to explicitly store checkpoints of what the state of the thing was a little bit back. We could just say, given where I'm at, I'm just going to run the computation backwards and see where I get to.
And so that's… that's a place where you might want to do that. This is still very much a kind of beyond-the-leading-edge sort of thing to be thinking about doing in practical computing, but that's kind of the idea of reversible computing.
Let's see… Well, there's a, there's a,
Comment here from Double, saying they think data centers should be built alongside residential buildings so the waste heat can, keep the homes heated.
Okay, it gets more amusing than that. There's a company we had some interaction with, I think it's called Quano, and their business model was to provide heaters for residential buildings, and particularly hotels, where the heater is a computer.
And where the, you know, heating the hotel room is doing computation.
And, that's, so yes, I mean, that's a… that's a concept, and you can also say, you know, let's have my computer
Double as my heater, And maybe you can even do something where you're, you know, you're saving
money by having your heater double as a computer, and then you're selling the compute back to some kind of cloud provider as a way to get sort of a rebate on your heating costs. So yes, that's an interesting thing one could think about doing.
In general.
Kind of data centers these days, kind of, they need a lot of electrical power, and they need, to be able to dump the waste heat.
And, often, you know, there's been a lot of, sort of, you know, data centers near hydroelectric plants where electricity is cheap.
There's lots of discussion about nuclear power for data centers and so on.
I have to say, I think that there's a…
There's more pressure now to figure out how to do computation in a way that doesn't take as much energy, because you might have thought, as people did back in the 1950s, it's inevitable that computation XYZ must take a certain amount of energy, must dissipate a certain amount of heat. We know from the theoretical idea of reversible computation that that just isn't true.
that you can, in principle, do computation reversibly. Now the problem is, can you actually build computers and hardware that successfully do that? And that's a whole challenge of its own.

Let's see…
Well, let's go back to,
Some earlier questions here.
Patrick is asking, If I cloned my brain exactly.
Would the copy think it was me?
I suppose, what does me mean, and what does think mean?
I think the, we are used to having this sort of single thread of experience.
We think of ourselves as ourselves, and even though we go to sleep.
And we're not kind of… we're not kind of remembering every… every past instant. We wake up in the morning, we're used enough to going to sleep and waking up again, that we kind of say, yeah, that's… that's the same… that's the same me as it was last night.
That we have this kind of continuity of thinking it's us from the inside.
Now, if you… If you say, does…
the copy of me have that same continuity of thinking it's the same… it's the me, it's the same thing going forward. The answer should be, if it's a perfect copy, it should be, yeah, it's gonna have the same… the same thinking that it's… that it… it is the… it is it.
It's gonna continue thinking it's the same it, the same, quotes me, that it was before.
Now the question is, does the copy think that?
It's… The same me as the original was.
I think the answer to that is really, in many senses, no.
It's kind of like…
imagining… so, so a couple of things to say. I mean, imagine you have identical twins. I don't know what it's like to be an identical twin, because I'm not.
But, the, you know, there's a certain sense in which assume, and it won't in fact be the case because of the way that brains form. Brains of identical twins, just like fingerprints of identical twins, aren't identical.
But they have many of the same features, but they're not, sort of, neuron for neuron identical. But let's imagine even that they were.
As soon as the… the, it is not the case that the, You know, each twin
has sort of a continuity of experience of, I'm twin A, I'm twin B, and they have a continuity of experience within themselves. But I don't think there's… there's a kind of, oh no, I'm actually twin B, thinking I'm actually twin A.
Now, confusingly, Twin B might be able to say, I know what Twin A is going to do next, because I can sort of run themselves in my brain, so to speak, and see what's going to happen.
So in some sense, from the outside, it might look like Twin B is doing the same as what twin A would do, but that doesn't mean that the inner experience of twin B is merged with the inner experience of twin A. Boy, it's a little bit complicated.
I think the, I mean, this is a question that certainly comes up when you think about, oh, what if I could upload my brain to some digital, brain vault somewhere, mind vault somewhere? Would the thing in the mind vault
Think it's me or not.
I think it would think it's it.
Would it think it's… if… If it…
It's sort of an interesting question. If its senses were sort of enough blacked out, that it had sort of no idea where it came from, and it had no ability to see the original me.
then…

I suppose if you ask it, you know, who are you, it's going to go back to its memory that came before the fork where there was a copy made, and it's going to respond, I am… I am the original, because what else could it do?
But on the other hand, as soon as it sees the original, it knows it's not the original, so to speak.
So, I think, and… and, you know, in… in the question of.
do you wake up, and now you're a digital you? What does that mean? Because it's…
It's… the thing that wakes up is going to have its own continuity of experience.
But the thing that wakes up could have, its…
if you've… if you prevent it from having a knowledge of the fact that there was, you know, what its origins were, and that there is that… that you that still exists, or maybe it doesn't, I think, boy, this is… this is tangled, isn't it? The, I mean, I think the…
yeah, it's… it's a bit of a tangled story.
It really depends on what it means to think it's me, and… What it means 2…
You know, it can think it is a definite, existent being that
that if it knew nothing else, it will think it is just the continuity of the memories that came from it originally. I think that's probably the best one can say about that.
Well, Charles is commenting, I think,
That the… the copied mind is like having the data with, without…
there's a question of, sort of, how much you're copying the memory state as well as the machine that's running. Yeah, I mean, for sure.
in… If you have, kind of, a copy of a mind.
And it is experiencing the world.
The experience of the world from two different minds.
Will be different.
Unless… unless those minds are hooked up to the exact same sensory input, the experiences of those minds will be different.
And so, very quickly, the kind of… the what's in one mind will differ from what's in the other mind.
I think the, when it comes to, for example, neural net models and things, and you can do this kind of thing with those, there's enough kind of randomness that's used in the training of neural net models that, sort of, even though you might have two neural nets with the exact same input.
unless you very carefully lock down the kind of randomness that's typically used in training, you'll end up with two different arrangements of actual remembered stuff in the neural net. And so you'll end up with, again, not sort of identical behavior in the two neural nets.
the,
Let's see… Boy, this is another…
sort of philosophical question from… from Pac.
inso… Well, they're sort of assuming, insofar as cats.
see humans as bigger, dumber cats? How would robots see humans? Would they understand the difference, or would they just assume that we are dumber robots?
It's a reasonable question, it's a question that goes to the core of a lot of the way that we perceive the world. I mean, the fact is, when people say there are, you know, spirits of the forest, so to speak.
There are things where, sort of, things are happening in the world that seem like they have minds associated with them, and they even seem like they have human-like minds associated with them.

you know, the… the gods are playing with us, sending this thunderbolt or something. It's… it's kind of a thing where…
We tend to project
onto, for example, things in the natural world, something which is like our inner experience of how we make things happen. I mean, for all of us.
We have our own individual inner experience of how things happen.
And the fact that we assume that other people have that same kind of inner experience is merely an assumption. We can never get inside their minds to have that same inner experience. It's just like, oh, that's a person like me, therefore they must be having experiences like mine.
Well, similarly, when we see things happen in nature, or we say animals do things, it is a natural thing to kind of project, from our human feelings about how things get done, those same feelings about what's going on inside the animal to make that happen.
It's also true with… we project onto how computers work. I mean, my computer is having a hard time with that. My computer is struggling. You know, it's… you can hear its fan running, its, you know, its CPU is running full out to do this particular computation. My computer is struggling.
Well, that's sort of projecting a human-like characteristic onto the computer, and that's something that it is very much our model for the world, and that's certainly true in the sort of pre-scientific era. Our model for the world
is much more… it's all… it's all human-like stuff going on in the world, even though, in some sense, it's really just nature and not mind-like or human mind-like stuff. So I think there is a natural tendency of at least minds like ours to project
That there are also minds like ours in other things that do anything that, sort of, in effect, is like what we do.
You know, that, I don't know, that, that fish.
is, what's a good example? That cat.
wants to relax after it's been chasing a mouse for a long time. It wants to kind of chill out and relax.
well, you know, is that really the right description for the cat? You know, that is a human… impression projected onto a cat. Now, the truth is that cats probably have many of the same emotional states that humans do, probably because the actual neurotransmitters, the actual chemicals in their brains, are not that different from human chemicals in human brains, and so they tend to have the same kinds of emotional
Possible emotional states that humans do. But that's, again, kind of a projection from us humans. So, I think it is sort of a natural thing that any mind with particular inner experiences can have a theory about how other minds or other things work that is based on its inner experience. So, yes, I think it's probably the case that, in some sense, the robots
are projecting their inner experience onto ours. Let's be more specific about that. If we imagine a neural net.
that is…
trying to model, sort of, human behavior, how is it doing that? For example, let's say it's a tutoring bot that's trying to make a model for what the student understands and doesn't understand. It's doing that implicitly by having a representation in the neural net way of thinking inside the neural net
That is kind of an approximation to the human way of thinking on the outside. So, in some sense. What, it's inevitable that there's a representation in the… in the thinking patterns of a neural net about what the thinking patterns of a human on the outside might be.

All right, let's see, maybe one more question, and then we should wrap.
day.
Oh, boy.
Should we try one more?
Very different kind of question.
All right, I'll do one very different kind of question.
from double.
If I had a spaceship that had a device that could disable or decouple all of the Higgs bosons from all the particles of the spaceship, would the ship be able to travel at the speed of light? The answer to that is yes.
Let me explain that a little bit.
So… Anything that has no mass.
And the only example… well, the examples… main example we know is photons, particles of light.
It is an inevitable feature of things that have no mass, that they travel at the speed of light. It's kind of almost tautological to say light travels at the speed of light.
There are other particles that are massless, like gluons, but the story is a bit more complicated there, because they get… they're always stuck inside… inside other particles.
But, anything that has no mass travels at the speed of light. That's almost a definitional point.
So the thing is, how do things get mass?
Well, in the standard model of particle physics, things get mass by… every time they move, they keep on interacting with this thing called the Higgs field.
And the idea, which is a little bit of a hack.
is that, in the universe, there's a sort of background Higgs field that fills the whole universe.
It's a bit embarrassing, because in the usual way that things are set up, that background Higgs field has so much energy and mass that it would curl the universe up into a tiny ball as a result of its gravity.
Doesn't happen in the models of physics that we built in the last few years, but that's sort of in the traditional model of physics. That's one of the problems.
But so the… the way that things get mass is they just keep on interacting with this Higgs field.
It's like, I'm trying to, you know, I'm trying to just move at the speed of light. But no, I keep on getting, sort of, I keep on getting a kick.
from this Higgs field that is, sort of, throughout the universe, and that series of kicks is making me not travel at the speed of light.
It's kind of like when light goes into a medium like water or glass or something, it's… what's happening is the photons of light keep on getting absorbed by an atom in the water, by a molecule in the water, then they get re-emitted. It takes a little moment.
for the molecule in the water to say, oh, I absorbed a photon, wiggle, wiggle, wiggle, eventually I'm going to re-emit that photon. It's not quite the same thing in the Higgs mechanism for mass, but it's more or less the same idea, that you're continually getting kicked by these interactions with Higgs particles.
By Higgs… actually by the… this Higgs field that… that sort of covers the universe.
Higgs particles are little wiggles in the Higgs field.
It's kind of like saying the Higgs field is like your ocean. It's just a whole bunch of water. But then, on the surface of the ocean, you can have a little wave that's wiggling that, you know, you could have this perfectly flat ocean, and the ocean is all there.

But then, you know, if you want to kind of send some signal across the ocean, you'd make a little wiggle, and the wiggle would propagate as a wave.
Higgs particles are those waves operating in this sort of ocean, Higgs ocean, that is the background Higgs field of the universe.
So, if you say, let me decouple everything from the Higgs field, from every interaction with Higgs stuff.
Then the answer is, if you could do that, you would have a thing that was massless.
And the, the, you know, one issue with that is, if you've got your spaceship, and your spaceship is, is, hanging together, it's, you know, it's got, it's the, you know, the spaceship is a… is an integrated thing, and it just is… is your spaceship, and it's a definite shape.
And suddenly that spaceship is all made of photons, it's not going to be that thing anymore. Sort of thingness requires
a certain, requires mass. In order to be a thing, you can just sort of pick up and be there. It has to have mass.
If you have things that are purely massless, then every two things that are sort of next to each other, but not in exactly the same place, are just going to go in different directions at the speed of light, and there's no longer sort of a thing that's there. Not in our usual sense of a thing.
I mean, you can… you could have a whole bunch of photons that are… that are all in some pattern, they're all traveling together, and I suppose you could imagine that as a little bit of a spaceship. Actually, one weird thing about that spaceship is
in no useful sense. You'd never be able to walk from one part of the spaceship to another and back again. It would just be… it's just like this projection of a spaceship that's just, you know, all the pieces of the spaceship are just traveling at the speed of light, they're never interacting with each other. So that's what sort of happens if the spaceship is converted to something that's massless. And, you know, I have no idea how you would
you would isolate yourself from the Higgs field, but that's what would happen if you did. It's like, you don't get a spaceship anymore. You just get this kind of traveling thing that is made of something like photons, of massless particles.
All right, well, thanks for some interesting questions.
Time for me to go to my day job. Actually, I think I'm doing a livestream design review in just a couple minutes here.
So, thanks for joining me, and see you another time.
Bye for now.