Exploring History on the Web Through the Programming Historian

By Ian Milligan, University of Waterloo

Have you ever sat at your computer, looking at a long list of records held at a place like Library and Archives Canada, the British Library, or the Library of Congress, right-clicking and downloading each record by painstaking record? Or have you ever had a fascinating collection of primary documents that you would love to make a digital exhibit of, but don't know where to start? What about a massive array of text documents, downloaded from an archive or digitized by a graduate student, that is so big that you put it off for, well, ever, due to its sheer size? Well, there are tools for that.

Enter the *Programming Historian 2* at http://programminghistorian.org/ (full disclosure: I am an editor-at-large of the project). In this column, I want to introduce it to you: what it offers to you as a researcher, as a teacher, or as a practicing historian in several other dimensions.

In short, the *Programming Historian 2* has a simple goal: to teach the basics of programming, drawing on historical examples and with an emphasis on bringing you up to speed quickly and practically. It is not a computer science course: it is instead a series of hands-on examples, focused on the what and how rather than the underlying architecture that drives programming languages. It's also a challenge to conventional forms of publishing, being a "community-driven collaborative textbook," soliciting submissions, constantly refining, and inviting comments at all stages.

The *Programming Historian 2* uses all open-source software: the Python programming language, the Komodo Edit editing environment, the Omeka digital exhibit platform, or the MAchine Learning for LanguagE Toolkit (MALLET). Defining open source can be tricky, as there are competing interests, but essentially to speak of "open source software" means adhering to the definition laid out by the "Open Source Definition" (http://opensource.org/osd). In short, open source software requires free redistribution, the source code, allowing derived works (modifications to the program), keeping the integrity of the authors' source code, no discrimination, and general licenses with a few other restrictions. The implications for historians include that it's all free (!), with obvious benefits in an era of diminishing budgets and austerity. Second, that source code can all be continually improved, and third, it means that historians can learn to participate, make suggestions, and give back to the open-source community as they learn to code.

Omeka

Putting history online has never been easier with the rise of Content Management Systems, or CMSes, like Wordpress, Drupal, and Omeka. The latter is specifically built by George Mason University's Centre for History and New Media (CHNM) and is designed to create beautiful exhibits with comprehensive information about each document or collection. If you want to show off items, tell stories, and make sure that each object has enough information that somebody could cite it properly; this is the tool for you.

With the *Programming Historian*, there are a set of lessons in getting up and running with Omeka. It takes you through the basic building blocks of what an item is, how to arrange them in collections, and how to take your visitors through stories that arise out of those items. Crucially, it's all built upon Dublin Core, a standardized set of metadata that makes sure items across different platforms are described similarly: that dates are written in the correct fashion, or that units of measurement are employed properly.

Again, it's all free. On Omeka.net, you can have a fairly limited, small website for free with some fees for more space, or you can download the software and run your own server for Omeka.com. If you work at a university, chances are that your IT department can set up the server for you.

For some great samples of Omeka in action and some installations to inspire you, check out:

- OMEKA Lessons: http://programminghistorian.org/lessons/up-and-running-with-omeka
- Portuguese Canadian History Project: http://archives.library.yorku.ca/exhibits/show/pchp
- Hurricane Digital Memory Bank: http://hurricanearchive.org/

Programming Historian 2 is not a computer science course: it is instead a series of hands-on examples, focused on the what and how rather than the underlying architecture that drives programming languages. It's also a challenge to conventional forms of publishing, being a "community-driven collaborative textbook," soliciting submissions, constantly refining, and inviting comments at all stages.

Python

"Programming? But we're not in Comp Sci?!"

That was basically the reaction of a classroom of University of Waterloo undergrad students in Winter 2013 as I told them what we'd be doing over the next two weeks. As I told them then, I think programming matters for historians, especially those who are sitting in my digital history class. As William Turkel and Alan MacEachern wrote in the first edition of the *Programming Historian*: "If you don't program, your research process will always be at the mercy of those who do." We use Microsoft Word, library searches, draw on Google, digitize things through Adobe Acrobat, record statistics into Microsoft Excel, generate word clouds with Wordle, etc., but we don't always know how it works. Programming is a process of creative construction, and it belongs squarely in the humanistic tradition.

But my course wasn't Computer Science, and neither is the *PH2*. Before long I had my students chugging through sources and exploring algorithms. By the end of the module, only about 1/5th of them were still thinking the same negative things they had going in, and they'd even humoured me as we went through. Programming has a myth of being really hard, and while it certainly requires an attention to detail, the *Programming Historian* helps to demystify it. And there's something to be said about a process that gives you immediate feedback - your program works or crashes! - as opposed to the long cycles of feedback our students are used to.

So what did we do?

- Installed an Integrated Development Environment (IDE) so students could quickly write code and execute it on their computers; starting with the simple **print 'hello world'** to get the traditional 'hello world' greeting.
- Learned to create text files using computer programs, combining ever-increasing complex strings.
- Opened up webpages with a programming language, downloading copies of *Old Bailey Online* transcripts by writing a few lines of code;
- Counted words that appear in court transcripts
- Normalized data so that all information would appear in lower case, with punctuation cleaned up,
- Created beautiful visualizations showing how terms appeared in various contexts within court transcripts, getting a sense of the past without reading the documents themselves.

For more on what you can do with programming, see:
- Python lessons: http://programminghistorian.org/contents
- Invisible Australians (the author started with the PH): http://invisibleaustralians.org/faces/

Other Topics

That's not all! There are lessons on the Zotero API, which shows users how to combine their knowledge of Python with the open-source reference database Zotero. In addition, there are lessons on topic modeling (a quick way to extract meaning from large datasets, as the computer reconstructs 'topics' that it finds) as well as automated downloading.

Conclusion

So whether you're looking for a way to speed up your research with digitized sources, interested in adding a substantial digital component to a methods course, or are just curious in hacking around, the Programming Historian is for you. It's occasionally frustrating (maybe even a bit infuriating), but the payoff is well worth it. And who knows? Maybe you'll get bitten by the digital history bug, and next thing you know, you'll be contributing to our textbook.

Ian Milligan is an assistant professor of history at the University of Waterloo.