Please note the following timestamps are approximate.

Torsten Grabs (Guest) (00:00):

We've tried to strike a reasonably well-balanced compromise now with what we are offering with Snowpark for Python together with the power of Anaconda because now that can actually bring the innovation from the Python community in a very well-defined way into Snowpark.

Peter Wang (Host) (00:18):

You're listening to *Numerically Speaking: The Anaconda Podcast*. On this podcast, we'll dive into a variety of topics around data, quantitative computing, and business and entrepreneurship. We'll speak to creators of cutting-edge open-source tools and look at their impact on research in every domain. We're excited to bring you insights about data, science, and the people that make it all happen. Whether you want to learn about AI or grow your data science career or just better understand the numbers and the computers that shape our world, *Numerically Speaking* is the podcast for you. Make sure to subscribe; for more resources, please visit Anaconda.com. I'm your host, Peter Wang.

Speaker 3 (00:54):

This episode is brought to you by Snowpark for Python, an Anaconda-embedded repository and package manager for secure and seamless access to open source in Snowflake's data cloud. This integration puts the power of Anaconda in the hands of Snowflake users who already benefit from Snowflake's elasticity and performance. Users can now run secure Python-based workflows directly in the database without needing to copy or move data. You'll have access to the most popular open-source Python packages without needing to manually install anything because Anaconda is built right in. We worked hard with Snowflake engineers to make sure that the Python libraries you're familiar with run well inside of Snowflake's fast, distributed processing engine. To make sure that this Python capability is production-ready right out of the box, Anaconda curated the packages and provides seamless dependency management, while Snowflake provides a highly secure sandbox. For more information, visit anaconda.com/partners/snowflake.

Peter Wang (Host) (01:52):

Hello. All right, big welcome to Torsten Grabs. He's joining us today. We're really excited to have a great conversation with him. Torsten is the Director of Product Management at Snowflake, who, among other things, covers the data science, machine learning, and AI practice area. So Torsten, welcome and thank you very much for joining us on the Anaconda podcast.

Torsten Grabs (Guest) (02:12):

Absolutely. Thank you so much, Peter, and I'm super excited to be here today. So looking forward to the conversation.

Peter Wang (Host) (02:19):

So before we get started, I have a bunch of questions and things that I think we could talk about. It'll be really fun, but why don't you first give our listeners a brief rundown of what your background is and what you do in your day to day and what interests you, right, about these areas that you're covering in your product management role?

Torsten Grabs (Guest) (02:36):

Yeah, so in my role today, I probably spend the majority of my time every day on things related to machine learning and data science and advanced analytics, and how do we want to support those in a better way in the Snowflake product. So I've been spending the last two years probably now, working a lot on Snowpark supporting languages like Scala, Java, and most recently Python natively on Snowflake. So that has been a big topic for us for some time. And obviously now we are very interested to expand on those capabilities and make the product even better for data science and machine learning practitioners. And prior to that, I've also worked on other areas always related to data. So I've worked on data engineering, I've worked on data lake use cases for Snowflake and related product capabilities. And I've been actually a data person since the start of my career in, back in grad school, I've been doing data warehousing on clusters. So quite similar to, in terms of the underlying basic principles of the architecture, to what Snowflake can do today or what data processing looks like these days.

Peter Wang (Host) (03:51):

And it's an interesting thing. My background is none of that. My background was physics and then I ended up doing a bunch of Python stuff and then somehow through numerical and scientific computing in Python, ended up having to learn a lot very quickly on the fly, so to speak, about data, distributed data systems, data warehouses, and all these kind of things, which it's like, oh wow, this is a whole robust area of practice and a very lucrative one from a commercial standpoint, but also it's a very well studied one, right. It's one with a lot of academic theory behind all these different kinds of things. And yet when Python started getting pulled into more and more data lake use cases, let's say, with the big data transformation and some of the disruptions that were happening 10 years ago, what I noticed was that a lot of theory got pushed to the side, right.

(04:35):

So on the one hand, you would have database nerds arguing about different kinds of schemas and stuff. On the other hand, you had a lot of data practitioners just pulling things out of the database into CSVs and cranking through them with Python or R. And so for me, just watching this happen, again, with no, I had no particular religion about any of this stuff. I was getting pulled in as a consultant to do Python stuff in various places. And it was just interesting to me to see where some of the let's say classic or maybe legacy architectures were clearly overshot what the customer needs were and yet there wasn't new things yet that were meeting customers where they were at.

(05:11):

And you can see some of the early noSQL stuff and then with cloud databases, obviously with Snowflake of course rising into the fore in that category. With all that stuff happening, to me it's been really exciting and interesting to see the last just 10 or 15 years of evolution in this space, how much has happened in a space that was somewhat stable for a few decades, I think, but somehow all of it changed. What do you think caused some of that change in your mind? Actually, do you even agree with me about that timeline? If you do, what do you think caused the change?

Torsten Grabs (Guest) (05:39):

I think I agree with the timeline and I think the change really was about the ecosystem that data practitioners could lean on to get their jobs done, suddenly got turned upside down. And I think actually, Python played a pretty big role in amplifying some of those changes. So if you think about the traditional way that database systems over many, many years tried to embrace advanced analytics, data science, or what we would call machine learning today, it had a very typical approach that you would just essentially select a set of libraries that you, as a database vendor, felt comfortable about. You would drop them into

your product and expose them to a SQL surface, right. And I would argue, that approach has seen very limited success and very limited adoption. It resonated with a particular audience that was comfortable writing SQL query, but all the other folks that don't necessarily see SQL as their preferred language, they really didn't have a lot of incentive to go and try that out and use these kinds of products.

(06:51):

And that, then, essentially created a space that then got filled by various alternative big data offerings that we all know, right, and that were catering to different programming languages: Java, Scala, and eventually Python as well. And these programming languages then also, and their support, created a whole ecosystem of open-source communities that provide a very, very rich set of tools, that made the data engineers, the data scientists, the data developers so much more productive in their preferred languages. And that really just bootstrapped this whole space. And I think when I reflect on that, it's super important now that we, and speaking of myself here to a degree as a more traditional database person, that we realize the value that this ecosystem brings and we embrace the preferences of the people that don't necessarily have grown up speaking SQL.

Peter Wang (Host) (07:48):

Yeah. And it's not so much that people rejected SQL, it's just that actually so many more people cared about data than just the people who could afford to buy an expensive SQL database. I think it's almost like when the Bible got translated into English so the regular people could read it, right, and they didn't have to learn Latin. All of a sudden you just got a lot more people into the whole thing. So I think that's absolutely right. And then part of my thesis around this also is that, just the explosion of the web went through the dot-com boom, a bust cycle, there's a bit of a winter for a while, but then during that winter, a lot of residences got wired up with broadband and web technologies were improving a little bit and HTML5 was coming around and browsers were getting faster.

(08:32):

And then you had both social media really passing the inflection point and then you had mobile coming in. All of these things created a viability around web and mobile and digital. Like the e-commerce promise of '99 got manifested seven or eight years later. But then you compound all of that with all these other new kinds of big companies now, right. Netflix, Facebook, and Twitter and everything. All of that just creates such a large volume of data that didn't start off being built into a well-architected classical data warehouse. It was just, all these apps are now dumping all this data. What the heck do you do? So for me, I see that, when I was running the meetups and stuff, it felt like for a period of time of two or three years, every other meetup was someone talking about some yet new noSQL database or some column store, or some other kind of thing.

(09:19):

And yet no one ever gave the talk of “why do I hate SQL.” It wasn't that people hated SQL. And people, in fact, there was a lot of Postgres and MySQL presentations too, but yet there was all this hunger to find new kinds of ways of storing the data more easily with less upfront crap, pardon my French, but then also opening it up to access from other languages. We wanted to use the same language to write the application, do the analysis, and then drive changes in the application. And so, you're absolutely right about that ecosystem. And we'll get back to that ecosystem question in just a minute. But before we get to that, one thing I want to ask about relative to the languages, you said in Snowpark, you guys have been doing enablement for many different languages inside the Snowflake database, besides SQL, obviously. What are some of the weird quirks? Obviously, I know Python has its quirks, but Java and

Scala, what was the one or two really hard things about getting really good support for those languages in Snowpark?

Torsten Grabs (Guest) (10:15):

I think probably, I'd say coincidentally, Java and Scala are the easier languages to support. And that's probably the reason why we started with them because the way that we've architected Snowpark is that we lean on a general-purpose infrastructure layer that provides us with sandboxing infrastructure to host pretty much any language runtime that we want. And that sandbox establishes a couple of really critical security aspects for us. So for instance, it locks down external access from custom code that runs inside the sandbox and that's an important property for some of our customers to make sure that we don't have unwanted data egress or data exfiltration from the code that runs. And so that essentially was the key challenge for us is to first pay down that infrastructure investment into this sandbox infrastructure, which we now have. And it gives us an opportunity to expand to other languages at some point if customers ask to do that. But I think the top priority for us right now is just to get our Python public preview shaped up for GA, which is what the team is currently working on.

Peter Wang (Host) (11:29):

Yeah. So Python was the harder one compared to Scala and Java, yet it was worth doing, I would hope you would agree. And so, let's talk about in your mind, the so-what-ness, so now you can do Python in your database, so what? What made it worth it in your perspective to have that ability?

Torsten Grabs (Guest) (11:47):

Maybe just to jump back to that question, why was Python the less obvious one to do? So I think the first part there is with both Java and Scala, you can lean on the virtual machine as the underlying common runtime. It's very well-defined properties. And that then also reflects back on what your sandbox needs to support. And with Python, I think it turned out to be a little bit more mixed because many of the Python packages actually have pieces of native code that run inside the libraries. And that does not map to the Java virtual machine at all. So this, we have to support all of these various native code packages in a good way on the sandbox infrastructure. But we also wanted to have reassurances around what packages can customers actually bring to the platform, which led to the conversation that you remember when we started talking.

Peter Wang (Host) (12:40):

Yes, right.

Torsten Grabs (Guest) (12:41):

Can we actually, can we provide customer benefits by leaning on a well-established, well-curated set of Python libraries that comes from the community but goes through additional vetting, right, before it gets deployed? And that then led to the deep integration of the Anaconda channel into Snowpark Python offering, right. And going through that journey and figuring that out, those were really some crucial points during our Python journey.

Peter Wang (Host) (13:12):

Yeah and I appreciate you pointing that out. I mean, we see this with many of our Embedded partners and customers, right. They have a use case where they are an ISV, they're a software vendor with a

well-packaged product, they want Python capability in it. And a lot of the reason you want Python in it is actually not for Python, it's for all of the libraries in Python. And then you immediately break into this horrific hellscape of packaging problems and compiler issues and all these other kinds of things. And so, when we've had these conversations, for instance, I think about ESRI with ArcGIS, which is a shrink wrap piece of software that you install to do geospatial analysis, they want a Python capability, but they want to offer, they want to have some semblance of a baseline that they know will just work with their examples, tutorials that they can put their system documentation around.

(14:00):

And so, for a lot of folks, they want to have access to the power of the ecosystem. They know that there's always escape hatches and back, not back doors, but there's ways for people to install other things in if they want to in some of these integrations. But in general, you really want that core set of things where you know how it was built, where it came from, you know who you can talk to to get support on some of these things. And that's definitely one of the areas that we have seen a rapid growth in terms of partner and customer conversations for us. Now, in the case of Snowpark, right, because it's running in a production environment, it runs on people's sensitive data. The amount of escape hatches you really want to have available for people is more limited in that case. You want them to have the power of some flexibility, but they cannot just go and install all sorts of random stuff off the internet. There's a balance that has to be struck between the power and capability.

Torsten Grabs (Guest) (14:51):

Exactly. Yeah and I think that's this, how do you strike this balance between complete flexibility where you can install whatever you want and then you are suddenly realizing that you are subject to supply chain attacks from the packages that you have picked up, which is the one extreme end of the spectrum, right? And then coming back to the beginning of the conversation, the other extreme is that the database vendor picks a particular set of libraries for you and locked that down in their database product, right. If you think about that, I think we've tried to strike a reasonably well-balanced compromise now with what we are offering with Snowpark for Python together with the power of Anaconda because now that can actually bring the innovation from the Python community in a very well-defined way into Snowpark, right. And I think that's really the best of both worlds. So relying on well-protected runtime environment in which you still benefit from all the innovation that the community brings to the Python ecosystem.

Peter Wang (Host) (15:49):

Right and I think something that practitioners, especially those who are maybe earlier in their career, or who work in a sole, in a smaller scale, they don't, not part of some of big business with a lot of these requirements and compliance and all these things, sometimes people don't quite appreciate this, but a lot of data scientists, a lot of our users, ML engineers and whatnot, they're in these business environments where if you want your work to see the light of day in touching production data on production infrastructure, there are other stakeholders who are going to get involved at some point in that process. And they're going to come down and they're going to absolutely be asking, “What are the software artifacts you're deploying into this infrastructure?” And if you wait until that point to start having the conversation about what's allowed or what's not allowed or whatever, you will have wasted a lot of time because more often than not, you will have brought in things or you'll be using things that are not acceptable.

(16:37):

And sometimes it is obvious things like this package is simply, whatever, it's some random package that only exists because some random PhD student released it. But there's other less obvious things like this particular version of a library of common one, pandas or SciPy or something. The version you installed links against some library which had a vulnerability that hasn't been patched and IT simply will not allow it to go into production. And now you have to go and go back and recode some of your stuff, which then means rerunning your tests, which all sorts of things happen. So to accelerate this process for data science practitioners in businesses, it's far better if they start with something that resembles the production environment as close as possible or at least have a way to know when they are bringing in things that are outside.

(17:21):

And so I think one of the things that's subtle but is a deeply important thing is that, when you use something like Snowpark, when you have an Anaconda integration in these kinds of places, you are doing your EDA, you're doing your exploration, you're doing your build in the same infrastructure that is also what the deployment infrastructure will be. So that question, that hurdle, which is sometimes insurmountable for people, it just becomes a nonissue, right. And I think that's a really, it's kind of a weird arcane thing a little bit to try to educate people about, but the fact that this accelerates people's workflows is a really important point.

Torsten Grabs (Guest) (17:55):

Yeah, that's a really critical point. And our hope there is also that we make it much, much easier for data scientists and machine learning practitioners to successfully deploy their models from the experimentation phase into the production workloads in their organizations. One other aspect that might be interesting in this context that I find fascinating is also the cross-language interactions and composability across languages.

Peter Wang (Host) (18:22):

Oh nice.

Torsten Grabs (Guest) (18:22):

So one example I as a data scientist, I start out with my raw data and using Snowpark, I code away using Snowpark DataFrame APIs and Python because I'm really familiar with the DataFrame concept. And eventually, I build a model that I actually now want to deploy so that it runs in a mission-critical data pipeline in my organization. So typically, in larger organizations, that is a point where I would interact with other folks and other functions, for instance my data engineering team, and they may say, "Okay, we write our data pipelines, we write them in SQL, and now you're giving me this model that you wrote in Python."

(18:56):

Now, what's great about the cross-language interactions, it's very easy then to publish that piece of Python code that you have under a SQL surface so that it is super straightforward for the data engineering team to consume that model and invoke it from a SQL-based data pipeline. And that hopefully makes interactions between different functions that may, in your organization, that may have different preferences around their programming languages so much easier. And that could just be one of the additional hurdles in the process of machine learning model quality, right.

Peter Wang (Host) (19:28):

It is. Yeah. It's one of the things that shocked me when I first learned about it, and it may be news to some of the listeners on the podcast, but when you go and you do the surveys, we do the State of Data Science survey of users and just get people's feedback on what they see in their daily workflows. For me, with the first few years we were doing it, I was shocked to see how many people said that the impediment to getting their work deployed in production was that they were being required to recode their models from Python into Java or into C++ sometimes. And yeah, I see you nodding your head because you know this as a PM for an enterprise software company that makes a lot of sense to you because you see it all the time, I'm sure, in your customer environments.

(20:04):

But for a lot of the, like the hip data science crowd that worked at startups where you didn't have a bunch of people telling you to do this, it just seems shocking that you couldn't just take this beautiful thing that you wrote in Python or whatever and just run this notebook in production. And the idea that they would be forced to code it into Java was, well I wouldn't say people fainted, but it was certainly, when we first figured that out, we were like, "Wait, what? Whoa!"

(20:25):

And I think this idea that you can have now an environment, not only are you moving code to data, but now you're actually moving these, you're unifying the workflows, you're integrating workflows, right, across multiple different groups and across multiple different kinds of stakeholders without this friction, without this war that usually breaks out, political chaos inside organizations that happens when you have these impedance mismatches. I would like to imagine that it really makes everyone happy. It makes the data management people happy, because all your data is now in one place, it's not moving out into shadow data CSVs, random shared file systems.

(20:59):

So also for the data scientists, they've got the tools they want to use and it's on all the data there. And then of course for the operations people, they can use their tools to get access to the same data and then be part of this pipeline. That's all great. And I'm curious, you mentioned something there that is very interesting. You mentioned about the Snowflake DataFrame and an API there that's compatible with the pandas DataFrame. Tell me more about that. How compatible is it? Where is the progression of development on that?

Torsten Grabs (Guest) (21:25):

Yeah. So I think the DataFrame concept has been around for quite some time and we took liberal, I'd say, inspirations from both pandas but also from the Spark APIs, tried to put together the best of both worlds here into our own twist on a Snowpark DataFrame API. The interesting piece here is, as I said, it's inspired by others like pandas and Spark, but what's really unique to it is, it is a really thin client in such that it leans on the underlying distributed compute engine for Snowflake. If you write a Snowpark DataFrame program that has a straightforward mapping to the relational algebra, all that the Snowpark APIs will do is translate that code into essentially a sequence of their relational operators.

Peter Wang (Host) (22:17):

Oh, that's interesting.

Torsten Grabs (Guest) (22:18):

A query plan that then runs in a distributed way across the Snowflake compute cluster assembles all the results and shifts them back as a result into the DataFrame.

Peter Wang (Host) (22:26):

Interesting.

Torsten Grabs (Guest) (22:27):

Now that's the simple use case where you have a trivial mapping between the relational algebra and the operations that you've expressed in your DataFrame program. The more interesting use case is the one where you actually compose custom Python code that you have written, for instance in a [inaudible] function, and you want to then invoke that in one of your DataFrame operations.

(22:49):

Let's say in an expression, in a filter. And now in this case, what would happen is we would actually take that piece of Python code, compile it, and then push it down to the server side so that it can run in the Python runtime on the server side and inject calls from the distributed query plan that runs, essentially, as the driver for the whole processing and is also responsible for populating the data into the cluster nodes. And it would call from that driver program into your custom code that runs in Python runtime. And all of that heavy lifting happens transparently behind the scenes. That's the responsibility of the Snowpark APIs to make sure all this is orchestrated and runs correctly so that you can really focus on just typing away in your Python code.

Peter Wang (Host) (23:35):

Yeah and to be very clear, the Snowpark Python support, there's several different levels of it. In the sense that there's full-table, there's table-level functions you could do, but then there is per-row UDFs that you could define, which is the really, really amazing and powerful part of this. And I'm really excited to see what people are going to do with it because to be honest, this is something that we wanted to see. And we have some of these kinds of capabilities in the open-source world that you put together…you can put together distributed computing infrastructure around Python, you can have different kinds of storage things, you can use things like Numba to push down the computation, to specialize it and run very fast, depending on what your storage looks like and all these kinds of things. And also we've had tools that try to map some DataFrame-ish expression things into SQL, into something like a SQL query plan.

(24:19):

A lot of these things have happened in the Python open-source world, but to have a, just a leading vendor like Snowflake with so much production data say, "Yes, we support Python. Not only just support Python in a weird way, no we're going to support it in a first-class way where you write using APIs that are familiar to you, Mr. or Ms. Python data scientist. And also, we will support this pushdown computation with custom Python code." That's really moving code to data in a foundationally, I think it's a transformative thing that we'll see people being able to do. I know, we've had the, it's not GA'd yet, but it is in early preview. We have customers that are using it. What are some of the interesting use cases that you've seen people do using the Python API?

Torsten Grabs (Guest) (25:00):

The use cases are across the board. I think there's a lot of data engineering that's happening. People just shaping their data in interesting ways. And probably the most exciting ones that we are starting to see are the ones where customers are tapping into what we from a database perspective would call

unstructured data, although that's generally not true. This is things like text documents, images, videos, audio files, right.

(25:26):

And then what's really phenomenal is to see that there's already such a rich support for these data formats from the Python community, that can literally for whatever file format I'm dealing with, I'm probably finding several options in terms of libraries of packages that I can take to actually throw at that file format, crack it open for me and drive value out of that, right. And this is really an interesting use case for us that customers are increasingly putting unstructured data into Snowflake for storage, but obviously they want to create business value, drive insight from that data and they're then driving data pipelines where at some critical part in that data pipeline, you are feeding that file into one of those packages to crack it open and to then pick out the valuable pieces for that and push them further down.

(26:19):

And again, that's connecting different folks with different expectations around their IP landscape and in a very interesting way. So at the end of such a data pipeline, you might actually have someone that's looking at the structured output of that data pipeline that started with unstructured data and is pulling up those structured results in a BI tool for example, right. And then doing BI over that. And all of that is being enabled through connecting all of these pieces with Python in the middle.

Peter Wang (Host) (26:48):

Yeah and I think there's a couple things I want to build on what you said there. First of all, the idea of structured versus unstructured data. It's this thing where, I don't want to get too philosophical about this, but all data has structure. There's just the question is, is it worth the pain and the blunt process of structuring it yet for the particular analysis you're trying to do? And as I hear you describe this, I'm reminded of this great paper from Jim Gray and others from Microsoft Research, and it was on scientific computing, future of scientific computing. It was written in the mid-2000s. And in it he explored this question of why don't scientists use databases? Why don't scientists use databases? Because again, for the listeners who are working from a business environment who come from a business analytics and data science background, if you talk to any scientists, you'll know that in any given area of science, there are just dozens of weird storage formats of scientific data.

(27:38):

In astronomy, it'll be FITS, in other places it'll be TIFF files, in other places it'll be weird HDF5 if you do particle physics, genomics has weird files, other thing, every single area has its own little weird thing. But across the board what you'll find is that scientists almost never use relational databases. And so, Jim Gray posed this question, looked at it, and part of it was the time, right. It's just like a relational database couldn't give you anything that the file system wasn't already giving you, the file system plus a README.txt wasn't already giving you. And then at the end of the day, people were building their own file systems and structures. I mean, some of these file formats like HDF5 are themselves really file systems. So it's not that people didn't want structure, but what they would never compromise is the ability to quickly access the data in columnar matrix numerical form to efficiently get the values they want to compute on.

(28:30):

So it was like, not that scientists hated databases, but just that databases didn't add enough value for the amount of BS to go through, run your data through them. But as you're describing, what we're seeing now, we actually started seeing with Hadoop and any kind of data lake thing, is people dump a ton of structured data, well of pre-structured data before it's been structured, proto-structured data, they

throw it all in there. There's invariably a ton of metadata you want to actually structure around it. So you're using the database now as a glorified file system. And then you have tools like Python or R or other kinds of tools that then go and pull data out of that to almost do a pivot into a different view, a different set of structures that are then implicit in the blob storage. And we saw people doing this with Postgres locally, and now as you're describing it, this is one of the use cases that people are doing with Python Snowpark in Snowflake, which is really, that's a super interesting thing. That's a really interesting thing that you see that happening.

Torsten Grabs (Guest) (29:25):

Yeah. It challenges some of the underlying assumptions, right. So with schema on write, you are not getting to this experience to this use case. This essentially leans on the capability that you can establish schema when you are reading, when you're accessing the file and depending on what your use case is, you may actually establish different schema of different types depending on what your use case is and when you read that particular file. And that opens up a lot of flexibility and a lot of possibility. And it's fascinating for us to see how that actually drives additional use cases for us.

Peter Wang (Host) (30:00):

Yeah. Well and also, as I think about how much of the, when scientists don't use databases or when people do shadow data management, they take the CSV, they do all the stuff. Invariably, there's a lot of intermediate artifacts because their data pipelines then are Bash scripts or Python or Pearl or, God forbid, AUC scripts that they run to generate other things. And so, you end up with a ton of data residue all over the place and usually those data processes are not very well documented and you have reproducibility problems, et cetera, et cetera. But the interesting thing is that when you have the data now in the raw data, pre structured, proto structured, whatever, and you have the ability to write functions in Python to pull interesting values out to compute various derived attributes and whatnot, and then you take that and you add to it, put it all inside this ability to do relational queries, you have this thing that's really mind-blowingly powerful because you could do GIS libraries, you can do all sorts of stuff.

(30:56):

You can do gene mapping and you can find distances between different kinds of genetic sequences. You can do all those things in your Python code and then take the result of that computation and use it in a select statement in SQL. And so, as more of these use cases show up where you're merging the relational query, the relational algebra and relational algebraic approach with this ability to transmute the data in situ, you remove the need to create all of these intermediate artifacts. You remove, then, a lot of the, well, sometimes compliance issues, but certainly the reproducibility and general data hygiene issues that come from that.

(31:33):

You're actually moving the code to data, but you're moving a much higher dimension of code into one central place for the data. And I look forward to seeing customer use cases that look like that. It's just interesting to see what'll come from that for sure. So one thing I wanted to switch gears and talk about a little bit is of course the big news this year that you all have not only made a bet on Python in embedding it in Snowpark, but also on the front end. So the Streamlit acquisition and this empowering users to write business applications and doing all that in Python, tell me a bit about that and about what you're excited about with the technology there.

Torsten Grabs (Guest) (32:12):

Yeah, maybe taking a step back, I think the first observation for us was that besides putting data out there on our data marketplace, which eventually led to the data cloud, we've seen that in addition to putting the data out there, people are getting the most value out of it if you can also bring the corresponding code to the same marketplace so that you actually have a package of data and code to process the data that makes sense or the most sense if you want things in tandem.

(32:42):

So in order to help with that and facilitate that and the intention then also with the data through an application, that is what motivated us to acquire Streamlit. And just to give one example that really resonates with me is thinking about a data scientist. Again, if you as a data scientist, you're probably very familiar with a notebook environment, Jupyter, to produce your models and train them up, right. Now, for the end user who is the eventual consumer of that model, it's typically not reasonable to assume that they are going to install a Jupyter notebook on a container on their laptop just in order to run that machine learning model for inference or for predictions.

(33:27):

If they are a business user, they need a much more curated experience than that. And this is exactly one of those use cases where we're looking at Streamlit to close or to bridge that last mile between the data scientist and the actual end user in an organization. You would wrap that model, that piece of Python code typically, in a Streamlit application and the Streamlit application would drive the interactions with the end user. So for instance, there might be some input parameters required to drive the model. You could ask those through the Streamlit application from the end user. There might be visualizations that the model produces data, adding the predictions onto the data, right. You could visualize that in really powerful ways and also do that Streamlit application. And again, we're looking at our marketplace then as a way to get these models wrapped by Streamlit applications distributed to the end users in the data cloud within your organization or beyond.

Peter Wang (Host) (34:25):

And it seems to me, and this may be a bit of a leading question, but in doing that acquisition and also having this thesis that people will want to not only consume but also make these applications for the data cloud in your marketplace, you're putting a bit of a bet on people writing these kinds of artifacts in Python, that there's obviously nice quality of life things in the Streamlit tool, but ultimately people are writing code to express their ideas and to build [inaudible].

Torsten Grabs (Guest) (34:49):

Absolutely. And all what I just said that applies to the Python community. I'm less looking to us on the Snowflake side. We're not that many. It's really upon the community and all of the Python users out there to do that.

Peter Wang (Host) (35:01):

Right. Maybe I'll build an app and sell it on the marketplace, make some beer money from it. That'd be great. Yeah, so good. So we talked about the rationale and the motivations behind getting Python inside the database, moving code to data, the different kinds of workflows that are empowered by that. And I think one of the things that comes to mind to wrap things up here is thinking about, even though this looks like a SQL database is swallowing the Python, right. By embedding the Python inside it and getting all this Python code to come into it, would you agree with the statement that one could also look at it

the other way, that Snowflake, to some extent, is using SQL as a Trojan horse, so to speak, to get more Python code into other use cases that prior would've been SQL only? Or is that me as the Python guy looking at it from a particular lens, biased lens?

Torsten Grabs (Guest) (35:47):

The way how I would probably phrase it is, bringing Python to the data cloud or bringing the data cloud to the Python community. I think it's really bidirectional and I think that's where we're going to see the most synergies. I don't necessarily expect one to wrap around the other.

Peter Wang (Host) (36:07):

Yes, there's value exchange both ways. There's value exchange both ways. And we certainly see, I think looking historically over the last 20 years, the growth of Python as a language, a lot of it is driven by the fact that it is a glue language, right. It glued together disparate libraries. I mean, the first thing you said about why Python was hard to bring in was all these extensions for Python have native code inside them, like raw C and C++ code that is not guaranteed by any VM. You really have to do isolation at the operating system level, right. And so, Python gluing all these different languages together, gluing these libraries together, being a scripting layer over all of that, that's how it came to prominence, certainly I would say. But now, it is definitely getting wrapped, right. It's definitely getting embedded and wrapped inside a larger environment.

(36:51):

But I think actually, this may be one of the biggest expansion opportunities for it, period. Because the use of SQL, the use of the kinds of data, how serious the production data is that lives in traditional databases and data warehouses, that's just massive. And so, Python going alongside, going inside this kind of environment is a huge growth opportunity for the language. And it will be interesting if we could track this to see how many users whose first experience with Python is Snowpark. But personally, and also I think for our respective businesses, that would be a really interesting number to track. I think that number will be growing quite quickly. I think that number will be probably shocking to all the community members involved. But yeah, any parting thoughts? I've just been spouting off on all this stuff, but any thoughts on your end about where we go next with all this stuff?

Torsten Grabs (Guest) (37:38):

Yeah, no, I'm very much looking forward. From my perspective, I think we are just at the very beginning of this journey and I'm super thrilled by some of the use cases I see emerge from Snowpark and the Python support. And I'm very curious to see what other use cases customers will bring. And we learn a lot throughout that journey and that's what I'm looking forward to.

Peter Wang (Host) (37:59):

Same. I'm really excited about it too. Thank you so much for being such great partners in this process. Really excited for the collaboration thus far and for all the great work we're going to do ahead. So thank you so much for joining us today. Really appreciate the conversation. So much fun.

Torsten Grabs (Guest) (38:12):

Yeah, absolutely. Thank you Peter, and thank you for the partnership as well. Appreciate it.

Peter Wang (Host) (38:16):

Thanks, Torsten.

Torsten Grabs (Guest) (38:17):

Have a good one.

Peter Wang (Host) (38:17):

You too.

Torsten Grabs (Guest) (38:18):

Bye.

Peter Wang (Host) (38:20):

Thank you for listening, and we hope you found this episode valuable. If you enjoyed the show, please leave us a five-star review. You can find more information and resources at anaconda.com. This episode is brought to you by Anaconda, the world's most popular data science platform. We are committed to increasing data literacy and to providing data science technology for a better world. Anaconda is the best way to get started with, deploy, and secure Python and data science software, on-prem or in the cloud. Visit anaconda.com for more information.