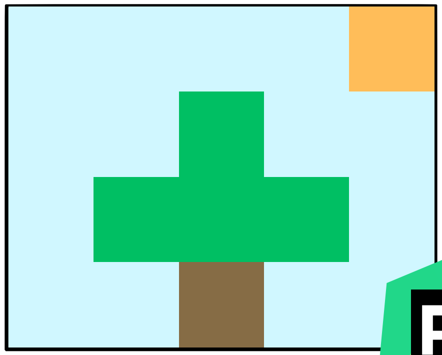


Images + PyCharm

by Ecy!

Stanford | ENGINEERING
Computer Science

Ecy King, CS106A, 2023

Housekeeping

- **Assignment 1, Bit is due this Friday, July 7th at 11:59 pm**
  - with Grace Period until Saturday, July 8th at 11:59 pm
- **Second section happening this week**
- **Coding can be difficult, but rewarding, keep at it!**

Photo Credits: Canva

Note on Style

- **Style is an important part of CS106A**
  - Descriptive variable names
  - Decomposition can be super duper useful
  - Write inline and function header comments
  - Have good formatting (spacing)

Style guidelines linked here!

Today

- **Recap Images**
  - Image functions, pixels
  - Double for-loop
  - Code demo

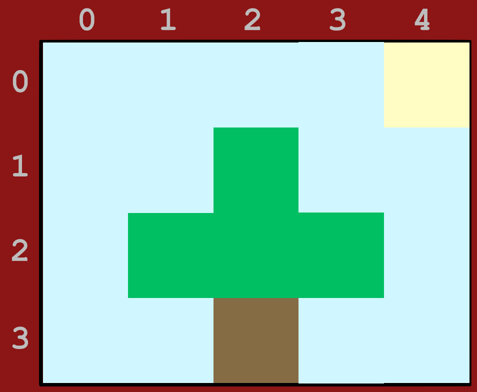
- **Look at New Functionality**
  - How do we make new, blank images?
  - How do we make two pixels the same?

- **Intro PyCharm**
  - How will we use PyCharm?
  - How can we run a program in PyCharm?
  - How can make a "bluescreen"?

Image Recap

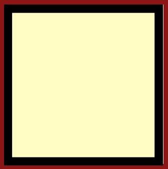
Images

Images are made of pixels that we can loop over with their x, y coordinates. We can load image files into variables using Simple Image.



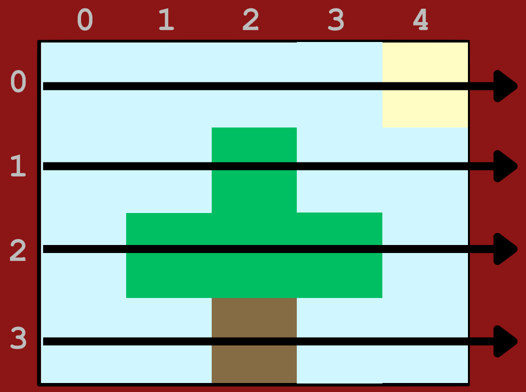
Pixels

Pixels have red, green, and blue attributes. We can grab a pixel at x, y in an image with the get_pixel() function.



Double For Loops

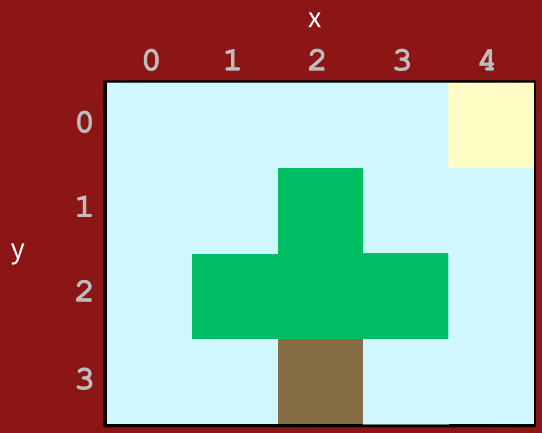
Double (or nested) for loops get us all possible x, y combos and thus, all possible coordinates. Thus, we can access every single pixel.



Images

Images are made of pixels that we can loop over with their x, y coordinates.

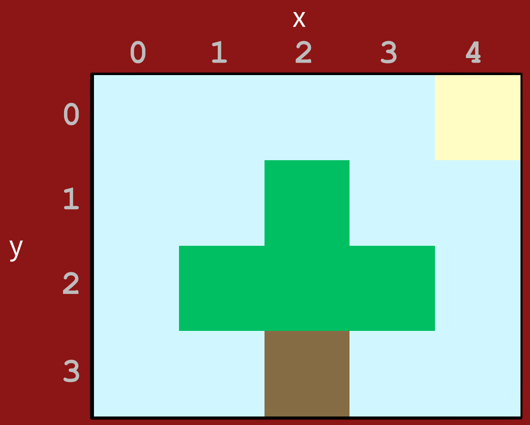
We can load image files into variables using Simple Image.

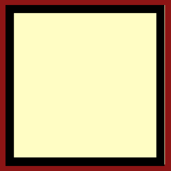


```
# we can now treat the image like a variable
image = SimpleImage('tree.jpg')
```

Pixels

*Pixels have red, green, and blue attributes. We can grab a pixel at x, y in an image with the get_pixel() function. Often we either **store** or **change** a pixel's value.*

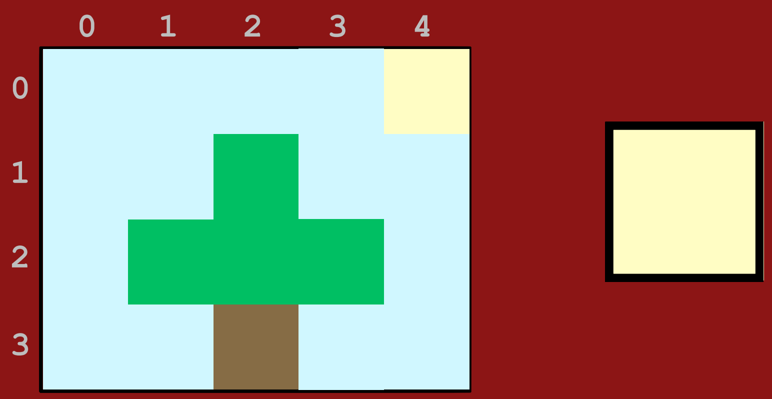


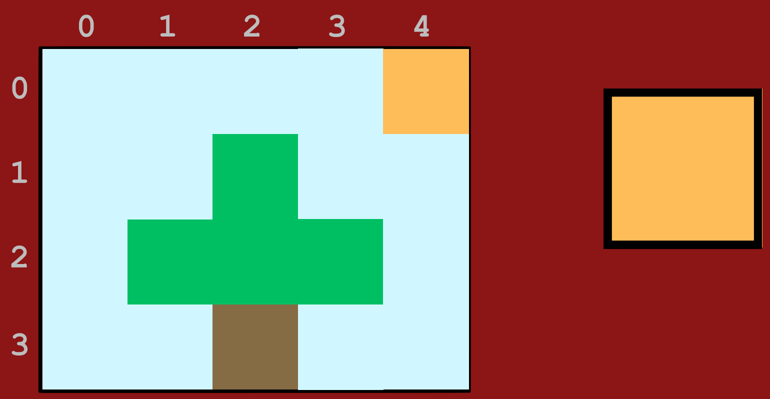


```
# we can now treat the image like a variable
image = SimpleImage('tree.jpg')
# we have access to the pixel now!
pixel = image.get_pixel(4, 0)
```

Pixels

*We can **store** and **change** a pixel's value.*

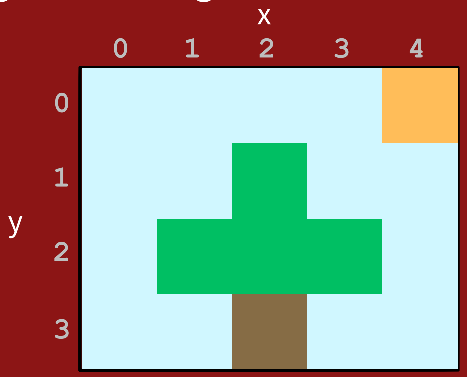


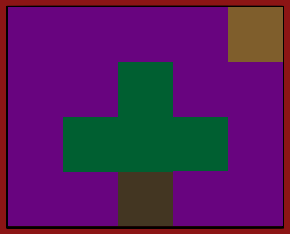


```
# we can now treat the image like a variable
image = SimpleImage('tree.jpg')
# we have access to the pixel now!
pixel = image.get_pixel(4, 0)
# we can change the pixel's values
pixel.red = 255
pixel.green = 189
pixel.blue = 89
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

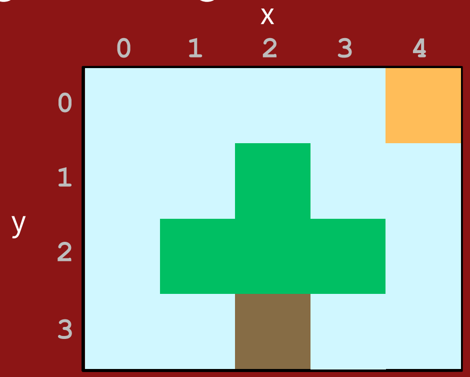


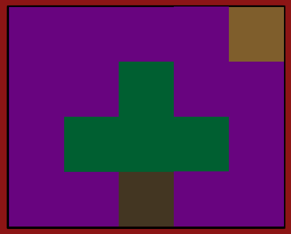


```
    image = SimpleImage('tree.jpg')
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

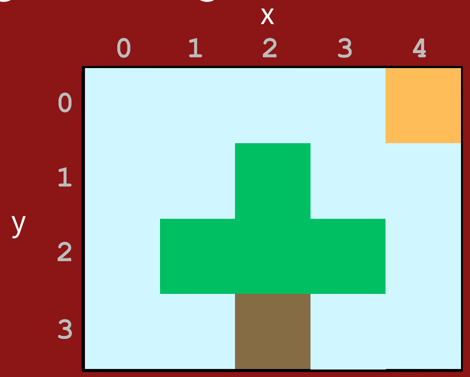


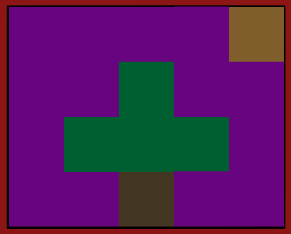


```
image = SimpleImage('tree.jpg')
for y in range(0, image.height): # is 4
   for x in range(0, image.width): # is 5
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

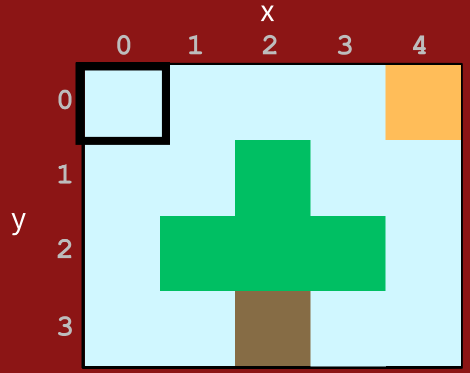


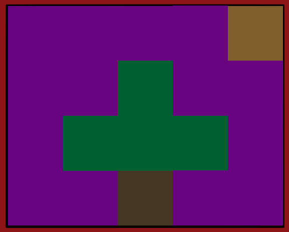


```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
    for x in range(image.width): # is 5
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

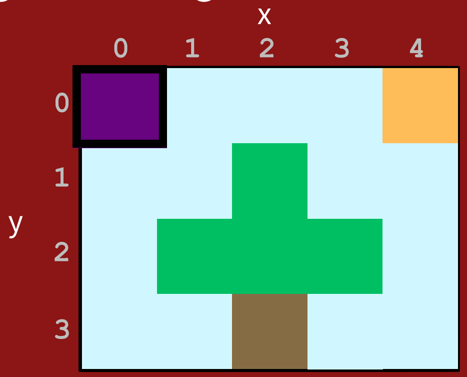


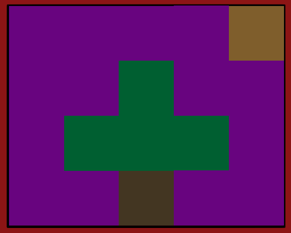


```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
    for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?



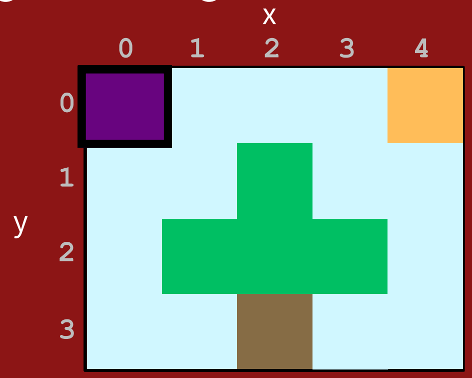


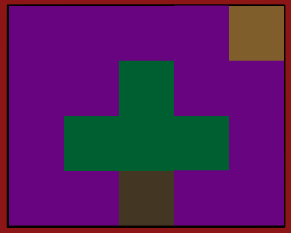
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 0
x = 0
pixel at (0,0)



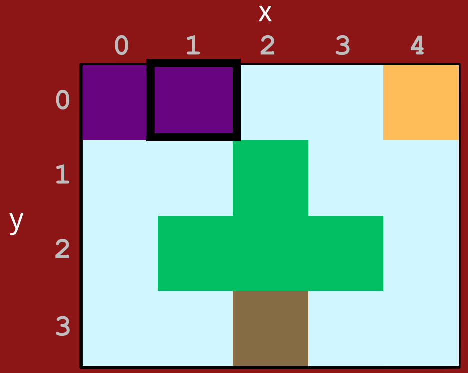


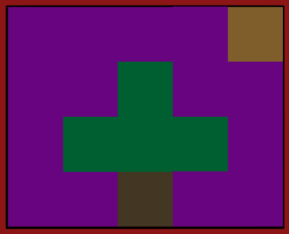
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 0
x = 1
pixel at (1,0)



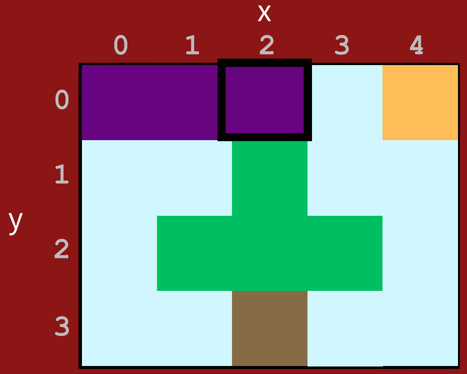


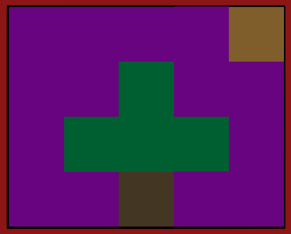
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 0
x = 2
pixel at (2,0)



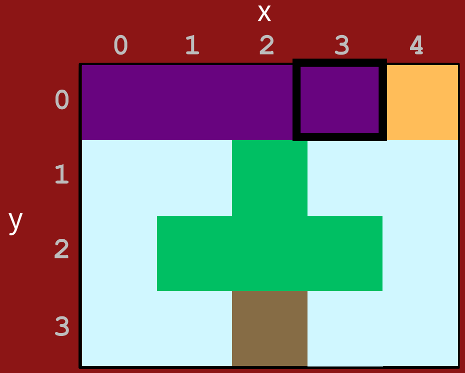


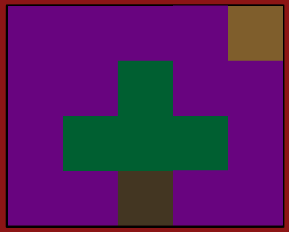
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
    for x in range(image.width): # is 5
            pixel = image.get_pixel(x,y)
            pixel.red = pixel.red*0.5
            pixel.green = pixel.green*0.5
            pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 0
x = 3
pixel at (3,0)



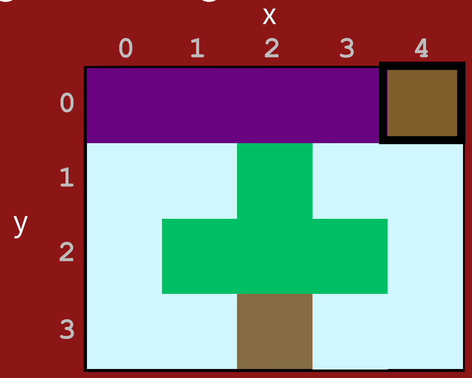


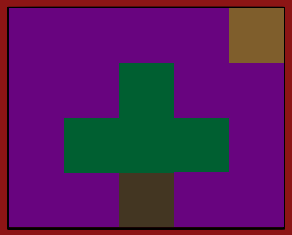
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 0
x = 4
pixel at (4,0)



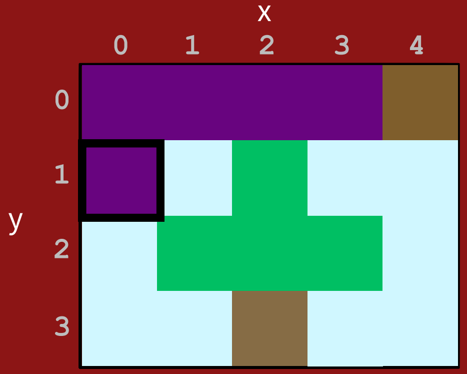


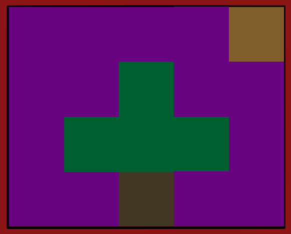
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 1
x = 0
pixel at (0,1)



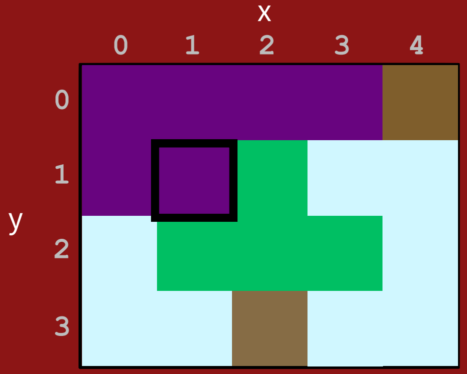


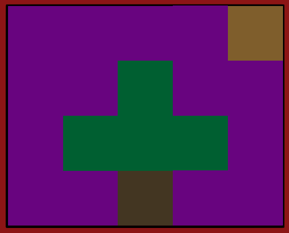
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 1
x = 1
pixel at (1,1)



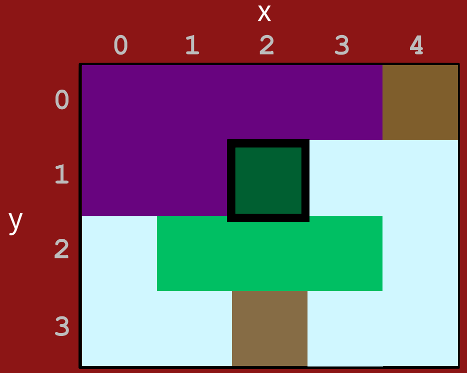


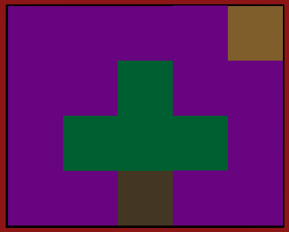
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
       pixel = image.get_pixel(x,y)
       pixel.red = pixel.red*0.5
       pixel.green = pixel.green*0.5
       pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 1
x = 2
pixel at (2,1)



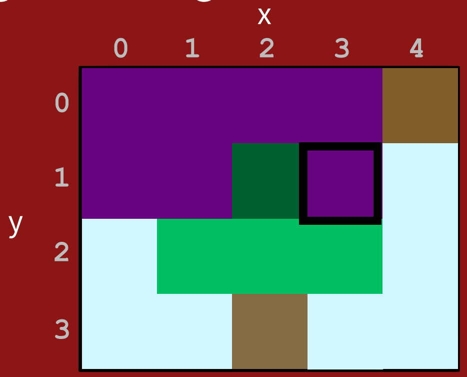


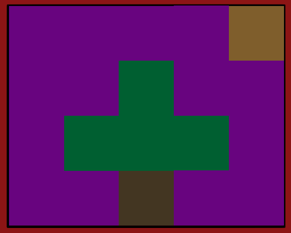
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
         pixel = image.get_pixel(x,y)
         pixel.red = pixel.red*0.5
         pixel.green = pixel.green*0.5
         pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 1
x = 3
pixel at (3,1)





```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

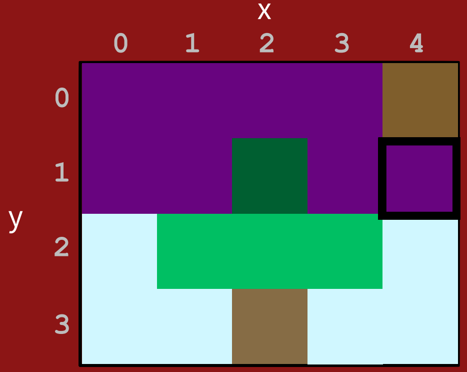
Double For Loop

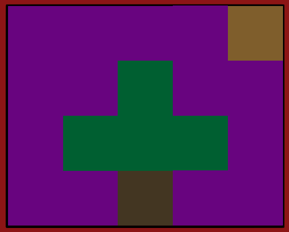
What if we wanted to go through all of the pixels and half their color?

y = 1

x = 4

pixel at (4,1)



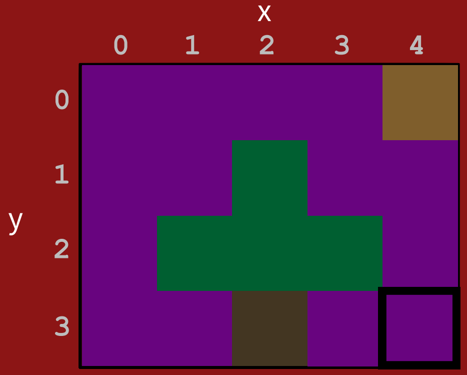


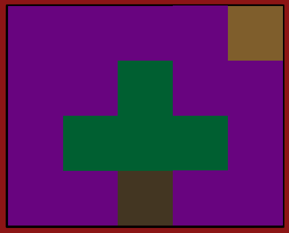
```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
   for x in range(image.width): # is 5
        pixel = image.get_pixel(x,y)
        pixel.red = pixel.red*0.5
        pixel.green = pixel.green*0.5
        pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

y = 3
x = 4
pixel at (4,3)





```
image = SimpleImage('tree.jpg')
for y in range(image.height): # is 4
    for x in range(image.width): # is 5
            pixel = image.get_pixel(x,y)
            pixel.red = pixel.red*0.5
            pixel.green = pixel.green*0.5
            pixel.blue = pixel.blue*0.5
return image
```

Double For Loop

What if we wanted to go through all of the pixels and half their color?

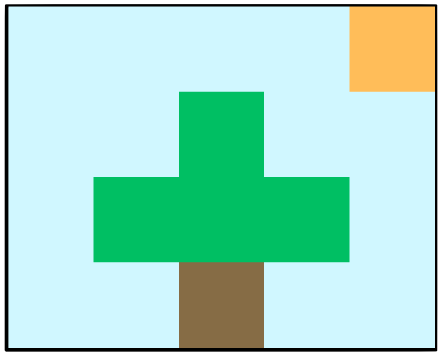
Darker Nested

Image Functions

- `image = SimpleImage(filename)`
- `width = image.width`
- `height = image.height`
- `pixel = image.get_pixel(x, y)`

Pixel Attributes and Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255 # set pixel to exact color`



New Functionality

Image Functions

- `image = SimpleImage(filename)`
- **`out = SimpleImage.blank(width, height)`**
- `width = image.width`
- `height = image.height`
- `pixel = image.get_pixel(x, y)`

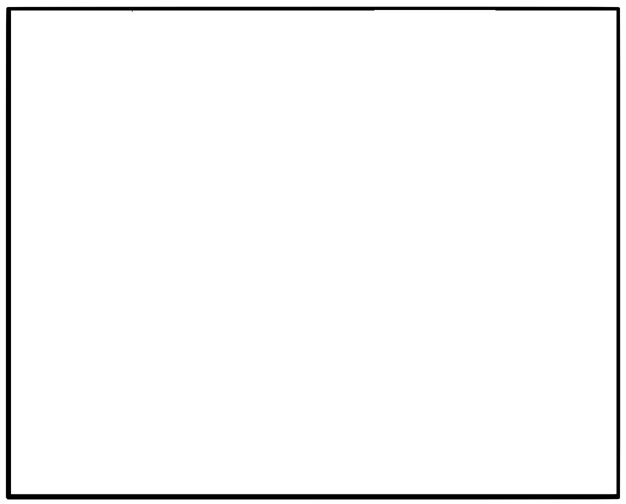
Pixel Attributes and Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255 # set pixel to exact color`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

New Image Functions

- `image = SimpleImage(filename)`
- **`out = SimpleImage.blank(width, height)`**

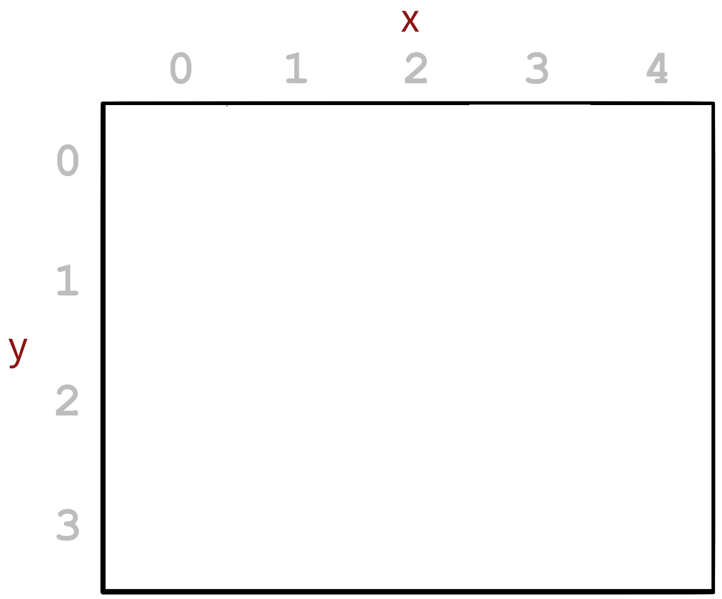
```
# Create a blank image of custom width, height
out = SimpleImage.blank(5, 4)
```



New Image Functions

- `image = SimpleImage(filename)`
- **`out = SimpleImage.blank(width, height)`**

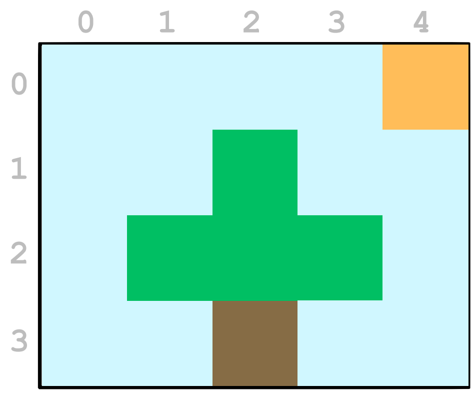
```
# Create a blank image of custom width, height
out = SimpleImage.blank(5, 4)
```



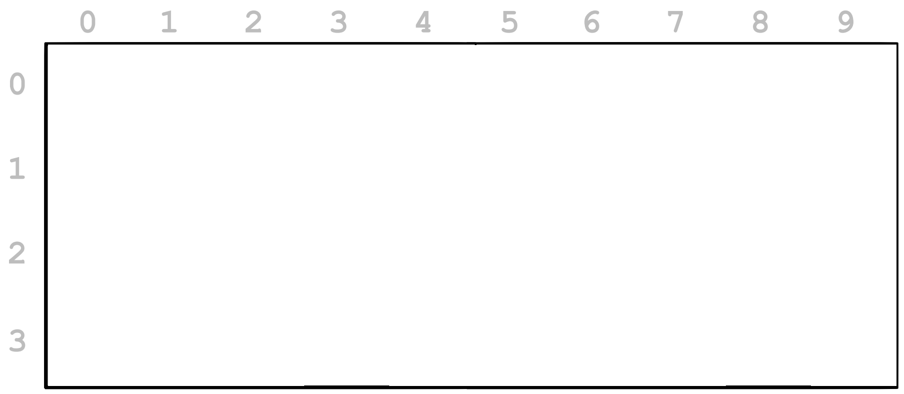
New Image Functions

- `image = SimpleImage(filename)`
- **`out = SimpleImage.blank(width, height)`**

```
image = SimpleImage('tree.jpg')
width = image.width
height = image.height
# Create a blank image twice as wide as the OG
out = SimpleImage.blank(width*2, height)
```



original

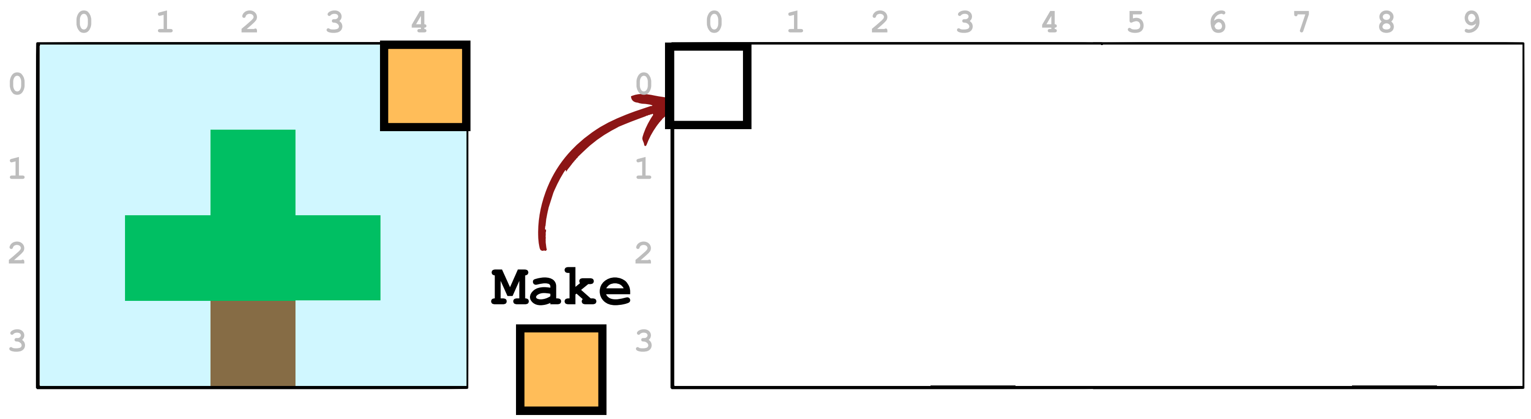


new image
twice as wide and BLANK

New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

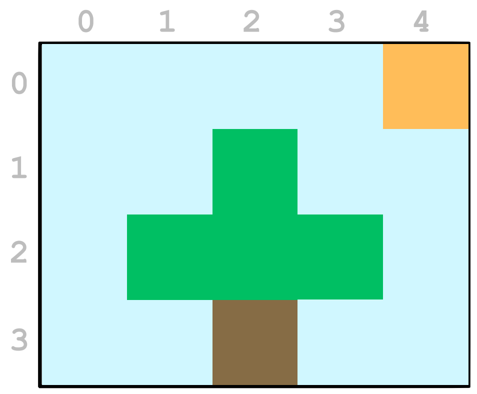
```
# GOAL: Set one pixel to another pixel's value
```



New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

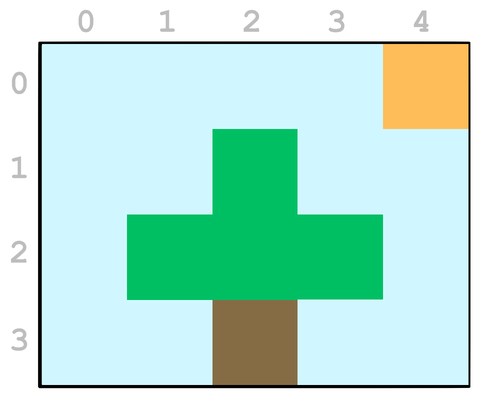
```
# GOAL: Set one pixel to another pixel's value
image = SimpleImage('tree.jpg')# get OG image
```

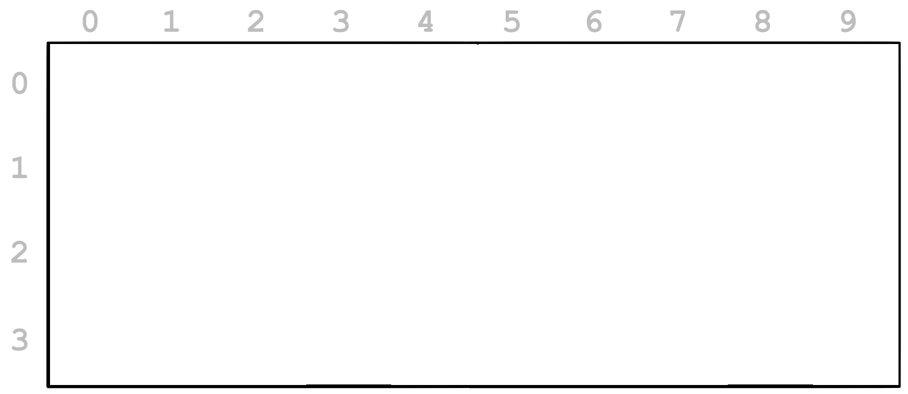


New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

```
# GOAL: Set one pixel to another pixel's value
image = SimpleImage('tree.jpg')# get OG image
out = SimpleImage.blank(10, 4) # create out
```

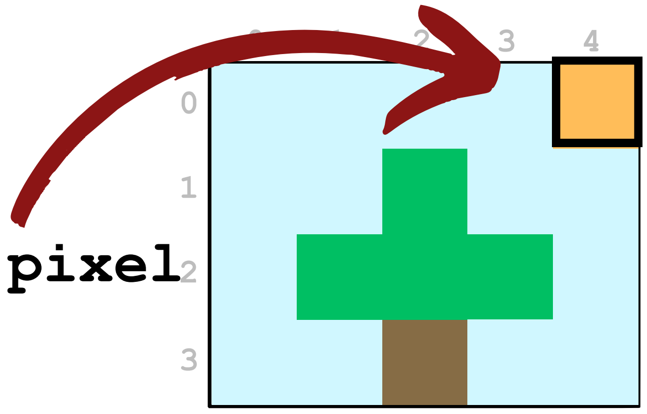


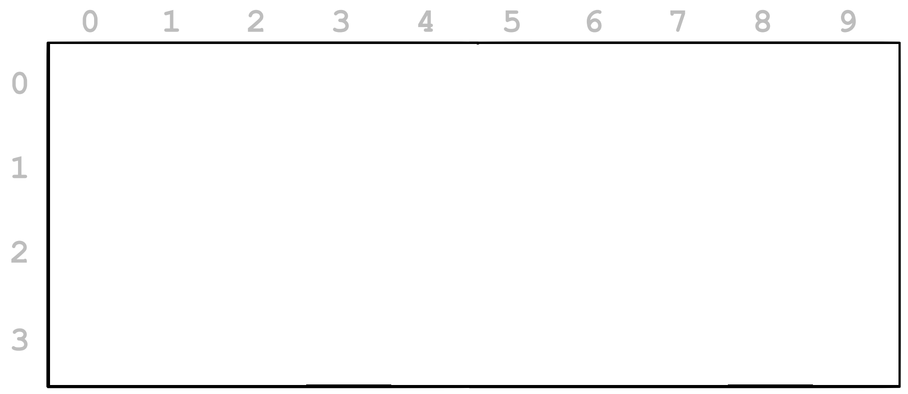


New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

```
# GOAL: Set one pixel to another pixel's value
image = SimpleImage('tree.jpg')# get OG image
out = SimpleImage.blank(10, 4) # create out
pixel = image.get_pixel(4,0)# get original pixel
```

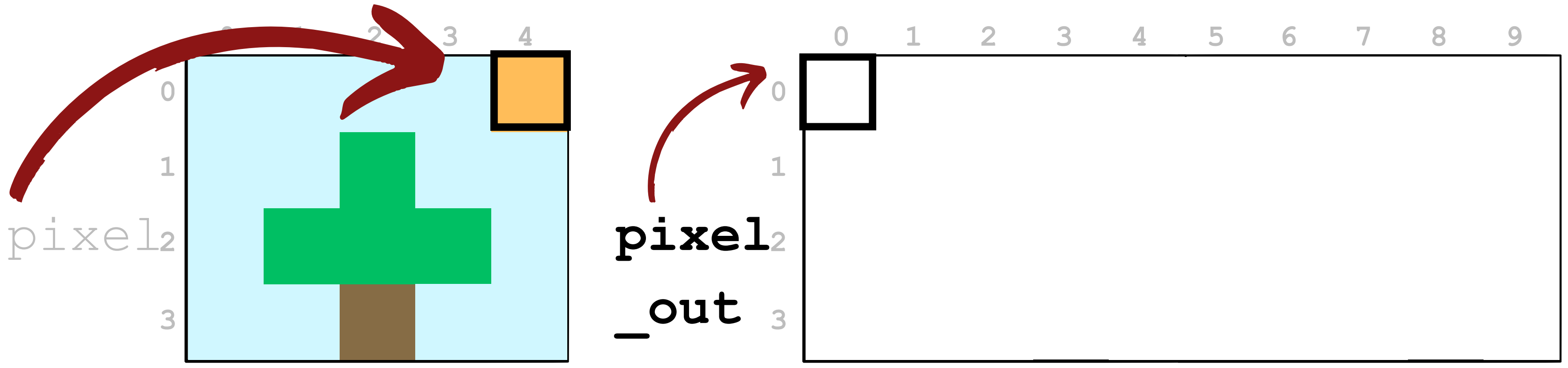




New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

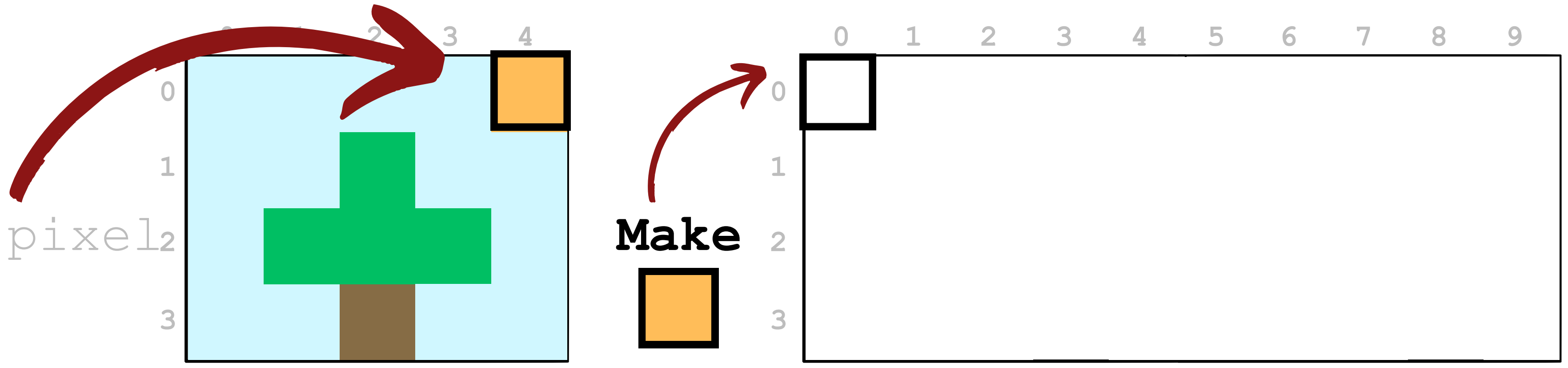
```
# GOAL: Set one pixel to another pixel's value
image = SimpleImage('tree.jpg')# get OG image
out = SimpleImage.blank(10, 4) # create out
pixel = image.get_pixel(4,0)# get original pixel
pixel_out = out.get_pixel(0,0) # get blank pixel
```



New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

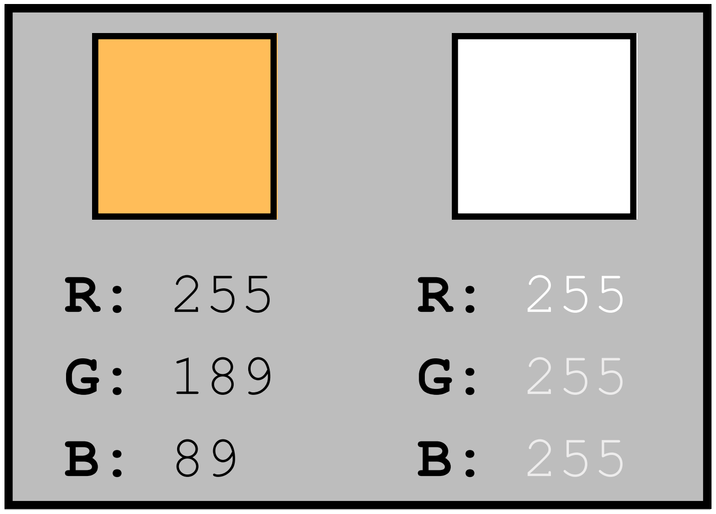
```
# GOAL: Set one pixel to another pixel's value
image = SimpleImage('tree.jpg')# get OG image
out = SimpleImage.blank(10, 4) # create out
pixel = image.get_pixel(4,0)# get original pixel
pixel_out = out.get_pixel(0,0) # get blank pixel
```



New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

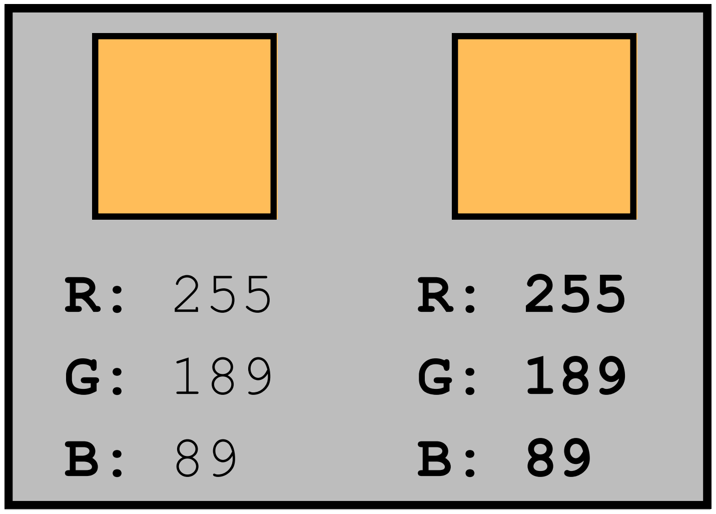
```
# GOAL: Set one pixel to another pixel's value
# set blank pixel to our original pixel's value
pixel_out.red = pixel.red
pixel_out.green = pixel.green
pixel_out.blue = pixel.blue
```



New Pixel Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

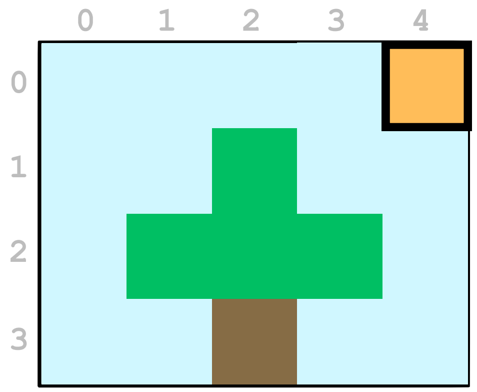
```
# GOAL: Set one pixel to another pixel's value
# set blank pixel to our original pixel's value
pixel_out.red = pixel.red
pixel_out.green = pixel.green
pixel_out.blue = pixel.blue
```

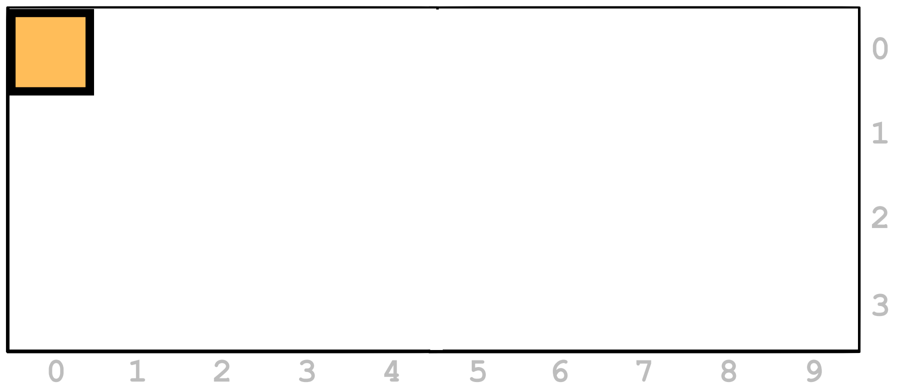


New Pixel Functionality

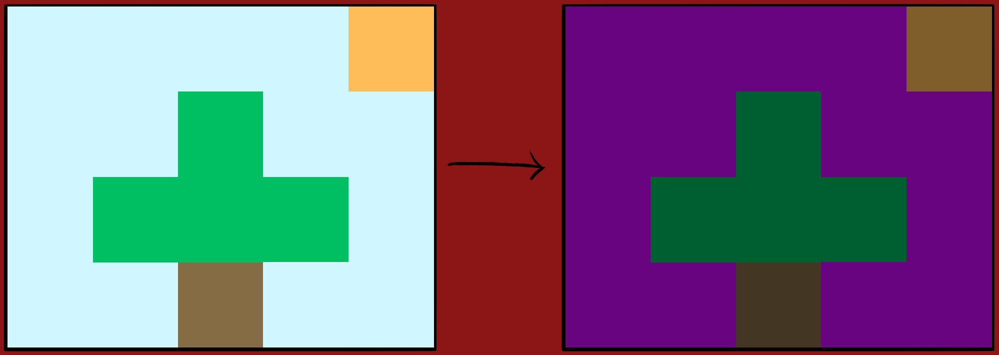
- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255`
- **`pixel_out.red = pixel.red`** `# assuming pixel_out`
- **`pixel_out.green = pixel.green`**
- **`pixel_out.blue = pixel.blue`**

```
# GOAL: Set one pixel to another pixel's value
# Achieved :)!
```

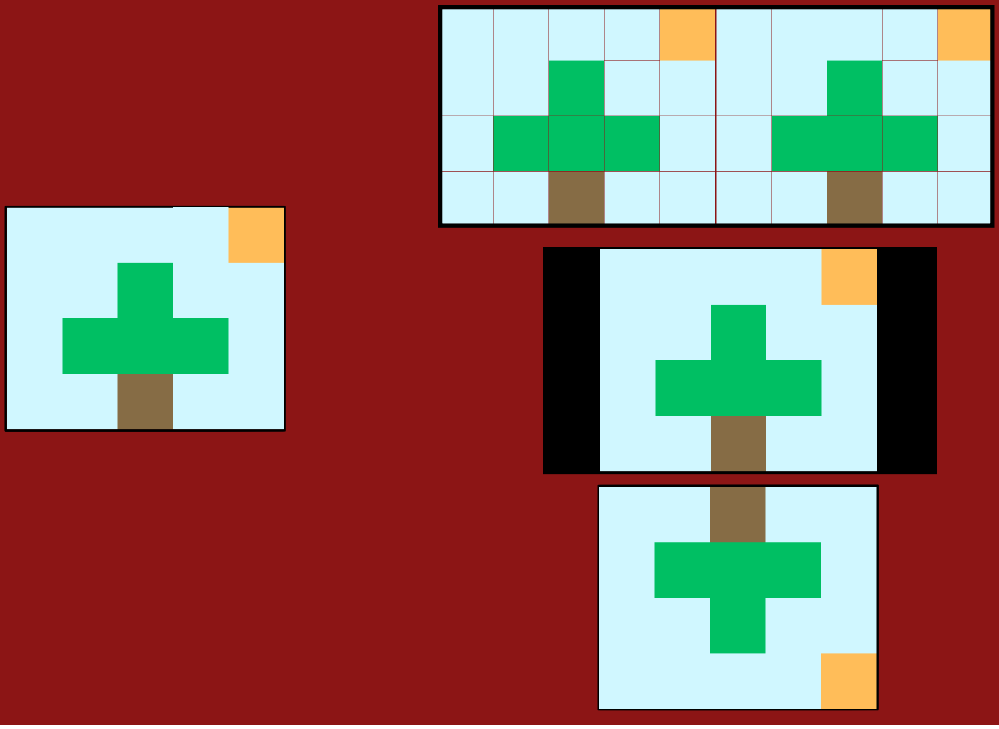




Editing the Same Image



Creating an Out Image



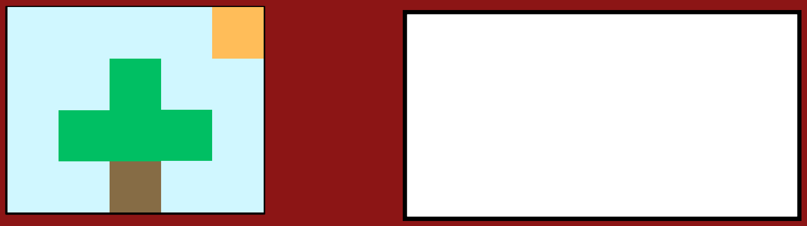
New image that's now doubled

New image that now has margins

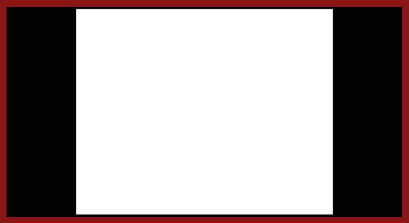
New image that's now flipped

General Steps: Creating an Out Image

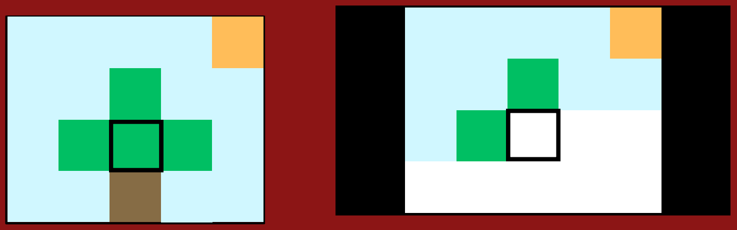
Step 1: Create a new blank image based on original image



Step 2: If margins, loop over new image to create

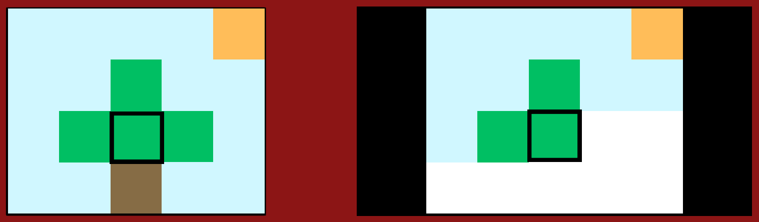


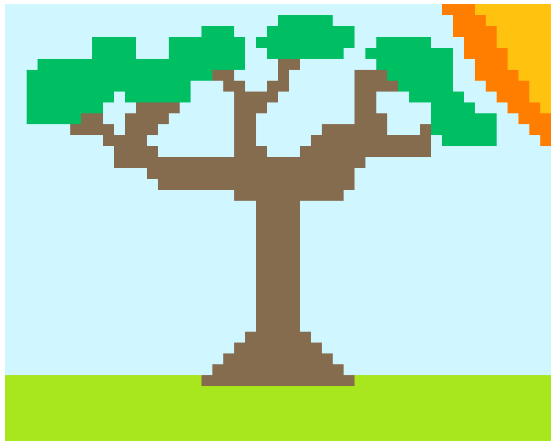
Step 3: Loop over the original image and find corresponding pixel(s) in the new image

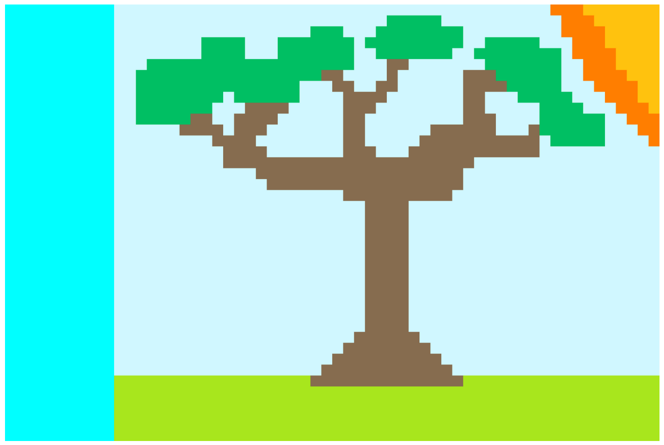


(x, y) -> (x + 1, y)

Step 4: Set corresponding new pixel values to old ones

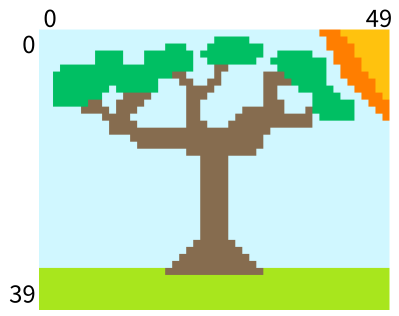






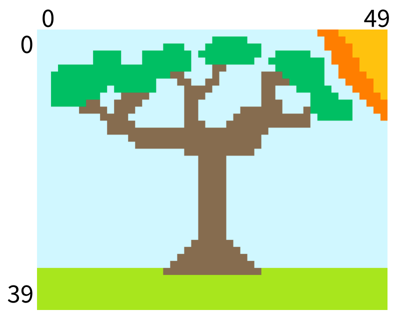
Aqua stripe problem

Step 1: Create a new blank image based on original image



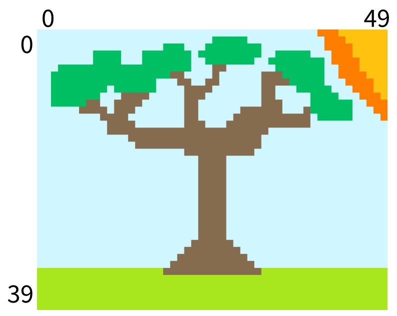
```
def aqua_stripe(filename):
    image = SimpleImage(filename)
```

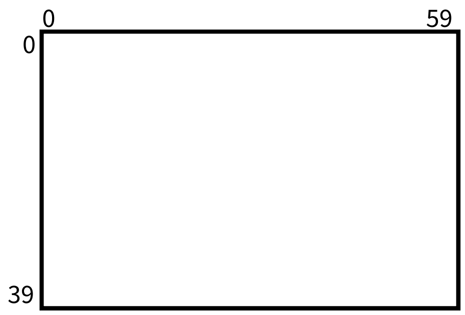
Step 1: Create a new blank image based on original image



```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
```

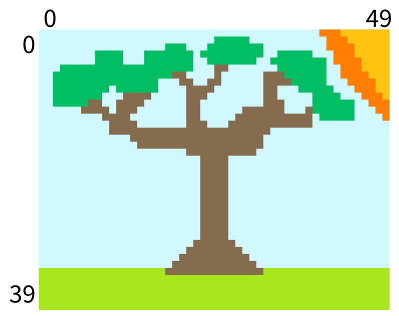
Step 1: Create a new blank image based on original image

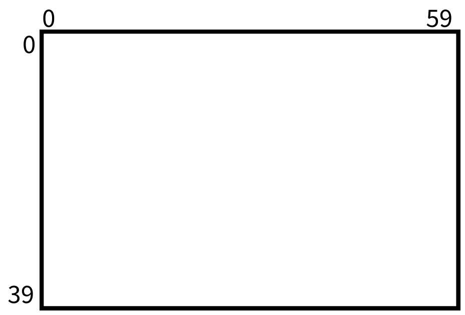




```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
```

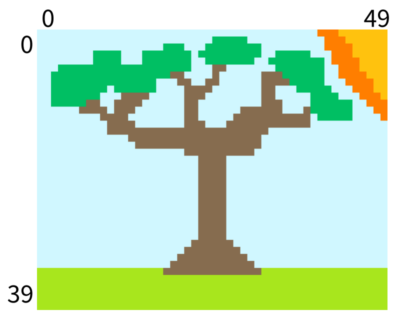
Step 2: If margins, loop over and create in the new image

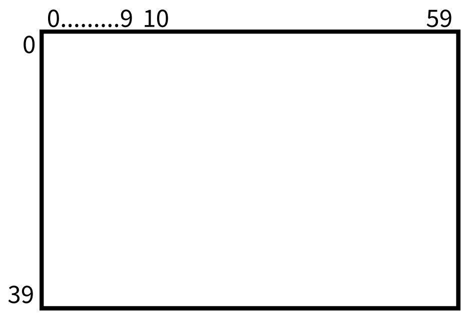




```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
```

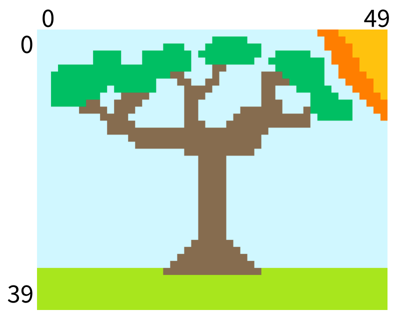
Step 2: If margins, loop over and create in the new image

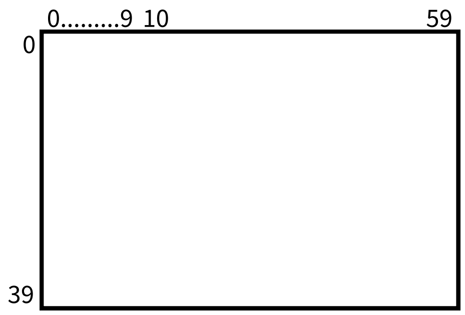




```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
```

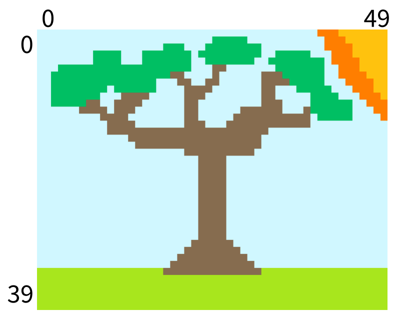
Step 2: If margins, loop over and create in the new image

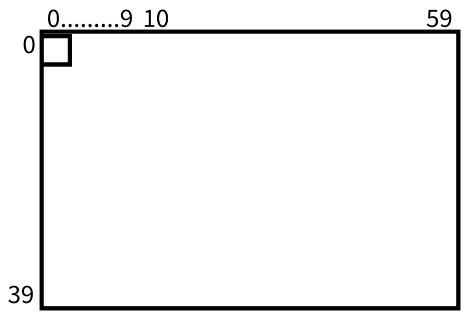




```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):  # only some has aqua
```

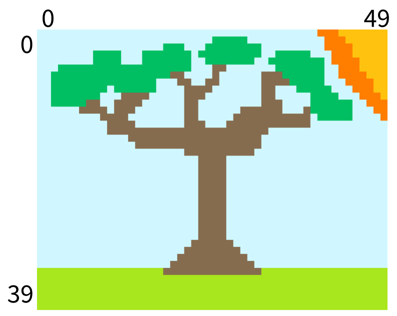
Step 2: If margins, loop over and create in the new image

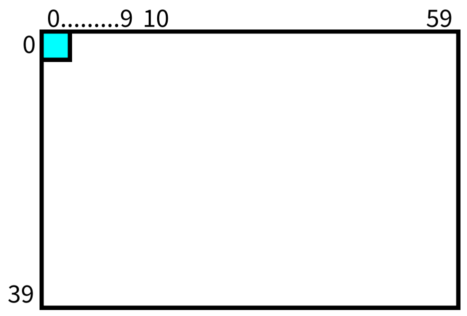




```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):  # only some has aqua
            pixel_out = out.get_pixel(x, y)
```

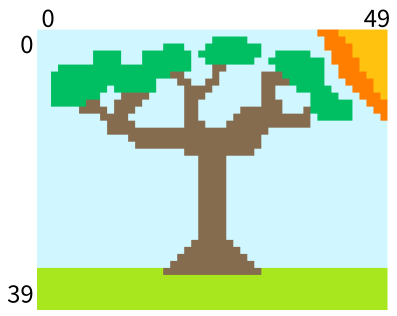
Step 2: If margins, loop over and create in the new image

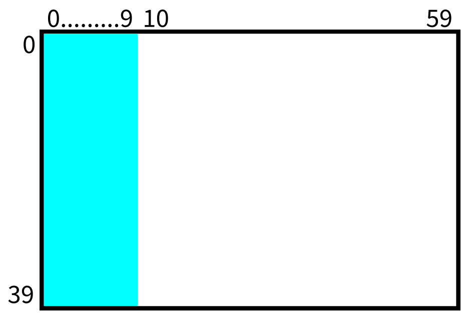




```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):  # only some has aqua
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
```

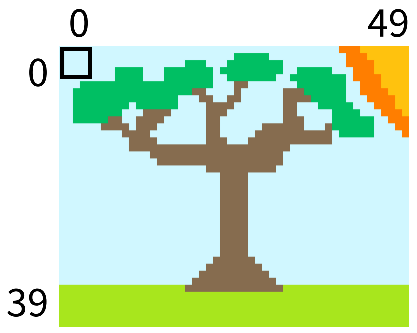
Step 2: If margins, loop over and create in the new image

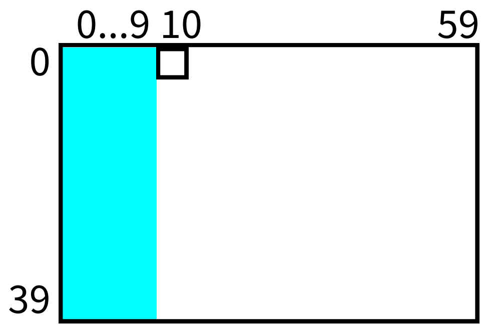




```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):  # only some has aqua
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
```

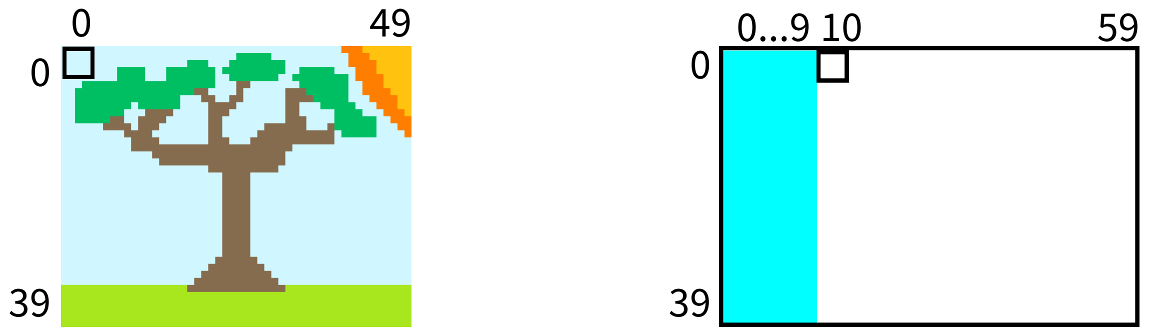
Step 3: Loop over the original image to find corresponding pixel(s) in the new image





```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):  # only some has aqua
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
```

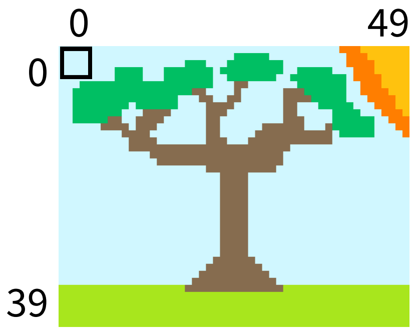
Step 3: Loop over the original image to find corresponding pixel(s) in the new image

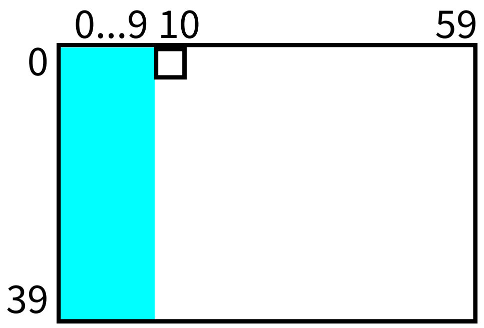


Let's trace the journey of a single pixel!

image	out
0, 0	
5, 10	
49, 0	

Step 3: Loop over the original image to find corresponding pixel(s) in the new image

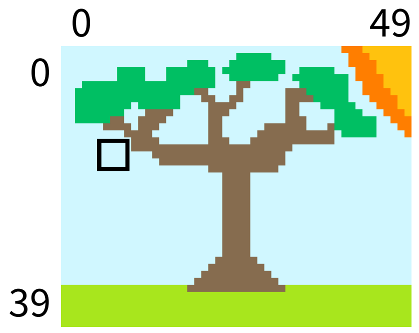


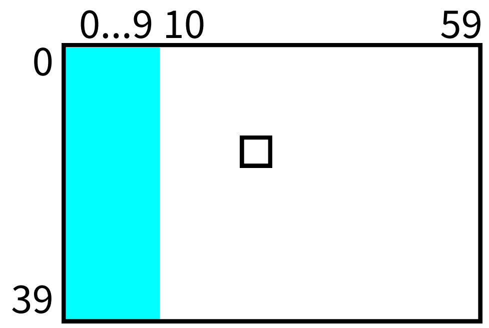


Let's trace the journey of a single pixel!

image	out
0, 0	10, 0
5, 10	
49, 0	

Step 3: Loop over the original image to find corresponding pixel(s) in the new image

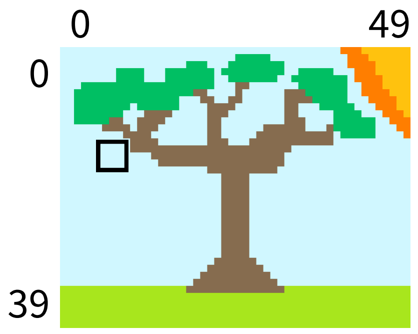


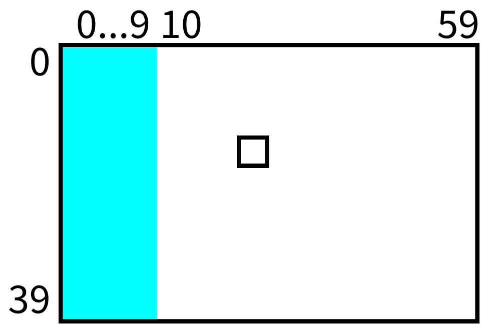


Let's trace the journey of a single pixel!

image	out
0, 0	10, 0
5, 10	
49, 0	

Step 3: Loop over the original image to find corresponding pixel(s) in the new image

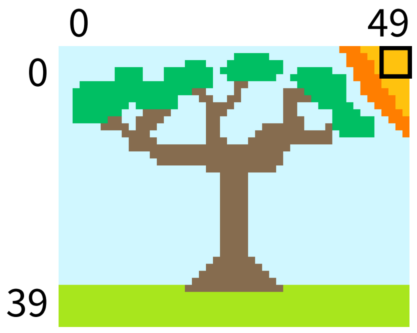


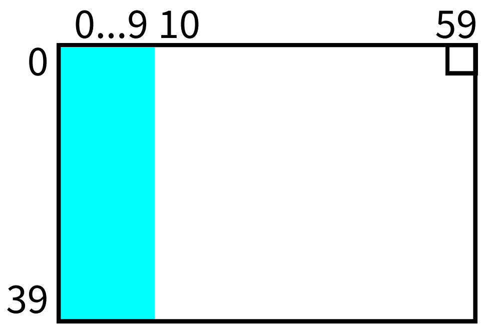


Let's trace the journey of a single pixel!

image	out
0, 0	10, 0
5, 10	15, 10
49, 0	

Step 3: Loop over the original image to find corresponding pixel(s) in the new image

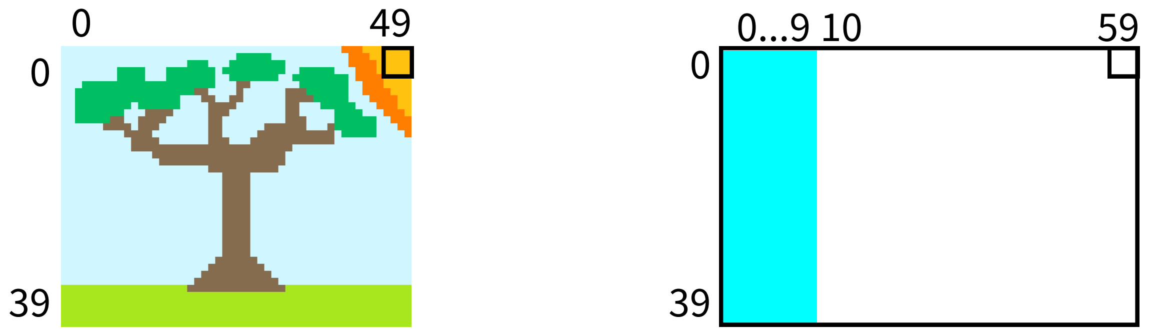




Let's trace the journey of a single pixel!

image	out
0, 0	10, 0
5, 10	15, 10
49, 0	

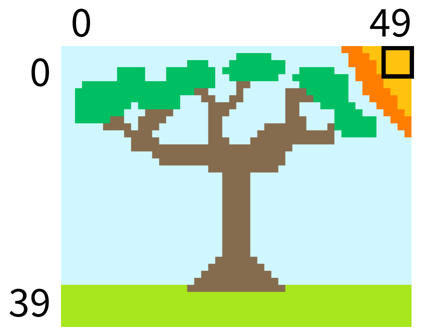
Step 3: Loop over the original image to find corresponding pixel(s) in the new image

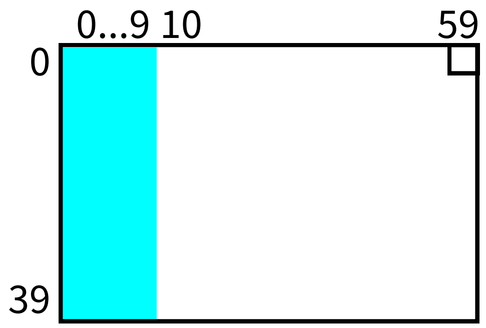


Let's trace the journey of a single pixel!

image	out
0, 0	10, 0
5, 10	15, 10
49, 0	59, 0

Step 3: Loop over the original image to find corresponding pixel(s) in the new image

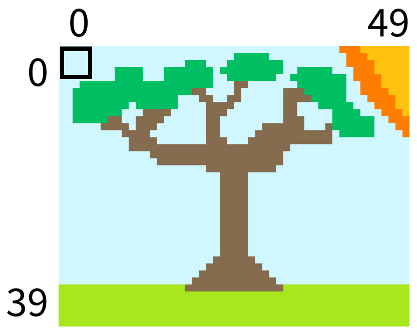


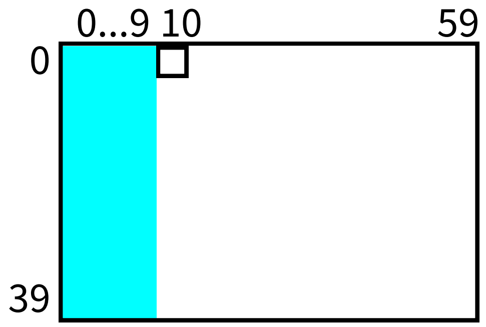


Let's trace the journey of a single pixel!

image	out
0, 0	10, 0
5, 10	15, 10
49, 0	59, 0
x, y	x + 10, y

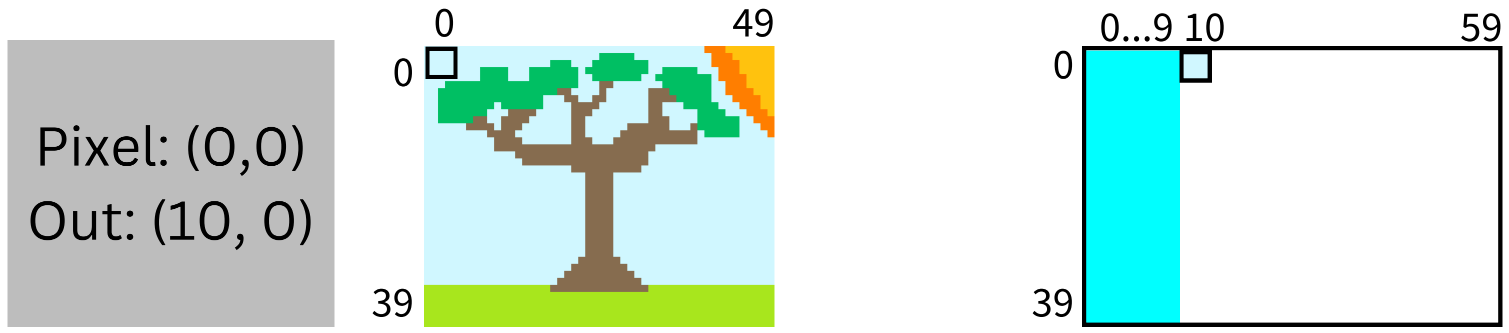
Step 3: Loop over the original image to find corresponding pixel(s) in the new image





```
def aqua_stripe(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    # creates BLANK image with proper dimensions
    out = SimpleImage.blank(width + 10, height)
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10)
```

Step 4: Set corresponding new pixel values to old ones

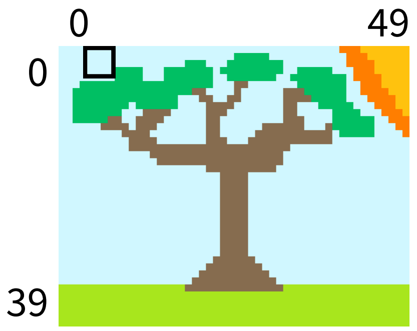


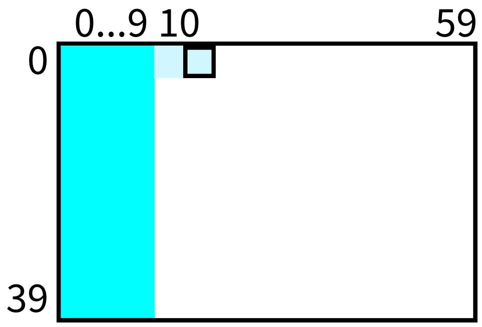
```
def aqua_stripe(filename):
    ...
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)

            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```

Step 4: Set corresponding new pixel values to old ones

Pixel: (1,0)
Out: (11, 0)



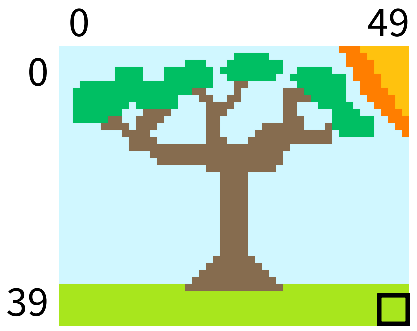


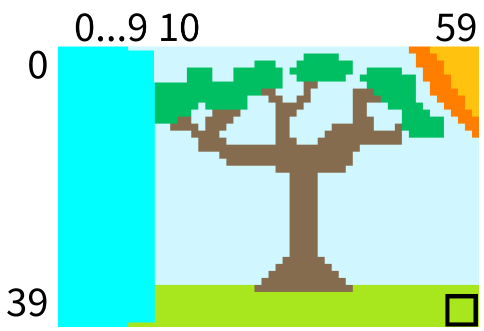
```
def aqua_stripe(filename):
    ...
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)

            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```

Step 4: Set corresponding new pixel values to old ones

Pixel: (49,39)
Out: (59, 39)



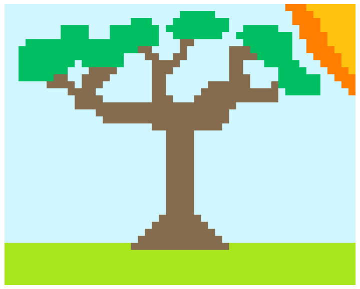


```
def aqua_stripe(filename):
    ...
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)

            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```

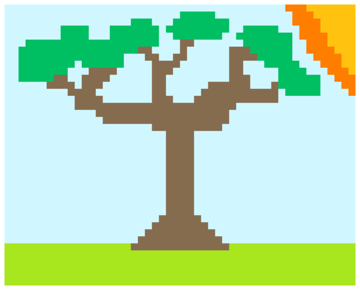
Whole Solution

```
def aqua_strip(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)
            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```



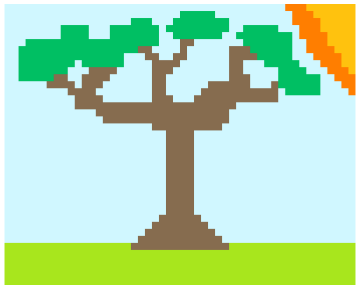
Whole Solution

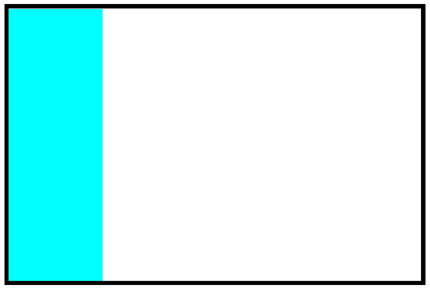
```
def aqua_strip(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)
            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```



Whole Solution

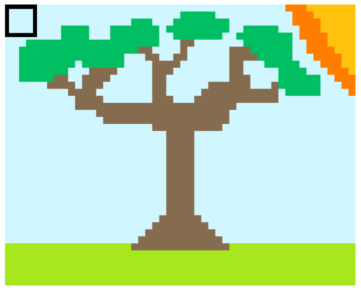
```
def aqua_strip(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)
            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```

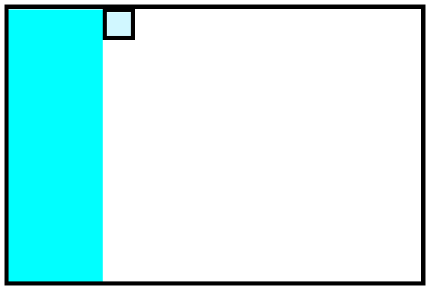




Whole Solution

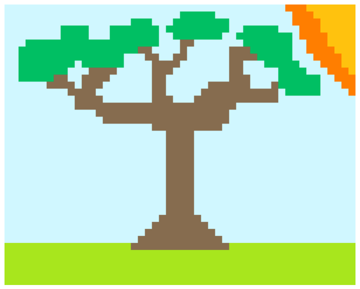
```
def aqua_strip(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)
            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```

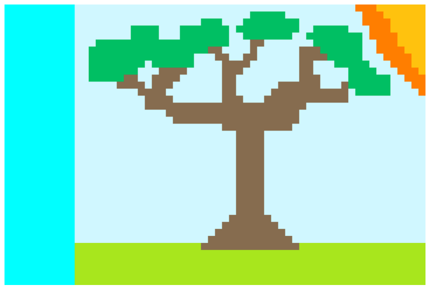




Whole Solution

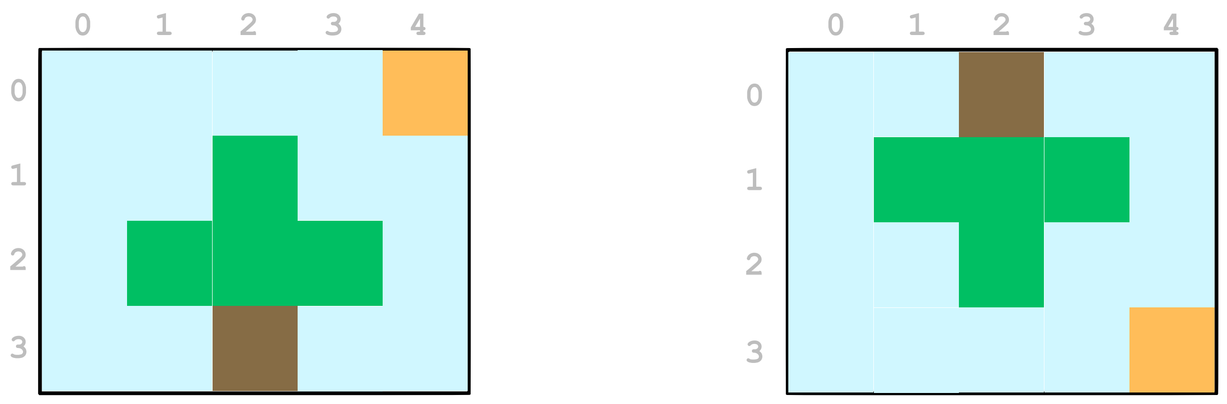
```
def aqua_strip(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height
    out = SimpleImage.blank(width + 10, height)
    for y in range(out.height):
        for x in range(10):
            pixel_out = out.get_pixel(x, y)
            pixel_out.red = 0
    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            pixel_out = out.get_pixel(x + 10, y)
            pixel_out.red = pixel.red
            pixel_out.green = pixel.green
            pixel_out.blue = pixel.blue
    return out
```



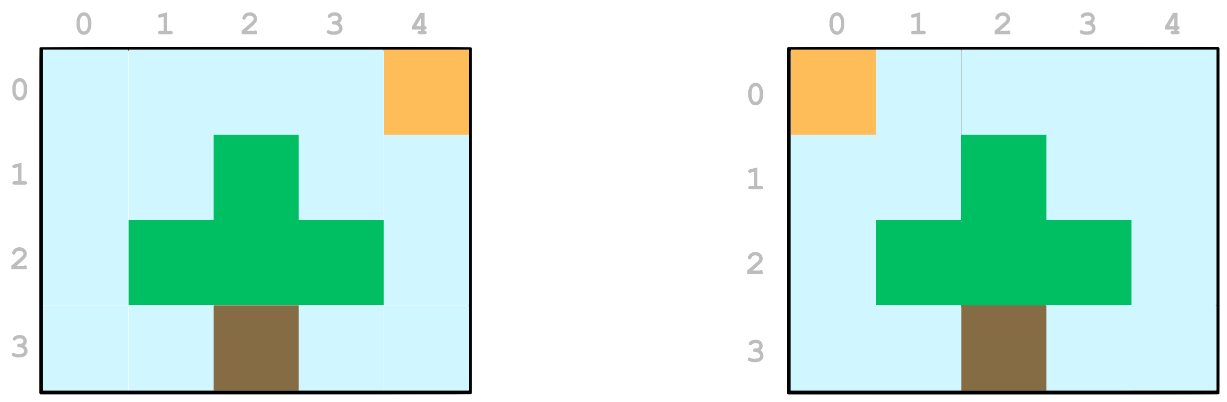


Mirror2

A Quick Note on Image Flipping



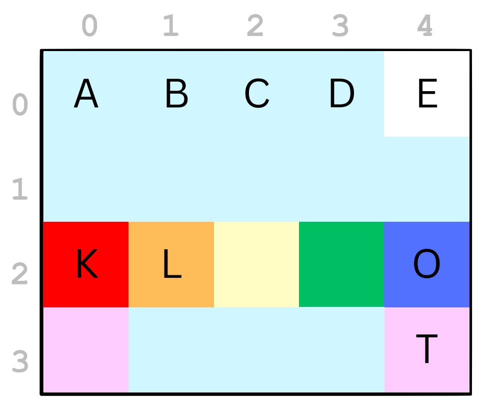
Flipping vertically(y)



Flipping horizontally (x)

A Quick Note on Image Flipping

Flipping Horizontally



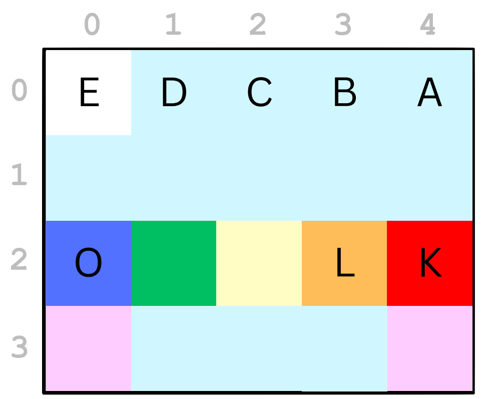
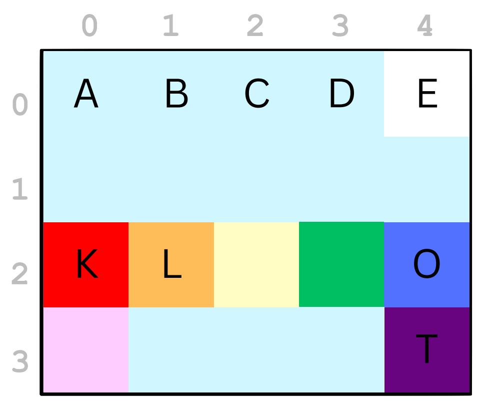


image	out
0,0	4,0
0,2	4,2
1,2	3,2
2,0	2,0
3,0	1,0
4,2	0,2

A Quick Note on Image Flipping

Flipping Horizontally



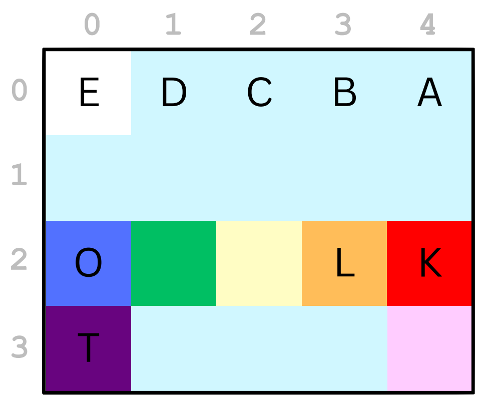
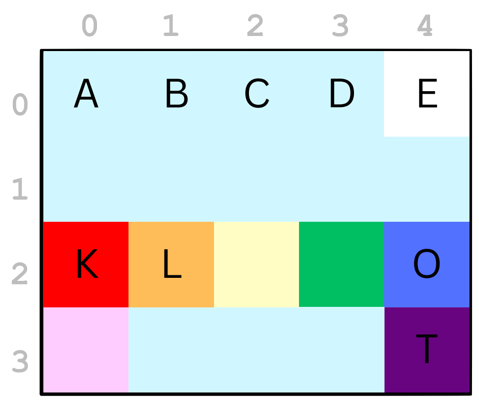


image	out
0,0	4,0
0,2	4,2
1,2	3,2
2,0	2,0
3,0	1,0
4,2	0,2
x, y	?, ?

A Quick Note on Image Flipping

Flipping Horizontally



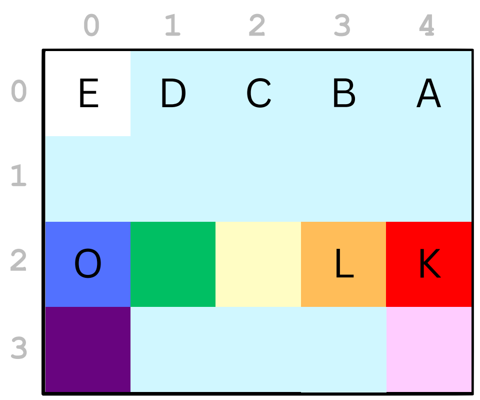
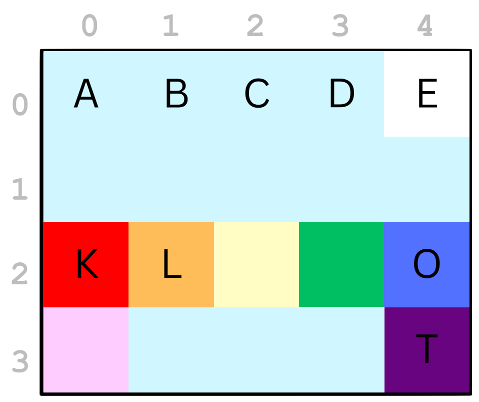


image	out
0,0	4,0
0,2	4,2
1,2	3,2
2,0	2,0
3,0	1,0
4,2	0,2
x, y	?, y

A Quick Note on Image Flipping

Flipping Horizontally



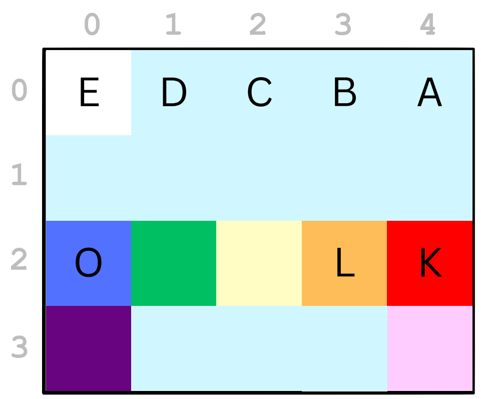
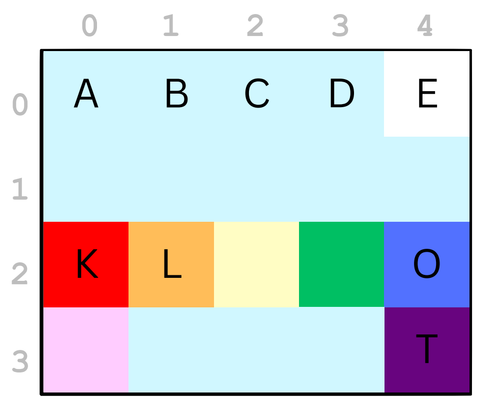


image	out
0,0	4,0
0,2	4,2
1,2	3,2
2,0	2,0
3,0	1,0
4,2	0,2
x, y	?, y

edge coordinate - x

A Quick Note on Image Flipping

Flipping Horizontally



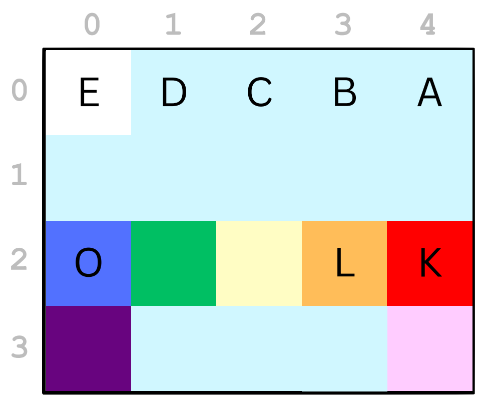


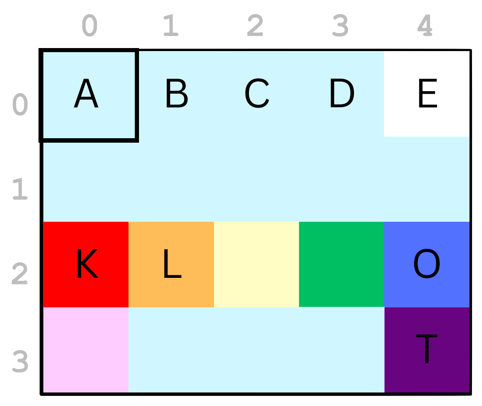
image	out
0,0	4,0
0,2	4,2
1,2	3,2
2,0	2,0
3,0	1,0
4,2	0,2
x, y	?, y

edge coordinate - x

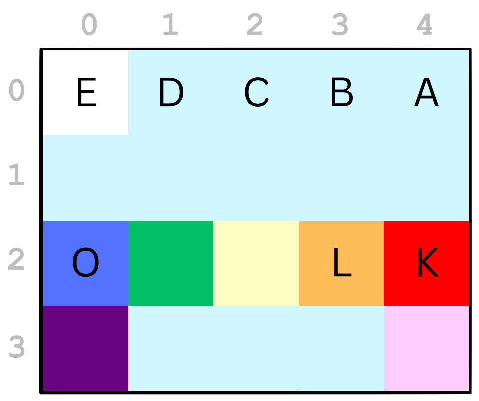
(out.width - 1) - x

A Quick Note on Image Flipping

Flipping Horizontally

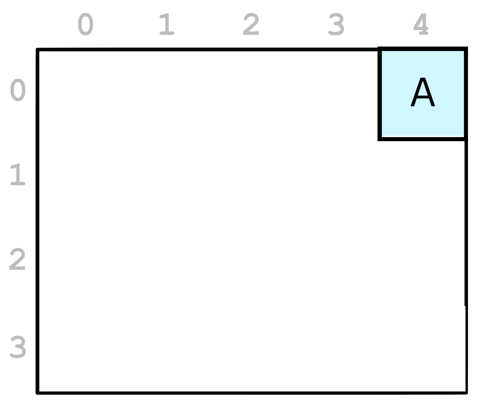


image



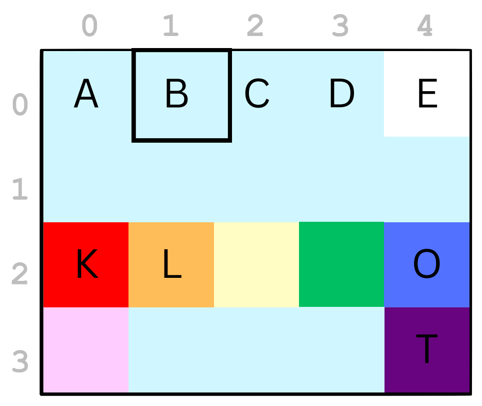
out

image: (0,0)
out: (4,0)

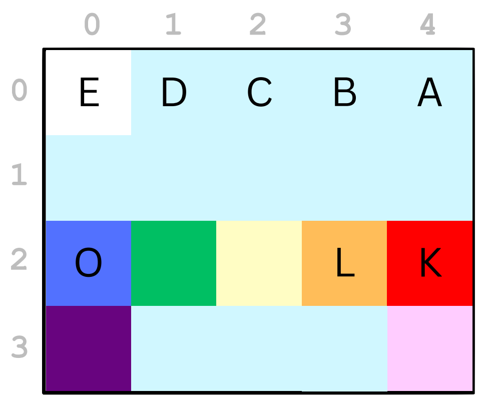


A Quick Note on Image Flipping

Flipping Horizontally

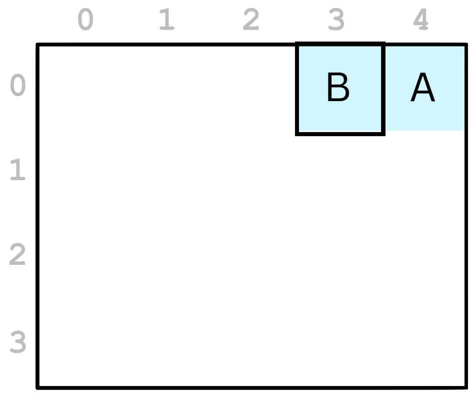


image



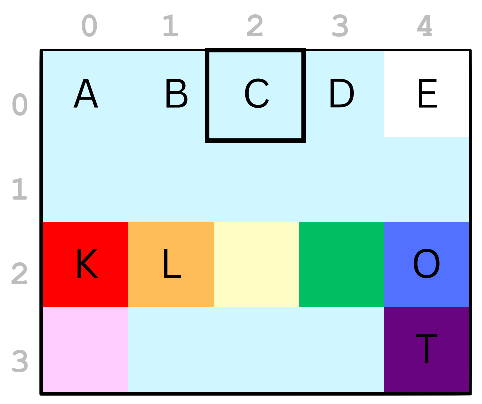
out

image: (1,0)
out: (3,0)

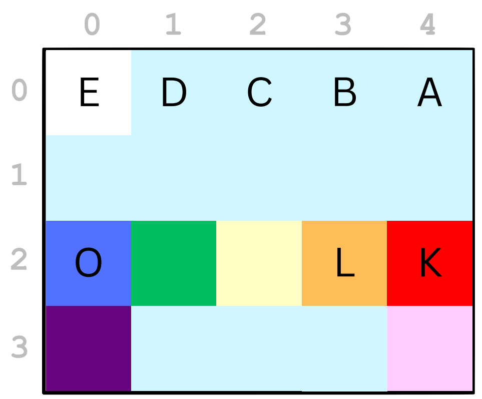


A Quick Note on Image Flipping

Flipping Horizontally

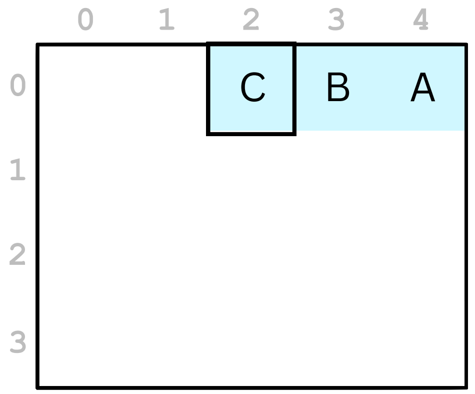


image



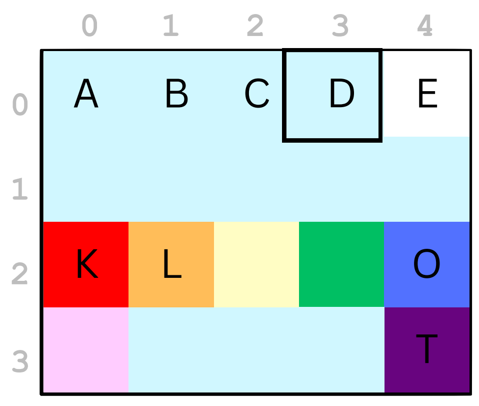
out

image: (2,0)
out: (2,0)

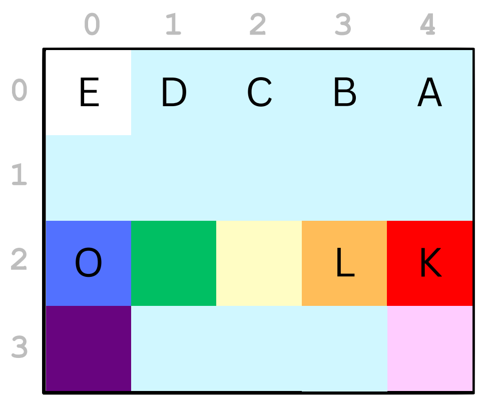


A Quick Note on Image Flipping

Flipping Horizontally

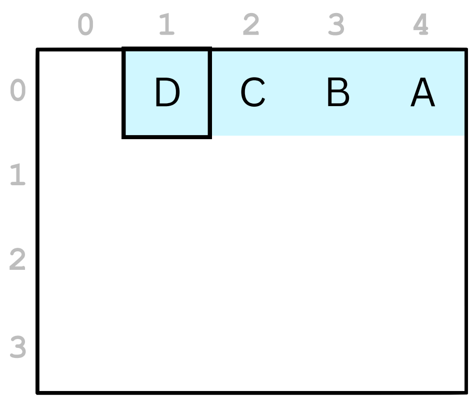


image



out

image: (3,0)
out: (1,0)



A Quick Note on Image Flipping

Flipping Horizontally

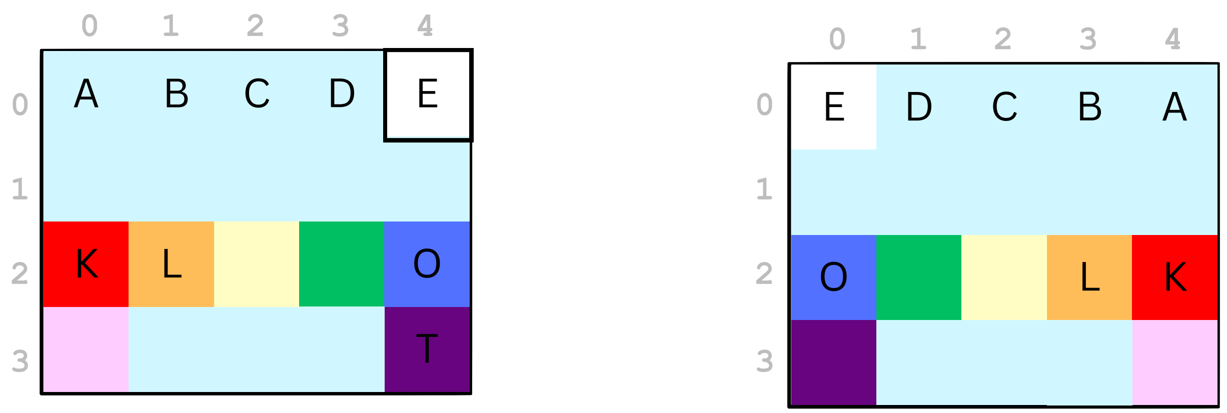
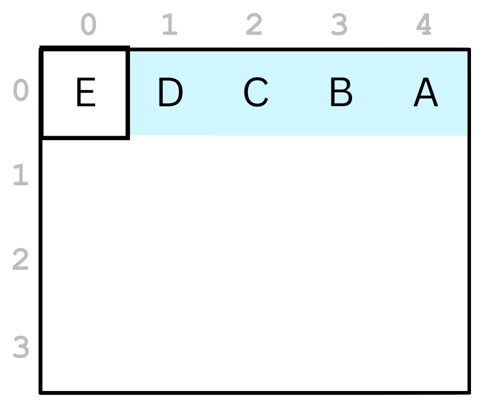


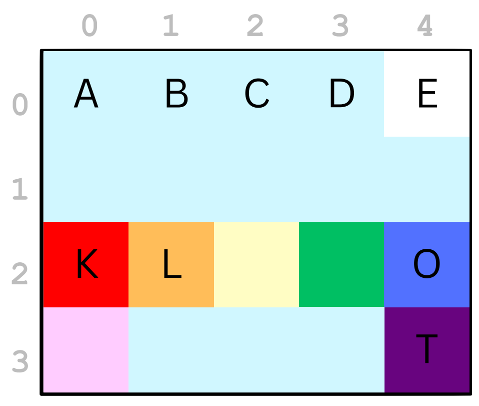
image **out**

image: (4,0)
out: (0,0)

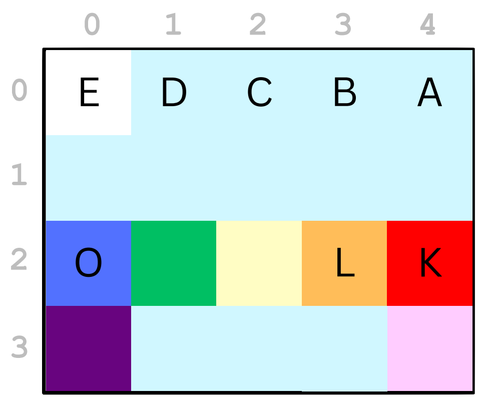


A Quick Note on Image Flipping

Flipping Horizontally



image



out

image: (4,3)
out: (0,3)

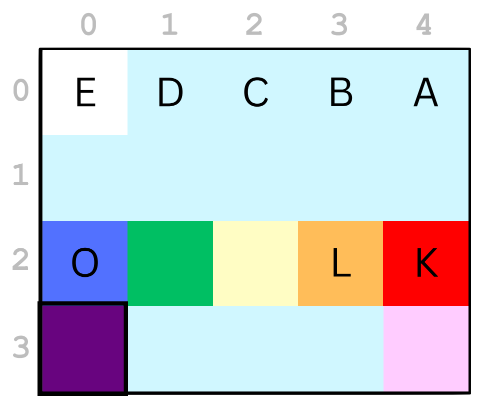


Image Functions

- `image = SimpleImage(filename)`
- `out = SimpleImage.blank(width, height)`
- `width = image.width`
- `height = image.height`
- `pixel = image.get_pixel(x, y)`

Pixel Attributes and Functionality

- `pixel.red, pixel.blue, pixel.green`
- `pixel.red = 255 # set pixel to color`
- `pixel_out.red = pixel.red # assuming pixel_out`
- `pixel_out.green = pixel.green`
- `pixel_out.blue = pixel.blue`

Image Problems

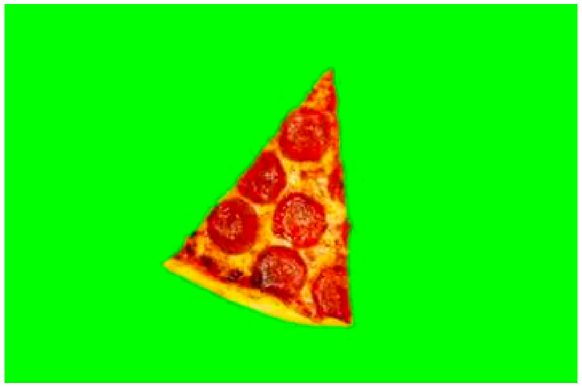
- Old Image Filtering
- New Image Creation

PyCharm!

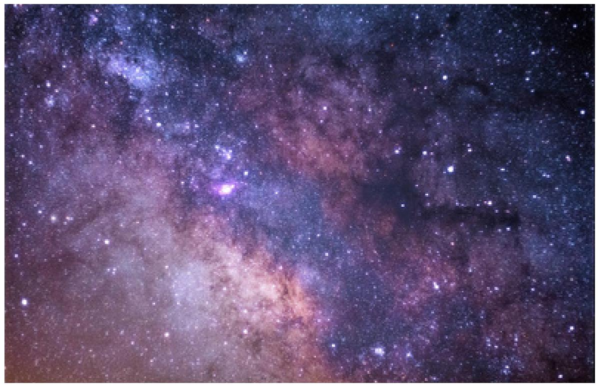
Tour of PyCharm

- Files
- Command Line
- Running the Code in our File
- Adding inputs from terminal
- Bluescreen example

Greenscreen Explanation


An **image** with a **greenscreen**

+


A **cool background**

=

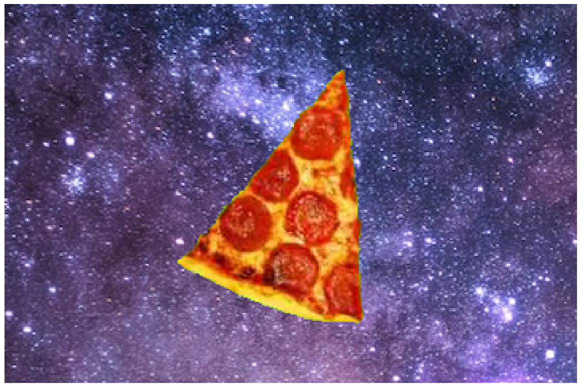
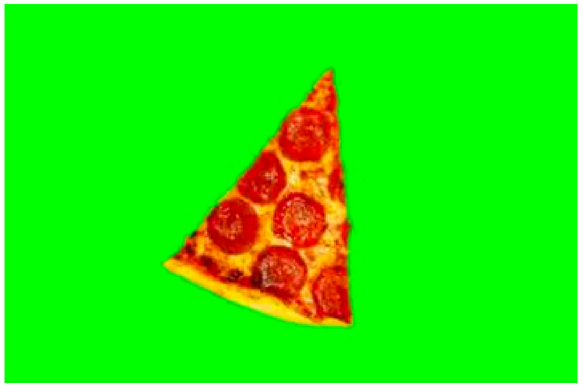

An **image** with a **cool background**

Photo Credits: TLDump, YouTube and Unsplash

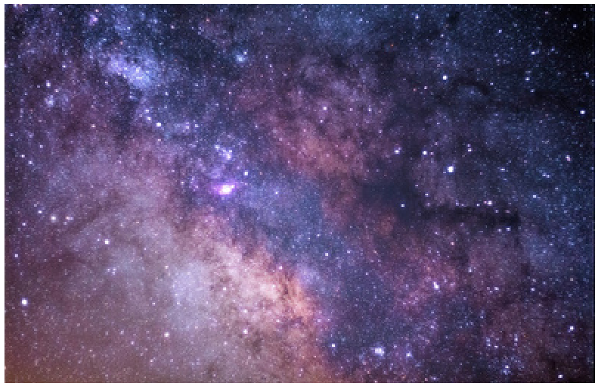
Greenscreen Explanation



An image with a greenscreen

front

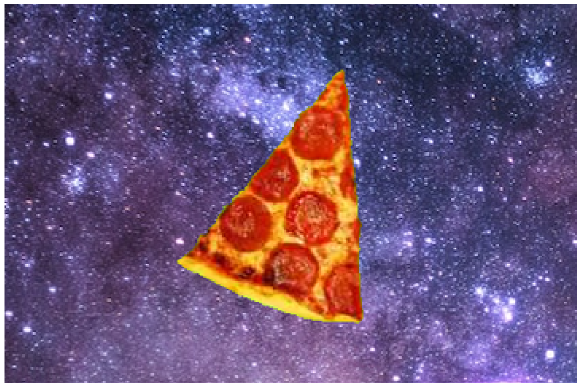
+



A cool background

back

=



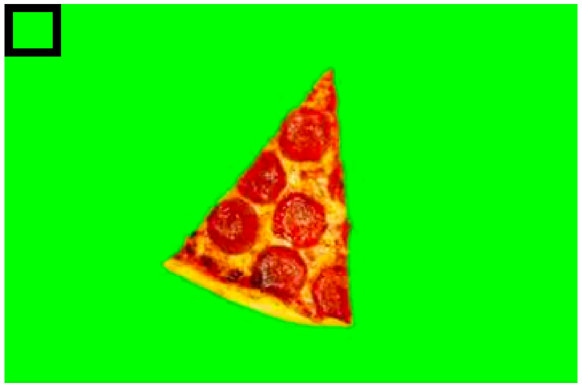
An image with a cool background

front

```
#if front has certain amount of green (supergreen)
    #replace with back(ground) image
```

Photo Credits: TLDump, YouTube and Unsplash

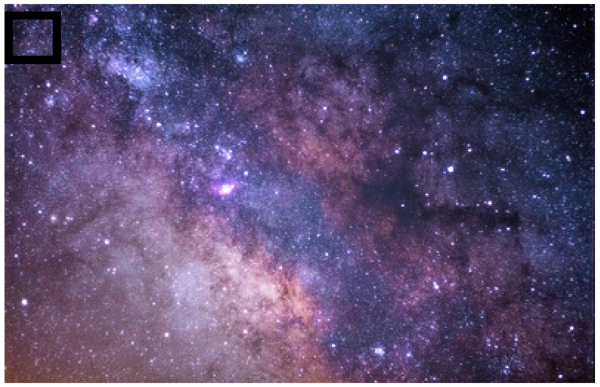
Greenscreen Explanation



An image with a greenscreen

front

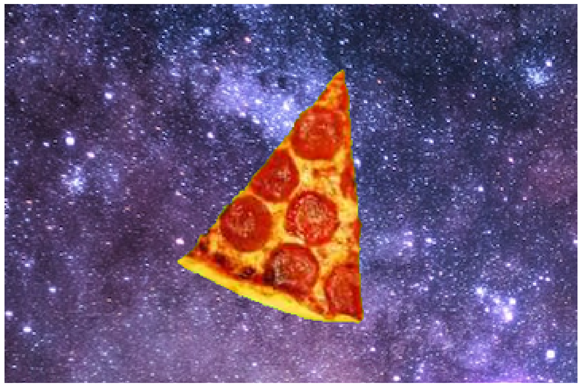
+



A cool background

back

=



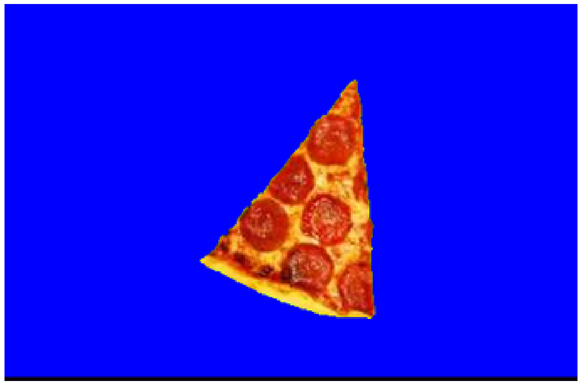
An image with a cool background

front

```
#if front pixel is "supergreen"
    #replace with back pixel
```

Photo Credits: TLDump, YouTube and Unsplash

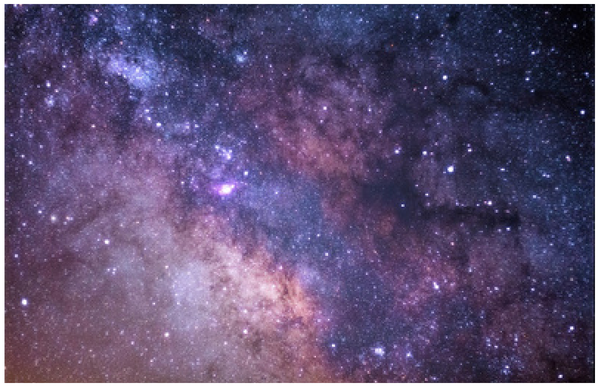
Bluescreen Explanation



An image with a *bluescreen*

front

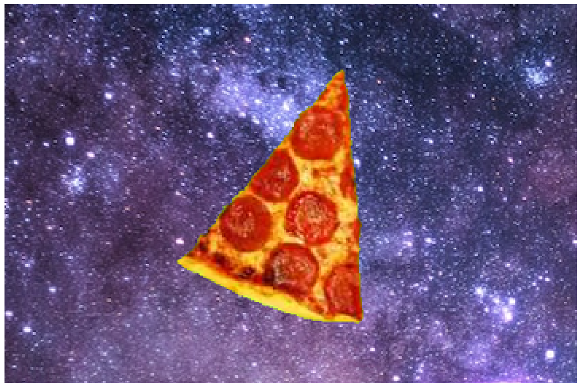
+



A cool background

back

=



An image with a cool background

front

```
#if front pixel is "superblue"
    #replace with back pixel
```

Photo Credits: Unsplash

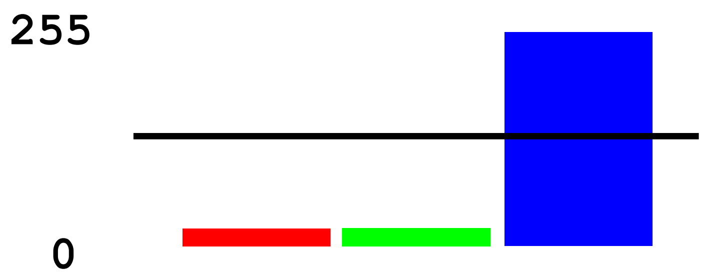
Bluescreen Explanation

What might "superblue" look like?

Bluescreen Explanation

What might "superblue" look like?

Significantly bluer than average, relative to other colors



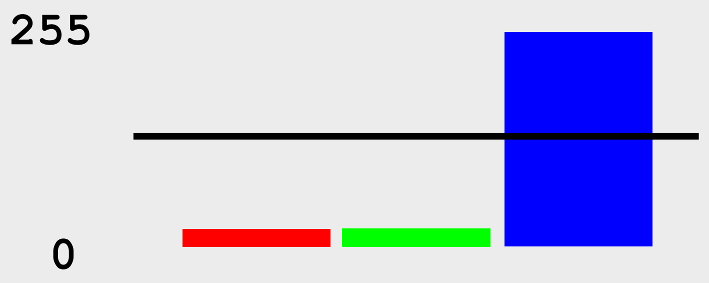
superblue pixel

pixel average: (pixel.red + pixel.blue + pixel.green)//3

Bluescreen Explanation

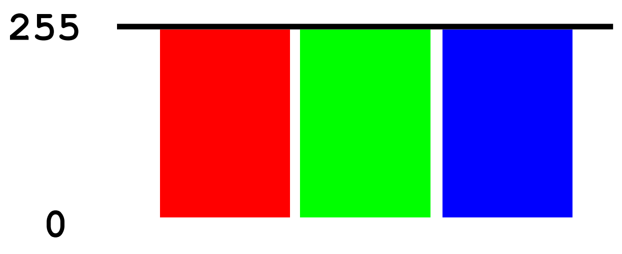
What might "superblue" look like?

Significantly bluer than average, relative to other colors

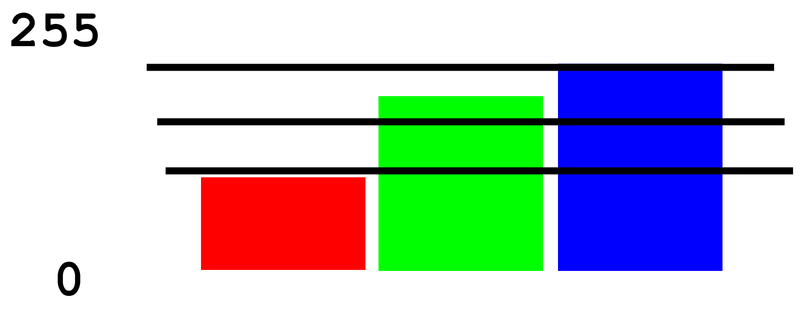


superblue pixel

pixel average: (pixel.red + pixel.blue + pixel.green)//3



white pixel



fine tuning w average

the higher our number, the less blue gets replaced (i.e. it is more selective/more "bluey")

Bluescreen Algorithm

Given we have the two filenames of front and back, how might we create a "bluescreen", filtering for pixels that have an abnormally high amount of blue?

```
Pseudocode of Algorithm
# Front image has special color in back
# Back image is special background

# Loop through front image
    #if front pixel is "superblue"
        #replace with back pixel

# return front image
```

This is called a Chroma Key!

Let's Code it Up!

(Download zip from website!)

Solution

```
front = SimpleImage(front_filename)
back = SimpleImage(back_filename)

for y in range(front.height):
    for x in range(front.width):
        pixel = front.get_pixel(x, y)
        # if front images have more than (weighted) average blue
        avg = (pixel.red + pixel.blue + pixel.green)//3

        # lower average threshold = easier to get rid of blue
        if pixel.blue > avg*0.9: # can manipulate weight
            back_pixel = back.get_pixel(x,y)
            pixel.red = back_pixel.red
            pixel.green = back_pixel.green
            pixel.blue = back_pixel.blue # BACK replaces FRONT

return front # front has been modified
```

Recap

Today, we talked about and learned how to...

Images and Pixels

- create a new, blank image of custom dimensions
- create an out image from an original image
- copying pixel values over
- code aqua stripe (mirror1, mirror2) examples
- make an out image from original image

PyCharm

- venture beyond the experimental server into PyCharm
- how to use the command line
- command line + args
- bluescreen image example

THE END