Setting up your environment

All of your labs will be graded on `burrow.luddy.indiana.edu`, so you should verify that your work compiles, runs, and gives the expected output there. Doing the programming and debugging on your own machine is usually more convenient, though, so you may want to get your code working on your own machine and then test it on the server once it is roughly complete. If you're new to burrow, you'll need to use SSH and SCP tool like the ones listed below (these are only suggestions). An SSH client logs you into machines with command line access, and SCP clients allow you to move files between your host machine and the server.

	SSH Client	SCP Client
Linux	SSH on Terminal	scp
Windows	PuTTY	WinSCP
Mac	SSH on Terminal	CyberDuck

While we won't put a restriction on what language you use for projects, we encourage using Python 3, and will only be able to give code-level support for it.

Lastly, you may use your own computers to develop and test your code, but please make sure that the code runs on the IU servers prior to submission.

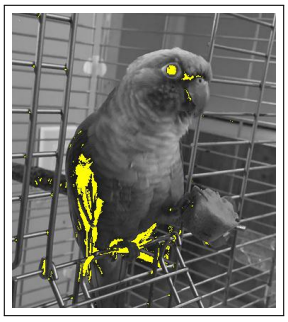
Getting up to speed with `Pillow`

Pillow is perhaps the most popular image processing library for Python Full documentation is available at `https://pillow.readthedocs.io/en/stable/`. Burrow already has this library installed, and you can freely install it on your own computer. Below we show some sample code that loads an image, checks some of its properties, and does some image manipulations and then saves the results.

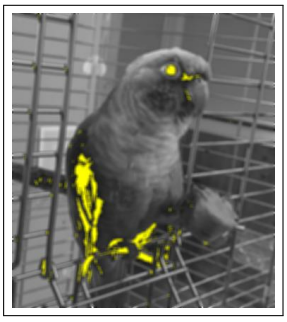
```
1  #Import the Image and ImageFilter classes from PIL (Pillow)
2  from PIL import Image
3  from PIL import ImageFilter
4  import sys
5  import random
6
7  if __name__ == '__main__':
8      # Load an image
9      im = Image.open(sys.argv[1])
10
11     # Check its width, height, and number of color channels
12     print("Image is %s pixels wide." % im.width)
13     print("Image is %s pixels high." % im.height)
14     print("Image mode is %s." % im.mode)
15
16     # Pixels are accessed via an (X,Y) tuple.
17     # The coordinate system starts at (0,0) in the upper left-hand corner,
18     # and increases moving right (first coordinate) and down (second coordinate).
19     # So it's a (col, row) indexing system, not (row, col) like we're used to
20     # when dealing with matrices or 2d arrays.
21     print("Pixel value at (10,10) is %s" % str(im.getpixel((10,10))))
22
23     # Pixels can be modified by specifying the coordinate and RGB value
24     # (255, 0, 0) is a pure red pixel.
25     im.putpixel((10,10), (255, 0, 0))
26     print("New pixel value is %s" % str(im.getpixel((10,10))))
27
28     # Let's create a grayscale version of the image:
```

```
    # the "L" means there's only a single channel, "Lightness"
    gray_im = im.convert("L")

    # Create a new blank color image the same size as the input
    color_im = Image.new("RGB", (im.width, im.height), color=0)
    gray_im.save("gray.png")

    # Highlights any very dark areas with yellow.
    for x in range(im.width):
        for y in range(im.height):
            p = gray_im.getpixel((x,y))
            if p < 5:
                (R,G,B) = (255,255,0)
                color_im.putpixel((x,y), (R,G,B))
            else:
                color_im.putpixel((x,y), (p,p,p))

    # Show the image. We're commenting this out because it won't work on the Linux
    # server (unless you set up an X Window server or remote desktop) and may not
    # work by default on your local machine. But you may want to try uncommenting it,
    # as seeing results in real-time can be very useful for debugging!
    # color_im.show()

    # Save the image
    color_im.save("output.png")

    # This uses Pillow's code to create a 5x5 mean filter and apply it to
    # our image. In the lab, you'll need to write your own convolution code (using
    # "for" loops, but you can use Pillow's code to check that your answer is correct.
    # Since the input is a color image, Pillow applies the filter to each
    # of the three color planes (R, G, and B) independently.
    box = [1]*25
    result = color_im.filter(ImageFilter.Kernel((5,5),box,sum(box)))
    # result.show()
    result.save("output2.png")
```

Input



Output



Output (filtered)

You can run the program like this:

```
python3 lab1.py example.jpg
```

This example, while simple, demonstrates almost everything you'll need to do this lab, although you should feel free to check the Pillow documentation for more.

This lab

We want to get you comfortable in the programming environment and some experience in Pillow-based image processing. In doing so, you will be applying linear filters to some images and see how they turn out — i.e.,

you'll be applying convolutions.

Exercise: Linear Filters

0	0	0
0	1	0
0	0	0

(a) Identity

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

(b) Box blur

0	0	0
-1	0	1
0	0	0

(c) Horizontal derivative

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

(d) Approximated Gaussian

$([1+\alpha]a - d)$

(e) Sharpening

$(c * d)$

(f) Derivative of Gaussian

Figure 1: Illustration of six convolution kernels. Note that in the sharpening filter in (e), a and d refer to the filters in (a) and (d), respectively, and α is a constant you'll need to adjust. Similarly, c and d refer to filters in (c) and (d).

Write a program that loads a color image and applies the kernels shown in Figure 1 to the image and saves the results (so 1 input image, 6 output images). Please name the output files as a.png, b.png, c.png, d.png, e.png, and f.png, corresponding to the letters in Figure 1.

Please write your own convolution code. To make sure that your code is working, you may compare your results against those generated by using Pillow's `ImageFilter` class. An example of using ImageFilter is shown on line 61 in the code above. Before performing the convolution, you need to make some choices regarding how to handle the edges of the original image. This is entirely up to you! As we discussed in the lecture, you can crop, mirror, etc.

What to turn in

Please submit (via Canvas) a zip file containing your source code, a sample input image (you can use ours or your own), and the six output images that you obtained for that sample image. Please make sure that your code runs on the university's linux server prior to submission.

Again, if you have any questions, please ask us on Piazza!