Lab:
Day 3

Exercises:

Copying

Cropping

Scaling

Projects: A Collage

Exercise 1: Copying

- Make a lab3 folder in your jes-4-3-nojava folder

Exercise 1(a): Copy into corner

- Define a new function named `copy(picture)` that takes a picture as input and copies that picture into the corner of a canvas, shows that canvas and then returns it.
    - Note: the canvas must be the same size or larger than the picture being copied
    - Reminder: a blank canvas is made by calling the function `makeEmptyPicture(width?, height?, white)`

Exercise 1(b): Copy into middle

- Define a new function `copyMidway(picture)` that takes a picture as input and copies that picture into the middle of a canvas, shows that canvas and returns that canvas.
    - Note: this time we need to start at a different (targetX, targetY) coordinate!

Exercise 1(c): Copy into a given location

- Define a new function named `copyInto(picture, destX, destY)` that takes a picture as input and copies that picture into the given location, shows that canvas and returns that canvas.

After you've fully created your functions, make a `picture` and call these functions at the bottom of your programming area to see your results.

Load your program. You should now see your picture!

(there are hints on the next page)

STOP

(if you find yourself finishing the labs quickly, you should not be reading this page)

Recall the structure of a double for loop…

```
for y in range(num1, num2):                  #traverse rows
  for x in range(num1, num2):                #traverse columns
    probably getPixel(picture, x, y)
     do something with that pixel
```

Here's the example of copy in pseudocode:

(this can also be used for copyMidway and copyInto -- the targetX and targetY will be different)

```
def copy(picture):
   initialize targetY
   for each sourceY in picture
      initialize targetX
      for each sourceX in picture
         find the picture's pixel
         find the color of that pixel & assign it to a variable
         find the target's pixel
         set the color of the target pixel to the color
         increment targetX
      increment targetY
   show the final result which should be on your canvas
   return the canvas
```

Exercise 2: Cropping

The Big Picture: Copy less of the picture into the canvas.

Write the function `crop` that takes in a `picture`, the starting coordinates `(startX, startY)` and a `newWidth` and `newHeight` that represents the dimensions of the picture we are copying over (or cropping) and copies part of the picture over onto a canvas, shows that canvas, and returns that canvas.

Note: we need to make sure the canvas is still as large as what we are copying into it.

```
def crop(picture, startX, startY, newWidth, newHeight):
```

- Two differences from copying:
  - The starting location for the original picture
  - And the range that we are copying!

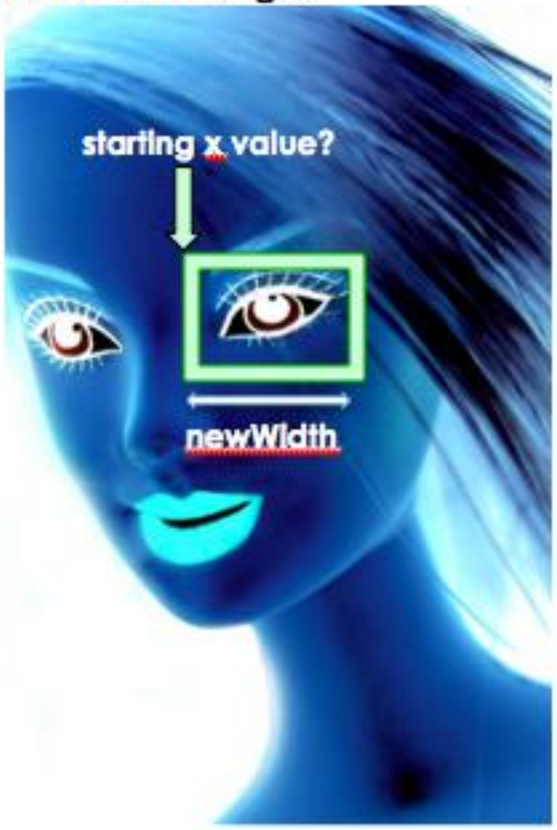
After you've fully created your function, make a `picture` and call this function at the bottom of your programming area. If you wanted to start at (10,20) and end up with a picture that is 50x100 (WxH) you make a call similar to the following:

```
crop(picture, 10, 20, 50, 100)
```

Load your program. You should now see your picture!

(there are hints on the next page)

Clue: new range?



- for the new x range:
  - Where do we want to start?
    - Given to us
  - Where do we want to end?
    - ???

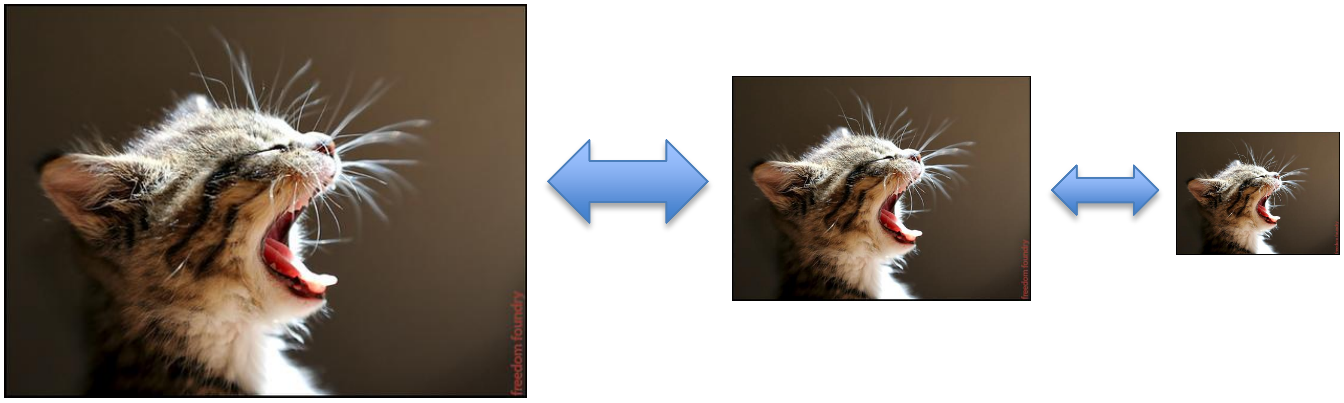
STOP

(if you find yourself finishing the labs quickly, you should not be reading this page)

Here’s an example in pseudocode:

```
def crop(picture, startX, startY, newWidth, newHeight):
   initialize targetY                  #start at 0
   for each sourceY in new Y range
      initialize targetX               #start at 0
      for each sourceX in new X range
        find the picture’s pixel
        find the color of that pixel & assign it to a variable
        find the target’s pixel
        set the color of the target pixel to the color
        increment targetX
     incremenet targetY
   show the final result which should be on your canvas
   return the canvas
```

Exercise 3: Scaling



Big Picture: Changing the size of an image by choosing a limited set of the pixels in the given picture to copy over!

The Strategy:

- If we want a smaller copy, we skip some pixels
    - We *sample* fewer pixels
    - Increment by 2!
- If we want a larger copy, we duplicate some pixels
    - We *over-sample* some pixels
    - Increment by 0.5!

a) Write the function `scaleLarger(picture)` that takes in a picture and scales the picture with the strategy given onto a canvas, shows that canvas and returns that canvas. The new height and width will be doubled; meaning a quadrupled area!

b) Write a function `scaleSmaller(picture)` that takes in a picture and scales the picture with the strategy given onto a canvas, shows that canvas, and returns that canvas. The new height and width will be halfed; meaning it will be ¼ the original area.

After you've fully created your function, make a `picture` and call this function at the bottom of your programming area by typing `scaleLarger(picture) or scaleSmaller(picture)`.

Load your program. You should now see your picture!

(there are hints on the next page)

STOP

(if you find yourself finishing the labs quickly, you should not be reading this page)

Here's some pseudo-code for scaleLarger to get you started…

```
def scaleLarger(picture):
  make a canvas that is twice the size of the original picture
  initialize sourceX = 0
  for targetX in a new range that is twice as large:
    initialize sourceY = 0
    for targetY in new range that is twice as large:
      get the color of the source picture’s pixel
      set the color of the canvas’s pixel to be that color
      increment sourceY by 0.5
    increment sourceX by 0.5
  show the final result which should be on your canvas
  return that canvas
```

Exercise 4: Collage: Copying and Cropping and Scaling, Oh My!

The Big Picture:

Could we do something to the pictures we copy in or crop or scale? Of course! We can call any of the function we have already made on the picture before we do the copying, cropping or scaling!

Note: these are going public! Spend some time on them!

- Create a collage
  - by coping your pictures into a canvas!
  - Be creative:
    - Scale, crop, change colors, grayscale, posterize, negative, etc.
    - Unity amidst variety
    - Use the same picture more than once
  - Save pictures with writePictureTo(picture,filename)

Note: you will have to keep track of the edge where you left off!

STOP

(if you find yourself finishing the labs quickly, you should not be reading this part)

Here’s an example in pseudocode:

```
def chromakey(source, background):
  for each y                                    #do the rows
    for each x                                  #do the columns
      get source pixel sourcePX
      if color of the sourcePX IS green    #???
        get background pixel’s color
        set color of sourcePX to be the background color
```

How to we check for green?

```
if(getRed(p)+getBlue(p)< factor*getGreen(p) and
   getGreen(p) >num):
```

Mess around with num and factor to get best possible result!

Bonus Exercises

- Write a function `notAllowed(picture)` that takes in a picture and creates a red circle with a cross through it. It should be around some action or object that is *not allowed.* Like so:

  You might need to take in additional arguments, such as a radius and a center of the picture.

- Write a function `halfTint(picture)` that takes in a picture and makes half of the picture more red, and half of the picture more blue

- Write a function `tripleTint(picture)` that takes in a picture and makes 1/3 of the picture more red, and 1/3 of the picture more blue and the last 1/3 of the picture more green.