**Outline**

- Quiz Results

Programming

- The story of developing blur()
    - lots of errors
- Processing images in a directory
- Presentations

Cool media art pieces

Prof. Angela Chang

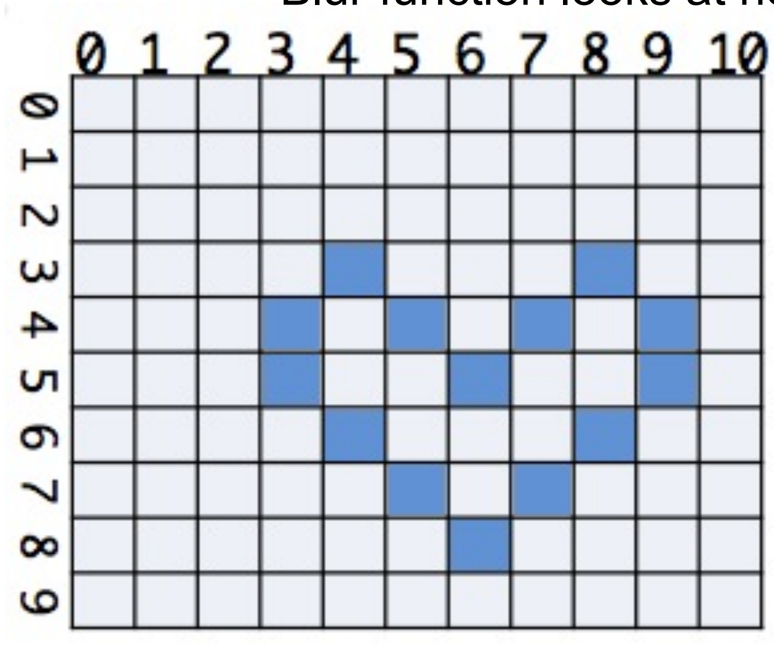
Lecture 13: Image Processing & Poster Presentation Challenge

Fall 2017. Oct. 23

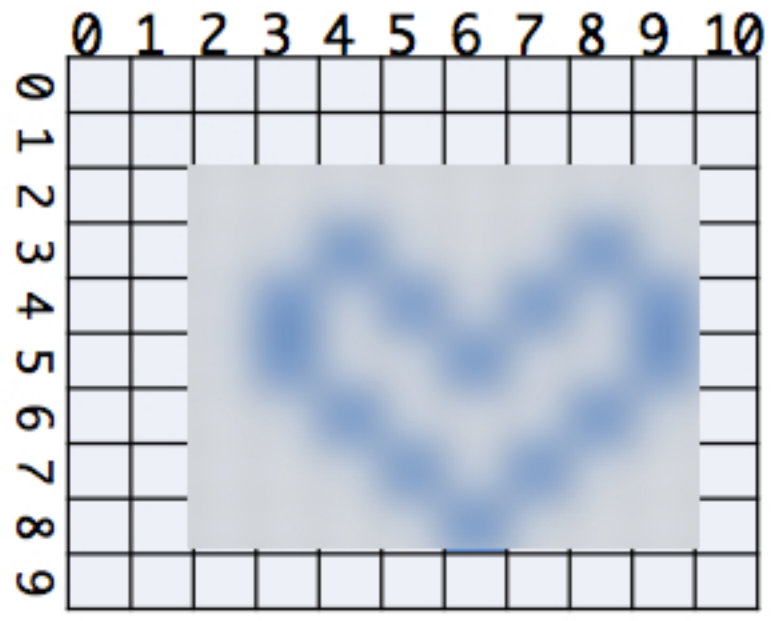
# CODE, CULTURE, AND PRACTICE

# Blur

Blur function looks at neighboring pixels  RGB values.



Sharp

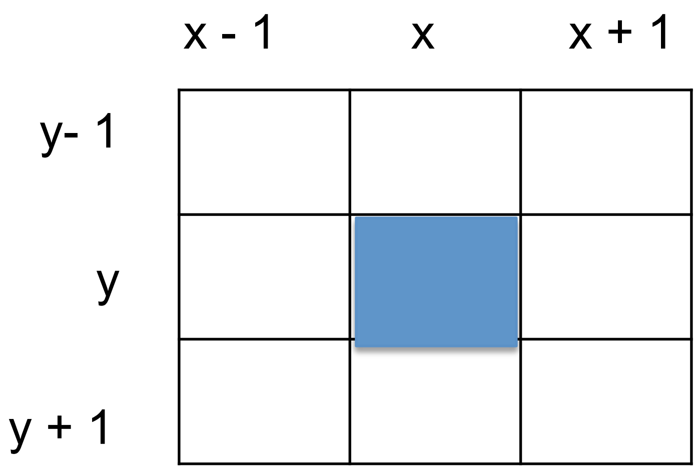


Blurry

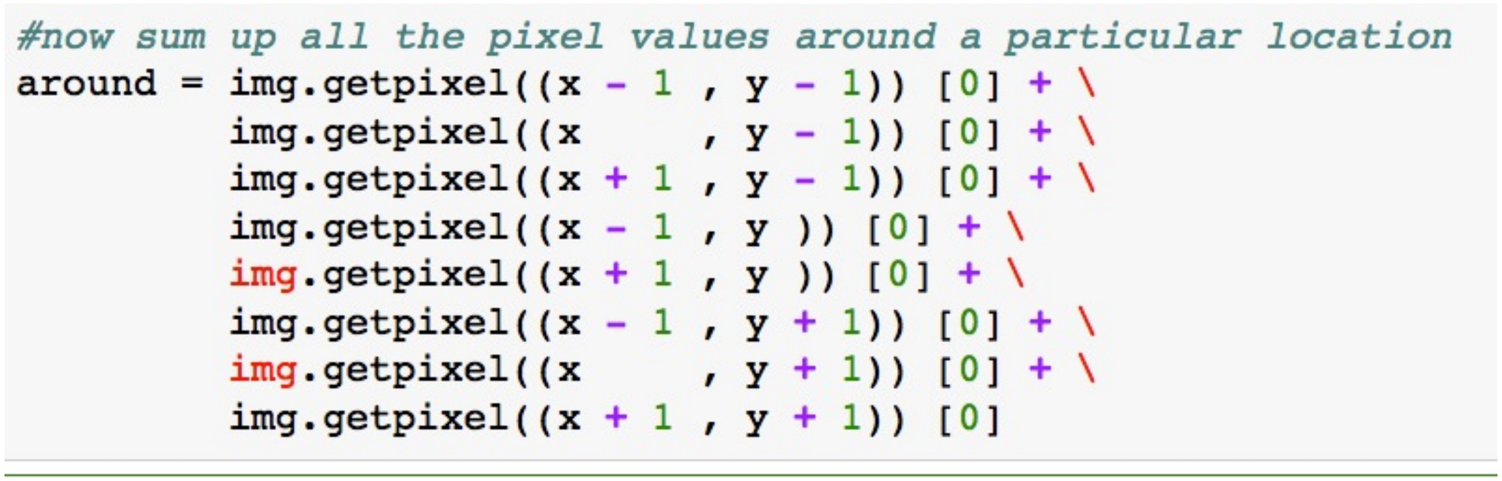
Blur should make neighboring pixels more similar.

# Look at the neighbors

Take one pixel, look at the colors of the neighboring pixels.

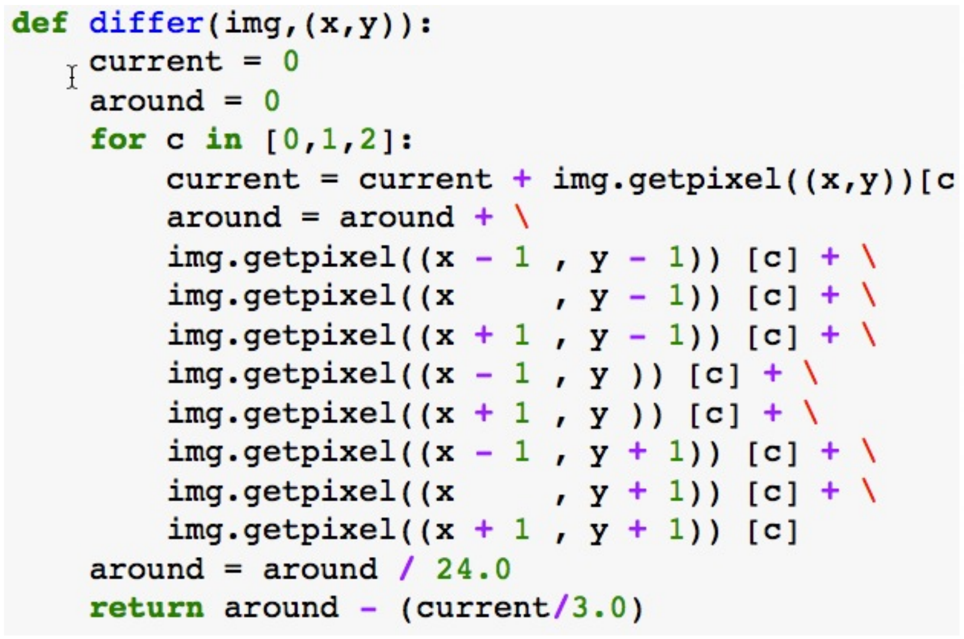


Average out the neighboring pixel values, and shift the center pixel closer to the average.

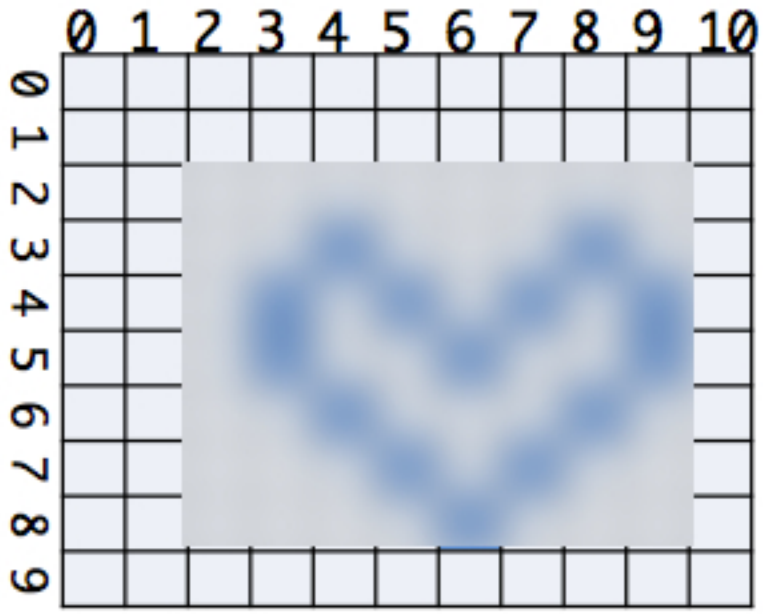


```
#now sum up all the pixel values around a particular location
around = img.getpixel((x - 1 , y - 1)) [0] + \
         img.getpixel((x     , y - 1)) [0] + \
         img.getpixel((x + 1 , y - 1)) [0] + \
         img.getpixel((x - 1 , y )) [0] + \
         img.getpixel((x + 1 , y )) [0] + \
         img.getpixel((x - 1 , y + 1)) [0] + \
         img.getpixel((x     , y + 1)) [0] + \
         img.getpixel((x + 1 , y + 1)) [0]
```

# Now add color and difference

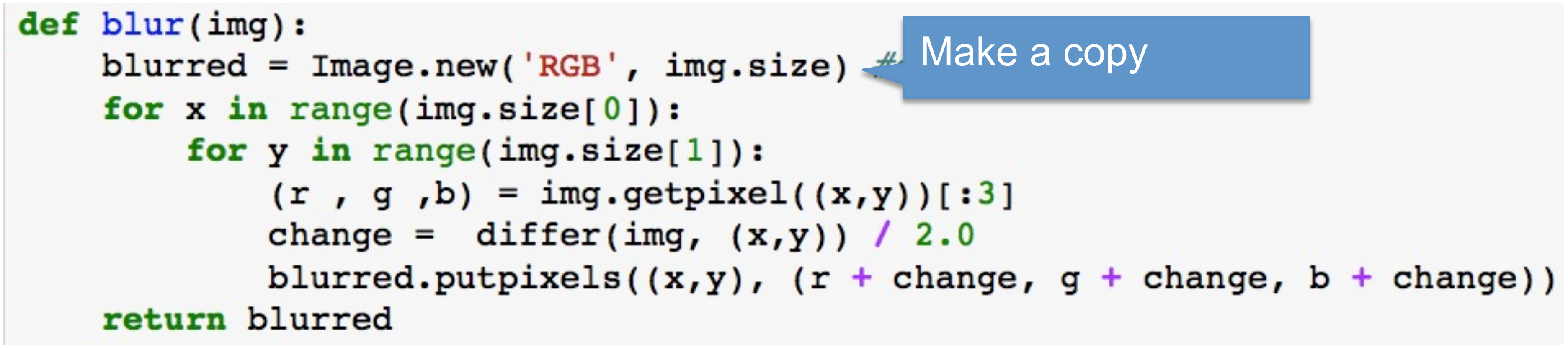


```
def differ(img,(x,y)):
    current = 0
    around = 0
    for c in [0,1,2]:
        current = current + img.getpixel((x,y))[c
        around = around + \
        img.getpixel((x - 1 , y - 1)) [c] + \
        img.getpixel((x     , y - 1)) [c] + \
        img.getpixel((x + 1 , y - 1)) [c] + \
        img.getpixel((x - 1 , y )) [c] + \
        img.getpixel((x + 1 , y )) [c] + \
        img.getpixel((x - 1 , y + 1)) [c] + \
        img.getpixel((x     , y + 1)) [c] + \
        img.getpixel((x + 1 , y + 1)) [c]
    around = around / 24.0
    return around - (current/3.0)
```



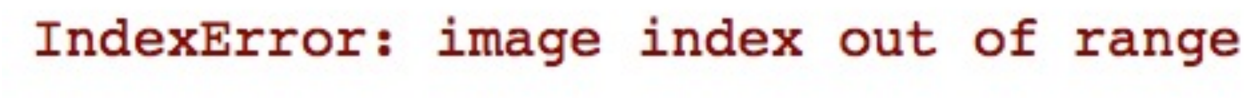
In continuous, similar-looking regions, result should be near zero. In “sharp” regions, it will have a larger magnitude, and may be negative (if the pixel is lighter than average) or positive (if darker). To blur the image a bit, we can shift each color by half of what differ returns..

# Blur



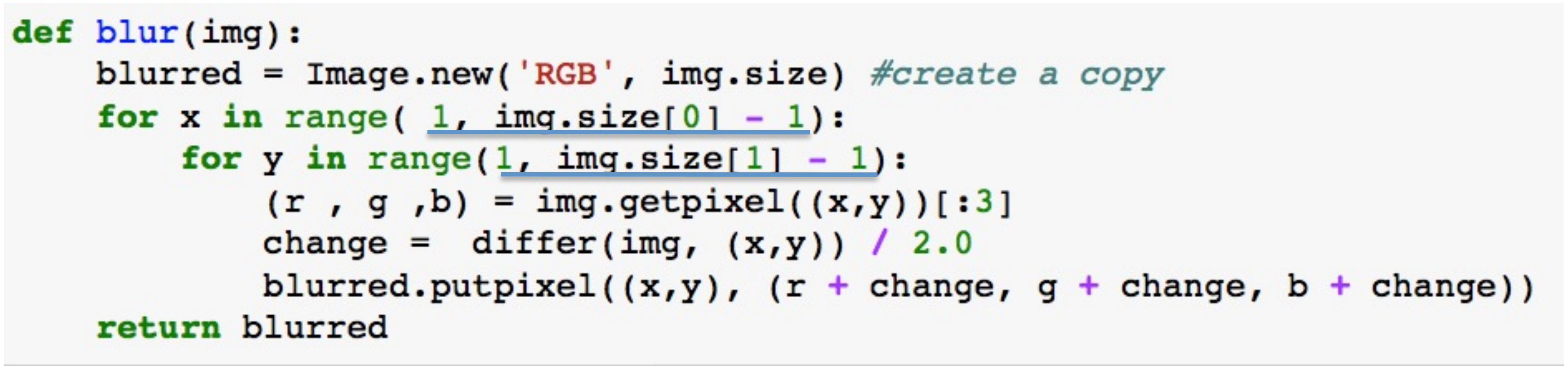
```python
def blur(img):
    blurred = Image.new('RGB', img.size) #
    for x in range(img.size[0]):
        for y in range(img.size[1]):
            (r , g ,b) = img.getpixel((x,y))[:3]
            change =  differ(img, (x,y)) / 2.0
            blurred.putpixels((x,y), (r + change, g + change, b + change))
    return blurred
```

Run it, and ….

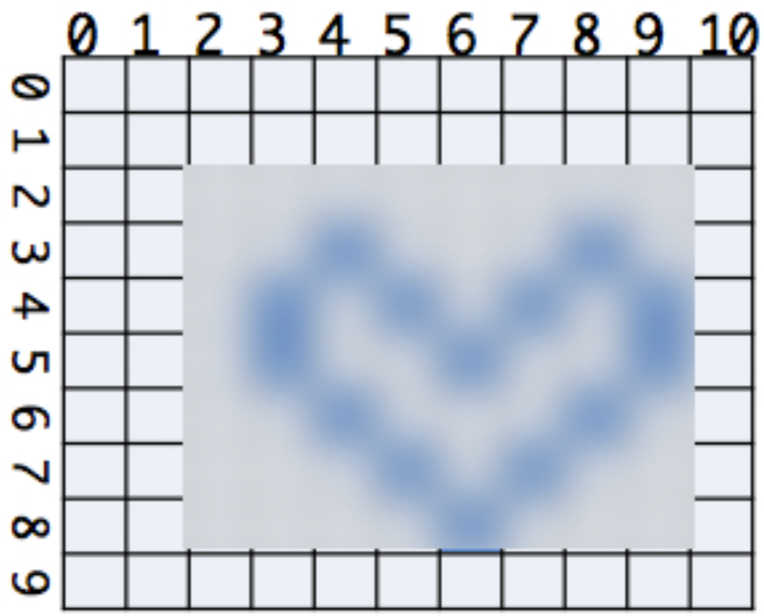


```
IndexError: image index out of range
```

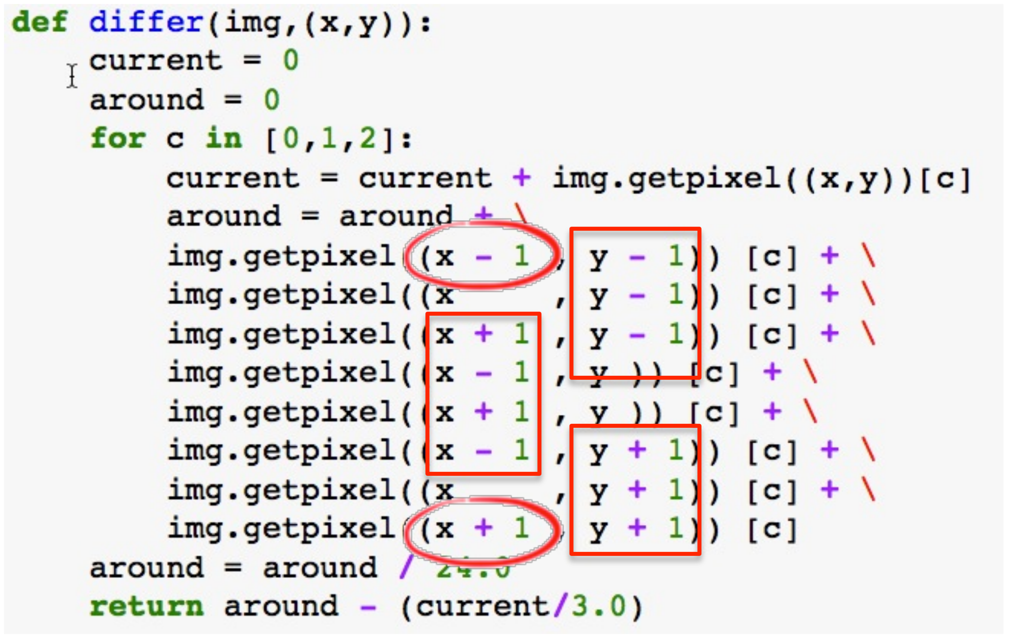
# Removing index error



```python
def blur(img):
    blurred = Image.new('RGB', img.size) #create a copy
    for x in range( 1, img.size[0] - 1):
        for y in range(1, img.size[1] - 1):
            (r , g ,b) = img.getpixel((x,y))[:3]
            change =  differ(img, (x,y)) / 2.0
            blurred.putpixel((x,y), (r + change, g + change, b + change))
    return blurred
```



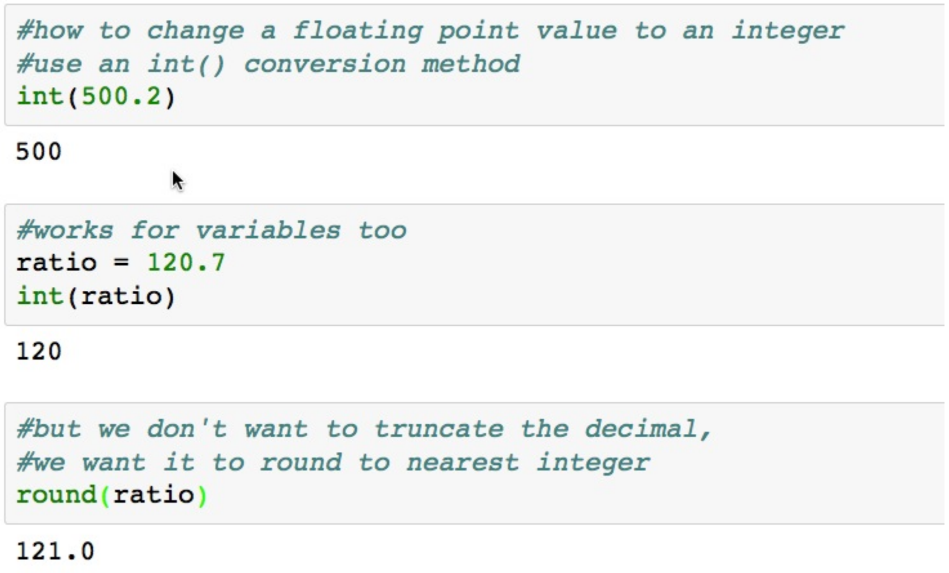
Pixels with negative coordinates don't exist….



```python
def differ(img,(x,y)):
    current = 0
    around = 0
    for c in [0,1,2]:
        current = current + img.getpixel((x,y))[c]
        around = around + \
        img.getpixel((x - 1 , y - 1)) [c] + \
        img.getpixel((x     , y - 1)) [c] + \
        img.getpixel((x + 1 , y - 1)) [c] + \
        img.getpixel((x - 1 , y )) [c] + \
        img.getpixel((x + 1 , y )) [c] + \
        img.getpixel((x - 1 , y + 1)) [c] + \
        img.getpixel((x     , y + 1)) [c] + \
        img.getpixel((x + 1 , y + 1)) [c]
    around = around / 24.0
    return around - (current/3.0)
```

# Removing type error

Pixels have to be integers….



```
#how to change a floating point value to an integer
#use an int() conversion method
int(500.2)
```

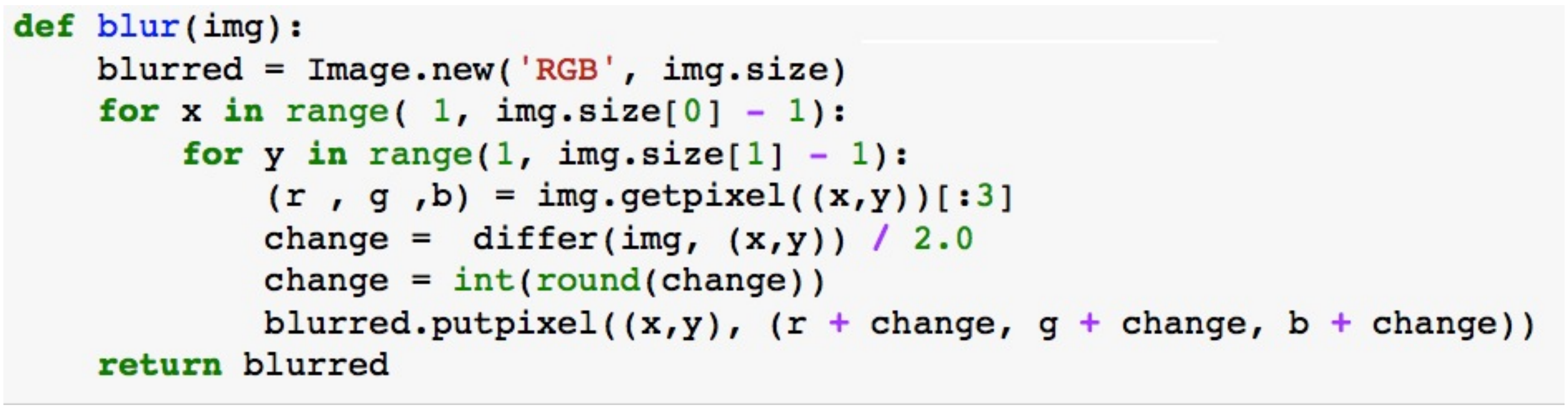
```
500
```

```
#works for variables too
ratio = 120.7
int(ratio)
```

```
120
```

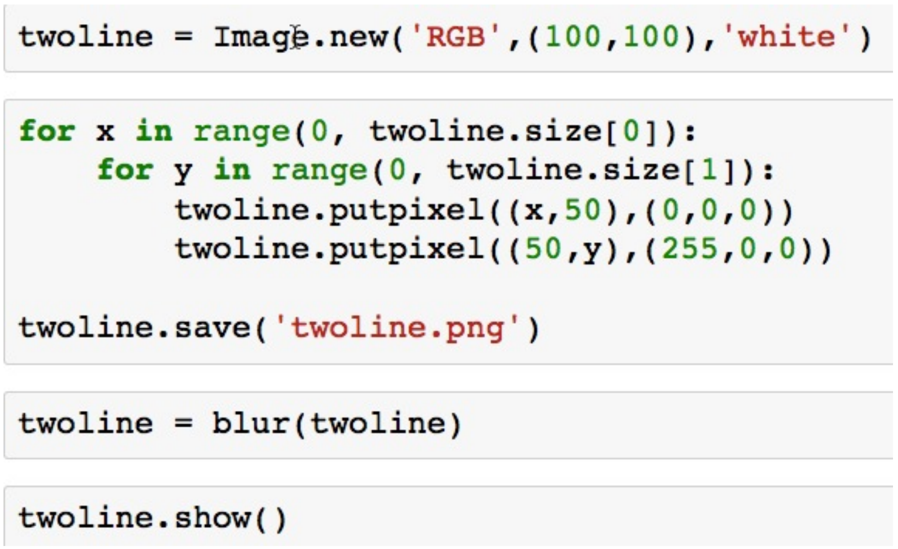
```
#but we don't want to truncate the decimal,
#we want it to round to nearest integer
round(ratio)
```

```
121.0
```



```
def blur(img):
    blurred = Image.new('RGB', img.size)
    for x in range( 1, img.size[0] - 1):
        for y in range(1, img.size[1] - 1):
            (r , g ,b) = img.getpixel((x,y))[:3]
            change =  differ(img, (x,y)) / 2.0
            change = int(round(change))
            blurred.putpixel((x,y), (r + change, g + change, b + change))
    return blurred
```

# Don’t take my word for it!



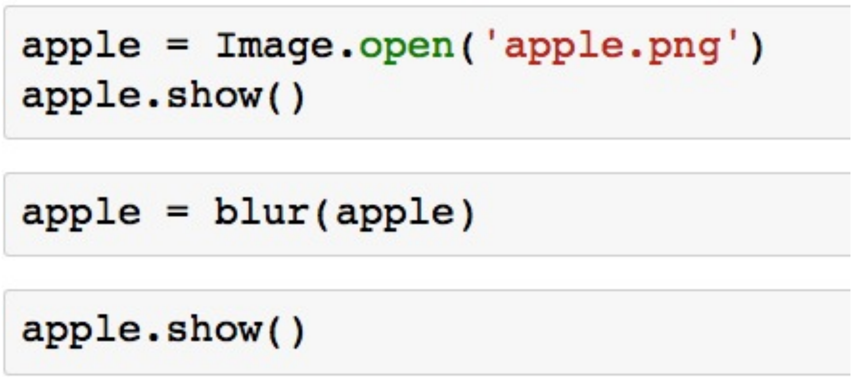
```
twoline = Image.new('RGB',(100,100),'white')
```

```
for x in range(0, twoline.size[0]):
    for y in range(0, twoline.size[1]):
        twoline.putpixel((x,50),(0,0,0))
        twoline.putpixel((50,y),(255,0,0))

twoline.save('twoline.png')
```

```
twoline = blur(twoline)
```

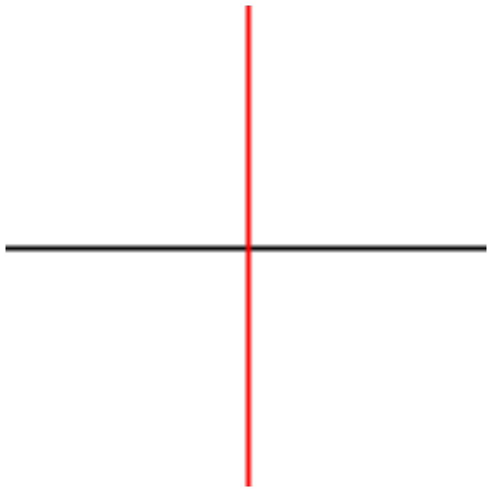
```
twoline.show()
```

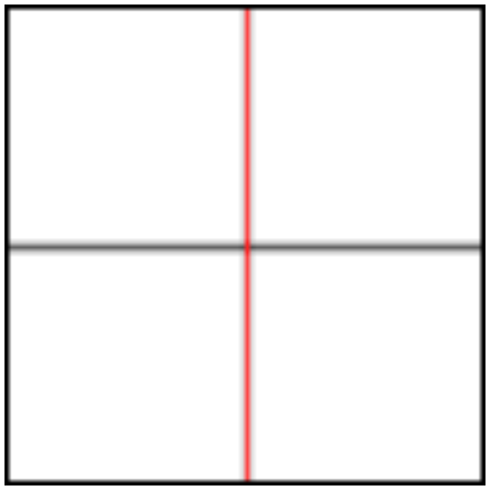


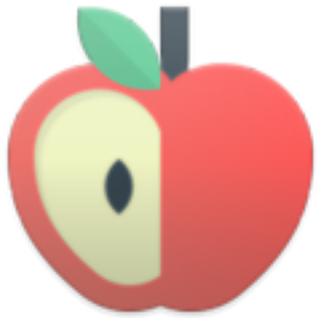
```
apple = Image.open('apple.png')
apple.show()
```

```
apple = blur(apple)
```

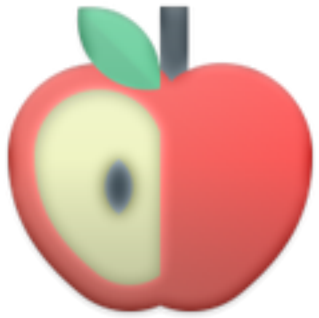
```
apple.show()
```







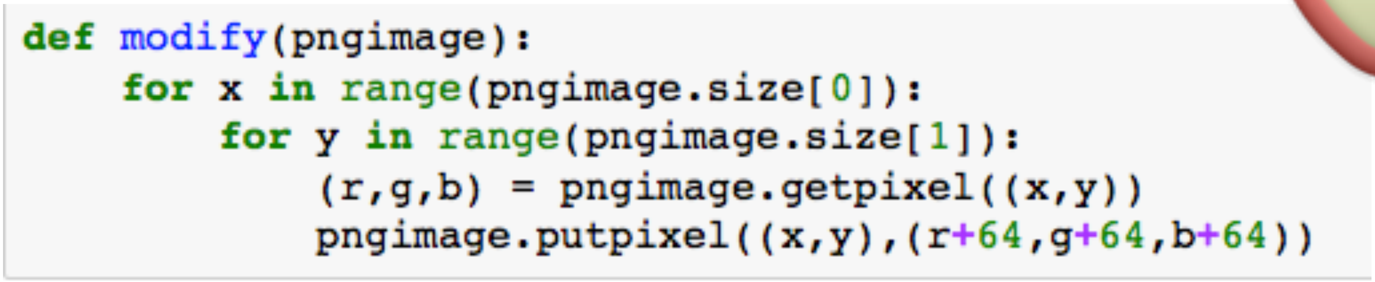
1 blur



20 x blur

# Blurring

how we did it:



```
def modify(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))
            pngimage.putpixel((x,y),(r+64,g+64,b+64))
```

1. Realized that we wanted to make a pixel more similar in color to its neighbors
2. Look at an example (e.g. modify above)
3. Do we need to create a copy of the image? - yes
4. As we iterate through each pixel in the image, we calculate the total color value for the neighbors– so write how to get color value around x,y
   1. Get an average for the colors around x,y
   2. Compare it to the pixel color at that location
   3. Bundle that into a difference function
5. We need to use the difference function to get the desired change value and halve it
6. Write the new pixel value.
7. Test it iteratively

alpha error
index error
type error
round-off error

# Points to understand

for working with images……..

Errors aren't bad things, they just help us get to a robust solution

- Type error resulted in converting floating point to integer, and why we needed to make pixels into integers
- Index errors usually mean you are trying to read or write a pixel outside the image
- Saturation arithmetic (in putpixel) allows us to assume that the pixel values will automatically be bounded between 0,255.
- Its often easier to work with a known entity then generalize
  - writing for one red, then all the color channels
  - working with RGB only, then adding alpha

# Practical Python

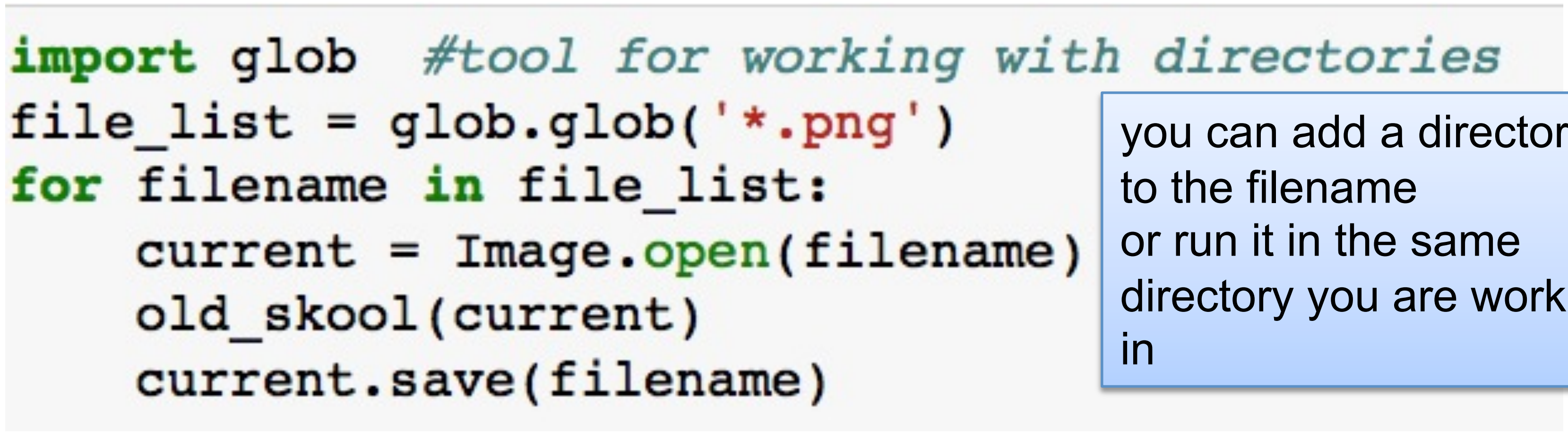
Sample code for doing common image tasks

- Using imageOps
- Downloading images from the internet
- Using images in functions
  - Creating a checkerboard
  - Making a meme
    - with ImageDraw and ImageFont
- Processing a directory of images
  - using glob.glob
  - or using imageMagick

# Processing all files in a directory

for a custom function

- Use glob library



```
import glob  #tool for working with directories
file_list = glob.glob('*.png')
for filename in file_list:
    current = Image.open(filename)
    old_skool(current)
    current.save(filename)
```

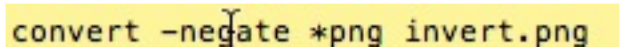
you can add a directory to the filename or run it in the same directory you are working in

**Bonus: Try it with your other filters!**

# Batch Image Modifcation

Command line library to process a  huge number of images

- Image Magic



```
convert -negate *png invert.png
```

Image Magick has a lot of optimized, efficient transforms and will work on a whole directory.

Download it and find out more:
http://imagemagick.org/script/index.php

Install ImageMagic via homebrew  on the command line by typing
  brew install ImageMagick

tutorial video
[https://www.youtube.com/watch?v=ZIT8z8xldmA](https://www.youtube.com/watch?v=ZIT8z8xldmA)

# Culture & Practice

Homework for next class, Wednesday, Oct. 25th

propose a media art project to investigate a relationship

First media arts project proposal

Pick a meaningful message, topic, or issue and propose an investigation using any of the techniques covered in the class. Create a poster about it and present it in the next class.

Components of a media arts project proposal:

- What's the idea, why is it important?
- Describe and sketch how someone might **interact** with your concept
- Make a poster describing your idea
    - Maximum poster size 2' x 2', minimum poster size 8.5" x 11"
- Illustrate how code might function in your project
    - e.g. flow diagram or pseudocode of how you will generate the artwork
    - (it should at least identify where you would get some data and
    - how might you process/analyze/change it)
- After you present your idea for 2-3 minutes, the class will give critique of the concept
- At the conclusion of class, everyone will team up to investigate the ideas further and refine them into a group project

# Homework

1

First project presentation poster proposal, as described in the previous slide.  Bring a poster (8.5” x 11” or larger (but smaller than 2 ‘ x 2’ square)) and present it next class. In class, each student will have 2-3 minutes to present. Each project should have

- a meaningful message or idea you are trying to explore,
- a computational aspect to it (e.g. scraping web pages, analysing some digital information, computing some data)
- have a poster that illustrates how someone might interact with your project

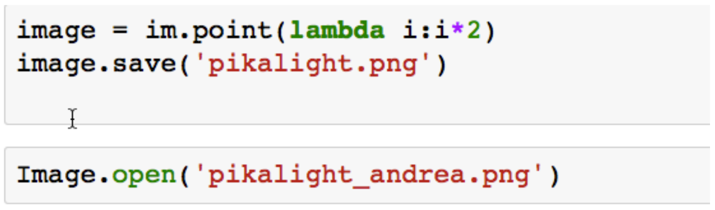
2

Take a look at the quiz results and understand what parts you need practice on. Submit a document stating what you missed and correcting the results by completing individualized challenges that I will provide via email. I’ll make a separate canvas assignment for people to “bump up” their quiz grades if these challenges are completed.

# Media Arts

Also check out the
MediaArts Works the class liked:

https://emerson.instructure.com/courses/1514833/files/folder/Lecture13?preview=80896590



```
image = im.point(lambda i:i*2)
image.save('pikalight.png')
```

```
Image.open('pikalight_andrea.png')
```

## Summary of today

- Technical practice
  - Working with Images, pixel by pixel
  - Creating image transformation functions
  - For extra credit, I'll upload instructions on how to use image magick
- Upload class participation to Canvas
- Optional Challenges to bump up quiz grades