SuperDataScience

SDS PODCAST EPISODE 991: PAIR PROGRAMMING WITH AI IN YOUR PYTHON NOTEBOOK (FEAT. MARIMO'S TREVOR MANZ)

Jon Krohn:	00:00:00	Imagine being in your notebook. So traditionally your Google CoLab notebook, your Jupyter notebook, and having a brilliant friend alongside with you for that whole journey that seems to know everything about code, about data science, about machine learning, collaborates with you. Wouldn't that be an incredible experience? Welcome to another episode of the SuperDataScience Podcast. Today, my guest is Dr. Trevor Manz from Marimo, and he's here to tell us about Marimo Pair, an open source project that does exactly what I was outlining in the hook at the top of this episode. Such an amazing experience, such a great use of the Agentic capabilities that we have today applied to data analytics, data visualization, data science, AI engineering in that traditional notebook experience. And you'll see in this episode that my mind gets blown. As Trevor describes how these technology works. So you'll hear about the user experience, you'll hear about Under the Hood, how this Marine Opair works, and you'll hear how Trevor's relationship with Marino grew out of open source work that he did, particularly on the open source any widget project.
	00:01:15	You're going to really enjoy this one, especially if you're a hands-on technical practitioner, enjoy it. This episode of Super Data Science is made possible by Anthropic, Excel Data, and Cisco. Dr. Trevor Manz, welcome to the Super Data Science Podcast. Great to have you on. How you doing today?
Trevor Manz:	00:01:31	I'm doing well. Thanks for having me.
Jon Krohn:	00:01:33	My pleasure. We had a good run of episodes recently in person in New York, in studio. It's so nice. Great to have you here. You came all the way from Bed-Stuy into Central Manhattan for this episode. You recently moved to New York from Boston, right?
Trevor Manz:	00:01:50	Yeah. Yeah. A couple years ago.

Jon Krohn:	00:01:52	You were a PhD researcher at a ... Can I pronounce this correctly? Harvard? Harvard? Is that how you say it?
Trevor Manz:	00:01:59	Yes, that's
Jon Krohn:	00:01:59	How you say
Trevor Manz:	00:02:00	It.
Jon Krohn:	00:02:02	Yeah. So you were doing PhD research in bioinformatics at Harvard University in ... I guess that's in Cambridge actually, not exactly in Boston, right?
Trevor Manz:	00:02:10	Yeah. I was at the medical school, which is in Boston. There you go.
Jon Krohn:	00:02:14	Yeah. T-I-L. Yeah. And so that bioinformatics research, it sounds like it focused largely on data visualization. You want to tell us a bit about that?
Trevor Manz:	00:02:24	Yeah. I worked in a lab called the High Dive Group, and we built a lot of interactive visualization software for biologists to help understand their data. And a lot of that, I'm certain we'll get into it today, focused on getting these interactive tools inside of notebook environments.
Jon Krohn:	00:02:40	Yes, yes. We will focus on that a lot. And for people who want to look up this high dive lab, it isn't spelled like jumping off of a high dive board. It's H-I-D-I-V-E. Kind of reminds me of the lettuce leaf endive in the way it's spelled. So yeah, so you were working there on a lot of open source projects, particularly any widget, which I guess took off, and that caught the attention of the Marimo people.
Trevor Manz:	00:03:06	Yeah. Yeah. I worked on Any Widget, which is a toolkit for building interactive widgets for notebook environments. And then the Marimo folks caught wind of it and sort of adopted as a way of creating interactive elements inside of Marimo Notebooks as well.

Jon Krohn: 00:03:21 Nice. And so we've actually, we've had a Marimo episode in the past. One of the co-founders of Marimo Akshay, he was on the episode in episode 911. So we'll have a link to that in the show notes and people can check it out. But for people who aren't just going to stop listening to this episode and go back to that one and listen, can you tell us a little bit about what Marimo notebooks are?

Trevor Manz: 00:03:40 Yeah. My elevator pitch for Marimotebooks is that they are a delightful, reactive Python notebook. And I think there are three keywords there, delightful in that we think a lot about the user experience and about how it actually feels to live and work inside of the data environment. Reactive in that Remo, unlike traditional computational notebooks like Jupyter understands the relationships between yourselves such that when you change a variable or update a value, we understand what cells are sort of stale and need to re-execute and we can do that automatically. And then we're Python focused. So unlike Jupyter Notebooks, which you can have many different kernels, we're hyper focused on the Python ecosystem. And so that has given us the ability to really cater to an audience that loves Python and creates extra superpowers that are oriented around the Python ecosystem.

Jon Krohn: 00:04:31 Nice. Yeah. And to highlight one of the key things there, I think that that big distinction of being reactive is what makes it so much different experientially. Probably most of our users, most of our users, most of our listeners, if they are actually hands-on practitioners in data science or AI or data analytics, they've probably used Jupyter Notebooks in the past. And something that is the common and everyday experience of using those Jupyter notebooks is that you're kind of messing around, you're executing cells in different orders unless you're extremely careful. And so I kind of get into this habit as I'm working in Jupyter Notebooks of making sure that I'm rearranging, resetting all of my variables and executing

		the notebook from the beginning. I'm doing that constantly. I'm just like, because I want to make sure that I'm not missing something or that I don't have something out of order that later it's going to break and I'm not going to remember what I did.
	00:05:32	Marimo solves all those problems.
Trevor Manz:	00:05:34	Yeah. I like to think of it as like, I had a similar muscle memory in working inside of a traditional or Jupyter notebook environment, and there's a lot of just cognitive overhead of trying to understand and keep up with what the state of the notebook is. And Marimo, because it's reactive, allows you to sort of offload that task to a system that will hold you accountable for making sure that you don't do things like delete variables and then rely on values that no longer exist. So our guarantee is like whatever you see in the notebook document is a reflection of the state that you actually have inside the notebook.
Jon Krohn:	00:06:05	Exactly. And so people might already have that kind of experience in a different environment like a Google Sheet or an Excel spreadsheet where it's stateful. And if you go and you update some intermediate value, like you're doing financial calculations and one of the values is the foreign exchange rate between euros and dollars, and that one value affects kind of all the downstream calculations. If you go and update that, all the downstream calculations automatically update and Marimo notebook does the same thing, but looks like a traditional
Trevor Manz:	00:06:41	Data science notebook. Yeah. I like to joke sometimes that we've disguised Excel, but instead of Visual Basic, we give you Python.
Jon Krohn:	00:06:48	Right. Yeah. Nice. So in today's episode, we're not going to rehash everything that we already did with Akshay and 911. We're going to talk about something completely new

and really cool that you guys are doing, which is Marimo Pair, which is an agentic experience. Tell us about that.

Trevor Manz: 00:07:04 So since the beginning in Marimo, we've had different types of integrations with AI coding models. And recently what we've been working on is a new type of experience for our users that we're calling Marimo Pair, which is essentially an agent skill that teaches an agent how to use notebooks as a tool and specifically Marimo Notebooks as a tool. And what this allows the agent to do is anything that you can do inside a notebook and more.

Jon Krohn: 00:07:29 Nice. And so there is something, it seems like huManz ... I mean, and not it seems like, obviously huManz and agents, despite this kind of idea, I think in public perception, especially among non-technical people, that an AI agent is kind of like a replacement for human intelligence. In fact, the way that we process information is vastly different. We have different strengths and weaknesses. And so for example, huManz really like that stateful experience of the Google Sheet, the Excel notebook, the Marimo notebook, but that isn't necessarily something that an agent innately is designed for. So how do you kind of make the notebooks bilingual so that it's optimized both for human intuition, which is stateful and messy, while also handling agentic reasoning, which is deterministic and functional.

Trevor Manz: 00:08:24 Yeah, that's a really good question. I think the definition of an agent is a model plus tools, so commands that that model can run, and then some kind of loop where it can iterate on that process and sort of call itself again. And those tools are really important to how useful that agent is at accomplishing a particular task. And so out of the box, coding agents are really good at reading and writing files and running commands. But as you mentioned, a lot of tools that we like to use for working with data, or we've built very specialized tools for working with data because there is a lot of value of loading your data into memory,

poking around, figuring out where you're needing to go next. But an agent historically has not had access to seeing that state or understanding that state. And so it's been on the responsibility of the tools to be able to sort of mediate how much access to that context that the model can get to do the next task.

00:09:19 And I like to call this as like, if you've been working with notebooks or trying to use notebooks in the past with a coding agent, it sort of feels like you're really babysitting the agent because you have to tell it like, "Oh, that code you added didn't run. Here's the error." And it really feels like kind of this prompting that we had a couple years ago with working with chatbots. Whereas now if you're using coding agents for traditional software development, there are these very long tasks where the agents are able to sort of self-correct, run edit files, update tests, and get in these really type loops where they can iterate and confirm what they did was correct. And so bringing that experience to data has meant trying to open up our system and really open up Marimo such that it can be used as a tool by the agent, the same way it'd be used as a tool by human to understand that state.

00:10:02 So it's not just the code, but it's also the code with whatever their values are in memory. And so we've had to design Marimo Pair in such a way that as you're working, the agent can sort of ask about anything that you could currently see on the screen. And that has really meant just kind of opening up Marimo and allowing agents to rip and write some Python code.

Jon Krohn: 00:10:21 You saying open up there reminds me of something critical that I feel like I should have maybe brought up right at the top of the episode, which is that if I remember correctly, Marimo is completely open source and accessible for free, right?

Trevor Manz:	00:10:34	Yes. Yeah. Marimo is MIT licensed and completely open source.
Jon Krohn:	00:10:37	Yeah. How do you guys make money?
Trevor Manz:	00:10:42	So we are an open source company. In the fall last year, we were acquired by Coreweave, which is- Oh,
Jon Krohn:	00:10:48	Wow. Congrats. I didn't know that.
Trevor Manz:	00:10:50	Okay. Yeah.
Jon Krohn:	00:10:50	Wow.
Trevor Manz:	00:10:51	That's super
Jon Krohn:	00:10:51	Cool.
Trevor Manz:	00:10:52	So now we're part of Coreweave and our direction at Coreweave is to really grow the open source.
Jon Krohn:	00:10:57	Wow, that is a big deal. And that's a really huge win because it allows you to focus so much on building a great experience for people like my listeners as opposed to needing to worry about the commercial aspect downstream because I know that the commercials at Coreweave are going very well.
Trevor Manz:	00:11:13	Yeah. So our direction at Coreweave at the moment is to just double down on the open source. And so we are expanding in many different ways. We're hiring new folks and we're not only working on Marimo open source, but we're taking in new directions like Marimo Pair, working on a VS code extension for Marimo. Our goal is to make Marimo sort of the new open standard for doing computational notebooks in Python.
Jon Krohn:	00:11:33	Yeah, and it makes so much sense. The Jupyter notebook experience, I mean, I hate to keep ... I'm mentioning that one specifically. You kind of had a more diplomatic way of

saying the computational programming notebook. I can't remember exactly what you said, but it is ... Jupyter is the one that I think everyone's used and it needs a rethink. And so I'm glad that you guys are doing this work and now have this kind of backing that will serve you forever.

Trevor Manz: 00:12:08 Yeah. I mean, everyone that is at the company has used or grew up, I'd say, grew up on Jupyter Notebooks. We like Jupyter a lot, but we've definitely had this advantage of thinking about having the second mover effect of being able to see how notebooks are used in practice and specifically Python notebooks. And then sort of from a bottoms up approach of thinking about, okay, how would we make the best sort of Python notebook? And so we definitely have a lot of inspiration from Jupyter.

Jon Krohn: 00:12:36 Yeah, for sure. I remember when my grandparents first showed me my first Jupyter notebook when I was a WeLad. Yeah, exactly. Nice. All right. So it's been described that the relationship between a human and an AI agent in Marimopair, it's kind of like a new kind of contract, not between programs, but between a runtime and a model that reads documentation. And in that, the notebook is like a working memory for the agent to work with.

Trevor Manz: 00:13:03 Yeah. So Marimopair is like maybe a little bit on the technical side. It's implemented as a skill and a skill is a folder of markdown files that teaches an agent how to use a tool.

Jon Krohn: 00:13:14 For sure. And I think a lot of us would probably get exposure to that through say, even if you're in the Claude user interface, like not even in Claude code, not even in some kind of Claude agentic experience, but just being in the regular Cloud in your browser. Now a lot of the time, if you upload a PowerPoint presentation or a PDF, you'll see as the model is thinking, it'll output something like

reading the skill for opening a PowerPoint file or for reading a PDF.

Trevor Manz: 00:13:45 Yeah. So if you have the Marimo skill installed and you say like, oh, I want to pair on this notebook or I want to work on a Marimo notebook, that should trigger the skill to get loaded in a very similar way. And then that skill teaches the agent how to start up Marimo and then once it's started or to connect to an existing Merimo session that you might already have, once that session is running, then how to execute Python code living inside of that Marimo runtime. And that's really the piece where we have this programmatic access point. So you have this, like with traditional software, you normally have APIs that have to talk to one another. And in this case, we have a model that's like talking to some APIs that we have hidden inside of Marimo for the model. And it's a much different type of software because the agent discovers what is available for it to use inside the Marimo Notebook.

00:14:32 And it's not as strict of a contract. So if we change APIs between releases, like the model will just figure out what it has to do and correct and figure out how to do things inside the notebook. And then if you upgrade Marimo, you'll just get more capabilities because we've improved how that works inside the runtime. So it's been a very different type of software to work on because of this boundary between like normally you'd have to worry about semantic versioning and things like that. And in this case, it's just, we just have the skill that says, "Hey, look, here's where all the tools live inside Marimo. When you load it, figure out what you can do. " And then with that, the agent is able to do things on the user's behalf inside the notebook.

Jon Krohn: 00:15:11 Yeah. It's borderline magical and wild. My grandparents with their Jupyter notebook experience when they were growing up, they would never have imagined that we'd have this kind of scenario. Even thinking about skills, I

feel like they're now everywhere. And two months ago, maybe I'm a bit of a dinosaur myself, I think maybe two months ago I had never heard of this skill concept and now I see it everywhere in so many AI interactions, obviously anything involving an agent, it seems like it's become a really helpful approach, an almost magical approach for making it easy to have an agent work with a Marimo notebook. And so since you were working on this, developing that skill integration with Agentic capabilities, do you remember when this whole skill movement started and how it happened?

Trevor Manz: 00:16:08 Yeah. Honestly, I think I might've been a little bit late to it as well. I think the shift in my experience with these coding agents happened around November last year with Opus 4.6 from Anthropic.

Jon Krohn: 00:16:23 I think 4.6 came out in February though.

Trevor Manz: 00:16:24 Okay, okay. Maybe- It might

Jon Krohn: 00:16:26 Have been 4.5.

Trevor Manz: 00:16:27 4.5, yes. It was around last fall, and I think I've heard many people comment on this before, but the industry kind of went dark for the holidays, but then everyone came back in January and was like, "Okay, these models have gone pretty good." Definitely. And then I think with that, the introduction of skills, or it started to feel like there was a lot more value in spending time writing up these skills such that it's a very high impact way to share processes across your team or very specific bespoke types of ways of working with a set of code that maybe before you'd put in a file that people have to really read or share amongst your team, it's like, maybe I can package that up as a skill and now everyone can just download the skill and get up to speed with things. But I had not been using that many skills in my day to day.

	00:17:13	And then when we started trying to think about what's this new experience for how we want our users to work with Marimo, skills seem like a pretty good fit. And then I found that through working on Marimo Pair, sort of building the muscle of like, okay, if I'm repeating myself to my agent, maybe that's something that should belong in a skill and building up the muscle memory of when something goes wrong, going and updating the skill, not just yelling at the model and telling it did something wrong.
Jon Krohn:	00:17:38	Yeah, exactly. In last week's Tuesday episode, so I think that was 987 with Linda Haviv. She also was talking about skills. And for her, it was this innate ... It seemed to her like she'd been doing skills her whole life. She was talking about using them in various context for making agentic workflows that are part of her daily life easier and faster. And I was kind of like blown away. Just like you saying, I feel like I'm so far behind when you see people that are adopting like that. But I think that's just the reality of the world that we live in today where it seems as soon as you come across ... I'm sure there's lots of listeners today who are hearing about skills or thinking about skills in agents for kind of the first time today and they're like, "Oh, I guess this sounds like something I need to know.
	00:18:26	" And once you do that, once you make that leap, once you have experience with it, then all of a sudden you are like, just like you've noticed how they're so useful in so many aspects of getting agentic workflows to work effectively in so many tasks as a data scientist, as a software developer, as just a human, for those of us who are using it in their personal lives as well.
Trevor Manz:	00:18:46	Yeah. I think I will say it was not natural for me to come as a programmer to like ... I've had to learn how to write skills because my instinct as a programmer is to be very prescriptive of exactly what needs to happen. And so the

way that I was going about writing the Merimo pair skill even in the beginning was very imperative, like do this, then this, then this. And what I'd find is that the models follow that very well now. And then what you realize is there are times where there are exceptions, but the models are going to follow the very dogmatic kind of like prescription that you've given. And so over time, the Marimo Pair skill has actually become a lot less prescriptive. It's a lot more ... So instead of imperative, more declarative, and I think that's kind of where things are going.

00:19:31 And that was a learning experience for me to actually be like, oh, maybe my job is to teach the model what the tools are and the nuance about how to use this piece of software, but then not to be so prescriptive about exactly how it should do that because a user's request might actually, if it understands how to use all the tools, it can figure out how to do that on behalf of the user versus if I'm too prescriptive, now there might actually be an annoying user experience because the model's going to dogmatically follow these steps that I've can't and I didn't anticipate something that the user might want.

Jon Krohn: 00:20:01 Yeah. This intuitiveness around the skill functionality reminds me of the intuitiveness, the increasing intuitiveness of even just crafting prompts for the back and forth conversation with an AI model and an LLM generative AI experience where a couple of years ago, supposedly there were jobs that paid like $300,000 a year for prompt engineer, but that quickly went away as they figured out with the training data sets for the reinforcement learning in the post-training of LLMs that we don't need to have somebody crafting prompts in a very specific way, we can actually have the model increasingly magically into it what the right thing to do is in a particular situation and anticipate what you're looking for. And I think that sounds, hopefully that sounds like a reasonable analogy to what the skills are

doing as well, where if you think about it like a programmer, you're very prescriptive, you could end up kind of pigeonholing the LLM when it has so much remarkable intelligence that trusting it to make the right decision ends up with a better result.

Trevor Manz: 00:21:09 Yeah. I think it's all that fuzziness and those edge cases that if you're too prescriptive, then you don't really benefit from the generalness of these models of being able to connect all the different contexts that they're able to grab. And I think when prompting was very, very important was sort of in the era where it was very turn based between the models. And so the only context that they had was exactly what you were sort of like saying inside of that prompt. And as we've shifted now into more like agentic coding flows, part of that context is now just spread on your file system. It's over the network. The agent is able to sort of bring what it needs to into the environment to help ... When you ask a very vague prompt, try to get more context to figure out how to answer that.

00:21:51 And so it's about equipping these models with more tools to bring in the context on demand rather than scoping the context in the beginning with only what you're telling the model.

Jon Krohn: 00:22:00 Yeah. So this wasn't on my plan for discussing what I'm about to ask next, but I think it's really interesting probably for a lot of our listeners and for me personally. So your development environment these days, how do you set it up? Do you use CloudCode? Tell us about how do you code these days?

Trevor Manz: 00:22:19 I use CloudCode with GitWorkTrees and I'm running many Cloud sessions. And then I've always been a terminal coding person. So my review- You've

Jon Krohn: 00:22:31 Always been a terminal coding person.

Trevor Manz: 00:22:33 For the most part, yes. So I'm like a NeoVim. So I feel like the first era of these coding tools came out and everyone shifted from VSCode to Cursor. I wasn't totally ready to give up my editor and then Claude or these Agentic Flows were really the first tools that really fit my workflow because- But you wouldn't

Jon Krohn: 00:22:53 Have been using VS code either.

Trevor Manz: 00:22:55 I wasn't using VS Code. Oh, no. But more so that I think that there was these interesting integrations getting built into IDEs and me with my very bare bones editor was kind of left out of those. Do

Jon Krohn: 00:23:08 You have a new Boon too?

Trevor Manz: 00:23:09 No, no. I'm a Mac user, don't worry. But yeah, so then over time, I listened to some folks that I really trusted talking about these agentic coding tools and specifically cloud code. And then I started giving it a try and it really felt like it fit my workflow. I could put this thing off on a task for a bit and then review it with the tools that I was at hand. And now I'm finding that I'm jumping around my code base a lot less, but I'm using the same kind of tools to help myself review code and look at the dips that are being presented by these tools. Yeah.

Jon Krohn: 00:23:44 Yeah, really cool. Beyond the coding, the development experience, you also obviously have a lot of experience doing data analytics, data visualization, data science in kind of a notebook environment. So maybe even walk through for us, you've now talked us through what your workflow is like when you're coding, but when you're exploring data, when you're doing data visualization, it could be interesting for our audience to hear what a power user of Merimo, how they use that kind of tool.

Trevor Manz: 00:24:15 Yeah. I've always thought that notebooks hold this really special place in the ecosystem because they are this

environment that marries your code and data together. And so although I've been sort of this bare bones text editor for my traditional software development, I've always used some kind of notebook when I wanted to do data work or some sort of live Repel. I think my first thing I was in was our studio and then I switched over to Python and started using Jupyter and now I'm a Marino user. Yeah. So I usually would spin up a notebook from the command line and just jump in and start loading my data. And now I find myself with the Marimo Pair skill.

00:24:56 Anytime I get an issue or often the place that I start to work on a problem is now in my agentic coding tool. So for me, that's often cloud code. If that looks like some tasks that I want to load some data, explore that data, I'm going to start asking to start up a Marimo pair session and then start describing a very high level sort of, these are the data sets that I want to look at. These are libraries I think that I want to do. And sort of being a lot, like I was saying before, a lot more declarative about the approach that I want to take. I know exactly what I want to do with my data and-

Jon Krohn: 00:25:26 This is maybe a really dumb question, but when you start doing that Marimo pair session, that's in the command line or that's in a notebook environment?

Trevor Manz: 00:25:34 That's in your agent. So it'd be like open code, cloud code, codex. You can just say like, "Hey, I want to start working on my notebook." That the Marimo Pair skill teaches the agent how to start Merimo or connect to Merimo for you and then you're driving Marimo from your agent.

Jon Krohn: 00:25:51 Whoa. So it would be kind of like you'd have it open in a browser window and so you'd be working and say you could have ClaudeCode running in a terminal and so you're typing into that terminal, but then the agent is changing things in your Marimo notebook. Exactly. It's open separately.

Trevor Manz: 00:26:08 Yes. And you can also collaborate on it. So you have the Marimo Notbook interface. So if the code that it generates or there's some tweak that you want to make to a cell, then you can go edit that by hand in the Marimo notebook. But then you can ask Claude, you say, "Hey, I circled something that's interesting in this notebook, can you tell me about it? " And Claude has access to that state now because of the way that they share this context. So it's like all that rich information that historically has been trapped in that environment now is more context that you can offer to the agent beyond your file system.

Jon Krohn: 00:26:39 That's so cool. I love that. Does it do things like if you ... So if it had kind of been developing a notebook, had been getting going on some data analysis for you, and then you go over to your Chrome browser or whatever, and you make a change to a cell, could you end up in a situation where like either in the notebook experience or in the terminal window, it's kind of like it's watching what you're doing and it's just kind of like, "Hey, I like what you did

Trevor Manz: 00:27:08 There." Yeah. Right now, well, we're trying to get our better feedback loops of if you've made edits for us to stream those back. There's a feature inside of MCP that's called channels I think we might explore in that direction. At the moment, it's a lot. Claude has, or your agent has access to running code inside the kernel. So if you've made a change, it can grab any state that it wants inside the notebook. It can see your cells, it can take screenshots of cells now and look at those. So if you say like, "Hey, there's something interesting here, I'm not quite sure what it is, go off and use the other tools at your expense to help me understand what this is. " At your expense. Yeah, exactly. Then that's now context that you can feed back into getting somewhere with your data. So I like to think of it as when I started really adopting agentic coding tools for traditional software development, ideas for software ideas started to feel a lot cheaper in the

sense that, okay, that's not going to take me an afternoon.

00:28:05 I could do that really, maybe just do this proof of concept. And I've had so many of those times inside of notebooks where there's something I know that I should look at or explore, but I don't want to figure out the API or I can't remember exactly this thing. And instead, I can just say, "Maybe we should try these different validations first and then let it work on that problem and then make some plots and bring me back when I can make my decision."

Jon Krohn: 00:28:26 I also like the idea of in the not too distant future, being able to say to the agent, "Hey, you know that $8,000 you made last month on that vending machine business you run?" I use some of that money if you feel like it's worth it on tokens to do this analysis.

00:28:45 Yeah, it is pretty wild. And it's interesting how this vending machine problem has become such a benchmark of agentic capability, but it's a good one. I like it. I'll have a link in the show notes too, information on this vending machine benchmark if people aren't aware of it, but it seems like agents could be making a lot of money on vending machine businesses in particular. All right, so thank you for that overview of the experience and actually you could probably even tell from the way I've been reacting, this has been a really eye-opening experience. It's great that you folks at Marimo are thinking so creatively about what the data scientist or researcher or data analyst, anybody who's working with data, reimagining what their experience could be like in this agentic era. And I think you've landed on something really cool here I candidly hadn't used, as you can tell by the way I'm asking questions and being blown away.

00:29:42 I hadn't used Marie Mo Pair before recording this episode with you, but now I'm so excited to do it because it sounds like such a fun experience.

Trevor Manz: 00:29:49 Yeah. I think the thing that has been really important to us and to our users is like, what are people trying to accomplish inside Notebooks? And as someone that like Myself worked in notebooks and have worked with a lot of scientists that use notebooks. Often the code was a means to an end of I have this data problem and the thing that they care about is the data and the problem. And so giving them a better tool to be able to maybe offload some of the coding part to answer that task. While at the same time, and it's worth iterating, the thing that you're working on is a Marino notebook. So it is this reproducible artifact. So the thing is not like a set of logs or just some Python code that ran or some scripts that end up on the system with an answer.

00:30:31 It's a Marino notebook. So that ends up being very important to stakeholders that want some artifact that they can audit and see like, how'd you get to this answer? It's all this code that's encapsulated inside that artifact.

Jon Krohn: 00:30:43 100%. I love when my stakeholders are getting into my notebook. And that happens all the time. No, I know exactly what you mean. It is super useful if even for you or people on your team to be able to come back later, to be able to collaborate on these things and to have a sense of where the data come from. But yeah, historically it's not my experience that the executive sponsor isn't there. Nice. So that gives us a really great sense of the user experience using Remo Pair. I'd like to get a little bit more into how this works so seamlessly. So you recently posted on LinkedIn about recursive language models, which candidly is a specific term that I don't think I'd come across before. But it intuitively makes sense when we explain what it is. So recursive language models in which recursive reasoning may represent the next frontier after chain of thought and react reasoning plus acting paradigms.

00:31:47 So yeah, tell us about recursive language models and why they could be the future.

Trevor Manz: 00:31:51 Yeah. I think honestly, if you're using a coding agent today, I think that they are some form of a recursive language model. And I'll elaborate a little bit on what that is. It's some research that came out last year, but the idea of recursive language model is a system where essentially you have a model that it has some environment that it can offload its context to. So previously, like everything that you had to do with your, if you've been using agents, a thing that people start to get very worried about is like context rot, where there's this phenomenon that as you increase the context for the model, they start to forget or like go off track. And so people are pretty conservative about what they put into context because they want to keep their model on track. And so one way that agents or a recursive language model helps with this is that they create an environment where the agent has the ability to, rather than taking the data and putting it in context, it can offload that to some part of that environment.

00:32:44 And so for a coding agent today, that's your file system. It can take your corpus of data and things that normally you might need to shove in as context and just say, actually those are files on the file system. So do you want to look there? That's where they live. And you have tools to access that context when you might need it. So Marimo Pair sort of extends this paradigm by also bringing it- Hair Maradigm. Yeah. Marimopair extends this paradigm by

00:33:11 Allowing the agent to extend that environment further, but with Python. So now that context isn't just files, it can be values that you have in memory. And so instead of it having to try to understand all of the context of your data and pollute the context, instead you have your Python environment as another place that it can store that.

Jon Krohn: 00:33:32 Really cool. How do you think that this kind of recursive reasoning architecture might reshape our understanding of intelligence and particularly within collaborative human AI workflows?

Trevor Manz: 00:33:48 Yeah. Well, at least in the context of Marimo Pair, your notebook sort of is a representation of that memory. So the way that we, a lot of coding agents folks are thinking about your file system as the working memory for an agent. Now when you look at a Marimo Notebook document, the notebook itself is sort of a representation of that memory. So if I put a coding agent on a data task many different times, like the same question a bunch of times, it might find 10 different ways to use the file system to answer that question, but Marimo sort of puts a little bit more of a structure around how that answer has to look. And now at the end, your notebook might have some answer in it, but you have this really similar scaffold and each of the cells is sort of a representation of what the memory that it took to get to that answer.

Jon Krohn: 00:34:39 Wow. Super cool to think how things are going to continue to quickly evolve and our experience engaging with agents that can help us on any kind of task, including our traditional data science tasks, really cool. Let's dig a bit now into what you were doing before Marimo, because you did a lot of really interesting work that led to a lot of public results and open source work. And so we have a fair bit of research on it. When you were in the high dive lab at Harvard Medical School, you were focused on building interactive visualization tools for computational biologists in particular so that they could analyze data more effectively. And as a consequence of that, over the years, you collaborated on several open source projects, including a library for multi-scale visualization of high resolution, multiplexed bioimaging data called Viv on a scalable data profiler called Quack with no C and a toolkit for working with chunked

compressed N-dimensional, so like high-dimensional arrays, and that was called Zurita.js.

00:35:50 And so I'll have links to all of those in the show notes. But as we mentioned earlier in the episode and ended up being how you transitioned into your role at Marimo, you created any widget, which is a unified standard that lets Python developers bring custom interactive widgets into notebook environments across platforms. So I don't know if you want to talk about any of those other projects or we can just dig into any widget in particular.

Trevor Manz: 00:36:14 I mean, maybe zooming out about a common thread of all the open source stuff I've worked on. I was a big visualization nerd and I love interactive visualization tools, but I always found that there's always a cost associated with using a visualization tool. So often we had users that lived inside something like a notebook environment, but we made this application that was like this, had some interesting visualizations and hopefully could help them come to new insider understanding with their data. But that ask of trying to convince someone to use a new tool is like a really something I took for granted in the beginning because people don't want, they are often conservative in the tools that they adopt and adopting a new tool. It has to do a lot of things. So I started focusing on the data science ecosystem and specifically notebooks because there was such this powerful and useful tool and they're everywhere.

00:37:06 So people use Google Collab, they use Jupyter, they use Marimo now. And what I found was like we had all these web-based visualization tools that didn't have a good way of getting fit inside of those environments. And so a widget in Jupyter terms is this interactive element that's based ... The front end is built with some web technology. And then the Python side is just a Python object and there's this machinery that is done that allows the Python side to talk to the front end and backwards. And it's

called like bidirectional communication. And that's a really powerful extension when you're living inside these notebook environments because it allows you to take something like a data frame or a dictionary, an object or something that is like data in Python and then get just this really custom interactive view inside the notebook. And any widget was sort of a tool that scratched my own itch of building these types of things was kind of difficult and making sure that they worked across all the environments that people were using was really challenging.

00:38:00 And so yeah, through that I came up with any widget was sort of both a standard, which is how do you define and create these interactive elements and then the implementation that wires it up on the Python side.

Jon Krohn: 00:38:11 Yeah. And it really took off. So congrats on that. In a Sci-Pi conference talk, which I'll try to remember to put a link to in the show notes, you drew a fascinating analogy between chemical catalysts that lower activation energy and widgets that lower the overhead required to achieve data insight. You want to elaborate on that one a bit

Trevor Manz: 00:38:30 More? Yeah. That sort of goes back to what I was saying about the overhead of using tools. When you're in the Python ecosystem, the overhead of bringing in something like an algorithm feels very cheap because it's an import away or like an install away. But I thought a lot during my PhD about what is it about visualization software that makes that feel like it's not quite at hand. And I think a lot of it is just because the way that they're packaged up or these tools aren't just a PIP install away. It's like, you got to download this app, you got to export your data. So even though visualizations are really useful and helpful for coming to Insight, they are not quite at hand as the rest of the Python ecosystem. And so the analogy I was drawing with the reaction diagram was like, catalysts are a thing that can lower that overhead such that you can

get to the outcome that you want faster with less overhead.

00:39:23 And I think widgets are a way of taking these interesting visualization toolkits and packaging them up in a way such that they are a lot more like your Python libraries that you just import and use inside your notebook.

Jon Krohn: 00:39:33 Nice. And I suspect that ease of use kind of relates to being able to be more creative, being able to be more playful, which brings me to my next topic. What's iPie Mario?

Trevor Manz: 00:39:48 Good question. iPi Mario is a fun toolkit that I made that is an any widget that I sort of did a live recording of making to just try to give like the bread and butter of how do you create a custom widget because I think we have a lot of folks in the community that either come from the Python side or they come from the website and they want it and you kind of have to know these two worlds to start making them some of your own. So yeah, I made this sort of dummy project that is a little Mario Brick that when you click on it, it jumps inside of a notebook. And we started with like a piece of JavaScript code I found on Twitter and then we go all the way to the end where it's a package that you publish to PiPI.

00:40:28 And

Jon Krohn: 00:40:29 What does it do?

Trevor Manz: 00:40:30 It does nothing other than just, it shows you how to build such an integration and take this piece of functionality and integrate inside a notebook. I see. I see. But sometimes play is nice inside of these environments. And

Jon Krohn: 00:40:42 So an interactive element, it shows you how to get all that from end

Trevor Manz: 00:40:45 To end, how to put

Jon Krohn: 00:40:45 That into a notebook.

Trevor Manz: 00:40:46 And it's hopefully trying to teach people the practices of, okay, you have this state and Python, you want to bring it into the front end, this is how you do that. And so it's a very toy and dummy example, but I've seen enough buttons where I thought I wanted something that was a little bit more fun.

Jon Krohn: 00:41:00 Nice. And so it's interesting how you have this PhD in bioinformatics, nominally. A lot of it focused on data visualization. It's interesting how you often think of data science, machine learning, AI, working with Jupyter Notebooks is kind of like backend development in a way, but actually a lot of what you've done is around user experience and the front end. Have you always been interested in this kind of like, I guess, the full stack of the application experience?

Trevor Manz: 00:41:36 Yeah. I think I did my undergrad in chemistry and not any software engineering, but I made websites for fun and I did not realize that these two worlds could be connected in any way. But then as I started doing more research, I started analyzing my more data and I guess I was using more of the data science tool stack. And so I think over time I realized that these ecosystems are very good at different things. The Python data ecosystem is just so rich and battle tested for doing anything that you kind of want to do with data, but then the web ecosystem is not good at any of that stuff, but it's incredibly rich and accessible for building interactive interfaces. So finding the right environment where you can sort of plug those two things together, I think is a really powerful way to provide new types of insight with data because often the visualizations that we have at hand might not be the best representation for what you're trying to understand.

00:42:36 And being able to add this extra dimension of beyond being able to type and run cells, type code and run cells,

now I can create a button that I click and that triggers something in my notebook or being able to make a selection inside of a widget that then is like data that comes back into my notebook, like something like Python. It's just adding these extra elements of interaction that ... Yeah, when you see an outlier, your thought is not, "Okay, how do I write the query that selects that point?" It's like, "I want to just touch that point and tell me what that is and understand that inside the notebook." So there are certain tasks that are better suited for an interface and some that are better suited in code and you just want an environment that lets you sort of play to whatever task you want.

Jon Krohn: 00:43:16 That's a really good example. I already called myself a dinosaur once, but here's another example of how you just described that great experience of there where I could have a box and whisker plot with a couple of outliers hanging out beyond the whisker on that box and whisker plot. And of course, my mind is in a traditional experience where I couldn't possibly imagine that I could grab my mouse and click on one of those outlying data points and get information on it. Instead, I would have to, before a code gen tool, I would have to come up with a way of slicing my Python data frame and be able to isolate a few rows of the table that are over a certain value.

Trevor Manz: 00:44:01 Yeah. You're looking at it, the axes and you're like, "Okay, what exactly is that? " I think

Jon Krohn: 00:44:05 That's higher than 95. So I'm going to say, yeah, anything with this column value of greater than 95, I'd like to see that row in this data frame. And that's obviously not a natural way of being.

Trevor Manz: 00:44:19 Yeah. And I think to bring it back to on the Marimo side and our reactivity, that selection that you make can now be a variable that retriggers a cell that runs downstream. So if you select a point, now you can write your other

cells that are dependent on that point and it flows through that reactivity. And that's a really powerful way of, rather than you manually typing in those predicates to filter your data frame, just use the tool that's good at selecting those points and then you have your code downstream that summarizes them or tells you something about them.

Jon Krohn: 00:44:51 Yeah, it's pretty cool. You have mentioned in an interview in the past that the web and Python ecosystems have been around for almost the exact same amount of time and have evolved in parallel and kind of mostly separately, but you are on a mission to kind of bridge these two worlds together. And so it seems like with any widget, with Marimo, with Marino Pair, we will have more and more of these interactions between front end, backend, as well as human and agent. It's

Trevor Manz: 00:45:24 Pretty cool. Yeah. I think the thing that excites me the most is that about this toolkit now is before if you wanted to make these types of integrations, I think you had to be very deep on both Python and the front end side to plug them together. But now if you've got something that you want to look at inside of a notebook, our Marinemo pair skill has some best in class instructions of how to build any widgets hidden inside of it as a reference. So if you start talking about wanting to make an any widget, the model is going to help you maybe if you're not a front end person, understand how to wire those things up and start building something. And I think what I was saying before about the value of skills is being able to package up domain or specialized knowledge and having high impact in the way that you distribute it.

00:46:09 Folks on my team build widgets, we build these types of integrated experiences and we want our users to just feel creative and not limited by the code to be able to build the things that they wanted and to help them inside their notebook understand their data problem.

Jon Krohn:	00:46:22	Yeah, my brain is so rooted in its old ways that when you talked about the best practices for creating in any widget being in the skill, my mind was like, "Oh, cool. I can open up the skill and read it, but it's not for me.
Trevor Manz:	00:46:35	" Yeah, it is. I will say the nice thing about skills is that they are very auditable and readable. So if you did want to learn about, I mean, it's in the anywidge documentation too, but it also makes sense in the Marimo context to explain exactly how widgets interface with Marimo. And if the model's going to go off, we can provide that extra opinion such that the user has a good experience inside the notebook.
Jon Krohn:	00:46:59	Right, right, right. Intended for the agent, but human useful too.
Trevor Manz:	00:47:02	Yes, exactly. Yeah. I had someone on my team recently say that the funny thing about everyone starting to write ClaudeMD files or agent.md files or skills is that these should all just be contributing.md file. The things that we should have in our repos for other people to contribute, but now because you can pick, maybe it was just N of one working on that project before and now it's me plus a fleet of these coding tools, it's been useful to ... It's actually been a forcing function for me to write it down somewhere, how to be most effective in contributing.
Jon Krohn:	00:47:32	Right, right, right. Yeah, that makes a lot of sense. Beyond your Python experience, which a lot of our listeners would have experience with as well, you have a fair bit of experience with Typescript. Yeah, what is Typescript for? Why is that a language that you use?
Trevor Manz:	00:47:47	Yeah, Typescript comes fully from working on the website. So it's like anything that ... When you want to start building out these interactive interfaces, like the primitive that you get in the browser as a language is JavaScript. Typescript is the typed version of that language. I

Jon Krohn:	00:48:05	See. Yeah. So that was the key thing. I'm pretty familiar with JavaScript.
Trevor Manz:	00:48:09	Yes. Probably
Jon Krohn:	00:48:09	A lot of our listeners are. It's kind of the default for building any kind of the front end of any web app these days, but okay, so TypeScript is hard typed, so you have to declare specifically every kind of variable type that you're going to be using.
Trevor Manz:	00:48:25	Yeah. It's like Python with typeins essentially. The main difference is that Python itself will actually run that code with the typeins in. The browser has no idea what TypeScript is. And if you put that in, it would just have a syntax there. So there is, when you make a package or you publish that code or prepare it for the browser, you have to strip out all that information. But when you're building really complex systems with JavaScript, it's helpful to have a Typechecker help you.
Jon Krohn:	00:48:50	And so that stripping out that kind of happens automatically as you compile
Trevor Manz:	00:48:54	Or ... Exactly. Yeah. There's like a transformation step and there's a bunch of tools that do it.
Jon Krohn:	00:48:58	Right. And then so I didn't just arbitrarily ask this TypeScript
Trevor Manz:	00:49:01	Question.
Jon Krohn:	00:49:03	I was doing it because a drawing from your work in both Python and TypeScript, how can something called the progressive software design philosophy be baked into languages themselves? So you've talked about this before, and it relates even in your answer there. You were kind of talking about Typescript being useful with very large projects. And so it sounds like this progressive software

design philosophy is about moving from a single file to a modular project.

Trevor Manz: 00:49:29 Do

Jon Krohn: 00:49:29 You want to tell us a bit more about that?

Trevor Manz: 00:49:31 Yeah. Well, so my inspiration there was when the original way that you'd create these custom widgets, you'd have to create a full project and then you were sort of burdened by all this tooling if you wanted to use something like TypeScript to be able, because now instead of just opening up your browser and writing some JavaScript code, you now have a build step before you're able to see that output. And I think that that overhead to play or trying out ideas means that it's just harder to try them out in the first place. And so you have to be confident in your idea before you try it out because it's only worth setting up all that architecture to be able to try it out if you know that you've got a good idea. And so one of the philosophies in any widget is that it should just be JavaScript and that the starting point, you should be able to prototype a widget inside of a notebook the way that you would prototype a Python function.

00:50:27 And then when you want to reuse that function, maybe you move it out of the notebook, you put it in a library. And I wanted that same kind of experience for building these interactive elements.

Jon Krohn: 00:50:35 It sounds like something important for us to be aware of as everyone in the AI space seems to be generating far more code than ever before, starting to think about how we make things modular, make things reusable as opposed to just having these monolithic files is important. And yeah, it sounds like this progressive software design philosophy that you're describing is a nice approach to be doing that, to do that.

Trevor Manz: 00:51:01 Yeah. I think just getting the muscle of, don't feel burdened by trying out an idea, but maybe have something in the back of your head of like, "Okay, this is growing at this scale, maybe we need to start applying different tools that help me make sure I'm accountable for maintaining this bit of code." Using types or type hints is a useful way of having very cheap checks that can eliminate a type of problem that ... To what you were saying about offloading context, being able to have a type checker, offload the thought of, is this value an integer here is really a useful context to have when you're focusing on a problem. And also if that code is being generated not by you and by an LLM, being able to look at a function and say, "Okay, this thing can only ever be an int, that can be useful." So it does take experience with understanding at what scale do these things apply, but then certainly at the size of the code base of something like Marimo that we're working on now, we really put a lot of time into making sure that we have good checks and making sure that what we're bringing into that code base, because we also are generating a lot of code is something that we can maintain over time.

Jon Krohn: 00:52:08 Brilliant. Great. Thank you for all of the useful tips today. I'd like to spend a little bit of time at the end of this episode on the bioinformatics research that you were accelerating with your data visualization research. So we've talked about Marimo Pair, we've talked about any widget. So yeah, let's now talk a bit more in detail about your time in the high dive lab at Harvard. So you focused on this critical intersection of biomedical informatics and composable visualization tools for genomics and microscopy. So this was tools like I candidly had never heard of these before and I'm probably going to mispronounce them, but things like Vitessi, Highglass Python and Goss. Yes.

Trevor Manz: 00:52:48 Yeah.

Jon Krohn: 00:52:50 And so from previous interviews that you've given, it sounds like you view visualization not just as a final output, which I think a lot of us do, but as a fundamental way for researchers to interrogate the foundations of their scientific models. So yeah, how do composable modular systems change how scientific tools evolve compared to tightly integrated platforms? Yeah. How do these kinds of visualizations that you recommend this whole way of working around visualizations around coding improve scientific inquiry?

Trevor Manz: 00:53:26 Yeah. I think I've heard it described before as they're visualization people and they're not visualization people. And I was very quick to being a visualization person because, and as a result, at every step of a pipeline or data process that I am working on, I want to visually have some check of what is going through my system. And the number of times that I have loaded a data set and there have been missing values or something at a very early stage that if I caught it and often visually it's very easy to catch those types of things, I wouldn't have spent an afternoon going down some path and then finding out, oh yes, or baking in some part of that data into a result. And so I think that a place where like my lab and a research in the high dive group, it's a natural fit for an area like bioinformatics or computational biology because there are a lot of experimenters that are building out new types of techniques for acquiring different types of data about molecular systems or cell systems.

00:54:26 And they often don't have a ... The data are very abstract and it's like hard to have some representation that is familiar in a way that you can understand what that signal is in the context of what you're trying to understand. And a classic example in genomics is like a genome browser. So there's not an off the shelf tool that lets you just look at regions of the genome.That's a tool that came out of biologists wanting to understand genes and proximity of genes together. And so now we have this

ability to have a genome browser when we're talking about something like this gene, we can look at and see where that is on the genome. We can compare that across different variants and things like that. And then when I worked in the microscopy side, we were taking these massive images of these tissues. And again, there aren't these great tools.

00:55:12 I mean, there are several different tools for doing these types of things, but these tools often are like one-off applications. They don't like live in the environment that you're working on your data. So maybe like I'm running an algorithm to try to segment this image and it'd be really useful to look at that segmentation over the image and not make sure that my thresholding or something is actually quite good. So being able to make these tools feel very at hand for doing those cheap kind of checks is like really where I got inspired to work on this type of work. And then after that, it's like a matter of coding, I guess, to try to fit them together.

Jon Krohn: 00:55:44 Right, right, right. Fascinating. Yeah, you probably wouldn't know this about me, but a lot of my data science exposure was ... I cut my teeth on bioinformatics data as well, like genomics data, and then mapping that to phenotypes or having intermediate RNA expression data. Yeah. So I find that whole area interesting. We probably could just spend a whole episode on it, but I'm actually, I'm going to get to my final topic area here, which is just a bit of a ... We've talked about all the exciting things that you're doing now and in your recent past, but actually your whole career arc is pretty interesting. And so I wanted to give you the chance for the audience to hear about it and for you to fill us in on maybe some takeaways from that. So before being a founding engineer at Marimo, your journey started in a small town in Wisconsin.

00:56:40 And during college, you were an accomplished competitive swimmer, earning high grades and being active in a lot of different campus community initiatives. And then you were a National Science Foundation graduate research fellow at Harvard. Wait, oh no, Harvard, right, right, right. And a visiting researcher at the University of Cambridge in the UK. So yeah, rigorous PhD research, a really exciting startup that's now owned by a really cool company like Coreweave, elite athletic competition, amazing universities like Harvard and Cambridge.

00:57:16 Do you have any advice for people on how ... Are there transferable things across all these different domains, habits that you've developed maybe that allow you to be so effective at these different things that you pursue?

Trevor Manz: 00:57:33 Well, that's kind of what you just say. I guess throughout my career, having the energy to just spend time and go deep on a topic is ... I don't know what matches that type of experience of trying to learn something new. And I even find myself fighting it today in the sense that we now have these tools that are very good at giving us answers very quickly that I am actively trying to find ways to keep myself sharp. And when I do get interested in something, to actually give myself the ability to go deep and learn about it. So I think it's easy to look at your career or your timeline and you construct a linear narrative of like, "I did this to do this, to do this, to do this. " And my story was definitely not that. It was like all along the way, I did chemistry, I swam, I had no idea what I was going to do after college and then opportunities come up, you take those opportunities and I think giving yourself the permission when something is interesting to go deep and even if it's just for the sake of whatever interest that you have in it, I think that that gives you the ability to cut yourself or allow yourself to become a domain expert in whatever that is and-

Jon Krohn:	00:58:49	Cut your teeth.
Trevor Manz:	00:58:50	Cut your teeth. Yeah, not cut yourself.
Jon Krohn:	00:58:52	Well, just cutting myself the whole way.
Trevor Manz:	00:58:54	Yeah, exactly. And I think there will always be value in going deep on a topic and becoming a domain expert on it.
Jon Krohn:	00:59:02	Nice. Yeah. And I think for you, for me, for probably a lot of our listeners, there's an inherent enjoyment in going deep and developing those new neural connections. There's something for me, when you think about the beginning of a calculus textbook and you work through all the exercises, if you were to open the calculus textbook at the back and start from there, it would look like gobbledygook, but if you do it from the beginning to the end, you get to the end and everything makes sense. That's a really cool experience. I don't know, for me, it's one of the kind of peak experiences you can have and yeah, it gets harder first with phones and constant access to the internet over laptops. And I think it was easier in a lot of ways to go deep on a given topic that you're interested in before I had a phone and I just went to the library and walked around and was like, "Oh, those books in this adjacent bookshelf also look interesting.
	01:00:09	Let me take a look at that. " In theory, any of those kinds of things could happen on the internet or in conversation with an LLM or while you're collaborating with an agent on some task. But somehow in practice, it seems to happen less or it seems like there's the activation energy to go and do that. It seems like it requires more for me to get over that hump.
Trevor Manz:	01:00:35	Yeah. I think curiosity will always be valuable and catering to your curiosity and indulging in some interest

		is just ... Well, wherever technology takes us, stay curious, keep doing things.
Jon Krohn:	01:00:53	Yeah. It actually is funny that I wasn't going to get to it, but the very last question that I had in this set of potential questions and topics that our researcher surge Masis put together, the very, very last question is that you mentioned that your curiosity led you from a single tweet by Mike Boestock to fully packaging a Python widget for Pipey. And so curiosity is ... Yeah, it sounds like it's kind of a common thread and a key part of your success. So thank you for that. Nice. Well, while I've gotten through all of our questions specific to you, you may or may not know that I end every episode with the same two questions. And one of them's very easy though. The final question is very easy. The Penn Ultimate one, I usually give guests a heads up because sometimes they like time to think about an answer to this.
	01:01:40	Do you have a book recommendation for us? And you can actually, we can cut the cameras and you can think about it for a little bit if you want.
Trevor Manz:	01:01:50	I've been reading Soul of the New Machine recently. It's by Karence. I'm forgetting the author.
Jon Krohn:	01:01:58	It's okay. See, that's the kind of thing that people get to do when I give
Trevor Manz:	01:02:00	Them a
Jon Krohn:	01:02:01	Heads up, which I remember to do 99% of
Trevor Manz:	01:02:03	The time, but didn't
Jon Krohn:	01:02:03	Today.
Trevor Manz:	01:02:04	Yeah.
Jon Krohn:	01:02:05	We'll find that. Soul of the Machine?

Trevor Manz: 01:02:07 Soul of the New Machine. Soul of the New Machine.

Jon Krohn: 01:02:08 Yeah. Yeah. I'll have that in the show notes and whatever Terrence's last name is, we'll get there. Nice. What's it about? Is it fiction, nonfiction?

Trevor Manz: 01:02:15 No, it's nonfiction. It's about a computer company in the 80s. People that care a lot about the craft. And I think that in an era where there's a lot of code that's getting put out into production, it's just cool to see people and hear about these stories of folks that really, really care.

Jon Krohn: 01:02:33 Nice.

Trevor Manz: 01:02:33 Yeah.

Jon Krohn: 01:02:33 Great. Sounds like a great read. And then I promise this would be an easy one. Okay. This episode has been sensational. I have loved learning from you. The way that you talk about everything is so easy to understand, even when it's relatively complex technical topic. And so I'm sure there are a lot of listeners that would love to be able to follow you for your thoughts after this episode. What's the best way to do that?

Trevor Manz: 01:02:58 You can follow me on X. I'm Trev Manz there. I'm on Blue Sky, LinkedIn. If you're interested in the GitHub or on open source contributions, I'm on GitHub. I'm on the Marimo repo. So yeah.

Jon Krohn: 01:03:10 Nice. Fantastic. And yeah, of course it's worth a mention that there's quite a few open source projects that were mentioned in this episode. Marimo, any widget, obviously. And so get in there and contribute to those projects as well if you want.

Trevor Manz: 01:03:25 Yeah, definitely.

Jon Krohn:	01:03:27	Nice. All right. Well, Trevor, thank you for taking the time. Again, such a great episode. And it was nice shooting with you here in New York.
Trevor Manz:	01:03:34	Yeah, thanks for having me.
Jon Krohn:	01:03:36	Absolutely stellar episode today with Dr. Trevor Manz. In this episode, he covered Marimo Pear, an agent skill that teaches coding agents like Claude Code, how to drive a Marimo notebook, reading Cell State, running Python in the kernel, taking screenshots of cells, and iterating on data tasks the way agents iterate on traditional software. He talked about recursive language models, which give an agent an environment, a file system, a Python runtime, a notebook to offload context into, instead of cramming everything into the prompt and triggering context rug. He talked about any widget, his open source project that defines a unified standard for building interactive widgets that work across Jupyter, Collab and Marimo notebooks, plus the Python machinery that wires the front end and backend together. And he also gave us his advice that no matter where AI takes us, curiosity and the willingness to go deep on a topic and become a domain expert will always be valuable.
	01:04:29	As always, you can get all the show notes, including the transcript for this episode, the video recording, any materials mentioned on the show, the URLs for Trevor's social media profiles, as well as my own at superdatascience.com/991. Thanks, of course, everyone on the SuperDataScience podcast team, our podcast manager, Sonja Brajovic, media editor, Mario Pombo, our partnerships team Natalie Ziajski, our researcher, Serg Masís writer, Dr. Zara Karschay, and our founder Kirill Eremenko. Thanks to all of them for producing another super episode for us today for enabling that super team to create this free podcast for you. We are deeply grateful to our sponsors. You can support the show by checking out our sponsor's links in the show notes. And if you'd ever

like to sponsor an episode of yourself, you can find out how to do that at jonkrohn.com/podcast. Otherwise, please do support this show by sharing this episode with someone who would love to listen to it or watch it, review the show on your favorite podcasting platform or YouTube.

01:05:25 Subscribe if you're not already a subscriber. But most importantly, I hope you just keep on tuning in. I'm so grateful to have you listening, and I hope I can continue to make episodes you love for years and years to come. Till next time, keep on rocking it out there, and I'm looking forward to enjoying another round of the SuperDataScience Podcast with you very soon.