

Repurposing Archival Metadata with the Python CSV Writer

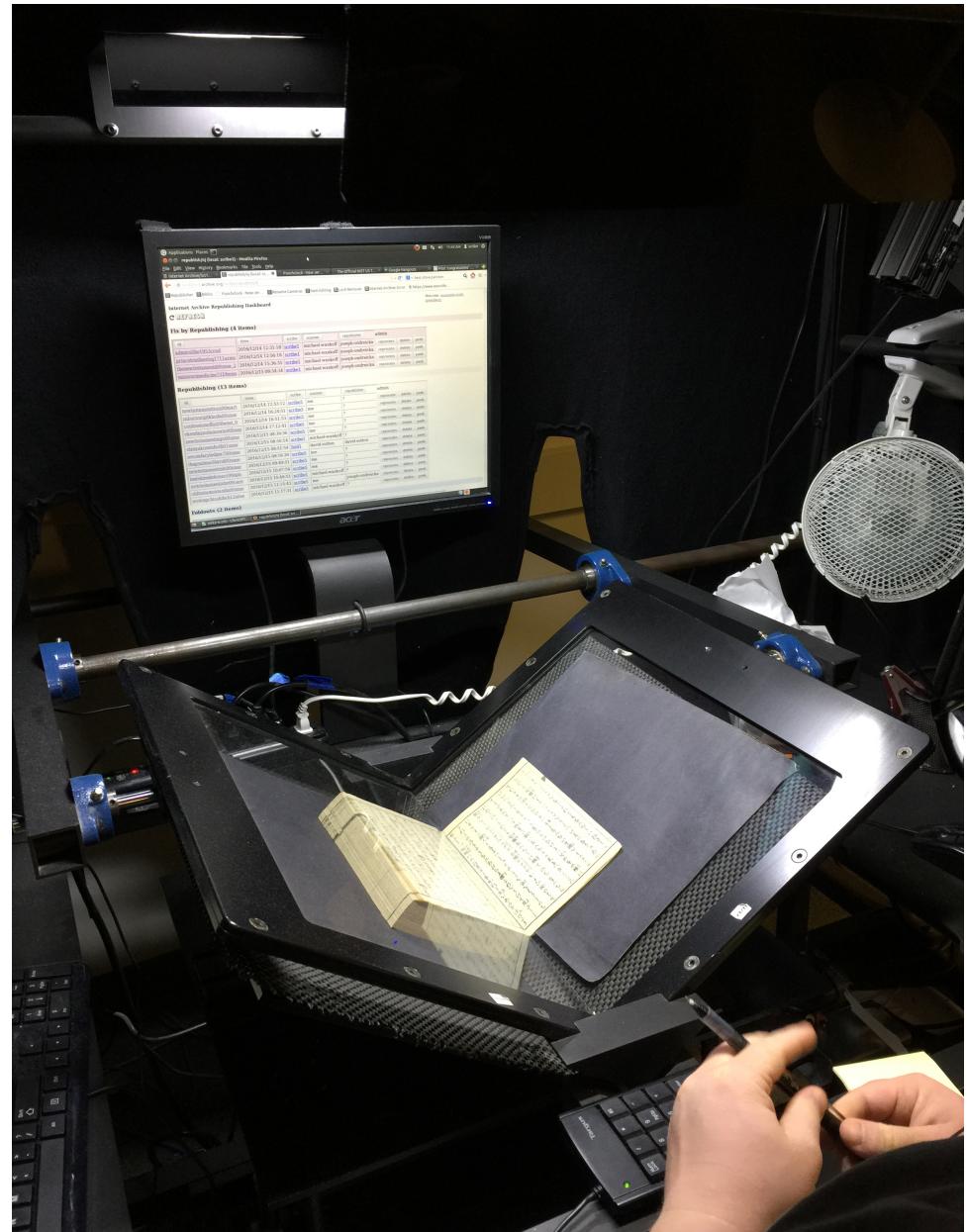
Jackie Rider

Programming for Cultural Heritage 2016

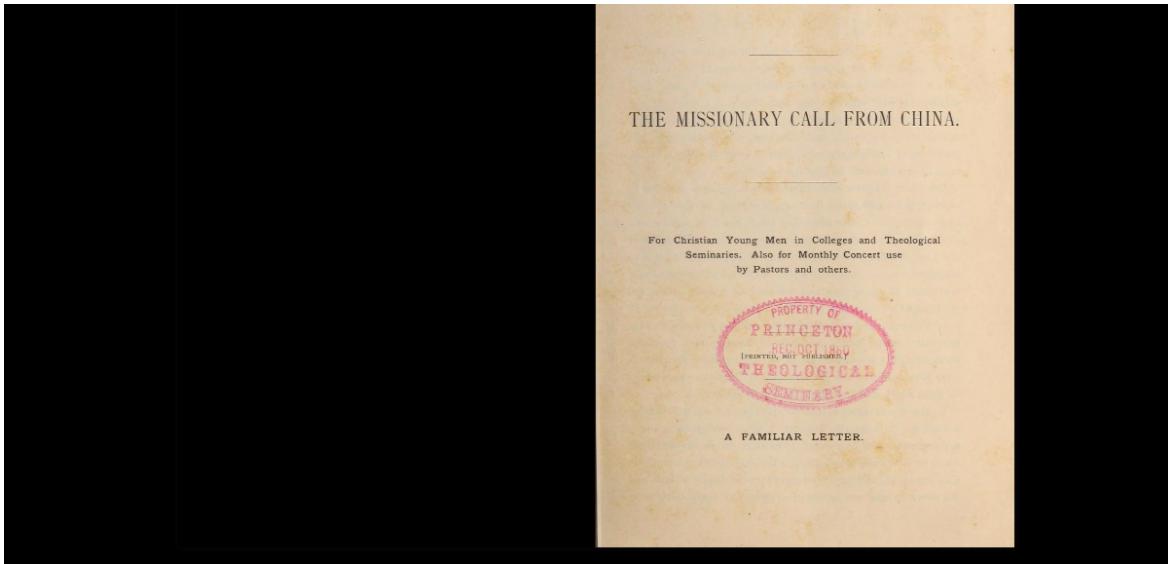
December 15, 2016

For my project, I wrote a Python script that will extract selected XML metadata from an EAD encoded finding aid and export it as a CSV file.

Princeton Theological Seminary Library contracts with the Internet Archive to digitize some of the Seminary's special collections. Part of the digitization process includes creating metadata for each digital object and supplying that metadata in a spreadsheet to technicians at the Internet Archive.



The Internet Archive
then displays the
metadata online with
the scanned digital
image of the collection
item.



The missionary call from China : a familiar letter

by [Chapin, L. D: American Board of Commissioners for Foreign Missions](#)

Published 1880

Topics [Missions](#)

[SHOW MORE](#)

"For Christian young men in colleges and theological seminaries, also for monthly concert use by pastors and others."

"Printed, not published."

Publisher [\[Boston : A.B.C.F.M.\]](#)

Pages 6

Language [English](#)

Call number [SCP #45,100](#)

Digitizing sponsor [Princeton Theological Seminary Library](#)

Book contributor [Princeton Theological Seminary Library](#)

Collection [majorityworldcollection](#); [Princeton](#); [americana](#)

[Full catalog record MARCXML](#)

This book has an [editable web page](#) on Open Library.

DOWNLOAD OPTIONS

[ABYY GZ](#) 1 file

[DAISY](#) 1 file

[EPUB](#) 1 file

[FULL TEXT](#) 1 file

[KINDLE](#) 1 file

[PDF](#) 1 file

[SINGLE PAGE ORIGINAL JP2 TAR](#) 1 file

[SINGLE PAGE PROCESSED JP2 ZIP](#) 1 file

[TORRENT](#) 1 file

[SHOW ALL](#) 15 Files

9 Original



Some collections have been processed and have EAD finding aids; others do not. Student workers are paid to create the metadata for each archival object to be scanned. When saved and exported as a CSV file, metadata created for scanning can be repurposed when creating collection finding aids.

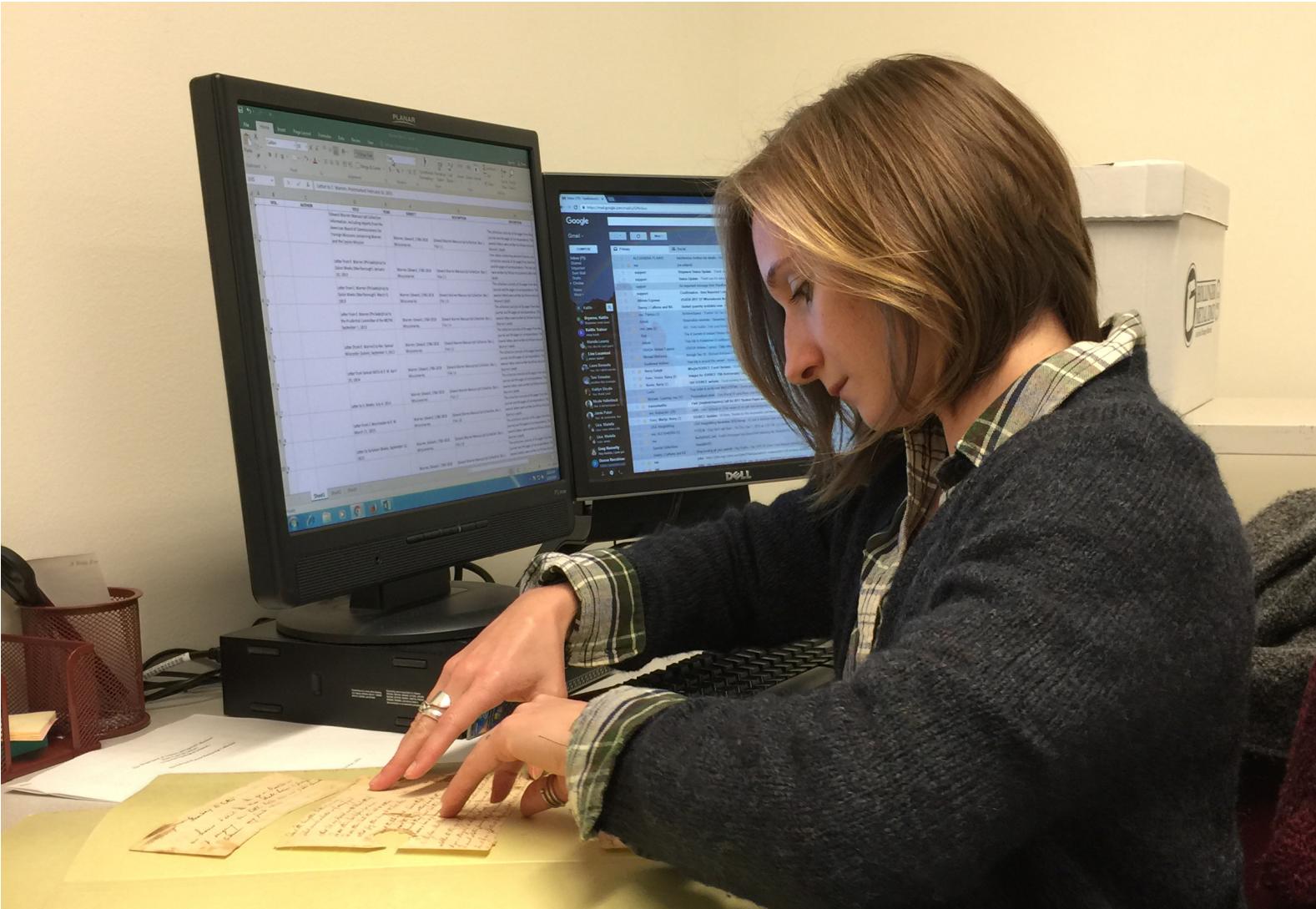
The screenshot shows the ArchivesSpace web application. At the top, there's a navigation bar with 'Select Repository', 'System', and a user account for 'admin'. Below the navigation is a search bar labeled 'Search All Records' with a magnifying glass icon. To the right of the search bar are PTSSC and settings icons.

The main area displays a hierarchical tree view under 'Frederick Neumann papers'. The tree includes categories like 'Academic notebooks, journals, 1920 - 1960', 'Alphabetical subject files, 1929 - 1990' (with sub-categories A-C, C-E, F-J, J-O, P-Z), 'Mixed materials, 1940 - 1970' (with sub-categories Daily planners and diaries, Sermon notecards, Published articles, Posthumous works, Faith and Knowledge, Der Römerbrief, Sermons, Publisher correspondence, Printers' galley), and 'Posthumous works, 1970 - 2000' (with sub-categories Daily planners and diaries, Sermon notecards, Published articles, Posthumous works, Faith and Knowledge, Der Römerbrief, Sermons, Publisher correspondence, Printers' galley).

To the right of the tree view is a table showing the details for each item. The columns are 'Collection', 'Mixed Materials', and 'Carton: 1-18'. The table lists various items such as 'Series', 'File', 'Text', and 'Carton: 1-18' through 'Carton: 18'.

At the bottom of the screen, a detailed view of the 'Frederick Neumann papers' resource is shown. It includes tabs for 'Basic Information', 'Dates', 'Extents', 'Finding Aid Data', 'Agent Links', 'Subjects', 'Notes', and 'Instances'. The 'Basic Information' tab is active, displaying fields like Title (Frederick Neumann papers), Identifier (2003.3 30), Level of Description (Collection), Resource Type (Papers), Language (Multiple languages), Publish? (True), and Restrictions? (False). The 'Dates' tab shows a single entry for Creation: 1929 – 1999, marked as Approximate.

Thus, the Seminary saves time and money by paying one student to create metadata for each object only once.





Describing Archives: A Content Standard (DACS), Second Edition

[Roster](#) [Description](#)

[SAA Standards: Approved](#) | [Technical Standards](#) | [Arrangement and Description](#)

Describing Archives: A Content Standard (DACS) is an output-neutral set of rules for describing archives, personal papers, and manuscript collections, and can be applied to all material types. It is the U.S. implementation of international standards (i.e., ISAD[G] and ISAAR[CPF]) for the description of archival materials and their creators.

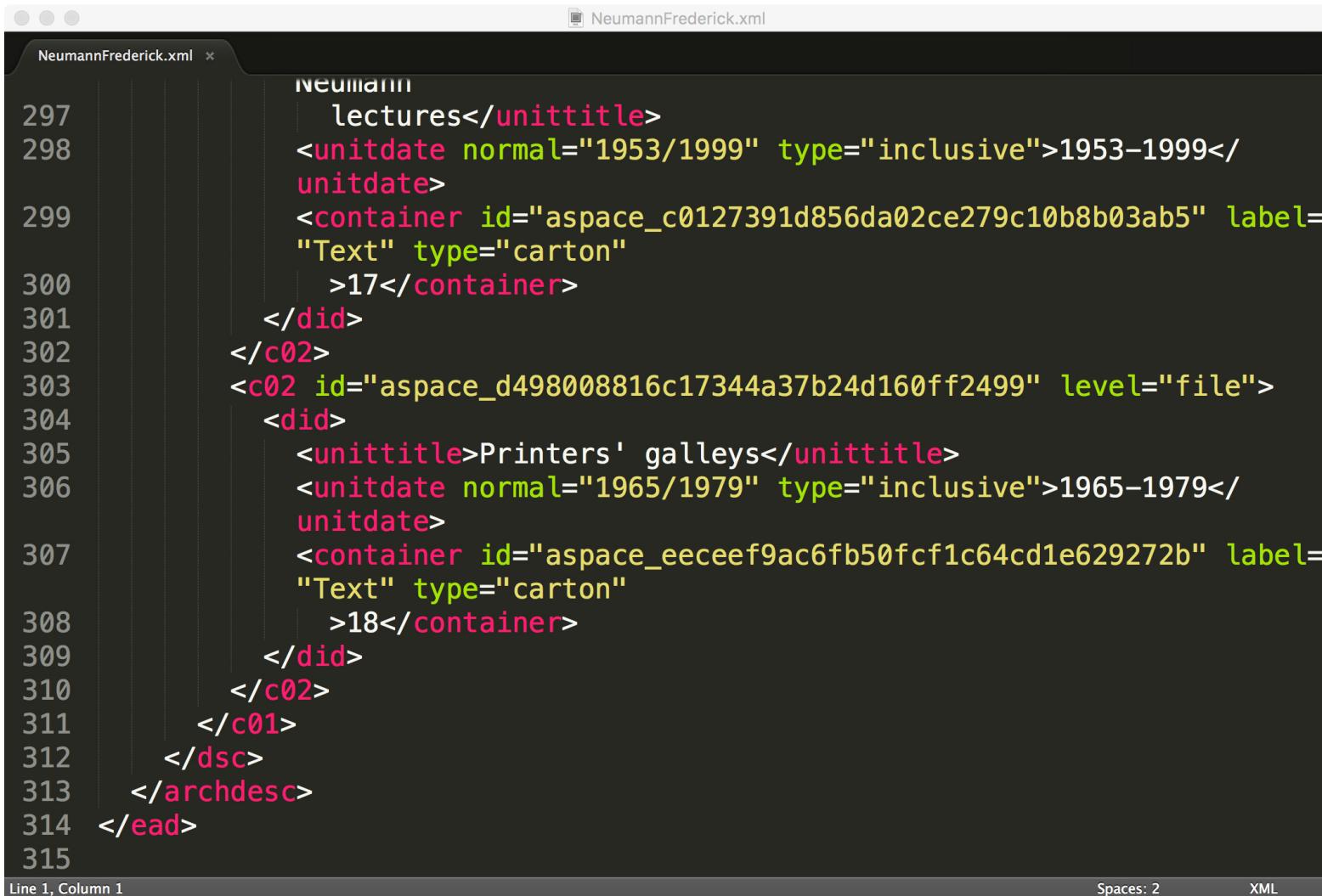
This metadata is pulled from Library of Congress Subject Headings and Name Authority files and complies with the archival descriptive standard Describing Archives: A Content Standard. Normalized across collections, it can link to other collections internally or content from other institutions with digital collections in the Internet Archive.

First, I grabbed the XML EAD finding aid file:

NeumannFrederick.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ead xmlns="urn:isbn:1-931666-22-9" xmlns:xlink="http://www.w3.org/1999/xlink"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="urn:isbn:1-931666-22-9 http://www.loc.gov/ead/ead.xsd">
5     <eadheader countryencoding="iso3166-1" dateencoding="iso8601"
6       langencoding="iso639-2b"
7         repositoryencoding="iso15511">
8           <eadid countrycode="US" mainagencycode="US-NjPT"/>
9             <filedesc>
10               <titlestmt>
11                 <titleproper>Frederick Neumann papers<num>2003.3.30</num></titleproper>
12               </titlestmt>
13               <publicationstmt>
14                 <publisher>Princeton Theological Seminary. Library. Special
15                   Collections</publisher>
16                 <p id="logostmt"><extref xlink:actuate="onLoad"
17                     xlink:href="http://www.ptsem.edu/library/images/PTSLogo-
18                     Library-large.jpg"
19                     xlink:show="embed" xlink:type="simple"/></p>
20               <address><addressline>Princeton Theological Seminary</addressline>
21                 <addressline>Library</addressline><addressline>PO Box 821</addressline>
```

Then, I drilled down through the hierarchy of nested EAD elements to the metadata elements I want to extract:



The screenshot shows a code editor window with the file "NeumannFrederick.xml" open. The code is an EAD (Encoded Archival Description) XML document. The editor has a dark theme with syntax highlighting. The XML structure includes nested elements like `<unittitle>`, `<unitdate>`, `<container>`, and `<did>`. The code is numbered from 297 to 315 on the left side.

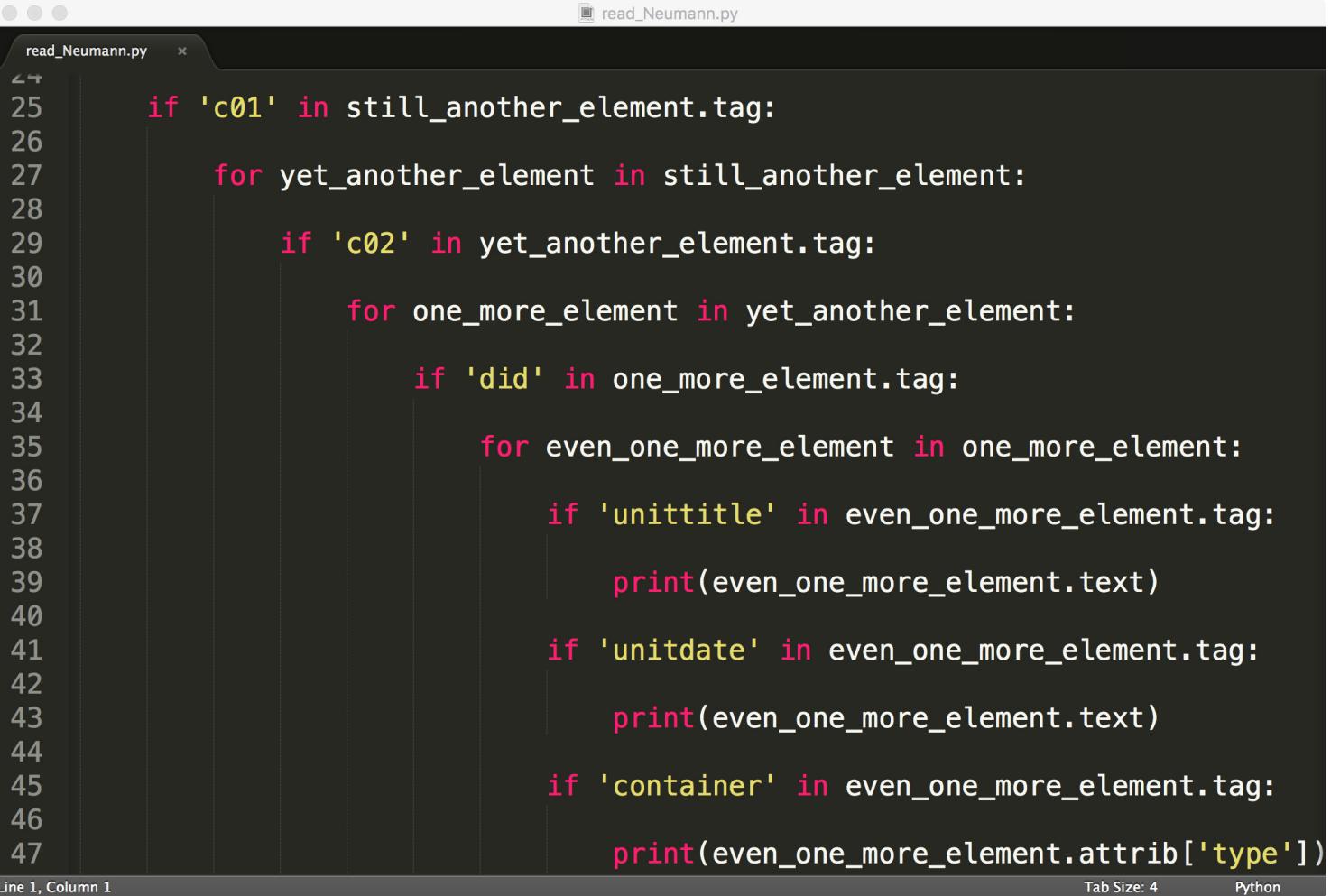
```
NeumannFrederick.xml * 297     lectures</unittitle> 298     <unitdate normal="1953/1999" type="inclusive">1953-1999</ 299     unitdate> 300     <container id="aspace_c0127391d856da02ce279c10b8b03ab5" label= 301     "Text" type="carton" 302     >17</container> 303   </did> 304 </c02> 305 <c02 id="aspace_d498008816c17344a37b24d160ff2499" level="file"> 306   <did> 307     <unittitle>Printers' galley</unittitle> 308     <unitdate normal="1965/1979" type="inclusive">1965-1979</ 309     unitdate> 310     <container id="aspace_eecef9ac6fb50fcf1c64cd1e629272b" label= 311     "Text" type="carton" 312     >18</container> 313   </did> 314 </c02> 315 </c01> 316 </dsc> 317 </archdesc> 318 </ead>
```

Line 1, Column 1 Spaces: 2 XML

Next, I created a Python file and imported the ElementTree Class from the xml.etree module and the csv module, and asked the xml module to load the xml file and parse it:

```
1
2 #Import ElementTree Class from xml.etree module
3 import xml.etree.ElementTree as etree
4
5 #import csv module
6 import csv
7
8 #ask xml module to load xml file and parse it
9 tree = etree.parse('/Users/jrider/GitHub/PFCH16_Final/NeumannFrederick.xml')
10
11 #return the root xml element and store it in root variable
12 root = tree.getroot()
13
```

Then, I looped through the XML file to get to my nested metadata: unittitle, unitdate, and container.



A screenshot of a code editor window titled "read_Neumann.py". The code is written in Python and uses nested loops to parse XML data. The code structure is as follows:

```
25     if 'c01' in still_another_element.tag:
26
27         for yet_another_element in still_another_element:
28
29             if 'c02' in yet_another_element.tag:
30
31                 for one_more_element in yet_another_element:
32
33                     if 'did' in one_more_element.tag:
34
35                         for even_one_more_element in one_more_element:
36
37                             if 'unittitle' in even_one_more_element.tag:
38
39                                 print(even_one_more_element.text)
40
41                             if 'unitdate' in even_one_more_element.tag:
42
43                                 print(even_one_more_element.text)
44
45                             if 'container' in even_one_more_element.tag:
46
47                                 print(even_one_more_element.attrib['type'])
```

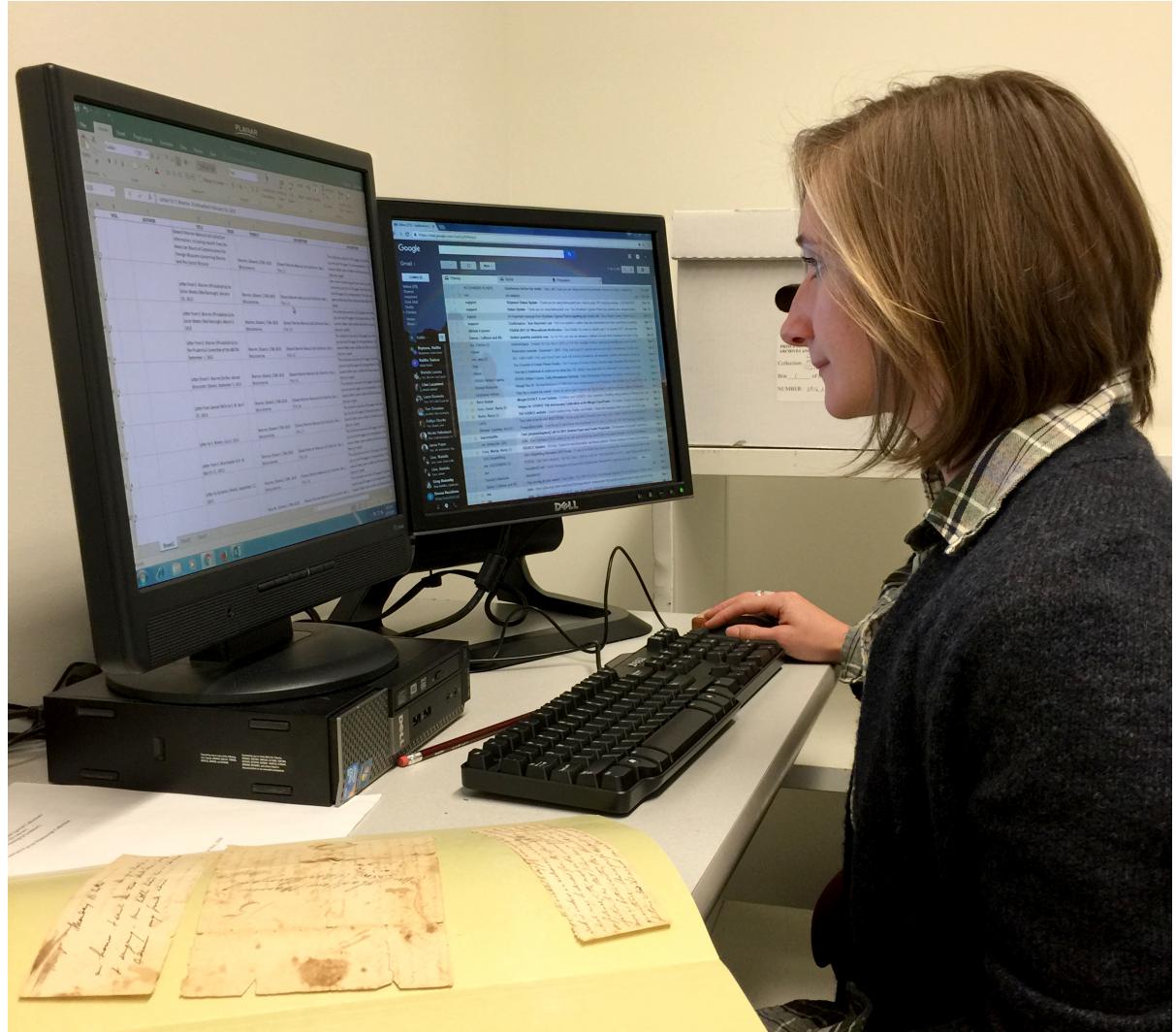
The code is color-coded: numbers are grey, strings are yellow, and keywords like "if", "in", "for", "print", and "return" are pink. The code editor interface includes tabs for "Line 1, Column 1", "Tab Size: 4", and "Python".

I exported that metadata to a CSV file, which can accompany items being sent to the Internet Archive for scanning:

```
read_Neumann.py *  
36 if 'did' in one_more_element.tag:  
37     csv_row = {'unittitle':'','unitdate':'','container':''}  
38  
39     for even_one_more_element in one_more_element:  
40  
41         if 'unittitle' in even_one_more_element.tag:  
42  
43             print(even_one_more_element.text)  
44             csv_row['unittitle']=even_one_more_element.text  
45         if 'unitdate' in even_one_more_element.tag:  
46  
47             print(even_one_more_element.text)  
48             csv_row['unitdate']=even_one_more_element.text  
49  
50         if 'container' in even_one_more_element.tag:  
51  
52             print(even_one_more_element.attrib['type'])  
53             csv_row['container']=even_one_more_element.attrib['type']  
54  
55  
56  
57     metadatawriter.writerow([csv_row['unittitle'],csv_row['unitdate'],csv_row['container']])  
58  
59
```

| | | |
|---|-----------|--------|
| A-C | 1930-1960 | carton |
| C-E | 1930-1980 | carton |
| F-J | 1929-1968 | carton |
| J-O | 1933-1980 | carton |
| P-Z | 1931-1990 | carton |
| Daily planners and diaries | 1950-1967 | carton |
| Sermon notecards | 1940-1960 | box |
| Published articles, works by others, biographical materials | 1875-1997 | carton |
| Faith and Knowledge, God's Fifth Columnist, the Jewish Question | 1937-1994 | carton |
| Der RÄmerbrief | 1943-1996 | carton |
| Sermons, Sermon on the Mount | 1943-1990 | carton |
| Publisher correspondence, publicity materials, PTS Neumann lectures | 1953-1999 | carton |
| Printers' galley | 1965-1979 | carton |

Result is normalized and repurposed metadata.
Last step: use the Python command module to convert all of this script to one command line a student worker can easily run without extensive coding:



The command/sys module presents an argument that pulls the scripts together into one command. Working in the terminal, a student worker simply changes the xml file being parsed.

```
11
12 tree = etree.parse(sys.argv[1])
13
14 #return the root xml element and store it in root variable
15 root = tree.getroot()
16
17 with open('metadata.csv', 'w') as csvfile:
18     metadatawriter = csv.writer(csvfile, delimiter=',')
19     #use for loop to loop through root element
20     for a_element in root:
21
```

Success!!

```
PFCH16_Final — -bash — 80x24

Sermon notecards
1940-1960
box
Published articles, works by others, biographical materials
1875-1997
carton
Faith and Knowledge, God's Fifth Columnist, the Jewish Question
1937-1994
carton
Der Römerbrief
1943-1996
carton
Sermons, Sermon on the Mount
1943-1990
carton
Publisher correspondence, publicity materials, PTS Neumann
    lectures
1953-1999
carton
Printers' galley
1965-1979
carton
jackierdersmbp3:PFCH16_Final jrider$ python3.5 read_Neumann.py '/Users/jrider/GitHub/PFCH16_Final/NeumannFrederick.xml'
```