Repurposing Archival Metadata with the Python CSV Writer

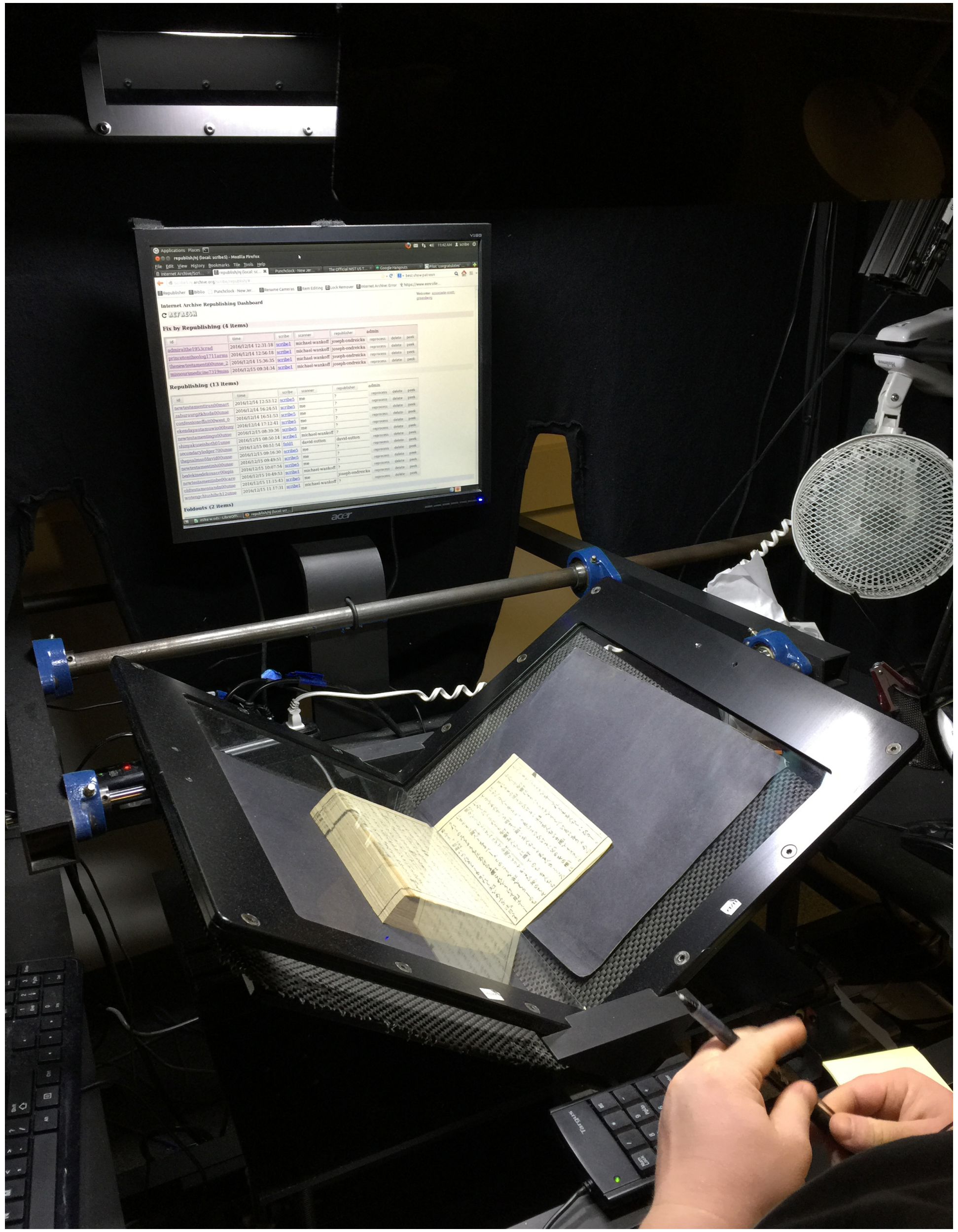
Jackie Rider

Programming for Cultural Heritage 2016

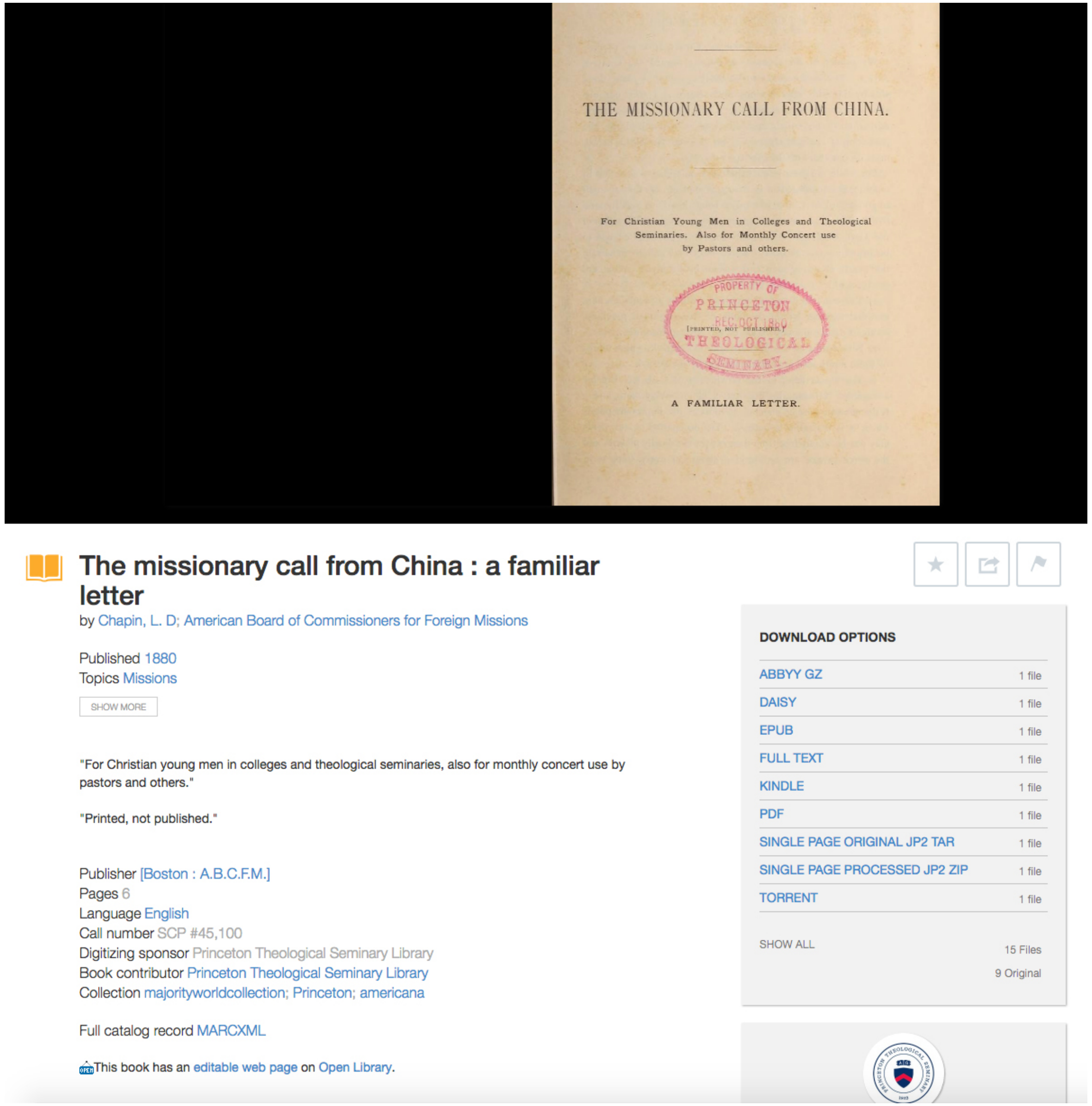
December 15, 2016

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.

For Christian Young Men in Colleges and Theological Seminaries. Also for Monthly Concert use by Pastors and others.

[PRINTED, NOT PUBLISHED.]

A FAMILIAR LETTER.

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

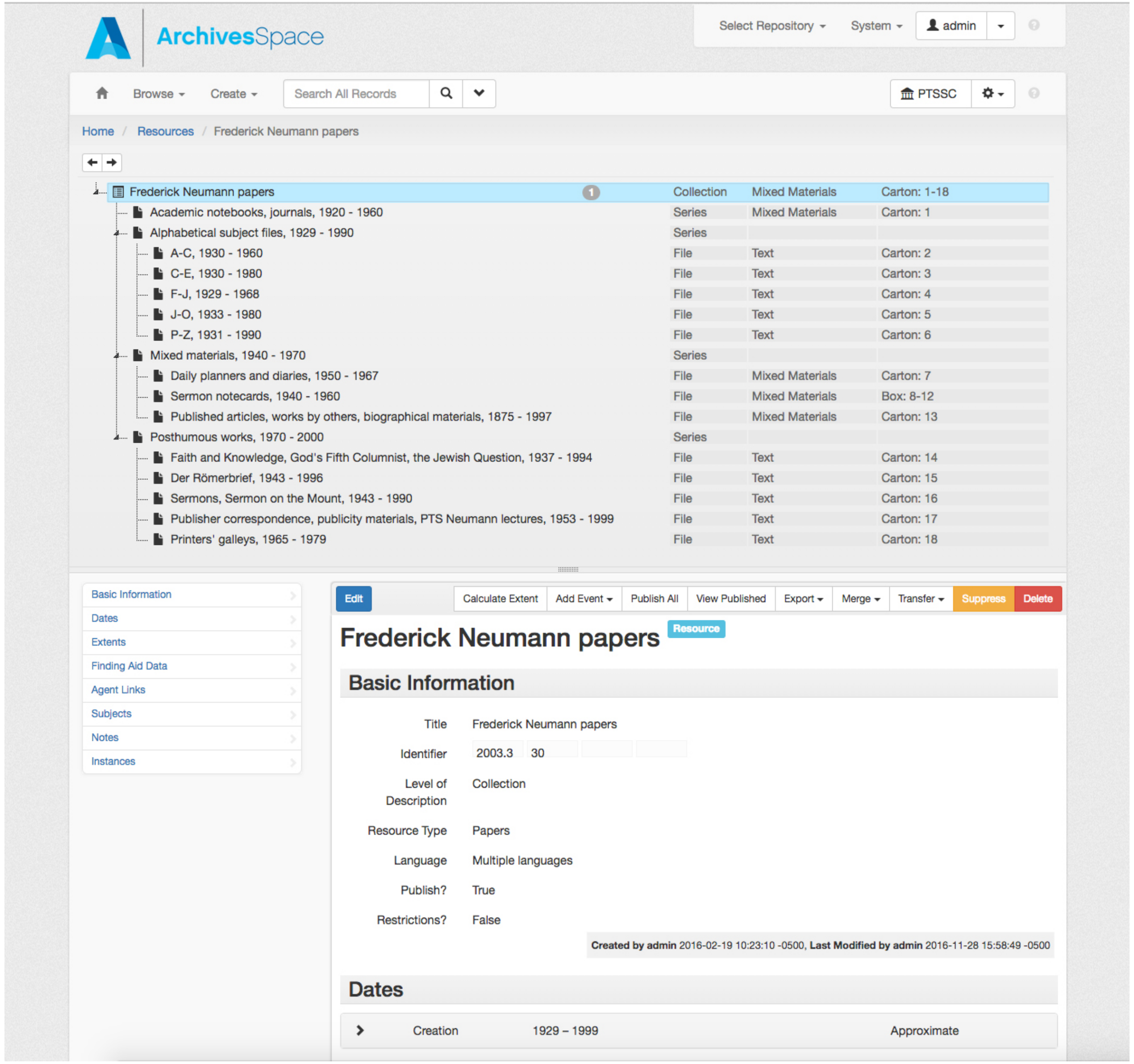
This book has an editable web page on Open Library.

DOWNLOAD OPTIONS

ABBYY 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.



Frederick Neumann papers	Collection	Mixed Materials	Carton: 1-18
Academic notebooks, journals, 1920 - 1960	Series	Mixed Materials	Carton: 1
Alphabetical subject files, 1929 - 1990	Series		
A-C, 1930 - 1960	File	Text	Carton: 2
C-E, 1930 - 1980	File	Text	Carton: 3
F-J, 1929 - 1968	File	Text	Carton: 4
J-O, 1933 - 1980	File	Text	Carton: 5
P-Z, 1931 - 1990	File	Text	Carton: 6
Mixed materials, 1940 - 1970	Series		
Daily planners and diaries, 1950 - 1967	File	Mixed Materials	Carton: 7
Sermon notecards, 1940 - 1960	File	Mixed Materials	Box: 8-12
Published articles, works by others, biographical materials, 1875 - 1997	File	Mixed Materials	Carton: 13
Posthumous works, 1970 - 2000	Series		
Faith and Knowledge, God's Fifth Columnist, the Jewish Question, 1937 - 1994	File	Text	Carton: 14
Der Römerbrief, 1943 - 1996	File	Text	Carton: 15
Sermons, Sermon on the Mount, 1943 - 1990	File	Text	Carton: 16
Publisher correspondence, publicity materials, PTS Neumann lectures, 1953 - 1999	File	Text	Carton: 17
Printers' galleys, 1965 - 1979	File	Text	Carton: 18

Frederick Neumann papers

Basic Information

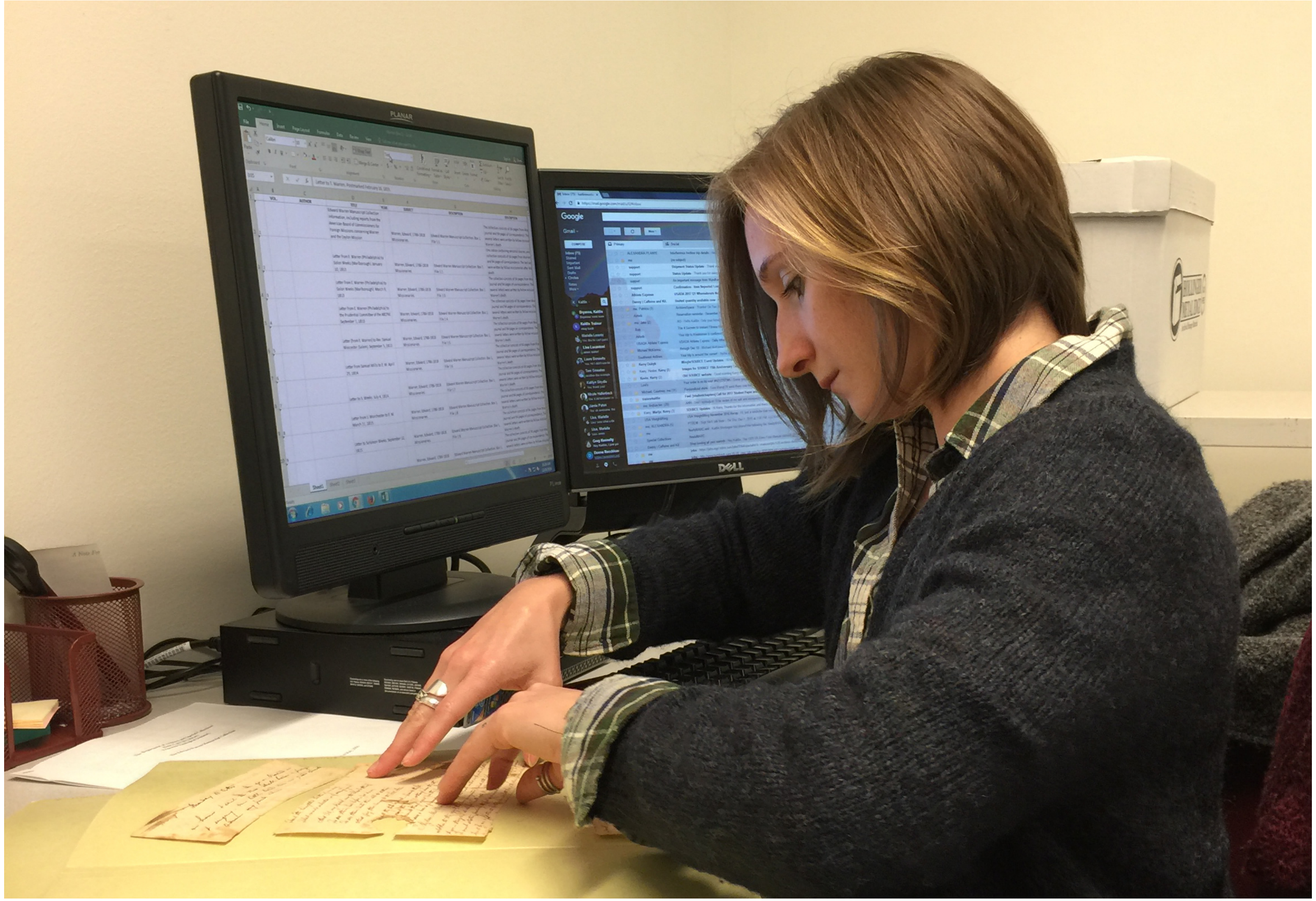
Title	Frederick Neumann papers
Identifier	2003.3 30
Level of Description	Collection
Resource Type	Papers
Language	Multiple languages
Publish?	True
Restrictions?	False

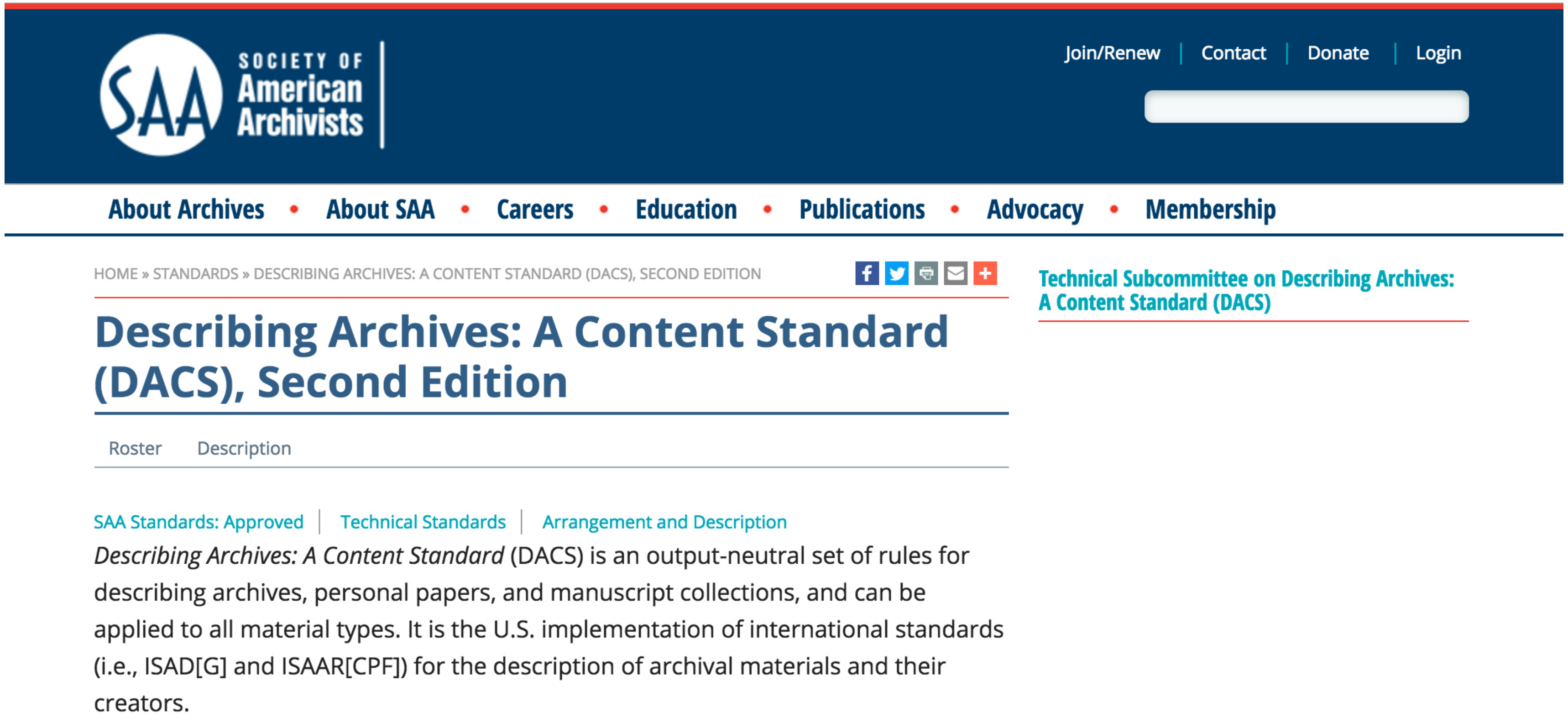
Created by admin 2016-02-19 10:23:10 -0500, **Last Modified by admin** 2016-11-28 15:58:49 -0500

Dates

Creation	1929 – 1999	Approximate

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





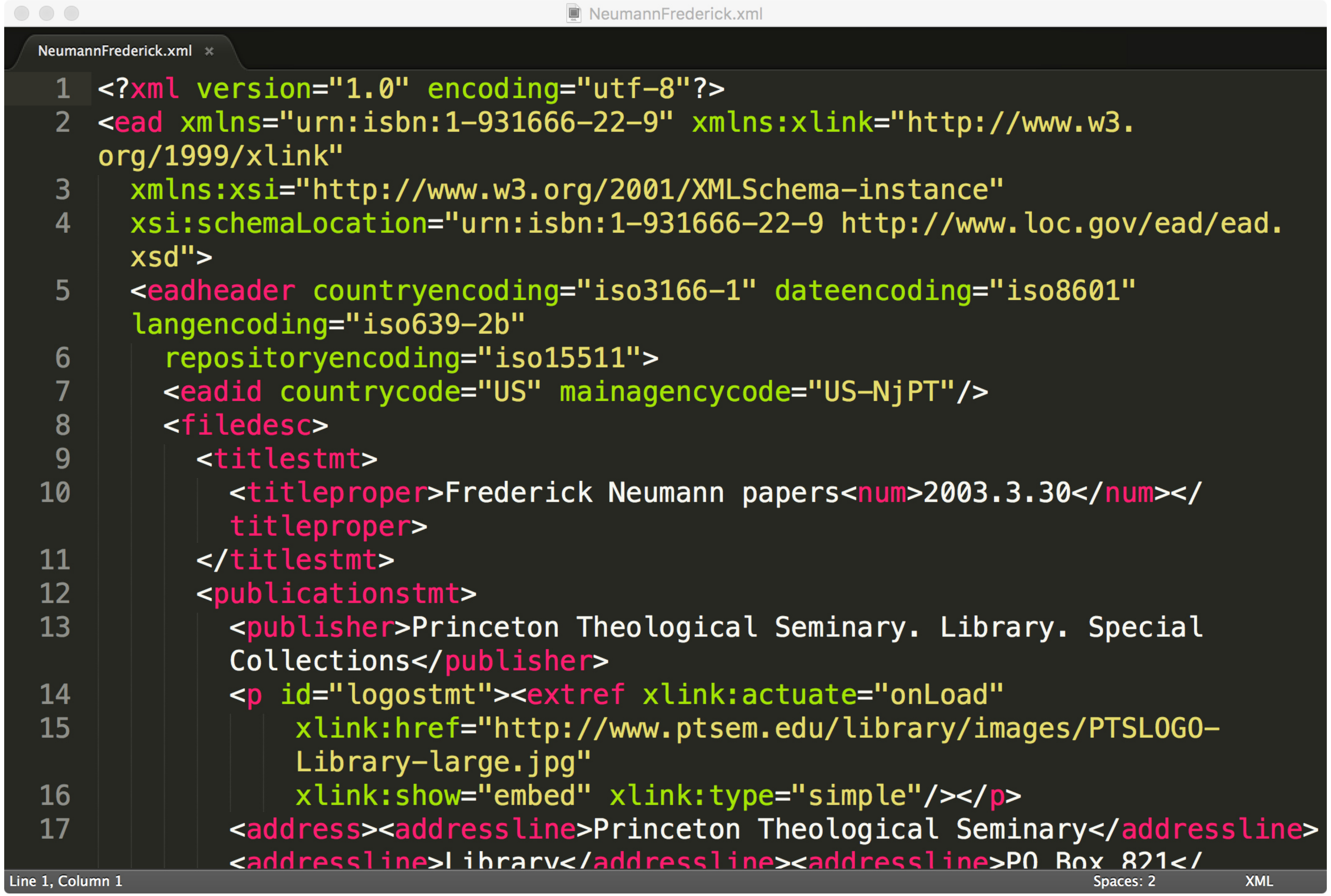
SOCIETY OF American Archivists
Join/Renew | Contact | Donate | Login
About Archives • About SAA • Careers • Education • Publications • Advocacy • Membership
HOME » STANDARDS » DESCRIBING ARCHIVES: A CONTENT STANDARD (DACS), SECOND EDITION
Technical Subcommittee on Describing Archives: A Content Standard (DACS)
Roster
Description
SAA Standards: Approved | Technical Standards | Arrangement and Description

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

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.

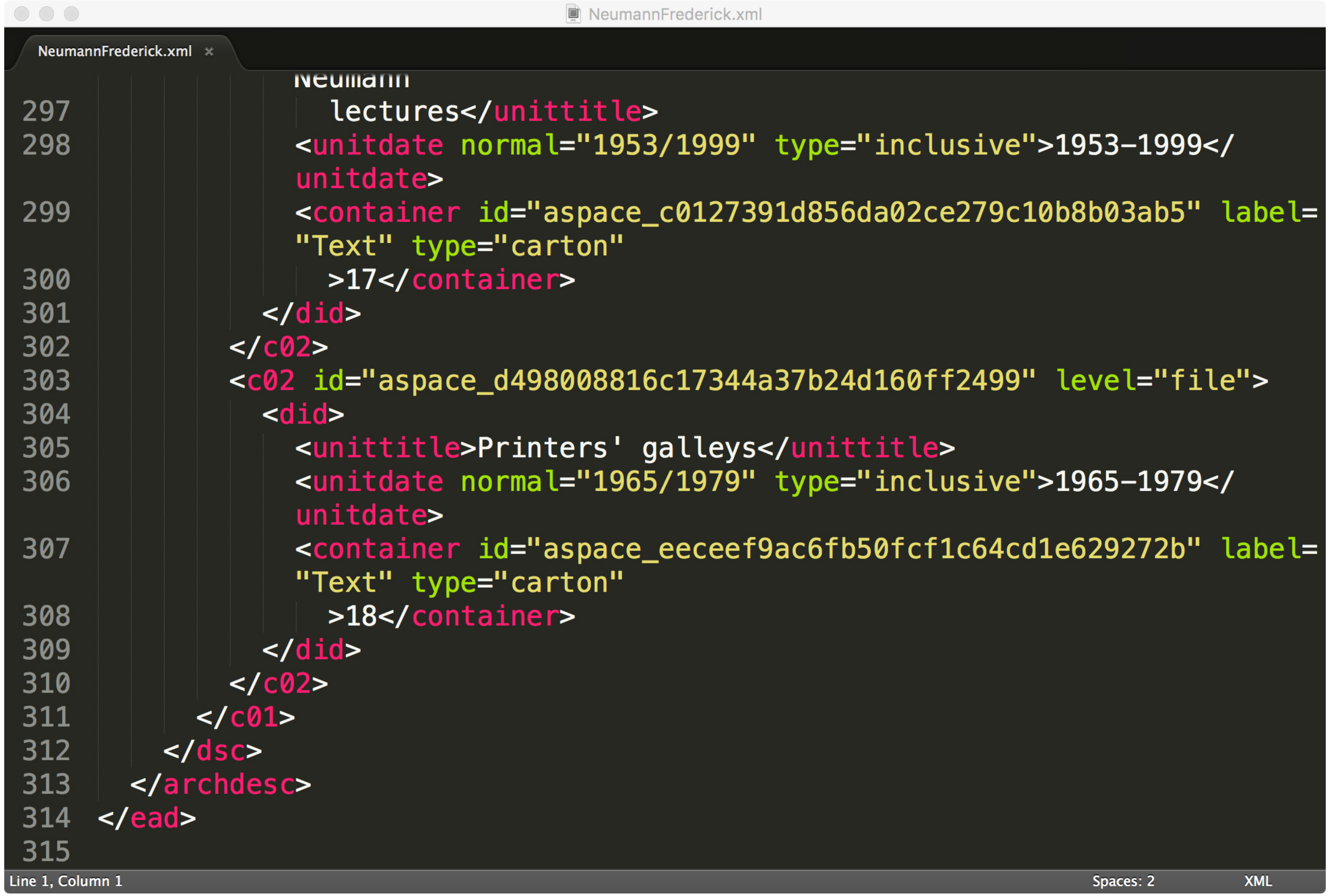
This metadata is pulled from Library of Congress Subject Headings and Name Authority files and complies with the archival descriptive standard Describing Archives: A Content Standard. Normalized across collections, it can link to other collections internally or content from other institutions with digital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

First, I grabbed the XML EAD finding aid file:



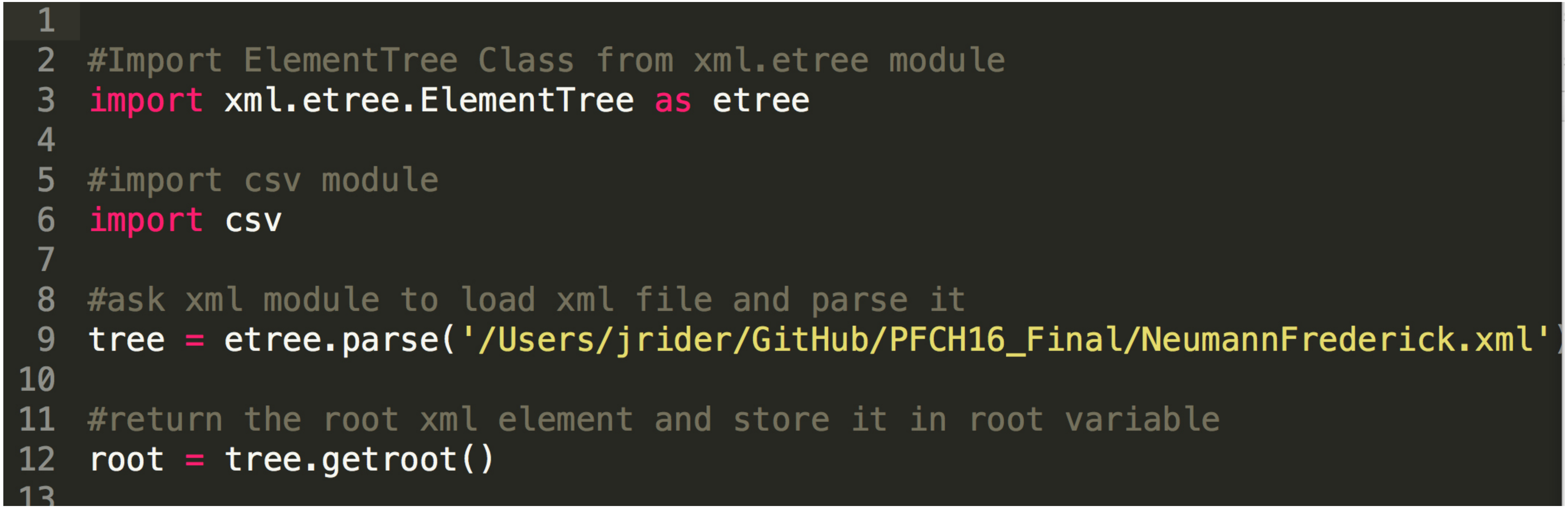
```
<?xml version="1.0" encoding="utf-8"?>
<ead xmlns="urn:isbn:1-931666-22-9" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:isbn:1-931666-22-9 http://www.loc.gov/ead/ead.xsd">
  <eadheader countryencoding="iso3166-1" dateencoding="iso8601"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

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



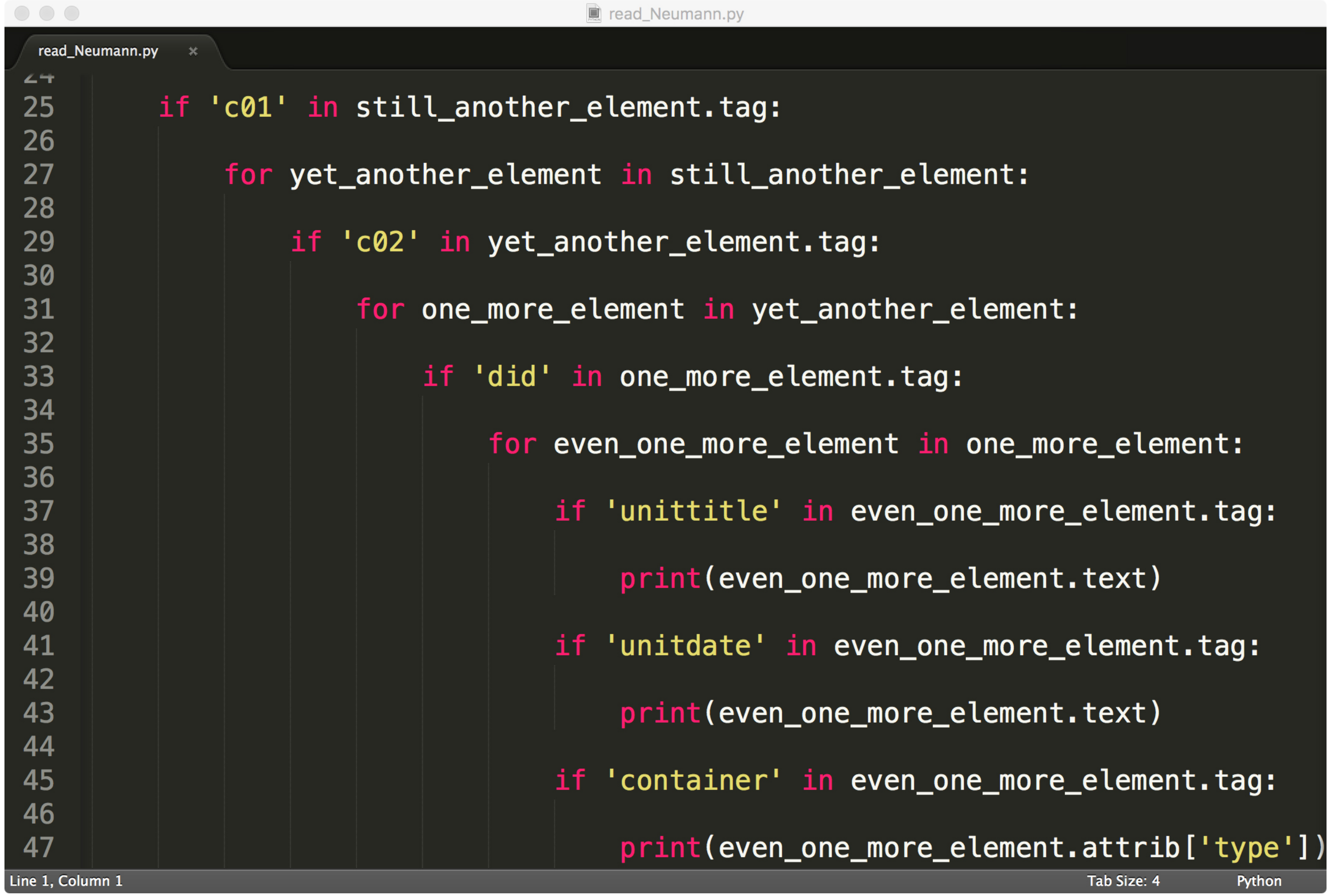
```
            Neumann
              lectures</unittitle>
            <unitdate normal="1953/1999" type="inclusive">1953-1999</
            unitdate>
            <container id="aspace_c0127391d856da02ce279c10b8b03ab5" label=
            "Text" type="carton"
              >17</container>
          </did>
        </c02>
        <c02 id="aspace_d498008816c17344a37b24d160ff2499" level="file">
          <did>
            <unittitle>Printers' galleys</unittitle>
            <unitdate normal="1965/1979" type="inclusive">1965-1979</
            unitdate>
            <container id="aspace_eeceef9ac6fb50fcf1c64cd1e629272b" label=
            "Text" type="carton"
              >18</container>
          </did>
        </c02>
      </c01>
    </dsc>
  </archdesc>
</ead>
```

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:



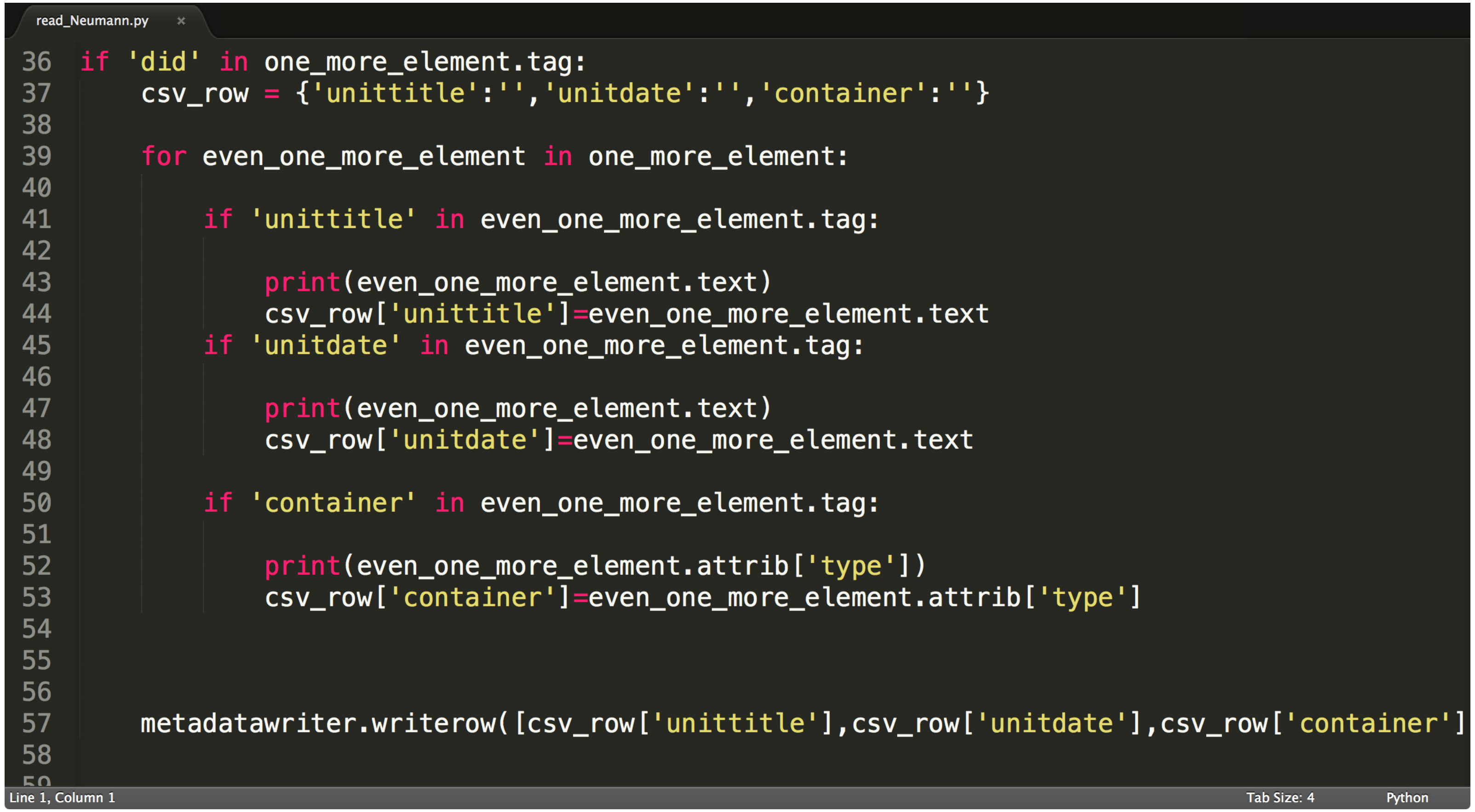
```python

#Import ElementTree Class from xml.etree module
import xml.etree.ElementTree as etree

#import csv module
import csv

#ask xml module to load xml file and parse it
tree = etree.parse('/Users/jrider/GitHub/PFCH16_Final/NeumannFrederick.xml')

#return the root xml element and store it in root variable
root = tree.getroot()

```

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



```
        if 'c01' in still_another_element.tag:

            for yet_another_element in still_another_element:

                if 'c02' in yet_another_element.tag:

                    for one_more_element in yet_another_element:

                        if 'did' in one_more_element.tag:

                            for even_one_more_element in one_more_element:

                                if 'unittitle' in even_one_more_element.tag:

                                    print(even_one_more_element.text)

                                if 'unitdate' in even_one_more_element.tag:

                                    print(even_one_more_element.text)

                                if 'container' in even_one_more_element.tag:

                                    print(even_one_more_element.attrib['type'])
```

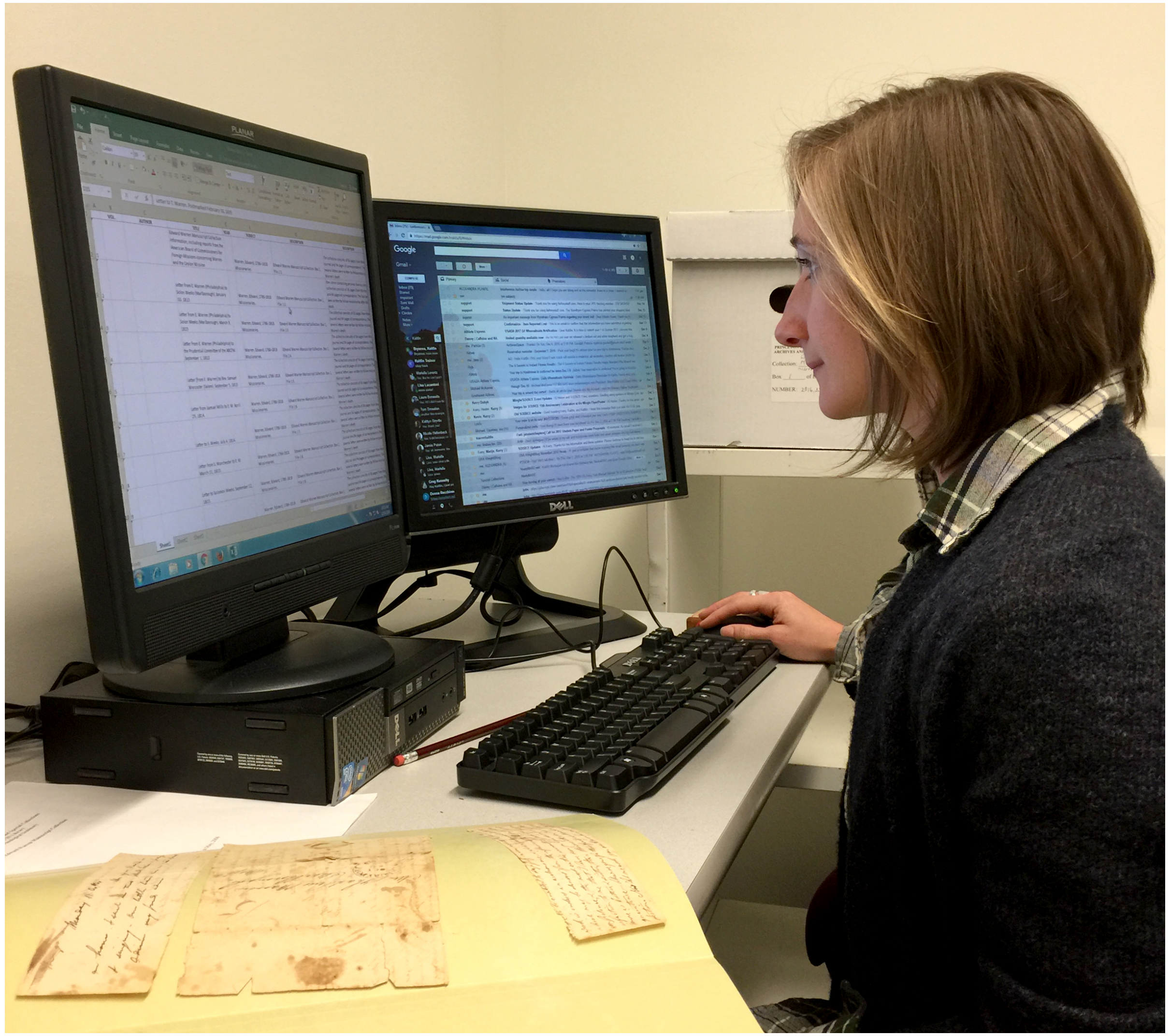
I exported that metadata to a CSV file, which can accompany items being sent to the Internet Archive for scanning:



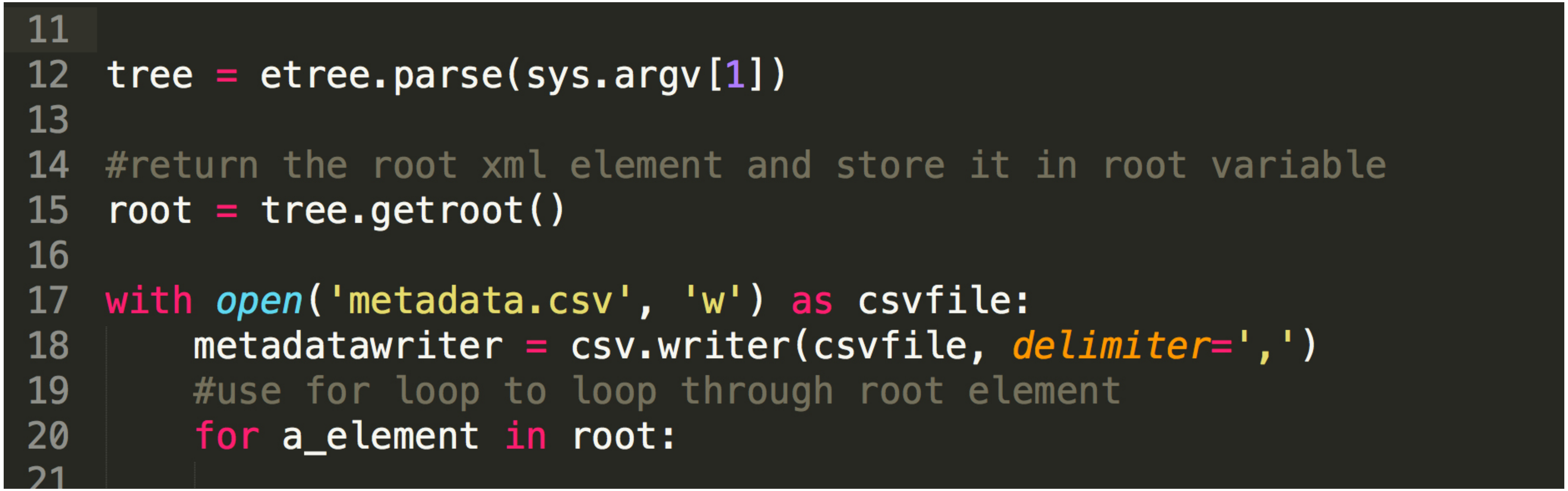
```python
if 'did' in one_more_element.tag:
    csv_row = {'unittitle':'','unitdate':'','container':''}

    for even_one_more_element in one_more_element:

        if 'unittitle' in even_one_more_element.tag:

            print(even_one_more_element.text)
            csv_row['unittitle']=even_one_more_element.text
        if 'unitdate' in even_one_more_element.tag:

            print(even_one_more_element.text)
            csv_row['unitdate']=even_one_more_element.text

        if 'container' in even_one_more_element.tag:

            print(even_one_more_element.attrib['type'])
            csv_row['container']=even_one_more_element.attrib['type']



    metadatawriter.writerow([csv_row['unittitle'],csv_row['unitdate'],csv_row['container']
```

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 RÃ¶merbrief	1943-1996	carton
Sermons, Sermon on the Mount	1943-1990	carton
Publisher correspondence, publicity materials, PTS Neumann lectures	1953-1999	carton
Printers' galleys	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:

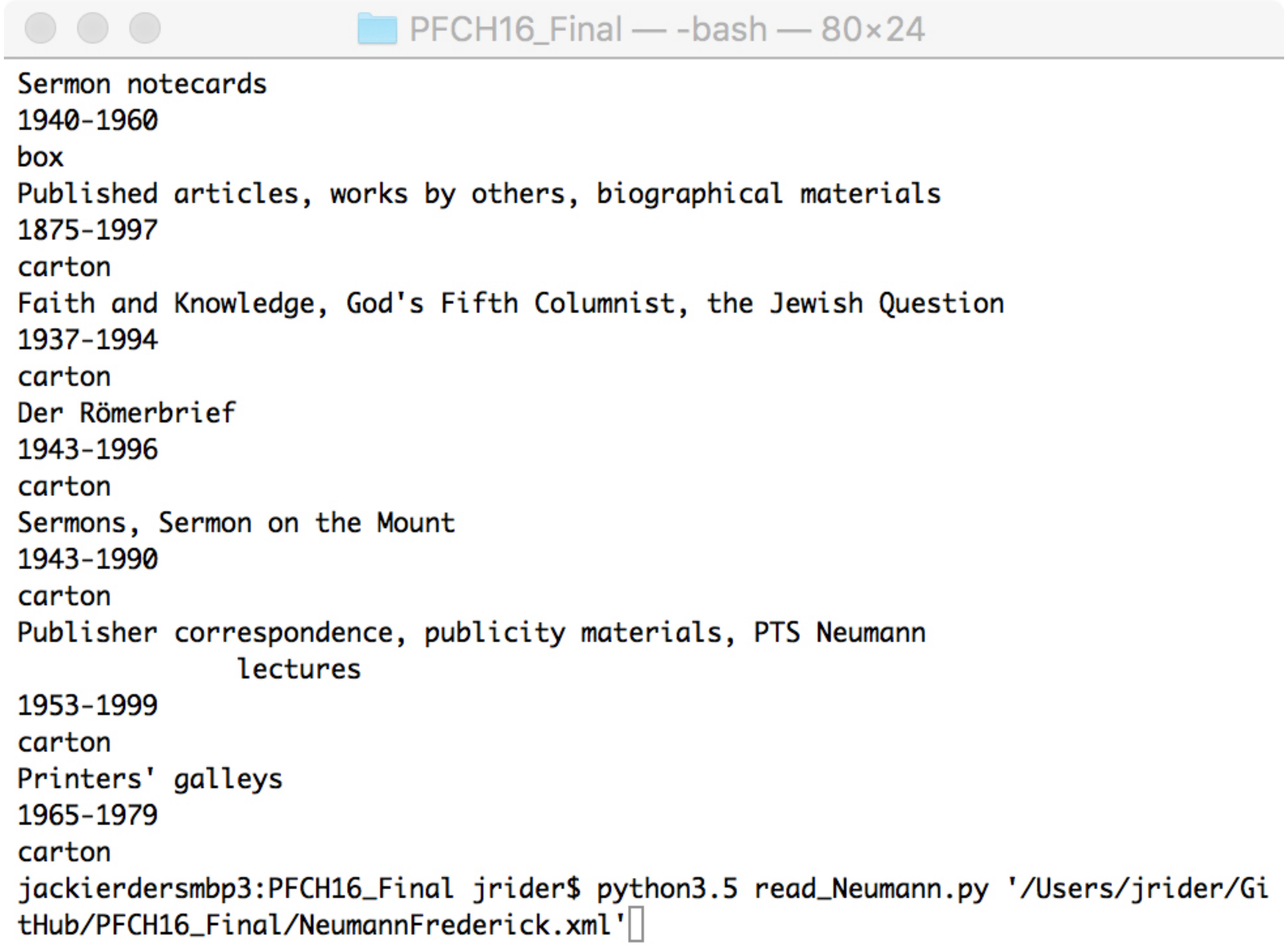


The command/sys module presents an argument that pulls the scripts together into one command. Working in the terminal, a student worker simply changes the xml file being parsed.



```

tree = etree.parse(sys.argv[1])

#return the root xml element and store it in root variable
root = tree.getroot()

with open('metadata.csv', 'w') as csvfile:
    metadatawriter = csv.writer(csvfile, delimiter=',')
    #use for loop to loop through root element
    for a_element in root:

```

Success!!



```
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' galleys
1965-1979
carton
jackierdersmbp3:PFCH16_Final jrider$ python3.5 read_Neumann.py '/Users/jrider/Gi
tHub/PFCH16_Final/NeumannFrederick.xml'
```