SARAH HANSEN: I'm Sarah Hansen, and today I'm talking with Ana Bell. She's a super smart programmer, author, and senior lecturer at MIT. Her computer science videos on our channel have captivated millions of learners around the world. Today we talk about Python, growth mindset, and rubber ducks.

I have to know what the ducks are now. I can't continue.

And I ask her, with the emergence of AI, do we even still need to learn how to code?

What's the value proposition of learning a language like Python? Do we still need to do it?

ANA BELL: So I'm not sure if this exactly answers the question...

SARAH HANSEN: Oh, tell me more.

Find out her answer today on *Chalk Radio.*

Hi, Ana.

ANA BELL: Hi. How are you?

SARAH HANSEN: I'm good. You're here. I'm so excited.

ANA BELL: Yes, me too.

SARAH HANSEN: This is one of our pilot episodes for something new that we're trying, which is a video version of our *Chalk Radio* podcast. In many ways, it's an experiment. How do you feel about experiments? Are you an adventurous person?

ANA BELL: No, not at all.

SARAH HANSEN: Oh, great. We're going to be great.

ANA BELL: Yeah. I like stability, and I like boring days.

SARAH HANSEN: Oh. Well, I'm not sure we're going to have one of those with these ducks.

ANA BELL: That's right.

SARAH HANSEN: So I actually don't know what the ducks mean.

ANA BELL: Yeah.

SARAH HANSEN: So I think you're going to tell me a little bit later.

ANA BELL: As we, yeah.

SARAH HANSEN: I think we're in for an adventure, and a little bit maybe of a choose your own adventure.

ANA BELL: Mm-hmm.

SARAH HANSEN: So, you're at MIT. You've been here for about 10 years. And you spend most of your time thinking about and teaching people who are new to programming, introducing them. And I'm wondering what that means to you, why you're interested in teaching, in particular, people like me. What do you get out of that? And tell me about that experience.

ANA BELL: So I recognize that-- well, I don't want everyone to become a computer scientist. But I do think that everyone should learn programming as a skill. So programming should become like what math is now, where you learn it in elementary school. And you obviously use it here and there in your adult life, but I don't know when you've last used calculus, right? Like, we don't all need to become mathematicians after we've learned basic math. Similarly, I do think people should learn basic programming, but they don't have to become computer scientists.

So the reason that I enjoy teaching students intro programming and exposing them to intro computer science is because it was really hard for me to understand programming when I first started, and I just always remember how tough it was. And I teach at MIT, and there's obviously a lot of really smart people here. And I think as you teach more and more, it's easy to forget how hard it was to start learning.

And so as an intro course, you might make the problems you give students harder and harder each year just because they get maybe boring for you. But it's not boring for the students, and I try to recognize that every semester. I just remember it was really hard for me. They've never seen this before. So it's really important to make sure that they have a really good experience.

SARAH HANSEN: You mentioned that when you first started programming, it was difficult. I'm wondering, what made it difficult? And then, what are you doing in your teaching to make it less difficult?

ANA BELL: I honestly don't know what made it difficult. The beginning was easy, just kind of understanding the basics of just writing. The basics of creating an algorithm is basically three things. So the first is a sequence of steps. So step one, step two, step three. Some sort of way to control the flow of those steps. So usually you go step by step.

But at some point you say, well, if this is true, do this. Otherwise, do this. So a branching kind of idea. And then there's also a way to control the flow by asking the program to repeat certain lines. So it's like, well, if this is true, go back and repeat this a million times, which is very easy for a computer to do.

So a sequence of steps, a way to control the flow of those steps, and then a way to stop. Those are the three things that make an algorithm. That's how you write a program.

And grasping that was not hard. It was just once you've gotten over that learning curve, it's learning some of the more advanced concepts that were harder, some of the details for how that specific language works. So I started with Java, not Python, which was harder than-- it's a lot harder than Python. There's a bunch of details that you need to think about there.

SARAH HANSEN: So I had the opportunity to watch some of your lecture videos.

ANA BELL: Great.

SARAH HANSEN: Yeah. And I noticed that you make space and time for students to practice, like right in the class itself.

ANA BELL: Definitely, yes.

SARAH HANSEN: Outside the class as well, but in the actual space where you're teaching. And I'm wondering if you could talk about that a little bit, why you do that.

ANA BELL: Yes. So I think I mentioned this before. I see programming like math is now, but this will be many years from now. It's a skill that people need to get that they don't have yet. And so if we think about the way we learn math in school, it's a lot, a lot of practice. You just always constantly practice adding, subtracting, multiplying, right, the kids. Yeah. So I think it's very important for students to actually get the time to do that practice.

The way I run the class is a flipped classroom or a blended learning style, where students watch the basics through video first, and then in the actual classroom, we do active learning. So I go over maybe something that's a little bit more confusing, and then I give them time to break, with the hope that me forcing them to have them actually start coding, they'll be able to see, oh, I totally understood what she was talking about in the video, and I can totally code that.

But then when they actually when I'm forcing them to actually do it, starting with a blank slate, it's pretty daunting. And so I want them to get over that bump where it's like you have a blank piece of, blank editor or whatever, and then you just have to start writing something down. The more you do it, the more you feel comfortable with it. So there's all these breaks for all these activities that I try to keep the class lively, of course, and kind of indirectly have them practice these skills. Like, just write something. Just write something down.

SARAH HANSEN: Yeah. And also, that does two things, I think. It de-emphasizes perfection. You're like, this is practice.

ANA BELL: Yep.

SARAH HANSEN: I think like an artist, right? So just get a sketch, write a few lines, just get something down.

ANA BELL: Yes. Yes.

SARAH HANSEN: And then we can improve it.

ANA BELL: Yep.

SARAH HANSEN: And then that speaks to something else you've mentioned before, and that's growth mindset, and that programming is not like a gift and you have it or you don't, but it's really something that can be developed.

ANA BELL: Yeah. A lot of-- you see some people who are really good at programming. They didn't get good just because they had that skill, they were born with it, right? There's a lot of growth mindset involved in that. And anybody can reach that level of being good at something, programming included, just by lots of practice, trying different things, challenging themselves. So not just sticking to things that are obviously way too easy, because then you'll never grow.

But you kind of, like, I don't know if this, the state of flow. It's like this chart where you want to be in this middle band, but kind of oscillating between doing things that are too easy and then challenging yourself and doing things that are a little bit harder. And you don't want to always do two things too hard, because then that's discouraging and you'll stop. But you don't want to do things that are always too easy because then you'll never grow.

SARAH HANSEN: I wonder if we could write an algorithm for staying in the flow. Like it tracks, oh, this is too easy for you.

ANA BELL: Yeah. So a lot of the education platforms are trying to give students, based on their responses-- Even in math questions, they'd be like, oh, this student totally got all these. Let me give them a harder one. And it's like, oh, that was too hard. Let me--

SARAH HANSEN: Yeah. How did you find your way to programming? Is this something that you've always wanted to do? Or did it develop along the way?

ANA BELL: So my parents emigrated from Romania, and we immigrated to Canada. And my dad was an electrical engineer, and then he went into software engineering. And being an immigrant, immigrant parents from Europe, we were highly encouraged, my sister and I were highly encouraged to get some sort of engineering job. So it's just what is done.

And so computer engineering or computer science, something along those lines seemed to be the best option given where things were going in the future. This was the '90s, so just the beginning of the internet, the beginning of all that, that boom. So my dad recognized that was what the future would be. And he said, you should probably learn to program and potentially go, probably go into that sort of career. And I wasn't super interested in anything else, so I just said, sure.

And so he taught us Java when I was about 12. I have a picture here.

SARAH HANSEN: Yeah, let's see it.

ANA BELL: And it's in this really cool frame.

SARAH HANSEN: Oh, I love that frame.

ANA BELL: OK. Very fitting. And so this is very '90s.

SARAH HANSEN: I love everything about it.

ANA BELL: This exudes '90s.

SARAH HANSEN: Yes.

ANA BELL: Peak '90s right there. So we've got the room with the computer. Right? As is in the '90s, you all had the room with the computer.

SARAH HANSEN: Yes. Yes.

ANA BELL: And then the big chair, and the side ponytail, or whatever. That's me.

SARAH HANSEN: The scrunchies.

ANA BELL: On the left there, that's me. Yep Yep. And that's my sister on the chair, sitting on the chair.

SARAH HANSEN: Yeah.

ANA BELL: And that's when we first, or maybe before, very close to the time that we had first started programming. My dad just said, here's some really basic things. And then I only got to the branching part of the control flow. I didn't really get to the repeating part of the control flow. So I made this choose your own adventure program where it was like, I forget what it was, like, get out, exit a castle, or whatever it was. And it was just like a bunch of if-else if-else if-else if-something like that. It looked really ugly. But I had a lot of fun doing it.

SARAH HANSEN: But in a way, you're helping people choose their own adventure today through programming, like gaining independence, critical thinking, doing things that they rely on other people to do, they can do themselves. That's kind of incredible that you do that.

ANA BELL: It's pretty fun. That's what I get out of it. Like I said, I don't want people to become computer scientists. I just want to empower them to, I don't know, like if they want to make their own app, they can totally do that using really basic computer science skills or programming skills. Not necessarily computer science skills. But.

SARAH HANSEN: Yeah.

ANA BELL: Yeah.

SARAH HANSEN: So controversial question. I asked you before the show a question, and I have not seen the answer yet, but the question was, in an environment when there are coding tools out there that allow people to build things, what's the value proposition of learning a language like Python to do that? Do we still need to do it? And what's the value proposition of doing it? So you're going to reveal your answer.

ANA BELL: OK. I'll reveal my answer. So not sure if this exactly answers the question, but I said, trust but verify.

SARAH HANSEN: Oh, tell me more.

ANA BELL: OK. Yeah, I've certainly, definitely used AI tools before. And even for really simple things, it can get it wrong, especially mathematical things, and even programming things. And so I have, I would say a basic knowledge of programming. And--

SARAH HANSEN: I, wait, I have to interrupt you. I would say it's a little more than basic for everyone out there. More than basic. Continue.

ANA BELL: And so I was able to start with what GenAI gave me as a program, and tell you know what, in this particular test case, it doesn't really work that well. And I didn't have to go back and ask it to fix it, because that requires a lot more back and forth with the prompting. And I was able to just fix it on my own. It wasn't that hard of a fix. And as soon as you go back and prompt the GenAI again and again and again, you start to introduce variability of English language.

Like when you're talking to someone online, it's really hard to tell what they mean or, right, there's all these double meanings. And GenAI is all probabilistic. So they're basically going to pick the highest probability of what you mean. And then when they give you answers back, again, they give it to you with whatever the highest probability for the next word to be, that's what they spit back at you. So as soon as you introduce that prompting back and forth, it can introduce errors that potentially stack on top of errors and misunderstandings, and things like that.

So having a really basic knowledge of programming could allow you to maybe start with a piece of code that the GenAI gives you back. So you're not starting with a blank slate. It's not super daunting to write it yourself. But you start with something, and then you potentially test it, and then debug it, and then figure it out. OK, now I trust that it works well. Yeah.

SARAH HANSEN: I feel like the ability to verify is the human component.

ANA BELL: Yes.

SARAH HANSEN: And that is what creates the balance with AI. Like if you let AI make all the decisions, then you've lost your self-reliance, and you can't choose your own adventure because you're going to get a bunch of errors and it's going to lead you down the wrong path. So you need the knowledge to be able to choose a path that makes sense.

ANA BELL: Yeah.

SARAH HANSEN: I have to know what the ducks are now. I can't continue. So.

ANA BELL: OK.

SARAH HANSEN: Oh, and the pig.

ANA BELL: Yeah.

SARAH HANSEN: Is that noisy data?

ANA BELL: It is not noisy data.

SARAH HANSEN: OK.

ANA BELL: So every semester, when we run our classes on intro to programming and computer science, there is one particular class that is on debugging.

SARAH HANSEN: OK.

ANA BELL: Which is the process of, OK, you wrote a piece of code.

SARAH HANSEN: Right.

ANA BELL: And it's not running as you expect it to. So how do you fix it? That is the debugging process.

And so a lot of times a bunch of the errors might be things that you can easily fix. But most of the time, your program runs fine and it gives you an answer, but the answer is not what you thought it should be. That's kind of where programming is kind of different than English, for example, because in English, if you write a sentence, somebody else can interpret it however they'd like. Like one example we give to students is, the chicken is ready to eat. What do you think about when I say, the chicken is ready to eat?

SARAH HANSEN: Well, I'm a vegetarian. So I don't want to actually eat the chicken, but I would, because I'm a vegetarian, I assume the chicken wants to eat his food.

ANA BELL: OK. There you go. But I would assume-- I enjoy meat, so.

[LAUGHTER]

So my take on that is I have a chicken on the table and I'm going to eat it. Right? So there's two interpretations to that.

But when we write code, there's only one way to interpret that piece of code that is written. So that's where things go wrong in our code because we're like, well, it should work because in my mind, when I kind of thought about it, it's supposed to give me the right answer. But it doesn't. And so obviously, the program is the one who's wrong because the computer is merely following the algorithm, the set of instructions, and there's only one way to follow those instructions. That's basically what that whole lecture, that whole class is about.

And then I bring out this big box of ducks and I say, and I say, well, programming is actually kind of creative because the solution to a problem, I mean, you could brute force a bunch of things, but oftentimes the solution to a debugging problem is just to take time away from it, and then potentially explain exactly what you think the code is doing to someone who knows absolutely nothing about programming. It could be you.

SARAH HANSEN: Yes, I'm available.

ANA BELL: Or it could be a rubber duck.

SARAH HANSEN: Oh.

ANA BELL: So this is called rubber ducky debugging because it's an inanimate object that doesn't know anything about programming, and it forces you to scrutinize every single line of code that you wrote, even ones you think are correct. Most often, it's the ones that you think are correct that kind of are the issue. It's kind of like, where did I put my glasses? And you spend a long time looking for them, and they're on top of your head. It's like that exact feeling. It's like, where is that error coming from? And it's going to be something super simple. So the rubber duck is what helps you figure that problem out, because you're explaining it in very grave detail to something that doesn't know. And if ducks are not your thing, there are frogs.

SARAH HANSEN: Oh, and the pigs.

ANA BELL: And pigs.

SARAH HANSEN: Of course. Of course.

ANA BELL: Or friends, if you want to explain it to them.

SARAH HANSEN: But friends can be judgy.

ANA BELL: Friends can be judgy. Yes. So I used to explain my code to my son when he was one when I had issues, because he doesn't know anything about programming.

SARAH HANSEN: Right. And so it's the act of talking it through. But you said it has to be in great detail.

ANA BELL: Yes. You can't skip because that's when-- because you already, like, clearly your code is wrong because of you. And so--

SARAH HANSEN: Right.

ANA BELL: And so don't skip any details.

SARAH HANSEN: I feel like this is very much life outside of academia, like 90% of the time the problem is probably you.

ANA BELL: Yeah.

SARAH HANSEN: And it's like, you have to figure out how to interact in the world or with others.

ANA BELL: So I brought all this, all these ducks, a wide variety.

SARAH HANSEN: Yeah.

ANA BELL: So now I have a question for you.

SARAH HANSEN: Uh oh.

ANA BELL: Which duck would you like to keep?

SARAH HANSEN: [GASPS]

ANA BELL: This is a hidden personality test.

SARAH HANSEN: Oh.

SARAH HANSEN: Hmm.

ANA BELL: No, it's not.

SARAH HANSEN: We'll probably have to cut this because I might take a long time. I would-- this guy. This guy. Yeah. He's a little bit offbeat.

ANA BELL: Yeah.

SARAH HANSEN: Eye-catching. And I feel like I could explain stuff to him.

ANA BELL: Yeah.

SARAH HANSEN: I get to keep him?

ANA BELL: Yeah, of course.

SARAH HANSEN: Thank you. That's so great.

ANA BELL: You can put one for the set as well.

SARAH HANSEN: Really? Oh, gosh. Does someone from the crew want to pick a duck for the set?

BRETT the producer: You took the one I would have picked. Oh.

ANA BELL: Oh no.

[LAUGHTER]

SARAH HANSEN: All right, then.

ANA BELL: There's another mohawk guy right over there.

SARAH HANSEN: Yeah. Yeah. We'll go two mohawks.

ANA BELL: There you go.

SARAH HANSEN: Thank you.

ANA BELL: Yeah, of course.

SARAH HANSEN: What a gift.

ANA BELL: So you can use it for programming if you decide to start. Or as you said, even in your day to day life, if you just explain the problem to--

SARAH HANSEN: Yes.

ANA BELL: --something that doesn't judge.

SARAH HANSEN: Yeah. This idea of taking a break.

ANA BELL: Yes.

SARAH HANSEN: The process of programming itself seems quite intense. There's an image in popular culture of a person working alone in the dark, like with a hoodie, and tap, tap, tap for hours on end. That sounds intense.

So there's two pieces. Is that accurate? And two, do people work in isolation like that? Or is it more collaborative?

ANA BELL: So I tend to encourage collaboration.

SARAH HANSEN: OK.

ANA BELL: I have certainly put a hoodie on, and the hood on in the past, just like a leave me alone kind of state, mostly for homework and things like that. But for students learning to program, I definitely encourage collaboration because it's when you talk to somebody else, you see their viewpoint. And in programming, I know I talked about an algorithm, but for a particular algorithm, or even just to solve a specific problem, there's a ton of ways to solve it.

So I learn from in the classroom when I ask students for how did they solve a particular problem after I give them some time to think about it. They have so many different approaches to it that I would never have thought of. And it's important for students to see all these different approaches, because my way might not be the way that they think to approach the problem. So I encourage working together, especially as you're learning. Now after you find your groove, maybe it's not necessary.

SARAH HANSEN: Yeah, as an outsider to this, I didn't realize there was such room for divergent thinking. So that's really cool.

ANA BELL: Yeah.

SARAH HANSEN: But the taking a break. So especially at a place like MIT, I don't know if taking a break is on the menu really. But it should be, I'm guessing.

ANA BELL: It should be, yeah.

SARAH HANSEN: And then I'm wondering if you can talk to us about the thing you created in the world to help people take a break, and why it's important.

ANA BELL: Sure. So this was a project for-- I just had an idea. I was like, let me just make a coloring book, one of those coloring books that you just kind of doodle into. And I thought, let me just make it computer science themed. And so each one of these is, like here's a circuit, and it's like, here's a network of connected people, and then here's a matrix kind of thing, with bits. And so it was just all these ideas from computer science.

And the hope is that if you are stuck and you don't have a duck, then you can take a coloring book, it doesn't have to be this one, but just something that gives your brain a break. But it's this idea of mind wandering. And this is an idea from psychology where mind wandering is not bad. Like if you're folding laundry, right, you're doing a very repetitive task, your brain doesn't have to super focus on it. And so that's when inspiration comes, because you're kind of subconsciously thinking about other things, maybe problems that you have or things that you're worried about.

And that's kind of when solutions come. Just like when you're taking a shower, that's mind wandering as well, because you're doing a test that you already know how to do. You don't need to dedicate any thought to it. So your brain kind of dedicates thought to other things that might be worrying you or on your mind.

SARAH HANSEN: That's great that you've created that. Do you feel like programming is, like, what's the percentage of logic to creativity when it comes to programming? Are we like, 50-50? 80-20?

ANA BELL: Interesting.

SARAH HANSEN: 60-40?

ANA BELL: Maybe 50-50 I would say. I think creativity comes into the process at all the different levels. So when you're coming up with a problem or something that you want to solve using programming, right, you have to be creative. Right? So just coming up with that is mostly creativity.

When you're deciding how to solve the problem or how to write the code, maybe what algorithm to use, or how to structure it, or what pieces to use and where, that requires some creativity, potentially putting things together that you might not have put before, pieces of programming concepts together that you might not have put before.

And then you're obviously going to run into an error after you write the code the first time. And so then, tons of creativity comes into play there.

SARAH HANSEN: Yeah.

ANA BELL: But drawing or talking to a duck--

SARAH HANSEN: I'm liking programming more and more, I have to say.

ANA BELL: I know. Once you're programming, it's kind of like reading a really good book. Like you're in this zen space where you're just like, yeah, this is working. I totally see where this is going. I know what I want to write next. The code just kind of fits together. So just like being in a good book is awesome.

And then the more you program, the more you want to program because you're just gaining these skills along the way. So it's like you're-- like, I play video games. So, you're just constantly looking for new ways to level up, but there's no cap.

SARAH HANSEN: It's really fascinating because unlike video games, most of them, you could use open source coding languages, and financial barriers don't come into play to get that area for growth and that dopamine hit of leveling up.

ANA BELL: Yeah. Yeah.

SARAH HANSEN: Increasing your skills.

ANA BELL: It's just, yeah, it's just kind of a different, like you said, it's a growth mindset. It's a different way of-- it's different fun.

SARAH HANSEN: It's a different fun.

ANA BELL: It's a different fun.

SARAH HANSEN: I am like, more and more I'm like, maybe I want to try this one.

ANA BELL: Well.

SARAH HANSEN: Oh.

ANA BELL: There's a book right here.

SARAH HANSEN: Oh.

[LAUGHTER]

SARAH HANSEN: *Get Programming-- Learn to Code with Python.*

ANA BELL: Yeah. So this I wrote a while ago. It's a book made-- lots of books or lots of things claim to be for beginners. But I wanted to write something that is actually for people who have never programmed before. So I think we don't even get into writing code till chapter 7 or something like that. It's mostly kind of just setting up motivation, but not boring. Like I wanted it to read like a novel, right? Like you're just like, oh, I'm just following along.

So if you want to start, you could also could try this book. You can keep this copy.

SARAH HANSEN: Oh, my gosh, I have to tell you a secret.

ANA BELL: Yes.

SARAH HANSEN: So I requested this through interlibrary loan at my home library. And I asked one of my family members to pick it up, and I was like, just so you know, it's really for me. Because they would be like, why is she getting this book? This must be the wrong thing. I'm like, it's not a mistake. It's actually for me.

But I didn't really believe that. But now I feel like it is for me.

ANA BELL: Oh, great.

SARAH HANSEN: Like I do want to read it now and learn.

ANA BELL: Yeah. Anybody can start. For sure.

SARAH HANSEN: Yeah, that's just great. Do you have any hot tips for newbies who might want to start? Or even people who are like deep into programming, deep into the hoodie already? Like things you're interested in reading or you think other people should view or listen to to start or continue their journeys.

ANA BELL: Mostly tips for newbies, I would say.

SARAH HANSEN: OK.

ANA BELL: Don't get discouraged. Learning curve is steep.

SARAH HANSEN: Yeah. OK.

ANA BELL: Yeah. It sucks.

SARAH HANSEN: Can I text you when I--

ANA BELL: Yes, you can text me.

SARAH HANSEN: Thank you. If the duck doesn't work.

ANA BELL: If the duck doesn't work. Yeah. No, it's, I mean, you're learning a new language. Like it's called a programming language, right? And it's basically learning a new way to write things that you already know how to do or how to write in English. You're just structuring it in a completely different way in a different language.

So the learning curve is steep. But don't get discouraged with GenAI tools now, though, it's easy to get started, right, that's not from scratch. So like I mentioned earlier, if you're stuck on a problem, just ask, prompt it to give you a solution, and then work on the trust but verify piece, where you'll actually get valuable experience debugging, because you already start with a piece of code. So the confidence level is already kind of there. You're like, OK, I'm not starting blank. And then you can try to fix it or agree that it works. And both are really valuable skills to practice as beginner programmers.

SARAH HANSEN: I love the idea of starting with code and practicing debugging.

ANA BELL: Yeah.

SARAH HANSEN: That feels low stakes and doable.

ANA BELL: Totally.

SARAH HANSEN: Yeah. What have I not asked you that I should have asked you?

BRETT the producer: I want to know what games.

ANA BELL: What games what?

SARAH HANSEN: What games you play.

ANA BELL: Oh my gosh. Like video games?

CREW: Yeah.

ANA BELL: I like driving a lot. So I drive--

SARAH HANSEN: Wait, you do not. You said you don't like being stuck in traffic.

ANA BELL: No, I don't being stuck. So I like to drive.

SARAH HANSEN: Oh, I see.

ANA BELL: So I also enjoy driving games.

SARAH HANSEN: OK.

[LAUGHTER]

ANA BELL: Not just around the track, but through random city streets--

SARAH HANSEN: Oh wow.

ANA BELL: --and things like that. Escaping the police, and you know.

SARAH HANSEN: You have a whole other side to you.

CREW: Like *Need For Speed?* Like *GTA?*

ANA BELL: Yeah. Yeah. *GTA* I play, *Need For Speed.* Yeah. Yeah. Yeah. I have the setup with the wheel and yeah. Yeah.

SARAH HANSEN: Oh, wow. Well, thank you so much for being here. I learned about myself. I learned about programming. I have a whole new world view on what is possible with and through programming, and that's just an incredible gift. And I'm so glad we get to share that with a lot of people who might also take that gift and run with it. So thank you.

ANA BELL: That's awesome. Glad to hear. Thank you for the chat. This was awesome.

SARAH HANSEN: And thank you for the duck.

ANA BELL: Yes.

[MUSIC PLAYING]