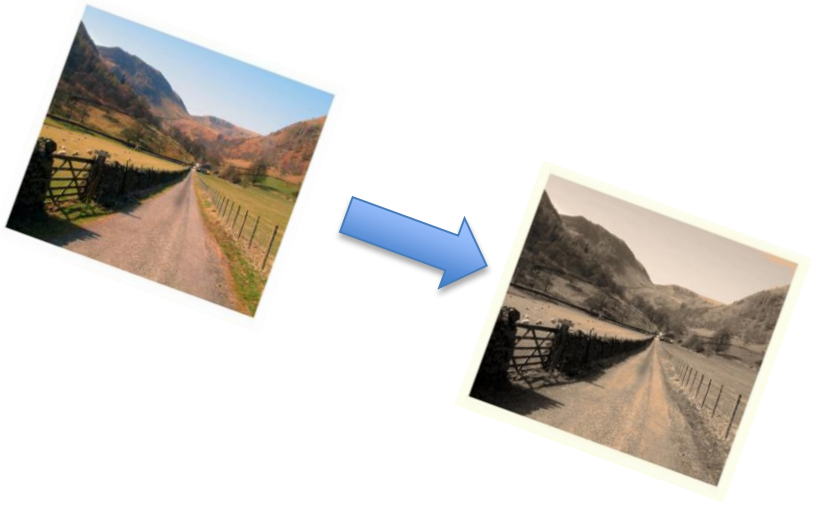
Lab: Day 2



What you'll do:

Replacing colors

Posterizing

Sepia-Tints

Mirrors

What you'll learn:

Conditionals

More for-loops

Using ranges

Implementing algorithms

Exercise 1:

Replace brown spots in a picture with red!

- Make a lab2 folder in your jes-4-3-nojava folder
- Find a picture online (smaller than 800x800 pixels) with some brown tints that you'd like to turn red. Save the photo into your lab2 folder as "filename.jpg".
- Open JES and save this JES document as "Lab 2" in your lab2 folder.
- Define a new function named `turnRed()` that takes no inputs. Now we'll write the `turnRed()` function.

The Big Picture: You'll turn your file into a picture and then use the MediaTools (drop-down menu) to find the RGB values for the brown in your picture. Then, for each pixel in your picture, if the pixel's color is close to brown, you'll set the redness of that pixel to be 1.5 times greater.

- First, within your turnRed function, turn your file into a picture.

```
def turnRed():
      #turn your filename into a file using "lab2/filename.jpg"
      #make a picture out of the file using the makePicture(file) function
```

- Then, load your program so that JES knows your picture exists now. Click on the MediaTools dropdown menu and select "picture tools." Select the picture variable you just made, and your picture should now pop up. Click on a brown spot and record its RGB values (see image example below). We're going to use these values later in our function.

3) Notice x & y coordinates

2) Record RGB values

1) Click a brown spot on picture

- Using these RGB values, make the color brown using MakeColor(redValue, greenValue, blueValue) and assign it to a variable. Now you're all set to write the rest of the function!

  Here's an example in pseudocode:

```
def turnRed():
      make a file using your filepath
      make a picture out of the file
      make a brown color and assign it to a variable
      for every pixel in the picture
            find the color of the pixel & assign that value to a
variable
            if the color's distance to brown is less than 50:
                  get current redness of pixel and increase by 1.5 (and
                  assign this value to a variable)
                  set the redness to be the new color you've just found
      show the final result
```

  Recall the structure of a for loop…

```
for item in items:
      do something with item
```

  Handy built-in functions that you'll find helpful:

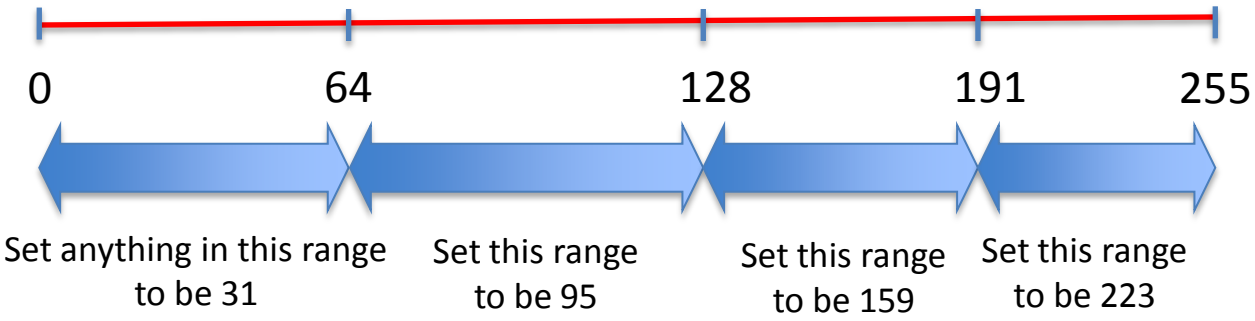
```
makeColor(redValue, greenValue, blueValue)
makePicture(file)
getPixels(picture)
getColor(pixel)
distance(color, anotherColor)
getRed(pixel)
setRed(pixel, redness)
show(picture)
```

- After you've fully created your function, call this function at the bottom of your lab2 document by typing turnRed().
- Load your program. You should now see your picture with browns replaced by redder tints!

Exercise 2: Posterizing!

Aka, reducing the range of colors in a picture.

The Big Picture: Look for a range of colors and map them onto a single color!



Here's an idea of how to posterize red colors:

```
def posterize(picture):
      for each pixel in the picture:
            get the red value of the pixel and call it "red"
            if red is less than 64
                  set red to be 31 (or some other intermediate value)
            if red is between 64 and 128
                  set red to be 95 (or some other intermediate value)
            (etc.)
```

Recall the structure of an "if" statement…

```
if condition:
      body
if another-condition:
      another-body
if another-condition:
# or alternatively, use "else: " to catch all other conditions
      another-body
```

Useful built-in functions for this exercise:

```
getPixels(picture)
getRed(pixel)
getGreen(pixel)
getBlue(pixel)
setRed(pixel, newValue)
setGreen(pixel, newValue)
setBlue(pixel, newValue)
```

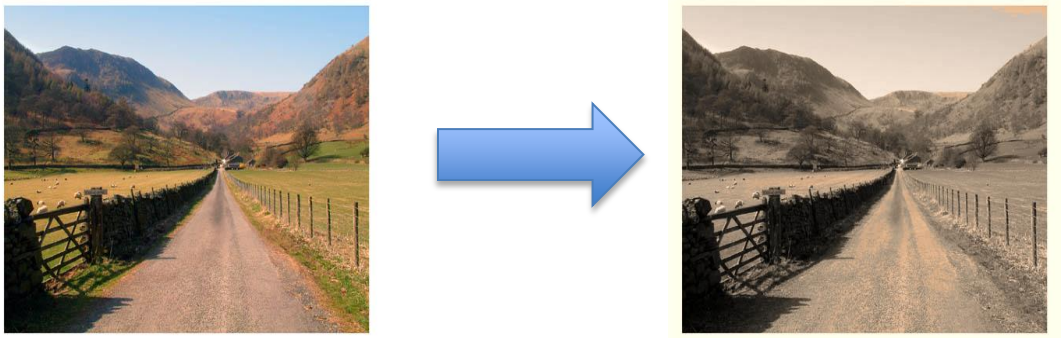
Below this function:

- Add code to make a picture using the `makePicture(file)` function, call the `posterize(picture)` function on this picture, and use the function `show(picture)` to show the picture. If you'd like, save your new picture into your lab2 folder using: `writePictureTo(picture, "lab2/filename.jpg")`
- After loading the program, you should see your posterized picture! Try changing the intermediate values you had set your ranges to be, and see how this changes the outcome of your picture.

Bonus exercise:

Add to your for loop so that the same posterizing modifications will be done for green and blue.

Exercise 3: Sepia-toned prints



Big Picture: we can't just increase the amount of yellow because it's not a one-to-one correspondence. Instead, colors in different ranges get converted to different colors!

The Strategy:

- First, convert the picture to greyscale.
- Then, for tint shadows (pixels where the red value is < 65 or so), set the red to be slightly greater and blue to be slightly less than the current amount. For example, you could try a factor of 1.1 red and 0.9 blue.
- For tint midtones (pixels where the red value is between 65 and 190 or so), increase red slightly more than we did for tint shadows, and decrease blue slightly more than we did for tint shadows.
- For tint highlights (pixels where the red value is greater than 190 or so), increase red slightly less than we did for tint shadows, and decrease blue slightly less than we did for tint shadows.

Here's some structure to get you started…

```
def sepiaTint(picture):
      convert picture to greyscale – see below for the function from lab1
      loop through picture to tint the pixels
            if pixel is a tint shadow (value is below 65)
                  change red and blue by certain amounts
            if pixel is a tint midtone (value is between 65 and 190)
                  change red and blue by certain amounts
            if pixel is a tint highlight (value is greater than 190)
                  change red and blue by certain amounts
```

Handy built-in functions that you'll find helpful:

```
def grayscale(picture):
  for p in getPixels(picture):
     sum = getRed(p) + getGreen(p) + getBlue(p)
     intensity = sum / 3
     setColor(p, makeColor(intensity, intensity, intensity))

getPixels(picture)
getRed(pixel)
getBlue(pixel)
setRed(pixel, newRedValue)
setBlue(pixel, newBlueValue)
```

Finally, add code to make a picture using the `makePicture(file)` function, call the `sepiaTint(picture)` function on this picture, and show the picture using the function `show(picture)`. If you'd like, save your new picture into your lab2 folder using: `writePictureTo(picture, "lab2/filename.jpg")`

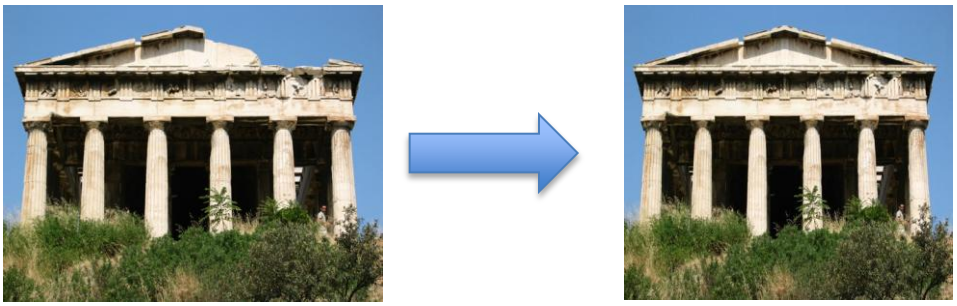
After loading the program, you should see your sepia tinted picture!

Just for kicks:

* Try changing the ranges of your tones in the conditions of your "if" statements to see how this affects your picture.

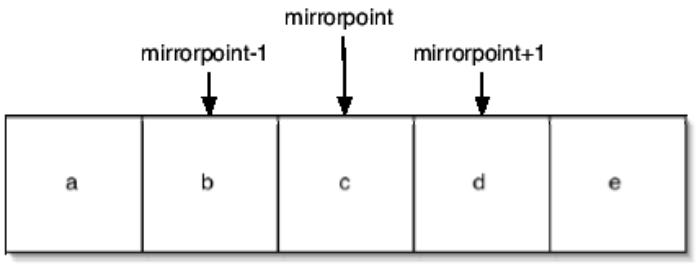
* Try setting your reds and blues by greater or smaller factors to see what this does to your picture.

Exercise 4: Mirroring!



The Big Picture:

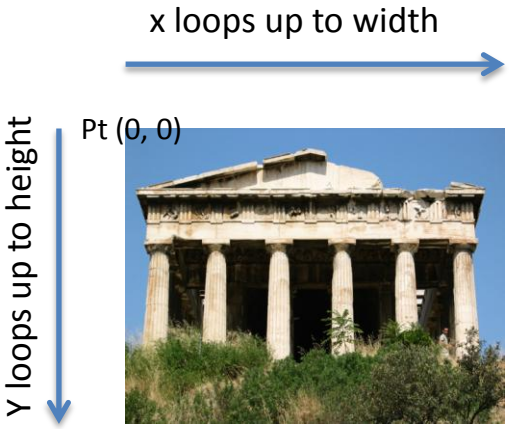
To mirror a picture, imagine slicing the picture down the middle and sticking a mirror on that dividing line. For each pixel on one side of the mirror, find its distance from the center (the mirrorpoint) and copy its color over to the point directly on the opposite side of the mirror, the same distance away. In the example below, pixel b would be copied to location d.



Find a small (under 800 x 800 pixels) picture you'd like to mirror-ify, save it to your lab2 folder and convert it to a picture using `makePicture(file)`.

Recall how to use **ranges** and **2 nested for-loops** to loop over an entire picture:

```
# y loops from 0 (inclusive) to the picture height (exclusive)
for y in range(0, picture height):
      # x loops from 0 (inclusive) to picture width (exclusive)
      for x in range(0, picture width):
            get the pixel at this x and y coordinate
            do something with the pixel
```



Exercise 4a

Use 2 nested for-loops to increase the red in every pixel of your picture. Show your picture at the end to test it!

Recall these useful functions:

```
getRed(pixel)
setRed(pixel, value)
```

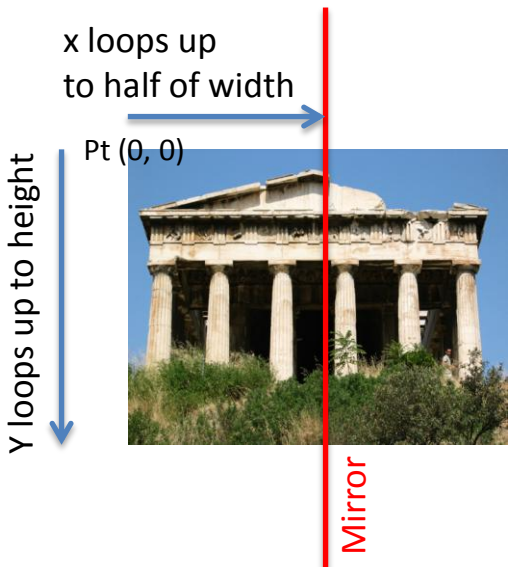
What if we only want to loop over the left half of the picture? For the x direction, we could start from 0 and loop up to the middle of the picture. The inner for loop (in blue) would change to look something like this:

```
for y in range(0, getHeight(picture)):
      for x in range(0, int(half of width)):
      # Note: max distance from middle must be an integer, hint: use
int(number)
      # xOffset represents distance from middle
            get the pixel at this x and y coordinate
            do something with the pixel
```

Exercise 4b

Try using this framework to increase the red component of only the pixels on the left half of your picture! Show your picture at The end to test it!

(see next page for useful functions)

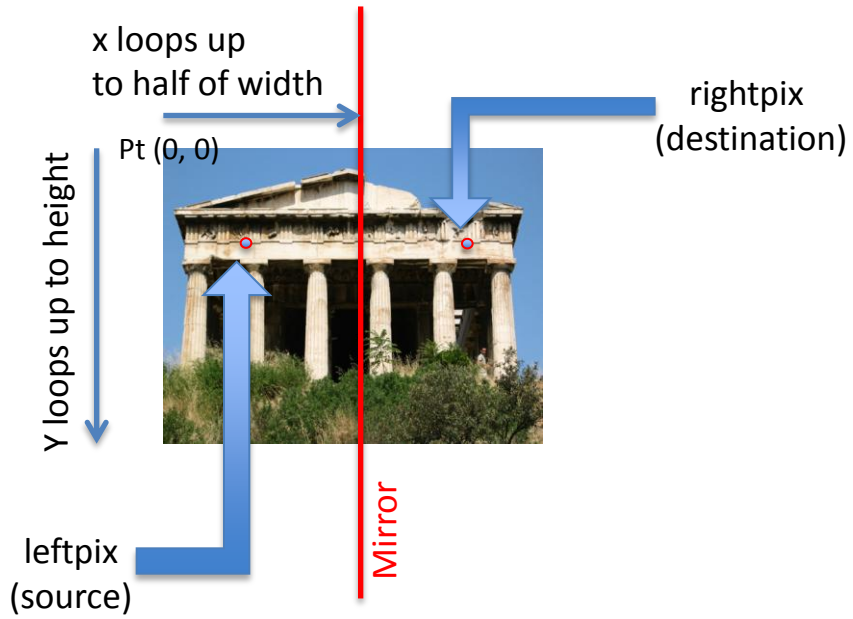


Handy built-in functions that you'll find helpful:

```
int(number) # rounds a number down to the nearest integer
getWidth(picture)
getHeight(picture)
getPixel(picture, x, y)
getColor(pixel)
setColor(pixel, newColor)
show(picture)
```

Exercise 4c

Modify the double-for loop you have written to copy all the pixels on the left side of the mirror to their corresponding position on the right side of the mirror.



STOP: See below only if you get stuck!

Here's some basic structure to help you.

```
def mirrorVertical(picture):
   width = getWidth(picture) – 1
   find half-width of picture, convert to integer, and assign to a variable
   for y in range(0, picture height):
      for x in range(0, half width):
            get pixel at the location (x, y) & assign to variable "leftpix"
            get pixel at the spot opposite of this one & assign to var.
"rightpix"
            set the color of rightpix to be the color of leftpix
```

Finally, add code to call the `mirrorVertical(picture)` function on your picture and remember to show the picture. If you'd like, save your new picture into your lab2 folder using: `writePictureTo(picture, "lab2/filename.jpg")`. After loading the program, you should see your mirrored picture!

Exercise 4d

Write a function `mirrorHorizontal(picture)` that mirrors your picture horizontally instead of vertically.

Bonus Exercises

- Write a function `rotate90cc(picture)` that rotates the picture 90 degrees counterclockwise.
- Write a function `rotate90c(picture)` that rotates the picture 90 degrees clockwise.
- Write a function `rotate180(picture)` that rotates the picture 180 degrees.
  - Now do it without using the function `rotate90(picture)`
- Write a function `removeRedEye(picture)` that takes in a picture of a person with red-eye and fixes it so that their eyes are black, brown, or blue.
- Write a function `jail(picture)` that takes in a picture of a person and puts that person behind vertical black bars. The bars should be evenly spaced.
- Write a function `window(picture)` that takes in a picture and adds a "window" in front of it so that vertical and horizontal black bars are added to the picture.
- Write a function `crop(picture, newPicture, x1, y1, x2, y2)` that takes in a picture, the upper left corner and lower right corner coordinates of the area you want to crop, and puts it into a new picture.