SDS PODCAST EPISODE 765: NUMPY, SCIPY AND THE ECONOMICS OF OPEN-SOURCE, WITH DR. TRAVIS OLIPHANT

Show Notes: http://www.superdatascience.com/765

Jon Krohn: 00:00:00 This is episode number 765 with Dr. Travis Oliphant, the creator of NumPy and SciPy. Today's episode is brought to you by the DataConnect Conference, by Data Universe, the out-of-this-world data conference, and by CloudWolf, the Cloud Skills platform.

00:00:22 Welcome to the Super Data Science Podcast, the most listened-to podcast in the data science industry. Each week we bring you inspiring people and ideas to help you build a successful career in data science. I'm your host, Jon Krohn. Thanks for joining me today, and now let's make the complex simple.

00:00:53 Welcome back to the Super Data Science Podcast. Today's episode is with the absolutely iconic and absolutely brilliant Travis Oliphant. Travis created the ubiquitous NumPy and SciPy packages, which are downloaded over 8 million and 3 million times per day respectively for numeric operations and scientific computing in Python. He also founded Anaconda, the company behind the also ubiquitous Python package manager. He founded the massive PyData conferences in communities, as well as its associated nonprofit foundation NumFocus. He currently serves as the CEO of two firms, OpenTeams and Quansight, and he holds a PhD in biomedical engineering from the Mayo Clinic in Minnesota.

00:01:33 Today's episode will be primarily of interest to hands-on practitioners like data scientists, software developers, and machine learning engineers. In this episode, Travis details how his journey creating open-source software began, and how NumPy and SciPy grew to become the most popular foundational Python libraries for working with data. He talks about how he identifies commercial opportunities to support his vast open-source efforts and communities, how AI, particularly generative AI, is transforming open-source development, and where open-

source innovation is headed in the years to come. All right, you're ready for this jaw-dropping episode? Let's go.

00:02:12 Travis, welcome to the Super Data Science Podcast. I am beside myself that you're here with us today. It's amazing. Where are you calling in from today?

Travis Oliphant: 00:02:20 Jon, thanks. It's great to be here. I'm calling from Austin, Texas. I've lived here for 16 years, almost 17 now.

Jon Krohn: 00:02:27 My first few times to Austin, Texas have all been post-pandemic. I'm relatively new to Austin. I guess there's a lot of that happening, there's a lot of people who pandemic era ... But our company has an office in Austin. And the hotel that we were staying in was right across the street from a big Anaconda logo on a skyscraper.

Travis Oliphant: 00:02:48 Yes. I don't know if the Anaconda logo is still there, but it was really fun when that got up. I pointed it to all my kids and people that came in. There's a lot of virtual presence that I have, you can see the impact virtually from the stuff you've worked on. But there's something about the city you're in having a logo that's visible to everybody.

Jon Krohn: 00:03:07 Yeah, that's cool.

Travis Oliphant: 00:03:07 It was pretty cool. I think in the pandemic. I think Anaconda shut down that office space. I shut down office spaces I was using pre-pandemic, so now I'm totally virtual. We use occasional office spaces, basically hoteling, but a lot much reduced, our office space, which has its own ... That's a topic we could cover, but remote work versus in-person work and some of the trade-offs is very real.

Jon Krohn: 00:03:34 Yeah, yeah. Well, around 2021, that logo was still up there at least.

Travis Oliphant: 00:03:38 Awesome.

Jon Krohn: 00:03:39 Yeah. And then speaking of my work actually, that's kind of how we connected. So Ed Donner, who is a co-founder of my company, Nebula, our CTO and whom I've worked with for 10 years, and something that I frequently say when we do 360 annual reviews, something that I put in them, is that I hope that I'll be working with Ed until I die because he's just unbelievably intelligent, hardworking, amazing at personal relationships and following through on everything he says. And yeah, one of the smartest people I've ever met. So amazing to be working with Ed, so I really appreciate him connecting us. And so I guess somehow his time at J.P. Morgan connected with you there then.

Travis Oliphant: 00:04:21 Yes, no, I'm thrilled. I was thrilled to reconnect with Ed. I've known him for 15 years. We first met while I was a consultant at Enthought when I first came to Austin. I was an academic before then and came to Enthought to learn business. And came to Austin, worked at Enthought. We went to J.P. Morgan and basically we're helping them build out their use of Python in risk systems, and met Ed there because he was basically managing ... He's a young guy, or either he looks young, he either is young or looks young, one of the two, maybe both.

Jon Krohn: 00:04:53 Actually. It is wild how old he is. People never expect-

Travis Oliphant: 00:04:59 I could believe that, because I met him ... He was like the boss of the boss of the boss who actually pulled us in, and I was like, "Oh, this big meeting with the boss's boss and boss," and he looks like he was 25.

Jon Krohn: 00:05:11 Yeah. And he still does, it's wild.

Travis Oliphant: 00:05:13 He was super capable, super intelligent, definitely visionary. That's how I got to know him first. And so I knew him when he left J.P. Morgan and I had lots of connections there, so I was really thrilled to hear from him. And then I saw the LinkedIn post he did, which is phenomenal, just about explaining how to use LLMs effectively. It was actually one of the best documented examples and experiments of how to do it. It was really great. It really was great.

Jon Krohn: 00:05:42 Yes. So he's an unbelievable teacher, he's an unreal explainer of technical concepts, and so I'll be sure to link in the show notes to Ed's LLM posts. He did a really fun project where ... And so he's left step-by-step instructions and everything is open-source tooling, which obviously you'll appreciate, Travis. And so it provides you with everything you need to know to download your own text message history off of your phone and create an LLM that can mimic not only you, but Ed was able to ... Anybody that he shared at least 100 messages with, the LLM was able to effectively replicate those people.

Travis Oliphant: 00:06:23 That's wild. Yeah. Definitely that concept has people's attention. Either to, could we create my mimic, who could I fake out with a text message with me?

Jon Krohn: 00:06:36 Yeah, there's a funny ... I probably shouldn't be sharing this on-air, but I don't think anyone would care. So someone that Ed and I worked with for many years was a guy named Gareth Moody. And so Ed simulated a conversation with Gareth, and Gareth kept saying, "I'm running late," the Gareth bot, and that is spot on. That is ...

Travis Oliphant: 00:07:02 Yeah, I'm probably guilty of that too. My wife would say, "That's probably going to be your LLM." That's hilarious. I'm actually thinking that may be helpful. Obviously there's dark versions of all of this, but I'm more of an

optimist so any ... Yes, I know there are potential negative things that can happen but I'm a big optimist because I feel like, "Great, we'll just respond to those and make them better." What we need to do is create a bunch of really talented people, distribute the capability to as many people as possible, and that way, yeah, there's a few bad actors, but you overwhelm them with goodness [inaudible 00:07:37].

Jon Krohn: 00:07:36 Absolutely. And enough good people out there looking for ways to offset, to red team.

Travis Oliphant: 00:07:42 Yes.

Jon Krohn: 00:07:42 And to be able to offset the relatively small percentage of bad actors out there, for sure.

Travis Oliphant: 00:07:47 Exactly.

Jon Krohn: 00:07:48 So we are going to come back, in our second topic area that we discussed, we're going to come back and talk about bridging open-source with business, the kinds of things like your collaboration with J.P. Morgan that we just alluded to, and bringing them around to seeing the huge value in open-source. But first, let's get into the huge open-source achievements that you have in your past. You are best known as the creator of not only popular, but absolutely foundational Python libraries like NumPy, SciPy, and Numba, the Anaconda distribution, as we already alluded to with the logo on the skyscraper.

00:08:26 And so the Anaconda distribution is probably, if not the majority, the plurality of how most beginners, the plurality of beginners, start their Python journey. It makes it so easy to get all of the key packages that you need, probably most of our listeners use Anaconda at some point. And then whether they used Anaconda or not, surely any of our listeners who program in Python,

which is probably most of them, have used NumPy and they have used SciPy, for sure. Yeah, I guess there's so many ways that we could jump off from here, but I don't know if it would be awesome to hear how you got started with this, how you ended up contributing these absolutely foundational libraries, from the start.

Travis Oliphant: 00:09:11 Yeah. Well, I'll try to give a summary, a brief synopsis. I've actually given talks about this in several venues, and they're always just giving a slight window into the journey because it's a big journey. I would say I started with a need. I wanted Python to be useful for my work. I was a scientist at the Mayo Clinic and I was doing five-dimensional derivatives to try to do image processing on medical imaging data, in particular ultrasound and MRI data, and I was looking for tools to help me. I could code in C reasonably, I used MATLAB a lot. But I was running out of space, I needed to take more control. And I very, very much did not like the fact that if I wrote in a proprietary language that I was essentially giving somebody a burden, if I said, "Here's my code," and I love the scientific ethos of sharing progress. And if I shared progress with somebody, I didn't like the fact that I was actually telling them they had to have a license for some ... I was kind of giving them a bug, I was telling them ... That was my problem.

00:10:08 And so I was looking for something to be open-source. At the time there was lots of things out there, but nothing really very well-developed. Came across Python, said, "Well, this is a pretty good language." Used it for a year. I'd used Perl before that, but then I went back to the code I wrote in Perl and I had no idea what I said. I was completely lost. But Python I've had the opposite experience. Used it '97 for the first time, and then '98 I came back to it and said, "Wait, I understand what I did." And so that kind of said, "Well, I should do more here." And since then I've been basically trying to help make

Python better for scientists. And so I guess because, as a scientist myself, I had a better sense of what scientists would care about. And one of the things I noticed pretty quickly is that Python was a very generic computer programming language, so the benefit of Python is it's not just a scientific programming language, like R, IDL, or maybe MATLAB. As an open-source comparison, R is probably the closest, where R has been used by a lot of statisticians, but it didn't attract a lot of computer scientists, whereas Python did.

00:11:10 Python had a lot of system administration, computer science work, but the gap between the computer scientists and the scientist in Python is real, was real, is still real. There's a real issue there, that essentially I've been seeing that bridge for a long time and a few other key pieces. In fact, that's a great place where anybody who wants to really contribute to Python, who has a science background, it's a great place to be a community worker. Somebody on the ground, translating the needs to the people, and there's a few examples of people that have done that really well. I was one of the people that did that to some degree, but I was always driven by my need. So I started in 1998 essentially writing packages and extension modules for Python. One of the great things about Python is its syntax was accessible to non-programmers, so it didn't force you to learn all this arcane syntax to do anything. You could just write functions, so relatively straightforward loops. You could write classes, but you weren't forced to. They were available to you, but they weren't in your face, like you had to deal with them from the very beginning.

00:12:13 And then you could extend it really easily, relatively easily. I knew C so I could get at the root, and when I looked under the covers, I realized, "Oh, Python itself, it's self-extended." It's not like, "Okay, here's Python." Then the extension language is this extra thing that's bolted on

the side. Like no, the list object, the dictionary object, these are all extension modules essentially, and so I just made one. There was existing already a numeric extension, someone had created an array object. It was Jim Hugunin. I've learned since, I know Jim, he hasn't worked in the Python space for years, but he built this incredible array module called Numeric. He was a grad student at MIT at the time, and I was a grad student at the Mayo Clinic when this is all happening, so I just wanted to use it to write stuff. I was writing an MRI simulator, so I was doing things to solve my problems. I wanted to read data in DICOM format so I could process it. So I wrote something called NumPy I/O, which was a way to read data into an array object directly from a DICOM format. It enabled other things besides that.

00:13:15 I was doing an MRI simulator, and so I needed to solve the bloch equations, which is an ordinary differential equation solution, and so I needed an ordinary differential equation solver, and I didn't have one. This was when I found a bunch of FORTRAN libraries that already did that, and I just bound them to Python, kind of wrote the glue to make them work with Python. And just sharing that on the net, which I did just, put it out there, I said, "Here you go." Of course my sharing was pretty bad, I just said, "Here's a tarball with code," and a website that looked terrible that it was totally non-accessible to anybody, but a few people did, used it, downloaded it. A guy out of Estonia downloaded it, Pearu Peterson, and he gave me a better makefile. I still remember the day, he gave me a better makefile. But it was more complicated and it did more, and so I was like, "Oh, that's pretty cool. I got a contribution." And that kind of started to get me hooked on just this community collaboration.

00:14:10 I loved the mailing lists. The fact that I could ask questions and people a lot smarter than me would say things and I'd be like, "Oh, that's a really good idea. Oh,

that's cool, I can learn from that." And I kept just engaging that way as a grad student, all through ... Then 1999 was the year I really dug in and said, "Huh, we need more here." Basically that year I probably released six libraries that year, all different libraries for integration, for special functions, for just a bunch of stuff. Along that year, Pearu looked at it and said, "Why are you doing this manually? You're taking FORTRAN libraries and manually creating extension modules. Here you go. I'm a real computer scientist. I will write a tool to build that." I'm a practical guy who's learned a lot of stuff, but my initial reaction wasn't to build a tool, my reaction was to get something so I could get my job done. Pearu built something to help everybody solve the problem, so he wrote F2PY. And that just read the source code of FORTRAN, automatically built the extension module for you.

00:15:11 That interaction all happened in 1999, 2000, all while I was finishing my PhD. So by the time I got my PhD, I was knee-deep embedded in the Python community and had lots of people who are using these libraries, and they were really ... An Anaconda user today would like ... This is the early days. You did not have Anaconda, you had to make it yourself. In fact, I tell this story because it's so critical, it's got me on the path to Anaconda. It was a guy named Robert Kern, who was a high school student at the time, and eventually a colleague at Enthought. He wrote not a makefile, but a Windows distribution for SciPy. It was called Multipack at the time. So I had a tarball out there. If you're a Linux user, it was pretty easy to build, but if you're a Windows user, you're like, "I don't even have a compiler on my system. I can't use this." But he basically built the tool and released a binary so people would just download the executable. So he built an EXE file.

00:16:03 And that service had been useful over the years, basically from 2019 to 2012, when Anaconda got started, for a long

time. We had done something at Enthought too to make distribution easier, we were charging them for it. But you saw that need dramatically, and what I saw is an increase in users. The minute people could actually install it on Windows, 10x increase in usage, maybe. It was massive. So I reluctantly became a build person for Windows. It wasn't my favorite thing to do, but by 2001 I was tracking down build problems on Windows so that SciPy, which became the grouping of packages, we could release it. So SciPy started in really 1999, 1998, but became in fruition in 2001 as a partnership with a few other people, Eric Jones, Pearu and others. Now it's grown.

00:16:55 Now the success of SciPy, again, I've watched this repeatedly, and so I have some ideas about how do projects succeed. I've also watched them fail. So I've been a part of other projects that didn't go very far, or they failed ultimately. And so you'll learn as much from those two as the ones that succeed, but you end up with a kind of a, "Okay, here's a recipe for how do you know." And then part of it is things you know it to do, and then some of it's out of your control. Some of it you just have to observe and see if it happens, and if it does, you can continue. If it doesn't, you have to move on. But you've got to get other people involved. The reason SciPy succeeded was because it wasn't just, "Hey, I'm going to do all this work." It's, "I'm going to do work and then go engage with a bunch of people." Go to conferences, go find other participants to involve, figure out ways to grow the community. So that SciPy has succeeded today and has succeeded for the past, gosh, since at least 2013 probably, because of these hundreds of other participants, people who are making it work.

Jon Krohn: 00:17:51 2 days, 50+ presenters, unlimited opportunities to connect with a global data audience. The annual DataConnect Conference brings together industry leaders, technical experts and entrepreneurs to discuss the latest

00:18:41 This sounds quite similar to ... We had Gael Varoquaox on the show-

Travis Oliphant: 00:18:46 Yes, yes.

Jon Krohn: 00:18:47 ... in episode number 737, talking about scikit-learn, and it was very much that it's this community involvement. It's the back and forth both ways.

Travis Oliphant: 00:18:58 Yes. Nurturing the community becomes a really critical part of early stage open-source development, and a really early part. Scikit-learn is an interesting one actually, because I remember when scikit-learn got started because SciPy was trying to do too much. Essentially SciPy had machine learning libraries, it had stuff in it that was ... SciPy became this kind of ... It really was a distribution of Python masquerading as a single library, because the EXE file, just getting it all installed was a big deal, and so it was easier to bundle. Not unlike my conversation with Guido that took place years ago, where I asked him, "Well, so why don't you care about packaging? We don't have a good packaging story for Python. What's going on?" He's like, "Yeah, I don't care about packaging. If I want a library, I put it in the standard library. So it's like, "Well, that's cute if you ..." And SciPy had that same [inaudible 00:19:46] model, but

scikit-learn was ... The scikits were like, "Hey, we got to have a way for people to share maybe documentation, maybe patterns of development, maybe some collegiality."

00:19:57 But it's got to be controlled by different people, because that's when I really realized pretty quickly that people don't scale. You need lots of people involved, and so you got to incentivize those people too, and if basically ... This is definitely true that I've had to deal with, is yes, there's a lot of people who recognize me from things I've been involved with and I was very, very critical to be involved early, but then I can create a shadow. There have been people who've said this to me, and I won't name names, and I've also watched it in others, where they're like, "I don't really want to be in your shadow. I don't want to be the Travis-helper. I want to have my own brand. I want to have my own story." And they may not even articulate that to themselves like that, but I get that, I understand that. A key part of the reason open-source is valuable is because it does let developers have their brand, have their story, so there's got to be a way to make sure people get credit.

00:20:45 That's actually one of the passions I have, is how do I help people get credit for the contribution they make so it isn't just masked behind the one figurehead that pretends to take all the credit. I don't try to pretend to take all the credit. I really want to celebrate the successes of all the people that have made NumPy and SciPy possible. If I get any notoriety, the goal of that notoriety is to drive towards activities that give other people credit. That's the only reason it's valuable ultimately, is it can help consolidate a message so you're not dispersed, but how do I build systems so that other people ... I want to incentivize this open-source engagement and make it possible for people to build careers on it, that's been ... So it will lead probably to the other things I've been doing, but I'll try to

finish more quickly. I'm trying to give some backstory for what I'm doing now.

Jon Krohn: 00:21:33 Well, we'll definitely we'll get to that when we talk about OpenTeams later on. I also just want to very quickly, in case people don't know, you mentioned Guido there quickly, quite conversationally. And probably for many of our listeners, that is something that people just know. Guido van Rossum is the creator of Python, and so yeah, that's just-

Travis Oliphant: 00:21:52 Yes, yes. A wonderful individual. A lot of Python's success is because of him as a community builder. And I've had the fortune of meeting with him, interacting with him, particularly as we got SciPy off the ground, and particularly as we got Anaconda off the ground. It's been a few years since I've talked personally with him, probably about five, but he's a great guy. Really admire what he's done. And love what he's created, a lot of similarities. So SciPy got started that way. A lot of people don't realize SciPy came first. [inaudible 00:22:22].

Jon Krohn: 00:22:22 Yeah, I actually didn't know that.

Travis Oliphant: 00:22:24 SciPy came first. And I was building SciPy, released it in 2001, was shepherding it. Had graduate students at BYU working for me, developing further modules in SciPy. And I started at that point talking to the folks at the Hubble Space Telescope who were building additional features, they wanted more that wasn't available on Numeric. They realized there was a need to build a better array library. So they had started NumArray. And so there was Numeric and NumArray were emerging. By about 2004 there was two array libraries in the Python ecosystem, both being used. And particularly there was a library called ndimage, and as a student of medical imaging, ndimage had a morphology image processing system that I was jealous of. Like, "God, I want that in SciPy, it'd be great. I wish I

had that tool." And I had it, but it was in NumArray, it was based on NumArray.

00:23:18 And at the time, if I loaded my data, I remember I was loading data in a numeric arrays early with DICOM format. "Great, now I have it in memory, oh, but now I need a NumArray version." And so I would have to copy it over to NumArray. And so if you're talking about gigabytes of memory, that's really a non-starter. And so this is a problem. We can't have libraries built on top of different array objects that don't share data in common ways. So that was why NumPy exists, because I saw the problem. There weren't that many people who could do anything about it. I didn't know that I could do anything about it either. I was a professor of electrical computer engineering focused on electromagnetism and signal processing, who use these libraries and knew some C, but I wasn't a creator of languages and type systems. I understood the ScipY extension capability, sorry, the Python extension capability. I knew how to extend Python, but man, taking on something like writing NumPy was challenging.

00:24:13 But I did. I had a class fell through as a professor. And even though my graduate degree professor, sorry, my department chair when I applied for tenure two years earlier, had said, "Well, we like the progress you're making, but you're kind of doing too much open-source stuff. So probably got to think about maybe more papers, more grants, less open-source." So I promptly go and write more open-source with NumPy. Again, not because I wanted to stick it to my department chair, because I just tried to follow where the need was. And I saw this need in the community, I saw this opportunity to do something, and I just felt like, "Someone's got to do this." And you look around and eventually you go, "Well, who's going to do it?" I had already known enough about the community, I was somebody that knew enough. I said,

"Oh, I can try." And it was daunting, it was, and I wished I'd actually had me now to advise me then because ... because there's tons of stuff I didn't know. And it's embarrassing because it's in the code now, and it's like the community's just recovering from that, honestly.

00:25:14 But I guess I say that because look, desire is important and you do your best you can, but you're not going to be perfect. And that's what open-source can help you with is actually make it better. And it takes a little bit of just, it's okay, you don't have to ... A lot of young people I meet, they have this impression that the first thing they do has got to be perfect. And it really can stall development. It can really stall doing something. I would say the most important thing is be productive, do things, get things shipped, and then be open to feedback. And the best realization you're doing something valuable is feedback, is getting criticism. The worst is apathy. You know you're not succeeding if nobody cares. You are succeeding if you get people criticizing it even. So, awesome. This is something that I really like to get feedback.

Jon Krohn: 00:26:04 Yeah, that's a great-

Travis Oliphant: 00:26:05 But it's hard emotionally. It can be hard emotionally because it is emotional a bit. The more effort you put into something, it's your baby, you put it out there, you want people to like it. And what I realized, and people did like what I produced, but not because they were critical technically. In fact, I needed some of that technical criticism because ... So, I wrote NumPy in 2005, took about another year for it to stabilize, and then I presented it at the SciPy conference, which we'd started in 2001. So, I presented it at the 2006 SciPy conference. And Guido came to that conference, by the way, and he saw me demo NumPy, actually, and he saw me demo the type system. And I wished I would've understand at the time what I built was a Python 1 type system in NumPy. I

extended the Python 1 version of the type system. Wouldn't be till later that I recognized exactly what Guido had done in Python 2, to integrate 1 style classes with Python 2 style classes. And if I had just done that in NumPy, we'd be better off. So, I've known that for a long time.

00:27:08 Now, finally, NumPy 2.0 is starting to fix that problem. Sebastian, I forget his last name, Sebastian's been great. He's been doing some really good work on the type system in NumPy, and I'm super happy that we have some other people working on NumPy full-time, or at least part-time, because of Quansight Labs. So, that's what I did. I got NumPy out the door. It did not help my tenure application, despite lots of scientists liking it and appreciating the work to get NumPy, and I knew I'd succeeded with NumPy as soon as Matplotlib made a dependency on NumPy. Right? It was before that making a dependency on numerics, which was letting you pick between the two.

00:27:51 I'll say one more thing about NumPy that's really critical because we fixed it twice. I fixed the immediate problem of, oh, there's two array libraries, let's get one. But then what's the ultimate cause of the problem was a lack of shared data structure. It was a lack of a way to have data that could be shared between arrays. So, we created the buffer protocol in Python 2, and that's how I got to know the Python developers better is they made several contributions to the Python language itself. Some small, the biggest one was the buffer protocol, which is basically a way for a sequence protocol ... Python has these protocols which properly understood or actually would be better served as Meta-type extensions. This is something that I should talk to the Python community about honestly, because once I understand that, and it took me years to understand this, it wasn't until about seven

years ago that I finally realized, "Oh, this is how the whole thing could work."

00:28:42 But the buffer protocol was a way for things to share data to each other. And we're back to that actually, because when I woke up from my entrepreneurial activities and came back in 2018 to the NumPy world and what had happened and where we are go from here. I look up and I see a dozen array libraries in Python. We solved the problem of unifying them in NumPy in 2005, but really, we didn't have GPU support. We didn't have automatic differentiation. We didn't have a funding model to get those added. And so, when that came up in the deep learning spaces, Meta, Microsoft, Google, they spent millions of dollars, a lot of money basically rebuilding tools, so PyTorch... And also because they came from not Python land, they came from C++ libraries, and so forth. I've come to know some of the history of those libraries where they came from. Deep respect for the developers thereof, but I'm still eager to see cooperative communication because I think it could save us a lot of time by just being aware of what's out there. But now there's tons of array libraries, and so we're going to write another one. We didn't do it this time. We actually went and wrote data-apis.org. If you're interested, go to data-apis.org and that will show you what ... It's a standard we're building, the array standard.

00:30:06 Building off of the buffer protocol, which already allowed interoperability between arrays, so they can share the same data because at that point, if the interface is a little different, that's fine, as long as you don't have to copy the data back and forth. So anyway, that's the history of where SciPy and NumPy came from, but they've been very important to me, to support the scientific use of Python.

Jon Krohn: 00:30:30 This episode is brought to you by Data Universe, coming to New York's North Javits Center on April 10th and 11th.

I myself will be at Data Universe providing a hands-on Generative AI tutorial, but the conference has something for everyone: Data Universe brings IT ALL together, helping you find clarity in the chaos of today's data and AI revolution. Uncover the leading strategies for AI transformation and the cutting-edge technologies reshaping business and society today. Data professionals, businesspeople and ecosystem partners — regardless of where you're at in your journey — there's outstanding content and connections you won't want to miss out on at Data Universe. Learn more at datauniverse2024.com.

00:31:08 There's a few threads that you mentioned in there that remind me of my interview with Wes McKinney who created pandas.

Travis Oliphant: 00:31:15 Sure. Sure.

Jon Krohn: 00:31:15 So, we had him in episode number 523 and similarly, he created pandas because it was a need that he had, and similar to what you're describing with NumPy, if he could go back in time, he would do it all very differently.

Travis Oliphant: 00:31:30 Sure.

Jon Krohn: 00:31:31 And, like you, he is doing that. So, things like the Apache Arrow project that he now does is to deal with some of the key limitations that are now baked into pandas. And so yeah, it's interesting to see that parallel journey.

Travis Oliphant: 00:31:46 Yeah. So in fact, what I'd love to see, one of the challenges both of us have faced is that funding for that is not prevalent, right? There really isn't a space to go and say, "Oh, cool, you want to make NumPy better? Cool, here's a million bucks." Instead, you have to go create a company idea and then try to get funding for that company and then hopefully some of that money for the company can go back into another project.

Jon Krohn: 00:32:10 Yep. Another [inaudible 00:32:10].

Travis Oliphant: 00:32:11 But it's not really solid. Yes, there are now some grants you can get. The grant applications can be challenging because that's a whole other style. So, there's multiple parallels with Wes. And Wes and I are friends. We've talked for years. I like to think that actually the D type we created, that enabled record arrays, gave rise to the pandas movement, because it was almost a data frame. It hinted at a data frame. And they did, record arrays gave you this idea of, "Oh, I've got this cluster of data, they're all together." But the big issue was it was an array of records. Right? So, it's row-based basically. If you think of the orientation, it's row oriented because all the record is together. The NumPy model is you have a big block of multidimensional something, right? But there's a defined something that it is, you can define the number of bytes and so forth. So, that's an awesome model for many applications. But having to ... You want to add a column, right? You have to insert into the record and then remap everything, right? So, it doesn't really make it easy.

00:33:18 So, data frames, they're basically structures of arrays. You can imagine a structure with multiple arrays, and that effectively is one of the foundational elements of pandas. Then of course, the APIs you put on it are a little different. The kinds of processing steps you want to do, the joining, the compute you do with the data frames is also a little different. But yes, it was an interesting world to watch. In fact, having written NumPy and then having seen it start to be used, I had a lot of conversations with a lot of people, including Wes, including the R community. At one point in 2008, I talked to many leaders of the R community who basically were like, "Well, could we work together? Maybe we can merge our efforts somehow." Because they were suffering under the strain of a massive user community and very few developers, right? Which is, it was a challenge because it's like there's lots of

scientists who use stats, but there aren't that many developers who can develop infrastructure for [inaudible 00:34:14] thinking. We talked about that at length. I think there are things we could have done, but there was no money to support it.

00:34:21 One of my desires has been to try to figure out how to get the people with the budgets to talk to the right people, because very often they don't. They end up going and talking to the person nearest them who knows some stuff, but is usually completely unaware of this other activity that's been going on, and they don't pull in the right people. Fortunately, I'm really fortunate that PyTorch and Soumith, who's been leading the PyTorch effort for years at Facebook then Meta, we were able to build a relationship. And I reached out to both the PyTorch and the TensorFlow teams and people in those teams to try to build a relationship to see how do we connect these emerging capabilities with the existing SciPy ecosystem, and had a lot of success with Soumith, because of him, we've been able to establish some things.

00:35:08 And TensorFlow, we're still working on, they're great people over there, but they have different ... You look at the business infrastructure, and part of the reason is because Meta, Torch was emerging in something called FAIR, which is a research group, whereas TensorFlow was part of the cloud infrastructure, they had to use it internally. And there is actually work to create open-source interfaces in a company, to create something that can interface the open-source ecosystem and your company. And you have to be intentional about that, and not only intentional, but also make sure you're doing some good things.

00:35:44 So, I really like to promote people that are good dev rail people, that understand that, and try to help work with them. I look around at, what can I do to help right now?

And I'm always looking for where can I help? I like to think I could code and I can code a little bit, but a lot of people code much faster than I do. Mostly what I really do well is help architect. Like I said, if I could help me younger, I could have really helped me younger. If we could somehow get that time machine loop going.

Jon Krohn: 00:36:12 Yeah. And there would be no harmful consequences to that at all. Everything would work out perfectly.

Travis Oliphant: 00:36:16 No, of course not. No, no, no. It'd be totally fine. No.

Jon Krohn: 00:36:21 So, really quickly here, I can't remember now ... Oh, I think it was talking about how NumPy was distributed or I can't remember exactly what put me on this-

Travis Oliphant: 00:36:31 No, NumPy wasn't distributed, but it needed GPU, it didn't have GPU support, nor was it working on more than one node. That was another area. Yeah, that actually leads to the Anaconda story, right?

Jon Krohn: 00:36:41 Yeah.

Travis Oliphant: 00:36:41 Those problems there. And so, I think it leads into the business stories.

Jon Krohn: 00:36:45 So, right before we get to Anaconda, something that I just dug up as ... So yeah. Now again, I can't remember what caused me to do this, but I just thought to myself, how much is NumPy downloaded every day? And do you know offhand? So, looking at just-

Travis Oliphant: 00:36:59 I don't know.

Jon Krohn: 00:36:59 I can see easily from PyPi, so just from PyPi installs alone. So, it doesn't include Anaconda, which we're getting into. It's over 8 million a day.

Travis Oliphant: 00:37:08 Yeah. Yeah, that makes sense.

Jon Krohn:	00:37:10	It's wild.
Travis Oliphant:	00:37:12	Now, some of those are bots, of course.
Jon Krohn:	00:37:14	Yeah. It's probably most of it.
Travis Oliphant:	00:37:15	I would say most of them because it's in the build infrastructure, especially recently. What I'd like to see is if you look at the growth there, we saw some of the things Anaconda, but it is hard. It actually illustrates one of the challenges of open-source is telemetry or just statistics about usage. Because one of the things I learned later is that people want to fund stuff, but they want to know, "Well, what am I funding?" And that kind of information about who's installing what, where, could be helpful as people try to understand, "Okay. Where is the energy here?"
	00:37:46	So, I joked before, nobody sends Christmas cards. Nobody sends postcards. They don't tell you what's happening. You don't. A lot of people when I was early, early contributed to open-source. Many of the old guard of the people who are older, they were really nervous about getting overwhelmed with support questions. I remember this like, "Oh, we can't open-source that. I can't handle all the support requests I'll get." I'm like, "Okay, I hear your point." So, that hasn't happened. I would say the opposite has happened. Right? I don't get enough support questions. And then of course not right now, I'm not at the front lines of NumPy support. So, please don't inundate me with your support questions about NumPy. You can have them, and I have a sales team-
Jon Krohn:	00:38:29	You just said, "I don't get enough support questions." Now all of our listeners are going to be sending you-
Travis Oliphant:	00:38:29	I know. [inaudible 00:38:35].

Jon Krohn: 00:38:35 You're going to get postcards from all of our listeners.

Travis Oliphant: 00:38:36 I know. Well, postcards are great. Feel free. That's great. I mean, I actually really love meeting people. And please do it. I love it. LinkedIn messages or emails, "Hey, this saved me in grad school. I really love this." I mean, I love that. That's helpful. It's what I've wanted to do. That's actually the whole point. I do have a family, one thing some people know about me but isn't often known. I mean, I had three kids in grad school. By the time I was writing SciPy at all, I had three kids. Right? So, I was also a fan of, "Well, how do I fund all this stuff? How do I make this work? My wife and children are expecting me to provide for them." We made the choice in our family, my wife would stay home with the kids and she'd essentially focus on making our family and having our children be productive people. And a lot of work there, a lot of work there.

00:39:26 But I had to figure that out. It was just a necessity. It wasn't just a, "Oh, I hope I can figure this out." It's like, if I want to spend time in open-source, I got to have a job to do it. And so, I got to find them or I got to make them, or I got to create something that makes this work. So, that's why I spent a ton of time as a grad student besides just writing SciPy, and early on was reading economic literature, reading about how does economics work? What is this business of business? How does this make sense? I learned a bunch of stuff that were really impactful for me in trying to think about, "How am I going to do this in the future?" You have to understand that about me.

Jon Krohn: 00:40:06 What does your biomedical engineering department head think about all the economics books?

Travis Oliphant: 00:40:11 Yeah. I didn't really tell him too much about all that stuff I was reading. My graduate professor was pretty good. Jim Greenleaf out of Mayo Clinic, really awesome ultrasound dude. As long as I was making progress

towards papers and getting a thesis out eventually, and I only took four years to get my PhD thesis done, so I was fine. They were pretty hands off. He didn't get into what I was doing. So, all the open-source engagement, I built a cluster of Apples. I went around Mayo Clinic and gathered these old MacBooks, they stacked really well together, actually, these little Mac boxes. Put Yellow Dog Linux on them, hooked them up with PVM and made a little cluster to simulate MRI, and stuck it in his lab, and he was fine. I mean, sure, it was interesting.

00:40:58 But I learned the internet that way, actually. I learned how internet works and TCP/IP, and I didn't use MPI. I just wrote C++ libraries to talk between the computers. It's where I learned the problems of C++. I like C++, but I learned the problem of over-abstraction. It's very easy to write abstractions where you end up wasting time. Abstractions at the wrong level can be a complete problem for speed, and that's true today, that's still a problem today, which is why C++, while very good, can be a problem if it's over-relied on. So anyway, learned all that. All that, I'm pretending I know everything, but learned a ton doing those projects.

Jon Krohn: 00:41:39 Data science and machine learning jobs increasingly demand cloud skills with over 30% of job postings, listing cloud skills as a requirement today, and that percentage set to continue growing. Thankfully, Kirill and Hadelin who have taught machine learning to millions of students have now launched CloudWolf to efficiently provide you with the essential cloud computing skills. With CloudWolf, commit just 30 minutes a day for 30 days, and you can obtain your official AWS certification badge. Secure your career's future, join now at cloudwolf.com/sds for a whopping 30% membership discount. Again, that's cloudwolf.com/sds to start your cloud journey today.

00:42:18 Yeah, and sorry, I've taken you off the track several times now. So, we were going through the creation of SciPy, NumPy, and then we were just about to get to Anaconda, and I completely took you off track.

Travis Oliphant: 00:42:30 Totally fine. Anaconda came about as a combination of technical innovation and business desires, right? So, I left academia because they told me they were going to give me tenure later. They said, "Well, wait and reply in two years." So, they didn't actually say leave, because I mentioned before how my department director said, "Hey, write more papers and more grants." And I wrote more software. And so it was like, "Well, we're not quite ready to make an assessment of that yet, so please just come back in two years." And at that point, I was already really interested in figuring out how to make business work in open-source, and I didn't want to go through the process again. And so, I left academia and came to Austin, Texas to explore business and had the benefit of working with a friend at the time, Eric Jones, who had started Enthought. And he brought me in. And it was a consulting company that was also figuring out how to build businesses and open-source and make them work together. And so, we worked closely together for a while.

00:43:34 And then at the time, there were two problems I really wanted to tackle that we weren't going to tackle at Enthought at the time. Eventually maybe they were going to, but it wasn't fast enough. So, I wanted to tackle them faster. One was scaling NumPy. I saw already NumPy array processing at scale, saw that problem at JP Morgan, saw that problem at other companies, said, "Oh yeah, NumPy works on a single node. How would you do a NumPy at scale? What does that even look like?" pandas was just emerging at the time, knew a little bit about pandas in 2011. There's previous data frames before that, data array. We talked about labeled array. We talked about the data frame problem for the past couple of years

before that. So, how do you scale that? And so, we started Anaconda, basically, we're going to scale array computing and scale data computing in Python. And then the other problem was we were building a lot of GUIs for desktops, and helping scientists build GUIs for desktop was a big part of our work.

00:44:28 And I saw the future was web GUIs and okay, well, I want to make sure that people can do that in Python. I even had a little meme that I did not want in 10 years, all my scientists had to write JavaScript. I was like, that'll be a terrible outcome. I don't want the world to have to write JavaScript. And so in 2012, there was a little ... So, that was why we started Anaconda, was to do that, right? To create this business that also experiment with a funding model. I'd learned consulting business model from Eric at Enthought and really appreciate him, and he's done tremendous work in doing that. I have a lot of respect for Enthought, but wanted to build a product company and do the VC route and figure out what it meant to raise money and understand that world a little bit better. So, that's why we started Continuum Analytics, which was a consulting company to start with, incubating products. Right? And we incubated a number of products and eventually found one that started to resonate and grow, and that became Anaconda.

00:45:29 But Anaconda came about, it was embedded in everything I was doing. Like I said, SciPy itself was a distribution of Python. We'd explored packaging problems, but again, partly because ... And it's still a problem, and this is a deeper conversation we probably don't have time to get into today, but Python still has a packaging problem, but it's actually not a technical problem. It's a social one. The technical problem is actually solved, mostly. There's incremental things you can do. There's definitely ideas of how to do those. But right now, it's a social problem actually. It's the problem that other people have

characterized as the tyranny of the unemployed, the tyranny of the occasional developer. And I was an example of this. People could say NumPy should have been better because I wasn't listening to the people that already had the knowledge, because they didn't talk to me and I was on the fringe and I didn't know how to communicate with them.

00:46:21 And this is true of Python packaging too. You get a lot of participants. People are eager to help Python packaging, but they don't know very much. They know about their problem, they know about the things they're seeing. And then you have the python.org site that ends up getting messages about, "Oh, this is what you should do." And it's very uninformed actually, most of the time. Now, they try, they want to fix it. They're open to suggestions. That's great. But it's also hard. Making a suggestion and getting a change in an open-source community like that, where there's the bike shed problem of the more people that have an opinion, the harder it is to get anything changed, means that a lot of people end up making choices that aren't helpful for them.

00:47:00 Now, that said, there are a lot of great people involved. This is not dissing people. This is acknowledging a feature that is the other side of open-source development where yes, it leverages the free time of people. And it's partly why I want to engage more with people, not just free time, but full time. And then how do you do that, engage with people full time, but while also not allowing corporate capture of the community, letting communities still have their independence and their participation-based methodology and community-driven as opposed to, "Oh, to run this project, you got to be employed by this company."? So, I've differentiated between company-backed and community-driven projects for that very reason and have obviously lots of opinions about that. But I'll probably articulate it best by the talk I gave to the

Python Packaging Summit last year. And I've strayed away from some of these conversations because at Anaconda, we ended up building a distribution of Python, and we created a package manager.

00:48:03 And why did we do that? Why did we create that package manager, Conda? Well, we looked at the ones that were out there, we said, "Well, maybe we can piggyback on top of what's out there. Can we piggyback on RPM?" We wanted to support Windows, Linux, Mac, and make a seamless experience for everybody on those ... On users in particular, not developers. That's an important distinction, actually, users, not somebody who's going to go develop something. And this is where I think if we'd done a better job engaging with developers and making a better bridge, we could have saved everybody a lot of headache over the past years, but our focus was making a really good user experience for the person who's going to start using Python for science and data. I think we did a good job of that actually.

00:48:44 What we didn't do was help the Python community understand that use case. Well, it was also developing a distribution mechanism that still does not solve the problem actually, and it creates other problems. So, there's this disconnect now. And I see people all the time today, they're like, "Okay, Conda is great, but it doesn't work because of X." And that doesn't work because of X is well, it actually works just fine. But you've gone and now you're using an emergent distribution from python.org. And then assuming that that'll work well with a Conda distribution, it's like that's actually a difficult problem. And so, well, you're stuck with the emergent distribution that pip is telling you about.

00:49:26 It works quite well to have Conda be your package manager for binaries, and you can pip install on top of that source packages. But when you start mixing the

binary distributions, you're going to have trouble. That's true of every single package manager out there if you ship binaries. But it's a problem. I try to characterize it like this, don't install vendored wheels. That's the problem. And it's all over the place. And if you're installing vendored wheels, just put an asterisk because you know you're going to have problems. You can do it and if it works, fine, but don't blame everybody else when you're ... Because the problem is those vendored wheels. Vendored wheels, they're a duct tape. There's answers to that besides vendored wheels. And what I mean by a vendored wheel is I'm installing NumPy. Well, NumPy has a reliance on BLAS. BLAS is a basic linear algebra subsystem that is a C library that's available in multi-flavors. But NumPy doesn't control BLAS, it just relies on it....It's a dependency.

00:50:27 But BLAS is not a Python program, it's a C program. And so how do I ship BLAS on PyPI? I don't do that. Why would I do that? But yet when I install NumPy, I need BLAS, right? And so Conda manages this by having a BLAS library that you install, that you have a dependency of separate build process. You have a separate versioning control and it all works together. And you can get NumPy working with a BLAS component. But on PyPI you have a vendor wheel. You've just basically installed the NumPy binary wheel with some version, that, hopefully you have an answer of which version it is, you're not quite sure. Just the binaries are copied into there, and you ship that. Okay, I mean it can work. And if you do that once, maybe it's all right. But you essentially immediately have a technical debt. And what unfortunately the Python community has encouraged is the growth of technical debt in the packaging world. And so this is my complaint, and the way I characterize it, I say actually the Python community has created a channel conflict. They have a channel problem.

00:51:30 A lot of companies have this problem where they want to enable their people, but Python should be enabling all the channel partners out there who help their people get Python. If you look today and say, "Hey, I use Python." Cool. How do you get it? Actually a dozen... We had a podcast actually where we invited all the people that create distributions of Python, kind of other ways to get Python besides the emergent python.org distribution, which I call it an emergent distribution because that's what it is. Nobody controls it. It's just kind of is whatever it is that's on the basis of a bunch of people out there. And it's cool. I mean, I'm not against it. It's like a cool reference distribution actually, but I am against the company relying on it. I think, if you rely on it, you have to own... You can just use it as a place to start. But you as the company then have to own the distribution.

00:52:19 And now your people who are installing Python under you as a company, they're getting it from you. And so it is helping the Python community understand that they need to have channel partnerships with all the people that they're helping. And effectively they're not. The Python Packaging Authority is essentially championing one emergent distribution instead of encouraging and being a system to help all the channels do a better job of cooperation integration. So that's just an example of a problem that's an open-source problem that emerges.

Jon Krohn: 00:52:52 And I suspect that this problem is also the one that you left... So reading all those economics books as the grad student at the Mayo Clinic, you saw this as, there is an economic opportunity there. Right?

Travis Oliphant: 00:53:05 Yes.

Jon Krohn: 00:53:05 So you're just describing how the corporates need to be able to rely on a distribution, and don't want to have to

manage that themselves. And that is how Anaconda got a huge-

Travis Oliphant: 00:53:12 That was the product.

Jon Krohn: 00:53:12 On a skyscraper.

Travis Oliphant: 00:53:12 Correct! That's how it got a global on skyscraper because we saw there's an economic opportunity there. And there's many, and Red Hat did the same thing. I mean, there's other companies that do this, but what I'm saying about the open-source communities is, rather than interfere with those opportunities, they should partner and essentially stand for something. And there's lots of things for the PSF and the PyPI to stand for, things like purity of the projects, let the communities govern themselves, have them their own footprint, have them have a place to have a voice. Because what you don't want is big companies, I think, this is what I don't want. I don't want to see big companies controlling the open-source communities they rely on. I want to see them supporting them. And there is absolutely a thing of community capture, right? You see that also in the Apache Foundation's mantra. Apache wants the same thing. They support the same model, which is great.

00:54:07 Anyway, that's a deep topic we could go on for hours about this, but it is at the heart of what I've seen and what I want to help happen. So Anaconda found a product to sell that was very helpful for companies relying on open-source, that could engage with communities. But of course, executing on that mission is always a challenge. There's a whole other chapter of the story there about the investor relations and how is it I was CEO and now I'm not at the company? And that's a whole journey that requires some nuanced conversation. That if you're interested, anybody, ping me? We can talk about that, but probably not best to air here.

Jon Krohn: 00:54:42 There we go. Okay. I won't press on that, but what I might press on then is another open-source library that you are hugely involved in, but might not be as familiar to our audience as SciPy, as NumPy, as Anaconda, which is Numba.

Travis Oliphant: 00:54:57 Yes. Thank you. So Numba is awesome, and fortunately I wrote the reference version, and then very quickly found Siu Kwan Lam, and then several other people who actually helped build Numba, make it better. But when we started Anaconda, that was one of the things we ended up doing was actually writing a compiler for Python called Numba. And I remember bringing that to market, bringing that to the community, bringing that to the Python community. I got all these questions, wait, have they even swallowed? PyPy was out there, PyPy, not the Python packaging index, but the Python compiler. PyPy, P-Y-P-Y. Anyway, lots of people who said Python compilation, illustrating again the dichotomy between the computer scientists and the domain scientists who use computers. That was the PyPy people. They were writing something for computer scientists, like a compiler for Python. Cool, awesome. I was running with Numba, a compiler tool to create extensions to Python without writing C code.

00:56:01 And it again came from a fundamental itch. I had a fundamental need I had. So when I wrote NumPy, NumPy is an array object and a ufunc object. The universal function, ufunc. And a ufunc is something called multiple dispatch. It's a multiple dispatch mechanism in Python. And it's a concept people talk about a lot, not in Python land, because in Python land they talk about multiple inheritance, single inheritance and object oriented. Multiple dispatch is like, I have a function, but they've got three arguments. Well, which object has a method that influences function? Well, if you have three arguments, which ones should it be? None of them, all of them.

There's a separate... So you have an independent table, you don't register the table with respect to the objects. You register the ufunc, essentially, there's a table for every function, and that's all it is. It's a lookup table to say, I've got these arguments. Pick which underlying implementation I'll call, if I have ints or floats or strings, whatever I call that function.

00:57:01 So ufuncs do that. There's a lot of things they do actually, make it very nice to do math and mixed math with floats and integers, and then particularly double precision, single precision complex, double precision complex, single precision complex. You have a lot of these different... And they all have to be written, compiled at least differently, the machine code that's run is a little different based on which type you're running for. So the ufunc, something has to navigate between the python spelling and which code snippet you're pulling in, which machine code you're pulling in. That's what the ufunc does. So great. How do you make new ones? Because NumPy comes with a bunch of them. Cool. What if I have a new function I want to write like sci-fi special added a bunch of them.

00:57:44 Okay, great. I want to write it. You can imagine writing a python function. A simple example is the Sinc function. It's the sign of pi x over pi x, is one incarnation of it. But what if I wanted that as a ufunc? Well, I could write it in C. The only way to do it, and there was a way to do it. You had to write C code and compile it. I wanted to write that in Python, and there was a decorator called Vectorize, and that's in NumPy from the beginning, NumPy Vectorize. So you could create a python function and the Vectorize would take that scalr kernel and then make a ufunc that would call that function at every element. Cool. It would add a new ufunc. Cool-ish, but the problem is it wasn't a compiler. You need a compiler to do that. The vectorize in NumPy didn't compile. What it did is it made an object array ufunc.

00:58:32 So object arrays are simply, every element of the NumPy array is a pointer to a python object. So logistically useful, because you can organize your thoughts in an array nicely, but it's not fast. You're basically going through the interpreter every time you make a call. So I had Vectorize, but it was going to the interpreter. So if you have a big array, lots of elements, could be 100 to 1000 to 10,000 x slower. So I wanted a way to write a function in Python that could go into that ufunc table. So I just wanted to compile and get machine code in the right table, a function pointer. So how do I make a function to function pointer that's not all of possible Python, just the use Python spelling to get code. That was the goal, and it's like, yeah, we can do that. LLVM was there, Low Level Virtual Machine. I said, "Oh!" Like I said, it's easy to write a compiler if you're not writing the parser or the code generator, because essentially what you're doing is translating Python either by code or Python, abstract syntax representation to the LLVM intermediate representation. So that's what Numba did, is it translated the Python bytecode to LLVM IR. Basically it played the Python bytecode and emitted LLVM IR for a subset of Python.

00:59:47 And it was actually surprisingly, I could do it as a person who never write a compiler before. And I learned a ton along the way, and then other people wrote better code and it worked. It's like, wow, I can actually now write a vectorized, and we do number has a vectorized function that lets you write a Python syntax and build a machine code level ufunc. And that's amazing. That was really the goal, and we did it pretty quickly. And then it was like, well, we have a compiler now. What else can we do? And so then we kind of built a jet and then kind of this gradual just in time compilation for NumPy-like code. Lots of extensions that really could have been... And Numba was another example of an open-source project, succeed because some really smart people got involved.

And then it's been funded by Anaconda. We can go into detail about Numba. Numba, it's a harder code base for users to get involved with because it's a compiler.

01:00:41 And if you're a domain expert, which uses Numba it's kind of hard to go from, I'm a domain expert using Numba to now I'm contributing to Numba because I've learned how a compiler works. It's a different thing. But that space has exploded. We were very early. That was 2012 when we released Numba. We had a version that actually targeted GPUs. You could actually have, it was a fast vectorized. Basically the vectorized code would go on A GPU. And we had massive, like 10,000 x speed up, because you could go from your object array you vectorized to GPU supported vectorize. It was amazing. And there's a little gamification there, but it's possible. And that's still happening today. That's still happening. But what I find is a lot of people get confused and they start talking about Python compilation. You have to be very clear about what you're doing because compilation just simply... It's translating high level language, high level specification to machine code. And I look forward to a future where Python interface to LLM gives you a... I'm basically orchestrating compilers. And Siu is a great... I still have regular conversations with Siu.

01:01:52 I think Numba has a bright future, but there's also JAX, there's PyTorch compile, there's Triton, there's lots of... There's five others, there's L Python. It's actually a fun place now. Where there's a lot of people realizing, oh yeah, this is possible to write Python compilation, and now lots of money involved. They typically don't talk to each other. They typically do their deep stack stuff and try to find... I'm excited today about finding high level cooperative IR, intermediate representations. So LLVM IR is pretty low level and bytecode is Python. Is there a, MLAR is one place. Is there a place to have these higher

level intermediate representations that are not quite low level IR, but a place to target?

01:02:36 Because then you can have other tools target that, and it's a really easy way to cooperate if you can find a plateau of cooperation. In fact, I think that's a big deal and will continue to be a big deal as LMs enter our world and let people write English code to generate something. But what are you generating? I think what you're generating becomes that plateau of cooperation. That's really interesting and will continue to be, because the thing you generate still has to be editable, still has to be maintained by a human, but it doesn't have to be all the domain experts to maintain it. We've got to have a class of people that can. So anyway, that's a whole topic of future potential progress I'm looking forward to.

Jon Krohn: 01:03:14 Also, generative AI also makes it much easier for people to be able to dig into code that they otherwise might not understand.

Travis Oliphant: 01:03:21 Yes. Oh yes, that's exactly right. Thank you. Thank you. Thank you. Yes. Could you imagine? I can translate from that to code. I think that's also a brilliant thing. You talk about a lot of people, and it's true, AI will replace certain kinds of tasks, but as the optimist I am, I say, that's fine. Let's get rid of the tedious jobs and let's use AI to help people have better jobs.

Jon Krohn: 01:03:42 Exactly.

Travis Oliphant: 01:03:42 Where they're doing things that are better and easier for them. And can we use it to shorten the training path? The reality is there is the problem of, okay, I need to get better at what I'm doing. I need to learn something different to participate in the economic model. And I'm totally for supporting people in that journey. But we should, this is

great. There's huge opportunities here all over the place, actually.

Jon Krohn: 01:04:04 Speaking of making things better or easier for people, a interesting item that came out of our research on you was when you were on the Lex Friedman podcast, you talked about how programming languages can limit or expand thinking. So I thought that's really, really interesting. So given all the work that you've done in Python, particularly for scientists, with that kind of use case in mind, coming from your science background, how do you think Python has influenced the thought processes of scientists and engineers and what they can do in their work?

Travis Oliphant: 01:04:33 Well, that's a great question and one that should be asked by a lot of people or to a lot of people. So I will say I language structures your thought, fundamentally. So it helps you say more, like when you talk, you then realize an ephemeral firing in your brain, and then it promotes a question you're going to answer then again. And so it actually inspires your thought. And so the words and the symbols, the tokens that are essentially a manifest by language can influence future thought. And in Python in particular, what I would say is it's enabled introduction of higher level programming concepts to domain experts. It's not hard to get your head around a function call, but even a function call has to be explained. Oh, the fact that I can take my lines of code. If you look back at the Fortran era, scientific code was statements a mile long, all in one big linear sequence.

01:05:29 And you go in there and say, "Ah, we can actually break this up a little bit. You notice how you're cut and pasting this segment of code three times. Let's make a function out of that. Let's actually change that." And now if you make changes, you don't have to make it for 40 places. You can make it in one place and everything benefits. So these are concepts, essentially programming concepts. So

I would say helping Python, the more domain experts get software engineering principles in general is a big way that Python has helped. It helps to the degree that there's a transition, because I don't recommend somebody new to Python diving into async programming or category theory from the typist, from the type theory folks. As Python has grown a bigger community, there are definitely a place to get lost. And that's true, but yet there's something to learn from that.

01:06:18 Because the nice thing in Python is, whatever it is, all the way from just how to do a simple loop structure, a simple function calls and stitch together variables, all the way to compiler theory and intermediate representations and equivalence graphs, it's all possible to learn. Some of that can actually help your thinking about other things. You think about how do I approach a scientific problem, hypothesis testing? Well, it's kind of similar. You have to structure your thinking, look at the results of the output, ask a question that's falsifiable, and try to get an answer to that question. And programming is also rigorous in the sense that the computer does what you tell it to do, and it only does what you tell it to do. It doesn't cooperate with you yet. I mean, maybe AI will help with this, but even then, I think we still need to be very clear. I've spent so much time tracking down a bug because the computer did exactly what I told it to do. I just forgot to tell it.

Jon Krohn: 01:07:19 A really cool conversation that we had recently on the show was in episode number 754 with Jason Warner, who has just started a company, well, he didn't just start it. It's a company called Poolside, and in the coming months, they're going to release their first LLM to the public, and it's an LLM specifically for coding. And I didn't expect his episode to be so mind blowing, but it was because he made the case, compellingly in my view, that a code generating LLM that writes code that compiles, could be a critical step towards an artificial general

intelligence. Because we have so much training data out there, so much code examples. And because it has to work, like you're just saying, unlike natural language, which can be ambiguous, the program language just must work.

Travis Oliphant: 01:08:13 Yes. I actually really agree. I think that's a limitation, and I agree that's a really insightful observation. I've been a little surprised actually, how LLMs can create code that works. It doesn't always.

Jon Krohn: 01:08:29 It doesn't always. But the amount that it does blows my mind. Absolutely.

Travis Oliphant: 01:08:32 It blows my mind too. It's almost like, well, it did train on existing code that worked. So if it's using those spellings, those aggregation of words work. So there's a structure there. There's a real structure to code. That's been a surprising result, but I think one that's super fascinating, and the feedback is what's essential. So yeah, I've heard lots of anecdotes, and myself also have this anecdote of the copilots help talented engineers work faster and better. And I agree with that.

01:09:06 It means that they're less inclined in you to use interns, but there's still a lot of people wanting to engage with interns. And so I think there's answers to that. There's actually a big thing I'm doing right now, it's called POSSEE POSSEEe.org, P-O-S-S-E-E.org. It's really new. Hopefully by the time this airs, there'll be more out there. If there's not... Some people realize that for me, I have big dreams, some of them I can bring to reality in the next month or two, and some are going to take the next year or two. But I want to hit this one because it's a big thing that has a lot of potential, and I'll be doing for a few years, even if only in my spare time.

	01:09:45	It's enabling the gap between the emerging work study, the emerging educational students, people all over the world who want to get into programming, get into making a difference in a world where internships are a little harder to come by unless they're managed. So we basically create managed internships for open-source contributions. That's what POSSEE does. Practical open-source sustainability experiences for education is what it means. So I'll talk more about that later. I just want to put a brief bookmark. So if you hear this podcast, take a look, see how you can help. Love your participation.
Jon Krohn:	01:10:21	I mean, now could be a good time in the episode to transition over to the commercial stuff that you're doing now. Let's do that. Yeah, so if you want to talk about POSSEE a bit more now, let's do it. And then we can get to OpenTeams and that kind of stuff.
Travis Oliphant:	01:10:31	I'll talk about kind of the things that I'm doing that... This is a new venture, and I will say generally commercially what I'm doing is a continuation of the Anaconda story, a continuation of my academic exploration of projects, and then the economic education about how do I make this work, first for me? I'm scratching my own itch first of all, I've got to have a career. I've got to support my family, but then of course, I want other people to have the same experiences. I want to be able to, well, how do I help others who are on this journey have an easier time than it was for me, for example?
	01:11:04	So that's really what defines me right now, and it's defined me for 20 years. So Anaconda is an outgrowth of that. When I left Anaconda, I realized, I want to see a world with a lot like thousands of multimillionaires, hundreds of thousands of multimillionaires, not thousands of billionaires, not hundreds of trillionaires. That's what I want to see. I personally would rather have 100 of my closest friends be worth 20 million each, than

me be worth two billion. Maybe that's a quirk of my nature. I think a lot of people are that way too.

01:11:38 I don't mind organizations with lots of money. I don't mind organizations having assets, but they need to be managed by a group of people, not just having one person who owns it all. Now, other people disagree. But it stems from my recognition that, yeah, I know some stuff and I want to ensure my knowledge is shared, but it's the collective knowledge. It's getting groups of people together. And there's dangers in having power concentrated in single people. There's real dangers, and it's a weakness in our humanity. The history shows us this throughout history, lots of human efforts have failed, essentially on the back of the human ego and on the problems of ego.

Jon Krohn: 01:12:20 Yeah, we're not a political show, but man, we would have a lot to talk about if we dug into that today.

Travis Oliphant: 01:12:27 That's correct. And the good news is I don't have... It translates lots of ways to the political spectrum. The principle is that there's this MRI study that was done that impacted me, when it showed the MRIs of CEOs. People had been in a situation where effectively they'd only been said yes to. And I've seen this already, even with my... I mean, I'm not a powerful person, but I've run companies, I've had a position where I impact the salary of somebody or the bonus of somebody. I try to share that load, but eventually there's an asymmetry there, and it's really hard. You have to be very intentional to get people to tell you bad things, tell you things you don't want to hear. It's actually not easy to do. And in fact, they've shown that if you're in a position, your mind will change. Your brain changes. The part of your brain that's more aware, your empathy part, that's looking for how to engage with other people. It actually shrinks in people who've been in a

position where they've just essentially had yes people around them.

01:13:30 So it's a human nature problem. It's a part of our nature. So I don't want that. I want a world where we have lots of cooperatives. So when I left Anaconda, I said, "Well, good. How do I make steps to try to make that world, or try to impact that world?" So Quansight, we created Quansight to build on the successes that led to Anaconda, which was data science consulting, and really tied to open-source projects. Added an open-source research lab, Quansight Labs. So we have, Quansight Labs, and then this other component, it's like an innovation component. It's a venture fund. It's an investment in other open-source companies, and some of the benefit of that will benefit labs if we make returns from these companies, it'll go directly to fund open-source research and labs. It'll also, we have an entrepreneur residence program so people can work and try to build a company, even though they can't full-time yet. So we help people kind of prepare for that, see if they want to do it, see if they're actually ready for it. Not everybody should make a company, but a lot of people can join early stage companies, but we have an entrepreneur residence program. That's Quansight, and we did it intentionally. We said, this is what led Anaconda, let's create the substrate to lead to 15 Anacondas if we can, more companies like this. So, built this thing and then we're always looking for capital partners. I don't have enough money to fund it all to the degree I'd like to myself, but we do a lot we can and find other capital partners to support us. And then, we've had a few spin outs from that, essentially. So, that's the foundation, Quansight, Quansight Labs, Quansight Initiate. And we've been able to create a few other companies spinning out of that.

01:15:08 Being able to, first of all, invest in about 13 other companies, these are just seed level investments into

other people's startups, particularly where we feel like we can have an impact, we can help them, provide guidance, provide advice, provide help, hiring the first staff, whatever. And then we've also, ourselves, spun out, as a venture studio, additional companies that feel like they have a role to play in fulfilling the whole mission of supporting, connecting open-source communities to company business opportunities, which is a massive space. The massive Quansight Labs Initiate, this could be replicated, there could be 15 of these companies out there, I don't have to have the only one. It's almost like here's a pattern you can follow and there could be 15 of them. Because I think, one, I just saw a notice that there's some venture capital people doing this, which is brilliant.

01:16:01 Every single venture capital company out there should have an affiliated open-source research lab. It's silly they don't, honestly, it's completely... They should. But they need to run it well. And so, if you're going to do that, please call me, because don't stick your head of VC investment in charge of that. Let's talk about how to do that well, because you can screw it up pretty badly, but you should, and you can, have a really good open-source resource lab. And you could support a lot of stuff, and they get tons of companies showing up out of that. So, I think that's a pattern I'd like to see replicated, we've been pioneering. And I can explain how that works to anybody who wants to know. And then, of course, open-source consulting. Every single open-source project out there that's successful can have several consultancies around it, and that's a way for people to... That's the way I've had take home, that's the way I've supported my kids is, "Okay, I don't know how to make money all the time, but I can make money consulting." That's how I committed up at JP Morgan. That's why I'm sitting there, and it's not always fun. It's not always working on exciting stuff, but I'm always learning, you meet great people. For me, it's

been fine, because I love people and I love meeting great people and I love working on interesting problems. And then learning what I can from that to build something new. So, that's what we're doing at Quansight.

01:17:19 Then we created this OpenTeams model. And people look at what I'm doing, going, "How are you doing all this?" Well, I'm not doing it myself. The only way is to find great people. And when I started Quansight, I said, "We're going to build products, we're going to build people, talent, projects, and products." So, it's about people. And every time I have a new company, there's somebody else there. Sometimes that doesn't work out. When it becomes really difficult is when that doesn't work out and the people move on and I'm left going, "Okay, I guess it's me still." Then I have to figure out how to fix the issue and find other talented leaders. But OpenTeams, its whole story is, I want people to be successful with open-source, I want every company to be. With all the Fortune 500 consulting I've done over the past 15, 20 years, I see a lot of bad use of open-source. And bad simply means wasteful. It either hurts the community or it doesn't use it, and they spend way more money than they should. So, lots of technical debt. Instead of getting the power of open-source to enable them to build their value add on top, they end up recreating problems inside the organization that are not solvable to have technical debt. So, I want to help people do that better.

01:18:29 And so, OpenTeams has that mission, and we do it by basically creating a marketplace to connect. We have a sales network and we have an architect network, and then we have partners who are selling out solutions, and then we basically go to provide business process operations to help our consulting companies. They don't want to build the HR team or the admin team or the other pieces that make a consulting go well. I want them to be able to focus on their story. So, Quansight can focus on

his data science story, on its generative AI story and not, "Okay, how do I have an HR team and how do my finance team work? And how do I support my sales team? How do I support my marketing operations?" OpenTeams does that for consultancies, and then also helps them cross sell to each other. So, one company sells data science really well. Okay, great, we have a salesperson with a strong relationship with their procurement guy at their company. Well, that company also needs web development. That company also needs Rust support. That company also needs JavaScript. Any kind of other development that isn't the sweet spot of Quansight, OpenTeams builds a connection to that salesperson so they can resell it from our other network of suppliers. So, we're basically a glue for enabling salespeople to never say no, and enabling a sales channel for talented teams.

01:19:53 If you look at the consulting companies that are out there, the big ones, Accenture, Tata, they all become big sales companies, and developers hate working for them. What we're trying to do is build a sales network that does the same thing, while allowing developers to work at companies that fit their culture, that let them be tailored to their communities, to be governed by that instead of just being fitting into some machine that's like a sales machine. But the sales is critical. You get deals because you've got a relationship with sales. And so, we let their salespeople benefit from other salespeople and let them kind of grow together.

01:20:30 That's the vision, and we provide all this operational support. In some sense, it's like I had to learn a ton of business to support my open-source addiction. How can I share that knowledge with other people without recreating the wheel? The other two parts of OpenTeams, which they use the same brand, OpenTeams Global, its key is building the open-source professional network, ospn.org. I want everybody who makes contribution to

open-source to be able to build a career out of it. I want them to get the recognition, the credit, enable them to have their contributions go with them to the dream job they're looking for. So, ospn.org is basically like a professional association of open-source contributors. And then, as a business, it's a payroll HR operations business. That's the basic piece. And it then does, what I call, long tail staff augmentation placement. For all of our companies, they may need to hire someone, they need a recruiter, they need to work... OpenTeams Global provides that for people.

01:21:30 Then the final one, is the one I'm going to be doing forever, which is OpenTeams Incubator. So, OpenTeams and Global, I'm looking for leadership to help me run those companies. But Incubator, I just want to participate in this forever, and this is basically helping people build startups with open-source affinity. So, we have an open-source fund that we manage. We do have a network of experts, product management experts, marketing experts, sales experts, to help our product company partners build businesses better. So, that's Incubator. Anyway, that's what I'm doing. It's kind of a lot.

Jon Krohn: 01:22:05 It is a lot, that's wild.

Travis Oliphant: 01:22:06 It's a lot. It is wild, but ultimately, I could go on and on about the partner companies we have, the portfolio companies, because what I want to have is a portfolio of wonderful entrepreneurs, engineers, and to be their biggest cheerleader and be able to help them build their companies. So, it's kind of what you're seeing as I'm trying to seed that and get people engaged and help people work on things. I can't do it all, obviously. I'm just trying to feel, "Well, what can I share of what I've learned? Make different mistakes than me. Learn from my mistakes."

Jon Krohn: 01:22:33 And how can our listeners get involved with these communities that you have, these commercial projects that you have?

Travis Oliphant: 01:22:38 Oh, thank you. Well, OpenTeams in particular has three communities that it's growing that are business communities, essentially, and they're different. Depending on who you are and your skillset, one of these may resonate. And it's very easy to get involved. Best way to ping me right now, or to go to our OpenTeams website and follow the leads to get involved in these communities, openteams.com.

Jon Krohn: 01:23:01 Openteams.com, yeah.

Travis Oliphant: 01:23:04 Openteams.com is a center point for these communities. One is what we call the open-source architect network, OSA community. There's a lot of people. Every successful engagement with open-source that a company has, is because they have somebody like this. They have an architect who knows how to use the product, or the open-source world, and is aware. So, we're looking for a community of those people to share practices, to talk to each other, because you don't know everything. One of the key pieces that OSA, an open-source architect does, is know what they know and know where to go get help for stuff they don't know. So, it's having colleagues and having a network of people you can lean on to build solutions for companies.

01:23:40 So, it's free to join, but there is an application process because I do look and make sure you have, one, some experience. You're not just fresh out of school, looking to get involved with your first open-source project. I want to see that you've actually done some work in open-source, you have experience building solutions with it. So, that's that OSA community and it's a free community and it has events, a Slack channel, and a place to participate with

people. The other one is, we call it Engineering Manager Community. This is to support people who are building teams at their other companies and understand the nuance of how do I hire from an open-source community? How do I get in touch with people? What if I need myself to build a product using open-source? What is the best way to do that? How do I avoid creating technical debt in my company, using open-source? What are the best practices out there? What are good tools? So, that's the Engineering Manager Community. It's on LinkedIn, it's also available from our website.

01:24:28 And then the new one, which should be available by the time this airs, but I hope, we're just getting started, it's our Open-Source Sales Community. So, it's a community of sales professionals who want to build relationship with clients, who share in the vision that my view of a salesperson is, they're an advocate for the customer. They're basically the customer's connection to the ecosystem. And to do it well, you really have to get the trust of your customer, so they trust you, not just to sell them something that makes your commission, but you actually have their best interest at heart. And the best salespeople I've worked with, and I've had a chance to work with hundreds, that's their mental model.

01:25:08 They want to help the customer, and that's what we want to do, is build a group of people who want to help their customer, who understand that customers, that clients, that enterprises need open-source to do their job, and they need to figure out, "Well, how do I connect with the right people? What are the best practices? What are the kind of products that I could buy? How does that work? What is licensing issues?" Just a place to share and talk with other like-minded professionals. So, those are the three communities you can get involved with at OpenTeams. We're excited to grow those communities and, again, solve the mission of the problem of how to

help enterprise use open-source better, more efficiently, more effectively, save the millions of dollars, trillions of dollars, certainly billions now, trillions later, that could be instead reinvested into open-source development. That's what we'd like to see happen.

01:25:51 So, that's OpenTeams. That's the best way to get involved. Otherwise, ping me if you're an entrepreneur who wants to start a company, you want to see if we have... We don't have deep pockets in the investment dime, we're looking to raise more money, but we have lots of connections. I do know a lot of people with deeper pockets and I can point you to people who might be able to fund your initiative. Super excited to get involved there. Just ping me on LinkedIn, Travis Oliphant, Twitter or X, I guess, teoliphant. There is a Threads, you can also get me there. I think it's teoliphant again. Just ping me, I'm pretty responsive usually, unless I'm overloaded, in which case be patient or ping me again, if you're really urgent. And I like to talk to people who are serious. I don't have a lot of time to give, I definitely can give advice, I can point you to places. I don't have a lot of money and I don't have a lot of time, so just be aware.

Jon Krohn: 01:26:47 Yeah, and to that also, I know we have several investors who are listening to our episodes, and so this also sounds like a potentially great opportunity to be providing capital into the kinds of projects that are going to have the biggest impact and therefore the biggest economic potential returns as well.

Travis Oliphant: 01:27:04 Yeah, correct. If you're an investor looking for where to have an impact, please talk to me. I probably have opinions and you might find some of them useful, and I'd be happy to explain why and perhaps that helps you get a better return. There is one project, it's still incubating, I didn't bring it up in my Lex Fridman Podcast intentionally, because I knew it was still early. It's still

early, but I am looking for like-minded investors. So, it's called FairOSS, faiross.org, and you can go there, it's an incubating project, which means it's hibernating, actually. I don't have anyone to lead it, and so therefore it's not making much progress, but we're still holding on to transactions.

01:27:46 This is a way to actually create a marketplace for open-source. I believe it's one mechanism that could actually solve the problem of connecting the $100 trillion of investment capital that's globally available. This is money that people want to put somewhere to make alpha, to make a return, and they have various horizons of return profile in that 100 trillion. Some want it in six months, some are willing to wait 10 years. But then, we also have open-source innovation, which is underfunded. And my goal is to create a market connection between the two, so investors can invest in open-source, but you've got to make a tie to the economic output for that to work.

01:28:27 So, FairOSS is really about creating a fair market value for open-source, a ticker symbol for open-source. And we do that in a couple of ways. If you're interested, and I don't want to go into more detail here, because I don't have the time and it's still hibernating, and I need funding to make that happen. And it's one of those things that, you've got to have the right mental model, you have the right mission and passion. If you're interested in it, let's talk, because I think we've uncovered an absolute way to solve this problem in 5 to 10 years. We'll have to work on it for the next 5 to 10 years, but it can absolutely transform the world.

Jon Krohn: 01:29:00 Yeah, no doubt.

Travis Oliphant: 01:29:02 That's why I'm excited about it.

Jon Krohn: 01:29:04 Yeah, I agree, there's a huge opportunity there as well. Conscious that we are running up to near the end of the amount of time that I said this interview would take. So, we had several audience questions. I'm just going to pick one that I thought was probably the best of all of them, so apologies to all the other people that had great questions. Siri, Starkey, Lanam, you had some amazing questions, but Svetlana Hansen, we're going to get to yours. So, Svetlana, she's been a long time listener, many years now of participating in the podcast. So, she's a senior software engineer creating real-time network engineering solutions for NASA Spacecraft, which is a pretty cool job. And so, unsurprising to hear somebody working at the frontier of space asking a question like, "Where are we going with the future of scientific computing and Python libraries?"

Travis Oliphant: 01:29:56 Great question. I think I was capturing it when I talked about, essentially, the compiler framework. I see the future in something like torch.compile, in Numba, in JAX. Now, it's not there yet. I wish I could tell you, "Just do this." I would say though, write your code at a higher level. If you're writing low level loops, question why? You should be writing array code. I still think the future is array or array computing. Write your code with, "Here's an array, or a data frame, or database computing," because then it enables you to then, you're writing code in a way that can be optimized by compilers in the future. As opposed to the more detailed you write it, like, "Oh, I'm writing this loop and grabbing that element of this loop," and the more detailed you're doing that, the harder. Today you can do it. High level array computing, write a Numba of uthunk. Write a uthunk to do the low level code you can't just do with operations that exist. You do that and that will be future-proofed for the future. And it works today, and in the future, you can take advantage of new innovations and compilation technology. So, where we're going, I think is a world where one, people write in a higher level and the code is faster still. It's going to take

time to get there still, but I think there's ways to do it today. And two, I do think generative AI is going to help us with better human computer interfaces, so that more people will write even higher level.

01:31:20 You can just use English language to express your ideas, and then that'll translate to frameworks that will spell that out, that then get edited and, depending on where you are in the cycle of the innovation cycle, you'll either be operating at the very most exploratory part of writing human language and interacting with the visuals, or supporting the technical libraries that emerge from that exercise and making sure that they keep working, or even building frameworks at the lowest level. You have a whole career path, and you can kind of go up and down if you'd like, but I really see a world where I really want experts to be able to think about their problem. Python's been popular because it got out of the way of people, let them think about their problem instead of their pointer arithmetic and their semicolon placement. They could just think about their problem. And that's still needed. We have scientists solving hard problems, thinking at a big scale, and I want to support it, and I think you can. But keep track of the compile world, ask where this run is happening. Is there an optimizer somewhere that's able to take your code and then make it faster? And if the answer is, it's nowhere, then okay, maybe you should be thinking about ways to do that, or looking for cooperation plows to do that with.

01:32:34 And then otherwise, it's still frothy. Interoperability I think is still... And data. The other thing I would say is, don't lock your data up in things you don't know. Make sure you know or a public spec exists for your data. The future is going to be bad for you if your data is locked behind a proprietary format that is not available via lots of people. So, that's the other, I guess, thing I would say. And you can use a proprietary database, you can use a

compute engine that you pay money for, that's fine, but just make sure the data you rely on has some existence in a public forum.

Jon Krohn: 01:33:14 Very cool. Great answer. And so, you've already filled this in. Usually, my last question to guests is, how to follow you or how to reach out to you and you provided that a few minutes ago, so we're good on that front. And so then, that means that my final question is my usual penultimate one, which is: Travis, do you have a book recommendation for us?

Travis Oliphant: 01:33:34 Book recommendation? I thought about this a little bit. Maybe a little unusual, but because of my economic background, it might even make sense. So, it's actually called Money, Bank Credit, and Economic Cycles by Juan de Soto. It actually does the best job of explaining what money is, where it came from, how it exists, and how to not let it control you. Now, we're all in the same boat. We're kind of struggling together. In fact, I think one of the key things is figuring out how do we actually make money work for all of us? But this will help you understand the roots, so we make better decisions about it in the future.

Jon Krohn: 01:34:10 Very cool, Travis. I expected to learn a lot about open-source from you, and it was awesome to learn so much about economics as well in today's episode. Thank you so much, Travis, for taking the time. It truly is an honor to have you on the show and to be able to speak with you. Wow. And you exceeded expectations, if anything, such a joy to speak to you.

Travis Oliphant: 01:34:30 Jon, you're great, I really appreciate being here and I love talking to folks about open-source, about their problems they're solving, and I have so much respect for the people who just use these tools to build so many incredible things. It's honestly a true honor to be part of the

community, to feel like a member of that ecosystem. So, the same reason I got started was my love of science and sharing, and that's still what drives me today, so I really appreciate this chance.

Jon Krohn: 01:34:55 Yeah, of course. All right, Travis, thank you so much and hopefully we'll catch up with you again in the future.

Travis Oliphant: 01:35:01 Looking forward to it, Jon. Take care.

Jon Krohn: 01:35:08 Obviously, Travis is incredibly intelligent and incredibly driven. I've also got to say that he must be one of the nicest people I've ever spoken to. What a wonderful experience it was to meet him. In today's episode, Travis filled this in on how his journey toward creating NumPy and SciPy began with his own needs as a biomedical engineering researcher. How language structures your thoughts so programming, with its rigorousness, encourages rigorous problem solving capabilities. How commercializing open-source is essential for supporting open-source initiatives and communities. And how the future of scientific computing will involve major innovations in compiling interoperability and open-source data formats.

01:35:46 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 Travis' social media profiles as well as my own, at superdatascience.com/765. And if you'd like to engage with me in person, as opposed to just through social media, I'd love to meet you in real life at the Open Data Science Conference, ODSC East, which will be held in Boston from April 25th to 25th. I'll be doing two half-day tutorials, one will introduce deep learning with hands-on demos in PyTorch and TensorFlow. And the other will be on fine-tuning, deploying, and commercializing with large language models, including GPT4 and Gemini. In addition

to these two formal events, I'll also just be hanging around and grabbing beers and chatting with folks, it'd be so fun to see you there.

01:36:32 Alrighty, thanks to my colleagues at Nebula for supporting me while I create content like this Super Data Science episode for you. And thanks, of course, to Ivana, Mario, Natalie, Serg, Sylvia, Zara, and Kirill on the Super Data Science team for producing another jaw dropping episode for us today. For enabling that super team to create this free podcast for you, we are grateful to our sponsors, and you can support this show by checking out our sponsors' links, which are in the show notes. And if you would like to sponsor the podcast yourself, you can get the details on how by making your way to jonkrohn.com/podcast.

01:37:04 Otherwise, please share, review, subscribe and all that good stuff, but most importantly, 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. Until next time, keep on rocking it out there and I'm looking forward to enjoying another round of the Super Data Science Podcast with you very soon.