Workshop Notebook 6: Working with File Structures

Mandatory Disclosures

1. This is a whirlwind introduction, not exhaustive instruction
2. All images are by courtesy of the University Archives at Texas State University: http://www.univarchives.txstate.edu (http://www.univarchives.txstate.edu)
3. img_qc_workshop is licensed under the GNU General Public License v3.0, https://github.com/photosbyjeremy/img_qc_workshop/blob/master/LICENSE (https://github.com/photosbyjeremy/img_qc_workshop/blob/master/LICENSE)
4. *Any and all code provided is done so without any warranty or expectation of support by Jeremy Moore, Todd Peters, or Texas State University*

Output Directory Contents to CSV

Wittliff Archivist Lauren Goodley asked about an Excel file with all of the files in it. Both ALL of the files (recursive counting) plus on a directory-by-directory basis.

In [ ]:
```
start_directory = 'data/'
```

In [ ]:
```
# === MAGICK HAPPENS HERE

# import
import csv
from pathlib import Path

# get path
start_directory_path = Path(start_directory)

# print status
print(f'Getting list of files from {start_directory}')
print('')

# get sorted list of all things in start_directory
all_things_list = sorted(list(start_directory_path.glob('**/*')))

# get sorted list of directories from all_things_list
directories = sorted([x for x in all_things_list if x.is_dir()])

# get sorted files list from all_things_list
files = sorted([x for x in all_things_list if x.is_file()])

# create extensions list
start_directory_extensions_dictionary = {}

# loop through all files
for file in files:
    file_path = Path(file)

    # .DS_Store is treated as a filename and not an extension
    if file_path.name == '.DS_Store':
        if file_path.name in start_directory_extensions_dictionary:
            # increase value by 1
            start_directory_extensions_dictionary[file_path.name] += 1
        else:
            # add it to the dictionary with a value of 1
            start_directory_extensions_dictionary[file_path.name] = 1
    else:
        if file_path.suffix in start_directory_extensions_dictionary:
            # increase value by 1
            start_directory_extensions_dictionary[file_path.suffix] += 1
        else:
            # add it to the dictionary with a value of 1
            start_directory_extensions_dictionary[file_path.suffix] = 1


# set CSV filename based on start directory folder name
start_directory_name = start_directory_path.parts[-1]
csv_filename = 'files_in_' + start_directory_name + '.csv'
```

```
# open in CSV file WRITE mode (overwrites file if it exists)
csv_file = open(csv_filename, 'w', newline='')  # add newline='' to stop extra lines in Windows
# https://stackoverflow.com/questions/3348460/csv-file-written-with-python-has-blank-lines-between-each-ro
w

# update status
print(f'Creating CSV {csv_filename} . . . ')
print('')

# open in CSV file WRITE mode (overwrites file if it exists)
with csv_file:
    # WRITE START DIRECTORY NAME, FILE EXTENSION AND TOTAL OF EACH FILE TYPE

    # set headers
    fieldnames = ['START DIRECTORY', 'FILE EXTENSION', 'RECURSIVE TOTAL OF FILE TYPE']
    # create csv writer
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    # write header
    writer.writeheader()
    for extension in start_directory_extensions_dictionary:
        writer.writerow({'START DIRECTORY': start_directory_name, 'FILE EXTENSION': extension, 'RECURSIVE
 TOTAL OF FILE TYPE': start_directory_extensions_dictionary[extension]})


    # add start_directory as our first directory
    directories.insert(0, start_directory_path)

    # loop through directories
    for directory in directories:

        # print status
        print(f'Processing {str(directory)} . . . ')

        directory_name = Path(directory).parts[-1]

        # create extensions_dictionary
        extensions_dictionary = {}
        directory_list = sorted(directory.iterdir())
        files_list = [x for x in directory_list if x.is_file()]

        # create file_size_dictionary
        file_size_dictionary = {}

        # add files to extensions dictionary
        for file in files_list:
            file_path = Path(file)

            # .DS_Store is treated as a filename and not an extension
            if file_path.name == '.DS_Store':
                if file_path.name in extensions_dictionary:
                    # increase value by 1
                    extensions_dictionary[file_path.name] += 1
                else:
                    # add it to the dictionary with a value of 1
                    extensions_dictionary[file_path.name] = 1
            else:
                if file_path.suffix in extensions_dictionary:
                    # increase value by 1
                    extensions_dictionary[file_path.suffix] += 1
                else:
                    # add it to the dictionary with a value of 1
                    extensions_dictionary[file_path.suffix] = 1

            # get file size: https://docs.python.org/3/library/pathlib.html#pathlib.Path.stat
            # returns size in bytes so return size in MB as Lauren requested
            # bytes / 1024 = kilobytes / 1024 = megabytes
            file_size_dictionary[file_path.name] = file_path.stat().st_size / 1024 / 1024

        # WRITE DIRECTORY PATH, FILE EXTENSION AND TOTAL OF EACH FILE TYPE

        # set headers
        fieldnames = ['DIRECTORY PATH', 'FILE EXTENSION', 'TOTAL OF FILE TYPE']
        # create csv writer
        writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
        # write header
        writer.writeheader()
        for extension in extensions_dictionary:
            writer.writerow({'DIRECTORY PATH': directory, 'FILE EXTENSION': extension, 'TOTAL OF FILE TYP
E': extensions dictionary[extension]})
```

```
        # WRITE DIRECTORY NAME, FILENAME, AND EXTENSION

        # set headers
        fieldnames = ['DIRECTORY', 'FILENAME', 'EXTENSION', 'FILESIZE IN MB']
        # create csv writer
        writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
        # write header
        writer.writeheader()
        for file in files_list:
            writer.writerow({'DIRECTORY': directory_name, 'FILENAME': str(file.name), 'EXTENSION': str(fil
e.suffix), 'FILESIZE IN MB': str(file_size_dictionary[file.name])})

# set csv path
csv_path = Path('.', csv_filename)

# update status
print('')
print('*** FINAL STATUS ***')
if csv_path.exists():
    print(f'{csv_filename} exists in {Path.cwd()}')
else:
    print('CSV file does NOT exist!')
```

Verifying Directory Structures

University of Houston's Jerrell Jones asked about verifying files in their Carpenters system

In [3]:
```
master_directory = 'data/workshop-6/master_tif/'
modified_directory = 'data/workshop-6/modified_tif/'
jpg_directory = 'data/workshop-6/jpg/'
```

In [ ]:
```
# === Magick happens here
# === IMPORT ===
from pathlib import Path


# create a directories_list
directories_list = [master_directory, modified_directory, jpg_directory]

# create a directories_paths_list IF directory is actually a directory
directories_paths_list = [Path(directory) for directory in directories_list if Path(directory).is_dir()]#
 === IMPORT ===
from pathlib import Path

# create list of TIFFs in master_directory
master_tiff_list = sorted(list(directories_paths_list[0].glob('**/*.tif')))

print(f'Master TIFF directory: {master_directory}')
print(f'Number of TIFFs: {len(master_tiff_list)}')
print('')

# create list of files NOT in modified & jpg directories
files_not_in_modified_directory = []
files_not_in_jpg_directory = []

for tiff in master_tiff_list:
    # foo.name includes the extension
    tiff_name = tiff.name

    # foo.stem does NOT include the extension
    file_stem = tiff.stem
    jpg_name = file_stem + '.jpg'

    # modified directory path
    tiff_modified_path = directories_paths_list[1].joinpath(tiff_name)
    if not tiff_modified_path.is_file():
        files_not_in_modified_directory.append(tiff_name)

    # jpg directory path
    jpg_path = directories_paths_list[2].joinpath(jpg_name)
    if not jpg_path.is_file():
        files_not_in_jpg_directory.append(jpg_name)

if len(files_not_in_modified_directory) > 0:
    # print lists of missing files
```

```
    print(f'ERROR: Files NOT in Modified: {files_not_in_modified_directory}')
else:
    print('All TIFF files in Master in Modified')

if len(files_not_in_jpg_directory) > 0:
    # print lists of missing files
    print(f'ERROR: Files NOT in JPEG: {files_not_in_jpg_directory}')
else:
    print('All Master files in JPEG')


pdf_path_list = sorted(list(directories_paths_list[2].glob('**/*.pdf')))

if len(pdf_path_list) > 0:
    pdf_names_list = []
    for pdf_path in pdf_path_list:
        pdf_names_list.append(pdf_path.name)
    print(f'PDFs in JPG directory: {pdf_names_list}')
else:
    print(f'No PDF in JPG directory')
```

Create Test Directories & Files

*For example only, the code below does **NOT** work on Windows 10*

In [5]:
```
# === Create Test Directories & Files
# set initial and jerrell_jones directory paths
jj_dir = '/Users/Jeremy/img_qc_workshop/data/workshop-6/'

from pathlib import Path
initial_dir = str(Path.cwd())

# move into jerrell_jones directory
%cd {jj_dir}

# create item directories
!for i in {1..6}; do mkdir -p $(printf "item_%d" $i); done\

# list to verify they were created
%ls

# need random number of images in each item directory
import random

# create a random number of images in each item folder and 1 PDF
# loop through the number of items we have
for i in range(1, 7):
    # set stub name for item
    stub_name = 'item_' + str(i)

    # get directory path
    directory_path = Path('.', stub_name)

    # verify it's a directory
    if directory_path.is_dir():

        print(f' . . . starting {str(directory_path)}')

        # set random integer
        random_integer = random.randint(15, 61)
        print(f'random number: {random_integer}')

        for integer in range(1, random_integer):

            integer_zero_padded = str(integer).zfill(4)

            # set filenames based on suffixes for Carpenters
            preservation_master_name = stub_name + '_' + integer_zero_padded + '_pm.tif'
            modified_master_name = stub_name + '_' + integer_zero_padded + '_mm.tif'
            jpeg_name = stub_name + '_' + integer_zero_padded + '_ac.jpg'

            filenames_list = [preservation_master_name, modified_master_name, jpeg_name]

            filenames_paths_list = [directory_path.joinpath(filename) for filename in filenames_list]

            # create each file
            for path in filenames paths list:
```

```
                for path in filenames_paths_list:

                    # need path as string for BASH command
                    bash_path = str(path)

                    # create file at path
                    !touch {bash_path}

                    # if our file does not exist then print an ERROR
                    if not path.is_file():
                        print(f'ERROR: {bash_path} does NOT exist')

            # pdf name, path, bash_path
            pdf_name = stub_name + '_0001_ac.pdf'
            pdf_path = directory_path.joinpath(pdf_name)
            bash_pdf_path = str(pdf_path)

            # create pdf
            !touch {bash_pdf_path}

            # if our pdf does not exist then print an ERROR
            if not pdf_path.is_file():
                print(f'ERROR: {bash_pdf_path} does NOT exist')

    # use bash to list all items and pipe it to word count
    bash_string = !ls item_1/ | wc -w
    print(bash_string)

    # move back to initial directory
    %cd {initial_dir}
```

```
/Users/Jeremy/img_qc_workshop/data/workshop-6
image-file_002.jpg* item_3/             item_6/             modified_tif/
item_1/             item_4/             jpg/
item_2/             item_5/             master_tif/
 . . . starting item_1
random number: 43
 . . . starting item_2
random number: 58
 . . . starting item_3
random number: 60
 . . . starting item_4
random number: 24
 . . . starting item_5
random number: 20
 . . . starting item_6
random number: 33
['     127']
/Users/Jeremy/img_qc_workshop
```

Create DigitalObject class

The following code will work on Windows 10 and macOS

In [6]:
```
# create DigitalObject class
class DigitalObject(object):

    # initialize the DigitalObject
    def __init__(self, path_to_digital_object):
        self.path = Path(path_to_digital_object)
        self.name = self.path.name

        # list of all content in DigitalObject
        self.contents = sorted([x for x in self.path.iterdir()])

        # list of directories in DigitalObject
        self.dir_list = sorted([x for x in self.contents if x.is_dir()])
        self.dir_total = len(self.dir_list)

        # list of files in DigitalObject
        self.files_list = sorted([x for x in self.contents if x.is_file()])
        self.files_total = len(self.files_list)

        # list of preservation master TIFFs
        self.pm_list = sorted([x for x in self.files_list if str(x).endswith('_pm.tif')])
        self.pm_total = len(self.pm_list)

           # list of modified master TIFFs
```

```
            self.mm_list = sorted([x for x in self.files_list if str(x).endswith('_mm.tif')])
            self.mm_total = len(self.mm_list)

            # list of access JPEGs
            self.jpg_list = sorted([x for x in self.files_list if str(x).endswith('_ac.jpg')])
            self.jpg_total = len(self.jpg_list)

            # list of access PDFs
            self.pdf_list = sorted([x for x in self.files_list if str(x).endswith('_ac.pdf')])
            self.pdf_total = len(self.pdf_list)

            # list of expected files
            self.expected_files_list = self.pm_list + self.mm_list + self.jpg_list + self.pdf_list
            self.expected_files_list_total = len(self.expected_files_list)

            # list of extra files
            self.extra_files_list = sorted([x for x in self.files_list if x not in self.expected_files_list])
            self.extra_files_list_total = len(self.extra_files_list)
```

In [ ]:
```
# create a DigitalObject
item_1 = DigitalObject('data/workshop-6/item_1/')

print(f'DigitalObject name: {item_1.name}')
print(f'number of directories: {len(item_1.dir_list)}')
print(f'number of files: {len(item_1.files_list)}')
```

In [ ]:
```
# comparison
if item_1.extra_files_list_total > 0:
    print('ERROR: pm_total + mm_total + jpeg_total + pdf_total != files_total')
    if item_1.extra_files_list_total == 1:
        print('1 extra file')
    else:
        print(f'{item_1.extra_files_list_total} extra files')
    print(f'All Extra Files: {[x.name for x in item_1.extra_files_list]}')
else:
    print('pm_total + mm_total + jpeg_total + pdf_total = files_total')
```

In [ ]:
```
# list comprehension
print(f'All PDFs: {[x.name for x in item_1.pdf_list]}')
```

In [ ]:
```
# list slicing by index
print(f'First JPEG: {item_1.jpg_list[0].name}')
print(f'Last JPEG: {item_1.jpg_list[-1].name}')
```