Getting Started with Coding

AALL Law Repositories Caucus Sandbox Series #3, Session 1
John Beatty, Charles M. Sears Law Library, University at Buffalo (jrbeatty@buffalo.edu)

Background:

I started coding long before I became a librarian. Like many children in the 1980s, I was fascinated by personal computers and learned some programming by reading books and playing with the few computers in my school. My formal training consists of two semesters of BASIC programming taken in my sophomore year of high school, a sequence of five two-day courses in Microsoft Access offered by Syracuse University to employees, and a database design and programming course taken in library school.

I learned VisualBasic for Applications on the job at Syracuse University while building a database application to order instructor desk copies of books. For those who are not familiar, desk copies are the free copies publishers give to instructors in return for them assigning the book in their class. The SU bookstore ordered those copies for instructors when they placed their class orders with us. I inherited the desk copy job when I started at the bookstore. At that time, the job consisted of going through large stacks of paper orders and filling out an in-house order form for each individual book by hand, then faxing the orders to the publishers. I proposed automating this by building a database that would store the book information, professors' office addresses, and contact information for the publishers. I spent a semester building the first version, but when it was done, I had turned a 20-plus hour weekly job into one that took about five-hours a week at peak times. Eventually, we put the backend on a shared drive and gave everyone in the department a frontend application so they could quickly look up orders without pulling the paper files.

I built another Access database application at my first library job. I moved the A/V schedule for the law school from a daily Word document and slips of paper to Access. Now, the job could be easily done with an Exchange or Google calendar, but at the time it was a big progression to be able to schedule an A/V setup three weeks in advance and know that someone was actually going to show up.

I have built yet another database at UB. I took the data from an already existing faculty publications database and moved it into a new database that I built out to export publication information to the batch spreadsheets used by Digital Commons. Over time, I extended it out so that changes to the data in the database can be exported to Digital Commons batch revise spreadsheets.

I learned Python for a repository project at UB. While we were importing decades of law review issues into Digital Commons, I realized that we had some types of content that had no metadata and had multiple pieces bunched together into single files. It was mostly book reviews and case notes. Shortly after realizing this, I attended Aaron Kirschenfeld's CALI presentation about how he, with the help of a newly-hired reference librarian and some student workers, had split up his school's case notes into individual files with metadata for import into Digital Commons.

I did not have a newly-hired reference librarian or student workers, so I decided to automate the process. I spent a couple of weeks reading some basic Python how-to books and started programming. Over the course of a semester, I built and refined a few command-line tools that would scan a PDF,

attempt to extract metadata of the individual articles within, split the PDF in the right spots and kick out the metadata in a form that could be imported into Digital Commons. Because of OCR errors and occasional non-standard formatting, the metadata had to be checked and corrected by hand before the PDFs were split. But ultimately, it took less time to learn enough Python to build the tools and use them than it would have to do the whole thing by hand.

My introduction to R, OpenRefine, and data analysis tools was from UB's former CLIR fellow, Rachel Starry. While she was with us, Rachel ran a number of classes on digital humanities, data analysis, and other tools for faculty. She also ran a summer retreat for the librarians where she taught us basics of Open Refine, RStudio, and data visualization. I have used this knowledge more for research and writing projects and for assisting law faculty with analyzing survey data than for repository projects.

I lay all this out not to show how awesomely amazing I am, but to convince you that if you have repetitive, time-consuming tasks, you can do your job more effectively by learning some basic coding skills.

Coding

Why Coding?

One reason is all of the above. Another is that you may have inherited some code that needs to be updated and maintained. Or someone at another institution may have written something that is almost what you need to do a task at your job, but it needs a little customization. If you have some skills, you can do that customization.

In addition, you will get back all the time you spend learning, and then some. Even if it takes you as long to learn the skills for the initial project as it would have to do it by hand, once you have coding skills, you will find more places to use them.

OK, you've convinced me. What language do I learn? There are so many.

Yes, there are. Two, in particular, are widely used and lend themselves to repository projects: R and Python.

Python

Python is a scripting language that can be used for simple data analysis, but is powerful enough to create standalone applications. There are countless libraries that add functionality to the language. I used two PDF libraries to help extract data from PDFs and to extract pages from PDFs to write into new files.

Pros:

- Good regex support
- Can be used to create command-line programs for automating processing tasks
- Lots of learning resources
- Lots of available code for tweaking and using in your workflows
- Very rigid syntax, which helps made code readable

Cons:

- Can be intimidating
- Very rigid syntax, which can make debugging frustrating at first

R

R is a scripting language built for data analysis and reporting. It has basic program flow control. Like Python, it has a lot of libraries that add functionality. A common set of libraries is called the Tidyverse. A useful library for repository projects is Readxl, which provides tools for importing and exporting data from and into Excel spreadsheets.

Pros:

- Robust support for data-munging, including text replacement
- Abundant learning resources
- It's easy to get started

Cons:

- Not built for creating standalone programs
- Regex can be a little weird compared to Perl and Python

Wait? What's this "Regex"?

Regex is an abbreviation for regular expressions, which are a powerful tool for searching text data for patterns. Even if you don't learn to program, learning regular expressions can help you when you have large amounts of data to search and update.

So how do I get started?

Install a development environment

If, like me, your workstation is locked down, you may not be able to install a development environment on your local machine. Installation of software isn't required to learn.

It's probably easier to learn Python without one to start. But once you're working on projects, you may find it easier to work in an application like PyCharm, which will not only offer debugging tools and a ready command line, but will also handle version control for uploading to a code repository like GitHub.

If you're using R, RStudio is a robust and widely-used development environment. RStudio is available as a standalone application or can be used online at Rstudio.cloud. If you work at a school, you can get a free cloud account and use RStudio in a web browser without installing anything on your local machine.

Visit some resources to learn about programming

The Programming Historian (https://programminghistorian.org/)

The Programming Historian offers free tutorials for multiple languages and tools including Python, R, OpenRefine, and more. A lot of the tutorials focus on data analysis, but there are good basics here that will be useful for repository projects, including ones on OCR and machine translation, working with batches of PDFs, cleaning OCR's text with OpenRefine, and basics tasks in Python.

Automate the Boring Stuff with Python (https://automatetheboringstuff.com/)

This is a basic Python book structured around how to do useful tasks, which can be read online for free. This was recommended to me by a friend who's in IT. Full disclosure: I did not work all the way through

this book. I also used three others when I was learning Python. This book is useful for someone who has not programmed before and assumes no previous programming experience. The chapters include exercises, which will help with retaining what you've read.

R for Data Science (https://r4ds.had.co.nz/)
R for Data Science is another book available to read for free online. It is written by Hadley Wickham, one of the creators of the Tidyverse, so it explores R using the Tidyverse libraries. There are conflicting opinions on whether it is better to learn base R without the Tidyverse. If you need visualizations, the Tidyverse functions work very well with ggplot2, a very good library for generating visualizations. That, in itself, can be a reason to learn the Tidyverse.

Regular Expressions 101 (https://regex101.com/) & RegExr (https://regexr.com/)
RegEx 101 and RegExr are both tools to help one learn regular expressions. Both feature boxes to paste in text and run practice expressions on that text. Both feature a command reference and multiple engines. RegEx 101 has more engines, including Python. After reading about regular expressions in chapter 7 of Automate the Boring Stuff with Python, head here to practice.

OK, I bookmarked all these things, what do I do with them?

Find a project! No matter how small, find a task that you can automate. It's easier to get started if you have a particular task in mind. If you start with a concrete task, you will learn to identify helpful commands, functions, and libraries more quickly. Learning is less abstract when you have the chance to immediately apply it.

What kind of project?

It doesn't matter. Anything that will be useful or at least halfway interesting. Once you learn how to do a couple of basic tasks, you will gain confidence and understanding on how to fit those tasks together to complete a big project. Here are some ideas to get you started.

Metadata standardization
Do you have some metadata that was entered into your repository before you had a controlled vocabulary? Did your institution change the form of its name several times? Do you need to make sure a faculty author's middle initial is always included in their name on the repository? Is there something that needs to be changed into a different format? If you're using Digital Commons, it's fairly easy to download a batch revise sheet and perform a series of find and replace operations in Excel, and then upload. Instead, think about using R or Python to read the spreadsheet, identify the metadata using regular expressions and then updating it with string replacement commands.

Build a faculty publication list
Pull the last five years of faculty publications from your repository and reformat it into a list of bibliographic entries in a standard format. The form of that will depend on the repository. From Digital Commons, it's easiest to use the batch revise files. But for other repositories, you may have access to an API, or another export format. Or you may want to use a web scraper. Bonus: If you figure out how to pull the information from Digital Commons batch revise files, you're one step towards knowing how to update Georgia State's DC to BibTex script to add new fields if they will give you a copy.

Perform OCR on a bunch of PDFs

Maybe your institution won't give you a copy of Acrobat Pro because it's too expensive. Use Python and an OCR library to add text to all those image-only PDF scans you have of your faculty or law review's pre-born-digital publications. And post-born-digital publications. Who are we kidding? They didn't save the born-digital versions.