Copying and Transforming Pictures

First, finding the min or max…

- Next homework asks you to write a function to find the darkest and lightest shade of grey in a picture
- Here is a similar example to find the value of the pixel with the largest red component

```
def findLargestRed(pict):
  largestSoFar = -1
  for p in getPixels(pict):
    r = getRed(p)
    if (r > largestSoFar):
        largestSoFar = r
  return largestSoFar
```

Moving pixels *across* pictures

- We've seen using index variables to track the pixel position we're working with in a picture.
- We can copy *between* pictures, if we keep track of:
  - The *source* index variables
    - Where we're getting the pixels *from*
  - The *target* index variables
    - Where we're putting the pixels *at*
- (Not really copying the pixels: Replicating their color.)

What can you do then?

- What can you do when copying from one picture to another?
  - Collages: Copy *several* pictures onto one
  - Cropping: You don't have to take the *whole* picture
  - Scaling: Make a picture smaller, or larger when copying it

Blank files in mediasources

- getMediaPath("7inX95in.jpg") gives you a JPEG canvas which prints out as 7x9.5 inches
  - Letter-sized page with 1 inch margins
- getMediaPath("640x480.jpg") gives a JPEG canvas at a common size: 640 pixels across by 480 pixels high

Copying pixels

- In general, what we want to do is to keep track of a sourceX and sourceY, and a targetX and targetY.
  - We *increment* (add to them) in pairs
    - sourceX and targetX get incremented together
    - sourceY and targetY get incremented together
  - The tricky parts are:
    - Setting values *inside* the body of loops
    - Incrementing at the *bottom* of loops

Copying Barb to a canvas

```
def copyBarb():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 1
  for sourceX in range(1,getWidth(barb)):
    targetY = 1
    for sourceY in range(1,getHeight(barb)):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

What's this naming something to itself?

- targetX = targetX + 1
- This isn't really naming something as itself
  - targetX + 1 is *evaluated*
    - It will result in the number after targetX
  - targetX = then sets the value of targetX
- The result is that targetX gets incremented by 1

Transformation = Small changes in copying

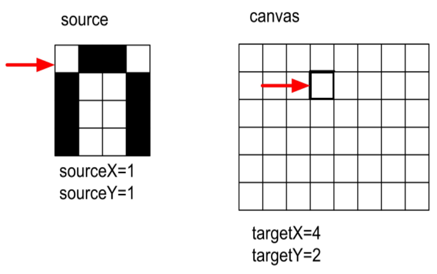
- Making relatively small changes in this basic copying program can make a variety of transformations.
  - Change the targetX and targetY, and you copy wherever you want
  - Cropping: Change the sourceX and sourceY range, and you copy only part of the program.
  - Rotating: Swap targetX and targetY, and you end up copying sideways
  - Scaling: Change the increment on sourceX and sourceY, and you either grow or shrink the image.

Copying into the middle of the canvas

```
def copyBarbMidway():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 100
  for sourceX in range(1,getWidth(barb)):
    targetY = 100
    for sourceY in range(1,getHeight(barb)):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

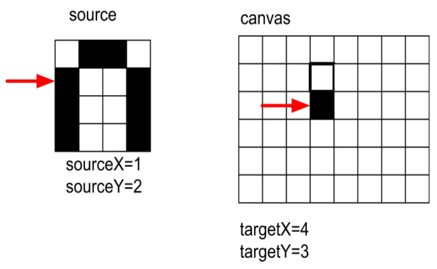
Copying: How it works

- Here's the initial setup:



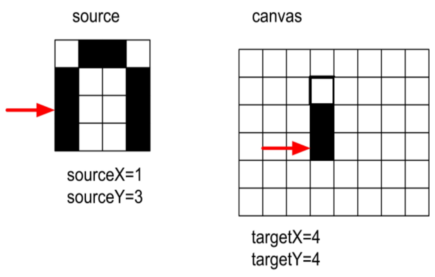
Copying: How it works 2

- After incrementing the sourceY and targetY once (whether in the **for** or via expression):



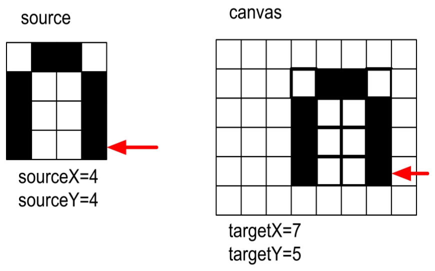
Copying: How it works 3

- After yet another increment of sourceY and targetY:
- When we finish that column, we increment sourceX and targetX, and start on the next column.



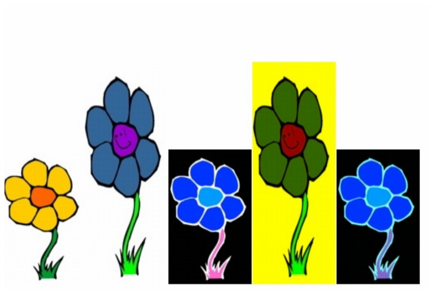
Copying: How it looks at the end

- Eventually, we copy every pixel



Making a collage

- Could we do something to the pictures we copy in?
  - Sure! Could either apply one of those functions *before* copying, or do something to the pixels *during* the copy.
- Could we copy more than one picture!
  - Of course! Make a collage!



```
def createCollage():
  flower1=makePicture(getMediaPath("flower1.jpg"))
  print flower1
  flower2=makePicture(getMediaPath("flower2.jpg"))
  print flower2
  canvas=makePicture(getMediaPath("640x480.jpg"))
  print canvas
  #First picture, at left edge
  targetX=1
  for sourceX in range(1,getWidth(flower1)):
    targetY=getHeight(canvas)-getHeight(flower1)-5
    for sourceY in range(1,getHeight(flower1)):
      px=getPixel(flower1,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  #Second picture, 100 pixels over
  targetX=100
  for sourceX in range(1,getWidth(flower2)):
    targetY=getHeight(canvas)-getHeight(flower2)-5
    for sourceY in range(1,getHeight(flower2)):
      px=getPixel(flower2,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
```

Exactly from book

```
#Third picture, flower1 negated
  negative(flower1)
  targetX=200
  for sourceX in range(1,getWidth(flower1)):
    targetY=getHeight(canvas)-getHeight(flower1)-5
    for sourceY in range(1,getHeight(flower1)):
      px=getPixel(flower1,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  #Fourth picture, flower2 with no blue
  clearBlue(flower2)
  targetX=300
  for sourceX in range(1,getWidth(flower2)):
    targetY=getHeight(canvas)-getHeight(flower2)-5
    for sourceY in range(1,getHeight(flower2)):
      px=getPixel(flower2,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  #Fifth picture, flower1, negated with decreased red
  decreaseRed(flower1)
  targetX=400
  for sourceX in range(1,getWidth(flower1)):
    targetY=getHeight(canvas)-getHeight(flower1)-5
    for sourceY in range(1,getHeight(flower1)):
      px=getPixel(flower1,sourceX,sourceY)
      cx=getPixel(canvas,targetX,targetY)
      setColor(cx,getColor(px))
      targetY=targetY + 1
    targetX=targetX + 1
  show(canvas)
  return(canvas)
```

Cropping: Just the face

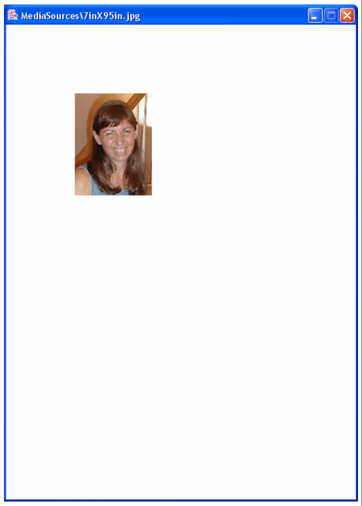
```
def copyBarbsFace():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  targetX = 100
  for sourceX in range(45,200):
    targetY = 100
    for sourceY in range(25,200):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      targetY = targetY + 1
    targetX = targetX + 1
  show(barb)
  show(canvas)
  return canvas
```

Scaling

- Scaling a picture (smaller or larger) has to do with *sampling* the source picture differently
  - When we just copy, we *sample* every pixel
  - If we want a smaller copy, we skip some pixels
    - We *sample* fewer pixels
  - If we want a larger copy, we duplicate some pixels
    - We *over-sample* some pixels

Scaling the picture down

```
def copyBarbSmaller():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  sourceX = 1
  for targetX in range(100,100+(getWidth(barb)/2)):
    sourceY = 1
    for targetY in range(100,100+(getHeight(barb)/2)):
      color = getColor(getPixel(barb,sourceX,sourceY))
      setColor(getPixel(canvas,targetX,targetY), color)
      sourceY = sourceY + 2
    sourceX = sourceX + 2
  show(barb)
  show(canvas)
  return canvas
```



Scaling Up: Growing the picture

- To grow a picture, we simply duplicate some pixels
- We do this by incrementing by 0.5, but only use the integer part.

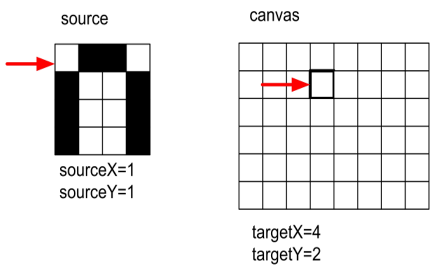
```
>>> print int(1)
1
>>> print int(1.5)
1
>>> print int(2)
2
>>> print int(2.5)
2
```

Scaling the picture up

```
def copyBarbLarger():
  # Set up the source and target pictures
  barbf=getMediaPath("barbara.jpg")
  barb = makePicture(barbf)
  canvasf = getMediaPath("7inX95in.jpg")
  canvas = makePicture(canvasf)
  # Now, do the actual copying
  sourceX = 1
  for targetX in range(10,10+(getWidth(barb)*2)):
    sourceY = 1
    for targetY in range(10,10+(getHeight(barb)*2)):
      color = getColor(getPixel(barb,int(sourceX),int(sourceY)))
      setColor(getPixel(canvas,targetX,targetY), color)
      sourceY = sourceY + 0.5
    sourceX = sourceX + 0.5
  show(barb)
  show(canvas)
  return canvas
```

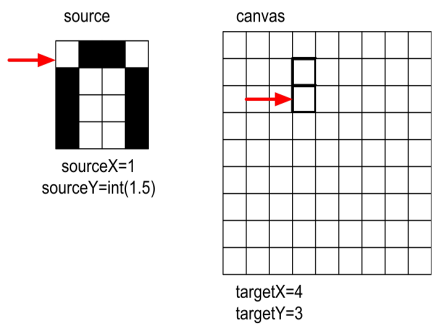
Scaling up: How it works

- Same basic setup as copying and rotating:



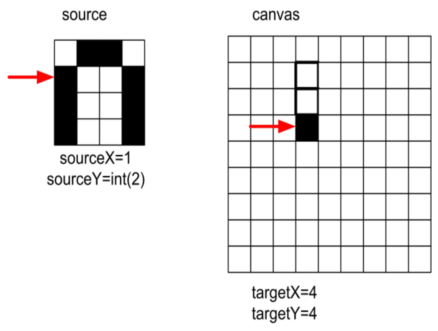
Scaling up: How it works 2

- But as we increment by *only 0.5*, and we use the **int()** function, we end up taking every pixel *twice.*
- Here, the blank pixel at (1,1) in the source gets copied twice onto the canvas.



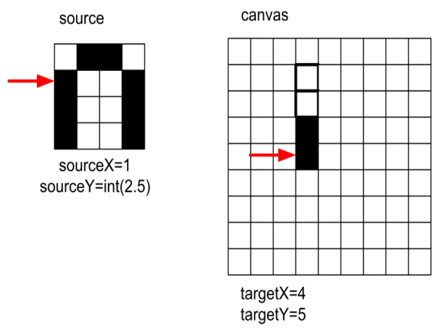
Scaling up: How it works 3

- Black pixels gets copied once…



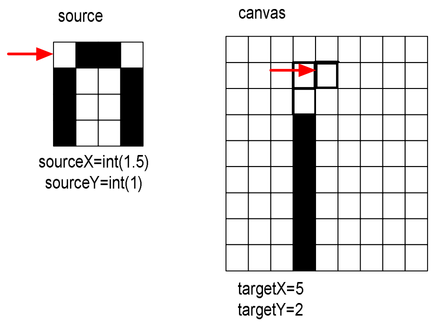
Scaling up: How it works 4

- And twice…



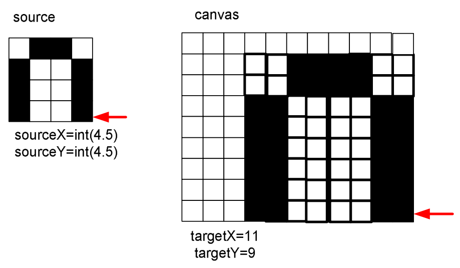
Scaling up: How it works 5

- The next “column” (x) in the source, is the *same* “column” (x) in the target.



Scaling up: How it ends up

- We end up in the same place in the source, but twice as much in the target.
- Notice the degradation:
  - Curves get “choppy”: Pixelated



Described in the text, but skipping here. Good things to try:

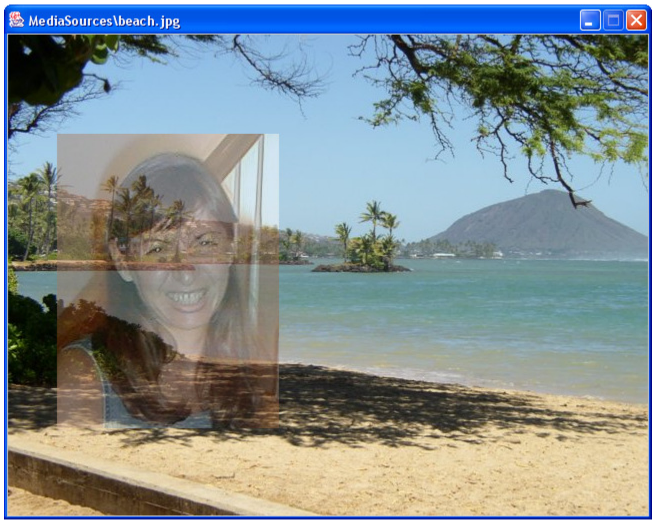
- Can you come up with general copy, rotate, copy, and scale functions?
  - Take input pictures and parameters
  - Return the canvas the correct transformation applied
- Also think about generalizing the transformations:
  - Scaling up and down by non-integer amounts
  - Rotating by something other than 90 degree increments

Blending Pictures

- Instead of copying from the source to the target, we can *combine* the source and target to create a new image
- Simple technique
  - Average the red, green, and blue from the source and target
  - Try putting Barb on the beach

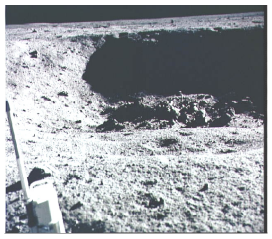
```
def averageBlending():
  # Set up the source and target pictures
  barb = makePicture(getMediaPath("barbara.jpg"))
  beach = makePicture(getMediaPath("beach.jpg"))
  sourceX = 1
  for targetX in range(50,50+(getWidth(barb))):
    sourceY = 1
    for targetY in range(100,100+(getHeight(barb))):
      barbPixel = getPixel(barb, sourceX, sourceY)                    # Get barb pixel
      redBarb = getRed(barbPixel)
      greenBarb = getGreen(barbPixel)
      blueBarb = getBlue(barbPixel)
      beachPixel = getPixel(beach, targetX, targetY)                  # Get beach pixel
      redBeach = getRed(beachPixel)
      greenBeach = getGreen(beachPixel)
      blueBeach = getBlue(beachPixel)
      color = makeColor((redBarb + redBeach)/2, (greenBarb + greenBeach) / 2 ,
(blueBarb + blueBeach) / 2)
      setColor(beachPixel, color)
      sourceY = sourceY + 1
    sourceX = sourceX + 1
  show(barb)
  show(beach)
  return beach
```

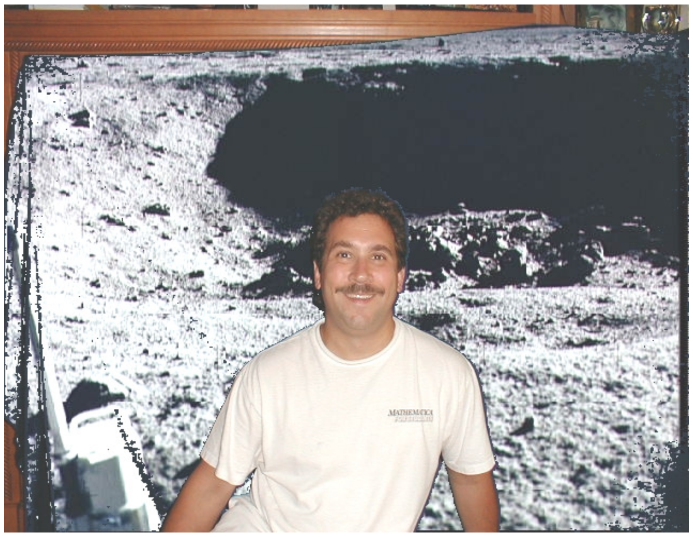
Blending through Averaging



Chromakey

- What the weather person does
- Pose in front of a blue or green screen
- Swap all “blue” or “green” for the background





Example Solution

```
def chromakey2(source,bg):
    for p in getPixels(source):
        if (getRed(p)+getGreen(p) < getBlue(p)):
            setColor(p,
              getColor(getPixel(bg,
                          getX(p),getY(p))))
    return source
```

Another way of saying the same thing

```
def chromakey(source,bg):
    # source should have something in front of blue, bg is the new
    background
    for x in range(1,source.getWidth()):
        for y in range(1,source.getHeight()):
          p = getPixel(source,x,y)
          # My definition of blue: If the redness + greenness < blueness
          if (getRed(p) + getGreen(p) < getBlue(p)):
              #Then, grab the color at the same spot from the new
                background
              setColor(p,getColor(getPixel(bg,x,y)))
    return source
```

Can I do this by masking in Photoshop?

- Of course!
  - How do you think Photoshop does it?
- But you can do it better, differently, faster, and for more kinds of pictures if *you* know how it works