

Coding for Catalogers

A Practical Approach to Programming

Carolyn Hansen

#alaac19 | #catalogingNormsSIG | June 22, 2019

#coding4catalogers

about me

>>> I'm a librarian, not a professional developer or programmer

>>> I resisted learning how to code for a long time because I was intimidated

>>> coding has made my cataloging life easier, empowered me, and improved my ability to communicate with IT and systems folks

quick thanks to the folks
who helped me learn to code

>>> Sean Crowe, Digital Analyst &
Developer Librarian, University of
Cincinnati

>>> James Van Mil, Digital Analyst &
Developer Librarian, University of
Cincinnati

outline

>>> coding 101 + why coding for catalogers?

>>> tools you can use with your code

>>> cataloging examples

>>> how to get started + tips

what is coding?

>>> using a machine language to write instructions that a computer can understand

>>> you may also hear coding referred to as “programming” or “scripting” – these terms have subtle differences in meaning

when to use code

>>> you're about to do something manual and repetitive to your data

>>> you're working with a large set of records and need to make consistent changes across the set

>>> bulk formatting, splitting, joining, or standardizing, especially with tabular data like spreadsheets

how can coding help catalogers?

- >>> automate bulk changes to large record sets
- >>> standardize and cleanup tabular data
- >>> transform records from one metadata standard to another
- >>> extract metadata and data from non-library sources to create new record sets
- >>> enhance, correct, and cleanup vendor records
- >>> fix data encoding issues

what coding isn't for...

>>> artisanal metadata creation
requiring specialized domain
knowledge

So...

>>> computers can't (and shouldn't!)
replace high quality catalogers, but
they can help catalogers automate
the boring stuff

what you need to code

>>> machine with one of the following
Operating Systems (OS) : Windows, Mac,
Linux

>>> administrative access to your
machine (sometimes in more controlled
workplaces this can be a problem)

>>> access to the command line

>>> text editor of choice (ex. Sublime,
Atom, Vim)

what is the command line?

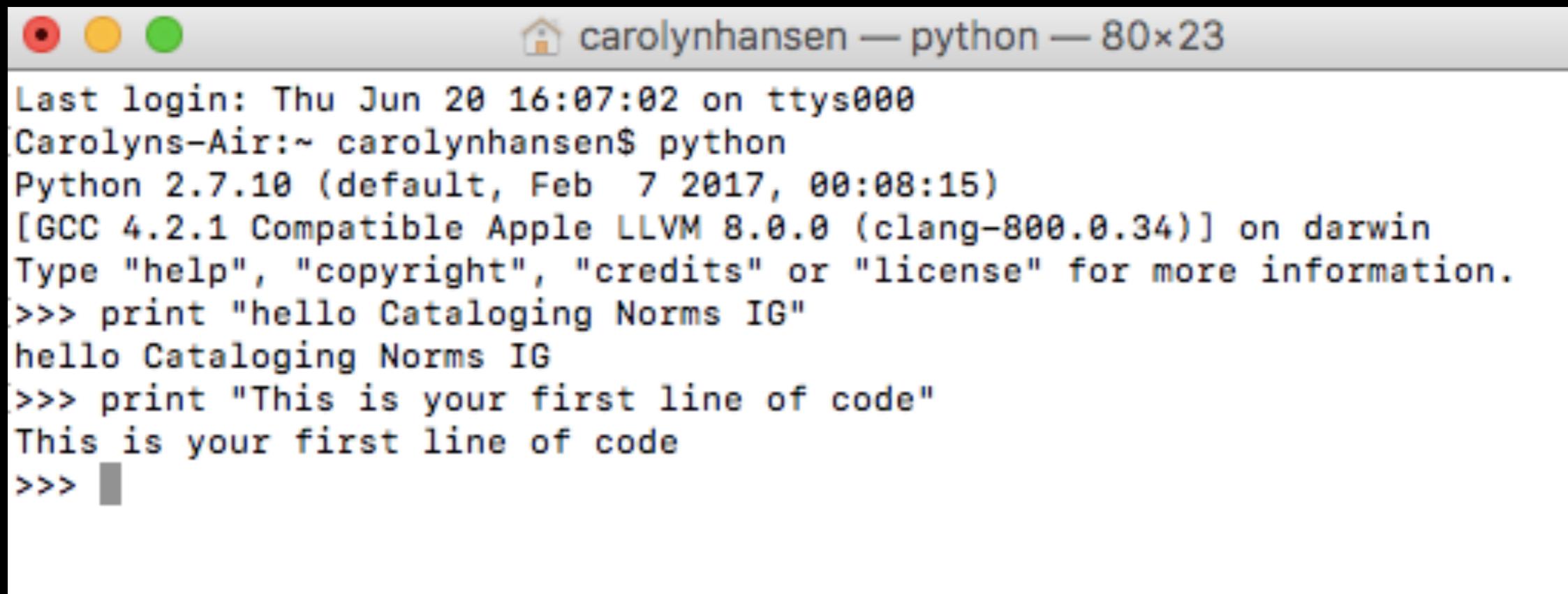
>>> text interface for your computer

>>> program that takes in commands, and passes them to your computer's OS to execute

>>> you can use the command line to navigate files and folders; you can also use it to run code

>>> on Mac or Linux, you can access the command line through terminal – it's a little more complicated on Windows

command line example: super simple code using python



```
Last login: Thu Jun 20 16:07:02 on ttys000
Carolyns-Air:~ carolynhansen$ python
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "hello Cataloging Norms IG"
hello Cataloging Norms IG
>>> print "This is your first line of code"
This is your first line of code
>>>
```

note: most programs (complete, executable lines of code) are written in a text editor and then run using the command line.

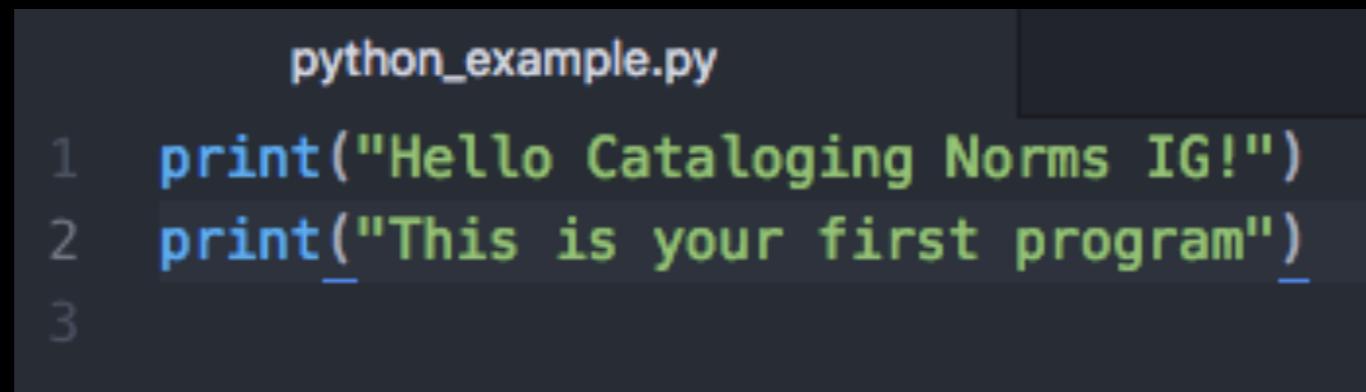
how to execute a short program on the command line

>>> write code in text editor of your choice, save to your local machine with appropriate file extension (ex. for a python script it would be .py)

>>> access the command line, then execute your program

>>> success! (or you'll get an error message)

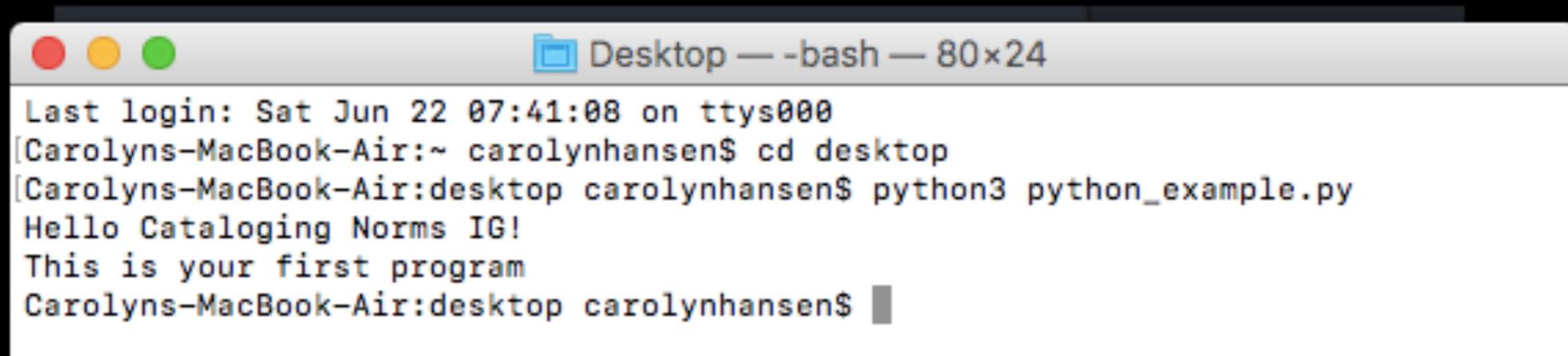
short program using python, written in Atom text editor and saved to my desktop



The screenshot shows a dark-themed code editor window titled "python_example.py". The code contains three lines of Python print statements:

```
1 print("Hello Cataloging Norms IG!")
2 print("This is your first program")
3
```

ran program using terminal on my Mac — first navigated to my desktop using “cd” command, then ran the program



The screenshot shows a terminal window titled "Desktop — -bash — 80x24". The session log includes:

```
Last login: Sat Jun 22 07:41:08 on ttys000
[Carolyns-MacBook-Air:~ carolynhansen$ cd desktop
[Carolyns-MacBook-Air:desktop carolynhansen$ python3 python_example.py
Hello Cataloging Norms IG!
This is your first program
Carolyns-MacBook-Air:desktop carolynhansen$
```

my favorite languages:



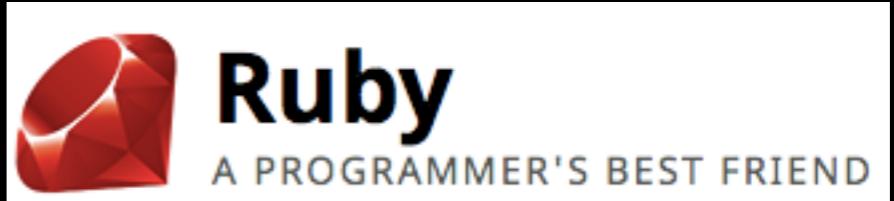
>>> why python?

>>> intuitive and easy to learn,
syntax is not complicated

>>> open source, robust community and
documentation

>>> lightweight and modular, can build
up functionality by adding libraries
(ex. re, csv, glob, numpy, etc)

my favorite languages:



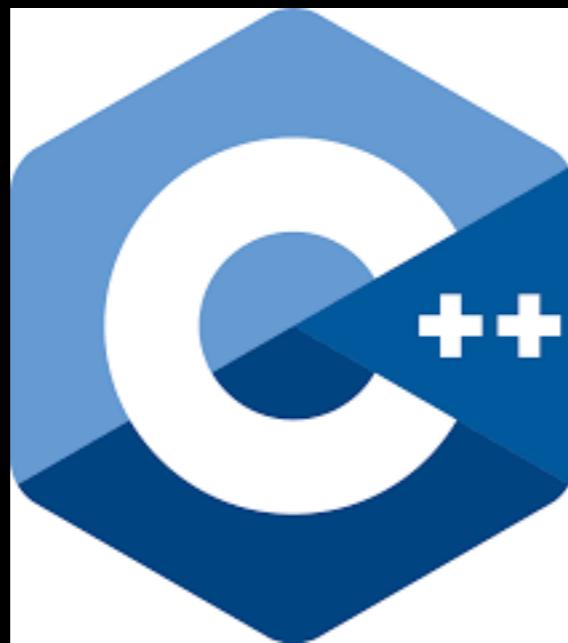
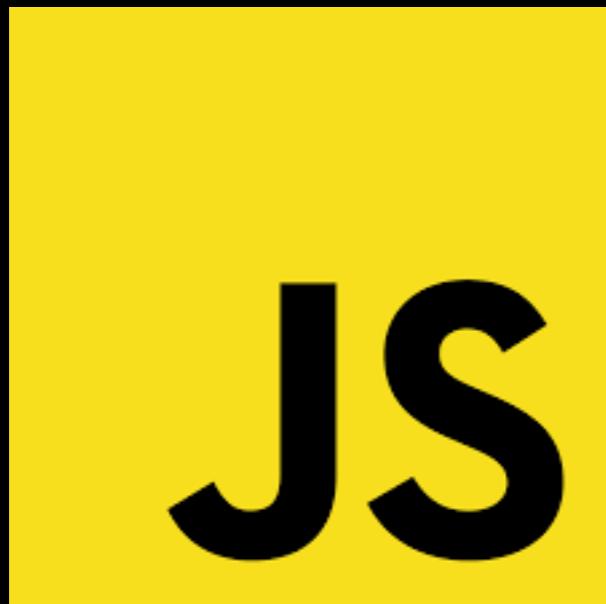
>>> why Ruby?

>>> intuitive and easy to learn,
syntax is not complicated

>>> open source, robust community and
documentation

>>> can be used with gems, which are
bundled packages of Ruby code – there
are thousands of gems available!

other languages to explore:



and many, many more....

how do you decide which language to use?

>>> comfort level and level of experience,
ex. Java is not a good entry level language

>>> what are you trying to do? some
languages are more function specific, ex.
python and Ruby are for automation, while
Javascript can be used for data
visualizations or make dynamic things happen
on a static website

>>> is this an established language? if the
community isn't robust, it will be more
difficult to get help if you need it

tools you can use with your code

>>> regular expressions

>>> shell scripts

>>> XSLT

regular expressions

>>> also known as “regex”

>>> sequence of characters that define a search pattern

>>> like a much more powerful version of “find” or “find and replace” functionality

>>> using regex in your code allows you to make global/bulk changes and standardize metadata

use case: global changes to MARC records using python & regex

```
545         #Global changes for MvI full records
546         #move 001 to 035
547         x = re.sub('=001 gp', r'=035 \\\\$a(GPO)', x)
548         #replace 913 with 949 stem
549         x = re.sub('=913.*\$\c', r'=949 \\1\$a', x)
550         #move 035 OCLC no to 001
551         x = re.sub('=035 \\\\$aoc', '=001 oc', x)
552         #delete 599 and 999 fields
553         x = re.sub('=599.*\n', '', x)
554         x = re.sub('=999.*\n', '', x)
555         #standardize 856 if present
556         x = re.sub('(^=856.*?)\$\u', '\\1$zConnect to resource online$\u', x)
557         x = re.sub('=856 7.', '=856 40', x)
558         # Remove "$2hhttp" wherever it appears
559         x = re.sub('\$\2hhttp', '', x)
560         x = x.split('\n\n')
```

note: lines beginning with # are comments, not lines of executable code
source: https://github.com/crowesn/batch_cave/blob/master/1.3_MvI_BatchEdit.py

shell scripts

>>> computer program that can be run on a Unix shell

>>> allows you to manipulate files, execute programs, and print text

>>> very useful when you need to run code on multiple files within a single directory

>>> shell scripts are like the “duct tape” of programming; they can be used quickly to fix a problem

note: similar functionality can be accomplished with batch scripts in Windows environments

use case:
shell script to process
multiple files in a directory

```
1 #!/bin/bash
2 FILES=./*.xml
3 out=".OUT"
4 for f in $FILES
5 do
6   echo "Processing $f file..."
7   # take action on each file. $f store current file name
8   xsltproc UC_EAD_to_MARC.xsl $f > $f$out
9 done
```

source: https://github.com/crowesn/UC_EAD_to_MARCMXML/blob/master/UC_EAD_to_MARC.sh

XSLT

>>> XSLT = eXtensible stylesheet
language transformation

>>> styling language for XML

>>> sort of like CSS for HTML

use case: EAD to MARCXML (custom) using XSLT

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <xsl:stylesheet version="1.0" xmlns:marc="http://www.loc.gov/MARC21/slim"
3      xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-prefixes="marc">
4      <xsl:import href="MARC21slimUtils.xsl"/>
5      <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
6      <xsl:template match="text()" />
7      <xsl:template match="/ead">
8          <marc:collection xmlns:marc="http://www.loc.gov/MARC21/slim"
9              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10             xsi:schemaLocation="http://www.loc.gov/MARC21/slim http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd">
11             <xsl:variable name="bulkdate" select="//unitdate[@encodinganalog='245$f']"/>
12             <marc:record>
13                 <marc:leader>
14                     <xsl:text>01125npc i2200289Ki 4500</xsl:text>
15                 </marc:leader>
16                 <xsl:choose>
17                     <xsl:when test="$bulkdate!=''">
18                         <marc:controlfield tag="008">
19                             <xsl:text>040320k</xsl:text>
20                             <xsl:value-of select="substring-before($bulkdate, '-')"/>
21                             <xsl:value-of select="substring-after($bulkdate, '-')"/>
22                             <xsl:text>xx\\\\\\\\\\\\\\\\\\\\000\\0\\eng\\d</xsl:text>
23                         </marc:controlfield>
24                     </xsl:when>
```

note: some values are hardcoded and others are variable

source: https://github.com/crowesn/UC_EAD_to_MARCXML/blob/master/UC_EAD_to_MARC_Winkler_2.xls

use case: converting XML ETD metadata to CSV

```
1 import xml.etree.ElementTree as ET
2 import csv
3 import glob
4 import re
5 forCSV = []
6 forCSV.append(['authorname', 'title', 'dept', 'keywords', 'pages', 'abstract', ''])
7
8 for file in glob.glob('*.xml'):
9     print(file)
10    tree = ET.parse(file)
11    root = tree.getroot()
12
13    for item in root.iter('DISS_advisor'):
14        advisorsurname= item.find('DISS_name/DISS_surname').text
15        advisorfirstname= item.find('DISS_name/DISS_fname').text
16        advisormiddlename= item.find('DISS_name/DISS_middle').text
17        advisorname= advisorsurname, advisorfirstname, advisormiddlename
18
19
```

note: distinct libraries are imported at the beginning of the code, showing the modular nature of python
full code: https://github.com/carohansen/SBU_etd_to_csv

sharing your code

>>> if you want to collaborate on code, publish it so other folks can use it, or want version control, GitHub is currently the standard option

>>> GitHub is a hosting service and creating an individual account is free

>>> if you put a project on GitHub, you can access and edit it through the command line on your local machine using Git commands. There are also desktop clients that you can use if you prefer.

>>> GitHub has some social network-y components (ex. you can follow accounts, there are feeds, etc)

how can you get started?

>>> there are lots of online tools for the basics and getting your feet wet

>>> My favorites:

>>> Codeacademy (basic service is free and you can upgrade to premium version for a fee)

>>> w3schools (free)

>>> GitHub CodeCamp (free)

tips

>>> having a real life project using your library's data will make your coding experience more meaningful

>>> just like with human language, if you don't practice, you will forget what you learned

>>> having consistent, scheduled coding time and someone to practice with keeps you on track. I've used weekly hackathons in the past and found those to be very effective.

resources

www.codecademy.com

github.com

github.com/freeCodeCamp

www.python.org

www.ruby-lang.org

www.stackoverflow.com

www.w3schools.com

Automate the Boring Stuff with Python: Practical Programming for Total Beginners. Written by Al Sweigart.
Free to read under a CC license:
automatetheboringstuff.com

thank you!

```
>>> email:  
carolyn.hansen@stonybrook.edu  
  
>>> twitter: @meta_caro  
  
>>> questions?
```