Programming for Librarians: Where to begin

A LIBER Digital Scholarship & Data Science Topic Guide for Library Professionals

Péter Király Peter Verhaar

2025-05-14

This guide aims to provide library professionals with some guidance on how and where to begin their journey into learning a programming language.

Introduction

Even just a little bit of programming knowledge can bring huge efficiencies to the work of library professionals. Librarians can use code to solve problems, automate tasks, analyse data, and explore new tools for curating, collecting, and conducting research into our digital collections and data. And though many of these tasks could be done manually, it is often far more efficient to develop or make use of existing computer code to perform such tasks, particularly those which are by their nature highly repetitive. It can be very helpful for librarians, because of this, to become proficient in a programming language. Examples of programming languages include Python, R, Java, Perl or PHP.

What is "programming"?

Programming can be described as the process of using computers to address specific problems. Computer programs consist of a sequence of statements which explain how an activity can be automated. Programming languages generally implement a certain algorithm. An algorithm is an explicit and unambiguous description of the steps that need to be followed to arrive at a well-defined end result. The algorithm constitutes the logic of the program. The statements or commands in a programming language also need to be expressed in a format that computers can understand. They need to be expressed using a syntax, which prescribes how words, numbers, and punctuation marks ought to be used.

There are a number of different languages in which you can write computer code. While it is difficult to find reliable data about this, it feels safe to claim that Python and R are currently the most widely used languages for experimentation and automation in the library sector. When you start learning how to program, it is probably best to focus on one language initially

and to develop skills in one specific programming language. It must be stressed, however, that when you develop your expertise in one concrete language, this will inevitably help you to improve your computational thinking and to broaden your understanding of how programming languages function in general. Once you have learned about the nuts and bolts of one language, it becomes much easier to gain mastery over another language.

Which programming language should you focus on? Python vs R?

Python, firstly, is a free and open source, general purpose language. It is widely used to carry out data science tasks, such as cleaning, enriching, analysing and visualising data, and as a 'glue' language to create interfaces to 'stick' other software together. Python makes use of a sparse, readable and intuitive syntax, and this makes the language relatively easy to learn for beginners. Within the Python community, numerous code libraries have been shared which extend the basic functionalities of the language.

R, is another free and open source programming language that was created originally with a specific focus: to support statistical analyses. Later it was developed towards a more general direction with the help of thousands of software packages created by the R community and today it is used to solve as many types of problems as Python or Java. R's syntax and its most important data structures are quite different from that of Python, and it requires a different way of thinking.

Because of this strong data analysis background we suggest you choose R if you would like to run core data analysis tasks that require a more advanced statistical toolbox, and does not require many additional steps before and after the calculations. Python on the other hand provides somewhat less statistical functionalities, but supports much more general purpose tasks out of the box like file management, string, date and time manipulation, and computer networking. None of these tasks are impossible with the other programming languages, but what they are designed for and what they make easy are different. In data science related tasks they are quite close, e.g. Python's Pandas library and R's data frames (particularly in the Tidyverse package) have similar functionalities. Another - subjective - factor of the decision could be the aesthetics of the graphics Python and R produces: both of them have more than one plotting library, each are easy to recognise by their distinctive graphical style.

Relevance to the Library Sector (Case Studies/Use Cases)

Programming languages can be used for a variety of library tasks.

- Converting one data format into another data format. A computer system may export descriptions in the MARCXML format, while another application may demand JSON input. Such transformations can be carried out using a computer program, for example using the PyMarc package.

- Downloading large numbers of files using computer code. This could include interacting with APIs for web services like OCLC, or library information systems like Alma, Aleph, or Folio. See the *DS Topic Guide* on APIs
- Analysing and visualising data to interpret the data or to see patterns in large datasets. See the *DS Topic Guides* on Working with Data, Collections as Data, and Data Visualisation.
- Creating AI and Machine Learning applications. See the *DS Topic Guide* on AI and ML

The British Library, The National Archives and Birkbeck University in the UK partnered on a trial of a one-year part-time postgraduate Certificate (PGCert), Computing for Cultural Heritage, as part of a £4.8 million University skills drive in 2019-2021. The 20 staff projects devised and undertaken by the students demonstrate the wide range of what programming skills can enable for those working in a library setting.

The code4lib journal publishes many articles with case studies on applications in library contexts.

Hands-on activity/self-guided tutorial(s)

Starting any of these tutorials requires somewhere to write code. This will likely either be in an Integrated Development Environment (IDE), a code editing program that has lots of helpful features for writing and running code, or a tool that runs in-browser. Which you choose is influenced by personal taste and whether institutional IT restrictions allow installing software on your computer. If you can install software on your laptop then the tutorials below will contain all the information needed. If not then we have tried to include options that work in-browser without installing anything.

Python Tutorials

The tutorials below recommend local installs of Python. If this isn't possible you can try Google Colab, a 'notebook' style interface, or this online IDE, for an IDE interface. Notebooks use different blocks of code called cells that can be run individually, and are good for experimentation. IDEs run an entire code file each time, and are better for reproducibility and larger code projects.

- The Bits and Bots study group is an excellent place to begin to get to grips with Python in a playful way. The study group was founded by professionals in digital preservation Francesca Mackenzie (National Archives UK), Lotte Wijsman, and Susanne van den Eijkel (National Archives of the Netherlands) who realised that a basic level of technical skills can significantly enhance their work in archives. You can join their study group when it's running or just follow along with the course materials on their project github at your own pace:
  - Guide to building games with Python Modules 1-8. pdf

  - Guide to Website/Twine Based Games Modules 1-8.pdf

- To develop a basic understanding of the main features of Python, you can also follow the Introductory course on Python that was developed at the University of Leiden. It is a self-paced course which you can follow to familiarise yourself with central concepts such as variables, operators, flow control, data structures and functions. The tutorial also explains some of the code libraries which can be used more specifically for data science tasks such as data acquisition, working with APIs, data analysis and data visualisation. The tutorial includes a large number of exercises with which you can hone your programming skills, together with answers which can help you if you get stuck.
- Once you've gained a basic proficiency of the central Python concepts, there are a number of useful resources to improve your programming skills specifically with library related materials.
  - You can follow the tutorial Python for Librarians, which was developed by the Library Carpentry.
  - Tim Sherratt: GLAM Workbench (2021- is "a collection of Jupyter notebooks to help you explore and use data from GLAM institutions (that's galleries, libraries, archives, and museums). It includes tools, tutorials, examples, hacks, and even some pre-harvested datasets." Sherratt focuses on Australian and New Zealand datasets, but you can reuse the code for other sources. The notebooks are written in Python using Jupyter Notebook, a web-based interactive computing platform that combines live code, equations, narrative text, visualisations, interactive dashboards and other media. You can find many tutorials and GLAM related scripts in this form.
  - Programming Historian is a set of programming tutorials (both in Python and R) focusing on particular computational problems (such as network analysis, data managements, APIs, machine learning) made mainly for historians, but several tutorials use GLAM data.

R Tutorials

To get started with R we recommend the Library Carpentry's tutorial Introduction to R which teaches you how to do basic data science tasks on tabular data, such as spreadsheets or relational database tables. If you can't install R locally then you can use Posit Cloud, which runs the standard interface to R in your browser.

Recommended Reading/Viewing

Python

Python.org/Documentation is a great place to start as it holds loads of useful introductions for getting started, including a Beginner's Guide.

The open access *Python for Data Analysis*, 3rd edition by Wes McKinney.

Paul Vierthaler's Hacking the Humanities Tutorials provides a quite useful video lecture series introducing python to beginners.

R

The R: Documentation webpage contains a huge amount of useful documentation to help you on your way.

We can also highly recommend this recently published book on R, *Hands-On Data Science for Librarians* written by Sarah Lin and Dorris Scot (Boca Raton:CRC Press, 2023, ISBN 978-1-032-10999-2) which hopefully your institution may have a copy of.

Though not written specifically for librarians, but probably the best introduction to the topic of data science with R is *R for Data Science*. 2nd edition by Hadley Wickham, Mine Çetinkaya-Rundel and Garrett Grolemund (O'Reilly, 2023, ISBN 978–1-492-09740-2), and available free online.

Finding Communities of Practice

When you're first getting started in programming, being able to ask others questions when you get stuck is so important! We recommend seeking out local study groups dedicated to learning programming (for example one like the Bits and Bots study group), or maybe even think about starting your own with like-minded colleagues!

Code4lib is a volunteer-driven collective of folks that has been around since 2003 and amongst their many resources are a large range of chat rooms for folks interested in the convergence of technology and cultural heritage. Visit Chat | Code4Lib to find the discussion you need or join their mailing list.

You might also want to have a look on Github, which is one of the largest source code repositories on the web, for the "code4lib" tag to denote repositories that are somewhat relevant for the library community. They are not tutorials, but during your learning process you can check if there is a software solution already for your particular problem, and you can even try to engage with one or more tools as a contributor. Contributions are always welcome on #code4lib projects, and even if you have just begun programming you can provide important feedback

about the usage of a tool, or you can improve code documentation as a first step. See our *DS Topic Guide* on for guidance on how to contribute to projects on GitHub.