CS 1124
Media Computation

Steve Harrison

Lecture 6.1 (September 29, 2008)

Today

- HW 4
- Bailing out of loops using return
- Drawing graphics
- Review of everything!

Batter up issues

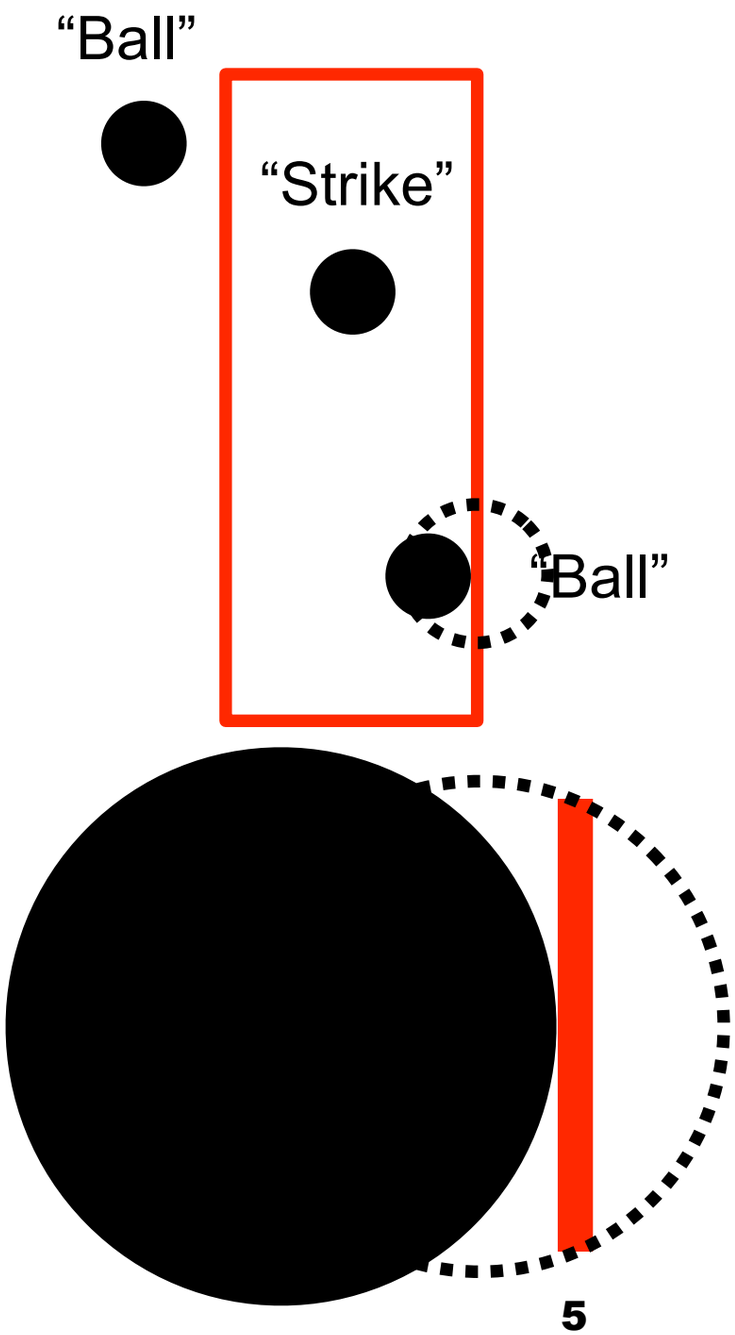
- How did you solve the puzzle?
- Here are some solutions ...

One solution

- How did you solve the puzzle?
- Lets count the black pixels in the strike zone ....
  - **if count == # of black pixels counted with MediaTools, then print “Strike”**

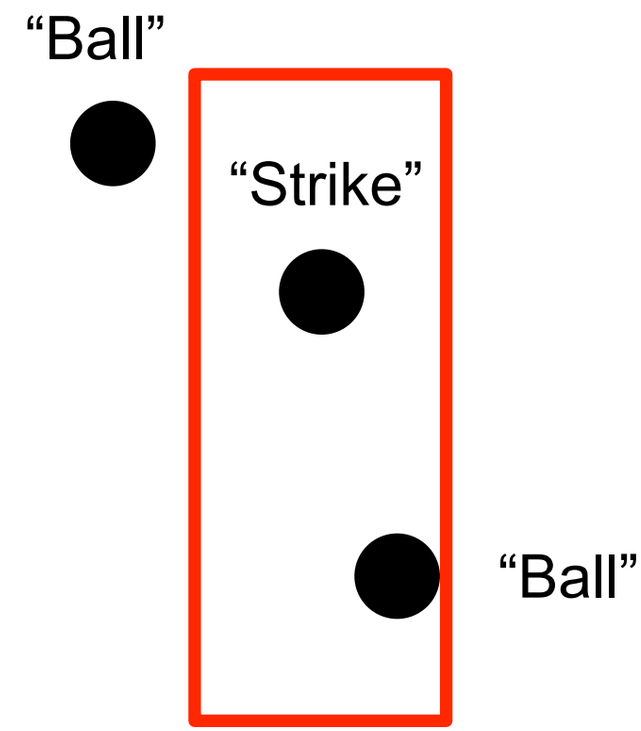
One “gotcha” and a solution

- How did you solve the puzzle?
- Notice that if there is any black touching the red lines then its a “ball”



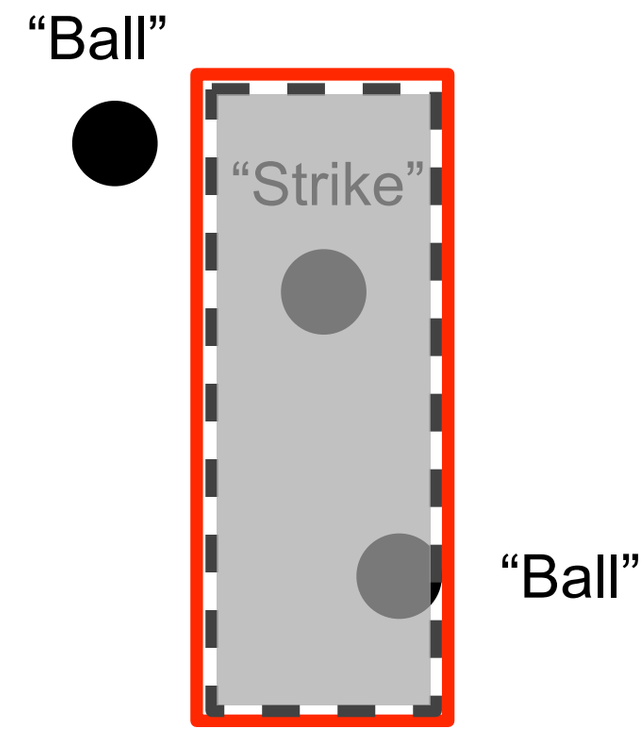
One “gotcha” and a solution

- How did you solve the puzzle?
- Notice that if there is any black touching the red lines then its a “ball”
- Therefore, a strike is:
  - **no black in lines just inside strike zone box**
  - **and any black inside that smaller rectangle**



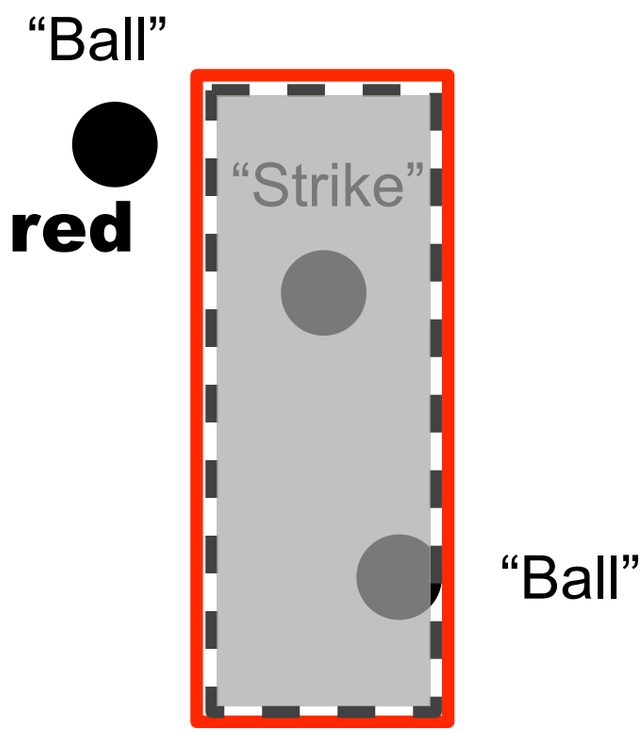
One “gotcha” and a solution

- How did you solve the puzzle?
- Notice that if there is any black touching the red lines then its a “ball”
- Therefore, a strike is:
  - **no black in lines just inside strike zone box**
  - **and any black inside that smaller rectangle**



One “gotcha” and a solution

- So the psuedocode is:
  - **search for black one pixel in from red line:**
    - top white line (yUL +1)
    - bottom white line (yLR - 1)
    - left white line (xUL + 1)
    - right white line (xLR - 1)
    - If any black then “ball” and DONE
  - **search for black in rectangle two in from red line**
    - for x in range(xUL+2, xLR-2)
    - for y in range(yUL+2, yLR-2)
    - If any black then “strike” and DONE



Batter up issues

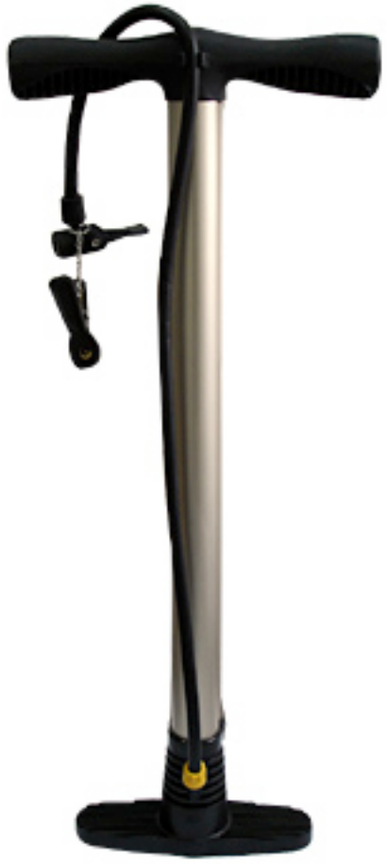
- Other solutions?

Today

- HW 4
- Bailing out of loops using return
- Drawing graphics
- Review of everything!

Today

- HW 4
- Bailing out of loops using return



One More thing ... Using return to bail out of loops

```
def findFirstBlackPixel(picture, xUL, yUL, xLR, yLR):
  for x in range(xUL, xLR):
    for y in range( yUL, yLR):
      px = getPixel(picture, x, y)
      red = getRed(px)
      green = getGreen(px)
      blue = getBlue(px)
      if (red < 2 ) and (green < 2) and (blue < 2):
        return “strike”
  return “ball”
```

So how would you call this function?

Lets say you want the x,y

```
def findFirstBlackPixel(picture):
  for x in range(1, getWidth(picture)):
    for y in range(1, getHeight(picture)):
      px = getPixel(picture, x, y)
      red = getRed(px)
      green = getGreen(px)
      blue = getBlue(px)
      if (red < 2 ) and (green < 2) and (blue < 2):
        return px
```

Why return the pixel?

Using the returned value

- return px

```
firstBlackPixel = findFirstBlackPixel( picture )
firstBlackPixelXLocation = getX( firstBlackPixel )
firstBlackPixelYLocation =  getY( firstBlackPixel )
```

- return x, y

```
“x,y = findFirstBlackPixel( picture )” is not valid Python
```

- return [x,y]
  - **is sequence**

```
pxlLocation = findFirstBlackPixel( picture )
firstBlackPixelXLocation = pxlLocation[0]
firstBlackPixelYLocation = pxlLocation[1]
```

- Any other?

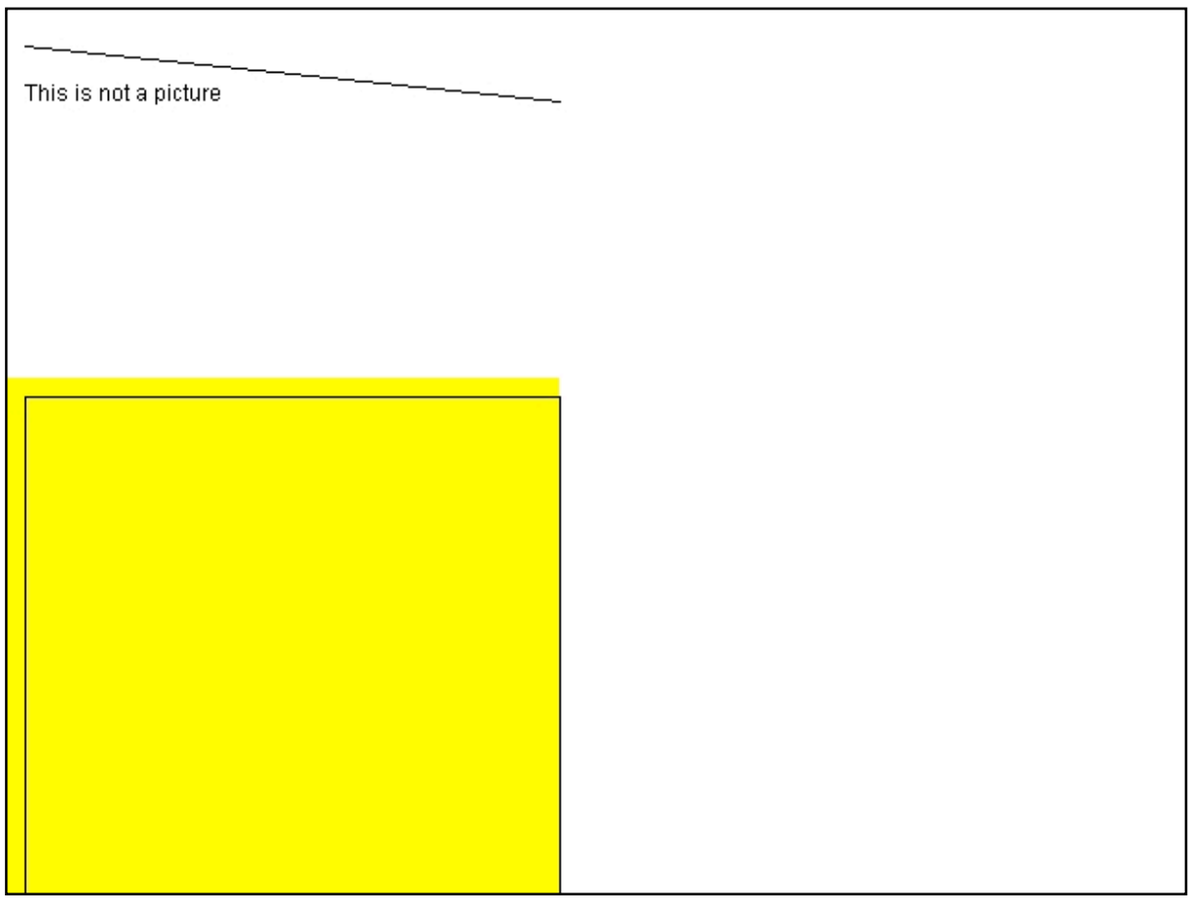
Today

- HW 4
- Bailing out of loops using return
- Drawing graphics
- Review of everything!

Example picture

```
def littlepicture():
  canvas=makePicture(getMediaPath("640x480.jpg"))
  addText(canvas,10,50,"This is not a picture")
  addLine(canvas,10,20,300,50)
  addRectFilled(canvas,0,200,300,500,yellow)
  addRect(canvas,10,210,290,490)
  return canvas
```

Notice that these draw outside the canvas without giving an error !



Vector-based representations can be smaller

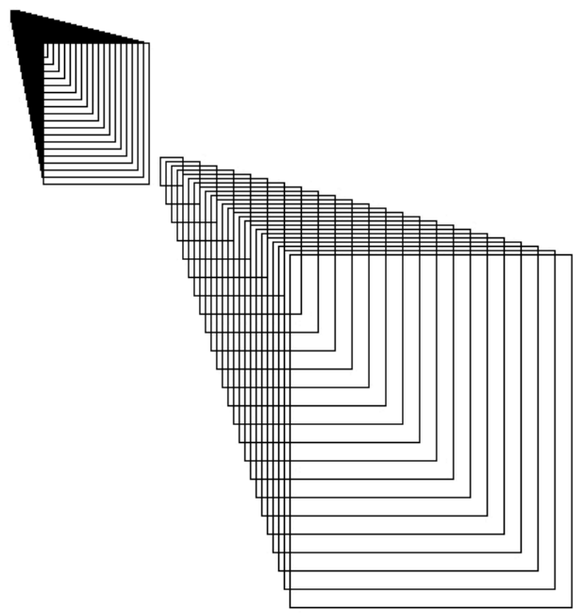
- Vector-based representations can be much smaller than bit-mapped representations
  - **Smaller means faster transmission (Flash and Postscript)**
  - **If you want all the detail of a complex picture, no, it's not.**

But vector-based has more value than that

- Imagine that you're editing a picture with lines on it.
  - **If you edit a bitmap image and extend a line, it's just more bits.**
    - There's no way to really realize that you've *extended* or *shrunk* the line.
  - **If you edit a vector-based image, it's possible to just change the specification**
    - Change the numbers saying where the line is
    - Then it *really is* the same line
- That's important when the picture drives the creation of the product, like in automatic cutting machines

And another

```
def coolpic2():
  canvas=makePicture(getMediaPath("640x480.jpg"))
  for index in range(25,1,-1):
    addRect(canvas,index,index,index*3,index*4)
  addRect(canvas,100+index*4,100+index*3,index*8,index*10)
show(canvas)
return canvas
```



Why do we write programs?

- Could we do this in Illustrator? Maybe
  - **I'm sure that you can, but you need to know how.**
  - **Illustrator is probably better, but still need to learn.**
- Could I teach you to do this in Photoshop? Maybe
  - **Might take a lot of demonstration**
- But this program is an *exact* definition of the process of generating this picture
  - **It works for anyone who can run the program, without knowing Photoshop**

We write programs to encapsulate and communicate process

- If you can do it by hand, do it.
- If you need to teach someone else to do it, consider a program.
- If you need to explain to lots of people how to do it, definitely use a program.
- If you want lots of people to do it without having to teach them something first, definitely use a program.

Drawing Graphics

- Referring to individual pixels by x,y location
- Drawing graphics by changing lots of pixels
- Graphics functions that are built in to JES
- Programmed graphics

- Each pixel knows its x,y position
- range() returns a list of numbers (not pixels)
- we can use range() to define which x,y pixels are interesting

- Works, but tedious & slow

- addText(), addRect(), etc.

-Vector graphics take less space
- & can be changed easily
-Really, small special graphics recipes
-Created by modifying canvas

Today

- HW 4
- Bailing out of loops using return
- Drawing graphics
- Review of everything!

Brief Review of Everything We've Learned in the Last Month

- What does this do?
- And how does it work?

```
def function(picture):
  for pixel in getPixels(picture):
    setRed(pixel,0)
```

```
def function(picture):
  for pixel in getPixels(picture):
    setRed(pixel,0)
```

Removes the red from every pixel

```
def function(picture):
  noRed = 0
  for pixel in getPixels(picture):
   pxlGreen = getGreen(pixel)
   pxlBlue = getblue(pixel)
   newColor = makeColor( noRed, pxlGreen, pxlBlue )
    setColor(pixel, newColor)
```

```
def function(picture):
  noRed = 0
  for pixel in getPixels(picture):
   pxlGreen = getGreen(pixel)
   pxlBlue = getblue(pixel)
   newColor = makeColor( noRed, pxlGreen, pxlBlue )
    setColor(pixel, newColor)
```

SAME THING -- MORE CODE

Removes the red from every pixel

```
def function(picture):
  for px in getPixels(picture):
    red=getRed(px)
    green=getGreen(px)
    blue=getBlue(px)
    negColor=makeColor(255-red,255-green,255-blue)
    setColor(px,negColor)
```

```
def function(picture):
  for px in getPixels(picture):
    red=getRed(px)
    green=getGreen(px)
    blue=getBlue(px)
    negColor=makeColor(255-red,255-green,255-blue)
    setColor(px,negColor)
```

Turns every pixel to negative of self

```
def function(picture):
  for p in getPixels(picture):
    value = getRed(p)
    setRed(p, value * 0.5)
```

```
def function(picture):
  for p in getPixels(picture):
    value = getRed(p)
    setRed(p, value * 0.5)
```

Decreases the red in every pixel by 1/2

```
def function(picture):
  for x in range(1, getWidth(picture)):
    for y in range(1, getHeight(picture)):
      px = getPixel(picture, x, y)
      value = getRed(px)
      setRed(px, value * 1.1)
```

```
def function(picture):
  for x in range(1, getWidth(picture)):
    for y in range(1, getHeight(picture)):
      px = getPixel(picture, x, y)
      value = getRed(px)
      setRed(px, value * 1.1)
```

Increases the red from every pixel by 10%

```
def function():
  # 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 = 45
  for targetX in range(100,100+((200-45)/2)):
    sourceY = 25
    for targetY in range(100,100+((200-25)/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
```

Makes a new smaller picture of barb by getting every other pixel

```
def function():
  # 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 = 45
  for targetX in range(100,100+((200-45)/2)):
    sourceY = 25
    for targetY in range(100,100+((200-25)/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
```

```
def function():
  # 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 = 45
  for targetX in range(100,100+((200-45)*2)):
    sourceY = 25
    for targetY in range(100,100+((200-25)*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
```

Makes a new larger picture of barb by duplicating every pixel

```
def function():
  # 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 = 45
  for targetX in range(100,100+((200-45)*2)):
    sourceY = 25
    for targetY in range(100,100+((200-25)*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
```

```
def function( p1 ):
  for p2 in getPixels( p1):
    setRed( p2 ,0)
  return p1
```

```
def function( p1 ):
  for p2 in getPixels( p1):
    setRed( p2 ,0)
  return p1
```

Removes the red from every pixel

```
def function( param1, param2) :
    if (param1 < param2) :
        return param1
    else :
        return param2
```

```
def function( param1, param2) :
   if (param1 < param2) :
      return param1
   else :
      return param2
```

Return lesser parameter

What does this do & what does this print?

```
def function(picture):
  columns = 0
  rows = 0
  for x in  range(1, getWidth(picture) ):
    columns = columns + 1
    for y in  range(1, getHeight(picture) ):
      rows = rows + 1
      pxl = getPixel(picture,x,y)
      value = getRed(pxl)
      setRed(pxl, value * 0.5)
print columns, rows
```

What does this do & what does this print?

```
def function(picture):
  columns = 0
  rows = 0
  for x in  range(1, getWidth(picture) ):
    columns = columns + 1
    for y in  range(1, getHeight(picture) ):
      rows = rows + 1
      pxl = getPixel(picture,x,y)
      value = getRed(pxl)
      setRed(pxl, value * 0.5)
  print columns, rows
```

# of columns processed (one less than total)

What does this do & what does this print?

```
def function(picture):
    columns = 0
    rows = 0
    for x in  range(1, getWidth(picture) ):
        columns = columns + 1
        for y in  range(1, getHeight(picture) ):
            rows = rows + 1
            pxl = getPixel(picture,x,y)
            value = getRed(pxl)
            setRed(pxl, value * 0.5)
    print columns, rows
```

of (rows * columns) processed

```
def function( variable1 ) :
    variable1 = makePicture( variable1 )
    one = 4
    four = 2
    for variable3 in getPixels( variable1 ) :
        if (getRed( variable3) < 127) :
            variable4 = variable4 + four
    else :
        variable2 = variable2 + one
if (variable2 > variable4) :
    return variable4
else :
    return variable1
```

```
def function( variable1 ) :
    variable1 = makePicture( variable1 )
    one = 4
    four = 2
    for variable3 in getPixels( variable1 ) :
        if (getRed( variable3) < 127) :
            variable4 = variable4 + four
        else :
            variable2 = variable2 + one
    if (variable2 > variable4) :
        return variable4
    else :
        return variable1
```

Count pixels with less red, return count of ???

What was wrong with that last function?

- It returned two different kinds of things - a number or a picture

```
if (variable2 > variable4) :
  return variable4
 else :
  return variable1
```

- The variable names are not representative
- variable1 is a filename then a picture
- Variables “one” and “four” are misleading
- Both variable2 and variable4 increment but are not initialized. (This would prevent running.)
- There are no comments

Study advice

- Re-read the book
- Try more of the recipes. Vary them.
  - **Take chances**
  - **make mistakes**
  - **learn from them!**

Coming Attractions

- Wednesday
  - **Exam 1 on visual progamming**
    - multiple choice
    - write programs (list of functions provided)
    - closed book
    - closed computer
  - **on-line study quiz**
- Friday
  - **Tom Igoe @ 4:30 in Squires Studio Theater**
- Next Monday
  - **read chapter 6**
  - **online quiz due 10:00 AM**