**Outline**

**Review**

- **homework stuff**

New

- Tuples & Images

Programming

- Regex in Python
- Processing large text files
- Tupes
- Images in Python

Practicalities

- Propose an exploration

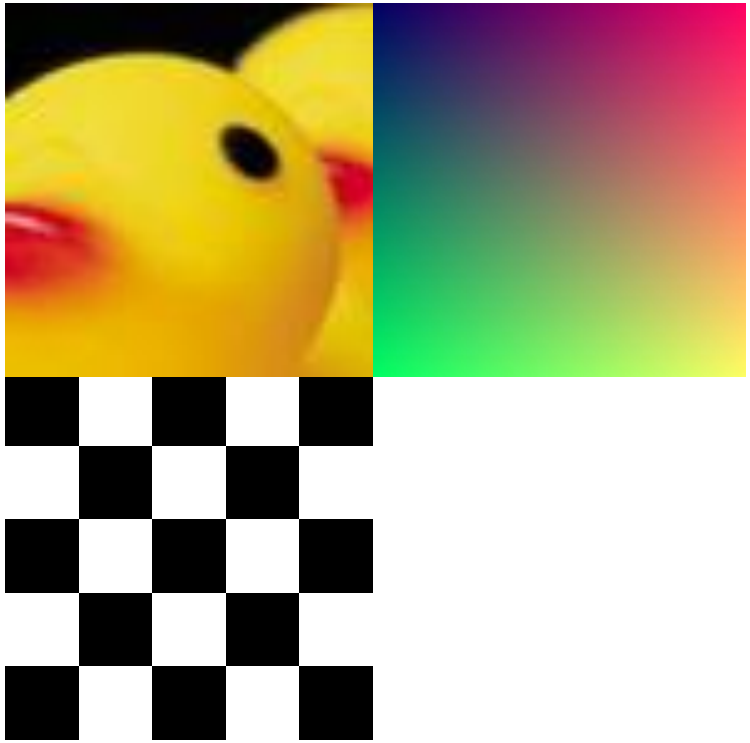
Cool media art pieces

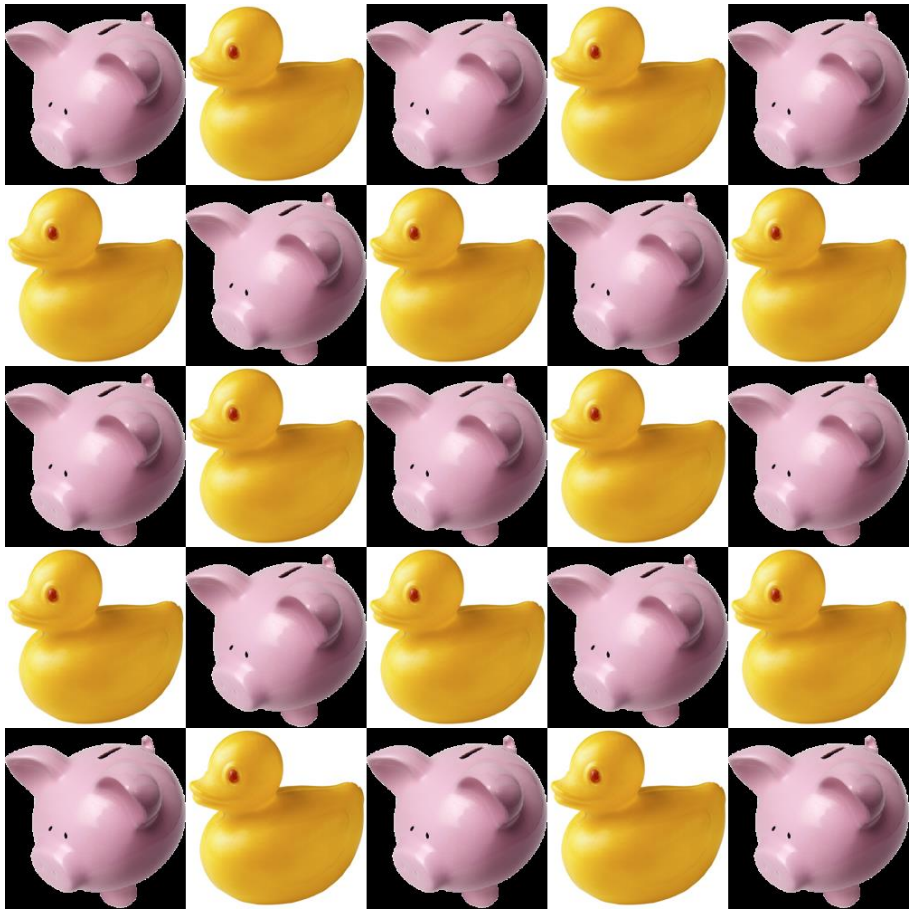
Prof. Angela Chang

Lecture 11: Images I

Fall 2017. Oct 16

# CODE, CULTURE, AND PRACTICE





# IMAGES IN PYTHON USING PIL

python imaging library

# Python librarires

`import` libraries to give us special functions.

# PIL

## python imaging library

other libraries we have seen

**webbrowser** **re** **random** **math**

# Group work through the jupyter

solve those challenges!

| | | |
|---|---|---|
| Benjamin Ehrlich | Alex Winzenread | Jude Hazlett |
| Victoria Jamieson | Evan Iaslovits | Chloe Warfford |
| Ryan Leveillee | Sienna Haines | Sandra Bustamante |
| Alexis Ellis-Alvarez | Derek Aiello | Cameron Murphy |
| Benjamin Epstein | Ryan McDonald | Isaac Switzer |
| | | |
| | | |
| Bin Li | Katja Vujic | |
| Rey Sleiman Sawan | Chance Molenda | |
| Samuel Hinrichs | Trevor Howell | |
| Elizabeth Skerry | David Mekibel | |
| Minh Do | Nathanael King | |

# Learning about libraries and docstrings

Try writing a doc string and try asking about an object using these techniques.

To use a library, call the import statement with library:name

```
import re
```

dir will list a library or object's the functions:

```
dir(re)
```

```
dir([])
```

Doc strings = "documentation strings", messages about functions

To read a docstring

Object.__doc__

note the double underscores

```
print [].count.__doc__
```

```
print len.__doc__
```

To create a docstring when you write a function, just it on a newline after def

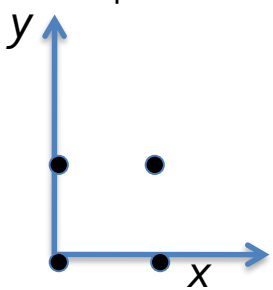
```
#write a docstring
def tax(subtotal):
    'returns the tax on a meal in Massachusetts'
    return subtotal * 0.0625
```

ps. Google / stackoverflow point PIL manual:

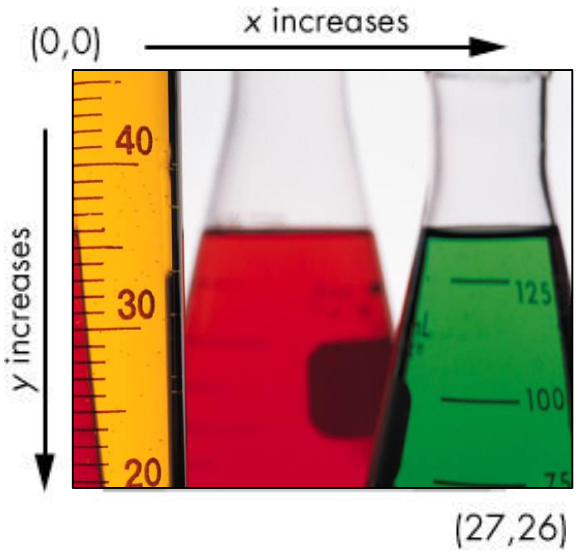
http://www.effbot.org/imagingbook/pil-index.htm

# From 1D to 2D

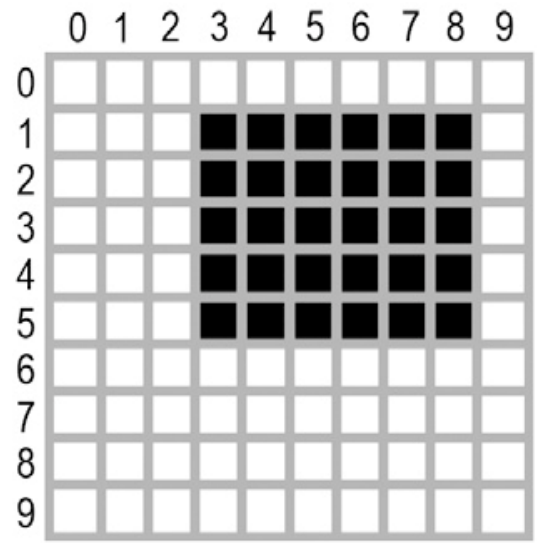
Extending from our work with numbers and lists…

| Dimension | Representation | Feature | Example |
|---|---|---|---|
| 0 | Point  | "No length, no width" | Single number by itself<br>20 3.14159 -13 |
| 1 | Line  | "has length or width" | Numbers … -3,-2,-1,0,1,2,3…<br>Lists, sequences have len()<br>'hello world'<br>[3,4,5] |
| 2 | Shape   | "length and width" | Pairs of points<br>rectangle (0,0),(0,1),(1,0),(1,1)<br>map coordinates<br>other shapes |

# Images

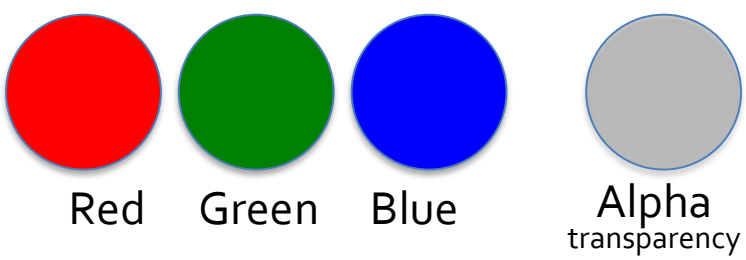



x, y increase to lower right

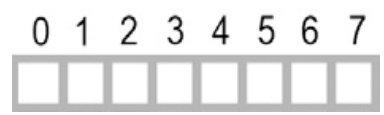



Area is represented by the box tuple (3,1,9,6)

Each pixel consists three (or four channels):




Each channel has a value:




($2^3$) or 8 bits = 1 byte

0-255 (0-FF hex)

Hex #FFFFFF is white
-Full red, green, and blue

# ( tuples )

pairs, triples, quadruples, and so on.

Specified by rounded parenthesis: ( o, o) (latitude, longitude)

Puts things together, but is immutable (once created it does not change).

Create some tuples in iPython notebook:

```
twod = (42,17) #note parenthesis for tuples
```

```
threed = (7,6,14)
```

```
twod
```

```
threed
```

Like lists, but you can't change them.

# Tuples vs Lists

```
twod[-1]
```

```
threed[-1]
```

```
threed[0] = 9
```

Guess what's going to happen before you press enter!

```
alist = [1,2,3]
```

```
alist[0]=15
```

```
alist
```

Does change happen? No, tuples are immutable!

Tuples are more computationally efficient. Useful for very large numbers of numbers.... like all the pixels in an image.

A new type also helps us prevent errors.

```
def to_f(c):
    return (9/5)*c + 32
```

```
to_f("hello") #helpful error
```

## But, can we iterate over it?

```
for i in range(5):  #can make a lot of tuples by iterating
    print (i,2,3)
```

# Double

```
#double double from earlier
def double(sequence):
    result = [];
    for element in sequence:
        result = result + [element*2]
    return result
```

```
double(alist)
```

```
[2, 4, 6]
```

```
double((2,3,4))
```

# iterating over a tuple here!

```
[4, 6, 8]
```

So we can use it to iterate computations---
and work on bitmap images this way,
we just need 2 iterators to go in each x and y direction.

# Import PIL and create an image

Python Imaging Library→ Image.new function

```
#import the PIL Library
from PIL import Image
```

```
ourimage = Image.new('RGB',(100,100), 'white')
#mode = 'RGB'
#size = (100,100)
#color = 'white'
```

Did you get an error?

NO

```
#save the images
ourimage.save("allwhite.png")
```

Switch to your directory and admire your handiwork. Try changing values.

YES

```
NameError: name 'Image' is not defined
```

If you see 'Image' not defined – that means python doesn't have the Image library. Import it again using

```
from PIL import Image
```

Then try again.

Image link

# Image creation

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

```
mode = 'RGB'  #the format could also take
size = (100,100)
color = 'red'
```

```
ourimage2 = Image.new(mode,size,color)
```

```
ourimage2.save('allred.png')
```

Make some images.
Admire your handiwork.

Which number is width, which is height in size?
Try using tuples for the color.

```
ourimage = Image.new('RGB',(150,100),'blue')
```

```
ourimage.save('allblue_notsquare.png')
```

```
ourimage = Image.new('RGB',(100,100), (127,127,127)  )
```

```
ourimage.save('allgray.png')
```

```
ourimage = Image.new('RGB',(100,100), (200,127,127)  )
```

```
ourimage.save('allrose.png')
```

# Changing images

```
allblack = Image.new('RGB',(100,100),(0,0,0))
```

```
allblack.save("allblack.png")
```

```
allblack.putpixel((50,50),(255,255,255))
```

```
allblack.save("almostall.png")
```

```
#try drawing a line
```

```
for i in range(100):
    allblack.putpixel((i,25),(255,255,255))
```

```
allblack.save('oneline.png')
```

Can you draw a line in the other direction?

# Iterating through all the pixels

```
#iterate through the entire image
rectangle = Image.new('RGB',(150,100),(0,0,0))
```

```
for x in range(150):
    for y in range(100):
        rectangle.putpixel((x,y),(127,127,127))
```

```
rectangle.save('rectangle.png')
```

```
#now change the color of each pixel
```

```
for x in range(150):
    for y in range(100):
        rectangle.putpixel((x,y),(x,x,x))
```

```
rectangle.save('gradient.png')
```

Now try a vertical gradient or even a diagonal gradient.

# Generalizing image operations

image.size → (width,height)
returns width and height in pixels

index 0 will give you the width
index 1 will give you the height

```
#generalize to images of any size
rectangle.size
```

```
rectangle.size[0]
```

```
rectangle.size[1]
```

```
bigger = Image.new('RGB',(500,600),(0,0,0))
```

```
bigger.size
```

```
bigger.size[0]
```

```
bigger.size[1]
```

now the for loop can change according to the input image

```
# bundle grayout function generalized for images of different sizes
def grayout(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            pngimage.putpixel((x,y),(127,127,127))
```

```
test = Image.new('RGB',(150,100), (0,0,0))
```

```
grayout(test)
```

```
test.save('test1.png')
```

# Load and save external images

```
#loading an existing image
ourimage = Image.open('heart.png')
```

```
ourimage.save('ours.png')
```

```
#check that these are the same image
#now tryout our grayout function
grayout(ourimage)
```

```
ourimage.save('ournew.png')
```

```
#check that this new image is all grayed out
#otherwise check your function or restart ipython
```

Now we can write functions that operate on images.

# Lighten or darken an image

Need to look at pixels.

```
ourimage.getpixel((0,0))   #get a pixel from the image
```

```
transparent = Image.open('heart_trans.png')
```

```
transparent.getpixel((0,0))
```

```
#fill in the blanks to generalize the size
def modify(pngimage):
    for x in ____:
        for y in ____:
            (r,g,b) = pngimage.getpixel((x,y))
            pngimage.putpixel((x,y),(r,g,b))
```

# Not quite modify:

```
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,g,b))
```

```
ourimage = Image.open('heart.png')
```

```
modify(ourimage)
```

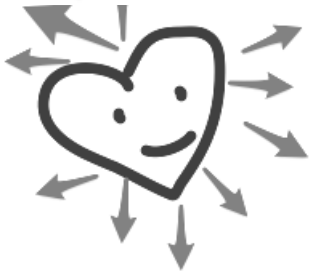
```
#verify it gets saved out
ourimage.save('modheart.png')
#now actually modify the function to change the intensity
#by lightening the image\ procesing\ 1.ipynb
```

# Lighten the image

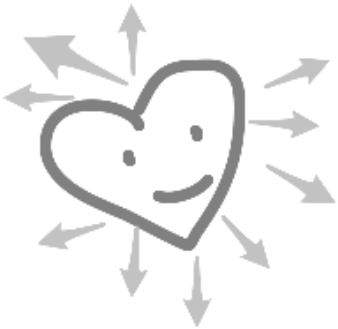
```
def modify(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))
            pngimage.putpixel((x,y),(r+64,g+64,b+64))
```

```
#note no errors if r,g,b > 255
#putpixel uses saturation arithmetic
```

before: heart.png



after: modheart.png



2x lighten:
modheart2.png

Transparent images? Use a slice to isolate alpha channel.

```
#generalize so modify works with transparent images too
def modify(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))[:3] #get colors only
            new_color = ( r+64, g+64, b+64)
            new_color = new_color + pngimage.getpixel((x,y))[3:]#add transparency
            pngimage.putpixel((x,y),new_color)
```

before: heart_trans.png

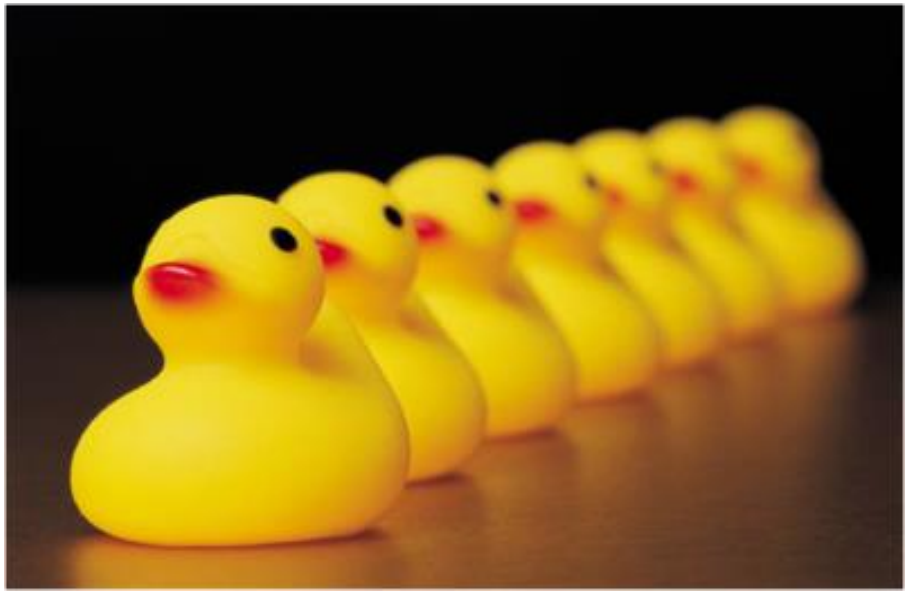
after: modtransparent.png

# How to darken an image?

Move every pixel closer to black. (r,g,b)=(o,o,o)

```
def darken(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))[:3] #get colors only
            new_color = ( r-64, g-64, b-64)
            new_color = new_color + pngimage.getpixel((x,y))[3:]#add transparency
            pngimage.putpixel((x,y),new_color)
```

before duckies.png

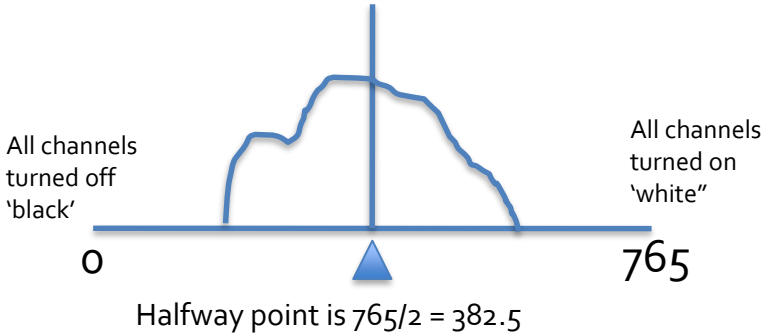


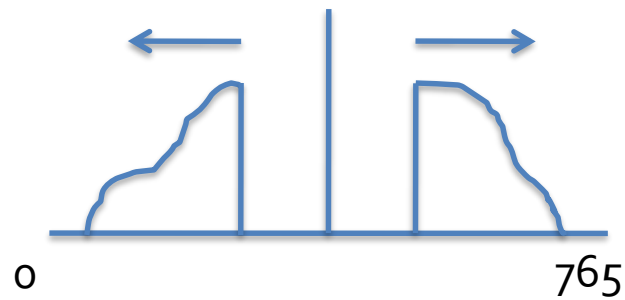
after: darkduckies.png

# Increasing contrast

Adding conditionals to pixel processing

Move the pixels depending on where they are relative to the halfway point.



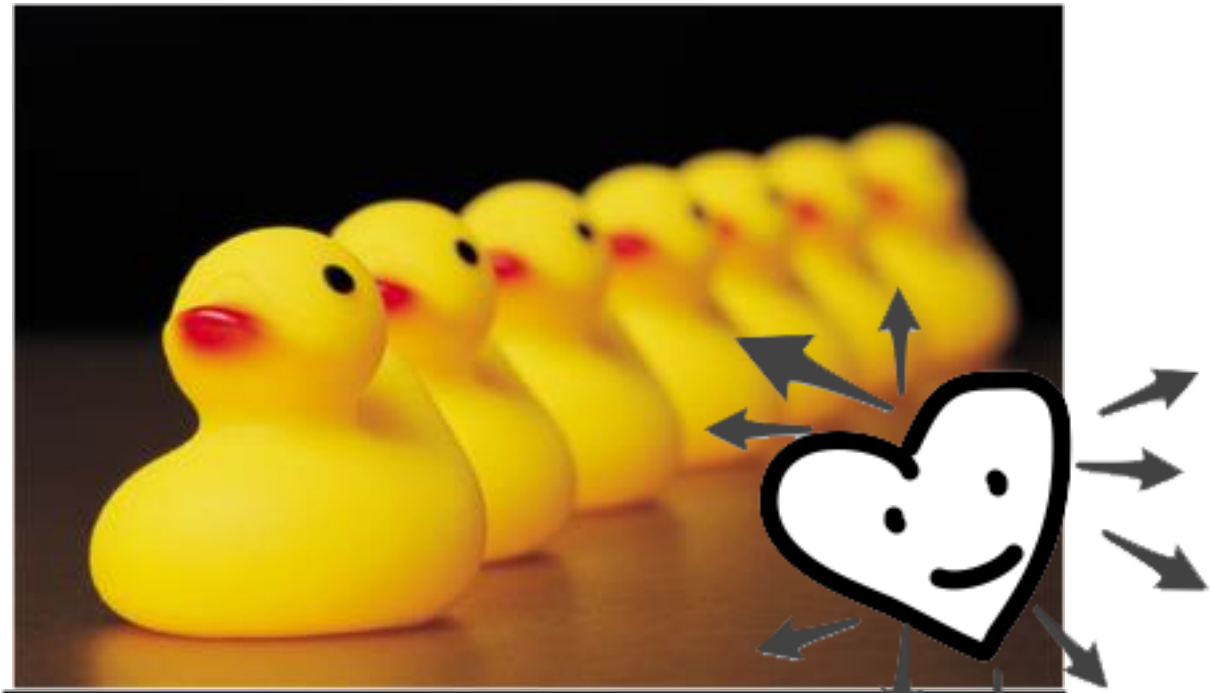




We want to move pixels closer to extremes:
Shift lighter values closer to white R+G+B = 0,
Shift darker values closer to black R+G+B = 255 *3= 765.

```
def contrast(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))[:3] #get colors only
            a = pngimage.getpixel((x,y))[3:]
            if r+g+b < 382.5:
                pngimage.putpixel((x,y), ((r - 64), (g - 64), (b - 64))+a)
            else:
                pngimage.putpixel((x,y), ((r + 64), (g + 64), (b + 64))+a)
```

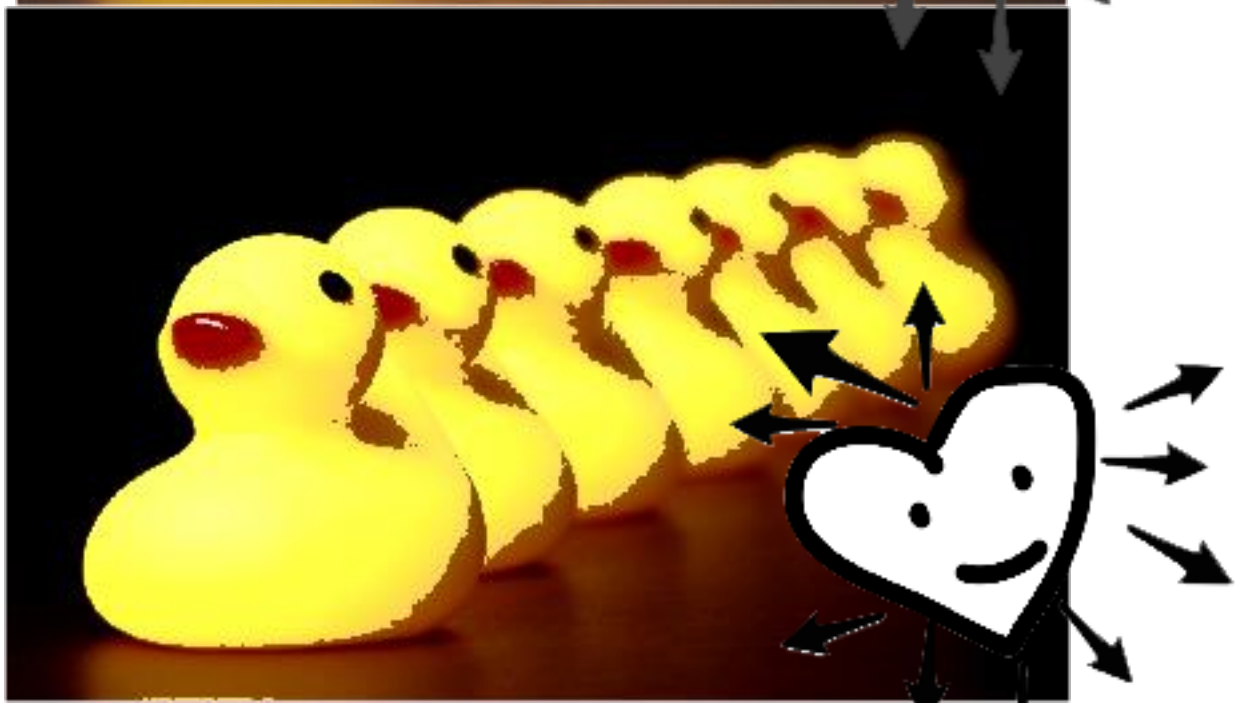
Save the alpha channel

Conditional statement: Lighten or darken if the values are closer to white or black



Before

duckies.png
heart_trans.png

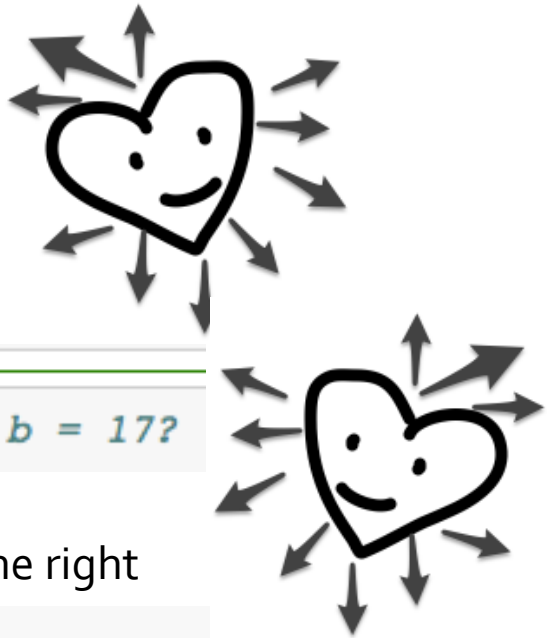


After

contrast_ducks.png
contrast_hearttrans.png

# Flip horizontal

Swapping pixel values



```
a = 17
b = 2
```

```
#how to flip the values so that a = 2 and b = 17?
```

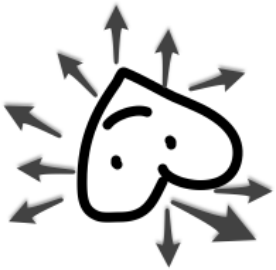
take a pixel from the left, and swap it with a pixel on the right

```
def fliph(pngimage):
    width = pngimage.size[0]
    for y in range(pngimage.size[1]):
        for x in range(width/2):        #width/2 –don't flip x twice
            left = pngimage.getpixel((x,y))
            right = pngimage.getpixel((width - 1 - x,y ))
            pngimage.putpixel((width - 1 - x, y), left)
            pngimage.putpixel((x,y), right)
```

maximum width position is width-1
at x=width-1 is the rightmost pixel

# Flip 180 degrees

template = use a temporary variable to swap values



```
a = 17
b = 2
```

```
#how to flip the values so that a = 2 and b = 17?
```

```
#create a temp variable
temp = a
a = b
b = temp
```

```
(a,b)
```

```
def flip180(pngimage):
    width = pngimage.size[0]
    height =pngimage.size[1]
    newimage = Image.new('RGB',(height,width),'white')
    for y in range(height):
        for x in range(width):
            newpixel = pngimage.getpixel((x,y))
            newimage.putpixel((height-x-1,width-y-1),newpixel)
    pngimage = newimage
    newimage.save('rot180.png')
```

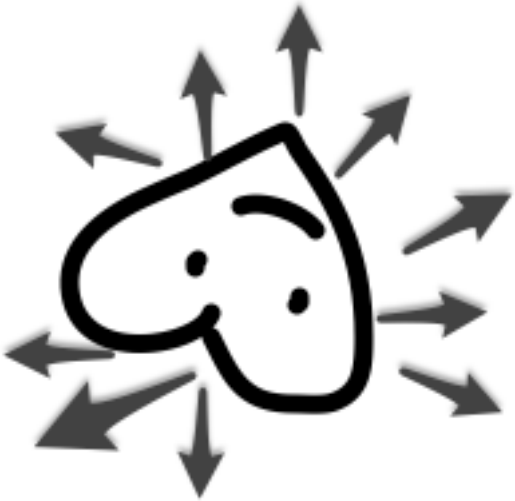
so the pixel at 0,0 moves to height-1, width-1

# More transforms:

group exercise: get together with a friend and create functions to:

How might you flip it vertically

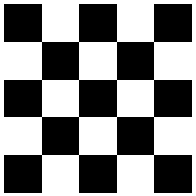
How might you rotate an image?



How might you flip x/y?

# Tiling blocks

Using loops to paste an image



```
tile = Image.new('RGB', (10,10),'black')
tilew,tileh = tile.size
print tilew, tileh
```

A small black block for tiling

```
background = Image.new('RGB',(tilew*5,tileh*5),'white')
background.show()
print background.size
i = 0
for left in range( 0, tilew*5 , tilew):
    for top in range( 0, tileh * 5, tileh):
        if i%2 == 0:
            background.paste(tile,(left,top))
        i+=1
background.save('checkerboard.png')
```

Make an image that s as big as 5x5 grid of tiles

Counter for the images we paste

These two loops go along the length and width of the image, by imcrements of the tile dimensions.

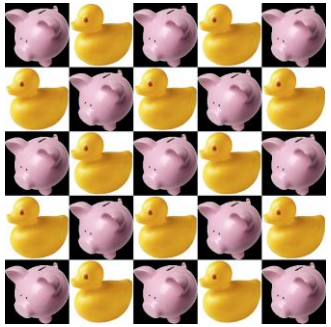
% is a *modulo* operator
"give me the remainder after division"
i%2 returns 0 if i is even, is 1 if i is odd

Paste the tile at left,top if the counter is even

Feel free to play with different parameters

USEFUL FOR HOMEWORK

# Tiling images



Note: you might want to resize images before tiling

```
sqduck = Image.open('squareduck.png')
print sqduck.size
sqpig = Image.open('piggy_trans.png')
print sqpig.size
sqpig = sqpig.resize((200,200))
```

If you have two png images that are 200x200 in size ,you can tile them.

```
background = Image.new('RGB',(1000,1000),'white')
background.show()
print background.size
i = 0
for left in range( 0, 1000 , 200):
    for top in range( 0, 1000, 200):
        if i%2 == 0:
            background.paste(sqpig,(left,top))
        else:
            background.paste(sqduck,(left,top))
        i+=1
background.save('pigsandducks.png')
```

Two images

# First Project Proposal

Homework for next week, Wednesday, October 20, 2017.
propose a media art project

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.
  - 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

# First project assignment

1

Work through the image processing 1 ipython notebook.
Play with the images, change variables, etc. **Answer any questions posed and create your checkerboard function** by altering the data in the exampl at the end off the notebook.
http://bit.ly/2017ImageClass

2

Media arts homework due Wednesday
http://bit.ly/2017HW6MediaArts
Pick an artist, critique his work from the point of putting your own spin on it.

3

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 technical aspect to it (e.g. scraping web pages, analyzing text)
- have a poster that illustrates how someone might interact with your project

# Summary of today

- Technical practice
  - Working with Images, pixel by pixel
  - Creating image transformation functions
- Class Participation – Upload a zip file with your Jupyter notebook from today, along with generated text and images
- Homework - Readings on technology and questions due Wednesday