# The Exercist

Jonni Bidwell meets up with exercism.io creator **Katrina Owen** to discuss Linux, languages and the perennial perils people face when learning to code...


**Katrina Owen accidentally became a programmer while studying molecular biology. Since then she's helped countless others become programmers through her website exercism.io. She's also an open source advocate at GitHub and has co-written (with Sandi Metz) *99 bottles of OOP*, a book that's dedicated to writing efficient and beautiful object-oriented code. We caught up with her at OSCON Europe back in October 2016 to discuss painful memories of Gentoo, friendly code-reviewing robots and what people really want in a coding tutorial...**

**Linux Format: Tell me about your history. Your bio says you have a background in molecular biology.**

**Katrina Owen:** Well, I was actually a secretary until I was about 25. It was around then I realised that there was no real future in secretarial work: it wasn't going to get any better and there wasn't a whole lot of respect to be found. So I decided to get a degree, but I didn't really know what to get a degree in. I just needed it to be something real, something scientific.

At first I didn't have enough of a maths, physics or chemistry background to get accepted to a university. So I spent about 18 months teaching myself those things, sat some exams and then was suitably qualified. But I still didn't really know what to do. The only criteria for me was that it would lead to some sort of real job.

I was accepted to two programmes: one was in aerospace engineering and the other was in molecular biology. I didn't really know which to choose, but I thought, "Well, I'm so new to all of this mathematics, so probably biology is the better choice." I thought I'd probably struggle with some of the heavier math in the aerospace program. In retrospect I think that I would've done better at aerospace engineering, because it was practical and you could resort to first principles to figure things out. Whereas with biology it's like, "It happens because of evolution" and there's a lot of theoretical stuff that's much more vague: "that's just the way it is and you just have to remember it" type of deal. So that was kind of frustrating.

**LXF: How did you get into computers and programming?**

**KO:** I did the biology thing for three years, but as I was doing that I got my first computer, put Linux on it, and learned how to do little scripting things with Bash. I got really excited about programming and problem solving, and ended up doing some freelance work, mostly just helping friends debug their websites and stuff. They got excited about how helpful that was, so I ended up getting quite a lot of work through friends and eventually found a real job.

**LXF: Was this in the halcyon days of early JavaScript, where every browser had their own idea of what the language was?**

**KO:** No, this was way back in the DHTML days. But I didn't really deal with it, mercifully. I ended up very quickly getting involved with back-end PHP-type stuff and not really touching the front-end stuff at all. After a few years of doing that I ended up doing Ruby, which was a very different kind of environment. It felt like people got together a lot more: conferences and meet-ups were more common. The Ruby community has a very whimsical feeling. They're very fond of their puns, all of their project names are very creative.

**LXF: I ask this as someone who chose Python more or less arbitrarily when faced with having to learn a 'mainstream' language. What made you choose Ruby over Python, or anything else?**

**KO:** Y'know what? I did consider both. They both have great communities, they both have a philosophy around quality and around wanting to test things and wanting to make things more delightful for the programmer. When I was trying to figure out what to do next, when I was working in PHP all the time and trying to decide where can I go from here and where I'm going to feel happier, I was considering both of them and didn't really have a way to choose. I didn't know much about either, and I just so happened to meet a whole hoard of Ruby programmers, and more or less by accident ended up doing projects with them.

**LXF: You mentioned putting Linux on a laptop. What was your first Linux experience?**

**KO:** I believe I started with Debian and then a couple of years later I switched to Gentoo.

**LXF: I once dabbled with Gentoo. It's sort of this masochistic chapter that people go through. Although at the time, 2004-ish, I guess, the documentation was much better than that of other distros.**

**KO:** I remember having a lot of trouble getting wireless working. Actually now I remember, for the first two years I didn't bother. Then I moved to the US [from Norway] and was at a hotel where there was no Ethernet and there was something I really needed to do online. I think it took me a whole day to figure it out. I still can't remember what I did.

I remember being amazed that I got it working. And then every time I went to upgrade I'd be desperately googling error messages, trying to figure out the magic incantation. But I found the whole process weirdly exhilarating – it really was a lot of fun.

**LXF: So your adventures culminated in the creation of exercism.io. For the benefit of our readers, can you explain what it is?**

**KO:** The core idea of Exercism goes back to Language Hunters, which is all about human languages that are dying out. It's this kind of in-person, game, interactive way of saving languages that are dying. They drew a

## WHO IT'S AIMED AT

**"You can have a high level of fluency and a low level of proficiency."**

distinction between fluency and proficiency, which I'd always thought were deeply connected. But it turns out you can have a high level of fluency and a low level of proficiency, or vice versa. So you can have a less-advanced levels of skill, but at that level still be fluent. And that's what Exercism targets: this low level of fluency while still being able to solve tiny little trivial problems in a programming language.

And to be able to do it with complete ease, without having to think about how you're going to phrase it, or where the braces go, or what the syntax is. Being able to gain this fluency, at the low-syntax level, frees up the cognitive resources that you need to be able to solve harder problems or do more of the real-world things. So lots of different people use Exercism »

» for lots of different reasons. It's this core idea of gaining a high level of fluency with only a low level of proficiency – that's where Exercism finds its sweet spot.

**LXF: What languages are people learning on Exercism? How does the site work?**
**KO:** The most popular I think are Ruby, Python, JavaScript, Haskell and Rust. And lately Elixir and Elm. Then there are a few niche languages where you have a few hardcore enthusiasts. We have a command line client written in Go. That's good because it means we can ship our client without needing any extra runtimes or dependencies. As long as it's in your path somewhere, it'll run. The client fetches exercises, then once the student solves them, with whatever tools they need, the client sends their solutions back to us. The actual conversations about the problems happen on the site, independent of the client.

**LXF: I've looked around the site and must ask you about Rikki, the friendly neighbourhood code review robot...**
**KO:** One of the things that happened, was that I spent the first two years giving enormous amounts of feedback. But it tended to be the same feedback over and over again. So I realised that there were categories of issues for every single exercise, there were a handful of different approaches and fairly typical mistakes that people would make. So I figured that with a little bit of back analysis a lot of this could be automated. That was the first foray, looking at the syntax tree and seeing if they'd triggered any of the red flags that I'd coded for. That's all Rikki the robot's doing.

**LXF: Exercism turned out to be quite a success, I think you mentioned a figure of 180,000 visitors in two days. How did the infrastructure stand up in the early days?**
**KO:** Well, I think most of those users didn't sign up or sign in or do anything too complicated. The homepage was resilient enough to handle that kind of traffic. Now it might totally fall over because there are too many rows in the database and things aren't optimised.

Right now we're running everything on Heroku. Up until last month we were on the free Heroku hobby dyno, but I ran into a Heroku dev at a conference who got me some free platform credits, so I've upgraded to a professional tier. They also gave me some database credits, but in order to upgrade the database a whole backup and reconfigure and reload needs to take place, and I haven't had time or been brave enough to do that yet.

**LXF: In your talk you divided your users into three categories: newbies, polyglots and artisans. What do these three archetypes want out of their coding experiences?**
**KO:** Code newbies are who I originally thought I wrote the site for. They're people who are learning to program for the first time, and usually that's in Java or JavaScript, or Python. They've done a bunch of tutorials or exercises where someone's been holding their hand and they're at a point where they want to solve problems on their own.

The problem they tend to have is how to cross that gap, and we try and teach them how to do that in a way that's not unreachable. We want for them not to take too long to get to the other side, but to do it in a way where they feel they've accomplished something. Then we give them problems that are more complex. We lead them through this process and they might discover, "Oh, I don't really know how loops work." Then they'll have to go off and do a deep dive and figure out how to do that. So it's all about the small wins.

The polyglots, they're experienced programmers, they usually know at least a couple of languages and might use them professionally. They might find themselves suddenly having to learn Clojure at work or something and one of the fastest ways for them to do that is through lots of small exercises.

They tend to get feedback like, "Hmmm, your Clojure looks like Ruby. It shouldn't. Here's how to do it in Clojure-looking Clojure." There are other polyglots who just do exercises for the sake of learning a new language, or for the pure enjoyment that comes with solving problems.

For the artisans, they're concerned with the deep reflection over design trade-offs. So even though you might have just a 20- or 30-line program, they want to know what it would look like if we did it in a functional style, or if we pushed object orientation as far as we can, how it could be done without conditionals. They'll set interesting constraints for themselves, maybe trying to optimise for the most beautiful readable code possible, or maybe at the other extreme: "Can we do this in one line of Python?"

**LXF: Have you received much heart-warming thanks from Exercism users?**
**KO:** I've gotten a lot of wonderful emails. One of them said ,"A year ago I wasn't programming and now, thanks to Exercism, I have my first junior developer job." Another was from a Lua programmer who was doing game development and ended up getting a job with fast.ly doing stuff with Go. The awesome thing about that was when she started working at fast.ly she was the only one that knew how to do test suites. That was because every exercise on Exercism has a test suite, so she was familiar with how all that stuff works. LXF