Jck2 – 4 credit hours

IS452

Final Project Narrative

21 December 2017

# Introduction

My final project ended up taking the form of a few files—several scripts, two spreadsheets, and a new access file. The original goal of the project was to append metadata to access files for researchers using born-digital material in an archive and to do some simple redaction. Redaction ended up being a bit overly ambitious, so I scrapped that component fairly early in the project.

# Run the code!

## file\_count\_by\_type.py

Option 1: file path provided by user; .py must also be found at that file path

Option 2: use the file path commented out in script (line 11)

## create\_access\_file.py

No input is required from user, but you must have droid\_profile\_md5\_included.csv file to run script and you must have already created “access files” in the current directory that contains create\_access\_files.py and droid\_profile\_md5\_included.csv.

# file\_count\_by\_type.py

**input:** file path provided by user, or file path commented out in script (line 11)

**output:** two .txt files – file\_type\_output.txt & file\_paths\_output.txt

## Narrative:

This script returns a list of files and file paths in a directory as well as a count of how many files of a certain number of file types are present in that directory. This is what I first started to play around with when I was working on my final even though I know there are lots of tools that can do this for you and typical operating system “Get Info” windows can tell you this as well. What I wanted to do was get comfortable navigating directories in a Python script. That took me a lot longer than I thought, but I eventually figured out how to do so and wanted to submit this to show some of the work I did prior to the heftier script I wrote.

I had to do a little research about using Python’s os module, but this didn’t take a whole lot of effort. I first figured out how to iterate through a directory so I could gather up a list of all the files and folders. At virtually every step, I made sure that I had a way to compare my work to the actual directory I was working with and confirm that my script was working correctly. I usually did this with simple print() or print(len()) statements.

After I pulled all the files and folders into two separate lists, I was able to work with just the files themselves. I wanted to get a sense of what kind of word processing/text documents were in this directory so I used a simple if/elif/else statement to separate my file list into lists of .txt, .doc and .docx, .pdf, and other files. At the end, I decided to write this data out to a plain text file.

I also wanted to generate a list of files and their file paths. What I had to do was keep track of the file names and the complete file path leading up to that file name. As I iterated through each folder, I took the file name and appended it to the file path for the folder I was working in, plus a backslash between the file’s path and file name. I initially had some trouble with this because I forgot to do this work with the files in the top level folder, but I was later able to go back and copy/paste the code I used for the files in subfolders. I probably could have written a simple function so that I didn’t repeat this code, but I didn’t want to over-engineer this part of my work, especially since this was more of a warm up for the rest of my final than anything else!

# create\_access\_file.py

**input:** no input required from user, but must have droid\_profile\_md5\_included.csv file to run script, must have already created “access files” in current directory, and must have file you want to create an access file for (in this case “The Bill of Rights as a Constitution.txt”)

**output:** droid\_metadata\_for\_access\_files.csv & access file, The Bill of Rights as a Constitution\_access-file.txt

## Narrative:

This script is the result of several chunks of code that I tested separately and then pulled together into one large program. In order to run this program, two things already have to be in place: the droid\_profile\_md5\_included.csv and an empty folder called “access files” in the current directory. This program creates a subset of the metadata in the DROID profile .csv and appends it to a .txt file, which is a copy of a born-digital item in a theoretical manuscript collection, and then creates an access file for users.

#### droid\_data\_for\_access\_files()

Because I decided to pull all my work into one large program, there are two main functions in the program. The first is the droid\_data\_for\_access\_files() function, which takes a DROID profile .csv file as its one and only argument. This function takes that .csv and parses through it to create a subset of metadata that will be of interest to the researcher and omits some of the more technical file metadata that (in my imagination) is of minimal value.

In order to write this function, I needed to learn how to work with .csv data and imported the Python csv module. I figured out how to do some of this by referring to a Python textbook I cataloged in my IS507 class over the summer and also consulted the Python 3 documentation. (Woe is the google searcher who doesn’t realize Python 2 docs sometimes come up first in the results…) The first thing I had to do was open and parse the file. Once I had done that, I was able to use what I learned in the Kalb textbook to save header information in one list and all the other data in another.

From there, I iterated through each row in all\_rows and pulled out the metadata I cared about. I indexed into the row, knowing that I wanted the data at indices 3, 4, 7-10, 12, and 14-17. When I did that I was (somewhat mysteriously) left with a tuple, so I converted that to a list, and then appended each row’s list of metadata values to the initially empty list, all\_rows\_metadata. I also needed to clean up the header information so that it would be a simple list, so I converted it to a list and identified the one and only item (header[0]) that I wanted to work with. I then identified the same fields I wanted to display from the header.

After that, I iterated through each row in all\_rows\_metadata and for every item in that row, I printed the header field: metadata value. This information printed to the screen so that I could get a look at what I was doing. In order to print the header fields and metadata values alongside each other, I needed to add a counter to my second for loop in this section. This allowed me to use the count of each row to create the index into the header list. Each time I grabbed an item from in the row, I could use the current count to index into the header and bring what was initially two separate loops with two separate results lists into one.

After I succeeded in manipulating and printing this data, I wanted to make sure it was reusable, which required me to write it to a .csv. I searched through Python documentation and some Stack Overflow posts in order to get this going. Like the .txt files we’ve written in this course, .csv files require opening a new file and identifying whether you want to read the file, write to it, or some other combination of actions. I opened the new .csv and made sure that I could write to it. Then I wrote the header\_fields\_for\_access\_files to the first row, and then iterated through the all\_rows\_metadata list of lists and wrote one list, which I called line in the for loop, to a line in the .csv file.

At the end of this function, we’re left with a .csv file with the metadata fields and have the same information printed on the screen.

#### create\_access\_file()

This function creates a single access file that is identified in the function itself. In order for this function to work, the user needs to provide two arguments: the file path for the original file and the file path for the new access file. The folder that the access file will be created in must already exist in the current working directory.

In order to accomplish what I wanted to do here, I had to figure out how to move between folders in a directory. This led me to the os and shutil modules in the Python 3 documentation. I relied on shutil.copy2() in order to create the copy of the original file that would become the access file with metadata. I also used os.chdir() a few times in order to navigate within the directory. This felt akin to using cd in a terminal, so it wasn’t terribly complicated!

At the beginning of this function, I changed directories so I would be in the access\_files directory and then read the .txt file I would later write to. I then moved back to my final-project folder and opened the droid\_metadata\_for\_access\_files.csv so I could read it. I reused code in the droid\_data\_for\_access\_files function so I could pull together the header information and metadata I wanted to append to this access file. I then returned to the access-files folder, read the access file to file\_text, made sure I could also write to that file, and then pulled together access information, nicely formatted metadata, and the file text itself into a single string. With all this in place, I was then able to write everything to the access file in the access-files folder.

### Executing the two functions!

After defining these two fairly long functions, I was able to call the functions. I provided the two arguments in the script, rather than asking for user input since there are a lot of moving pieces in this program. I would expect/hope that someone who was using this program would be familiar enough with what’s going on here to provide the two arguments as input if that turns out to be more efficient than editing the script itself.

# Next steps:

The next step for this project would be to expand it on several fronts: figure out how to work with multiple document types (not just .txt files), automate the creation of multiple access files whose originals are saved in multiple locations, and incorporate any relevant digital preservation standards into the process as a whole. For instance, I don’t know that the way that I’ve copied files from one folder to a new folder is done according to best practices, or if that really matters when we’re talking about access files, rather than originals.