Building the most AI-pilled engineering team in the world | Fiona Fung (Anthropic)

98 MIN · YOUTUBE · HTTPS://YOUTU.BE/YBRL4FYM57C
https://youtu.be/Ybrl4FYM57c

SUMMARY

Fiona Fung, an engineering leader at Anthropic, discusses the transformative impact of AI on software engineering, emphasizing the shift from traditional coding to AI-assisted development. She highlights the importance of maintaining a strong team culture and the need for engineers to adapt to new roles while ensuring accountability and quality in their work.

- *Anthropic engineers now ship eight times more code per quarter, showcasing the efficiency gained through AI tools.*
- *The shift in engineering roles emphasizes high agency and accountability, encouraging proactive problem-solving.*
- *Engineers are encouraged to maintain a strong product sense and engage with the product they build, fostering a culture of continuous learning.*
- *The loneliness of working with AI tools has led to initiatives like pair programming lunches to enhance collaboration and knowledge sharing.*
- *Fiona advocates for "just-in-time" planning, focusing on monthly priorities while allowing for flexibility in response to rapid changes.*
- *The importance of maintaining a healthy team culture is paramount, especially during periods of rapid growth and change.*
- *Fiona emphasizes the need for engineers to understand the underlying architecture and dependencies of their work, even as automation increases.*
- *She encourages sharing knowledge about AI tools with those who may be hesitant, to bridge the gap between early adopters and those resistant to change.*

Anthropic engineers, on average, ship
eight times as much code per quarter as
they did compared to 2025. [music]
>> Coding is no longer the bottleneck. It's
lifted the ceiling of what anyone is
able to do.
>> Everything is now possible in theory.
Now, it's about how ambitious [music]
can you be?
>> It's always something we ask ourselves,
what's better than me doing it? Haven't

thought, David.
>> The people that seem to be doing best are taking the most initiative, getting the most proactive, have the most agency.
>> We say with high agency is also high accountability. So, it's all about making sure folks have that freedom to code. But, then it's also like, okay, what's the accountability for it? What's the hypothesis of what you're trying to solve?
>> I'm curious what is lost in this new world of software engineering.
>> It can start being a lonely experience cuz we all started just working with our agent so much. And on the Claude Code team, recently, we started up pair-wise programming lunch.
>> Something you think about is this gap forming [music] between people that are leaning into AI, killing it, and then people that are not, super frustrated, fighting, resisting.
>> In terms of frustration, I think sometimes I also see a little bit of fear. For anything that there is a fear, my advice is lean in and ask, [music] "What can I do about it? What is within my control?"
>> Today, my guest is Fiona Fung. Fiona leads the teams behind Claude Code and co-work at Anthropic. She oversees both Boris Cherney and Kat Wu, both of whom who have been on the podcast and whose episodes are in the top 10 most listened to episodes of all time. Before Anthropic, at Microsoft, Fiona ran the teams that built TypeScript and Visual Studio. After that, she went to Facebook, where she started the Facebook Marketplace team, which [music] she took

from idea to launch. Today, Facebook Marketplace generates over 100 billion dollars in GMV every year. Also, while at Meta, she oversaw work on Meta's first smart glasses product, and then she helped build Orion, their first AR glasses product. Then, she went Instagram, where she led infrastructure, growth, integrity, and safety teams. While at Instagram and at Meta, she [music] oversaw an org of over 500 people. Fiona has been an engineer for over 25 years and as a long-time engineering leader, especially now [music] at Anthropic, she has such a unique lens into where things are heading, what's worth paying attention to, and what teams should be thinking about right now as [music] AI transforms the world of building. A huge thank you to Kat Lo, Boris Cherney, and Mohammad [music] Hegazi for suggesting topics and questions for this conversation. Before we get into it, don't forget to check out Lenny's product pass.com
>> [music]
>> for a free year of the hottest and most well-crafted AI products in the world available exclusively to Lenny's newsletter subscribers. With that, I bring you Fiona Fung.
>> [music]
>> Fiona, thank you so much for being here and welcome to the podcast.
>> Thanks so much for having me, Lenny.
>> So, I was at the Claude uh Code with Claude event uh I don't know, a month ago at this point and I was and I went to your talk and I I was just like, holy I got to get Fiona on this podcast. She's thinking so far ahead of where everybody else is going and what

where people are at with AI. So, you've been an engineer for 25 years. I was browsing your LinkedIn. Uh you started at IBM of all places. Such a different [laughter]
place to be these days.
And it's just insane how much the job of an engineer has changed over the past just like 2 years. It's like a completely different job. Like, people may forget 100% of code was written by humans not long ago.
And now it's getting to 100% of code written by AI. Uh as Boris uh famously said, code coding is salt.
Along these lines, there's just this tweet that you guys put out yesterday where you showed uh here's the tweet. Anthropic engineers on average ship eight times as much code per quarter as they did compared to 2021 to 2025. We'll show this chart on the screen. It's just like stable stable stable stable boom shooting off into the moon. So, it's just insane how much this role has changed. I'm curious about your kind of path as an engineer going living through this, having been an engineer for a long time, what have been kind of like the big moments along the way where it's shifted your
way of thinking and operating that have led you to what you do now and how you operate now?
>> Oh, I I I love this kind of look back in time. Yeah, like IBM, working on DB2, the operating system services team. Like back then I was thinking, oh, how can I how can I be like the like what's a hard area of the stack? And I really thought the lower level you go closer to the OS, then it's like more hardcore and you you

learn more. So that was a I was really fortunate to gotten into the IBM internship. Um but the funny thing was I would I think there's even a big shift from IBM to Microsoft. So at IBM it was I think Vim. Like I didn't have an IDE that we used. I think there might have been
an Eclipse license, but for some reasons most of us didn't use it. So I remember it it was mainly like Vim and and you know, like kind of terminal debugging. And then when I joined Microsoft, I mean this is how naive I was. I didn't even know about IDEs and such and and back then you didn't really get to pick teams. Like this was early 2000s. Actually first off I should say I was so grateful that I landed the internship and the role cuz to take us all the way back in time the dot com bubble burst in 2000.
And so for my graduating class like a lot of the companies weren't hiring or or were having freezes. So I'm so fortunate when Microsoft extended me an offer.
And so they're like you're going to work on Visual Studio. I did not even know what Visual Studio was cuz I came from like a Unix school. So I remember to asking, oh, this cuz I was thinking, well, the name Visual Studio, I'm like, oh, is this like a a better paint program? And I could tell the look of my manager's face like, what is going on?
>> [laughter]
>> But then it ended up becoming like the love of my life for the first, you know, 11 years of of my career. But that was the first time I used an IDE. So joining the Visual Studio team, seeing, oh, wow,

like here's an IDE with like debuggers and you can set breakpoints and do multi-threaded debugging. Like that was also always mind-blowing for me to think about the the stepwise change. Um so yeah, like that was kind of the story going from I beam to Visual Studio. And then what I really loved actually about Visual Studio is I was on the Visual Studio editor team. So I use a VS editor to build the VS editor and that's where my whole love of dog feeding comes from. Like I I remembered I wanted to first and foremost create a delightful experience not only for myself, but for my teammates. Cuz also if we go back in that time, if you remember, I mean when did Twitter come out? Like was it two 2000 and six or something?
>> it's been around my whole life.
>> [laughter]
>> But back then, like before social media, it was also harder for most engineers to um
hear fast customer feedback. Like for sure there you would be user research sessions or we would have customers visit us.
But you didn't get the rapid feedback that you you you you do nowadays. But back then I was so lucky because I was on VS. We ourselves gave each other so much rapid feedback cuz we were all heavy VS users on the team.
>> This episode is brought to you by our season's presenting sponsor WorkOS. What do OpenAI, Anthropic, Cursor, Vercel, Replit, Sierra, Clay, and hundreds of other winning companies all have in common? They are all powered by WorkOS. If you're building a product for the enterprise, you've felt the pain of

integrating single sign-on, SCIM, RBA, audit logs, and other features required by large companies. WorkOS turns those deal blockers into drop-in APIs with a modern developer platform built specifically for B2B SaaS. Literally every startup that I'm an investor in that starts to expand upmarket [music] ends up working with WorkOS. And that's because they are the best. Whether you are seed stage startup trying to land your first enterprise customer or unicorn expanding globally, WorkOS is the fastest path to becoming enterprise ready and unblocking growth. It's essentially Stripe for enterprise features. Visit workos.com to get started or just hit up their Slack where they have actual engineers waiting to answer your questions. WorkOS allows you to build faster with delightful APIs, comprehensive docs, and a smooth developer experience. Go to workos.com to make your app enterprise-ready today.
>> People think about these milestones along the way of an engineer's journey and we forget there's also there's been like a lot of transformation over the years. Not quite what we're living through, but just like IDEs, Visual Studio. Uh so I love that I love that you're reminding us of these moments in uh the history of software engineering that that have changed the the work in a big way.
>> Yeah, when I worked on Visual Studio back then we also shipped software on CDs.
I mean [laughter] before like and and that's why there were really hard deadlines cuz you had to make sure the software is ready for us to give to

manufacturing to then, you know, put on the CDs for us to then put on the shelves. And uh and then so once that was another shift when we actually started to be able to, you know, ship software online. And I think that's the interesting thing and I kind of mentioned this in my talk. It's before when you like engineering time was like like really precious resource. But you also have these really hard deadlines, for example, like printing [clears throat] software on CDs. And so back then you would do a lot more planning cuz you just wanted to make sure given the time you have, you you make the best of it. And that's the shift that, you know, like we're seeing with with uh Clockwork and Co-work is coding is no longer the bottleneck. And so now like, you know, you showed up you showed the the tweet and that graphic. So but now it's all about like where has that shift happened? Like now not only engineers but we also have designers, PMs, everybody on the Clockwork team checks in code. So like when not only more people checking in code but like kind of different disciplines, but also the throughput is so high. How do we think about verification? Like that's kind of this other shift that I'm seeing.

>> Maybe just kind of set this theme that I want to have for this conversation. A lot of people are just wondering what is what is software engineering, managing software engineers, managing software and product teams look like in the future and you are living through that right now. So, you mentioned this um point about a more focus on

verification, making sure the quality of the code being
uh 8X is actually high and something uh that's that you know that will work. So, just like let me ask this broad question and let's kind of see where this conversation goes. What is a What does an AI-pilled software team look like in 2026?
>> Because the roles are blowing, it's shifting more to this builder
uh like everybody starts being a a a builder, I would say. The other shift that I've recently done is I actually have a cloud code remote session that I enlist in all of our repos. And so, this way I have full visibility into the work that everybody's doing. And this instance it also has access to all our Slack channels and and I'll I'll have access to like how how are the metrics of of everything we track. And so, every month like I when when like I have been like, "Hey, you know what's fun? Like let's let's take a look back." Like and so, we'll actually do it together. I'll share my, you know, like cloud code you know, I've shared my screen and we we we do a cloud code
session. And it's just about, "Hey, what were the focus areas? Like what were some of the the products that got shipped? How did they do? Oh, what were the feedback channels?" And and so, I even though like before I would have just used these sessions to like generate PRs and bug fixes, I actually have these sessions to enable me to have conversations with folks that I support.
>> Say more about that. So, this is this is like a management technique, let's say, to help people

not just ship, but actually
ship better understand if they're
shipping things that have impact. Is
that kind of the idea here? Just like
use cloud to keep on top of all the
things people are shipping and then make
that a conversation with them.
>> Exactly. So, like yeah, outside of just,
you know, the action of shipping is how
did it do in market? Or hey, did we
have, you know, like did we cause some
bugs? And it's okay to like I have this
saying, "Make new mistakes." Like it's
okay to make mistakes, just make new
ones so that we're always learning cuz
if you aim to make zero mistakes, like
that probably means you're not, you
know,
moving fast enough or being a little bit
too cautious. And so yeah, by by having
Claude, like so then it can also look at
like some like for example, actually,
yeah, we were I was just looking at, oh,
you know, given some of the, you know,
incidents that we have, like let's look
across all of this. Can we generate a
theme? Like what's a good area
investment for us not just especially
when we think about like quality. Like
are we seeing any hot spots of where
there could be a gap? I think that used
to be just a much more manual uh
process, I would say. Like if I look
back a year ago, I don't think I would
have been able to, you know, have some
of these insights with Claude.
>> Yeah, well, partly is cuz like there's
that and also people weren't shipping as
much, so you could just make a little
bullet list. Here's the things I shipped
last quarter. This feature, that
feature, this feature, that feature. So

I think what I'm hearing here is this is like one of the only ways to stay on top of all the things that 8 mate is shipping.
So this is a really cool thread of just like how you found ways to stay on top of this 8 X increase in code. What else has worked in helping you and your team stay on top of and maintain quality of all the stuff that y'all are shipping cuz that's obviously a challenge, a bigger challenge.
>> Yeah, and so definitely like the feedback channels are really important to us, but then we also get a lot of feedback and like for example, even I myself usually what my my morning ritual would be, you know, I get my morning cup of coffee and then I look at the feedback channels and then I try to pick up what are, you know, if I have some maker time, what's something that maybe I would be able to help out or what I could like see some gaps. Like that just used to be something I would do every morning.
And uh yeah, I think maybe a month or two ago we we launched routines. And that's also completely changed. Like now I just have a routine that automates all this for me and then there's also it's almost like before I would, you know, be able to kind of like,
you know, generate some prompts, but now with routines, it's almost like I'm,
you know, having an agent help me generate the like generate the prompts and the PR. So for example, one is,
"Hey, keep a look on this feedback channel, you know, what are some of the themes?" And then like when I wake up, then you know, I have a really good

summary of that. And then even some like PRs that I'll be able to take a look and review.
>> And the feedback channel, where's that feedback coming from? Is that like like emails, Twitter's, or a combo of everything?
>> [laughter]
>> Our feedback channels are definitely like we have a lot from internal and but also like emails, channels. Actually, everybody like when we all get feedback on even like when friends ping us or on LinkedIn or socials, we'll all actually like post all of that in Slack as well. And so and and we also have like of course partnerships. And so we have different channels for for all the various sources. But but that's what I mean of I need Claude's help to help me stay on top because there's so much incoming feedback.
>> Got it. Okay, so this is cool. So, this is a
like a way of working that you've built to stay on top of all the stuff shipping which is this kind of daily ritual / routine where you as a manager look at what people are saying about the current state of
Claude Code and Co-work and used to just like okay, someone go fix this, go fix that. Now it's like here's the PR that will fix this thing, check it out. We're ready to ship it if you want.
>> That's right.
>> Okay, like obviously a big challenge for people is also just code review. I imagine Claude is also doing a lot of its own code reviews. There anything there you've
recently figured out that allows your

teams to ship faster stuff that they are confident is great?
>> Yeah, and honestly, it's crazy when you think we didn't even have Claude Code Reviews last year. And so speaking of bottlenecks, that that was a really really big bottleneck of, you know, the human reviewers. So, we definitely for the important like like areas that need deep subject matter expertise, we definitely want to make sure we have the proper you know, like human still reviewing. I would say what helps us though is the more that we can automate to almost check in the framework for what good looks like. Claude is very good when you give it a framework to validate against those frameworks. So, I had mentioned like you know our late you know like recently we just updated the content design to to have a scale in it. Like my like I and this is why like um I think if you have specs or like or like like check those into the repo and then make sure the spec also keeps up to date with the code like frequently. But that's what I found like works really well of any time you have like a statement of what good looks like, get that into the repo and then Claude code review can make sure it's still matching what you set up to do.
>> Basically, it's like a the evolution of test-driven development.
>> Yes, for TDD it's cuz I remember that was like a a big thing my gosh maybe like in the 2000s of write the test first and then and then you can like make sure the test fails then you do actually the code. In principle, it's really good but I think I know I remember that I myself struggling a bit

because it was almost like you have to eat the broccoli first. That's why I'm like ah you have to write this test first and I just get so much thrill out of shipping and building product. So, that's funny. Actually, the first bug I fixed on Claude code I remember asking Claude, "Hey, I want to do test-driven development. Help me write the test first. Make sure it fails and then we'll actually, you know, do the fix and then and then now that test pass." And the fact that that used to be, you know, like that test generation used to just be this tax that I remember having to pay. Like the fact that that's now automated and you can even revisit all these principles that have been around for a while but now they actually might be even more efficient just because you have the models that can do more of the work for you.
>> Yeah, like that's it's so unfair. It just writes the test for you first.
>> [laughter]
>> So, on this point of builders, one of my favorite slides from your talk that um I want to ask you about is who you are hiring and who what you look for in people. And so, I'll read what you said there and I want to hear more here. So, the two profiles that you now look for when you're hiring are creative builders with product sense and deep systems experts for the hard parts.
>> Yeah, the deep subject matter expertise. Like for example, when I first joined Cloud Code, we had really great kind of like product generalists. And then I realized, oh, we were missing folks with systems background. And so, that was definitely an area that we we needed

more folks with kind of like systems and distributed systems expertise.
And so, I would say whatever are the parts that it's all about trust but verify. The models are really good, but there are definitely a lot of areas that still need verification. And so, wherever you need the deep subject matter expertise, I would say that's, you know, an area to definitely still invest in.
And the other one is kind of like the the product the product sense works almost like the dreamers. Like these are folks that usually will be like, "Oh my gosh, I'm really passionate about a product." And they have an idea, they build it, and then it's always like looking at the feedback, and then iterating, and polishing, and making sure that the product is a delightful experience. Like owning that product end-to-end. Uh that's that's like another skill set that served us really well on Cloud Code.
>> This super resonates. There's this word ambition. I don't know if you used it, but that's what I thought of as you were talking. That's been coming up a bunch recently in the podcast and in other work I've been doing. There's this 10X engineer I was talking to the other day, and he was just like, "I used to like I heard about a feature idea, like I have someone's like, 'Hey, we should build this.' And I was like, 'No, that's really hard and complicated.' And he's like, 'But now I'm like, no, no, that's so possible. I just ask Cloud Code to do it.' And it just does it. And it's this whole mindset shift. And he's just realizing now it's about how ambitious

can you be? Like everything is now possible in theory. Now it's about how ambitious and how big can you think versus just like, 'Okay, it's all these stupid little features and things I have to unblock.' Does that Does that resonate? Is that something that you think about?
>> Yeah, so actually I was just catching up with an engineer yesterday, and Um, uh, actually is not a mobile engineer by trade, but we really needed to, uh, update this feature to also have, uh, a mobile footprint. And it was just amazing. Like he's like, "You know what?" Cuz it it it and it's common cuz you might think, "Wait, but I don't I'm not like an Android expert." But now, thanks to Claude, actually I can actually have a partner and actually also do this on the mobile surfaces. And so that definitely resonates. It's like the it's lifted the ceiling of what and anyone is able to do.
>> Let me follow that thread.
As the role has transformed in such a crazy way,
some people are thriving, some people super frustrated, unhappy, fighting, resisting.
What do you see common across the people that are doing really well, the engineers that have adapted and are thriving, versus the engineers and, you know, even outside engineering that are just like frustrated and having a bad time?
>> I would say that a growth mindset really, really helps. Like actually even before, um, AI tooling, I found that has been just so valuable. And I I learned that a lot. Actually it was the shift

from that, um, Microsoft to Meta. That was where I first real That first year I'm like, "Oh, this is what having a growth mindset really means." It's really this concept of always be learning. Be also what served you to get you to this point may not uh, will serve you no longer. But it's really hard because, of course, everybody like we all, you know,
we've all gotten to the state by acting or, you know, like operating in a certain way. And so sometimes it is a little bit scary to think, "Wait, you're ask like but I've been successful so far. You're asking me to change what has made me successful." And so I I would say like the growth mindset is is really, really has has really served me well and I think it's also served others well, um, that that I notice. Like always leaning in with curiosity and always being able to learn. Um,
and then in terms of like the frustration, I think sometimes I also see a little bit of fear.
And so my advice there is at least this is how just in life I
cuz we all have and fear is of course an evolutionary like I I didn't study anthropology, but I think it you know, it makes sense, right? It helps us make sure we were able to survive and not get eaten and you know, by by larger predators.
But for anything that you know, that that there is a fear my advice is kind of lean in and ask, "Okay, is there some what can I do about it? What is within my control?"
Cuz sometimes the frustration comes from fear and feeling like but everything is

outside of my control and so it's
happening to me.
And so if you think about, "Okay, what
is in your control?" Um
it instead of happening to you, is it
happening for you? And then what's
what's something that you might be able
to kind of like do and change? Like I
felt that's been helpful cuz if not it
it it it is super frustrating to have
the fear and then feeling like
everything's outside of your control.
Like I actually remembered um
when I was in actually when I was in
high It's funny when we were going back
to IBM. When I was in high school, I
actually didn't go into computer science
or engineering. Like I really want to be
a visual artist. And this is how far
back we go. Back then computers were
really expensive. I actually didn't even
have access to a computer. My first
computer with access to computer was
grade nine I think in high school. I
don't even know this happens in high
school anymore. My high school had a
typing class. You know, a typing class
just to learn how to type on a keyboard
and learn to be really proficient and
that was my first time and then you
know, the next class was maybe like some
HTML programming. So anyways, it it it
but the reason why I fell in love with
it was while what I love about art is
creating and being able if you have an
idea, you can go ahead and create and
tell story. And then I realized, "Oh,
computers and programming, you know,
that that's that's enables me to do
that." So anyways, I I tried to really
fast try to make up for all the, you
know, science classes for me to get into

university, but I had this fear of, "Oh, but how will I afford to get into an engineering school?"
And it was this big unknown. I grew up in Ontario, so I was very grateful. I knew there was an Ontario, you know, school assistance program, OSAP, I think it's called, which I'm very grateful for, but I didn't know how much of that would cover my tuition or like expenses, and it was just this unknown, and it would be like a year a year out. And I remember thinking, "Okay, what can I what can I do about it?" And then as as luck would have it, the National Bank of Canada just posted this flyer in our high school saying, "Hey, we're hiring high school interns to be a bank teller."
And I remembered, "Oh, that like I And of course it was going to be minimum wage, but I I I I thought that that could be a lifeline." But it was funny cuz the class I hated the most in high school was accounting, so I don't know if I'd be any good at it at all, but I signed up to be a bank teller, and that ended up being such a great decision cuz I worked all summer, saved up, and then then actually I was able to work as a bank teller on the weekends. So, like yeah, well I would went to I would go to school Monday to Friday, and then on Saturday I'd be a bank teller, and that ended up being this lifeline that enabled me to, you know, like pay for all my school expenses. Um and you know, we talked about the the year 2000 dot com crash, then when folks weren't hiring as much interns, actually I continued being a bank teller for for two two years. And And so, but

but like that was the one action I thought I could take that was within my control to try to counter this fear I have of if I'm really going to go down this path, I don't even know if I could have afforded to go to school. So, that's probably my other advice of like growth mindset, and the source of, you know, frustration or anxiety, if it's coming from, you know, the like see is there some Is there one action that's within your control that you can take um to, you know, so it it it like cuz I think there's you you there there was a saying like do something What would you do if you're not afraid actually? Those were my two favorite sayings before like what would you do if you're not afraid and do something scary once in a while cuz that's also usually how we grow. Um I found that when you're when you're really good and professional at what you do, you're kind of at peak, you know, like maximum efficiency, but then how do you keep growing is you then do something scary that you might not have done before and yeah, you will cut have a dip because you need to learn. But that's kind of how you um keep pushing yourself to learn.

>> The quote that I have probably used the most on this podcast of all quotes is the cave you fear contains the treasure you seek.

>> Oh, I love that.

>> Yeah, and it's so true. Just like like I forget who put this, but just the thing that is scariest is like that's a compass towards that's what you should be doing.

>> Mhm. I'm going to I'm going to steal

your quote.
>> Yeah, then please do. Like, you know, there's like don't do all the scary things, like maybe don't jump off a cliff, but maybe in career in career moves it's probably a good a good choice.
So, kind of following this path, something that
I know that is important to you, something you think about that I also think about is this kind of gap that is forming between people that are just like leaning into AI, killing it, doing super well, and then people that are just like not. And this is like
a scary time for people that may be left behind in this uh new world that is emerging. Uh I know you spend a lot of time with small businesses. That's a big passion of yours to help people learn how to use AI in their work. Talk about just like how you think about that and what maybe we should be thinking about to help folks, you know, stay not not fall behind basically.
>> Oh, I I I love this. Yeah, it's one of my passion topics cuz uh I kind of mentioned, you know, like growing up in Canada. Um and I I moved there when I was a kid. Uh I was born in Hong Kong. So, I didn't speak any English and my parents had to work all the time. So, my grandma who is the best grandma that anyone could have ever asked for. I know everybody thinks their grandma's the best, but I really was so lucky. I had the best grandma. She moved with us uh just to take care of me while my parents were working. But neither of us spoke English, but I was able to learn how to speak English by you know going to

school and speaking with classmates. And when I think about my grandma, it was a very alienating for her to be you know in the country new country where and you know back then it wasn't as walkable. Um but one summer I remembered we just happened to find this little yarn shop uh that was owned by a lady that also spoke Cantonese. And so that became every week to this yarn shop this summer and my grandma found her knitting circle and then I think I learned how to do like macrame.
Uh which I think is having a comeback by the way. It's all it's always funny to see what's
>> Macrame coming back?
>> Macrame I think yeah. You heard it here first on Lenny's podcast. I think macrame's coming back.
>> All right.
>> Um but but that was probably where my love came from. I'm like oh wow this little small business created this wonderful sense of community. And I've been really fortunate to you know become friends with all the small businesses that that I love. So that that's kind of like where the passion comes from. And then um what how it happened was I was using Co-work for my own uh business uh expensing travel. I don't know what it is. I really don't like doing business expensing. And when I was using Co-work, it was magic. I'm like all these things that I don't like to do like Co-work is doing for me. And I'm like wait a minute all my friends and honestly small business owners they really bust their butt. Like they work incredibly hard um and they use you know like sometimes might be operating on

really small margins.
And so I thought if it's if Cloud
Co-work is so good at helping me do my
little you know business travel expense,
this would be huge for it cuz I I see my
friends sometimes sitting at the bar
with stacks of bills and all they're
doing is invoicing and expensing. And
and I think nobody actually really likes
to do it. So that's kind of uh where it
came from. And then so yeah I remembered
uh
like helping uh a couple of them
onboard. And and uh it was also very
humbling to to see them go through our
onboarding flow. Actually, it found some
really good bugs. So, [snorts] it was
really like a win-win, but it was also
delightful for me cuz they also use Coda
in in ways that I I wouldn't have
thought about cuz I was all fixated on
look at how it is amazing with PDFs and
invoice. And then, um
you know, one of my friends who who runs
two restaurants, she's like, "Oh my
gosh, this folder I have is like a junk
drawer of like it it's basically our,
you know, documents folder just becomes
this junk drawer of every or downloads
like just a you know, like junk drawer.
And she's like, "I know I have a few
menus in here and I can't find it." And
I'm like, "Well, let's ask Coda." So, we
gave Coda access to directory, found the
menus. And then, she uses it in a really
unique way. She goes like, "I want to
make sure I keep my prices reasonable
for locals and tourists." So, she goes,
"Hey Claude, look across my style of
cuisine in this area is a comparable."
And it came back with really cool almost
like market analysis. And she goes,

"Hey, I actually just went to that restaurant in Seattle and that was pretty good." Um so, I learn something every time. And uh and yeah, they've been like giving me great feedback as well.
>> So, the question then is just like how do we how do we spread this to everybody? Because as you see, you know, there's like like a lot of people are just like, "I don't have time for this." Or "I hate AI." It's just like I want to ignore it. Is it like talking is it just talking about this sharing examples? What do you think? How do it What's like a way to make a dent in this uh in this problem?
>> For all of us as especially to your listeners are probably very AI pilled. If there's anyone either in their community or their family that that has like I would start with what has something that really you felt um has that you really have felt has made a meaningful life change uh for you with with the AI tools. And then, seeing if that is a conversation starter for cuz for me AI is a tool. Again, it's the whole light in the dark. I totally understand the frustrating part as well. But for me, I'm also felt like knowledge is power. Like have to learn how to use the tools because it could actually, you know, be the light part of that light and dark equation. So, I think that that would that's something that I would love everybody's help with of if there's whether it's a community member or even if there's a business you really like and think, "Hey, have you ever It's It's It's a little awkward to start." Like I remember when I first reached out to my

friends, I'm like, cuz I don't talk
about what I do
with them a lot, but I'm like, "Hey, I
kind of you know, work in AI. Can I Can
I It sounds It's It's even unnatural for
me to do cuz I'm you know, like, I'm
like, "Hey, can I show you what you
know, Codoxer is possible?" And and and
then but it end up we end up having a
lot of fun. So, yeah, I would love for
like that conversation starter
would be great of Yeah, cuz I just want
to make sure we keep sharing the the
knowledge and you know, making the tool
equitable cuz if not, I'm
concerned that the divide grows larger
and larger.
>> Me, too.
I find that sharing use cases as you
said is it's so powerful. I was just
using Codoxer the other day to fill out
my my son's camp forms. And just sharing
that on Twitter just like a lot of
people are like, "Oh, wow, I didn't
think about that."
Yeah, it's just like these little things
you don't think about that Codoxer and
Claude can do.
Kind of along those lines, speaking of
Codoxer,
if you think about it, Anthropic has
been super early on uncovering these
really big opportunities ahead of
everyone else. For example, coding. So
far ahead. Like realizing this is a huge
market. Whether it was intentional or
not, uh it was just like, "Whoa, that's
maybe the biggest
new business opportunity in history." Uh
and then Codoxer is a great example just
leaning into knowledge work. Let's just
make Let's just solve all the knowledge

work. Why not?
So far ahead of everyone else.
Another element is it feels like a focus
on the personality of the model,
something you all were very early on
just how important that is. Not just for
the experience, but just also the
intelligence and success of the model.
What is it that you think Anthropic and
the teams do
differently that allows you to uncover
these opportunities and then just go big
on them before other labs, let's say.
>> Well, I haven't worked in any other
labs, so I'm not sure how other labs
operate, but I I will share like yeah,
with um
and like actually on the Claude Code
team as well and co-work like we also
keep an eye on like latent demand. Now,
we're very fortunate with coding as a
use case because so many ants that we
were, you know, like our our own first
customers and we're able to do like
really rapid feedback and and so I think
um but latent demand has has been like a
like for example, like co-work we
noticed hey, a lot of folks that were
not necessarily coders were were using
Claude Code. Can we make that experience
better? Um I think that's actually even
outside of Anthropic that served me well
in in all the different products I've
worked on. Um but uh and and actually
it's funny you mentioned like, you know,
the the Claude's for uh you know, my
passion with small business after I, you
know, had had a few of these visits, we
ended up now launching Claude for small
business.
Which is really cool cuz I and and it
totally didn't come from me, so I'm not

taking credit, but I noticed it too of
like when I was working with and they
because they were asking me, oh, does it
have this plugin right this plugin? I'm
like, I think it does and then we have
to, you know, go and search for it. So,
actually now Claude with small business
bundles it all up, so inside co-work you
just have this little toggle. And uh
there were there were there was a
wonderful team that's been kind of like
uh
you know, doing uh co-work sessions with
small business that probably found hey,
we could have like an efficiency gain
here. Um
or like make the experience better.
And so I would I would say like always
uh not only for products that you're on
like making sure you're always
you know, listening to feedback and
always iterating, trying to make a
delightful, reliable, high-quality
experience, but then also keeping an eye
out for oh, what are these other use
cases that that are popping up and can
we also make that experience better? Um
it's it's interesting like after like,
you know, in software I've learned that
customers will use your product in ways
that you did not intend for good or for
bad. And so the the best way is really
like it's all about the iteration learn
and keep close to the feedback.
>> Wait in demand. That's a that term's
come up a bunch on this podcast.
>> It's on there.
>> Might be something there.
>> So essentially it's just watching
closely for
behavior that you may not have expected
or that is just kind of emerging.

And then just going big on that
essentially, exploring it, building
something.
>> Yeah.
And having a hypothesis for hey, like
because actually when you see people
jumping through hoops to make something
work,
can you actually make that an even like
a smoother and and better experience?
>> Yeah.
Coming back to the way that your teams
operate,
you guys are so at the edge of what's
possible.
And the role of just engineering has
changed so much. I'm curious what you
think is the next kind of frontier of
how engineering in particular is going
to change. Is it like I don't know,
fleets of agents? Is it something else?
Just like what's like the next big shift
to how engineers operate that either
you're already working on or
implementing or just like it's starting
to happen?
>> I would say we're shifting more towards
async, like asynchronous. And so to your
point, the the fleets of agent, like
that that's why routines is so
interesting cuz almost like I used to be
like you know, doing a prompt and
synchronously. And then I would like
maybe like kick off different prompts
async asynchronously. But now I can
actually have a routine that actually
generates, you know, these these prompts
for me. So it's almost like the level of
abstraction keeps pulling up a little
bit. And so I I would yeah, I
wish I had a crystal ball to see what
this is going to look like. It'll be fun

actually for you and me to revisit a year from now.
>> No, I I want to hear more about this. This is So help us understand what is what is routines and then what is what do you say asynchronous? You're writing a prompt and it just kind of goes off and is it writing it immediately? Talk about what this actually looks like.
>> Uh yeah, so routines is like how you can like so if you remember I was talking about my what I used to do is always like you know, wake up with my morning cup of coffee and hey, look at through this slack channel for me. But then now like I have a routine that I set to run every morning at a certain time to say hey.
>> Mhm, that's right. But then it's able to actually go and kick off agents on your behalf. And so that's cuz like that's what like you know, even you know, before it would be oh you you could like yeah, do a cron job to automate like but now it's hey, look at these feedback and then if you're seeing some of these bugs that what are some polish fixes that you might be able to to knock out and then it would like go kick off and then I wake up and I end up having PR's that I could review versus before which is still more of a kind of different agents that and then it's still I'm still thinking about okay, now what do I do with this information? So it's like that higher level abstraction of okay, now how can I actually write a routine that also basically does prompts for me for spawning different agents. So I think we'll we'll move more towards that kind of like asynchronous style of working.

>> So interesting. So like in a sense as a manager, you have these kind of things you do daily and what you're saying here is you can set up these prompts that kind of check in every day on the things you would have be be doing like how are the projects going? What's falling behind? What should I unblock? Who is who's struggling?
What's how do I fix some improve some some polish? And so the idea here what you're describing here is kind of write these things that you almost do as a manager daily and have Claude essentially do them for you and then show you here's what I'm doing here's what I here's what there's to review.
>> And then it's it's even like then giving even more autonomy at some point of it. You know what like go for it. Like for example, if the verification is really good, go go [laughter] for it.
>> see. So you give it like a little bit of freedom to do more of this.
Yeah. That is so interesting. It reminds me speaking on this idea of go for it. I don't like I wasn't planning to talk about this, but I just watched Tyler Cowen had this awesome talk about just like what's happening. I don't know if you know Tyler Cowen is really smart dude economist has a podcast.
And he just had like there's so much talk of
the people that are doing best in this world,
his term is initiative. They have initiative. Another term people use agency.
Is that something just like you know people hear this a lot just like the people that seem to be doing best are

taking the most initiative, getting the most proactive, have the most agency. Does that Does that kind of spark any thoughts to you just what people may need to I don't know think about or improve on?
>> Actually that's agent it's that word agency. That's what we really hold important on the cloud code and code work team. But it's interesting I say along with high cuz it's like we really it's about like hey here's a problem and then it's really everybody on the team has ideas for how to address the problem. So it's really high agency and then we say with high agency is also high accountability. So it's all about like making sure folks have that freedom to cook. You know they and but then it's also like okay what's the accountability for it as well and that's where and again it goes back to like what's the hypothesis of what you're you know like trying to solve. So I think yeah the the balance like or you know almost like two sides of the same kind of agency and then accountability I think has served our team really well.
>> Such an important element. Have a agency. Okay cool. Everyone's off doing stuff but okay what have you actually [laughter]
what have you busted up?
What have you done?
So kind of along those lines
it feels like there's this vibe shift recently from token maxing just go crazy spend as much as possible just see what's possible to like wait what are we actually getting out of this? How much is it costing? What's the ROI? Interestingly Boris was actually like

head of product inch productivity at
meta that was like his job is measuring
inch productivity and uh
and then also just this like tweet of
lines per code like code ship like
there's this like interesting discussion
of just how do you actually measure
productivity increase and ROI
of
AI tools in spend. How do you like you
know you guys have the infinite
advantage of getting free free tokens
working at Anthropic.
With that in mind,
what what have you learned about just
how to measure, say, ROI of engineers in
today's world?
>> This end productivity is such a
fascinating topic. Um yeah, cuz I
remember, you know, like like even for
us mentioned, like you know, we'll first
start with like lines of code. And then
that's throughput and then I remembered
once then there was a debate of, well,
lines of code, this engineer had this
crazy lines of code, but they just took
some library and then just was porting
it. So, they checked it in and then I
was like, well, maybe it's significant
lines of code. And then it's, well, but
what if we're, you know, updating our
frameworks and now we're generating less
code, but like the output's still the
same. So, now I'm like, okay, maybe it's
it's like time to land PR. Like but but
it's very interesting of it's always
been whichever metric, it's it's um like
if if you're really focused on the out
like my advice here is
one is output like is the output really
going towards the outcome? Cuz yeah,
like the the the the token maxing, it's

kind of like a
it's almost like the lines of code that
we used to have, but I'm really much
more about a um what is it that we're
trying to to do? Like all all of this,
you know, there there was another saying
I really like, don't forsake motion for
progress. Um because if you're measuring
like, you know, like tool user usage,
then you're you're measuring the action,
but is it really making whatever the end
outcome of yours like important?
Um and so I I really uh try to zoom out
and focus on what is the problem we're
trying to solve? What's a good way to
measure that? And then that's what we we
focus mostly on versus kind of like the
the the productivity measurement. Um
I would say though, like outside of
metrics, especially if on teams as
you're having more people adopt AI
tooling, I would say definitely go um on
a listening tour for for the for the
leaders listening to a podcast,
especially, you know, um I I love to
focus on the senior engineers as well.
Like hear from them on what's working,
what's not, how can we actually make it
better cuz they will also help you
multiply and scale that out across a
full engineering team. And sometimes
there's those conversations where you
might get sparked an idea that comes up
and also really good shared learning
versus um, you know, like metric
dashboards. It's very interesting. I've
learned some good lessons from metrics.
I'm all about like it's really great
when you have a a metric that you can
actually hill climb on, but always keep
in mind that is you know, my whole
growth mindset of is it still serving

you? Always keep in mind like is that metric really uh, still serving the outcome that you were aiming for.
Um, like a fun example I have with this is in the early
Facebook Marketplace days, we were kind of like launching by region and we really want to make sure we're building a delightful product before we expand. And I remember in the early days what one thing we would keep an eye on is kind of like number of sellers. And I remembered after launching our first region, I'm like how in this area the number of sellers is low, but actually people are like people are finding items that they're looking for which is what we're aiming for like helping people find items that that they need. And then I realized in that region it wasn't uh, large number of sellers, but there were power sellers.
But our first gate before we expand would have just been like you know, factoring heavily number of sellers and I remember that quick conversation of hey, and this goes back to that whole people will use things in ways you may not expect and ship better rate learn. And so so then we updated the the metric to go oh, you know, like it's not uh, number of seller cuz it didn't factor in power sellers. And so that's the advice I have to like whatever metric whether for productivity or even for product, always keep an eye and make sure that you're not just having blinders on that's blindly following a metric that used to make sense cuz sometimes the landscape can change so fast. Uh, even the metrics themselves might need to be adjusted.

>> And this comes back to your
uh, process that you shared of having
Claude watch all the PRs shipping per
engineer, let's say, and then not
focusing on the metrics but focusing on
that leading to a conversation about
what impact this have, what was maybe
some bugs that happened, and that being
a really powerful way of understanding
how what's going on with this engineer.
>> Awesome.
>> Let me come back to this just this
question of speed and quality and just
like
impact. Uh is there anything else you've
learned about just how to
balance those things? Just like there's
a crazy velocity of code that's being
generated and just stay on top of
quality and impact. Is there anything
else there that might be helpful to
other teams that are trying to wrangle
all this all these PRs being shipped
every day?
>> I would say, and this is what honestly
we we were we want to keep doing more of
and and being better at it, too. Like
the proactive quality, so especially for
quality, making sure that what are the
experiences that are key and making sure
you actually is you actually speak of
metric. Those are really good things
that you you make sure you can look kind
of like see trends over time.
Um and so like on the quality front, we
found like um
and this is like the more proactive we
can be of like making sure we can get an
earlier detection into uh quality. And
and so like um
that's been one thing that we've been
paying a lot of attention to. Like so

like you know I I I started this, "Hey, let's have a concept of what's bad versus what's sad." And bad is like a very bad irrecoverable error. And sad is something that's kind of like a pain point recoverable, but it's interesting when you stack up sads, it could, you know, generally go to bad. But even having a like starting with a high-level framework like that and because of not like I think sometimes with dashboards you can have you know, like uh time to load or all these other, but when you're dealing with a lot of different product surfaces, it's harder to go, "Wait, is that a good number or not a good number?" And so one thing that's helped us is versus just raw, you know, like performance or uh uh, you know, like reliability numbers, also having some framework of what we think is, you know, like a a bad experience and making sure we're focused on addressing those and then also keeping an eye on where we seeing in terms of the sad. That That's helped us a little bit.
>> I like the bad and sad. So, these are kind of like thresholds of this is bad. Okay, this is serious. And this is in terms of like performance or or or failures or what what what sort of things are you measuring here?
>> Uh, so for example, like we we allow each team like speak of agency. So, knowing that bad is like a really bad irrecoverable error. We enable each team of further surface areas or it could be services that, you know, they lead. What What is like so for example, on CLI could be crash rates. Like a crash is pretty bad. You lost work.
Um, and for example, a sad might be,

"Hey, is it flickering?" Like it might be recoverable. But we have each team like and and that's why it's been interesting because before each because surface areas are different, like we would have all these dashboards of but it's it's harder to zoom out and go, "Okay, what's the overall theme of the experience?" So, to your point, we give high agency to each team what we think constitutes a bad and what's a sad and then what's the goal that um, each team wants to take.
>> One of the like the main takeaway I'm hearing here is uh, one of the best tools for staying on top of quality is just uh, monitoring and tests versus spending more time reviewing, which makes sense because the speed is just impossible to stay on top of and it almost speaks to this idea of closing the loop for agents to be able to kind of figure things out themselves. They know what success looks like. Cool, we'll fix it ourselves. So, that's a really interesting takeaway is just invest more in the tests.
Evals I imagine is part of this and then just like monitoring failures and speed and things like that.
>> That's right.
>> Something I think this is public. I I know that you guys have this dashboard that tracks just like F-words, like how often people are like
>> [laughter]
>> cuz they're so pissed and frustrated. There's like a funny term for it, I think. I forget what it's called.
>> Yes. Actually, I remember yeah, that was last September cuz we were all we were all seeing some frustrations and uh,

yeah, that was an engineer on the team of hey, we should maybe track swear words. I'm like, oh that's a great idea. I remember it. I had just, you know, joined and we were having that really fun conversation. Um yeah, so it's it's again like it it's very interesting to kind of like look at um and and that's why like Evals is hard too because it's going back to like that user experience and how we can make sure it's a delightful experience and less frustrating. But yeah, the the swear word dashboard is is a fun one. [laughter]

>> This episode is brought to you by Mercury.

>> Radically different banking loved by over 300,000 entrepreneurs and now with command. I've been a customer of Mercury's for over 6 years. I have never once thought about leaving. Mercury is basically what happens when banking is built by product people, not by bankers. [music] They make it so easy, dare I say fun, to send invoices, move money around, set up virtual cards for folks on my team. Does your bank have an API, a terminal native CLI, or an AI ready MCP server? I don't think so. And just recently they launched command, a conversational interface built directly into Mercury, which acts as your financial operator. I've been using command to transfer money around, to figure out what categories I've been spending the most money in, analyze my cash flows, and [music] just today I used it to find out how much I've made from a specific sponsor over the past year. I just asked, [music] "How much

have I made from X over the past year?"
10 seconds later I have an answer.
>> [music]
>> It is so freaking cool. Visit
mercury.com to learn more and apply
online in minutes. Mercury is a fintech
company, not an FDIC insured bank.
Banking services provided through Choice
Financial Group and Column N A, members
FDIC.
>> Kind of going back to the way you
operate and ways that you've figured out
to work in this crazy new world that
we're in.
Um I've heard about a couple things that
you've implemented that are pretty
unique, I think, to uh how teams operate
and I think is something that has worked
really well for you. One is making every
manager start as an IC and then just
every manager has to continue being an
IC part-time kind of this player coach
approach. Talk about that, why that's so
important in today's world.
>> Mhm, I love it. Um, it's funny when I
first joined cuz I I and I have amazing
recruiting partners and I I noticed we
were you know this whole theme of um
kind of growth mindset. It's just
because the landscape is changing so
fast. Like
what worked well before like may not
make sense and even what makes sense
today may need to change tomorrow,
right? Like that's what I have to keep
reminding myself. So yeah, when I first
joined recruiters like, "Okay, yeah, we
have these um couple like manager
postings." And I'm like, "You know, cuz
actually how it came from like a
listening tour I did with like all the
all the um members on the team. Of and I

heard a lot of these, "Hey, I really appreciate the agency, but how can I make sure prioritization?" And And so I kind of through that I I um I realized and then there was some really good feedback too of making sure that um it's not too many layers of reviews. Like there was good feedback. Some folks have might have drawn from like companies are like, "You know, and and then I'm like, "Hmm." When
I actually think as a as a leader, if you actually start as IC first without a like worry of supporting people cuz that's a very heavy responsibility that, you know, I think like managers, but like that but before you have to take on that full responsibility, give yourself that maker time to actually type deep into the code and learn the code base and I I
or or the product, like whatever it is. It doesn't have to like honestly the the PRs I do are like but it's more about for me it's important because it keeps me in the flow because we're making so many changes to co-work and code. So even me doing PRs it's less about what it is I'm fixing, it's more about me using the the product every day just to keep that touch cuz as amazing as metrics and everything are and I do look at those dashboards, if you as a leader if if you're not, you know, like living and breathing your your product every day you you sometimes kind of like lose touch of uh touch and feel of the product.
But anyways, on the manager front, I think giving time for managers that join to be able to go deep into do that before supporting people. And then they

actually like end up building really great rapport with the team. Like because if not, I I think sometimes as managers, you know, you might come and join a new team and you instantly think, "I have to manage. Let me dig into my manager toolbox and and do manager-y things." But if you actually give yourself time to not have to worry about that first and actually learn what it's like to be an engineer and teammate on the team. That also goes really far um to to building rapport. And and in and in terms of like using the product, I think that's that's it's it's interesting. Every team I've joined, one of the first actually across all the different products. It could be VR, it could be smart glasses, it was like Instagram. Usually when I first joined, one comment that comes back to me is, "Hey, you know what? I really love how you're actually using what we build. It's It's refreshing to see, you know, you giving user feedback." Like Like and and so I think as leaders too, it's also a way for us to experience that the work uh personally that the team does.

>> Something that people may not realize, uh you were overseeing a an org of like 500 people at Meta before you moved to Anthropic, right?

And you moved to IC IC engineer basically at Anthropic from that.

>> I started out like it was for a very short short amount of time. But but um actually this was my journey between Microsoft and Meta. So at Meta, I interviewed as a manager, but I think at least for the first quarter, I was also an IC um because I really wanted to

learn what it's like to be a Meta engineer because before then I was at Microsoft kind of like cut my teeth on engineering at Microsoft. So I knew all the code bases, the tools, the languages. And uh yeah, but it was so valuable to have the the those first months like actually learn what it's like to ship uh as a Meta engineer. Plus it was also fun. Like I think sometimes we forget if we like And but that's by the way what I love about Claude code. Because um the last time I shipped production software at Metal is probably 2017.

And every year I might start like I do a lot of dog fooding and a lot of, you know, like I I also use dog fooding to help me like that the quality of of product as well.

Um, but it's been a very long time since I shipped production software. It's because part of it is I don't I don't want to screw something up. Like I'm always I'm always so scared of what if I do something and then I cause a bug and then I'm not verifying properly or am I wasting someone's time? Because, you know, like also the the the tool flows would change. But I remember that first week on on Claude, I'm like at first I almost again went did my usual, let me go meet all the engineers and treat them to coffee and then I'm like, oh wait, wait, wait, let me ask Claude. Claude was this really good onboarding buddy of mine cuz I was like curious about the code base, asking it questions and and then it also really,

you know, helped me like, you know, do the automated test, but I also want to still do some manual testing. And I I

asked Claude, hey, help me come up with like what's a way for me to manually test this to make sure I cover all the case. But all that then gives me confidence that okay, I can ship peers again. And then I got like kind of more comfortable with it again. And and so I've actually had a lot of friends reach out to me that have been managers for a while that's like, hey, I'm shipping code again thanks to Claude. And so but in in general it's it's yeah, like it's it's just important to me as leaders to make sure you're kind of like using the product that your team builds.
>> It's so interesting that you are overseeing the team as an angel leader that is most changing the role of an engineer. It's such a meta role. Like the work of an engineer is transforming because of the software that you're building.
One question along these lines is just do you worry about
engineers skills to code atrophying from not actually writing code anymore? And does it even matter? Is this like something you think about? Is it a big deal?
>> It's It's interesting cuz we actually have this uh discussion on the team a bit. Like there Like are there things that we miss? Or like actually to your point, I'm always like thinking about like when someone's onboarding and ramping. Like you have Boris who hand wrote the code in the early days and of course not does anymore, but he gained that knowledge from before cuz he was in the code base. So for especially all the engineers that join, I'm also like make sure you're like taking the time to to

all the work that you do still get the understanding of the architecture or the change because it goes back to that trust but verify. Maybe one day it won't matter anymore, but at the at the pace that we're going, I actually still think understand like it's always double clicking on that layer that you depend on. Like may maybe that to me is a metal like cuz
um it's always take that time to learn about your dependencies cuz this way when your dependencies change, like you're kind of like more aware or you might not be taking advantage of changes to you know, dependencies. So I think it's always about kind of like uh doing that double click. But the other thing that we found interesting on the Cloud Code team is after a while we felt uh it could start being a lonely experience cuz we all started just working with our agent so much. So recently we started a maybe a like a pairwise programming lunch. And cuz what we also learned was on Cloud Code, everybody uses a Cloud Code co-work. Everybody uses a flow so differently. And so we found that wow, when we do pairwise programming, we actually learn so much from each other. Um and but and then the other thing too is we we make sure that like we also uh have them make your time together too. So like for example, hackathons is is another thing we really like to do just to make sure we're kind of like interacting together as a team.
>> That is such a good point. Just like the loneliness that emerges because used to be inch teams building code together. Someone's doing back end, someone's

doing front end, someone's doing iOS app. Probably a bunch of people, you know, on the back end. And I was like, you know, well if 10 clouds running in parallel doing all these things.
That is such a good idea of just like finding ways to connect engineers. So, the idea here is like almost pair programming, but not. It's like parallel It's like parallel play when kids are growing up. Like, you're kind of working next to each other, but building your own things. But, even just watching how other people build is your finding is really valuable.
>> Yeah, and it's because our our own tool chain is changing so fast as well. But, yeah, it's very interesting to me that like everybody on on our team just uses Claude code and and code work in different ways. And so, time I watch someone work, then I I learn something myself as well.
>> How do you manage people getting over like obsessed with the optimizing their workflows? Is there anything you just say, "Okay, it's fine. Just
Just keep [laughter] going."
>> You know, I I haven't seen too many folks on our team like it's I think it's everybody is just really excited about, you know, either a like an architecture update that we think will be higher reliability or some product experience. So, most folks like that's what we we talked about a lot. Like, yeah, we we um we don't over It's fun lunchtime conversations, but I don't think we over optimize because there's no cuz there's no perfect answer actually.
>> much to do.
>> [laughter]

>> Is there anything These are really interesting insights. Just like as the role transformed, things come, things go.
So, I'm curious what else is kind of lost in this new world of software engineering. I used to be an engineer. I don't know if you know this. I was an engineer for 10 years.
Uh and it was like so fun just to sit there in flow coding. You know this feeling. Just like, "Oh, wow, it's working. Look at it compile. It's amazing. I'm making progress." And now it's like you don't do that anymore. You just sit there and kind of wait for agents to build the thing. What else What else is kind of lost in this new world for engineers?
>> Yeah, it's so funny. I just had that conversation with the engineer on the team of talk about flow. Like, remember the old days you have this really gnarly problem and you pop that music soundtrack in and then you're just in the in the zone. Um
Uh so, yeah, there is a little and then there was always that big aha moment at the end. I would remember. Like is it Yeah, you remember you're like you finally cracked it. You know that yay. So we we do
but it's like I think now we get a lot of joy from like the product but I do think there is cuz I hear from other engineers as well of it Well, I
some of the hardest parts is what I used to enjoy.
Like yeah, I heard that from another engineer as well. And so I think we're all just kind of like shifting um Uh,
but I I I do see like the the thing most

and that's why I was I I was talking about like the pair wise programming in the hackathon. That did recently come up more of folks were starting to feel like it's starting to be a lonely experience.
>> So interesting.
Within Anthropic I'm curious so engineering like the most transformed I think of any role right now. What other What's like the second most transformed role so far would you say within Anthropic that's most different from how it was like a couple years ago let's say?
>> It's it's all the coding adjacent roles are shifting. Like PM is I know like you you were chatting with Kai. I think PM is also transformed quite a bit because PM bar no longer bottlenecked if if they have an idea waiting on engineering bandwidth. Um so that's kind of like the the next role I've seen shift. Like actually our PMs have also helped us roll up sleeves and help shift you know like some features when we when we were um you know like when when there wasn't uh when there was a you know something we wanted to do and an engineer wasn't able to. So I think um it's all the coding adjacent roles are starting to shift. Um but I think that's where again the verification is important cuz when you have more different disciplines checking in how do we make sure everybody has high confidence. I also think we need to do more to keep automating these other portions of the workflows. Like for sure we focus a lot on coding but next when you think about like design or data science like those are starting to be the next areas I think are good opportunities for us to

see how we can yeah like um
uh start and improving kind of like
experiences there as well.
>> Yeah, I have a speaking of data science,
I have a data science friend and he was
just saying how data science is so
different now
where
now most of their job is people doing
their own like not amazing data science
work using AI and it and then just like
showing it to the data scientist, "Here,
here's the analysis I did. Can you just
make sure it's right?" And half the time
it's not right.
And so the job is just like a whole
different job now instead of doing the
work that they thought they wanted to do
and that's like, "All right, I'm just
reviewing all this AI data science all
the time. What the hell?"
>> [laughter]
>> Kind of along the lines, coming back to
just how your team operates and how
things are changing,
let me see if something comes with this
question. What should an engineering
manager expect from their team now
versus like a couple years ago? What's
like a normal, I don't know, baseline
expectation of how things should work?
Is it other than just, you know, we're
shipping faster? Is there anything else?
>> Oh, interesting. Well, definitely I
think most commits are cloud assisted
and so that was a shift.
Um
I think
you know, I kind of like mentioned we
have like Slack channels with all the
feedback and also our dashboards that we
have

get to cloud. I think having engineers building like keep building that stronger product sense muscle is I I think also like another and I think that in general helps kind of build these really strong minded product engineers. I would say more of these roles that were traditionally non-engineering, you do now see engineers being like and sometimes you're just blocked by, you know, waiting for you know, a cross-functional partner. I think there's there's less of those blocks now just because the models are able to augment additional capabilities that you may not have as an from as an engineer.

>> So it's kind of interesting that it's both directions. Engineers becoming more product minded and responsible for the quality and success of a product and then everybody becoming an engineer more and more.

>> Yeah, that's right. It's all blurring.

>> Yeah. I forget who said this, but there's this like like what is a role anymore in this this guy said that it's like what's the average of what you do? What's like the highest percentage of what you and that's kind of your role now, whatever it is going to be. [laughter]

>> Yeah.

>> Oh man, I want to come back to your point about obsession with the the product living breathing the product dog fooding. I talked to a bunch of people that work with you about you and that's the thing that came up most just like your

obsession with living and breathing the product using the product constantly, whatever it is you're building uh this idea of dog fooding.
Talk about just like why that is so important, why that's something you instill within your teams and and and reports.
>> Oh, I love it. Um yeah, I think it's it's really and and this has worked for me um it's just been a really good way for me to keep a pulse on you know, anytime you build product there's there's a dream. Like you're really hoping to enable an experience or make an experience better. So I think um being able like like that helps me keep really close to the pulse. I also think uh maybe and maybe something like for sure on Visual Studio that was like, you know, where I where I got this love of it um
but it's interesting cuz even Marketplace, I remembered every time once in a while I'll do like even after I left the team actually. Um one time I had like a MacBook Air and I wanted to sell my old and you know, I haven't sold anything on Marketplace let let me and I could not believe it the minute I I put it up for sale a seller or buyer tried to scam me and it was an interesting like new scam vector I didn't detect and so but that goes to again like people will use uh your products in ways that that you may not expect and so especially as leaders or even like anyone on the team, we all have different like life like actually it's it's funny when it when I was a you know, supporting the VR and AR teams, somehow how I used VR, the setup, I

would always be able to find these really weird floor height issues. So, that ended up being a I took a you know, like I'm going to help us you know, like cuz somehow I got a good repro environment. Um
And so, I think it's number one like making that's how you keep your pulse on the product that you're building and don't get too lost in metrics and dashboards only or presentations.
>> I think that's such an important point you're saying there that I just want to make sure people hear is just like like there's always this idea of anecdotal evidence and just like examples versus the data. And what you're saying here is as a product leader as an engine leader, there you found a lot of success in like the anecdotes, the specific uh little one-offs that you experience as a as a as a user versus like obsessing with the data only.
>> Yeah. And and actually, sometimes it's also how I've been able to most effectively help the team. So, for example, the last team I was on I I was a you know, leading a VR team and you know, because like back then the like I I it it would have been I would I I was not I have not checked in any code into that code base just cuz I was really worried about like messing up the operating system. But what was a gap? We were doing a lot of polish fixes and I was I wanted to I'm like, "Hey, you know what? I'll use my dogfooding time to actually vet how the experience looks." And and so, that was also probably a way that I felt I could still meaningfully contribute to help hold the quality bar for the team. And and then like I say,

like every team member usually always really appreciates cuz I I think um as a lead like you know, leader you're supporting folks on team that outside of metrics and everything, everybody really wants to make sure their work matters and yeah, just making sure that you know, leaders use your product. I think folks feel the the leaders um remain like really engaged and not too I'm
>> And this connects a lot with your point that engineers need to become more product minded. That engineers need to become more PM-y, PMs need to become more But this is like as an engineer, this is one way to do it. Use the product constantly. And that'll help you understand what is missing in the product as a user cuz you're just using it.
>> That's right. And And I would say if you're leading a team where it's really hard for you to use the product, then meet with customers. Like whatever the other avenue is that kind of um like I think that's also been really important to you. Every time I've done customer visits, I always learn something new. Like actually I remember that was uh Facebook Marketplace we're trying to launch to uh Latin America. And so we had you know, we were testing in Chile. And I remember that it it just wasn't doing as well as the other regions we'd done. And everything else we've we've tried. And then I remember I'm like we And we had a really small research trip where I went to Chile. I remember getting a whole bunch of Android phones for us. It was very small crew, like just three of us. And upon landing and upon me opening up these Android phones

and I'm like, ah
you know, the LTE connection was real was much slower than what we were used to in the US. And so I'm like, oh, the Marketplace feed didn't even load very well on these little LTE situations.
What a What a growth blocker you know, you can't even load. But again, that's why it's so important to always uh you know, like um listen to customer feedback and and get that fast feedback loop.
>> I think it was Jeff Bezos that said if you have the data and you have an anecdote, trust the anecdote over the data.
Surprisingly. And that's a great example. Okay. Uh just a couple more questions.
One is in your talk, you had this really interesting slide at the end of what questions you're kind of rethinking about right now that you haven't figured out how to solve
with how much is changing. Uh I'm going to read the three. I'm curious just like if it's still these three, if there's anything else like current problems we need to figure out that we haven't solved in how we operate. What you shared in your talk was, do we still need separate iOS and Android orgs? How far do you push fully auto- automated reviews? And which role with role blurring, how do you ensure everyone's equally productive?
Still problems? Is there anything else that you're thinking about like, okay, we need to crack this. We haven't figured this out.
>> You know, the iOS Android one, like I think we're still actually it's funny

cuz I speak of like the deep expertise. We definitely feel that it's really important like it's really important to still bring on folks with those expertise, but we probably don't need like as large because people are flexing. So it's again making sure we have
you know, like the Android and iOS experts, but then less of a the larger kind of like a mobile org one per. And so but again, that's still a balance that we're trying to figure out of do we have the right and enough expertise? Um The the second one was Sorry, what was the second [laughter] one?
>> Oh, yeah. It is how far do you push fully automated reviews?
>> Ah. Yes. Actually, well, this is a a fun one like, you know, with the content design check. I think we're we're looking for across all of it like how do you actually get what good looks like? Um and so I have a like actually the verification is still one that we I think that's kind of like the second and the third that I think we there's more a lot more opportunities there for us. Um but the how far to push, I think looking at where we think that experts still matters and then also again have to keep asking ourselves, okay, is there a way to leverage our expertise to also automate? Like I think um
like looking across the whole end-to-end experience, like making sure we're not missing in like I think we we come at it from like an engineering standpoint, but making sure um experiences how we think about those other areas. Um I I think there's definitely still more that we can do there.

>> Basically, how do you solve how do you set up a verification that the experience is what you want it to be?
>> Exactly. That and I think that one is still a hard one to crack cuz you kind of like mentioned the eval cases. It's Some of it for sure is accuracy, but it's also the experience. Um so that's something that's uh yeah, we're we're still kind of thinking through.
>> Awesome. Is there anything else that's like uh this has changed recently we got to figure out how to rethink the way we operate or is this kind of the the big ones?
>> You know, I think because with routines and everything being more async, I I I think there is starting to be a high load on our context switching. Cuz I even remember I myself was like, oh I I kicked like I kicked off like and and so I think that's probably another thing we have to think about of how do we uh you know, whether for our team members or our users like how can we actually make the experience better to reduce that load cuz I I do see the context switching load increasing.
>> I can like if you have 20 agents running there's just endless checking in and reviewing and you have to remember what you're doing there. It's like such an interesting world where like the idea flow we talked about before like engineers and most people like there's less of the just hours of flow, but now the agents can kind of just remind you. Here's where you're at like the reset to switch almost is easier cuz you don't have to like relearn everything. You don't have to like re-understand the code base and the

the architecture. You can kind of just like, okay, but here's what we're trying to do. It's kind of like both got better, got worse.
>> Well, it's interesting cuz I used to do book out like focus time for like you know, the focus time for coding cuz you want that dedicated and then the whole context switching and then it's interesting now that because I can context switch more with more async agents, I'm noticing I do actually have to go back and block like a focus time for me to catch up on all the you know, different async work that I've kicked off.
>> Yeah, do you have any thoughts on a solution there cuz that's hard. Just like there's like people just want to do more and more and just how do you do that without constantly context switching? It's really hard and annoying.
>> Yes.
I agree. Like definitely I haven't cracked it yet.
>> [laughter]
>> So interesting. Okay, there's a question I I of as we were talking that I I'm like so excited to hear your answer on that. I wasn't even planning to ask this, but it's so important these days, which is just like and jobs and hiring. So interesting that you would think AI would uh make engineers less necessary. On the other hand, it feels like you guys are hiring engineers like crazy. Open eyes hiring engineers like crazy. Just like there's so much demand for engineers.
Where do you think this goes? What do you think about just the future of the

end role? And this is a big question,
but just
>> [laughter]
>> thoughts.
>> I I'll I'll share I'll actually I'll
I'll I'll share some vulnerability here.
One, um speaking of like big open
questions, I really do think how we
grow the the next generation just
because how you and me got to our
engineering path is just so different.
It's almost like, okay, now when you
graduate from school, it's like how do
you kind of fast-forward it? But but the
important thing to me is that still
understand kind of like, you know, that
double-click I talked about to to the
the layer of an eight. Um but that that
is
a big question I have, and I wish I have
the answers, but I I wonder if it's um
for software engineering, it's almost
like you go more towards a fellowship or
apprenticeship program.
I know it's hard cuz technically we have
like in internships that would but those
were the kind of like 3 months and a
little projects, but I I do wonder if um
and I I wish I have a crystal ball here,
but it's it's yeah, like how do you
almost like cram in some of the
And and maybe it won't matter, but you
know, some of the life experiences that
we all got, how do you actually enable
um
us to kind of like teach that to, you
know, the next generation of builders?
>> Right, like if you don't have to ever
look at code,
well, what's the incentive for a new
software engineer to truly understand
how

infrastructure works and and memory allocation all these things that are like kind of foundational.
>> And and it's interesting. Maybe the the models will get good enough that it it doesn't matter, but I do
>> Yeah.
>> You know, but I do think there's something about that that double click cuz I think that's where there might be an opportunity to improve the product or the system. But, figuring out how to learn that not necessarily by, you know, like year years of uh
>> of typing code.
>> Yeah.
>> Yeah, like somebody's got to have to understand code at some level. Like, it's like some COBOL engineer they have to like pull out of retirement one day like, "Do you remember how to write Python?"
>> You know what you was really fun? One of my
you know, previous managers. I mean, he started software engineer when it was punch cards.
And it's so fabulous. Like, he's been messaging me everything he's been building with cloud code. And I'm like, "Wow, what a career that you go from like he's really like he you know, he's really kind of like seen this whole change. So, maybe I think for us to think about it is like when I think about his career, how he got started with punch cards was also totally totally different than
than now. And so, it it might be that um I I think it'll be interesting for us to see what remains important. And then what changes in terms of importance. And

then it's like how do you um have gained
the Like, I have a theory maybe maybe
that's the the what is important will
shift over time. And then how do you
kind of uh
kind of build the proficiency into it.
>> Oh, yeah, just learn the things that
really matter. Like, the argument that I
I constantly hear is just like it's a
new level of abstraction just like
assembly and binary like it just keeps
going up and up. And now, okay, we don't
need to actually look at the code. It's
like a new layer of abstraction prompts
and
and uh Claude's thinking uh you know,
messages.
>> Yeah, and it's maybe like, "Okay, what
is a interesting problem? What's the
prioritization for experience to build?"
Like, it's it's interesting. Maybe we're
kind of Yeah, and then when you build
things, how do you know it's actually
resonating? And and and uh it's kind of
like doing what you intended.
>> And is good, yeah. Like, is this going
to are we just building a bunch of slop
here or is this actually
architecture [laughter]
that will work? I think the advantage
though for young people is they are so
it's so much easier to just to lean in
and work in this new way versus being
stuck in the way that things used to be.
It's like rare It's like amazing how
many long-time engineers like yourself
have adapted and embraced this. It's
like so hard to just change everything.
Okay, let's do it. Everything [laughter]
>> Well, cuz the rate of change is also so
fast. Actually, that that was the one
thing of

I remember the first time I It was probably Sonnet 3.5 or or 3.6 like and I I remembered it was still like making some mistakes when I was doing things on the side. I'm like, "What are you doing?" And and and then what I noticed was some of the engineers that were resisting AI tooling. They're like, "Ah, but see, it's you know, like look at all of these." But then I think it was hard to understand of how kind of like exponential rate of improvements. And so always And maybe that's another interesting thing that I myself am learning too. Like there might have been something I tried to automate that Claude wasn't quite good enough. And then actually in the next model, oh well, now it is good enough. So it's always also thinking about what may have not worked. Like it might be worth a time to revisit cuz you know, that now might be a new capability.
>> Yeah, that comes up a lot in this podcast. Just build something that is almost working that is at the edge because once the model gets there, you'll be so far ahead of everybody else.
Okay, final question. You may have already answered this with what you just said, but what keeps you up at at night?
>> You know,
the thing that keeps me up at night is probably um
is how we So, you know, we talked about kind of like Claude code and co-work team culture. That's and the team culture is really important to me. Like it's the one team mentality and I I you know, I I share with folks. And and by the way, culture is like a living

breathing thing. It's not just a poster
you slap on a wall and it changes over
time and it's it shows up in how we
treat each other, how we how we're there
for each other. And so, the culture of
the team is important to me because we
are we are growing and and
and since the culture shifts like making
sure that maintaining the things are
important that we are we still really
like it's really important for me to
have like diverse perspectives so then
we can have you know like good healthy
open honest debates on in the open and
we kind of like welcome those and kind
of what I call that one team mentality
that when you get close to the finish
line look behind you and see is there
some of our team to help cuz we kind of
finish as a team.
That's probably the thing that keeps me
up at night and it's
it's like you know there's so many other
hard problems right but I think maybe a
lot of the other ones are product or
engineering challenges that yes we have
you know like dashboards or theories or
hypotheses like but the culture is like
a human aspect that is
um
Like like I I think that's the one that
I I always want to make sure that we're
as we grow we're still kind of like
maintain that culture cuz it is kind of
like the fiber
of the team and it like when it starts
drifting it it's it I'm always worried
of you know like if it if it drifts are
we catching it and having conversations
as a team together to to to make sure
we're kind of all wanting the culture to
grow in the right direction.

>> Yeah, I imagine everybody is struggling with this considering the pace of change and the pace of hiring just like especially a company like Anthropic that's in this crazy like the most unprecedented growth trajectory in history. I could see how that could be some a challenge with so much change so it makes sense like even you know at Airbnb when I was there like that was quite a growth trajectory and that was nothing like what you guys are going through. And that was a constant topic of conversation how do we maintain the culture?
>> Actually I'm curious what was your experience at Airbnb to maintain the culture as you were growing?
>> things one is just the what worked well is the founders being obsessed with it like every every all hands every every big meeting is just like reminding of like the culture and the value of the culture and what the values are just the founders top-down being obsessed with it was a really big part of it. It just like couldn't have a meeting without that coming up as a as a thing.
The other is a memory I always come back to is we had Sheryl Sandberg's became meta come to come do a fireside chat and somebody asked her just how do you maintain culture as you scale? Because we're just going so fast and it's so hard to mean deal with all this change and culture and all these new people and her advice was this is actually the what the problem you want to have because this means you're growing and doing well and this is

normal versus you can
nothing will change if you're doing
badly like that's that's the that's so
much worse situation when you're not
growing and you're not hiring like
crazy. That's a much worse situation
that will cause even more even
suffering. So this is what this is a
good problem you're you're you're
dealing with. is is her advice which has
always stuck with me.
>> Oh, that's great. Like it's interesting
like here you talk I think one of the
important things to kind of cloud code
and co-work team is
um whether uh ICs or managers but this
is a thing I especially ask for managers
on the team is really important that we
all talk about for sure what's going out
but also just be open about what's not
going well.
Because then if we can actually have a
conversation what's not going well
that's how we can actually go ahead and
address it. Like my my my speaking of
what keeps me up at night my nightmare
is especially if someone's in a manager
position and and I'm like hey how are
things going? Everything's fine. I'm
like oh my gosh I'm not doing [laughter]
fine. I know this things are and like
like it's that that whole like you know
how there was this meme of the doctor
cat cup of coffee in a room that's on
fire. This is like that that is my my
nightmare. So that's actually
a discussion I have with a lot of folks
on the team especially managers of when
they first joined up hey let's always
have these open conversations so that we
can solve problems together.
>> I imagine there's something a lot of

people struggle with seeing so many people around them doing super well, at least seemingly doing well. Everything's going great. I'm growing this this awesome company or like just like everything it's hard to like hard to actually be honest and say it's not going great. I'm struggling here. I'm falling behind cuz everyone around you is just like on the surface feeling like they're killing it.
>> Mhm.
>> Yeah.
Before we get to our very exciting lightning round, is there anything else that you either want to leave listeners with, anything else that we didn't cover, anything else that's important that you wanted to share?
>> Maybe one thing is like just a suggestion cuz you know we talked about how how can Cloud do like automate like one other thing that's really big on Cloud Code and Core team culture is explicit permission to kill processes that no longer serve us. And so maybe a suggestion is for you know any anyone you know like on our work and our team or leading teams
like take your like what's one process that you either dread doing
or is really highly noisy or is like really expensive in terms of like just a lot of like it it might
or is something that's just very manual. Like pick one thing and first ask is it still having its purpose? Like for example
one like even our planning like Actually that was my own big first learning when I first joined Cloud Code. I'm like hey maybe we should you know do

a six-month road map doc and and but we're going to do it super lightweight cuz I don't want to waste a lot of time planning and I I felt we did a really lightweight process. But that was such a good learning for me cuz the exercise was good to kickstart conversations and ensure we're aligned but like three months into it I'm like wait have we still referenced
because so much has changed and that was also something I like and so that was also something I changed to that I myself brought in thinking hey maybe this will this will help. So always be open to learning and always ask yourself whatever process you have is it still serving its purpose? Just because the field is changing so fast.
>> I love that advice. I got to follow up on this real quick. So, what is it? How do you think about planning now? Do you do any planning? Is it just like a month-long road map? What's kind of like the simple way to explain where you're at with that?
>> Yeah, I
I call it JIT planning now, like just-in-time planning. So, it it is like around like cuz yeah, I think 6 months was too long. So, now for sure some projects will take more than a month, but we try to do like a a month of planning, like really lightweight. Actually, there's not even docs. It's really just us aligning on a on a little spreadsheet of what we think is important. But even that one, I'm I'm kind of thinking through, "Hey, like every week we should probably still keep a like what we're trying is here are the the month's priorities."

And we're going to try it out, but I have a feeling like every week we'll probably want to do really quick like, "Hey, just to check. Yep, this is the still this month's priorities. Good." Um but yeah, like but now we we've shrunk it to JIT monthly planning.
>> So, it's monthly meaning for the next month, here's a little uh sheet/Excel spreadsheet of what we're planning to do for the next month. And then every week check in, is this still what we're planning to do for the next month?
>> Yeah, it's it's yeah, like like very very we But even that one, I'm also still feeling, "How can we even automate this more?" Cuz I don't I never want it to be feeling like a tax when someone has to, you know, up update the spreadsheet. So, this is actually yeah, like just yesterday we're chatting, "Hey, how can we actually uh automate this?" Well, it's getting to that question, always ask yourselves, "Can we actually automate this better?"
>> Like my PM brain is like, that's so like how could you not do something like that?
>> [laughter]
>> Okay, here's what we're thinking for the next month. Let's just check it once a week. Let's make sure this is like it's hard to imagine that not happening. Um and you don't have a lot of items on the spreadsheet is what I'm hearing also.
>> Yeah, but like we really try focus. So, like we will will share out like here's what we think are the highest priorities. And again, for that agency of given the priorities, then like everybody's like their kind of like item

for how they think uh addresses those priorities.
>> Is there anything that's like here's for the next like 6 months bigger bets kind of stuff or is it just like let's just think 1 month ahead?
>> I like so we'll usually definitely there's themes of where we think the the work and so we'll we'll definitely like um and actually the the whole team will bring everyone together
uh like every every 6 months. So there we'll usually kick off like some themes but then it's really the
making sure we keep the pulse of what because again like even though themes change you know so fast when uh the landscape changes.
>> All right, let's pull up the spreadsheet and let's take a look.
>> [laughter]
>> Oh man. Man, this this whole podcast could have been just talking about this planning stuff they do. Okay, I'm going to have to find someone else to talk about this cuz this is so interesting just how y'all plan. Uh yeah, okay. Well, Fiona, with that we reached our very exciting lightning round. I've got five questions for you. Are you ready?
>> Ready.
>> Okay, first question. What are two or three books that you find yourself recommending most to other people?
>> Ooh, I will say for fiction
uh actually two authors I recommend to everyone, Margaret Atwood and uh Haruki Murakami.
Um those are just two like like I mean I grew up in Canada so so I I read a lot a lot of Atwood growing up but her books are fascinating cuz it's almost like

speculative fiction of
you can kind of squint and say okay
could this actually you know like happen
to to a society. So so I love her take
on speculative fiction and Murakami I
love his um magical realism style.
It's it's um but then the one book so
there are authors but the one book I
always recommend to everyone to read at
least once a year or you know um The
Little Prince.
I like I I think I'm I'm sure we
probably all read it at at some point in
our lives but I think it's a I I read it
at least once a year. It just you know,
to to remind me to to think about like
kind of like what's truly important.
>> Wow. That doesn't come up a bunch. Okay.
I love it. Uh favorite recent movie or
TV show you have really enjoyed?
>> I haven't watched TV [laughter] shows.
>> That's very common across Anthropic
people I have on the podcast.
Common theme.
>> But I will share with you what I always
have downloaded on my phone so that if
I'm on on the airplane um
So there's three movies I I always have
on my phone cuz I I think they're just
so so fun to to to watch if I have time.
One is Amelie.
It's this uh French movie that oh my
gosh, all of these movies are going to
be very old by the way. It's going to be
like vintage movies. But I I I loved
Amelie. Super whimsical. So really
highly recommend it to um anyone that
haven't seen it. It's uh you know, if I
if you remember I told you I was you
know, going on a I thought I was going
to be a visual artist. So when I was 16,
my high school we took a a trip a high

school trip to Paris and that that just
I I've so many memories of that and
Amelie really captures the magic I felt
at Paris. And the other two are are
Ghibli films. Uh I love Spirited Away.
That's um it's just such a
like just such a
I just love that story. I I I love the
yeah, like I I just love every
everything about Spirited Away. It's
probably one of my favorite Ghibli
films. And then the third was another
Ghibli film um Nausicaa Valley of the
Wind. And uh I I think about this one
quite a bit because if anyone asked me,
"Hey, how did you kind of think about
you know, all these leadership traits?"
I think what I watched that movie
probably when I was eight or nine.
And the the heroine Nausicaa and how she
seeing how she leads just left such a
like um just left such a footprint in my
heart I guess you could say that
probably Nausicaa has inspired me to a
lot and a lot of my different leadership
principles.
>> Wow, what what is that book called
again?
>> Uh the the movie's called like Nausicaä
of the Valley of the Wind, but it was
actually based on a on a manga.
>> Valley of the Wind. So, it's like High
Output Management Andy Grove, Valley of
the Wind Nausicaä.
>> [laughter]
>> That's right.
>> Two top management books. So cool. Okay.
Uh do you have a favorite product you
recently discovered that you really
love? Could be an app, could be
clothing, could be a gadget, could be
kitchen equipment. Uh

>> I'll I'll share the product that I was reminded of how much has made a difference in my life recently. Uh cuz I've been just traveling a little bit and so I don't I and I like to travel really light. So, I I use whatever shampoo and conditioner, you know, that the hotel gives. And I forgot that um so the the the product that I I actually have one of their hand I promise this is not an infomercial. But
Sweet Sisters Bodycare. It's, you know,
a
a local business on on Whidbey Island. Um but the reason why their product has made such a big difference in my life uh it's a full line of organic hair, body, skin care. But a few years ago I started getting this rash on my nose right here that was really painful, like actually bleeding. And I could not for the life of me figure out how to stop it. Like I tried not using any lotions. Like I cut everything out on my face and it was still really hurting and then somebody says, "What's the shampoo you're using?" I'm like, "It's the same shampoo I've used since I was, you know, like a teenager." And they're like, "Maybe your body has now um and it's, you know, like generic, you know, brown shampoo that you get anywhere." And they said, "Maybe your body's just started developing an allergy to to it because of the chemicals uh in it that I like it."
>> Wow.
>> I'm like, "What?" And so anyways, this was an organic shampoo that I found. Lo and behold, I use their shampoo and then cuz I you don't think that when you wash your hair, it actually then, you know, like goes goes over to the to the rest

of your body. But so since then I've switched everything that I used to be Sweet Sisters. Um but I'm recently reminded of how important this is cuz after a week of uh yeah, hotel shampoo, I I actually started having some skin reactions again. I'm like, "Ah, maybe I should get like travel-size bottles that I can bring with me."
>> This is an awesome pick. I love local local business picks, even big bonus points for that. Uh and by the way, this travel you're doing just for folks that may not know this, there's a there's these Code with Claude events that are happening all over the world. I went to the one in SF. There was one in London. You're going to one in Tokyo. Is that the end of it or is there more after that?
>> Uh Tokyo is So, yeah, that's next week and that's the last leg of of the trip.
>> And then probably probably more in the future. Uh
>> [laughter]
>> so cool. I love that that's happening. Okay, two more questions. Do you have a favorite life motto that you find yourself coming back to often in work or in life?
>> Ooh. Um so, in work, I really love to remind folks keep it simple. Like, what is the the thing that you're really trying to do well and and focus on that? Do you know, like keep it simple cuz I think sometimes we could overthink. Uh so, it's always a good mantra to think about that. Uh and then in life, um you know, probably in a world where you can be anything, be kind.
>> Yeah. I love that one. I We were at a at a Montessori school uh tour and

that's what the teacher had up on the wall.
>> Aw. Like, it's you know, we have we have so many things going on that you never know what's going on in someone else's life and that one small act of of kindness can make the biggest difference. Like, um
>> Yeah.
>> For for me, actually, that was what Remember Do you remember COVID? We were all like working from home during, you know, the COVID days.
>> Um I it it just always really stuck struck with me cuz I was in these, you know, back-to-back meetings and I and I always one-on-ones are really important to me because, you know, that I actually that time is usually also really important for the other person. It's something that, you know, they've been looking to. They might have things they discuss. So, I always try it, I always really prioritize one-on-ones. Uh but during that time I my grandmother wasn't doing well and she was in a home in Canada and because of COVID I couldn't go visit and actually even my aunt uh you know, aunt and mom couldn't go visit. And it was this very rare of if if they have a helper that can help we could do FaceTime. But you never know what time that is going to be because you know, there's so many people to take care and all of a sudden I got a message from my aunt going, "Grandma can FaceTime at like 12:00 p.m. today." I'm like, "Oh no, there's this one-on-one that I I've been meaning to have and I I just messaged uh you know, my report to say, "I'm really really sorry. It's so last minute. Is it okay to and I know it

now it's when I say it it doesn't seem
like it's a big thing cuz it's like but
to me it's always really important to
keep but he was like, "Yeah, totally. No
problem at all." And for me that was
just a small act of kindness. He doesn't
and he totally didn't make it into a big
deal but that made the biggest
difference to me that I got to you know,
say hi to my grandmother on FaceTime.
>> I love that.
>> [laughter]
>> Okay. Uh final question. Uh so Boris
Cherny, when I asked him about you, when
he had this interesting insight where he
said, "In very important meetings you
can often hear Fiona you can hear the
click-clack of Fiona knitting
in the background."
>> [laughter]
>> Uh
maybe talk about just what's going on
there and what's [laughter] what are a
couple things you knitted recently?
Oh my gosh. Well, I I knitted this top
recently.
>> You make your own clothing. Unreal. This
is [laughter] very meta for Cloud Code
building itself.
>> You know, we're always building our
clothing.
>> [laughter]
>> Yeah, actually actually this is whole
fun thing I I think between knitting and
programming cuz it's kind of like two
stitches, knit and a purl, so it's zero
and one. And anyway, so many concepts of
stocks and queues you actually could
Like I'm kind of like a compiler. I'm
kind of like you know, generating a an
executable when I knit. Um but I don't
[laughter] know about that. Yeah,

actually it was my my grandma that
taught me to to knit when I was eight. I
kind of mentioned her and I um going to
that yarn shop and so every time I I I
knit I think of her. Um but I I always
like to multitask as you can see. Like
yeah, I kick off multiple agents and so
anytime I'm sitting and because um I
practice enough and got proficient I
have to I don't have to look at at what
I'm doing. So it's almost almost like
how you know sometimes people have like
fidget spinners and such. Uh knitting is
just so anytime I'm sitting still I'm
like, oh this is time to you know like
generate more knitting. It's cute. Well,
cuz I got so much yarn that if I don't
do this yeah, my my yarn
I'm I might have a slight yarn addiction
problem.
>> I love this. When I when I asked Boris
what he would do when AGI hits when we
don't have to work he said he's going to
make me so I'm guessing your answer
would be just knit and make beautiful
clothing. [laughter]
>> Oh my gosh, my dream is to actually open
up a a yarn store in my grandmother's
name and and create that community.
>> And then co-work would help out you
automate everything.
>> That's right. Especially invoicing.
>> [laughter]
>> Oh my god. Fiona, you're awesome.
It's just incredible the work that you
and your team are doing just changing
the world in such profound ways
and there's no it's clear why it's
growing so fast.
So good job. Good job over there.
Thank you for making time for this.
>> Well, I'm just really honestly lucky and

humble that I get to work with such an amazing team. Like I know how lucky and I am I am and I'm so grateful. But thanks a lot for having me on the podcast as well. This was a lot of fun.
>> Uh to follow up on questions, where can folks find you online if you are online if they want to follow up on anything and how can listeners be useful to you?
>> Ooh, uh so definitely I'm on LinkedIn and I'm sure most people have shared this already but would love feedback of what's going well, what's not going well. Um also any latent demand that you are all using that might be interesting use cases. You know, I had a friend message me recently to go, oh I'm using Claude to help me generate uh a building plan for my shed and he actually showed me his his shed, which was cool. So would would love to hear about that. And then yeah, maybe also, you know, we talked about reaching out across over there's someone whether a small business that you love or or someone that you feel like hasn't you know, as some of the listeners are super AI pill. Yeah, maybe take a time to hold somebody's hand to show what's AI might be able to help them with.
>> Such a good one. That is such a good answer to this question. Especially coming from you, Fiona. This was awesome. Thank you so much for being here.
>> Thanks so much for having me, Lenny.
>> Bye everyone.
Thank you so much for listening. If you found this valuable, you can subscribe to the show on Apple Podcasts, Spotify, or your favorite podcast app.
Also, please consider giving us a rating

or leaving a review as [music] that really helps other listeners find the podcast. You can find all past episodes or learn more about the show at lennyspodcast.com. [music]
See you in the next episode.