oneAPI Code Together Podcast Series
Building Common Standards for Python Data APIs

Brenda Christoffer: Welcome to Code Together. A discussion series, exploring the possibilities of cross architecture development with those at the forefront with those who live it. I'm your host, Brenda Christoffer.

Adoption of Python has been enormous over the last decade. According to SlashData, there are more than 8 million developers globally, who code with Python. It's easy to use, versatile and can be used for AI, machine learning, data analytics, data visualization, and much more. Our first guest today is Travis Oliphant, Quansight's CEO. Travis is also the creator of SciPy, NumPy and Numba; founder and creator of Anaconda and founder of NumFOCUS. Welcome Travis.

Travis Oliphant: Hi. Thanks Brenda. It's great to be here.

Brenda: Next, we have Ralf Gommers, co-Director of Quansight Labs. Thanks for joining us today, Ralph

Ralf Gommers: Great to be here. Thanks for having me.

Brenda: And last we have Areg Melik-Adamyan. a Principal Engineer and Engineering Manager at Intel. Welcome, Areg.

Areg Melik-Adamyan: Hi Brenda. Thanks for inviting me.

Brenda: So let's get started. Travis, why is Python becoming the center of the universe for data science?

Travis: That's a great question. I think the answer lies in some of the reasons why I started to use it. Why many of my scientific friends started to use it as well. First I think those numbers are vastly underestimated. I think it's much more than 8 million. Actually as the founder of Anaconda, we had a distribution of Python with 25 million users. And Anaconda is distribution of Python. One of the reason I think it's underestimated is because many of the ways to estimate developers, focus on a very particular narrow use of computing.

I want to use people that write new programs. There's a lot of people that use computing that actually write script. They think of something else besides programming every day. They think of science, they think of engineering, they think of data. As a scientist myself, I was looking for a way to experiment with MRI images and do rapid development of ideas and algorithms without deep dives into programming. And I found Python. I think other people like that did too. So one of the reasons Python's growing in popularity and continue to grow is because of its accessibility. It's easy to use. You can understand a little bit of logic and then start to use it and write scripts. So I think that's one big reason why it started to get adoption.

Ralf: And I guess nowadays the second reason is that there's just so many libraries, basically for your average data scientist or scientist, or engineer, it's very unlikely there's not a library that does pretty much exactly what you need. And that's an advantage that's, today, even more important than a language, probably.

Travis: That's a good point. I agree.

Areg: And if I may add, as I like to say, quote, unquote, "Python is a language without bones", so it can adapt to any domain and very, very accurately reflect the domain specific APIs, which is very convenient for users.

Ralf: That's an interesting analogy. And no bones, sounds like it's about to die.

Travis: It makes it sound like that, but it's really about adaptability, right? So that was really a cool analogy, I agree, Ralf.

Brenda: So Travis, can you tell us about what were the problems that we're trying to solve through Python?

Travis: There's a lot of problems people solve through Python, its like this vast. One of the great privileges of being a part of the open source Python community has been just meeting so many brilliant people, doing amazing things. From discovering black holes, to gravitational waves, to training and managing risk at the large banks and trying to avoid another meltdown like in 2008, to drug discovery, to mixing. You have figure out how to make your toothpaste not settle on a shelf. Anywhere science is used, anywhere science has application, Python has application. That's not to mention all the other reasons Python gets used as a programming language, web, IOT systems. But for me, and then data science space, it was all about scientists, people have questions about data. They want to get answers to data. They want to do it quickly. And Python became really popular in that space.

So you have the problem, Python likely has a solution. As Ralf mentioned, there's a library out there for it. Many people have joked actually, there's a joke going around, that XKCD it's a comic that engineers like, and the guy's flying and he said, "How'd you get up there?" "I imported antigravity from Python." It's sort of there's a running joke of how many libraries are available. So a lot of problems can be solved. One of the challenges though, early on was as people started to write libraries, how do we talk about fundamental data structures? There's two fundamental data structures that have kind evolved to help people manage large sets of data and do their data management and data processing. One is an ndarray structure. It's basically an array of numbers. Think of it as a matrix. And then you go n-dimensional, three dimensional, four or five, 20 dimensions, that's an... tensor or an interdimensional array.

The other data structure is more like an Excel spreadsheet. If you have an Excel spreadsheet with columns rows, and you have a bunch of data tabulated, they call out a dataframe. And so there has been two popular libraries that have emerged to handle those, one is NumPy for ndarrays and tensors, and the other Pandas for dataframes. That's been the state of the world, or was the state of the world. People would come to Python and go, "Oh, what do I use to do these kind of calculations?" I'll use NumPy. Up until about 2015, 2016, that was something to tell people. But that's not true anymore. We can kind of talk about some of the difficulties that have arisen since then.

Brenda: Now all right, we've talked about the large adoption of Python across the industry and it's an open source language. So can you tell us about why is it popular? What makes it different from other solutions out there?

Areg: Yeah, as Travis mentioned, there are lots and lots of libraries. As soon as you want to solve some problems, the obvious first and very fast thing to get is to get a Python and add some libraries and go from there. And as computing power gets more and more performant, the complexity and scale of problems being solved also gets bigger and bigger. And apparently at some point we are not very satisfied with the performance of the libraries and neither we are satisfied with the Python performance itself. So then Python used to the C bindings, for example, the libraries like TensorFlow, PyTorch, etc., use C bindings or Cython to accelerate things. And it's working well in some cases and it's not working well in other cases. And through these things being added, now you can do very, very fast computations, certain types of computations on GPU or on other specific accelerators.

Now you have lots of contradicting libraries, where to move data, you need to do copying, converting. Libraries are using contradicting or incompatible APIs. And it becomes very confusing for users to just do the thing that they want to do. Quickly compute some mean of the dataframe. And it's not that easy. And that's when we're discussing with Travis about all these challenges that happened. And we agreed. It's very, very important to kind of align around some standard APIs that can help users to go from one library to another. And the challenge there was, I remember we discussed with Travis how not to get to this another XKCD cartoon to produce the 15th standard when we only have 14 standards around.

Travis: Right. Exactly. It was a big concern, right? I mean, NumPy actually existed itself in trying to merge two other array standards that arised in 2005, 2004. The American NumArray. At NumPy we managed to create an array that several other libraries could develop on top of. Then SciPy, SciKit, lots of libraries start to emerge. You see the same thing happening now in the larger ecosystem, except we have libraries and systems built on top of different array standards, different data frame standards.

So you'll have MLflow on top of the Spark. Or you'll have PyMC or Pyro. There's a lot of probabilistic computing standards on top of TensorFlow or Torch or NumPy. All of a sudden you start to see different stacks of scientific development. Ralf having been a long time leader and contributor to SciPy understood the value of a large set of collective libraries that sit on top of common standards, and that was very alarming, realize that that collaboration of... many often it's the scientist themselves running the algorithms that they're inventing or creating or know a lot about. And they're trying to figure out what do I build on, do I build on TensorFlow, or PyTorch. And then the dataframe side do I build on Vaex, or Dask, or Ray or Pandas. All of a sudden there's this enormous complexity.

And so we have the idea with Areg and Intel and then working with Ralf, who's also been thinking about this for several years and with the NumPy community of just really standardizing on APIs. That was how we got this consortium started with Intel, which we're really excited about.

Brenda: So Travis, could you tell us about the Python API consortium?

Travis: Yeah. It's the Python Data API Consortium. And the purpose is to really combine everybody who wants to talk about making a common API for array computing and a common API for data frame computing. And it's just gotten started about one and a half years old, thanks to the motivation of Intel and then the sponsorship of others like TensorFlow and Jax and Microsoft and LG Electronics and others. I want to say we're really excited that this is organized to really help bring people together and solve a fundamental problem.

Ralf: Yeah, it might be worth also pointing out that one of the ways we got here with this, we have so many libraries that do array data structures and so many to do dataframe array structures is because, well it was more a social problem than a technical problem, right? Really like when deep learning started becoming popular, people basically wanted NumPy on GPUs with Autograd, but rather than contribute to NumPy or build the library around it... that happened. But you know, then Google came and built TensorFlow, which was like NumPy but slightly different. And then there were other companies that, well we want our own things. So they built PyTorch, and Microsoft built CNTK and it went on and on like that. And then we ended up with like seven, eight different things that were more or less the same, but not really. When we started brainstorming at the end of 2019, how do we deal with this problem? Is that we should avoid writing more code.

Writing yet another library is very likely not going to be the answer. So rather than do that, we wanted to like first ensure we had the social buy-in. So we talked to maintainers of every single library, like. And then basically build a standard that they can adopt within their own library. And that I think is the core idea of this whole consortium that we put together. And yeah, Areg was very instrumental in this initial discussion. So maybe he has some thoughts on it as well.

Areg: Yeah. I think the most important part was the social contract. Cause technically everything is possible and we can do lots of things, but how to align all the participants in the industry to use the common thing. And here you guys were instrumental to bring together all the big participants to start discussing. Because we never discussed that problem in public. And maybe there were some articles, some talks, but we never sat down and discussed, okay, here is what's happening. Here are the problems and how are going to solve this? And that was the most important part from my perspective that we sat down in this consortium and start discussing all these problems and suddenly we found that there are lots of common ground. Of course there are some disagreements and there will always be disagreements. Things are going slowly as any standard committee, but that's fine. As long as we are talking and solving problems.

And that's what we do. And we're happy that Array API standard first version is out. When we see that is being quickly adopted by NumPy, CuPy, TensorFlow... Not TensorFlow, sorry. PyTorch is adopting. I don't know whether TensorFlow has any plans, but I hope so. And this is a huge step forward because suddenly you can have same API working seamlessly from NumPy to PyTorch and that is very, very convenient for the users. And the second, we as Intel are always concerned about the performance and when you have common standards, then you can go and create your magic underneath. And that's what is our main focus. And the same is true for dataframes. We fixed and used the first draft for dataframe interchange protocol. It's already a huge deal. And the next step is the APIs. First of all, it'll allow to have the common parts standardized for the convenience of users. And also it'll allow library developers like us to put target CPU specific optimizations underneath without disturbing anything on the API level.

Travis: Yeah, I think this is really important to understand. There's a couple of points to make. I think one is as a domain scientist or domain expert first, it really starts to be difficult when you have 50 competing standards. And now instead of you thinking about your high level problem and yeah, I want a distributive system and I want the machine to just take care of the library and the system to take care of where it goes, which set of memory is stored in, which computers takes care of this computation. I want to think about a high level array and the reductions and the broadcasting computations that I care about. And then have some system do that. And that was the promise that was kind of single machine, that's kind of what NumPy allowed. And then the data frame side, what's interesting, the data frames been harder for a couple of reasons.

One, the array computing has had a long and rich history. Eraser concepts in the FORTRAN, from the '60s and the '70s. And in the Python space They've been since 1994, a lot of discussions about them. They're inheriting from a rich ecosystem from

APL, to J, to MATLAB. Lots of history. And so there's more acceptance of what an array is and what the right functions and what the right concepts are.

So then you take data frames and they're still fairly new. They come from S and then R and then Python adopted Pandas. Now, do I do reduction? Do I have labels on my rows or labels only on the columns? How do I do groupbys? You know, what kind of type system do I have? How do I adapt that types... There's a lot more complexity, the dataframes that there isn't complete agreement about. And so I think that's one of the challenges is you can get enough of the commonalities so that people can build performing, useful libraries with the API. Once you get there, then you can have a really nice robust ecosystem. And I believe we're actually closer. There's so many people using dataframes and there aren't that many people using data frames who will spend time building the data frames, working on them. I don't know if Ralf, you have comments on that since you're in the middle of it every day.

Ralf: Yeah, for sure. I mean, that is hard. And I think the big difference is deep learning put so much energy behind array and tensor libraries and something like that does not exist on the dataframe side. But it's also interesting to think about timescale. Like you said, Pandas is from 2008, 2009, right? So it's 12 years old. Areg just said, "It's going slow with the standardization", which is true. We would like it to go faster, but it's been a year and three months since we really started putting this thing together, right?

And we have a working version, an implementation in three libraries now. That's probably going to be released by the end of this year. So, end users can have it in their hands. If you think about a library, like pandas, it's still in many ways immature after 12 years. The other comment I often hear is like why don't we just take the API as it is, and then build like a compiler to make it faster. We build a delayed evaluation system on it to solve all the problems. That takes at least five years. If you look at... We still do not have a good JIT compiler. Numba is the best one, I guess. And it's nine years old by now.

And there's still lots of things it cannot do. If you look at the... Jax has its own JIT compiler. PyTorch has its own JIT compiler and they're all full of limitations. And there is a ton of effort being spent on them.

Areg: Yeah. Ralf, you completely correct. When I say it's going slow. It's not because I think it's going really slow. I want it to go faster, because we have like huge experience with the MPI, OpenMP, etc. So we're with lots of frustration there. but I understand that actually we're moving pretty fast and the things that we accomplished for this year and a half is amazing. For the first time now we have common APIs in arrays and common protocol for dataframe, I think that that's huge achievement and I'm very, very happy to see it happen. And I'm very anxious to see how fast it can be adopted so we can do other things and give more and more good stuff to our users.

Ralf: Yeah, I totally agree. And the proof is in the pudding there, right? Because in the end it's not the implementation in NumPy or PyTorch that matters. It's when libraries start building on top of it. So all of a sudden, parts of SciPy and Scikit-learn, start to work for GPUs. That's when the users should really get the benefits.

Travis: Absolutely. 100 percent. That's what's sometimes hidden and not seen, is how much work goes behind the scenes. And all of us have been in technology long enough to understand that we are building on the shoulders of giants already. And some the things we take for granted, they have become because of the efforts of so many years ago. I think it's very exciting. Just kind of understand the human timescales and the complexity of several hundred projects that are actually now interoperating. but I'm really impressed that we've got not only a team, but a community engagement and interest that's continuing the conversation. So I really, really appreciate the work Intel has done. And Areg personally has done to make this a reality.

Brenda: So you've talked a little bit about the background and the work, the importance of standards that the Python API consortium is working on. Ralf, where do you see things heading in the future?

Ralf: Oh, that's a great question. Yeah. We started to think about what's the next step. Now we have a first array API. We also have a data frame interchange protocol, which didn't exist before that you can turn any kind of dataframe into another kind, like Pandas to Vaex, Vaex to Modin, Modin into cuDF and so on. So that really needs to mature in the next six to 12 months. Then there will be like a more complete dataframe API that's developer-focused. Is one of the things we realized, for arrays, developers and end users kind of want the same thing. For dataframes is not really true. Developers want explicit, predictable APIs, while end users want things that are a little more magic and kind of just do the right thing, which is not the same thing for dataframes.

But beyond that, there's lots of wishes like both up and down in abstraction level. For example, we spend a lot of efforts working with the team that builds DLPack, which is a low level interoperability protocol that also supports like more heterogeneous hardware. And there's lots more that can be done there, for example, in supporting distributed libraries. So right now we don't have specific concepts for supporting distributed libraries. And I think that would be a big topic. All right. Maybe you have other thoughts on that.

Areg: Yeah. As you mentioned, if we look to the future, like three to five years, the ideal thing that we're going there that having interoperability compiler capabilities for doing lazy valves and supporting the heterogeneous processing is one of our main focuses. Because I think that eventually what we're doing here will become kind of standardized APIs and procedures for converged data platform that will allow data scientists to just increase data, do whatever they want, get results. Especially in machine learning and data processing, of course. And this will require lots of support, both on APIs, standard tensors representations, standardized APIs top down, and all this stuff that kind of naturally goes into this Data APIs Consortium umbrella. Travis, do you have any addition to this?

Travis: No. I like to see tools that keep enabling the occasional programmer. I think Ralf talked about the difference. Now I already began the show by saying people are underestimating the Python users. I've heard that for years, and I think because they're using predictive techniques and approaches for developers. And there's a whole smattering of users of Python that are not traditional developers. They're experts, domain expert scientists. And I think ensuring that the things we produce enable that group of people to keep doing their job quickly efficiently with modern machines, and as machines are growing, GPUs and distributed machines. So how do we extend this so that library writers can keep writing at a higher and higher level. And then just making that much more accessible. So I like what I'm hearing, I'll be constantly encouraging and pushing on that end for end users.

Ralf: It also... I think, is going to evolve a little, like what kind of people are writing these libraries, because if you see how it started, it was really like physicists and bio mechanical engineers, and mathematicians writing these libraries that now everybody uses. And they did a really good job, but they kind of broke a whole bunch of software engineering rules in very fundamental ways. And what you see now is, hardware is becoming more complex. The stacks of software people use are much deeper. So there's got to be more software engineering that happens at the lower layers. And that includes like foundational libraries, like NumPy, like SciPy, like CuPy that people use every day, that have to obey software engineering principles and become more extensible. And at some point that then makes it easier for higher level libraries. You have NumPy and you have say SciPy and you have Scikit-learn. And then you have multiple levels above that, which is really where most scientists spend their time, and engineers. Those are the ones that we have to enable to work with distributed code, with GPU code, etc.

Areg: In addition to what you said, Ralf, I think that all this work that we're doing and all these layers of abstractions defining and designing should help to drive requirements and drive the language changes. Because some of these changes actually should go into Python itself. And eventually I think that in maybe five years span, I'm optimistic, some of these changes can land into the Python.

Ralf: I'm just optimistic. I try to keep everything possible outside of Python because for starters you have to write a PEP. Then you have to wait a year before it's implemented. And then you have to wait three more years before that version where it lands in Python becomes the lowest version that your library supports. So basically it's a five year process. So you have to really, really need it in Python for that to be the right plan.

Travis: Yeah. I agree with Ralf. That's one of the reason NumPy wasn't pulled into Python from day one, even though that was a motivation to call it for NumPy to get to Python 3, but that policy of Python releases has been established for a long time. It has that cadence. And it absolutely can limit. But I do think, Areg, you're right. We got to identify what those key things are. And if we can get momentum for them and it really is... Python community, it's got a lot of developers in it, not just the domain experts, the scientists, the people where often and I like to hang out with and talk to. It's other people we like to hang out to talk with. They're caring about recursive trees and compiler issues and parsers and stuff that I use to start my mind every day.

Brenda: So we're about out of time. Travis, could you provide some insight on where can developers go to learn more and how can they get involved in the Python Consortium?

Travis: Absolutely data-apis.org, data-apis.org, or they can DM me @teoliphant in Twitter, or email me, travis@quansight.com. They can reach out to Ralf and I'll let him give us his contact information. Love to hear from anybody interested in moving this sort of thing forward. We're actively looking for people who want to work on this sort of thing and are eager to hear from folks that want to help.

Brenda: Ralf, do you have any resources for how developers can connect with you?

Ralf: Yes. So first of all, data-apis.org is great or the GitHub repo. Everything's public. You can reach me on Twitter, ralfgommers, or email or GitHub, easy to find.

Brenda: Areg, do you have any other resources to add from the Intel side?

Areg: From Intel side I think... Worth to mention that these APIs are being adopted by Intel AI kit, which is available on Intel AI side. Because Travis mentioned, we're very interested to hear folks will go try and go to Intel AI and download the AI kit, give us feedback.

Brenda: Great. So Travis, thank you so much for joining today.

Travis: Thanks Brenda. It was a pleasure to be here.

Brenda: Thank you, Ralf.

Ralf: Thanks for having me. It was fun.

Brenda: Thank you Areg. It's a pleasure having you here.

Areg: Thank you, Brenda. Nice to meet you guys. It was really interesting.

Travis: Thanks Areg. Thanks Ralf.

Areg: Thank you.

Brenda: And thank you to all of our listeners for joining. Let's continue the conversation at oneapi.com.