Learning to Code

Some tips from librarians who have done it

by Andromeda Yelton

Interest among librarians in learning to code is huge and growing. Lately, numerous library conferences have featured programming tutorials or hackathons. Short workshops are wonderful for introducing fundamental concepts and creating positive experiences around code, but participants don't necessarily know what to do next.

For my *Library Technology Report*, I reached out to the LITA-L, Code4Lib, and LibTechWomen discussion lists, as well as my own network, to survey librarians on how they use code in their jobs. I looked for people who are not primarily developers and who could share examples of short scripts. I also asked for recommendations and resources on learning to code and found common themes. Here are their suggestions:

Find a project. This is the most important. Whether for work or fun, having a goal will help you persevere through the inevitable challenges. You will feel a sense of accomplishment when you make progress. You'll get natural answers to questions like "What programming language should I learn?" and "What do I need to learn next?" Whatever you choose, make it as small as possible (or break it down into small parts) so it doesn't get too overwhelming, and feel free to incorporate working code snippets you find online.

Even experienced programmers regularly look up syntax details.

Rely on Google and existing code. Modifying existing code is not cheating. Someone else may have written code to do most of what you want; the ability to read and edit others' code can get you a long way, even if you never write your own programs from scratch. Even experienced programmers regularly look up syntax details and copy code snippets. Googling for something like "[programming language] [problem keyword] example" will often turn up helpful code samples and StackOverflow advice. Browsing library coders' GitHub repositories can yield much useful code and inspiration, too. The Code4Lib wiki page Libraries Sharing Code (bit.ly/1C8M9M2) is a good starting place.

Write documentation. Google, StackOverflow, and (to a lesser extent) GitHub work as learning tools because people have invested time in documentation. Pay it forward! Writing up your learning process helps those who come after you—and yourself in six months. Writing requires organizing your thoughts and is a good self-teaching tool. Additionally, many open source projects want help with documentation as well as code, and this can be an easier route than code to begin contributing. Read the project guidelines, look for a bug tracker with open documentation bugs, and make things better while your memory is fresh. Writing documentation increases the chances that others will build on your work, which will be motivating and rewarding.

Persevere. Learning to code is hard. You must devote a lot of time to it. Also, you'll make mistakes and some of them will be hard to debug. Beginners often think this means they don't have the aptitude, but they're wrong; coders at all levels constantly run into challenging bugs. As web developer Kate Ray writes in TechCrunch (tcrn.ch/1D9SiJo), "There is no mastery, there is no final level. The anxiety of feeling lost and stupid is not something you learn to conquer, but something you learn to live with."

People don't talk enough about emotion in learning to code—about the intense ways learning code can push us into impostor syndrome, can make us feel we don't belong (particularly if we're not a 19-year-old white male in a hoodie), can make us feel frustrated and anxious and overwhelmed.

Find a mentor. Mentors are great for answering technical questions and for telling you about tools and best practices that may not be written in books. But they're also great for holding your hand, cheering you up, and bolstering your self-confidence. ▮

ANDROMEDA YELTON *is an independent consultant. This column is adapted from her Library Technology Report "Coding for Librarians."*