Oral History of Guido van Rossum, part 2

Interviewed by:
Hansen Hsu

Recorded March 29, 2018
Mountain View, CA

CHM Reference number: X8483.2018

Hsu: Alright, it is March 29th, 2018, I am Hansen Hsu and we're back today with Guido van Rossum.

van Rossum: Hello.

Hsu: So we're going to pick up where we left off. We were talking about Python's change over time. So going back into that topic, maybe could you describe or summarize how Python has changed over time, since the beginning to now?

van Rossum: Wow, since the beginning… well, the language has expanded. It has many features that I hadn't even thought of, that I wasn't aware you could do with language design, to be honest, in the beginning. It has a much larger library—it has a much larger community. The philosophy of the language has also changed quite a bit. I'm pretty flexible, I like whatever users are doing with the language obviously. My original idea was for Python to be in between shell scripts and C programming because that was—in the early '90s, that was the bread and butter of my work… my programming work, and these days there are many other ways that Python is used where it's either an alternative to Java or Go or it's a tool all by itself. Scientists use it to drive incredibly powerful libraries. So one particular aspect that also has changed quite a bit is how the language is managed and how the—how changes to the language happen. In the early '90s, everything was basically me. I think I already mentioned that I was like the mailing list administrator and I reviewed all the patches and I answered all the email questions. There was also—it was very easy for people to get new things into the language. If they had an idea and I liked the idea, it was basically in, and we didn't care that much about backwards compatibility because the user base was small enough that if I changed my mind about some detail of the language or even some big very visible part, that was okay, people would just update their code or write new code. Nowadays we have backwards compatibility, we have—there's a high bar for proposing new features. We can't break any existing code and now there are billions of lines of Python code in the world that could possibly break if we add a new feature even or change something in a very subtle way and even if all the Python code I have written in my life would never be broken by particular change, chances are that someone else's code will break if we are changing existing behavior. So much more thought goes into that and it's—it's much more a community process. Many times changes go in, I don’t get involved at all. It's other people, you might call them lieutenants, or they're just the volunteers who are active as core Python developers who have been around the longest who often make the call, "This is a good idea, this is a bad idea," and then there's a whole cycle of development where even a good idea doesn't always get accepted for any number of reasons and that's—but that's a long process nowadays and that was—I think we introduced the idea that there was a formal process in early 2000 when we introduced the Python Enhancement Proposal process and so now we have PEPs and they're used for all sorts of things but the basic use of PEPs is to have a way of getting changes into the language that acknowledges the need to think things through and debate things and have a good motivation and a careful specification.

Hsu: Right. Talking about backwards compatibility, one of the major breaks in backwards compatibility was Python 3.0. Why was it necessary to break compatibility in that instance versus previously?

van Rossum: Well, why was it necessary? I don't know that it was necessary. We decided to break compatibility and this was a collective decision of the core developers at the time. The reasons that led us to break compatibility had to do with certain types of changes where we didn't have—we didn't feel there was a gradual backwards compatible way to introduce those changes and at the same time, we really did not want the language to be stuck forever with certain mis-designed features. Unicode behavior was probably the biggest one and the one that would have been the hardest to change gradually or in a backwards compatible fashion.

Hsu: How well does Python support concurrency or functional programming?

van Rossum: For concurrency, Python's approach is mostly based on operating system threads which is not a very advanced mechanism. It's more of an easily available mechanism. We introduced that I think in the early '90s. We—myself and a small number of people actually around me at CWI were learning about threading models on hardware that was available to us at the time that were—some workstations had multiple threading built into the operating system and looking back I think we were fairly naïve. And so Python's multiple threading is in some sense fake because of the global interpreter lock—the GIL as it's called. It effectively means that you can create multiple threads and they correspond to operating system level threads which is good because you have—you have all the behaviors and properties of OS threads with one exception. You cannot use multiple CPUs simultaneously and in the early '90s, multiple CPUs on a single machine was like—oh, that was very fancy—that like the workstations we actually had had a single CPU and maybe for very advanced applications, some hardware vendors could sell you multiple CPUs and they were super expensive and they had worked hard on the operating system support for that but that was not really the focus of Python because when are you going to use multiple CPUs? At least this was how we were thinking at the time. Well, when you have something that is incredibly computationally intensive, well, you're probably not going to write that code in Python in the first place because if your core for loop is written in Python, it's just going to be slower than when you write it in C. So that kind of parallelism was really not—it didn't seem interesting to us at the time and most of the hardware we ever saw and that we expected to be using Python on did not have multiple CPUs. But it did have multiple processes and several OS vendors, I think Sun and Silicon Graphics and who knows who else—companies we haven't heard of in 20 years—were adding multiple threading to their operating systems and we saw that there was a certain advantage, even when using multiple CPUs was not the point of your program, you could do multiple independent I/O actions. You could like have one thread manage the user interface and another thread talk to the network and maybe a third thread talk to a database service and those kind of use cases for threading we thought were very useful. Also, we were just—we were learning about this and somehow an easy way to learn about all sorts of things was to write a Python extension to support that new thing because then it was much easier to play with it. I remember teaching myself how [BSD] sockets worked in the same way. So like with so many other things in Python, we didn't want to spend the time to—and we probably also didn't have the technical finesse to be honest, at the time, to take our existing single-threaded interpreter and make it truly concurrent—make it so that if two threads were updating the same database—sorry, dictionary, which is an in-memory data structure in Python, that—the results would be reasonable and you would never have a crash or a deadlock just due to that. I mean, obviously if one thread is changing a dictionary and the other thread is looking at it, the other thread may see—may or may not see exactly what the first thread put in, it may have to wait a little bit. That was okay but we—so we didn't want to add concurrency locks to all [of] Python’s fundamental data structures, most of which—many of which are completely mutable, like lists and dictionaries and a lot of things are built out of those like regular classes and instances and modules are all built out of dictionaries. So they're also fundamentally mutable.[1] So we came up with this quick hack where there was a lock that was essentially protecting all [of] Python's data structures, with the exception of I/O buffers and we—because we had control over the actual I/O operations in Python, we designed a set of macros and we did some work in that area so that if it actually went out to the network and you were blocking for a receive or a send operation to complete, another thread would be able to run and just to—I guess to keep it interesting or to follow the most common threading model we found in other languages, we also made it so that if you have had multiple threads that were all just doing CPU operations, the interpreter would just occasionally hop between different threads so that they would all get their share of the single CPU. And this is like— at some level, the earliest Unix kernels had a similar threading model where different pieces of kernel code had independent control flows but there was only one CPU. The early versions of Unix didn't have multiple CPU support and I think in many cases, the threading notion was not exported out of the kernel, out[side] of the kernel it was all processes which is a heavier abstraction. But so, we did a similar thing as kernels do which is, when you're waiting for I/O is a good time to let someone else do some work and then we had this added thing where we said, "Well, if nobody gives up the CPU, we'll just switch occasionally for you," and that is called preemptive scheduling, and this has always been controversial—it's been very useful. The use case I described like one thread taking care of the UI, another thread taking care of the network, or for example, 10 threads that are all doing network operations where the network is just sort of slow where you're going out way far on the internet. You can do a lot of useful work using Python's threading model. It's no good however, to just speed up a slow computation by adding CPUs which—well, I certainly did not predict that at some point, Moore's Law would run out or at least we couldn't just buy computer chips that were twice as fast every few years. We would—suddenly hardware manufacturers were giving us computer chips where there were multiple cores and you could execute multiple instructions concurrently if you wrote your software that dealt with that and that—I'm actually mostly glad that we didn't jump on that bandwagon too early and for specialized cases like numerical Python, NumPy, the libraries can take care of allowing multiple CPUs to do different work concurrently. The libraries have to be aware of how Python's internals work and release the GIL and things like that.

Hsu: Let's move on to talking about users of Python. So initially, who were the key adopters or what organizations were the key adopters when it was gaining acceptance and what key adopters helped the Python gain acceptance?

van Rossum: That's tricky. Often I had no idea even which organizations people were working for who were contributing to Python. I think I was quite surprised to find in 1994, that there were people at NIST, the National Institute for Standards and Technology in Maryland, who were using Python and interested in doing things with it that I had absolutely no understanding what exactly they were using it for. I don't know how far that particular project went but that's an example of research labs in a sense, adopting Python or at least trying to do things. In the late '90s, very influential was Lawrence Livermore National Labs in Livermore here across the bay, which adopted Python as the standard scripting language to drive their numerical computations and they were ahead of things like NumPy in a sense. They hosted one of

[1] [Interviewee's footnote] In fact, even immutable Python objects have a mutable part: the reference count.

the very early Python workshops. I forget when that was but it must've been in like around '96 or so[2]. Another place—yeah, also, now I think of who was hosting Python workshops in late '90s, USGS, the Geological Survey, there were people there—again, scientists, who were adopting Python for—to drive their numerical processing code and they were not writing numerical algorithms in Python, but they were using the fact that Python was easily extensible so they would write a small amount of bridge code that linked Python on one hand with their Fortran or C++ or C code on the other hand and they found that that was a very effective way to let scientists experiment with their data and that development has never stopped really. That world has grown bigger and bigger and now we have like a lot of data science happening in Python.

Hsu: What about early corporate users or any major applications that were written in Python?

van Rossum: I think in the early web days, Python was also popular with startups who were developing web applications. Actually, yeah, a core developer at the time named Greg Stein worked for a small startup and they were doing an e-commerce platform and he somehow convinced all his colleagues or maybe he convinced the director of engineering or whatever, he convinced everyone that they should write their code in Python using like Python's interfaces to databases, interfaces to the web, standard library that made all these things easy and basically they could concentrate on the functionality and they won in essence, a race to be first to market or at least first to beta and then the very interesting thing happened, they got acquired by Microsoft and suddenly Microsoft owned and was maintaining a huge bundle of Python code—and this was in the late '90s.[3]

Hsu: Was Yahoo Mail also Python?

van Rossum: Oh, now you mention it, yes, that was also a big one. I forget if that was late '90s or early 2000s but it might've been early 2000s.[4] They were definitely very big Python users. Yeah, thanks for reminding me of that. I had totally forgotten—and I don't know if it was just Mail but Yahoo Mail was the big thing that took off at the time.

Hsu: We've previously talked about how your work with the Python community caused you to move to the US, so I'd like to go back to that and talk about you know, your career trajectory from that point until now. So starting—so you moved to the US in 1994, partly because you had gotten this invitation from NIST?

van Rossum: Yeah, so I spent two months about in Maryland as a guest of NIST, then I went back to CWI but I basically had a job offer in my pocket and in the first half of '95, I moved back to the US to work for CNRI. That was Bob Kahn and Vint Cerf's company. I think Vint was already no longer much involved with the management, so at the time it was maybe just Bob Kahn's company but it was Bob and Vint's creation, Corporation for National Research Initiatives—and they were very cleverly located a short drive

[2] [Interviewee's footnote] The workshop was organized in June 1996: https://legacy.python.org/workshops/. The USGS workshops are also listed there (both 1995).
[3] [Interviewee's footnote] Actually, 1996: https://en.wikipedia.org/wiki/Microsoft_Merchant_Server.
[4] [Interviewee's footnote] Actually 1997, as RocketMail by Four11: https://books.google.com/books?id=eulODwAAQBAJ&pg=PA223.

away from Arlington where I think DARPA or the NSF had their headquarters and we—CNRI got a lot of funding through those organizations for various very interesting, sometimes internet-related, research and Python was supposed—two programmers there actually looked at Python and thought that it would be a really good tool for a project that they were tasked with, that was part of some research and they met me at that NIST—NIST also organized the workshop in November I think, '94, for essentially local Python users.

Hsu: Oh, okay—the first Python Workshop?

van Rossum: The first Python Workshop[5]—and so these two guys at CNRI came up with the idea that Python would be a good language—they needed a language that had a virtual machine and at the time there were different ways of interpreting that word and an interpreter—like the JVM essentially, and so Python has a similar concept inside the interpreter and they thought that that would be a good—kind of technology to use in their research project and it turned out that that particular idea never really went anywhere. Bob Kahn was very proud of it and I remember going through all the motions with lawyers for a patent application but the technical idea behind it was never—never ended up being all that useful, but Python grew as a result of that and CNRI also started using Python for other research projects. So several other projects that were also funded by DARPA or NSF or some other funding source that CNRI had figured out, were also using Python to write big systems essentially and a few of those people also ended up as core Python developers.

Hsu: So what were you specifically working on?

van Rossum: I think I was mostly one of the programmers and architects for the code.

Hsu: The code that was part of this project that Bob Kahn?

van Rossum: Yeah—so there was a lot of programming, there was also a lot of thinking through the architecture of the whole system because it was a very networked kind of application. Yeah—it was—I think it was a funny situation because as far as I know the project, it was called KNOWBOT, K-N-O-W-B-O-T—that project had a manager who was really an engineering manager, they run meetings and do the hiring and firing and make sure that the reporting to upper management is okay, that kind of stuff, and we had a number of programmers—myself and Barry [Warsaw] and Roger [Masse] and maybe one or two more actually [Jeremy Hylton, Fred Drake, Ken Manheimer] and I don't believe that we had like researchers on the staff and I'm actually very surprised because most of the projects that were happening at CNRI, there was someone in a PI role—what is it, Principal Investigator or something—yeah, and I think for a different project that we did later after the KNOWBOT funding ran out, I was promoted to PI but I found that it wasn't really a role I was very compatible with. I think for KNOWBOT, Bob Kahn himself acted as PI. But he—yeah, he was running the whole organization so we didn't have all that much regular intense interaction with him, although every once in a while, he tried to micromanage what we were doing.

[5] See footnote 2.

Hsu: So you also wrote a DARPA funded proposal called Computer Programming for Everybody?[6]

van Rossum: Yes, that was—in some sense, that was driven by the need for funding because the KNOWBOT project was not sufficiently successful that [a] followup funding round was feasible and so we were trying to find other things that we could do with what we had learned and by then it was obvious that Python was actually a big hit. It might have been a sleeper hit but it was—it was clear that that [some] sort of work on Python itself was a good use of our time and we were trying to find a way to turn that into a research project and in order to qualify for DARPA funding—I think we went for DARPA funding—in order to qualify, you have to present it as research because they won't fund a project that is basically—well, there are a number of existing ideas and we're going to implement that and maybe we're going to implement it slightly differently than before but implementation work or like improving a programming language was rarely a fundable thing. So what we found was that Python had interesting prospects for education and I had this vision of people, sort of kids learning Python in school which at the time looked pretty farfetched because there weren't even that many computers in most schools, but that's where the Computer Programming for Everybody idea project proposal came from. I'm also pretty sure that it wasn't just me but that other people in the Python community and there were like workshops every year and other times that some people got together, there was some talk about that where people said, "Oh, Python is such a cool language—it's so simple to learn, why couldn't we use that in education? There are people who are currently teaching C++ or Pascal and C++ is way too complicated for all but like a very small number of brilliant high schoolers and Pascal does not really have much of a future as a technical skill so you can say, well we're not teaching them programming so that they have an employable skill but we teach them programming so that they learn a certain way of thinking and they know what's going on inside a computer and sure, but nevertheless, maybe if you teach them Python instead of Pascal, you can—they can focus more on that learning experience and less on where the semicolons do or don't go— and that was—it turned out for me personally, it was a very frustrating experience. I felt like I wasn't very competent in writing a research proposal because I had—even though I had worked at the periphery of academia, I'd always been in the programming role, not in a PI role. I had not written a PhD thesis, I had never really published a paper or cared much about citation indexes or even how to format your references, for that matter, and so I felt that was not—I wouldn't say I was pulled into it kicking and screaming but I was pretty skeptical about how that would work out and as it turned out, that with a lot of help, we got a small amount of funding—enough to keep the team together and then it turned out that I had completely misunderstood how research funding actually works and only after we had—we thought we had gotten the grant, it was explained to me, "Well, okay, now you have to do more talking to people and writing proposals and convincing them that they should actually give you money," plus the money you've been allocated is not enough, you have to try to get more money but even the funds that you've been allocated, it came from DARPA which means that there had to be some defense organization that was doing its own research that thought they were interested in our potential results and wanted to fund us and there was a whole separate set of negotiations, meetings with people from the West Coast that worked for some lab in San Diego I believe, naval research if I remember correctly. So that was a big downer because I hadn't expected that I had to go to cocktail parties and chat up top researchers who had their names in all the CACM publications and things like that. And—plus I had to manage a team

[6] [Editor's footnote] To access the text of the proposal, please see https://www.python.org/doc/essays/cp4e/

which also turned out the engineering manager side of that, I didn't really enjoy that much. And then on top of all that—so that was—I don’t think I was super happy with that. On the other hand, I was very happy because we did actually have continued employment and we didn’t have to work on the Knowbots. We could spend all our time on doing stuff with Python, which we thought was the most fun and which had the most rewards, because whenever we released a new version of Python, users would fall over us and say, “Oh, this is so great. This is amazing.” There’s a really positive feedback loop there. So, the team was very happy in that sense and then another dark cloud started looming on the horizon and coming closer very quickly, which was licensing issues. Because, by the end—so, I started at CNRI in ’95. And, so, through ’99 or so, we did regular releases of Python and every once in a while we bumped the minor version number and it went from 1.3 to 1.4 to 1.5 and sometimes it was the micro version number, 1.51, 1.52. And Bob Kahn started getting uncomfortable with the fact that there was an open source license on there that didn’t grant CNRI any rights into the code, because it was like, “Well, CNRI has employed all these people and secured the funding for all this work and, well, there was mention of CNRI in the license,” but the license—it was a very liberal open source license. I think I explained that in the previous session that was then the MIT license. And, so, Bob Kahn started getting uncomfortable with that and, at the same time, open source became a thing, the term itself became popular and there was the Free Software Foundation on the one side and Open Source Initiative on the other side and there was, like, Debian-Linux was a very strict open source project. At the same time, other projects at CNRI, not Python-related, had been released under licenses that had been, I don’t know, made up by CNRI’s lawyers that were very clearly not open source and not free software or libre software. And I think that CNRI, at the time, underestimated the significance of that and thought, “Well, we’ll—” there was this license for, I think, the digital object identifier system. And there was software that was written by some people at CNRI that implemented all the infrastructure and the license was essentially not free. It was like, “Well, if you’re just playing around with this or using it strictly for research, you can use it for free. And otherwise let’s talk and you have to pay for a license.” And we were very worried that CNRI wanted to change Python’s license to a similar non-free, non-open source license. And, so, CNRI did require us to change the license and I felt like I didn’t have legal standing to refuse that, because they did fund my time and several other people’s time for several years and surely that amounted to at least some significant fraction of the code that we were releasing regularly. And it was hard to tell where the boundaries were. Like, if a file originated at CWI in Amsterdam and then was modified or a bug was fixed in it during my time at CNRI, who owned the copyright to that file? Does it really matter? But there were some worrisome things going on and I remember a very emotional meeting where, like, the Python team and Bob Kahn and his lawyer and probably some other people really had a big clash about should Python stay open source or should it not? Or can CNRI just change the license and what are the consequences? I was really scared at that point that the next version of Python would have a license that was not open source and that, as a result, Python’s users would say, “Oh, we can’t touch that! Because that’s commercially too risky!” because there, like, people in the Python community who had built their business around Python by then. There was a small company named Digital Creations, which later became Zope, everything they did was Python. They were big contributors to Python as well. Those contributions were not always directly acknowledged in the source code. And they had thought, “Well, Python is clearly open source.” That license was easy to read for a layperson. And the worry both with me and the Python core team at CNRI as well as with the user community was that somehow CNRI would suddenly come and say, “Hey, Digital Creations,” or whatever other company—there was a guy who did computations for pricing for aircraft, like big commercial airliners. Turns out there's a super complicated financial model that determines what the payment schedule is if you buy an airplane. And he had written all this code in Python and that was his business. He had, like, a few people worked for him, but he had written all the code, another example of a really cool application in Python, and the worry was that CNRI would suddenly come and say, "Hey, you are using this software. That is our property and you have to license it from us." And there was like no reasonable price. What would CNRI be charging if they wanted to charge? Or just the threat that they could come after you with all their lawyers was—that worried me a lot. And, fortunately, I think with some intervention by Eric Raymond and Eben Moglen, the Free Software Foundation's lawyer, [a] compromise was reached where CNRI did put a license on Python that was much longer and wordier and harder to read[7], but was still approved by the Open Source Initiative and by the Free Software Foundation as open source and free software and compatible with the GPL license, the GNU Public License. So, that was pretty scary and that happened concurrently with a different development which was—I think we were tired of the DARPA funding story and the threat to the Python license story and I had been approached by someone who had a startup that was all open-source-based and he desperately wanted to employ me. And I was pretty green where it comes to start-ups at the time, because I had worked for Dutch government-funded research organizations and a U.S. research lab that was also essentially government funded, even though it was a little less direct. And I had no idea what was going on in the world of venture capitalism and start-ups and, plus, we were in Virginia and all the action was happening in California and I—well, we had some Python users there, for example, Greg Stein and his team for e-commerce[8], and we knew that there were all sorts of interesting things going on there. And there were conferences: I remember going to early O'Reilly Open Source Conferences. And, so, somehow, a little on the late side, I got the start-up bug and I thought, "Whoo, wouldn't it be cool to be done with this whole CNRI slog," and "I could just be doing what this guy with his start-up says he wants me to do," which is just make Python better and, so, I decided to give that a try. And—

Hsu: This was BeOpen?

van Rossum: This was BeOpen. Yeah. And I convinced three of my co-workers at CNRI to also join me. There were a few others who decided to stay at CNRI, they were not so into risks. But myself, Barry, Jeremy, and Fred decided that we would start working for BeOpen. And I remember I called a meeting with Bob Kahn to talk to him to, basically, present him with the news that we were leaving and he was like in shock. He also immediately realized that I didn't know what I was doing—

<laughter>

van Rossum: —but also he realized that he couldn't stop us, because we had—without letting him know, we had negotiated contracts and start dates and all sorts of things and we would just stay in Northern Virginia where we all lived, except for Barry who came from Maryland, but that was just on the other side of the Potomac River. So, Bob saw suddenly four of his most valuable developers depart and one of his

[7][Interviewee's footnote] https://docs.python.org/2/license.html
[8] [Interviewee's footnote]The eShop team that was acquired by Microsoft.

projects implode, because he—yeah, I have no idea what happen with that project, with anything we were working on at CNRI that—I don't think any of that continued.

Hsu: Like, the Knowbot and—

van Rossum: Well, the Knowbots were dead by then, but the Computer Programming for Everybody research was also just us. There was another project that was about, I think, nano-machines or maybe they had a slightly different—nano-technology, they stayed. But, anyway, I remember in the following days Bob Kahn coming to me and asking, "Well, are you sure you know what you're doing? Have you thought about what would—? I mean, this is a start-up. There must be at least an epsilon probability that it fails! And what is your Plan B in that case?" I think he also tried to make it clear that we couldn't just come back to CNRI at that point, which was completely understandable, and I had never thought—I hadn't—I had never thought of a Plan B, but I didn't think that going back to CNRI would a viable Plan B. But I definitely had vastly underestimated what the probability was that everything would go upside down. And, so, as it turned out, it's a crazy story. By the time we started at BeOpen they had already gone through several very dramatic management changes. One of the co-founders had been kicked out due to some unspecified conflict and apparently was paid out his share of ownership in cash. At least, that's to this day what I believe happened. He was never heard from again. And it was—he was paid off. And if he had been paid off in a share of the company he would have had zero. But, as it was, he was probably just happily retired somewhere on the Atlantic coast. But—and, so, that was one of the big changes that happened and I think in early 2000. And the other thing was that there was another guy there, whose name I can't remember right now[9], but was, like—it was never really clear what his background was and he seemed to always claim things that were too good to be true and he claimed he had an investment company, but he would never go into details. He would never even mention the name of that company. I remember someone—I had a long phone conversation with someone as I was trying to get clarity whether I should really join BeOpen or not, and somehow I [knew] found another open source developer who knew this person and who basically tried to warn me and saying, "This guy fakes it. He's a con man." And he had a particular anecdote where he said this guy[10]—so, this was not one of the founders of BeOpen; this was someone who had attached himself to BeOpen in a very early stage as, I think, the CTO. And his exact title was fluid. It was very strange times. But this guy—so, someone tried to warn me again and said, "This guy is not real." And I was just too—Bob Weiner, who was the founder who had most of the connections with me and who was an Emacs core developer or at least a big Emacs contributor—there was a particular Emacs package that he had written that was very well known and, so, I thought he was solid. And he was very convincing and he was also vague on details, but my inexperience in these matters made it that I didn't really know what questions to ask or how to critically look at what he was telling me. And, so, I convinced three of my colleagues to leave our employer, CNRI. I also convinced another core Python developer[11], who was in Cambridge, Massachusetts, to move to Reston, Virginia, to also join BeOpen and—

[9] [Interviewee's footnote] Don Bell.
[10] [Interviewee's footnote] The unfinished thought here is that the guy (Don Bell) had claimed to have written a Perl script that he clearly was incapable of writing, according to my source (whose name I don't recall).
[11] [Interviewee's footnote] Tim Peters.

Hsu: And BeOpen was headquartered on the West Coast or East Coast?

van Rossum: BeOpen was headquartered I think in Santa Clara. Yeah, we visited their offices once or twice. But, yeah, we were—we stayed in Northern Virginia and we had weekly meetings in my living room where I had one of those fancy speaker phones and we discussed the state of the world with—and the company and our work with Bob Weiner. And after a few months Bob Weiner started trying to get us to do other things that didn't look like were part of what he had hired us for before and we went on recruiting missions trying to drum up business. I think we went to HP and I remember the Motley Fool I believe we were trying—they were a big Python shop apparently in 2000. And we were trying to sell, essentially, the best Python consulting service money could buy, but nobody would buy because Bob Weiner had—he could talk a good story, but he didn't have the salesperson's skill of closing the deal. Well, I mean, that's one thing that became apparent. We saw all these amazing opportunities and nothing ever came of it and he was probably also a terrible judge of character, because a lot of the people he hired for senior positions in the company were terrible. I don't think that in the overheated start-up world of Silicon Valley in 1999 and 2000 people thought that they should vet people they were trying to employ, because there was so much competition. If we didn't snatch this great guy, then surely someone else would snatch him. And that was all that matters. And, so, like, we had a terrible VP of sales. It was just pathetic. He was always losing his cellphone and missing appointments. We went through I think two or three VPs of engineering. And, so, they were, like, every new VP of engineering did a complete pivot of what we were going to do at some point. I think the third one said we were going to develop anti-spam software, which was a great idea—if you could pull it off, that was great, but I don't think we had any particular competency in that area.

Hsu: What was the company trying to do?

van Rossum: Well, yeah, that was also this really vague thing that there was, like, it was—the core business was supposedly an open source portal website and so portal website[s] was [were] all the rage, like, Yahoo started out like that. And, so, there wasn't any search facility or maybe you could search for one keyword in a small database or something, but it was like there was a very fancy homepage with links to all sorts of articles and daily updates and the hope was that people would visit that website frequently to find out news about open source. And we didn't really understand what it—I mean, the Python team was just cranking out a new Python release and which we did successfully! But the core BeOpen business was something completely different and there was a team of web developers and—well, we never knew who they were or where they were and the only contact was through email and we never quite understood who was doing what and why and how and when. And the website—there was basically no way to make money off of that website. And that's—there was all this early Silicon Valley web start-up idea where, well, we'll be losing money, but we'll be making it up in volume. That kind of crazy stuff. And, like, oh, well, first we have to be the most popular open source website and then we'll figure out how to make money. I guess, advertising. Well, there was like—there was a secret section of the business plan I'm sure was just made up and didn't make any sense either and it was something about points. Who knows what that is? So, the company was churning through real money, because it—the Python developer team, it was five really well-paid programmers. And they paid competitive salaries otherwise we wouldn't have left CNRI. Plus, they had the web development team and the hardware for the website and there was no income! So, it was clear that they were getting more and more stressed and that's when we started going on these consulting missions where we were trying to rustle up business and none of it ever worked out. I don't think I wrote a line of anti-spam code or did anything for any of the other potential customers we had. We cranked out Python 2.0—I forget exactly what the timing of that was, but I think that was late in the summer and we were hard at work at subsequent things and the only smart thing we did was that we were aggressively open source, ourselves. Python was open source and the code was hosted—well, yeah, one of the things we did was we moved the code from a private CVS server to SourceForge, which also, I think, at the time didn't have a business model, but wanted to become the big open source service provider for hosting. And, so, they hosted our revision control. We converted everything. I don't know if that's also when we started a bug tracker, but everything we did was out in the open. Lots of people had their own copies, because anybody could just check it out if they had the Subversion software. Oh, yeah, Subversion was written by Greg Stein and a few of his friends. So, the same guy, he had reinvented himself. Anyway, Subversion is now mostly [un]known, because it couldn't make it against the competition of Mercurial and Git, but at the time Subversion was amazing. It was so much cooler than CVS and it worked. And it was fast and it had lots of cool features. It was very reliable. And, so, we hosted everything on SourceForge using SVN and, so, when BeOpen finally folded and I think what happened was that one day our paychecks were all sort of reverted. Like, there was—everybody had direct deposit and every two weeks or twice a month or so, you have your direct deposit in your bank account. And one day someone noticed, hmm, the direct deposit came in and then it was rolled back.

<laughter>

van Rossum: And that was the first we heard about it and after that we got an email from Bob Weiner saying that the company had folded. And after—that was on a Friday and I think the next week we got some email or a phone call from one of the investors who was cleaning up loose ends and I think—I forget where we got this advice, but somebody who knew about these things said, "Okay, guys, that was that. There's nothing you can get out of BeOpen. They don't have a penny, but there's also no way they can ask for their computers back." So, every one of us had a really nice home computer set-up. Like, we had big—the beefiest PC you could buy with a big monitor and a fancy keyboard and Ethernet cards and modems and, I don't know, some kind of zip drive and some—well, everybody got what they wanted. I think some of us had laptops. And the advice we got was, "Well, okay, just keep your hardware." That's your reward for being very suddenly unemployed. And then we started—well, this was obviously kind of panicky, because we all had families and we needed employment, and we ended up negotiating with two small companies that were actually doing real stuff in the Python world. One was ActiveState, which I think had become big on doing a Perl port to Windows, but was also branching out to Python and had a lot of interest in Python. And the other was Digital Creations, which was just about to re-brand itself as Zope. I think the software they had written was called Zope and that was just a nonsense word they had made up. And Zope was successful and they were changing the company, which was Digital Creations, into Zope.com. And, well, lucky for us, Zope.com came through—or Digital Creations, I think at the time still, came through with a competitive offer to take the entire team and we—oh, yeah, we named ourselves PythonLabs. That was also the internal team name at BeOpen. So, the five of us were PythonLabs. And all of PythonLabs stayed together and joined Digital Creations and we would do—part of our time we would continue cranking out Python releases and part of our time we would be working on Zope. And Zope did actually have real customers who were paying the bills. And I'm very happy that we didn't choose ActiveState, because they were in Vancouver, Canada. Plus, their CEO and founder at the time—dang, I'm blanking out on the name.[12] He was quite a character. And ActiveState has had—I think the next few years were rough for ActiveState, although they are still around and they still have an "Active Python" distribution. It turned out that things at Zope were also kind of rough, but it was a really nice company to work for and there was real people and real work and I had a very good time there and it was very different from BeOpen, which was basically built on bullshit.

Hsu: And Zope, where were they located?

van Rossum: Oh, yeah, so Zope was located also in Virginia in Fredericksburg. And, so, we also—we lucked out: We continued to basically work from home and once, maybe twice, a week we would all pile into a car and drive to Fredericksburg on, I think, I-95, which wasn't too bad.

Hsu: And, so, you spent half your time working on Python itself and the other half working on Zope's own systems.

van Rossum: Yeah.

Hsu: So, what was their primary business?

van Rossum: So, Zope is a web framework. It was very influential in those days in the Python world. It was all written in Python, with the exception of some networking and database layers that were Python extensions. But they—a lot of the database layer was also open source and there was, like, only a little bit of secret sauce. Their main customers were a network of newspapers that are mostly in Virginia and other East Coast states, I believe, that had still—they were very much live newspapers, but they also very much wanted a web presence and Zope hosted their websites and wrote the applications to run the websites. The newspapers themselves just entered the content and it was not like a modern newspaper website where the entire newspaper is online. It was I think mostly classified advertisements and a few other specialized things that were very suitable for putting in computers and accessing online.

Hsu: So, then you were there until 2003-ish?

van Rossum: Yeah, that's right and I think by then Jeremy Hilton had decided that he wanted to move back to Pennsylvania.

Hsu: So, he was part of the—

van Rossum: He was one of the PythonLabs guys. He left—I think he first moved to Pennsylvania and worked from there, but that didn't work very well. And then I think he joined Google. He's still there. I don't

[12] {Interviewee's footnote] Dick Hardt.

think he did anything in between and it had to do with family and his wife's employment. So, he left. And Zope started having some trouble getting enough business and I got a phone call from a guy who was doing another start-up in California!

<laughter>

van Rossum: Believe it or not. And I had learned my lesson and I did more research but I did let them convince me to join but not with the whole PythonLabs team. They had a very different role for me in mind. So that was a place called Elemental Security and they were in San Mateo. And the guy who called me was Dan Farmer, who was an internet celebrity at the time for—or at least a hacker idol because he knew how to break into any system. But he used his powers for good and he had written this, together with another guy, I think a Dutch guy[13], actually, later they wrote a book together on forensics, which was really good. But they had also written software that probed a system's network defenses, and depending on your mood, the name of the software was either SATAN or SANTA.

Hsu: <laughs>

van Rossum: And people hated that. I think it was just a bunch of Perl scripts that just had a huge database of known vulnerabilities in many different systems and would just say, "Oh, yeah, well, if there's something on port 238 try sending it this magic packet," and boom. And—so he and Wietse had apparently done this for a long time and then they had written—they had automated the whole thing and released that as open source, and they had become somewhat famous and infamous for that, obviously. And his idea was to use his skills to increase the defenses of various enterprises, and so from the inside you could also check for vulnerabilities. Oh, is the mode of the password file maybe world writeable, or is there a user with no password in there, or things like that. And so, again, the software was intended to have a large and constantly updated database of various vulnerabilities but in a much wider sense than your typical virus checker. It wasn't just looking for files with bad contents but whole system configuration. It turned out that even though Dan and his cofounder, who was more of a businessperson, which really helped me get over the fence, but even they couldn't figure out how to sell enterprise software, which turns out is just really hard. You have to have sold a lot of enterprise software in order to be able to sell enterprise <laughs> software, and that chicken and egg problem they never got around. So after a few years having fun there, and I designed a custom programming language for them and implemented it and did all sorts of other stuff as well, some mentoring and some rescuing Java code. And when I thought, hmm, yeah, there was a change in management and I really could not get along with the new VP of engineering, and it turned out that several of the other lead engineers also couldn't, so people were leaving and I thought, okay, well, I'll try something else. And I asked some of my friends at Google including actually Jeremy Hilton if I could apply there.

Hsu: So then you started—

van Rossum: Yeah, that's a lot of very personal history. I don't know how much it bears on Python itself.

[13] [Interviewee's footnote] Wietse Venema.

Hsu: Right. Oh, but we're interested in your personal history as well.

van Rossum: Sure.

Hsu: So you started at Google in 2005 and you were there until 2012, so you could talk about—

van Rossum: Seven years.

Hsu: Yeah, seven years and your work on various things there.

van Rossum: Well, I always try to do everything with Python and—or in support of Python developers at Google.

Hsu: So what were you hired to do?

van Rossum: I was hired—at least my first team was actually build tools, which is very internally focused team that at the time owned a variety of developer tools, not just build tools. Actually, the thing that got open sourced maybe two years ago under the name Bazel by Google had its origins in those days, although I had nothing to do with it but the people who were creating Bazel under the name Blaze at the time were in—we all reported to the same manager. My project, yeah, it wasn't clear what I would bring to the table for build tools, although it was clear that I could do something because it was a place where there were a lot of Python applications. I remember there was a huge wrapper script for Perforce that took care of all sorts of crazy extra stuff that was a huge Python script. There were Python libraries, and as a starter project, I thought I would do something about the problem of code review, and I wrote one of the very early web-based code review applications. So there's no intelligence in there. Sometimes people seem to think that code review is about finding bugs. That's a different set of tools. Code review is where another engineer can basically check your code and comment on it and tell you, "Well, okay, you've got to change this, this, and this, and then you can check it in." And Google has this and always had, I think, this great system where every piece of code has to be reviewed by at least one other person before it can be committed to source control, and so there were elaborate rules for how to do the reviews, but the tooling for reviews was mostly email, and so you had to just handcraft an email or there was a tool that I think you could copy and paste the diff into the email and then start typing your comments in the middle of the diff, and it was all kind of painful. And there were bots that were watching the email, and if the reviewer responded with certain phrases the bot would automatically notify the author of the code, "Okay, it's been approved," and then the author could commit it. This was always the author who had to commit it. The flow was complicated because you had to go to your email client to write up the review, and the author had to read the review in their email client, but to see the comments in the context of the code you had to fire up your editor, and then there was a third tool[14] where you had to fire up—where you could see the diff highlighted with deletions in red and additions in green. I think that was about the extent of it. And so there were all these different applications that were involved in doing a review both for the author of the code and for the reviewer. And I remember a colleague who was also a core Python developer at the

[14] [Interviewee's footnote] Tkdiff.

time, Neal Norwitz, suggested that as a starter project I should start to try to improve that workflow by making a web application for it, and so he sketched maybe some basic ideas like, well, the web application can look at the source code and it can look at your email and it will present the diff in the web and then you write a little bit of JavaScript so you can click on anywhere in the code and start inserting your comments. And I thought, hmm, okay, well, I've written a web application before. Python seems well supported for this kind of stuff, so let me try that. And I don't know if it was probably a few months before it was really ready, but it became my main project for almost two years, and we sort of—I think after about a year every engineer at Google was using this tool which I named Mondrian to do their code reviews, and code review is a pretty big part of software development, so I would—if I visited another Google office, I would just walk through the open office space and I'd recognize <laughs> the color patterns—

Hsu: <laughs>

van Rossum: —of the review tool I had written on every second screen.

Hsu: <laughs>

van Rossum: That was pretty cool. I think in 2011 they replaced it with a new generation, which is basically the same thing, but it had a really good run. And then after it turned—that became too much focused on just keeping the production up, which was never my strong point, and then I joined the Google App Engine team, which was a completely different thing but also very Python focused, so that sort of web hosting of applications written in Python with a database interface that is customized. It's a non-SQL object database, I think it's called. NoSQL I think is the buzzword. Anyway, it's very different from virtual machines because you don't get a whole machine but you get once process where you can write any Python code you like. And I spent about five years there through all sorts of different parts of the project and quite a bit of growth. I mean when I joined the team there had just been an internal launch, and so we went through the first public launch, and then a year later there was another public launch where the second language was supported. So the first language was supported was Python, and then a year later Java was also supported, but I think the majority of App Engine users always preferred the Python approach. I did all sorts of things there. Towards the end I also worked on [an] attempt to port App Engine to a more traditional virtual machine-based environment, which I think is what's—currently if you look up Google App Engine that's usually how it's done. And the version I worked on originally is called Classic App Engine or something.

Hsu: Why was Python so popular, or why is Python so popular at Google?

van Rossum: Oh, at Google specifically I think that Larry and Sergei, when they started coding, they happened to know some Python and started writing their first prototype in Python, and they had some friends help them who were better skilled Python coders, and so there was always a long tradition of using Python at Google. And by the time I joined in 2005, most of Google's applications were written in C++, the search engine and everything around it. But for internal tooling, whenever there was anything that had to be scripted, it was almost always Python was the lead choice of language. I mean, yeah, I'm sure there was Bourne shell scripting to glue a few things together, but that was very minimal and a lot of tooling was always programmed in Python. And every once in a while products would use Python, but not all that many. There was—for, I don't know, maybe four or five years there was a big project where Google would host open source code, basically the same idea as GitHub but using Mercurial, and the web presence of that project was all implemented in Python.[15]

Hsu: And during your time at all these various companies from BeOpen to Zope through Google, you spent half your time continuing to work on Python. Was that pretty much continuously? Like half of your work was on Python throughout this period?

van Rossum: Yeah, that's pretty much the gist of it, and often I think—so where did that start? Well, at BeOpen it was basically full-time Python except during the latter half of the year there was a lot of pressure to turn that into consulting. At Zope it was pretty much 50/50. Again, maybe towards the end of my stay there was a bit more pressure to help out the projects that were working for paying customers. If the developers there were stuck or if they urgently needed a certain piece of functionality, the PythonLabs team would help and I would dedicate a lot of my time to that. At Google I felt that I could just negotiate for what I wanted, so there I just said, "Well, I would like to have the freedom to spend 50 percent of my time pure on Python things, whether that's answering email from the community or attending conferences or reviewing PEPs or coding up new features," and they agreed to that. I think Elemental Security is the one place where I didn't have a deal like that. And so it was a startup, funds were somewhat scarce. Well, I think at the beginning it was fairly well funded. I think at Elemental I may have had like one day a week or like 20 percent of my time, so I was eager to bump that back up to 50 percent because that really—if all I'm doing is using Python, yeah, it's obviously my favorite programming language but I also really crave that sort of interaction with the community and the feeling that I'm responsive to the needs of the community.

Hsu: Right, because you continued to be the head of the community.

van Rossum: Yeah.

Hsu: At what point did the Python Software Foundation get started?

van Rossum: The Python Software Foundation started in I think 2000 or 2001. There had actually been some failed attempt at having some kind of user organization. While we were still at CNRI we actually created something called the Python Software Activity, the PSA, and this was like in response to some people in the community who said that it would be useful or fun or whatever if people could claim they were a member of Python, whatever that means, and maybe that was something that they thought would look good on their resume or maybe it would help them explain to their manager why this was important. The PSA was never very successful. I think we had a few hundred members and I think everyone paid $50. I don't know if it was annually or once, but when you paid you just got your name on a mailing list, so there was a separate PSA mailing list, I believe, but we didn't really do anything. There was no board or bylaws or meetings or minutes or anything, so that was not very successful. I don't remember if it was in

[15] [Interviewee's footnote] By a team led by Greg Stein.

the BeOpen time and definitely in the Zope time we had Python Software Foundation. Yes, it must've been in the Zope time, so it must've been 2001 [that] the Python Software Foundation started, was created in 2001 because I remember sitting down with Zope's lawyer to nail down the details of the bylaws and the guy explained that it had to be registered in Delaware, for some reason. That was, like, the best corporate law.

van Rossum: So that's what we did. There was also then a Python conference. We had an official meeting. There was a group of about 20 founding members, so I was like the, I don't know, chairman or something and several other people—well, I think everyone in the PythonLabs group was present and I think Eric Raymond was there and Greg Stein was there. There are probably minutes somewhere in the Python archives. Oh, yeah, Dick Hart was there, and now I remember. Dick Hart was the name of the ActiveState founder. And so somehow all of us were like software developers, not very well versed in business and how you run a foundation. And every time some kind of action item came up Dick Hart said, "Oh, I'll do that. Yeah, I've got my corporate lawyers. I've got experience. I've been running businesses for years and [a] nonprofit is not all that different." So all the action items ended up on Dick Hart's plate, and then for a whole year he didn't do anything.

<laughter>

van Rossum: And then the next year we had another meeting of the Python Software Foundation board and nothing had happened, and we essentially started over.

<laughter>

Hsu: So what was the motivation for starting the foundation at that point in time? What was the reason?

van Rossum: Yeah, my original motivation was to secure the ownership of the source code, because after CWI and CNRI and BeOpen I was very worried that at some point I wouldn't make a narrow escape or Python wouldn't make a narrow escape, and some part of Python would suddenly be no longer open source and some company would claim ownership and say, "Everybody who's using Python 2.2 or later owes us licensing fees." And there were, like, horror stories in the industry about that, still are. I mean the Oracle/Google lawsuit is essentially about something like that. So there were some examples. Greg Stein was a member of the Apache Software Foundation, and so he said, "Well, you should do it the way we did it in the Apache Software Foundation. We have pretty simple bylaws." And we basically just copied that. And over the years some small changes have been made and the membership structure has changed a few times, but we were essentially pretty happy with that. But the goal really initially was just [to] have an officially registered organization, a foundation that has a legal presence and a claim to the source code copyright and licensing rights. And then the board will make sure that the PSF always ensures that Python stays open source and that everybody can use it without paying any licensing fees. And it was like, well, sometimes a trademark was an issue. We wanted to make sure that if people started creating software that was also called Python and claiming that they own Python, the name, that we could say, "well, the Python Software Foundation already owns that." And so the only reason that we were—we were collecting money from members and the idea was that we had corporate members who would pay a few thousand dollars annually for the right to be members, and we would just save that money as a rainy day fund in case some kind of legal assault would happen. And then we got the idea of using those funds as a starting fund for a conference because we had like workshops that were just essentially run by whichever organization hosted it like NIST or USGS [US Geological Survey] or LLNL [Lawrence Livermore National Laboratory], and then for a few years a conference bureau[16] that was a subsidiary of CNRI ran the conferences, but they were never able to make it a commercial success and people were complaining that despite the lack of commercial success the entrance fees were too high, and so at a convenient time—I forget it was 2002 or 2003[17], thereabouts—we started organizing our own conference. And, again, I had way too much involvement with this personally. Through some connections we found a venue. It was a really nice venue. I think it was George Washington University in downtown D.C. and was, like, three really nice conference rooms. And I learned a lot because it turns out that almost all the money you pay the venue goes to catering.

Hsu: <laughs>

van Rossum: And so the fact that the PSF was an existing organization made it possible to make the necessary deposits and we didn't just have to tell those people, "Oh, trust us. Lots of people like us. We'll pay you back once they've paid their entrance fee." We could—the PSF was taking the financial risk for the conference, and somehow we were smart enough that it was actually successful. And after all the bills were paid we had a little bit more money in the bank account than before and we still had, like, a number of corporate members and other occasional random donations. And so we did that again, and by now the PyCon is this gigantic event. Well, gigantic, but it's like 3,000 people show up and budget is in the millions, and it's still financially closely tied to the PSF, but except for one year in I think 2008 it was not financially successful, but we also didn't go out of business[18]. And ever before but also ever since PyCon has made a modest profit for the PSF, and that has always been used to, well, seed funding for next year's conference but also just handouts for the Python community. If someone wants to start a local Python event, then the PSF can give them 100 bucks for pizza or something or to pay for a local small venue. And so the PSF, now one of its very important roles is to foster the community, and there's also sometimes stipends for people who want to attend PyCon but can't afford it also are paid by PSF funding.

Hsu: And moving onto Dropbox, you've been at Dropbox since 2013. Can you talk about—

van Rossum: Yeah, five years already.

Hsu: Yeah. Can you talk about what motivated first that move and also what you do there?

[16] [Interviewee's footnote] It was named Foretec, run by a CNRI VP and friend of Bob Kahn named Al Vezza, who was also a founder of Infocom. Foretec also ran a number of WWW conferences, but eventually folded.
[17] [Interviewee's footnote] The first conference named PyCon was in 2003 (https://en.wikipedia.org/wiki/Python_Conference) and it was definitely at GWU.
[18] [Interviewee's footnote] To be clear it was financially successful every year *except* in 2008 (due to the financial crisis).

van Rossum: Well, so my move in part was motivated by a very personal appeal of Drew Houston, who just came to me and said, “Look, we’ve got this great little startup. We are super fans of Python here. All our code is written in Python.” Drew and Arash, the two founders, wrote the first client and server of Dropbox in Python and all that code is still around and still mostly written in Python. And I think I gave a talk at Dropbox, which was, like, partially for Dropboxers—well, there were, like, 50 employees there. This was like in 2003 or 2004. No, sorry, I mean 2010 or 2011[19] or so. I gave a talk there and Drew and I got to talk there again. I remember him visiting me at Google and having lunch and I had a very interesting chat with him about his startup and how they were using Python and I blogged about it[20]. And so at some point, I think in late 2012, he approached me and said, “Well, you should really consider coming to work for us. We’ve got lots of cool stuff. We’re big Python fans.” And he was lucky. I mean I honestly got requests like that before and was never very interested, but the mood in the Google App Engine project had changed quite a bit from the days of the first launches. It was a much larger organization. Management was completely different. The project founders had all left Google, and so I was—without actively looking I was also ready for a new challenge, and I thought, well, it’s actually pretty cool to see what life is like in a startup with 200 people. Well, by the time I started it was 250. But it’s always been a lot of fun and definitely very different things and quite a bit of variety of what I’ve been doing there. Yeah, I started out with also a starter project that someone suggested, which was a sync API for structured data. This turned out to be not a great hit with developers, so that API is no longer around, but it was a lot of fun to create that and work with a whole team of people on getting it ready for production and launching it and presenting it at the launch activity. We probably worked a little too hard on that particular project <laughs> because after the launch event everyone was suffering from burnout and all <laughs> the developers went on really long vacations—

<laughter>

van Rossum: —while the customers were wondering why the software was buggy and why no one was fixing it.

<laughter>

van Rossum: But we recovered from that. My current project which I’ve been working on pretty much full time for the past two years, I think, at least, is mypy, which is static type analyzer for Python.

Hsu: Oh. Yes, I think you mentioned something similar last time we were talking about ABC.

van Rossum: There was probably some overlap.

Hsu: Yeah. So it’s sort of similar to—oh, wait, this is a static analyzer?

van Rossum: It’s a static analyzer.

[19] [Interviewee’s footnote] Definitely 2011.
[20] [Interviewee’s footnote] http://neopythonic.blogspot.com/2011/06/depth-and-breadth-of-python.html

Hsu: Okay.

van Rossum: Yeah, so we have an optional syntactic extension to Python where you can add type annotations to critical parts of the language, mostly function signatures, and the regular Python interpreter ignores that. But a separate program called mypy can check those type annotations, and it basically analyzes your whole program and checks that whenever you call something the arguments that you provide in the call side correspond to the types of the arguments expected by the function definition.

Hsu: And does this bear any relation to the kinds of type checking that ABC had that we talked about last time?

van Rossum: Oh, that is a good question. Let's see. There is a certain similarity in that when you don't specify types explicitly in Python, the mypy type checker will infer types based on the values that go into an expression. So the simplest case is if you say X=1, then the type of X is an integer. And now if you say Y=X+1, then Y is also an integer. And there are all sorts of transformations. If you call a function that takes an integer and returns a string it'll know that—it'll infer that it's a string, and you don't have to declare anything explicitly. There is a difference in that in ABC there were no type annotations at all. Everything was derived from context and from literals. And in using mypy you don't get very far without putting some annotations in. Basically, every function needs to have an annotation. It's just the variables that don't need to.

Hsu: Let's talk a little bit about your personal life. What are your interests and activities outside of Python and outside of work, computing?

van Rossum: Well, mostly hanging out with my family and watching TV together. I like to read books. I like to go out for bike rides—

Hsu: What kind of books?

van Rossum: Well, I like to read popular science, also science-fiction, also other novels. Yeah, that varies a lot, to be honest.

Hsu: Can you talk a little bit about your family?

van Rossum: I've got a wife and a son. I got married in 2000, pretty late in life. We got a kid in 2001. He's sixteen now. He's a teenager. He's a good kid.

Hsu: Is your wife American?

van Rossum: She's American, yeah. She's actually from Texas, but we met in the Washington, D.C. area.

Hsu: And how's your work affected your family life?

van Rossum: It affects it a lot. Yeah, I really have to force myself to spend time with the family rather than being on the computer always working. Okay, like, there's always more email from python-dev or from Dropbox, honestly. So, yeah, I'm the main provider. My wife has a PhD in economics. She used to teach, then she started a self-defense school in Baltimore. Now, she's teaching—the self-defense stuff is pretty hard on your body, so, she can't do that anymore, but she's a personal trainer at Peninsula Jewish Community Center in Foster City. But, so, that I'm the breadwinner of the family and, yeah, that has affected everything. I mean, we moved to California because I got a lucrative job here.

Hsu: Mm-hm. And are you able to afford to live here on a single salary?

van Rossum: Yeah, it's—well, because of my reputation, I get paid pretty well.

Hsu: What was it like moving to the U.S. from the Netherlands and was it a significant adjustment for you?

van Rossum: Well, I guess it was. I mean, I lived in Amsterdam in a beautiful apartment on one of the canals. And I moved to a suburb, Reston, in Northern Virginia and suddenly I had to drive a car to get everywhere. That kind of stuff. And my social life was very different. I remember that there was a little bit of an adjustment, that you couldn't really go out drinking with your buddies and get all that drunk, because everybody had to drive home afterwards. But, all in all, it was a fairly easy transition. I was somewhat familiar with life in America. So, I had been here on vacation in '86. I had been to numerous conferences in various parts of the country. I had seen the Grand Canyon, Yellowstone, Yosemite. Oh, and I'd lived in Maryland for two months during that stay at NIST; that also gave me some experience. Oh, yeah, and in '88 or '89 I had spent a summer in Palo Alto working for DEC SRC.

Hsu: Mm.

van Rossum: So, I had actually lived in different parts of the U.S. briefly, at least. So, it was not entirely unfamiliar with all the many differences and the importance of your social security number and the crazy financial system with credit cards and checks. None of that was all that challenging.

Hsu: How would you compare the Netherlands, the Washington D.C. area, and Silicon Valley as places to live and to work?

van Rossum: Well, yeah, the Netherlands is actually very densely populated. Amsterdam is incredibly busy, always was. Where I lived in the D.C. area was much more suburban. I lived in a townhouse, but [it] was much less dense and you couldn't walk to the corner store to buy your groceries. The weather was very different, much hotter in summer. Winters in Northern Virginia sometimes there was, like, two feet of snow. Oh, yeah, in Amsterdam, if there is like two inches of snow, everybody's excited.

<break>

Hsu: So, yeah, we were talking about comparing the Netherlands, D.C., and Silicon Valley.

van Rossum: I do remember that the job offer in California was—felt like a great thing. I really was looking forward to living there and I'm still very happy living here. And I think that might have had something to do with the mild weather, but also just with the cultural and social climate in the Bay Area, especially, compared to D.C., which is like—well, D.C. has its moments, too, definitely, but the—it is very much a federal town and it can be a little dead on weekends.

Hsu: What about the cultural climate here, particularly attracts you?

van Rossum: Well, actually I really enjoy the outdoors things that I can do nearby. Like, recently, I've picked up birding. My wife gave me a nice pair of binoculars for my birthday and, like, wherever we go we have, like, great shorebirds, raptors, and I really enjoy those things, bike rides. The weather is good pretty much the year round. Well, I got sort of—the funny thing is the longer I live here, the narrower my comfort zone seems to get. So, now I'm really glad I don't live in D.C. anymore with the hot summers and frozen winters. Although, actually, it wasn't so bad. And the Netherlands is actually relatively mild. It's just never very warm. And I think of the Netherlands as a lot of outdoors activities, too, because—in part, because that's just what my family when I grew up enjoyed doing; and there is so much, like, very flat land where you can see forever and ever and ever. And I just, because I grew up there, and spent so much time there, that's still one of my favorite outdoors landscapes, but I remember really enjoying the mountains of West Virginia, all the different things we have here. My wife grew up in a desert town, El Paso. I've enjoyed that, too. She is sometimes bored when I say, "oh, I'd love to go to the desert," because she grew up there! <laughs> But for me it's very special, all the different types of cacti and stuff.

Hsu: So, I have several more summarizing type of questions. So, who are the—or what organizations—or what are the major users of Python today?

van Rossum: I have to think about that, specifically, the users, because there are so many, but I don't keep a list. I mean I used to be able to rattle off, "Well, okay, NASA is using Python, Boeing is using Python, LLNL is using Python." Now, I'm sure that any name in technology or even in finance that you can mention has a large code base in Python. I mean, I spoke to some investment firm a few years ago and the amount of Python code they have is unbelievable. It's probably all crap—

<laughter>

van Rossum: Like, thousands and thousands of programmers writing scripts that are not really maintainable, but it's a lot of code. So, I think the areas where Python is used most is, on the one hand, web application development. Dropbox is a big one there. Tooling, like, the internal tooling we used at Google—there's a lot of that at Dropbox, too, and again, Facebook. Instagram, the Instagram service is written in Python. Then there is the numerical Python and scientific Python world, which I don't actually know enough about, but I know it's a huge growth area for Python. And sort of Python's recent jumping up on all the lists of popular programming languages are largely due to data scientists and, like, Jupyter

Notebooks, those kind of things. So, that's very different type of development than what traditionally was one of Python's strengths. Because these people are not developing websites or accessing databases, they're deriving very sophisticated data processing or machine learning libraries, like Google TensorFlow.

<break>

Hsu: So, you were talking about TensorFlow?

van Rossum: Yeah, TensorFlow is this large machine learning package written by Google.

Hsu: I think they happen to be here today actually.

van Rossum: Oh, yeah, that's right, there was a TensorFlow developers conference. Now, I'm sure that all the essential bits of TensorFlow are implemented in C++ or machine language. On the other hand, all the users of TensorFlow start by writing Python code that imports their data and presents it to TensorFlow. I mean, if you look at tutorials for TensorFlow everything just says, "Start with 'import tensorflow.'" It's not even, like, "If you're using Python, do it this way. If you're using JavaScript, do it that way." It's just like, "Here it is."

Hsu: What makes Python particularly useful in machine learning applications?

van Rossum: It's probably the fact that Python is easily extensible with code written in other languages. So, Python is a great glue language. Yes, you can write applications that are millions of lines of Python, but a lot of people write very small Python applications that farm out all the heavy lifting to libraries written in other languages. And Python has a very rich environment that makes it possible to link Python with other languages. You can write a Python extension module very easily. There's a well-developed, well-defined API, well documented. There are also tools that write the extensions for you, something called Cython. There are other tools in that area, plus a lot of existing tools for dealing with data and reading data files, like the Pandas project is a good example. And, so, all these things—the more tools there are that make Python useful in this field, the more tools other people will develop to build on that. So, it's not always necessarily key features of the language, although I think the extensibility is a very important one. But the fact that Python has a convenient for loop or allows class definitions is probably not all that relevant for the choice of language, to be honest.

Hsu: And Python about ten years ago broke into mobile and embedded computing and has since also become popular in Internet of Things. Could you maybe talk about that?

van Rossum: Mobile is actually still not a great place. There are some projects that do use Python on mobile devices, but the owners of the major mobile platforms aren't all that interested in supporting a wide variety of languages. They prefer to push the one language that they support for that platform and iterate on their environments for that language and the features very quickly, because it's a crazy field, to be honest. So, Python on mobile devices is pretty marginal. On the other hand, on the Internet of Things, one thing that I think is making a difference is something called MicroPython, which is a really small, lean

Python implementation that was written from scratch by someone who had a very different goal than what I had in mind twenty years ago, but who managed to follow Python's syntax exactly. So, if it's Python syntax, MicroPython will recognize it and work with it. Now, MicroPython's standard library is much, much smaller and I think it's also in a sense more varied in that MicroPython can be customized for different hardware. It runs on really small pieces of hardware and then it also has a really small minimal library; and often if there is hardware that has one particular special feature, like it can drive an array of LEDs, then there is a special library that is just dedicated to that platform and there's no pretense of portability there. It's just the core language, not the library. But I think it's a great piece of work.

Hsu: So, why has Python been so successful at so many different things outside of your initial vision for it?

van Rossum: Well, probably the extensibility comes back in there, again, because it means that—I mean, that—as a programming language, it's very likeable. It's concise. You can show someone, "Here are the basics of Python." And if they know a bit of programming in another language, they'll see "Oh, wow, this is very simple. This is very easy. I can understand this already." So, it's easy to get into, but then in every field of endeavor it turns out that you can adapt Python to do things—say, the Internet of Things or a particular piece of hardware or tooling or developing websites or driving numerical libraries or driving a user interface; all those things are possible because people wrote extensions. And, so, Python's extensibility, the mechanism where are you can write some C code, but then it presents to Python programmers as just another module in the world of Python modules and that's like a third-party library—that's has somehow inspired, like, by now, generations of programmers to create new applications using Python and—it's very easy to turn an application into a library for Python. And, so, people tend to structure their applications as a little bit of top-level code that drives the application and a lot of library code that can be connected to differently by different applications once you're in roughly the same field, like, visualization. Lots of different people have different visualization needs. A visualization library is much more powerful than a visualization application. The other thing that probably helped a lot is open source, which—so, I'm really glad that—we were always successful at the debate over the Python license at the time when I was leaving CNRI and the PSF to keep the ownership of the source code clear and free. All those things have helped to encourage people to do things with Python. And that community has made it what it is.

Hsu: And where do you think Python will go in the future?

van Rossum: Well, I, obviously, hope that it will have a long and successful career in front of us. I actually feel almost comically inadequate in predicting the future. I never know what's going to happen next. So, I don't know if the future is the Internet of Things or machine learning or web 3.0 or whatever it is, I think Python will continue to play a big role, because of its flexibility and I expect that Python will also keep evolving to meet the needs of its users and not just Python the language, but Python the community and the ecosystem will change and grow and improve and learn and maybe thirty years from now we won't recognize the community. We will probably still recognize the language, because the language as it was twenty-five years ago is pretty darn similar to what we have now.

Hsu: We want to get another take on the story of the naming of Python. So, would you just tell that story again on how you named the language?

van Rossum: Well, let's see what I remember. So, when I started implementing a new language I don't think I already had a name in mind. I do remember that I had been very frustrated with the process through which ABC obtained its name, because it was like naming by committee. There was endless submissions of all the team members, what name they thought would be good. And then there was some kind of filtering process and in the end none of the cool-sounding names were deemed adequate or appropriate or acceptable and something very bland came out. And, so, I was probably unhappy with the blandness of ABC as a name. I also remember that there—I had recognized certain trends in naming software systems, where at some point, I think, my internship at DEC SRC they were big on the Greek mythology and the names from antiquity; somehow in the '70s and '80s everybody seemed to name their big system Olympus or Zeus or Hercules or some pun on any of those things. There was too much of that and then there were like some languages had acronyms or somewhat pronounceable acronyms like Fortran and Algol. And then there were the languages named after important historical figures from science and engineering, like Pascal or Eiffel or Ada. And, on the one hand, I thought that naming it after someone else was kind of cool, but I was like—I was not so happy with picking a famous name from science like Euler or—I also wasn't into "Lord of the Rings" that much, otherwise I might have called it Frodo. But I was into somewhat subversive TV shows and, so, Monty Python was one of those that I enjoyed watching. And then the name Python has, like—it is not complete—I mean, I would never name a language "Monty Python." I would probably also feel like a copyright infringement or trademark infringement. If you name it "Python", it could arguably be named after a snake. I named it after Monty Python, but—I took it from Monty Python and it was like easy to type, not too many letters, not too many consonants next to each other, easy to pronounce, and it—I don't know, maybe there was also in some subtle sense where it was similar to Perl. At least started with a "P". I don't know. Perl is a brilliant name, because it's a nonsense word and it's so pronounceable. But Python felt like a pretty good choice and I didn't want to think about it too much. I wasn't expecting major success and, so, I wasn't really carefully weighing the downsides and upsides of this name versus that. I just went with a gut feeling and—oh, yeah, I should also mention that in the Amoeba project we had named some other tool, which I forget, after some other TV show, which I also forget. But I know that there was definitely at least one other instance, maybe several, where we used current popular culture as our naming inspiration rather than a history of established science or other pompous sources of names.

Hsu: I want to go back to the question I asked earlier today. This will, I guess, be my second-to-the-last question. So, could you talk about Python support for functional programming?

van Rossum: Oh, I'm sorry. Yeah, I got into the threading so much that I forgot about the functional programming. That's actually—that's kind of a weird story. Python doesn't have much support for functional programming and if you look at what the strengths are of functional programming languages and then we're looking at Haskell as one prime example—there are a few others like F#. Those languages typically are really strong on the compiler technology. The idea is that if you write in a purely functional style and if your language enforces that you write in a purely functional style, your compiler has a lot of freedom to generate optimal code; and when you want to parallelize your code in the compiler, that’s easy because the compiler knows what the mathematical properties of your program are. In Python, on the other hand, we have adopted a few functional idioms like a map function and a filter function to go with lambdas and we have famously something called “comprehensions.” But underneath, all those things in Python are very sequential in nature and they do not allow the compiler to rearrange the order of evaluation for different array elements. And that’s—in my view that actually makes Python fail the test for functional programming support, which is totally fine with me. It’s not meant to be a functional language. It’s meant to be a procedural object-oriented language. And I—there are certain data types that are immutable and the implementation can use that to optimize certain cases, but there are other data structures where I’m much happier with the simple mutable data types that Python has compared to the potential immutable data types that a functional language might have. I’m also pretty sure that a functional language would have a much harder time with an extension module. The kind of extensions that are very easy to write in Python are much harder to write for functional languages. And I’m not saying that it’s impossible, but you have to work much harder at it while in Python it's relatively easy to write extensions. And the data model that an extension sees is pretty straightforward. You know what a list means, it’s just an array of object pointers. And that helps.

Hsu: So, yeah, is there anything that you would like to add? Sort of talk about anything in general that we missed.

van Rossum: Wow, it’s—we’ve spent at least five hours talking.

Hsu: Yep.

van Rossum: I don’t know that we missed much. I cannot emphasize enough that the community did it. It’s really—I mean, it is such a great place. There are so many people contributing and, yes, we also argue, but the—I learn so much from just listening to other people explain why a certain thing cannot work or why we should do a certain other thing and I enjoy the debate. And I’m very happy that the community takes care of all these things that I couldn’t possibly do all by myself, like organizing conferences with three thousand people or selling T-shirts or even solving the problem of software distribution for Python. There are so many things that are outside my own scope. I can’t possibly solve everything, but I can make the language good enough that other people can solve whatever it is that they need to solve.

Hsu: And, so, do you see your role as being—like, providing a focal point or a figurehead for the community? Like, is that a useful thing for the community to have, is to have sort of this leader?

van Rossum: If I read my fan mail, I think that for many people it’s still very important to have a person they can point to. For the people that are doing the actual work, I think my role is mostly that of a mentor and—I see myself spending a less and less active role in the development of code and even the development of language features and more helping people figure out how to think about solving software problems and sometimes how to think about solving people problems or community problems or other real-world problems.[21]

Hsu: All right. Well, thank you very much.

END OF THE INTERVIEW

[21] [Interviewee's footnote] Looking back, this was a pretty good insight. I have followed through by resigning as BDFL…