Setting up your environment

All of your projects will be graded on `burrow.cs.indiana.edu`, so you should verify that your work compiles, runs, and gives the expected output there. You should already have an account and many of you will have had similar course requirements in the past. 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.

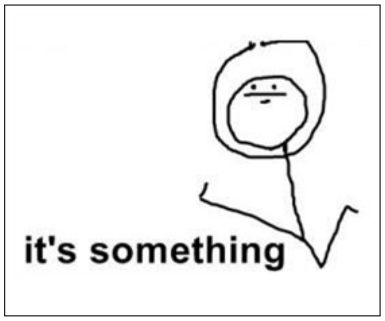
Lastly, you may use your own computers to develop and test your code, but please make sure that the code runs on the IU servers prior to submission.

Getting up to speed with `Pillow`

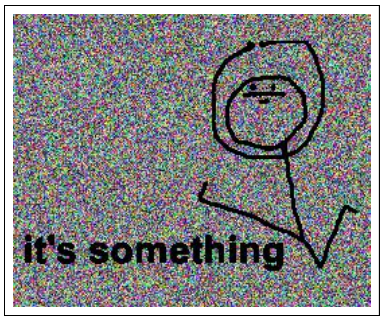
Python does not have native support for image processing so we encourage students to make use of the **Pillow Library**, supported at `https://pillow.readthedocs.io/en/stable/`. The IU machines already have this library. If you wish to install it into your own system, simply do `pip install Pillow`. Below we show some sample code that loads a grayscale image, checks some of its properties, and creates a new color image based on its pixel values.

```
#Import the Image and ImageFilter classes from PIL (Pillow)
from PIL import Image
from PIL import ImageFilter
import random

if __name__ == '__main__':
    #Load an image (this one happens to be grayscale)
    im = Image.open("example.jpg")

    #Check its width, height, and number of color channels
    print("Image is %s pixels wide." % im.width)
    print("Image is %s pixels high." % im.height)
    print("Image mode is %s." % im.mode)

    #pixels are accessed via a (X,Y) tuple
    print("Pixel value is %s" % im.getpixel((10,10)))
    #pixels can be modified by specifying the coordinate and RGB value
    im.putpixel((10,10), 20)
    print("New pixel value is %s" % im.getpixel((10,10)))

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

    #Loops over the new color image and fills in any area that was white in the
    #first grayscale image we loaded with random colors!
    for x in range(im.width):
        for y in range(im.height):

            if im.getpixel((x,y)) > 200:
                R = random.randint(0,255)
                G = random.randint(0,255)
                B = random.randint(0,255)
                color_im.putpixel((x,y), (R,G,B))
            else:
                color_im.putpixel((x,y), (0,0,0))

    #Show the image
    color_im.show()

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

    #Using Pillow's code to create a convolution kernel and apply it to our color image
    #Here, we are applying the box blur, where a kernel of size 3x3 is filled with 1
    #and the result is divided by 9
    result = color_im.filter(ImageFilter.Kernel((3,3),[1,1,1,1,1,1,1,1,1],9))
    result.show()
```

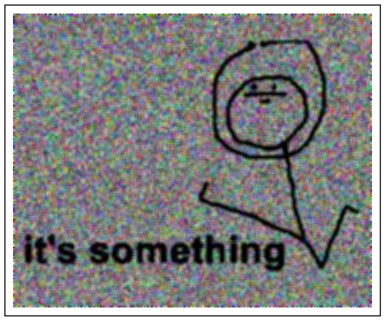
Running the code with the input below produces the corresponding outputs.



Input



Output



Output (filtered)

This example should provide you with everything you will need to build on for the projects and while it's not the most impressive thing in the world ... **it's something**.

This week's Lab

For this week, we want to get everyone comfortable in the programming environment and to get you used to manipulating pixel values of images. In doing so, you will be applying linear filters to some images and see how they turn out (i.e., you will be doing convolutions).

Exercise: Linear Filters

To start things off, we want you to write a program that loads a color image and applies some of the linear filters that were introduced in the lectures. In particular, your program should take 1 color image as input and perform convolutions to apply the following kernels to the image and save the results (so 1 input image, 4 output images):

0	0	0
0	1	0
0	0	0

(a) Identity

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

(b) Box blur

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

(c) Approximated Gaussian

$([1+\alpha]e - H)$

(d) Sharpening

Note that for the sharpening filter, you can use the Identity filter from (a) for e and use the approximated Gaussian from (c) for H. This just leaves you with the constant α to play around with.

You may or may not see a difference between the box blur and the Gaussian depending on your input image. Feel free to calculate the difference between the output of these two filters and visualize it (this step is optional).

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! You can crop, mirror, etc.

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 46 in the code above.

What to turn in

Please submit a zip file containing your source code and the image(s) that you used for this lab. Please make sure that your code runs on the university's linux server prior to submission. If you used any external libraries other than Pillow, please include a README file explaining why you did so.

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