CS Kickstart – Day 2

Defining Functions
Conditionals (If Statements)
Double For Loops
Review: Parts of a Function

* Function Name
* Input Values (optional)
  * Zero, one, or many
  * Sometimes called “parameters”
* Function Body
  * Indented
def decreaseRed(picture):
    for p in getPixels(picture):
        value = getRed(p)
        setRed(p, value * 0.5)
What do functions do?
What do functions do?
def square(x):
    return x*x

y = square(4)
print y

What would print?
def \texttt{square}(x):
    \text{return } x \times x

y = \texttt{square}(4)
\text{y = output of function “square”}
\text{called with input value 4}
\text{y = 16}
def decreaseRed(picture):
    for p in getPixels(picture):
        value = getRed(p)
        setRed(p, value * 0.5)
    return picture
def decreaseRed(picture):
    for p in getPixels(picture):
        value = getRed(p)
        setRed(p, value * 0.5)
    return picture

result = decreaseRed(myPicture)
result = output

>>> result = decreaseRed(myPicture)
>>> show(result)
How ‘if’ Statements Work

If condition:
BODY #1 ← Only IF the condition is true
THEN BODY #1 is executed

x = 2
If x < 2:
    print “Body 1”

* Is something printed?
If condition:
  BODY #1
else:
  BODY #2

Only IF the condition is true THEN BODY #1 is executed
Otherwise (the condition is false) BODY #2 is executed

x = 2
If x < 2:
  print “Body 1”
else:
  print “Body 2”

* What is printed?
Conditions need to evaluate to **true** or **false**.

Here are other **conditions** you can use:

* Test equality with ‘==‘
  * Ex: 4 == 5
* Text not equal with ‘!=‘
  * Ex: 4 != 5
* Test comparisons >, >=, <, <=
  * and, or
if \ x < 10:  
    print "small"

if \ x >= 10 \ and \ x < 20:  
    print "medium"

if \ x >= 20 \ and \ x < 30:  
    print "large"
Let’s Make Barbara a Redhead

Why can’t we use our increaseRed function?
def turnRed():
    make a picture using a file
    for each pixel in the picture:
        figure out the color of the pixel
        if the color is close to the brown color in her hair:
            increase the redness of this pixel
    show the picture
    return the picture
def turnRed():
    make a picture using a file
    for each pixel in the picture:
        figure out the color of the pixel
        if the color is close to the brown color in her hair:
            increase the redness of this pixel
    show the picture
    return the picture
Let’s do exercise #1!

If you finish exercise #1, feel free to move on to exercise #2. In a minute, we will go over hints for how to complete exercise #2.
Posterizing:
Reducing the range of colors
Posterizing: How we do it

- Range of colors maps to a single color
- If statements to find which range
- End result: many colors → few colors
def posterize(picture):
    for each pixel in the picture:
        get the red value of the pixel and call it redValue
        if redValue is less than 64:
            set red of pixel to be 31
        if redValue is between 64 and 128:
            set red of pixel to be 95
        if (redValue > 63 and redValue < 128)
Exercise #2

If you finish exercise #2, you may move on to #3. We will break in a minute to talk about hints for exercise #3.
Exercise #3: Sepia-toned prints
Generating sepia-toned prints

- Yellowish tint that we associate with older photographs.
- Can’t just increase the amount of yellow.
- Range of colors converted to other colors.
  - We can create such conversions using if.
def sepiaTint(picture):
    Convert image to greyscale
    Loop through pixels to tint each pixel
        find red and blue values of pixel
        tint shadows
        tint midtones
        tint highlights
    set new pixel color values for red and blue
Double For loops and Ranges

This will help you complete exercise #4
Accessing Each Pixel

How do we access pixels?

for pixel in getPixels(picture): ➡️ seems like magic
do something to pixel

Is there another way?
A Picture is a *matrix* of pixels

- A continuous line is an *array*
  - 1 dimension
- Pictures have 2 dimensions
  - Height
  - Width
- Our array needs 2 dimensions
  - a *matrix*
Referencing a matrix

We talk about positions in a matrix as \((x,y)\), or (horizontal, vertical).

Element \((1,0)\) in the matrix at left is the value 12.

What is the value of element \((0,2)\)?
How to access each pixel

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>12</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

- **x** loops up to width
- **Y** loops up to height

Pt (0, 0)
We’ll have to use *nested loops*: One to walk the width, the other to walk the height.

for each y position: \(\text{for each row}\)
for each x position: \(\text{for each spot in the row}\)
do something to the pixel at \((x, y)\)
for each y position:AZE “for each row”
for each x position:AZE “for each space in row”
mark pixel (x,y)
Using ranges

for i in range(0, 10):
    print i

Breaking it down:

1. i is set to 0
2. print i
3. i is then set to 1
4. print i
5. etc...
6. When i is 10, stop
Introducing the function range

* Range returns a sequence between its first two inputs

```python
>>> print range(0, 4)
[0, 1, 2, 3]
```
```python
>>> print range(-1, 3)
[-1, 0, 1, 2]
```
```python
>>> print range(3)
[0, 1, 2]
```
for each y position: \(\gets\) “for each row”
for each x position: \(\gets\) “for each spot in the row”
do something to the pixel at \((x, y)\)

**VS**

for y in range(0, height):
    for x in range(0, width):
        do something to the pixel at \((x, y)\)
Using Ranges

for y in range(0, getHeight(picture)):
    for x in range(0, getWidth(picture)):
        pixel = getPixel(picture, x, y)
        setRed(pixel, 5)

VS

for pixel in getPixels(picture):
    pixel = getPixel(picture, x, y)
    setRed(pixel, 5)
for y in range(1, 3):
    for x in range(1, 4):
        pixel = getPixel(picture, x, y)
        setRed(pixel, 5)
def posterize(picture):
    for y in range(70,168):
        for x in range(56,190):
            get the red value of the pixel and call it \texttt{redValue}
            if \texttt{redValue} is less than 64:
                set red of pixel to be 31
            if \texttt{redValue} is between 64 and 128:
                set red of pixel to be 95

What will the Posterizing function do now?
For each pixel in the range covering the left half
  * Get distance from the center line
  * `currentColor = Get color of the pixel`
  * Set pixel of same distance from center on right side to the `currentColor`
For each pixel in the range covering the left half

- Get distance from the center line
- \texttt{currentColor} = Get color of the pixel
- Set pixel of same distance from center on right side to the \texttt{currentColor}
Complete the rest of the lab!

If you have extra time, move on to the extra exercises or experiment further with results from exercises #1 to #4.

WE WILL STOP 10 MINUTES EARLY TO SUBMIT YOUR WORK