



COMP2720: Automating Tools for new Media

Referencing pixels directly by
index number



Learning objectives of this lecture

- Referencing pixels by x and y coordinates
- Manipulating only a certain area within a picture
- Use of the `range` function
- Loops within loops

Remember that pixels are in a *matrix*

- Matrices have two dimensions: A *height* and a *width*
- We can reference any element in the matrix with (x,y) or (horizontal, vertical)
 - We refer to those coordinates as *index numbers* or *indices*
- We sometimes want to know where a pixel is, and `getPixels` doesn't let us know that.



Tuning our color replacement

- If you want to get more of Barbara's hair, just increasing the threshold doesn't work.
 - Wood behind becomes within the threshold value.
- How could we do it better?
 - Lower our threshold, but then miss some of the hair.
 - Work only within a range within the picture...

Introducing the function range

- The Python command `range` returns a sequence between its first two inputs, possibly using a third input as the increment (step).
- `range` with only one input returns a sequence starting with 0.

```
>>> print range(1,4)
[1, 2, 3]
>>> print range(-1,3)
[-1, 0, 1, 2]
>>> print range(1,10,2)
[1, 3, 5, 7, 9]
>>> print range(5)
[0, 1, 2, 3, 4]
```

That thing in [] is a sequence (list)

```
>>> a=[1,2,3]
```

```
>>> print a
```

```
[1, 2, 3]
```

```
>>> a = a + 4
```

An attempt was made to call a function with a parameter of an invalid type

```
>>> a = a + [4]
```

```
>>> print a
```

```
[1, 2, 3, 4]
```

```
>>> a[0]
```

```
1
```

We can assign names to sequences, print them, add sequences, and access individual pieces of them.

We can also use for loops to process each element of a sequence.



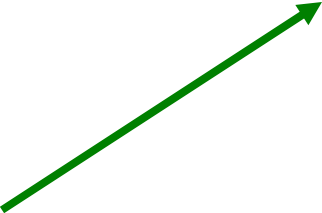
We can use range to generate index numbers

- We'll do this by working the range from 1 to the height, and 1 to the width
- But we'll need more than one loop.
 - Each for loop can only change one variable, and we need two for a matrix

Working the pixels by number

- To use range, we'll have to use *nested loops*
 - One to walk the width, the other to walk the height

```
def increaseRed2(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)
```



Bug Alert:

Be sure to watch your blocks carefully!

What's going on here?

The first time through the first loop, x is the name for 1.

We'll be processing the first column of pixels in the picture.

```
def increaseRed2(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)
```

Now, the inner loop

Next, we set y to 1.
We're now going to
process each of the
pixels in column 1.

```
def increaseRed2(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)
```

Process pixel (1,1)

With $x = 1$ and $y = 1$,
we get the leftmost
pixel and increase its
red by 10%

```
def increaseRed2(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)
```

Next pixel

Next we set y to 2 (next value in the sequence *range(1,getHeight(picture))*)

```
def increaseRed2(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)
```

Process pixel (1,2)

x is still 1, and now y is 2, so increase the red for pixel (1,2)

```
def increaseRed2(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)
```

We continue along this way, with y taking on every value from 1 to the height of the picture.

Finally, next column

Now that we're done with the loop for y, we get back to the for loop for x.

x now takes on the value 2, and we go back to the y loop to process all the pixels in the column x=2.

```
def increaseRed2(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)
```

Replacing colors in a range

Get the
range using
MediaTools



```
def turnRedInRange():  
    brown = makeColor(57,16,8)  
    file=r"F:\comp2720\pics\barbara.jpg"  
    picture=makePicture(file)  
    for x in range(70,168):  
        for y in range(56,190):  
            px=getPixel(picture,x,y)  
            color = getColor(px)  
            if distance(color,brown)<50.0:  
                redness=getRed(px)*1.5  
                setRed(px,redness)  
    show(picture)  
    return(picture)
```

Walking this code

- Like last time:
 - Don't need input
 - Same color we want to change
 - Same file
- Make a picture

```
def turnRedInRange():  
    brown = makeColor(57,16,8)  
    file=r"F:\comp2720\pics\barbara.jpg"  
    picture=makePicture(file)  
    for x in range(70,168):  
        for y in range(56,190):  
            px=getPixel(picture,x,y)  
            color = getColor(px)  
            if distance(color,brown)<50.0:  
                redness=getRed(px)*1.5  
                setRed(px,redness)  
    show(picture)  
    return(picture)
```

The nested loop

- Use **MediaTools** to find the rectangle where most of the hair is that we want to change

```
def turnRedInRange():
    brown = makeColor(57,16,8)
    file=r"F:\comp2720\pics\barbara.jpg"
    picture=makePicture(file)
    for x in range(70,168):
        for y in range(56,190):
            px=getPixel(picture,x,y)
            color = getColor(px)
            if distance(color,brown)<50.0:
                redness=getRed(px)*1.5
                setRed(px,redness)
    show(picture)
    return(picture)
```

Scanning for brown hair

- We're looking for a close-match on hair color, and increasing the redness

```
def turnRedInRange():  
    brown = makeColor(57,16,8)  
    file=r"F:\comp2720\pics\barbara.jpg"  
    picture=makePicture(file)  
    for x in range(70,168):  
        for y in range(56,190):
```

```
            px=getPixel(picture,x,y)  
            color = getColor(px)  
            if distance(color, brown) < 50.0:  
                redness=getRed(px)*1.5  
                setRed(px,redness)
```

```
            show(picture)
```

```
        return(picture)
```

Similar to scanning whole picture

**We could raise threshold now.
(Why?...)**

Could we do this without nested loops?

- Yes, but only with a complicated if statement
- Moral: Nested loops are common for 2D data

```
def turnRedInRange2():  
    brown = makeColor(57,16,8)  
    file=r"F:\comp2720\pics\barbara.jpg"  
    picture=makePicture(file)  
    for p in getPixels(picture):  
        x = getX(p)  
        y = getY(p)  
        if x >= 70 and x < 168:  
            if y >=56 and y < 190:  
                color = getColor(p)  
                if distance(color,brown)<100.0:  
                    redness=getRed(p)*2.0  
                    setRed(p,redness)  
    show(picture)  
    return picture
```



What to do now...

- Finish reading chapter 3 in the text book.
- Read sections 4.1 and 4.4 in the text book.
- Read assignment 1 specifications and start working on it.
- Read portfolio specifications.
- Labs have started – print and read lab specifications.