 **Sckikit-image image cheat sheet:**

1. Why use skimage (as oppposed to something else)?

A numfocus project that aims to be the Python FOSS standard reference for image processing.

1. How to use skimage (basic starter syntax and code)?

Step 1: our backbone imports for images:

import numpy as np

from matplotlib import pyplot as plt

# from PIL import Image

from skimage import data, io, color

Step 2: Basic syntax for basic operations using skimage:

Read in an image :

other\_image = skimage.io.imread(pict1) (skimage)

Split image to channels:

Red, green, blue = other\_image[:,:,0], other\_image[:,:,1], other\_image[:,:,2]

NB: opencv and other libraries may put channels in a different order

Nota Also Importante: You can essentially handle the skimage like a numpy array now in many ways I.e. otherimage.shape, otherimage.max, other\_image.mean() are all valid (the library uses numpy arrays as image objects)...and you can exploit numpy (especially numpy.lookfor)to look im skimage e.g. try np.lookfor(‘boundaries’, ‘skimage’) inside your notebook

Dealing with datatypes in skimage:

Function names: Description

img\_as\_float(): Convert to floating point (integer types become 64-bit floats)

img\_as\_ubyte(): Convert to 8-bit uint.

img\_as\_uint(): Convert to 16-bit uint.

img\_as\_int(): Convert to 16-bit int.

 

Do inversions:

from skimage import util

inverted\_img = util.invert(res\_h)

Look at historgrams:

*remember fundamentally you are dealing with a numpy object and If not provided, range is simply (image.min(), image.max())*

histogram, bin\_edges = np.histogram(image, bins=256, range=(0, 1))

Changing the size of your image:

*Rescale* resizes an image by a scaling factor you give along each axis.

*Resize* is similar but uses an output image shape instead of scaling factors.

*Downscale* down-samples an n-dimensional image by integer factors using the local mean on the elements of each block of the size given as a parameter to the function.

Aliasing: In terms of imaging this is an artifact that introduces a differing image from the original and is rarely desirable. This can be avoided with smoothing, other algorithms or inside of rescale and resize functions.

img\_rescaled= rescale(=image, 0.25, anti\_aliasing=True)  
 Img\_resized= resize(image, (image.shape[0] // 4, image.shape[1] // 4), anti\_aliasing=**True**)  
 img\_downscaled= downscale\_loacal\_mean( image, (4, 3))