Lab 3 / Part 1: Manipulating photos with loops

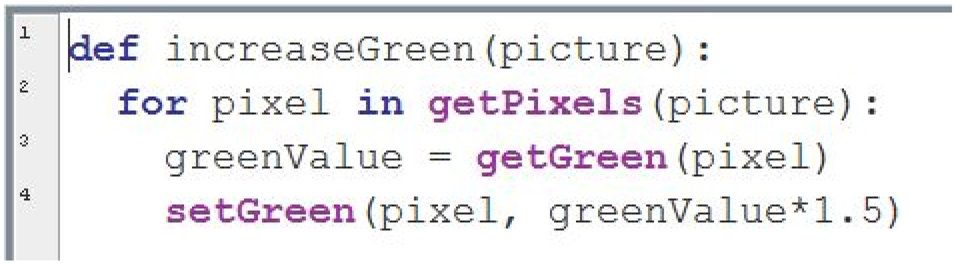
Getting Started

In my code below I will be using an image file, beach.jpg, available at
http://sergey.cs.uni.edu/courses/cs1120/mediasources/beach.jpg

The first time through this lab I would ask you to use that image as well so that our answers are consistent. But I will later ask you to use any second photo of your choice. You can grab a different photo from the mediasources or use any photo of your choice. My only suggestion is that you use one whose dimensions aren't much larger than 500 in either direction. If you have a large hi-res image you have taken with your digital camera or cell phone you should open it in some photo tool and scale it down to an "appropriate size".

Activity A: Getting started with functions that manipulate entire photos

Launch JES. Now, type the following program into the Program Area of JES and save it as "lab3a.py". Make sure that you pay attention to the spelling and the spacing of the different features.



```
def increaseGreen(picture):
  for pixel in getPixels(picture):
    greenValue = getGreen(pixel)
    setGreen(pixel, greenValue*1.5)
```

When you have it done re-save and load this file.

Fix any syntax errors you may have made.

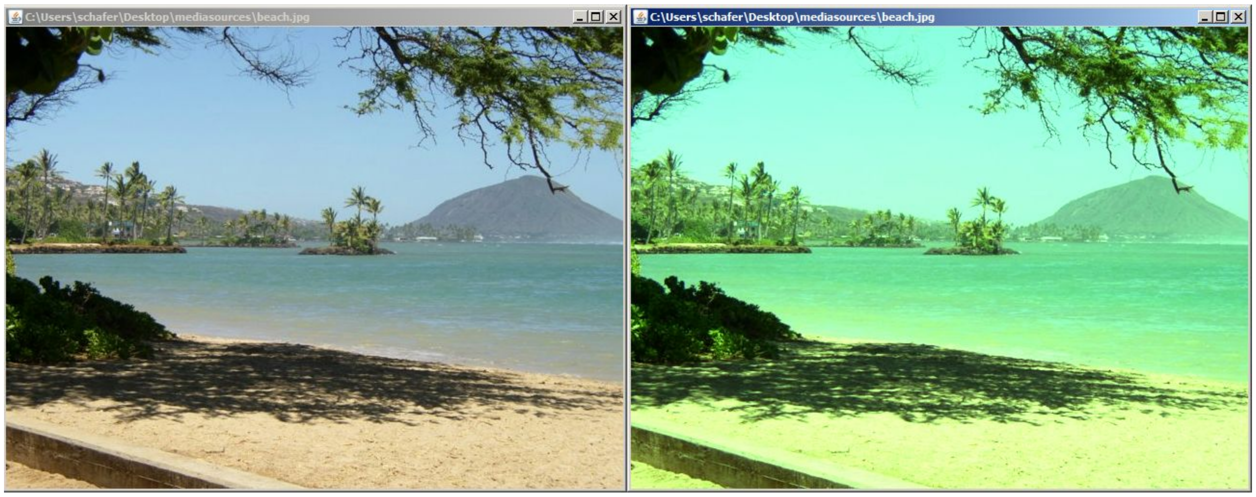
When everything loads properly type the following commands:

```
>>> setMediaPath()
>>> original = makePicture("beach.jpg")
>>> changed = makePicture("beach.jpg")
>>> increaseGreen(changed)
>>> show(original)
>>> show(changed)
```

When you invoke setMediaPath() you will need to navigate to wherever your images are stored.

At this point two pictures should be on the screen - although changed may have popped up right over the top of the original photo. Move them side by side so you can compare them:

Can you see the difference? Mine looked like:


C:\Users\schafer\Desktop\mediasources\beach.jpg
C:\Users\schafer\Desktop\mediasources\beach.jpg

Suppose that we wanted to confirm that they really ARE different however. Well, then we might type:

```
>>> explore(original)

>>> explore(changed)
```

This pops up two windows that look similar to the ones above, but that allow you to identify the colors at specific pixels. Let's consider one spot out on the water. Using the boxes at the top of each explore window enter the pixel (225,250).

[Q1] What are the RGB color values of the pixel from the original image?

[Q2] What are the RGB color values of the pixel from the changed image?

[Q3] Compare the red values between these two pixels. How do they compare?

[Q4] Compare the green values between these two pixels. How do they compare?

[Q5] Compare the blue values between these two pixels. How do they compare?

If you did everything correctly you should notice that red and blue stayed the same but that green changed. In fact, the green value of the changed image is exactly 1.5 times that of the green value in the original image. You might expect that is true for all pixels, but in fact, it is not.

Repeat this process with the pixel (100,400).

[Q6] What are the RGB color values of the pixel from the original image?

[Q7] What are the RGB color values of the pixel from the changed image?

[Q8] Notice that the green value in the changed pixel is NOT 1.5 times the green value of the original pixel. Why do you think that is (HINT. What number is 1.5 times the green value of the original pixel? Why can't the computer set to that value?)

There are times that you will want to do what we just did using the explorer windows but there are other times where it will be more helpful to do this with code/commands instead. Type in the following:

```
>>> originalPixel = getPixel(original, 100, 100)

>>> print originalPixel

>>> changedPixel = getPixel(changed, 100, 100)

>>> print changedPixel
```