Checking out the neighbors
In *Rear Window, 1954*

Lecture 12: Image Processing 2

Fall 2017. Oct 18

CODE, CULTURE, AND PRACTICE

Outline

Review

- Tuples & Images

Programming

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

Practicalities

- Syllabus review
- Homework

Cool media art pieces

Things to know

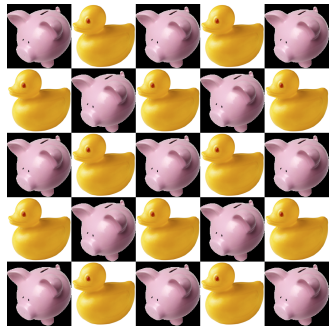
about image processing. Do you know how to:

- iterate in 2-dimensionsthrough an image?
- read a pixel's color value and store it in a tuple?
- write a pixel's color value?
- create a function to process an image globally?
  - e.g. (invert, lighten, darken)
  - define a function to process the each pixel by scanning every row and column
  - move each pixel's color closer or farther to end of range (0, 255)
- processes pixels conditionally?
  - test (r,g,b) value and do something based on that
- return the image from the function?

Tiling images

If you have two square 200x200 images,you can tile them.

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



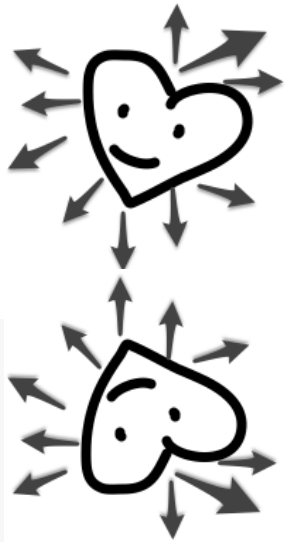
Resize images before tiling

```
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')
```

You should be able to use any input images and create a new checkerboard.
You can change the tile size (200x200) and background size, but it has to be an odd #.

Animating a flip

show the steps when you aren't sure what's happening



```
def flipxy(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('flipxy.png')
```

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

We're going to step through it by saving each pixel transfer as a sequence of frames.

Lecture 12 images.ipynb

Tiling blocks

Using loops to paste a small image inside a big one

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

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')
```

small 10x10 tile

white background that can hold 5 x 5 tiles

count every tile as we paste it

use range skip to lay out tiles according to tile dimensions

% is a *modulo* operator
i%2 ==0 whenever I is even
“paste the black tile on every other space”

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

Picture transforms:

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

How might you flip it horizontally

read through your pixels moving from left to horizontal midpoint
grab each pixel from the left and swap with the complementary right pixel
save the new image

How might you flip it vertically

read through your pixels moving from top to vertical midpoint
grab each pixel from the top and swap with the complementary bottom pixel
save the new image

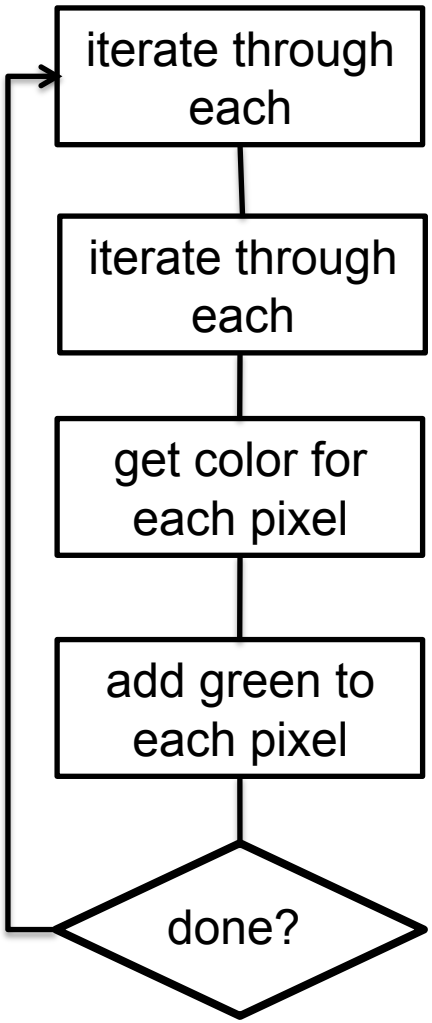
How might you rotate an image?

create a new image with height, width swapped
read through your original pixels
write new pixels in new image with swapped x,y
save the new image

Drawing out image processing

Group exercise.

Diagram the process for creating the following image manipulations.



invert-- Inverting al the RGB colors of an image.

oldskool - Convert ordinary pictures to make them look monochrome, green on black.

sick – Tinting a yellow/green/brown tinge on a picture so things look sickly

Lecture 12 Blur & Group Challenges & Homework.ipynb

Example: Inverting

How might we design an image inverter?

Black pixel value = 0 White pixel value = 255

1. Iterate through the image
2. Do we need to create a copy of the image?
3. Look at an example (e.g. 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+64,g+64,b+64))
```

4. As we iterate through the image, we check each pixel value
5. Then we change the value. In this case, we need to invert the pixel value. If we have black (0), we want to change to 255. What arithmetic?
6. Write the new pixel value.

Invert

```
def invert(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r, g, b) = pngimage.getpixel((x,y))[:3]
            alpha = pngimage.getpixel((x,y))[3:]
            pngimage.putpixel((x, y), ( 255 - r, 255 - g, 255 - b) + alpha)
```

Example: Oldskool

How might we design an oldskool fitter?

Bump up the green and decrease the other values

1. Iterate through the image
2. Do we need to create a copy of the image? N
3. Look at an example (e.g. 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+64,g+64,b+64))
```

4. As we iterate through the image, we check each pixel value
5. Then we change the value. In this case, we need to add some value to green, and decrease red and blue
6. Write the new pixel value.

Example: sick

How might we design a sick filter?

Bump up the read and green and decrease blue values—
lemon yellow FFFF00 , vomit yellow = **c7c10c**
(255,255,0) 199, 193, 12)

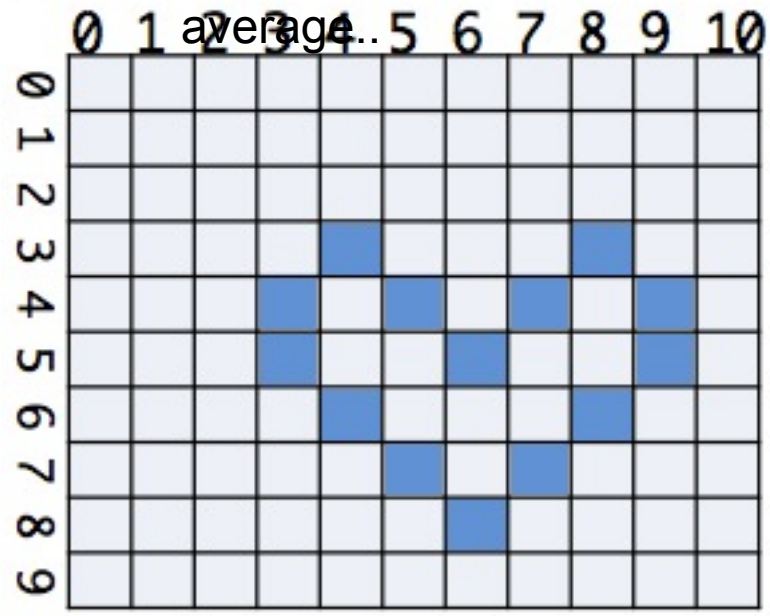
1. Iterate through the image
2. Do we need to create a copy of the image? N
3. Look at an example (e.g. modify)

```
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+64,g+64,b+64))
```

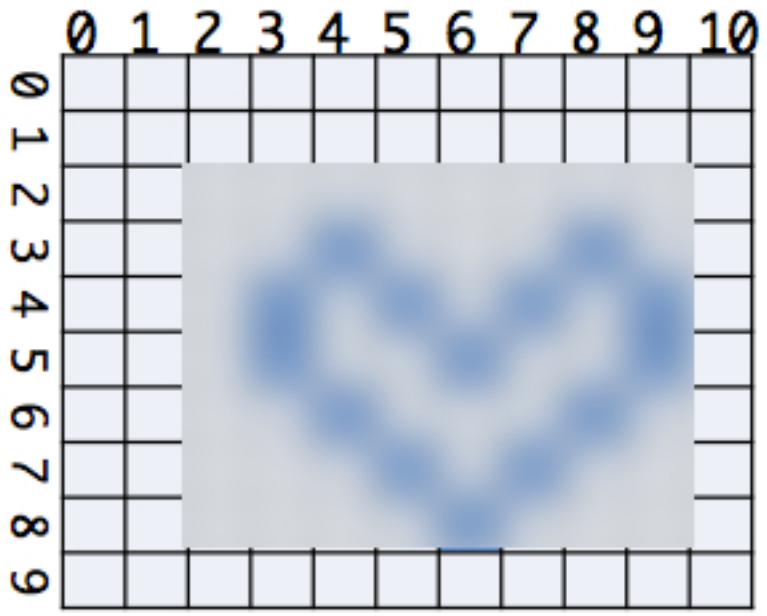
4. As we iterate through the image, we check each pixel value
5. Then we change the value. In this case, we need to add some value to red and green, the blue
6. Write the new pixel value.

How to write a Blur function?

Look at neighboring pixels and try to be more similar to the average...



Sharp



Blurry

Need some code to “look at the neighboring pixels” for every pixel.

Blurring

how we did it:

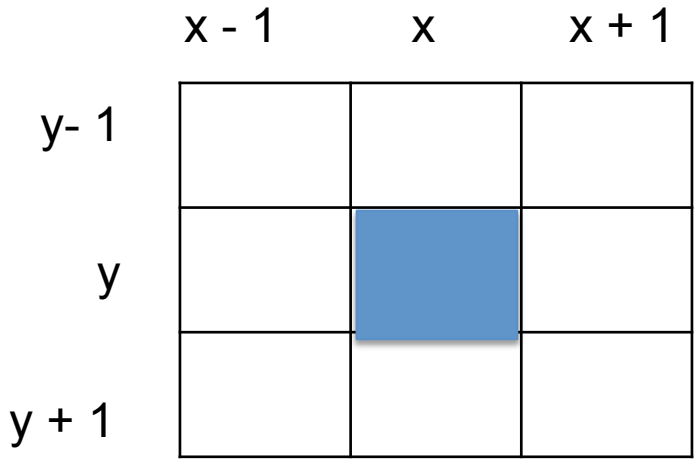
```
def modify(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))
            pngimage.putpixel((x,y),(r+64,g+64,b+64))
```

1. Realized that we wanted to make a pixel more similar in color to its neighbors -→ be like the average
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

Look at the neighbors

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

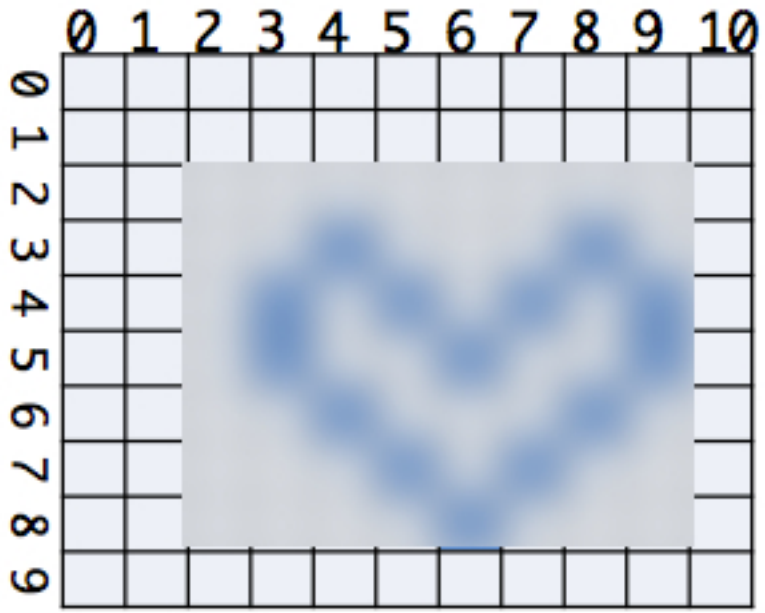


Average out the neighboring pixel values, and shift the center pixel closer to the average.

```
#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

```
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
```

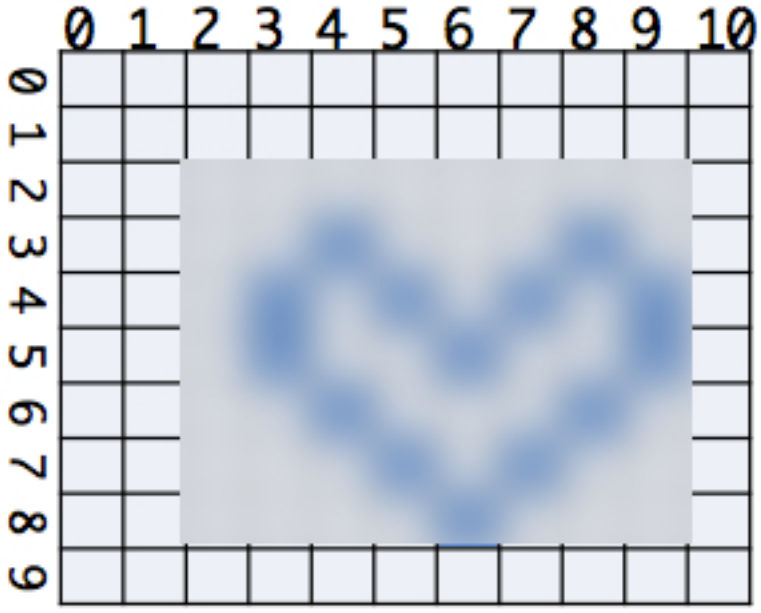
Make a copy

Run it, and ….

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

Removing index error

```
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….

```
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!

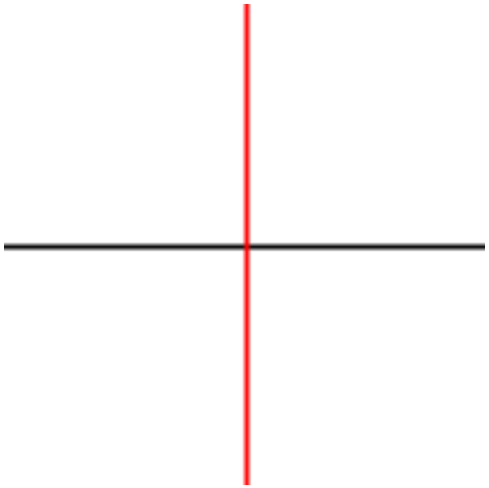
```
twoline = Image.new('RGB',(100,100),'white')
```

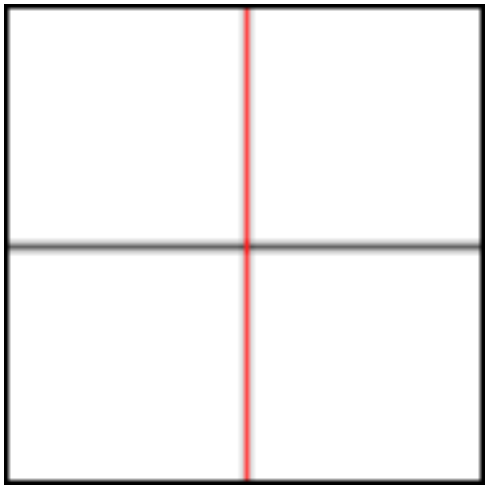
```
for x in range(0, twoline.size[0]):
    for y in range(0, twoline.size[1]):
        twoline.putpixel((x,50),(0,0,0))
        twoline.putpixel((50,y),(255,0,0))

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

Homework

1. Work through the image processing 1 ipython notebook. Play with the images, change variables, etc. **Create your own image manipulation function to do something fairly simple (such as flip along a different axis) or something less usual (posterize the image).** Try to use arithmetic or conditionals of some sort.

2. I'll be getting your quiz results to you this week. 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.

Summary of today

- Technical practice
  - Working with Images, pixel by pixel
  - Creating image transformation functions
- Upload the Lecture 12 Blur & Group Challenges & Homework.ipynb for homework