SDS PODCAST EPISODE 911: THE FUTURE OF PYTHON NOTEBOOKS IS HERE, WITH MARIMO'S DR. AKSHAY AGRAWAL

Jon Krohn:	00:00	Welcome to episode number 911. I'm your host, Jon Krohn. Starting with today's episode, we're trying out much shorter intros, so I'll quickly tell you that you're in for a treat with today's guest, Dr. Akshay Agrawal. Akshay has built a clever open source replacement for Jupyter Notebooks, based on the pain points he experienced with Notebooks while completing his PhD at Stanford. Akshay is smart, well-spoken, humble. I think you'll really enjoy this one.
	00:25	This episode of SuperDataScience is made possible by Dell, NVIDIA and AWS.
	00:32	Akshay, welcome to the SuperDataScience podcast. Where are you calling in from today?
Akshay Agrawal:	00:36	I'm in the Bay Area in Redwood City. Thanks for having me, Jon.
Jon Krohn:	00:39	Classic tech founder, AI tech founder, in the Bay Area. What are the odds? And you actually, you were introduced to me by Shaun Johnson, who is an investor at AIX Ventures in San Francisco, and if people are interested in hearing an episode involving what investors are looking for in AI startups and where the opportunities are for AI startups, or anybody building AI solutions for enterprises, that's episode 895. It's a great episode. Shaun is amazing. I guess you spend time with him in person in San Francisco sometimes?
Akshay Agrawal:	01:16	Yeah, I do at the AIX office. I do spend time with him, and he is amazing. He definitely is. I've known him for how long now? A bit over a year, and I'll see him usually almost every week.
Jon Krohn:	01:29	Oh wow, really? From what I could see in the background, and people can see it in the video version of 895, it's a beautiful looking office.

Akshay Agrawal:	01:38	Yeah, the AIX office is very, very nice, up in San Francisco.
Jon Krohn:	01:42	Nice. And so the reason why you're there once a week is because AIX has invested in your startup, which you are co-founder and CEO of, and it's Marimo. And so I want to start off with that name, because our researcher Serg Masis dug up that marimo means bouncy play ball that grows on water. And so it's a species of ball-shaped green algae native to Japan. Does that relate in any way to your company name?
Akshay Agrawal:	02:13	Yes, sort of. I can give you two stories, okay? The outer story and the actual inner story. So the outer story is that Marimo is a next-generation computational notebook for Python, data and AI. And well, what are the previous generation ones, or what are related ones? So the one that most people know about is the Jupyter Notebook. Now the Jupyter, it's a very large sphere, a very large sphere. There's another notebook for a different programming language called Pluto.jl, but it was a smaller sphere. It's kind of cuter, some would say. And Marimo is the next-generation one for Python, even smaller sphere. Very adorable, very beloved, especially in Japanese culture, but also plant enthusiasts.
	03:07	And Marimo moss balls, one thing that's actually cool about them is that they're actually assembled from these tiny strands of moss that all get clumped together and they live at the bottom of lake beds, and they intermix and mingle, and together they're greater than the sum of their parts. And Marimo Notebooks, in ways that we might discuss, are similar. They're collections of cells that when taken together are greater than the sum of the parts. So that's the story, conceptually, of what the name is.
Jon Krohn:	03:41	Nice.

Akshay Agrawal: 03:41 Also-

Jon Krohn: 03:42 There's a lot to that. Yeah, nice. Give me the also.

Akshay Agrawal: 03:45 There's also an also, which is during the pandemic, my partner and I, we had a Marimo moss ball in our apartment, and we just really loved it and we were like, okay, this is the thing. We love this thing. Let's just call... Yeah. It just makes sense. It also abbreviates well. Import Marimo as MO, which is you need that two character abbreviation for Python libraries.

Jon Krohn: 04:09 I love that. That's a great story. And they are super cute. I wasn't aware of them before, but I looked at some. And so I'm going to make a special request to our video editor, Mario, to overlay some cute photos from the internet of marimo balls, so that at least the people watching on YouTube can see some of those as you were describing it. And people who are in the audio only format, you're just going to have to look it up. If you're driving, don't do it while you're driving. But when you get to a stoplight, check them out. They're super cute.

04:45 Nice. So your work at Marimo was inspired in part by reproducibility pain. So anybody who has used Jupyter Notebooks has experienced difficulties with reproducibility, and I have lots to go into around that later in the episode. But just as a starting point, that kind of reproducibility pain that you experienced as a PhD student at Stanford, especially with things like coauthors handing you broken notebooks, and model training often beginning without checkpoints, version control or recorded data providence; tell us about those pain points that you experienced as a PhD student and how you came up with your solution.

Akshay Agrawal: 05:27 Yeah, definitely. So for a bit of background, so I did my PhD in machine learning. It was 2018 to 2021, so I did it

at Stanford and I specialized in vector embeddings, high dimensionality vector dimensionality reduction. So finding structure in these high dimensional spaces. And so when you work on things like that, it's really important to be able to see your data while you work on it. So notebooks, and Jupyter Notebooks were super, super useful. Without that iterative experience, that kind of research would be way harder. So very grateful for having Jupyter Notebooks. But as I did that research, there were a few shortcomings, some of them having to do with reproducibility that came to the forefront as I worked with Jupyter Notebooks on an almost daily basis. And so when it comes to reproducibility and maintainability, there's a few different aspects.

06:32 So for one, the file format for Jupyter Notebooks, they are stored as these JSON blobs. And so you can't really reuse code across them easily without jumping through some hoops. So you end up just duplicating your notebooks for slightly different experiments. Whereas if you were writing software, you would just import functions from one module to another. But also this JSON blob made version control pretty hard, because you'll run a notebook and you might not even change any code, but maybe the image changes, or some metadata changes and you get a big git diff. So it's hard to know how your experiment code actually changed over time.

07:22 Another issue that would come up, and so for some uses of Jupyter... This is a feature that I'm about to describe, for some uses it is a feature. For the kinds of stuff where you're doing science or you're doing data engineering and you need things to be deterministic, it is not a feature, in my opinion at least. So in a Jupyter Notebook, you can run cells out of order. You can run cell number one, then cell number three, then cell number two, then cell number one again, and then maybe you forgot to run cell number three. Cell number three maybe depended on cell

number one, and you just totally forgot you never ran it. And the point is, now your program memory is in some strange sort of quantum state that you don't understand. And then you'll figure out four hours later, eight hours later, like, oh crap, none of my experiment results are valid.

08:17 And so that problem is broadly referred to as hidden state. And it just comes down to the fact that in a Jupyter Notebook, the notebook runtime doesn't know how your cells are related to each other. And so that was a big pain point. If you contrast that to an Excel, for example, if you update a value in a cell, everything recalculates and the world makes sense. And that doesn't happen by default in a Jupyter Notebook. And I'm talking a lot, but the last thing, and we can go into any of these, is package requirements. So when you're working with a notebook, you might forget to track what packages. If you use PyTorch, like this specific version of PyTorch, and this specific version of Transformers, you might just forget to put that down in a requirements.txt or a PyProject.toml, like researchers often did. But then that means when you pass your notebook to someone else, they can't get the same results that are serialized in it. And so then that's just another angle in which the reproducibility problem becomes front and center.

Jon Krohn: 09:28 So yeah, we're going to get into more detail on reproducibility, all of the issues over the course of this episode, all the pain points that probably many of our listeners have experienced, and the solutions that you have for them. So I've got some of the key adjectives here around Marimo from our research. So it's open source, it's reproducible, it's git-friendly, it's AI native, and it's a reactive notebook. So a few things to dig into here. The open source thing, for example, you guys are doing very well. 4 million downloads I understand already at the time of recording, which is very cool. It must be amazing

to see people just voluntarily being like, "This is a solution for me. I need this tool," something that you've spent all this time writing and developing and leading the product on. So that must be cool. But what I want to talk about is the word "reactive" here. And so the idea is that it's a better tool, Marimo is a better tool, because it nudges users towards better data science practice by being reactive in this way. Tell us more about that.

Akshay Agrawal: 10:30 Definitely. Nudges is a good word. One of our users described it as gentle parenting. So yeah, what reactive means, it's just a... What it means is that Marimo keeps the code on the page in sync with the outputs you see. So basically, say you have three cells, the first one says A = 1. The second says B = A + 1. And the third one is C = A + B. And then you output those, you output the variable in each. So you see in the first cell A = 1; B = A + 1 = 2; and then C = A + B. 1 + 2 = 3.

11:16 Now what reactive means is if you go back to the first cell and change the value of A from 1 to 2, and then you run that cell, so you change the value of A = 1 to 2, and then you run it, the next two cells will automatically run with the updated value of A. And so their outputs will update to, if I can do this in my head right, 3 and 4 or something. So they'll recalculate like a spreadsheet, like you're used to. And why does this matter? Why is this a good thing? Two things. So one is actually it just leads to way faster data exploration. Like instead having to manually change the value of a variable and then hunt for all the other cells that use that variable, Marimo will just automatically run them and you can just see really quickly in real time, what did the new outputs look like? So that's one.

12:09 But the other one, which is maybe more subtle, is the reproducibility aspect, which is... Because if you forget to run a cell that depends on a variable you just changed,

now you can't trust any of the outputs in your notebook and your experiment is borked. So that's what reactivity gives you. And one thing, I'll just pause, because if there's machine learning practitioners or AI folks in here, the next question that always comes up, which I'll anticipate is, "Oh, but some of my cells are going to start kicking off like a GPU training job, and I don't want that to start automatically. It's expensive, I can't interrupt it."

12:50 And so for those situations, Marimo does have a button that says, "make execution", we call it lazy. And so if you do that, and if you run a cell, Marimo will just mark the other ones that depend on that cell as stale, but it won't auto run them and it'll give you a button to run them. So you still get guarantees on the reproducibility in the state, but you can rest easy that you're not accidentally spending your OpenAI credits or whatever you're doing that's expensive.

Jon Krohn: 13:18 Nice. And so I guess that ties into how you've previously mentioned that Marimo's reactivity makes UI elements like sliders and plots substantially more useful than Jupyter Widgets. And I guess it's because of what you just described, where in the Jupyter Notebook you don't know necessarily... Any UI elements, any plots that you're outputting, any sliders you might have in the notebook, you don't know in the Jupyter Notebook whether those are being updated properly based on other changes that you've made. Is that where that reactivity becomes useful there or is it something else?

Akshay Agrawal: 13:54 Yeah, no, that's essentially it. So in a traditional style notebook, if you had a slider, first of all, getting that hooked up to even propagate its value back to Python is challenging, you have to have callbacks and stuff. And even if you do, the things aren't going to automatically run with the updated value of your slider. So it's like, okay, what was the point? Whereas in Marimo, you

import Mario as MO into your Marimo notebook, and now you have access to a bunch of UI elements such as sliders. So you could replace the variable AI set. Instead of it just being an integer, it can be a slider. And now as you scrub the slider with your mouse, the other cells are automatically running as well. And so that's like the interactivity feeds into reactivity to give you a truly dynamic experience that can really, really speed up your data exploration.

14:48 One concrete example from the PhD that really helped what I really wanted to achieve, and I think we achieved pretty early, was you can even output plots in Marimo where you can select a cluster of a scatterplot with your mouse, and then those points get sent automatically back to Python as a data frame and then for downstream analysis. So it really opens up new kinds of interactive experiences.

Jon Krohn: 15:14 This episode of Super Data Science is brought to you by the Dell AI Factory with NVIDIA, two trusted technology leaders united to deliver a comprehensive and secure AI solution. Dell Technologies and NVIDIA can help you leverage AI to drive innovation and achieve your business goals. The Dell AI Factory with NVIDIA is the industry's first and only end-to-end enterprise AI solution, designed to speed AI adoption by delivering integrated Dell and NVIDIA capabilities to accelerate your AI-powered use cases, integrate your data and workflows, and enable you to design your own AI journey for repeatable, scalable outcomes. Learn more at www.dell.com/SuperDataScience. That's dell.com/SuperDataScience.

16:01 Nice. That's really cool. I appreciate the tangible example, something that's always useful for our listeners, for sure. And another feature that seems really cool to me is that when you have the notebook maybe into a state that

you're happy with, you actually could hide the code cells and then all of a sudden you have a data app just ready to go. Is that right?

Akshay Agrawal: 16:26 Yeah, that's exactly right. So a Marimo notebook collapses the space between notebooks and data apps. Now all your notebooks can be just notebooks. They don't have to be anything more, but if you want to, yeah, you just push a button and then boom, you don't have just a notebook, you have a full-fledged data app, which by the way is actually pretty performant. Unlike Streamlit, in Marimo, you slide a slider, it's only going to run the things that depend on the slider. It's not going to rerun the whole notebook. And yeah, that data app, you can serve it on a server, you can even run it entirely in the browser with a technology called Web Assembly. So really easy to share these artifacts.

Jon Krohn: 17:05 Nice. I love that. And then, so it just occurred to me, this actually wasn't in our research, but this is an open source project, Marimo, but you have Venture Data, Sean's invested in your company, AIX and other investors are expecting a return, what's the commercial angle for you?

Akshay Agrawal: 17:25 So Marimo as the open source, what we're doing is we're providing the best programming environment for individuals to work with their own data. And you can still use it successfully at your own company or at a company. We have many people using Marimo in companies, very large as well as small, cutting edge startups. As far as commercialization goes, whereas Marimo is the best place for individuals to work with their own data, there's still a lot of gaps in order to work with data in your enterprise at scale, especially doing rapid experimentation with very large datasets. So there's a lot of unsolved problems there, and that's where we plan to commercialize.

Jon Krohn: 18:06 That sounds like a really great strategy, a great commercialization strategy, so people can get the full use of your tool as individuals. And so I think this is something that would often be called product-led growth, PLG, where you have users that are volunteering to try this product out, and in a situation like yours where you're getting millions of downloads, it's obviously working well, it's about as well as PLG goes. But then there's kinds of features that enterprises are looking for around security, for example, maybe collaboration, maybe file storage. There's all kinds of ways that it would be impossible to offer those things for free to people. There would be substantial costs to the creator of that kind of functionality. And so it makes perfect sense to then be offering those as a commercial offering.

18:59 And it's something that we've seen, going back a couple of decades, tools like RStudio, they followed a similar kind of trajectory. They become the default IDE for, at that time, using R, which was very popular for data science, data analytics some years ago. And it still is. It still gets used today, but Python is now the lingua franca in data science, and it's great to see Marimo developing a great environment, and I love how you can get so quickly from a notebook to a working data app. That's something cool that I haven't seen. I haven't seen it done exactly like that before. I think that could be great.

19:46 So onto my next question. You've talked about a vision where you go from, as we've been talking about this whole episode, where you go from these error-prone JSON-based scratch pads that Jupyter Notebooks are, into a full stack developer platform, where exploration and deployment converge, just as in having a data app there ready to go instantly. So when tools collapse the boundary between notebook and application, that seems to change not just workflows, but also how roles work. So as this line blurs, what kinds of new responsibilities and

skills should our listeners adapt? Or does this just mean that they don't have to develop engineering skills? So how does it change things for our listeners who are maybe developing data applications, developing models for deployment? And then after you've thought about how it changes things for an individual, how does it change things for an organization in terms of research, engineering, operations, something like Marimo seems to break down a lot of silos.

Akshay Agrawal: 20:58 Yeah, that's a really good question. I think it's an astute observation. So in terms of the individual, like the practitioner, data scientist or even ML engineer, AI engineer, data engineer, I think the way that I think about it is that Marimo gives them new capabilities to make their work just far more useful in the organization. And that gets into your second question, too: how does that change things within an organization? So the data app is one good example. Previously you may have done your experiment in a notebook and then you're like, "Okay, where's the front end engineer who can help me actually build an application around this to make my work actionable?" At best, you might try to reach for something like Streamlit, but you would hit performance bottlenecks there rather quickly. It's like with Marimo, there's no migration phase, like your notebook. Should you want it to, just change a couple of variables to sliders or dropdowns or tables or whatever you need. And then all of a sudden you have a data app that you can then share with your team, but also to your CEO.

22:05 So actually in a number of companies that are using us, CEOs are using Marimo notebooks that their data scientists made for them. Sometimes the CEOs are actually making their own Marimo notebooks to run their own operations, just because the barrier entry is so low and it's even lower once you consider LLM integrations that we have, just like vibe code your way through a

pretty simple tool that you can make. So that is one way that I think Marimo gives the data scientist new capabilities and also brings new people into the fold. There is another way, and it has to do with what you mentioned of Marimo notebooks being stored as Python files, because actually every Marimo notebook is an executable script. It's just a Python file. You can go to the command line, say Python, mynotebook.py, even pass a command line arguments. So from your notebook, now you actually have a workflow that you can run as a pipeline or as a Cron. And so that's yet another way, by reducing that friction, we've now made you hopefully a lot more productive.

23:13 And also, there's a lot of alsos, but because it's a Python file, you can actually say from my notebook, import my function or import my class. And when you do that, that means that... There's famously, for many years now, people have talked about the notebook to production handoff, the researcher to engineer handoff, that makes that handoff a lot less difficult, a lot more streamlined. Because as nudges you to write better code, and it gives you these reusable functions, you can actually give your notebook to someone and they can import it and just use the logic. You could even write tests for your Marimo notebook. Marimo works with Pytest, so that's another way that it makes notebooks actually more useful in the engineering context as well.

Jon Krohn: 24:05 Nice. Probably during that part of the video version of this, we would've had the camera all on you, Akshay, but I was nodding my head aggressively throughout that whole time because that kind of functionality that you just described there is so useful. That would be an amazing functionality in terms of being able to call my notebooks from the command line or be able to pull in classes into other programs that I have running, or other scripts that I'm developing. That is super cool. And it's

pretty amazing because, well, maybe it's just because it flows naturally from topic to topic, but our next question that I had for you was on exactly that.

24:49 So let's get to what I had right after that, which was around AI assistance. So this is something that a lot of our listeners, including myself, are huge fans of Cursor as an IDE, and part of that is because LLMs make it very easy to write code. All of a sudden whole functions are just appearing before our eyes. So it's kind of like an autocomplete on steroids if people have the experience. For people who haven't used Cursor, it's similar to what something like Gmail does today or even a Google search where you get the next few words suggested, but it ends up sometimes it's like whole function appears in front of you in Cursor. And so it sounds like Marimo also has a kind of AI assistant involved. Tell us about it.

Akshay Agrawal: 25:45 Yeah, definitely. So in the Marimo editor, Marimo as a notebook works like most notebooks. You see something in your browser Python cells where you can type code in, you see outputs. And also there's a few different points, four, I think, at which AI is integrated into our editor. So the most prominent one you'll see is a button that says Generate with AI. And so that's what it sounds like, so you can generate Python code, or SQL code, I should mention. So Marimo has native support for SQL, and you can use essentially whatever AI backend you like. So you can connect to OpenAI, Anthropic, Gemini, you can also use Ollama to run local models. But the point is, what makes a notebook a unique experience, and makes our notebook a unique experience for using an LLM, is that while you actually have data in memory, and that's really different from traditional software engineering where it's just text files, here you have a running notebook session, you have data frames in memory, you have tables in memory.

	26:52	So what you can do is you can say something like, "Hey, using my data frame," and you tag your data frame, "using @df, do a group by the country column and aggregate over all numerical columns.' And then what Marimo will do behind the scenes, it'll pull the data frame from memory, look at the schema, get some sample values, and pass that in as part of your prompt, so that when you get the generated code, it actually has the column names in it and it'll actually work. And that's something subtle, but you won't be able to get that from just using Cursor on the text files. And that alone is a huge productivity boost, I think.
Jon Krohn:	27:34	Nice. I'm so glad that you were able to... I feel kind of bad, almost, when someone like you, startup founder, comes on the show and is talking about their product, and I'm like, how about this other product?
Akshay Agrawal:	27:47	No, no, no, no.
Jon Krohn:	27:48	But Cursor is something that a lot of people have experience using, and so thank you for that explanation of how it works in Marimo, and also the kinds of things you can do. It sounds like you've been thinking about ways that data scientists in particular, or AI ML engineers, in particular data analysts, people who are used to the Jupyter environment, how they would benefit most from an AI assistant, which adds kinds of features like you just described there around data frames being populated like you'd expect. So, super cool. And that actually is a great segue to my next question, which is around, you've previously said that there was a joy to rethinking a system from first principles as you set about creating Marimo. So you get to have an opinion on how the tool should be designed. And so it sounds like, if our research is correct, you began Marimo with a 2,500 word design document. So yeah, tell us about that experience

of coming up with this opinionated tool from scratch, what that was like.

Akshay Agrawal: 29:00 Oh yeah, it was a ton of fun. So this was right after I finished my PhD, so 2022. At that point, like we talked about, I used notebooks a ton during my PhD. There were still some gaps. And so what I got the opportunity to do is that I had some free space and I got to just study the landscape, got to really dig into, well, what do people like about notebooks? Interactivity, seeing your outputs live, and also what are people doing in other ecosystems? What kind of innovations, what kind of ideas are people playing with, what are they innovating on? And so that process took me through... I did research basically, and that's when I discovered the Pluto notebook for the Julia programming link, which it felt like something out of the future to me, because it has these concepts. So Pluto predates Marimo, and it's a big source of inspiration.

29:58 It has reactive execution, it has UI elements, it's stored as a Julia file. It doesn't have the web app thing, but I looked at that and I'm like, well, wow, that's really close to a data app and we can build that functionality in. And so that was a huge inspiration for me. Python has its own challenges of making something like that work. So then the next step was writing a design doc, like, well, how do you... It's like you're threading this needle. How do you design a batteries included experience where you have a notebook, it doubles as an app, and it triples as a script or as a module. And what I didn't want was a bunch of kluges to... It had to feel seamless and it had to feel frictionless. And that's why there was this design iteration. And I was lucky, though, because I wasn't designing in a vacuum.

30:49 So it's part of what we didn't chat about, so before we got funding from AIX Ventures, Marimo was built with close input from researchers at Stanford SLAC National

Laboratory, Department of Energy Laboratory, as well as with some industry researchers at a couple of startups. And so I would have ideas and throw them at them and they would give feedback, and you could see some of these early design documents, and some of the ideas were just totally horrible. But I got to work with them to refine them and make them a lot better. And I think that was a really special time.

Jon Krohn: 31:27 Curious about Trainium2, the latest AI chip purpose built by AWS for large scale training and inference? Each Trainium2 instance packs a punch, with 20.8 petaflops of compute power. But here's where things get really exciting. The new Trainium2 Ultra Servers combine 64 chips to deliver a massive 83 petaflops in a single node. These Trainium2 instances deliver 30 to 40% better price performance relative to GPU alternatives. Major players in AI like Anthropic and Databricks, along with innovative startups like Poolside, have teamed up with AWS to power their next gen AI projects on Trainium2. Want to see what Trainium2 can do for your AI workloads? Check out the links in the show notes. Now back to the show.

32:15 Nice. That does sound like a great experience and hopefully something that's inspiring for any of our listeners out there that are thinking about building a product. It sounds like you had a great way of getting going on that. Thinking about the best parts, the worst parts of the development experience for data scientists. You've put a great product together, it's really cool. Something that we talked about earlier, right at the onset of the episode and that would've come up, I'm sure, as you were developing the product in that Word document, is this reproducibility issue. So the Jupyter Notebook has been a very standard tool for a decade now for data scientists, but you've cited a study previously showing that only 4% of Jupyter Notebooks reproduce their original results due to issues that you call the hidden

state trap, among other kinds of problems. Do you want to dig into these kinds of problems with reproducibility that Jupyter Notebooks have and how Marimo resolves them?

Akshay Agrawal: 33:18 Definitely. So just to caveat this discussion, I think every notebook does have its place. And with Jupyter Notebooks, I think they are well suited to REPL workflows where maybe you're not shooting for reproducibility, but you're quickly really hacking things. The issue is, though, then, I think because it's such a great environment for interactive coding, people started using them and really relying on them for use cases where reproducibility is paramount, like data engineering, like science and like machine learning. And so the question was dig into some of those reproducibility issues and how Marimo solves them. So two main classes. So one is the issue of hidden state where in a Jupyter Notebook you run a cell, Jupyter's not going to run the cells that depend on it, like the cells that use its variables. And it's up to you to remember all the complicated dependencies that might flow through your notebook, and you're going to forget some. I often did. And so now your notebook is in some weird inconsistent state. So that's one issue.

34:24 To make that really pronounced, so in a traditional notebook like Jupyter, say you have a cell that, it's got a bunch of code in it, and maybe one thing that that cell does is create some PyTorch model class, instantiate some class. And then say you delete that cell, and then you continue coding elsewhere in the notebook. You deleted that cell, but say you didn't realize that that was the cell that defined the model class, and you really wanted that model class around. But in Jupyter, if you delete that cell, that model is still in memory for the time being. And so your rest of your notebook will work as you want it to work. But you come back the next day and you run the notebook from scratch, nothing works, and you're

like, what happened? And it's like, oh crap, I deleted the cell that defined the model and I really needed that.

35:14 So in Marimo, if you do that, if you delete the cell that defines your model variable, it'll tell you. It'll remove the model from memory and it'll invalidate the other cells. It'll be like, "Yo, that variable is no longer around. You can't do this anymore." And so it'll catch bugs immediately when you introduce them. So that's one big way. And the second big way is package management. And so Marimo has a special opt-in package manager, which I think is really neat. So basically if you start Marimo in a mode that we call sandbox, from the command line or however, every time you install a package (Marimo has a very nice, slick package installation UI), we'll save the package that you installed and the version as a comment or an entry in the notebook file itself.

36:06 And now when you come back to the notebook a second time and you do run it, Marimo will create an isolated virtual environment for you that has just those packages. So that means that you can just send a single notebook file around and people can just run it without even thinking about what packages they need to install, which makes them more reproducible, but also just a lot more portable. It's really easy to just create these single standalone tools and share them.

Jon Krohn: 36:32 I had more aggressive head nodding there while you were describing all that. I'm going to have to start using Marimo myself. Very, very cool. So something else that I think is really cool about functionality that we've learned that I want to... It's something that's so visual and so easy to understand. It's something that would feel to me like magic, as somebody who has really only used Jupyter Notebooks before for this kind of script development. So it sounds like it's possible, correct me if I'm wrong, that because of the way that you've thought about data

analysts, data scientist, data people's experience from the ground up with developing this tool, it allows you to do things like highlight, select the data with your mouse in a scatter plot, and then get it back as a data frame. Is that right?

Akshay Agrawal: 37:25 Yeah, that's right. Yeah, that's an example.

Jon Krohn: 37:27 That's crazy.

Akshay Agrawal: 37:29 Yeah, it feels really nice. And I really needed this during my PhD and I didn't have it. So it really enables really, really tangible data exploration, data analysis workflows, that were really, really hard, if not impossible, to do before.

Jon Krohn: 37:48 And so how do you decide, when you're doing product development, how do you decide to put something on your roadmap at all, or to then prioritize highly a feature like this where... Yeah, how do you decide which of these kinds of force multipliers to include in your product?

Akshay Agrawal: 38:08 There's two angles, especially earlier on when we didn't have that much feedback and it was me and my co-founder just developing and jamming. With our built-up experience of seeing, okay, here are some issues that we've hit with working with traditional notebooks, and we know a lot of others have, we have strong opinions and we think this will land and we're just going to trust our gut. So there was a lot of gut trusting in the beginning, and there still is. Now with the product being more mature, there's still gut trusting for big, new features that we're working on, but we have a big community, too, and so we can chat with them. And actually they come to us, they're very vocal about what features they want and they don't want.

	38:54	And I think the way we think about it is, is this something that enables a broad class of users, so like data scientist, data engineer, ML engineer, AI engineer, to be far more productive than they would have been otherwise in their previous tool of choice. And I think that that is something... We call them big rocks. What are the thing that really moves the needle for folks? And it needs to move the needle a lot and it needs to move the needle for a lot of people. Of course, we care a lot about craft and design and visual design and usability. And so those things are just always top of mind and they're ongoing, but for the big, new features, they have to be big rocks, how we think about it.
Jon Krohn:	39:40	Nice. I like that. Big rocks. And so you talked there about community and how you can leverage them. So open source projects often promise community, but not every project earns it. So what do you think separates tools that spark tons of devotion, like the Pluto project that you were describing there, and Julia, what do you think separates those from open source communities that fade? And what are you going to do to ensure that you're in the first category?
Akshay Agrawal:	40:11	Yeah, it's a good question. I think, honestly it sounds basic, but a lot of it comes from the maintainers just being really kind and open. If you want community, you can't just say, I want community, but there are people out there who are excited and you need to encourage them to contribute and be really, really vocal about how much you appreciate your community and also help them make PRs if they want to, be super responsive on your Discord. We try to respond really quickly to issues and especially bug reports. One of the feedback we usually get from our community who file issues is that they're shocked by how quickly we fix their bugs. If someone files a bug, you triage it and fix it and ship a release the same day, you won a supporter for life.

Jon Krohn:	41:13	Oh, wow.
Akshay Agrawal:	41:14	Yeah, that's something that we've noticed. Something we've also noticed from other popular Python projects, like from Charlie Marsh, Astral's UV project. And I think community's many levels. Not everyone has to ship code to your project, although many people may want to, but just encourage all kinds of engagement, and just make it a fun place for people to hang out and learn.
Jon Krohn:	41:39	Nice. If we have listeners who would like to contribute to the project, how should they get started?
Akshay Agrawal:	41:43	There's a few ways, and it depends on how you want to contribute. So if you want to contribute code, you can check out our GitHub issues, and some of the issues are tagged as good first issue. And these are great places for new contributors to get their feet wet and learn what the code is like. And some of them are even improved documentation. Just by the way, just generically a really good way to start contributing code to a project; improve the documentation. Other ways you can contribute is you can just file a feature request or a bug report. We have a little checkbox saying, "Are you willing to submit a PR for this?" And if you are, and if it aligns with our roadmap, then we'll work with you. And then more generally, we have a Discord, so you can get the Discord link if you go to marimo.io/discord. And there we have lots of free flowing conversation and I think that there's a lot of good touchpoints to get involved.
Jon Krohn:	42:43	Fantastic. And what if this is a listener's very first... What if they've never contributed to open source before? Where would you recommend they get started? How should they get their feet wet with open source development?
Akshay Agrawal:	42:58	So actually we do have, I think, a number of contributors we have made their first ever contribution with us. For

that, I'd say you see a typo in our docs, something like that, I think that's a really good way to contribute. The docs ones are, I think, I could be wrong, but I think if you're just making a simple change, you might be able to click edit in the repo itself online on github.com. It'll create your fork for you and simplify that process. But yeah, make a docs change. We have a contributing .MD guide in the GitHub repo that tells you about, oh, you'll need to make a pull request. What's a pull request? And walk you through that workflow.

Jon Krohn: 43:46 Build the future of multi-agent software with AGNTCY. The AGNTCY is an open source collective building the internet of agents. It's a collaboration layer where AI agents can discover, connect, and work across frameworks. For developers, this means standardized agent discovery tools, seamless protocols for inter-agent communication, and modular components to compose and scale multi-agent workflows. Join CrewAI, LangChain, LlamaIndex, browser-based Cisco, and dozens more. The AGNTCY is dropping code, specs and services, no strings attached. Built with other engineers who care about high quality multi-agent software. Visit AGNTCY.org and add your support. That's AGNTCY.org.

44:33 Nice. And then that brings me to maybe one of my last technical questions for you, which is, this is completely beyond Marimo, and so this is going back to the research that you've done, you focused on machine learning and optimization, and as an engineer you've contributed to several open source projects from the deep learning framework TensorFlow, that probably most of our listeners are familiar with, to the Vector Embeddings Computation Library. You're going to have to tell me how people pronounce this, but it's PyMDE?

Akshay Agrawal: 45:06 Perfect.

Jon Krohn: 45:07 Oh, yeah. So it's P-Y and then the letters M, D and E. And I'll have a link to that in the show notes. And so that Vector Embeddings Computation Library, PyMDE, as well as there's a convex optimization parser, which is also just letters: CVXPY?

Akshay Agrawal: 45:30 CVXPY, yeah.

Jon Krohn: 45:34 CVXPY, that makes sense. And so half of your published research relates to convex optimization. For our listeners unfamiliar with the topic, could you expand on how convex optimization differs from machine learning and the kinds of questions that they can answer; precision interpretability, scalability? Tell us about convex optimization.

Akshay Agrawal: 45:57 So mathematical optimization in general is like, you have some variables that you're trying to make an assignment of values to. So for example, say we're trying to choose a good stock portfolio, the variables is how much to invest in each stock. And then you need something that tells you whether or not your assignment to variables is good. So that's a mathematical function. It's an objective function that says, "Hey, what do I predict my return will be if I make this investment?" And maybe it trades off. Maybe it factors in risk into it as well. And then you have some constraints like, "Well, I can't invest more money than I have, and maybe I'm not allowed to short stocks." So you have some constraints. And mathematical optimization is then the process of finding an optimal assignment of values to the variables to minimize the cost or maximize the reward while satisfying constraints. That's mathematical optimization.

47:02 Convex optimization is just the subset of those problems we know we can solve super efficiently, super reliably, provably, and the use cases are somewhat different. So in some sense, machine learning, especially classical

machine learning, logistic regression, SVMs, all these are actually, under the hood they're using convex optimization techniques to fit the models. In terms of use cases out in the wild, what are people using CVXPY for today? Well, one huge one is financial portfolio construction. So many billions of dollars daily are allocated through portfolio optimization problems, which are solved with convex optimization problems. Energy management, there's a lot of usage there as well. Real-time control, like controlling a vehicle, landing a rocket, like SpaceX uses convex optimization to land, I think it's like the Falcon or something using software developed by our lab. And so these are cases where you can model the world and you have some understandable constraints and you can really exploit the structure.

48:13 Machine learning, sometimes there's typically not many constraints are that kind of implicit. You don't really have as good of an understanding of, well, what is the model doing? And also you're just trying to find a solution that's kind of good enough, that you know you're going to test it out in the wild on unseen examples. And so the use case of finding the optimal assignment, it doesn't even necessarily always make as much sense. That said, there's a lot of overlap. So the Vector Embeddings Library that you mentioned, for that to fit these embeddings using a GPU, we used, even though the problem was in the machine learning domain, we used techniques from convex optimization to solve it really, really efficiently.

Jon Krohn: 49:00 Nice. That was a beautiful explanation, as actually all of your explanations today. You do a great job of taking complex concepts and making them seem really approachable. And it isn't just the language that you use; you also have this really accessible tone. You make everything just seem light and relaxed. I really like that. It's been a joy interviewing you. Actually on the note of you making things feel so accessible, on your personal

webpage, so AkshayAgrawal.com, which I'll have a link to in the show notes, it says that your goal is to make machine learning and math accessible and actionable. What does actionable mean in this context? What's the biggest gap today between mathematical tools and being able to take action with those tools?

Akshay Agrawal: 49:55 Oh, I love that question. When you're in school or you're taking a course, especially a traditional course, ie most of them, they're teaching you, okay, this is logistic regression, and then you write down the optimality conditions by hand. You're computing some gradients and you're like, okay, great. I did a bunch of homework on my paper and I submitted a P-set. That's fine, and that's good. I've done a lot of that in my life. But to make something actionable and to make... I think what's really cool about our field is that all that math, you can do real things with it.

50:38 And that's why PyTorch and TensorFlow and JAX and PyMDE and CVXPY all exist. It's about using concepts from math in order to affect real change in the world. And so that's been the theme of the kind of projects that I've chosen to work on from TensorFlow to CVXPY, PyMDE and Marimo. Even though there is no math necessarily in the Marimo code base, what it does is give you a really, really tangible, interactive environment to work with your data. And so it's making your data actionable. It's actually useful, and you can actually run your notebook as a script or share it as an app or reuse the code in it. And so that theme is what resonates to me.

Jon Krohn: 51:28 For sure. You're building a platform to make the change that you want to see in the world. I've loved everything you've said about the Marimo product today and I'm so glad that Shaun introduced us. It's been a great episode. Before I let you go, Akshay, I always ask my guests for a book recommendation, and actually I can see, and our

YouTube viewers can see, a huge bookshelf of books behind you. So what have you got for us?

Akshay Agrawal: 51:52 I can recommend a couple. One that I most recently read that really stuck with me, as well as one that I'm currently reading. So I most recently read American Prometheus, which is the biography of Robert Oppenheimer. And so like many people, I watched Oppenheimer. Like, I want to learn more and I want to learn a lot more. So I read the book and it was really fascinating, because it actually talks about not just the events that happened, and not just the witch hunt of Oppenheimer that happened after the war, but also what is the social process of doing science and what are the social factors that influenced someone who is as much of a genius as Oppenheimer was to make certain decisions that led him to develop the atomic bomb? And I thought just having that broader context was super interesting. So that's one.

52:55 Another one that I'm currently reading that I would recommend because it's already making an impact on my life, is The Design of Everyday Things. So a classic in just product design. It's taken me a long time to read it because I'll go two pages, I'm like, oh man, I need to go and improve something in Marimo. And so if you want something, if you work in design or just like observing the world, I think that that's a good read. It's already made our product better.

Jon Krohn: 53:23 Fantastic. Two great and very different recommendations for us. Love it. And then for people, our listeners like me who have really enjoyed this conversation with you today, what's the best way to follow your work or connect with you going forward so that we can continue to get your thoughts after the episode?

Akshay Agrawal: 53:41 Definitely. So I am on the major social media platforms. So me personally, on X, my handle is Akshay K. Agrawal. I sometimes post under the official Marimo channel, which is Marimo_IO. I'm also on Bluesky as well. Those are the best ways to follow me personally. And then Marimo also has various social channels. So we're on YouTube, we're on Bluesky, we're on Discord, and we have a newsletter, marmo.io/newsletter if you want to subscribe, which I personally write once a month.

Jon Krohn: 54:24 Nice. Love that. Thanks so much for taking the time out of your busy founder schedule with us. Running an early stage tech startup like this must be exhilarating, but also very time-consuming. So it means a lot to me and to our listeners that you took that time out. And thanks for providing us with such a great episode.

Akshay Agrawal: 54:43 Thanks, Jon. Really appreciate it. It was a blast.

Jon Krohn: 54:49 I hope you enjoyed that episode and I hope you liked the shorter intro to today's episode. Reach out to me on LinkedIn with a DM or a comment if you weren't happy or if you have any other ideas how we can improve the intro or any other part of the show, really. Always love to hear from you. We assume that if you listened to the entire episode like you have today, that you'd probably still like the full outro that we usually do. So here you go. In today's episode, Akshay Agrawal covered the core reproducibility problems with Jupyter notebooks, such as hidden state traps, where cells can run out of order, JSON file formats that break version control, and missing package dependency tracking that makes notebooks impossible to share reliably. He talked about how Marimo's reactive execution model, that automatically runs dependent cells when variables change, ensures your notebook state always matches what's displayed on screen.

55:40 We talked about advanced interactivity features in Marimo, including UI elements like sliders that trigger real-time updates, the ability to select data points directly from plots and receive them as data frames, and instantly, an instant conversion from notebooks to deployable data apps. That's cool. Akshay also talked about the technical innovations that make Marimo notebooks stored as pure Python files, enable command line execution, function imports, and seamless integration with existing development workflows. And finally, we talked about how you can get involved with open source projects like these Marimo notebooks if you'd like to as well. You can get all the show notes including the transcript for this episode, the video of recording, any materials mentioned on the show, the URLs for Akshay's social media profiles, as well as my own at SuperDataScience.com/911. Thanks to everyone on the SuperDataScience podcast team, our podcast manager, Sonja Brajovic, our media editor, Mario Pombo, our partnerships team, which is Nathan Daly and Natalie Ziajski, our researcher, Serg Masís , writer, Dr. Zara Karschay, and our founder, Kirill Eremenko.

56:45 Thanks to all of them for producing another excellent episode for us today. For enabling that super team to create this free podcast for you, we are oh so grateful to our sponsors. You can support this show by checking out our sponsor's links, which are in the show notes. And if you're interested in sponsoring an episode yourself, you can get the details on how by making your way to Johnkrohn.com/podcast. Otherwise, please help us out by sharing the podcast, sharing this episode with someone who would enjoy this episode. Review the episode on your favorite podcasting app or on YouTube or wherever you watch it or listen to it, subscribe obviously if you're not already subscriber. But most importantly, I hope you'll just keep on tuning in. I'm so grateful to have you listening and 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.