Workshop Notebook 4: Batch Processing Images

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*

```
In [ ]:  image_directory = 'data/workshop-4/graduate_catalog-1966/'
```

```
In [ ]:  # importing
         from pathlib import Path
         from PIL import Image
         import matplotlib.pyplot as plt
         import img_qc.img_qc as img_qc
```

```
In [ ]:  # matplotlib options

         # magic that lets us plot directly in the notebook
         %matplotlib inline

         # parameters for matplotlib to increase our default figure size -- NOTE: figure sizes are in INCHES
         plt.rcParams["figure.figsize"] = (12,12)  # set as needed for your screen and eyes

         # on a high-dpi monitor this will increase the quality of plots on-screen
         %config InlineBackend.figure_format = 'retina'
```

Get Image Paths Using a Generator

```
In [ ]:  # get image_paths for TIFF images
         image_paths = Path(image_directory).glob('*.tif')

         image_paths
```

```
In [ ]:  # get image_paths list
         image_paths_list = list(image_paths)

         image_paths_list
```

```
In [ ]:  # try to get sorted image_paths list
         image_paths_list = sorted(image_paths)

         image_paths_list
```

Generator Objects Get Used Up

`image_paths = Path(image_directory).glob('*.tif')` created a generator that gets used up, which saves on memory and speeds up operations. So we need to call it again if we want to get a sorted image_paths_list

```
In [ ]:  # get image_paths for TIFF images
         image_paths = Path(image_directory).glob('*.tif')

         image_paths_list = sorted(image_paths)

         image_paths_list
```

Load First Image

We will use the first image to find the settings we need for all images

```
In [ ]: # open first image in our list
        image = Image.open(image_paths_list[0])  # list slicing

        # show image
        plt.imshow(image)
```

Crop

Crop into the image on the top and left to make sure the black is cropped out

```
In [ ]: # crop image
        image_cropped = image.crop(box=(15, 15, 3400, 5100))  # start at pixel (15, 15) in upper-left to (3450, 51
        00) in bottom-right

        # show image
        plt.imshow(image_cropped)
```

Expand Canvas

```
In [ ]: # sizes for expanding image canvas

        # get width_old & height
        (width_old, height_old) = image_cropped.size  # (width, height)

        # set width_new & height
        width_new = 600 * 6 # 600 ppi * 6 in.
        height_new = 600 * 9  # 600 ppi * 9 in.

        dimensions_dictionary = {'width_old': width_old,
                                 'height_old': height_old,
                                 'width_new': width_new,
                                 'height_new': height_new
                                }

        for dimension in dimensions_dictionary:
            print(f'{dimension}: {dimensions_dictionary[dimension]}')
```

```
In [ ]: # get border sizes

        # set width_border & height_border by subtracting old dimension from new and
        # divide by 2 to account for each side of the image
        width_border = (width_new - width_old) // 2  # integer division so we don't get part of a pixel with a flo
        at
        height_border = (height_new - height_old) // 2

        width_border, height_border
```

```
In [ ]: # expand image with ImageOps.expand

        # import Pillow's ImageOps
        from PIL import ImageOps

        # add white border to image
        image_with_border = ImageOps.expand(image_cropped, border=(width_border, height_border), fill='white')

        # show image
        plt.imshow(image_with_border)
```

```
In [ ]: # get image dimensions to verify it's 6 x 9 in. @ 600ppi
        image_with_border.size  # (width, height)
```

Expand Canvas, Take 2

ImageOps.expand doesn't allow us to adjust each side independently and we have an odd border size to add.

Can add 1 pixel to the border we're adding above, but then the sizes will be 1 pixel too much! We need a different way of expanding the border.

Let's create a new image the size we want and paste our image in the center! (Or 1 pixel off from center.)

```
In [ ]: # create new bitonal image

        image_new = Image.new(mode='1', size=(width_new, height_new), color='white')

        # show image
        plt.imshow(image_new)
```

```
In [ ]: # paste image_cropped into the center of image_new
        image_new.paste(image_cropped, box=(width_border, height_border))  # box = 2-tuple for upper-left corner

        # show image
        plt.imshow(image_new)
```

Save Image

```
In [ ]: # save image
        image_new.save('data/workshop-4/test.tif', compression='group4', dpi=(600., 600.))  # set dpi with floats,
         ints fail

        # open image
        test_image = Image.open('data/workshop-4/test.tif')

        # get info on image
        print(test_image.mode)
        print(test_image.info)
        print(f'width: {test_image.size[0]} pixels')  # (width, height)
        print(f'height: {test_image.size[1]} pixels')
```

Batch Process all Image Paths

```
In [ ]: # crop, expand, and save all images

        # set width_new & height
        # NOTE: already set above, but including here to remember what we set
        width_new = 600 * 6 # 600 ppi * 6 in.
        height_new = 600 * 9  # 600 ppi * 9 in.

        def crop_and_expand_bitonal_images(image_paths_list, width_new, height_new, crop_box):
            for image_path in image_paths_list:

                # open image
                image = Image.open(image_path)

                # crop image
                image_cropped = image.crop(box=crop_box)

                # get width_old & height_old
                (width_old, height_old) = image_cropped.size

                # get border sizes
                # set width_border & height_border by subtracting old dimension from new and
                # divide by 2 to account for each side of the image
                width_border = (width_new - width_old) // 2  # integer division so we don't get part of a pixel wi
        th a float
                height_border = (height_new - height_old) // 2

                # create new bitonal image
                image_new = Image.new(mode='1', size=(width_new, height_new), color='white')

                # paste image_cropped into the center of image_new
                image_new.paste(image_cropped, box=(width_border, height_border))  # box = 2-tuple for upper-left
         corner

                # get image name
                image_name = image_path.name

                # set output path
                output_path = Path('data/workshop-4/output/').joinpath(image_name)

                # save image
                image_new.save(output_path, compression='group4', dpi=(600., 600.))  # set dpi with floats, ints f
        ail

                # create a new MatPlotLib figure so we can plot each image
                plt.figure()
                # show image
                plt.imshow(image_new)

        crop_and_expand_bitonal_images(image_paths_list, width_new, height_new, crop_box=(15, 15, 3400, 5100))
```

Rotate every other image, save as Group4 compressed TIFF, resize, & save as JPEG

Every other image (odd numbered images) needs to be rotated 180 degrees.

Save as Group4 compressed TIFF image.

Resize to 900 pixel width.

Save as JPEG: http://pillow.readthedocs.io/en/5.1.x/handbook/image-file-formats.html#jpeg

In [ ]:

```
# crop, expand, and save all images as Group4 compressed TIFFs and 900 pixel width JPEGs
# rotate every other image

# set width_new & height
# NOTE: already set above, but including here to remember what we set
width_new = 600 * 6 # 600 ppi * 6 in.
height_new = 600 * 9  # 600 ppi * 9 in.

def crop_expand_and_rotate_bitonal_images(image_paths_list, width_new, height_new, crop_box):
    for image_path in image_paths_list:

        # open image
        image = Image.open(image_path)

        # crop image
        image_cropped = image.crop(box=crop_box)

        # get width_old & height_old
        (width_old, height_old) = image_cropped.size

        # get border sizes
        # set width_border & height_border by subtracting old dimension from new and
        # divide by 2 to account for each side of the image
        width_border = (width_new - width_old) // 2  # integer division so we don't get part of a pixel wi
th a float
        height_border = (height_new - height_old) // 2

        # create new bitonal image
        image_new = Image.new(mode='1', size=(width_new, height_new), color=255)

        # paste image_cropped into the center of image_new
        image_new.paste(image_cropped, box=(width_border, height_border))  # box = 2-tuple for upper-left
 corner

        # get image name
        image_name = image_path.name  # includes extension

        # get image stem
        image_stem = image_path.stem  # does NOT include extension

        # get last character from image_stem
        last_character = image_stem[-1]  # list slicing

        # if last_character is even
        if int(last_character) % 2 == 0:  # set last_character as integer for modulus operation

            # rotate 180 degrees -- rotations divisible by 90 degrees do not require interpolation
            image_new = image_new.rotate(180)

        # set output path
        output_path = Path('data/workshop-4/output/').joinpath(image_name)

        # save image with group4 compression and 600 dpi
        image_new.save(output_path, compression='group4', dpi=(600., 600.))  # set dpi with floats, ints f
ail

        # set jpeg name
        jpeg_name = image_stem + '.jpg'
        # set jpeg output path
        jpeg_output_path = Path('data/workshop-4/output/').joinpath(jpeg_name)

        # resize image_new to 1500 pixel height
        image_resized = img_qc.get_image_resized_pillow(image_new, width=900)

        image_resized.save(jpeg_output_path, quality=80, optimize=True)  # default quality is 75

        # create a new MatPlotLib figure so we can plot each image
        plt.figure()

        # show image
        plt.imshow(image_resized)

crop_expand_and_rotate_bitonal_images(image_paths_lists, width_new, height_new, crop_box=(15, 15, 3400,
5100))
```