EN.540.635 "Software Carpentry"

The Python Imaging Library

When programming, one does not expect to have to interface with a computer at the lowest level. This inherent assumption is commonly overlooked, and beginning programmers often forget that a wide assortment of features have already been developed and made accessible to be expanded upon. These *functions*, *objects*, and more are written into special files called *libraries* (either *static* or *dynamic*), and can be *linked* to when compiling code. By default, many standard features are made available in programming languages (such as the print statement); however, many more are easily accessible via an import or include statement.

Regarding Python specifically, as we do not compile our code, we will be using a special *import* statement to include pre-written objects. Many of these are actually written and compiled in C++, and made into a *shared library* that we *dynamically* link to. In linux and mac, these libraries are commonly saved as .so files, while in windows these are .dll files. Alternatively, we may also import another python file (.py) that has functions/objects to simplify our lives. Either way, this has already been done for us! Let's take the case of processing an image (say a .png file) and saving a new one.

When tasked with such a complicated assignment, a programmer's first thought should be "Has this already been done before?". Simply put, as Isaac Newton said, "If I have seen further it is by standing on the shoulders of Giants". There is no need to re-invent the wheel, nor is there a need to have superfluous code where *bugs* may creep into. In the case of image analysis, a simple google search of "python open image file" gives the first link to stackoverflow (here) where the first answer introduces the standard PIL module that many use in image analysis. The second google link shows the standard *open* function in python (where one would read in *bytes* of a file), and the third once again discusses PIL. At this point, one should stop and consider "what is PIL, and can I use it"? The answer is simple: yes. PIL stands for the Python Imaging Library, and should be available by default with most Python installations.

Using PIL

When working with an unknown module, the first course of action is to seek out its documentation (here). From this, we can write up a simple demonstration of how to use PIL:

```
# This will import, from the PIL module, the Image object/module/function
from PIL import Image

# This will load a file into a variable as an Image object
img = Image.open("cats.png")

# This will get us the number of pixels of our image
WIDTH, HEIGHT = img.size

# We can get pixel information by simply requesting it
px1 = img.getpixel((10, 23))
print("Pixel at (10, 23) was RGB %s" % str(px1))
# We will then overwrite that pixel with gray
img.putpixel((10, 23), (0, 0, 0))

# We can save a file
img.save("cats2.png")

# We can also make new images from scratch!
SIZE_W, SIZE_H = 800, 600
img2 = Image.new("RGB", (SIZE_W, SIZE_H), color=(0, 0, 0))
img2.save("all_black.png")
```

The above example does not appear in the documentation, but instead we have gathered information we think pertinent and written a simple script to illustrate how to use PIL. Many functions exist within PIL; however, unless we expect to need to become experts at PIL specifically, we can simply get away with the basics.