if (you are a digital archivist) { you = "might need to learn to code"; }

**Case 1: NYPR**

Proprietary software used to extract MiniDisc audio tracks at New York Public Radio's archive failed unexpectedly. Historic field recordings are stuck on physical media.

Digital Rights Management built into MiniDisc hardware and software prevented The Archive from accessing ATRAC data in a simple, direct manner.

Researching a solution, I discovered a collection of open-source of Python scripts created by a group of developers under the name netmdpython that worked with certain MiniDisc players.

The command line was used to install python and ffmpeg, and to launch scripts and display error logs. Ffmpeg was used to transform *.aea files into *.wav files.

**Takeaways:** I did not need to become a Python expert to run netmdpython scripts. However, it helped immensely that I already had decent command line skills.

**Case 2: CUNY-TV**

Archival and preservation workflows at CUNY TV needed enhancement and refinement. Code was written in bash and executed using the terminal application on a Mac OSx.

Experiential learning made learning code much more meaningful. I was further able to reflect and share what I learned through blog posts and presentations.

Using automation in the scripts made it possible to process materials quickly and efficiently, and at scale. This frees up time for archivists to do other work that can't be automated, such as cataloging.

Digital Preservation processes were facilitated by developing open source scripts for performing fixity checking and transferring files to LTO tape for long-term storage and access.

**Takeaways:** I was able to learn the scripting language bash and implement changes to processing scripts that had a positive impact on the archival and preservation workflows at CUNY TV.

**Case 1: RHIZOME**

Rhizome, a born-digital arts organization, uses a variety of approaches and a mix of open-source web archiving tools for capturing & providing access to internet-based artworks.

Dynamic web content poses challenges for crawler-based capture, while the high page counts of complex websites requires automated methods to make conservation processes manageable.

Pywb, a python-based replay system for archived web content, runs in the command line and allows for building, indexing, and replaying collections of WARC files locally via a handful of simple commands.

Extensive documentation available online makes it very possible to learn these tools on your own! ArchiveTeam, the IIPC WARC specifications, and the pywb docs on github offer excellent starting points, links, & resources for experimenting with web archiving.

**Takeaways:** Using pywb made it possible to combine visually-oriented appraisal and capture of artworks via Webrecorder with traditional recursive web crawler methods such as wget.