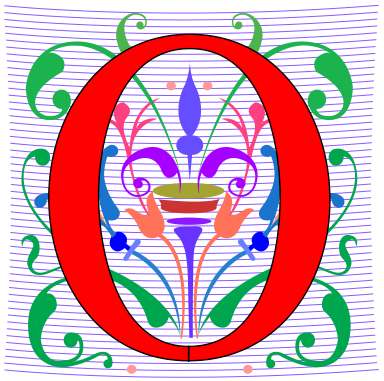
Compsci 101
Images, Tuples

Brandon Lindsey
Susan Rodger
March 3, 2022

O is for …



- **Open Source**
  - Copyright meets the Creative Commons
- **Object Oriented**
  - Using classes and more in programming
- **Occam's Razor**
  - Not just compsci. Simple is good

Cynthia Rudin

- Duke CompSci Professor
  - Univ Buffalo, BS Mathematical Physics, BA Music Theory
  - Princeton, PhD.
- Works in interpretable machine learning, which is crucial for responsible and trustworthy AI
- Winner of Squirrel AI Award for AI for the Benefit of Humanity – 1 million
  - Detecting crime series
  - Con Edison NYC – underground electrical distribution network

She uses AI's power to help society.

Announcements

- APT-4 due tonight
- APT-5 out, due March 24
- Assign 4 due Thursday, March 17

- No Lab Friday

- Have a great Spring Break!

PFTD

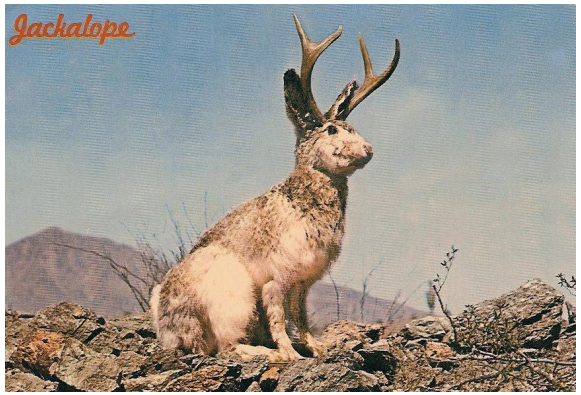
- Images
- Classes and Objects
- Tuples sprinkled about

Images

What is photoshop?

Image Processing

- Convert image into format for manipulating the image
  - Visualization, Sharpening, Restoration, Recognition, Measurement, more
  - Resizing, Red-eye Removal, more
  - CrashCourse: Navigating Digital Info
    - http://bit.ly/dukecs101-cc-images



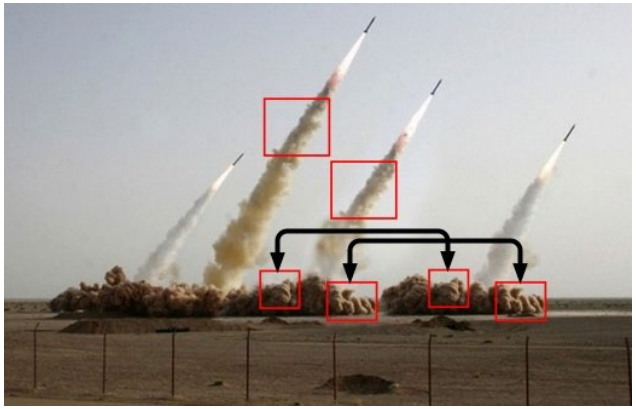


Image Library

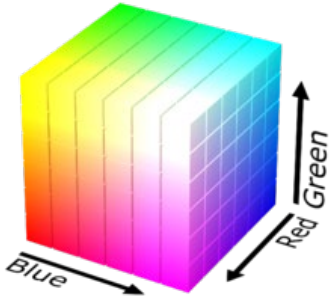
- PIL: Python Image Library -> Pillow
  - To install run the command below in a terminal
    - Terminal in PyCharm, not “Python Console”
    - `pip install Pillow`
      - `If that doesn’t work try:`
      - `Python3 –m pip install Pillow`

- Library has extensive API, far more than we need
  - Concepts often apply to every image library
  - Realized in Python-specific code/functions

Color Models

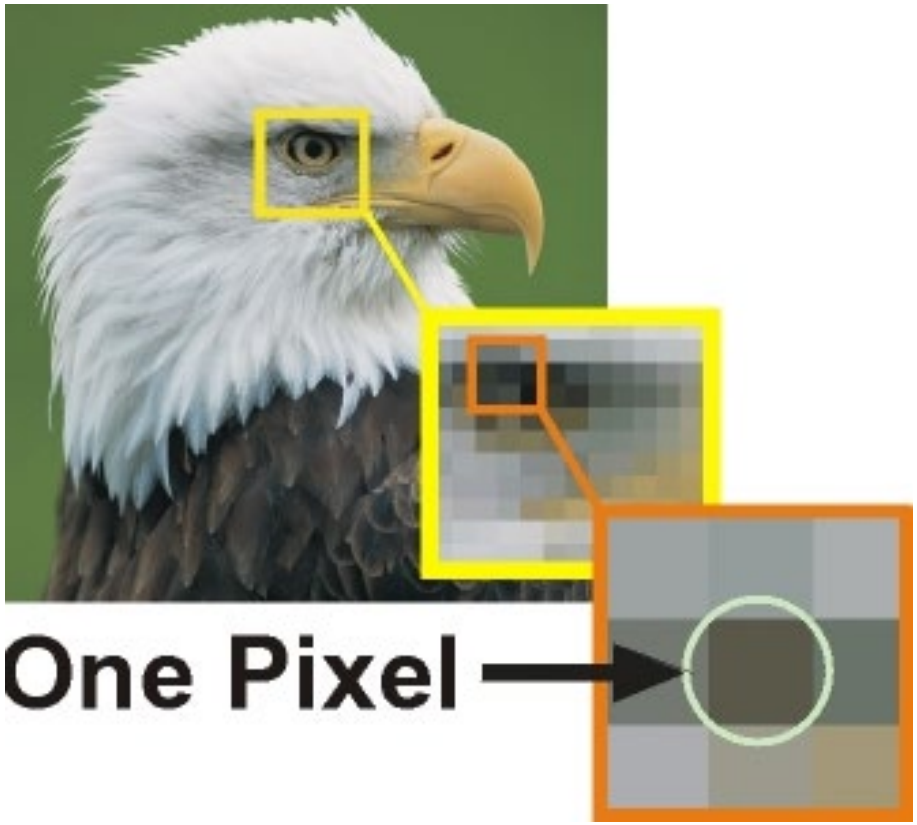
- **Cameras, Displays, Phones, JumboTron: RGB**
  - Additive Color Model: Red, Green, Blue
  - https://en.wikipedia.org/wiki/RGB_color_model

- **Contrast Printers and Print which use CMYK**
  - Subtractive: Cyan, Magenta, Yellow, Key/Black



An image is made up of Pixels

- A pixel is a square of color



Images and Pixels

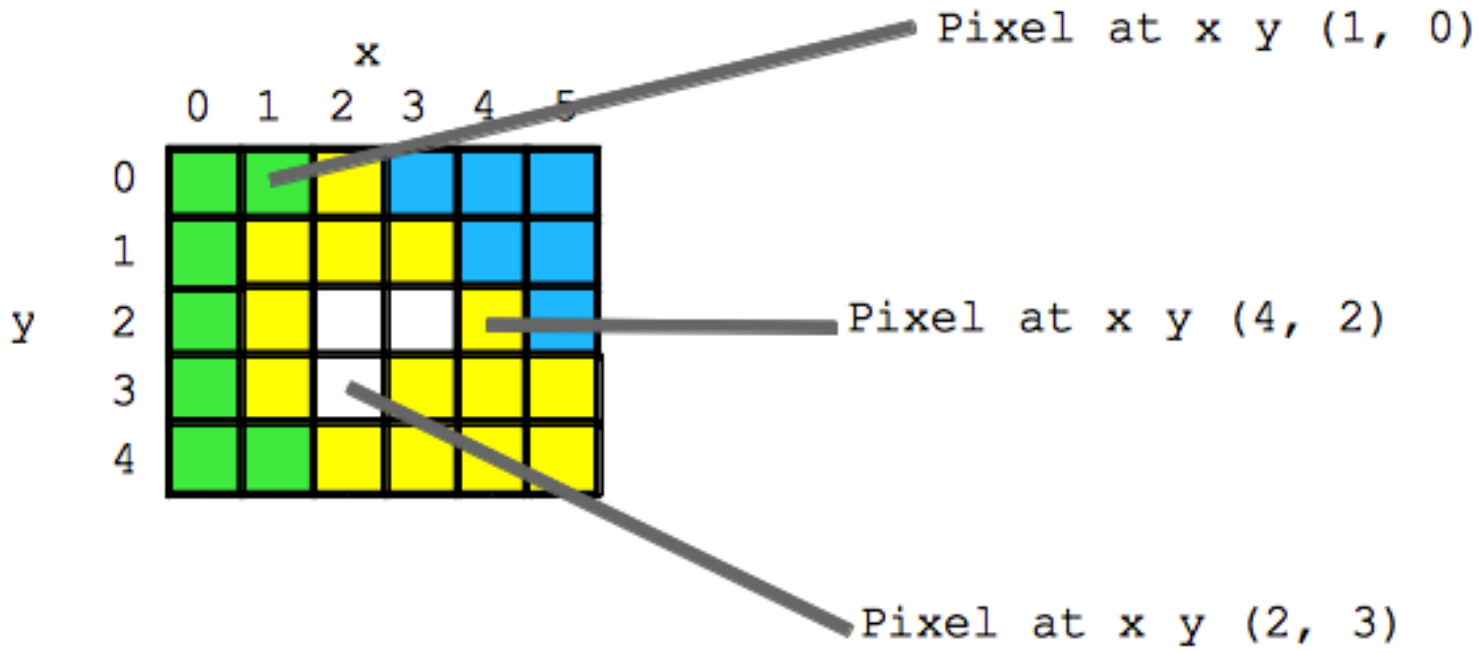
- Image is a collection of pixels
  - Organized in rows: # rows is image height
  - Each row has the same length: image width

- Pixels addressed by (x, y) coordinates
  - Upper-left (0,0), Lower-right (width-1,height-1)
  - Typically is a single (x, y) entity: tuple

- Remember: Tuple is immutable, indexed sequence

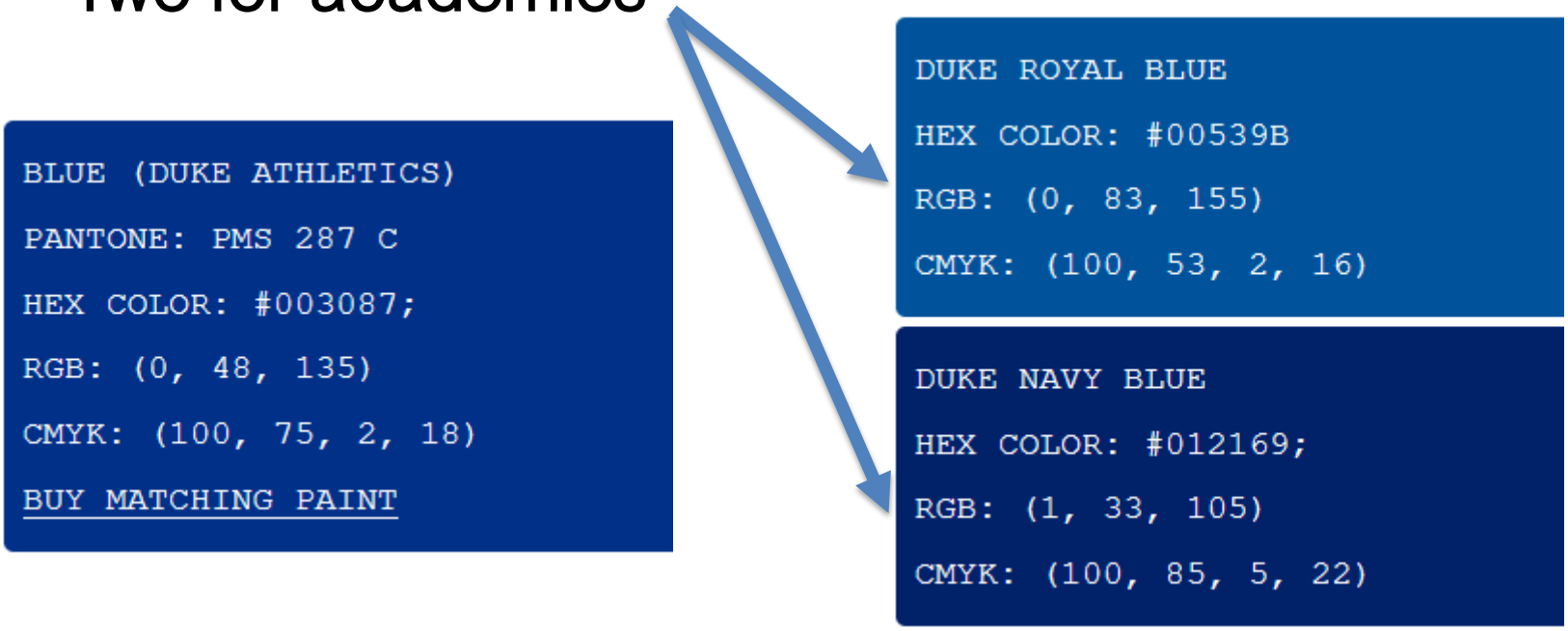
  (a, b, c)

Each pixel has a location in Image



Each pixel has an RGB color

- Duke has three Duke blues
- Duke Athletics RGB: (0, 48, 145)
- Two for academics



SimpleDisplay.py

- Access to PIL and Image module
  - What type is img?
  - https://pillow.readthedocs.io/en/latest/

```
6   from PIL import Image
7
8   if __name__ == '__main__':
9       img = Image.open("images/bluedevil.png")
10      #img = Image.open("images/easterеggs.jpg")
11      img.show()
12      print("width %d height %d" % (img.width, img.height))
```

String formatting with % operator

- Use formatted string with % in string to show where to put values
  - Followed by % and tuple of values
  - %d is for an int
  - %f is for a float
  - %.xf is to specify x digits past the decimal
  - %s is for a string or something that could be shown as a string

String Formatting Examples

```
name = "Xiao"
age = 19
print("%s is %d years old" % (name, age))
alist = [6, 7.8643, 2]
print("%f is a list %s" % (alist[1], alist))
print("fav in %s is %.2f" % (alist, alist[1]))
```

OUTPUT:

WOTO-1 Images
http://bit.ly/101s22-0303-1

What is a class in Python?

- Class ≈ module ≈ library (for this CS101)
- Class – Also blueprint/Factory for creating objects
  - We've used int, float, str
  - **`<class 'int'>, <class 'list'>`**
  - Everything is a class in Python3

- Objects are created from a class
  - x = [5, 6, 7]
  - b = “Moe”
  - c = “Charlotte”

Types

```
print(type(6))
print(type([1,1]))
print(type('blue'))
print(type((6,[7])))

img = Image.open("images/bluedevil.png")
print(type(img))


img = Image.open("images/eastereggs.jpg")
print(type(img))
```

What is a class in Python?

- Use . dot notation to access object's innards
  - **`word = "Hello"`**
    - word is an **object** from the String class
  - **`word.lower()`**
    - **`.lower()`** is a function, but don't call it that!
    - Function that goes with a class is called method
    - **`.lower()`** is a **method** from the String class
  - **`img.width`** is an attribute aka field/property
    - Note there are no ()'s, like a variable

Image Library Basics

- Library can create/open images in different formats, e.g., .png, .jpg, .gif, …
- Images have properties: width, height, type, color-model, and more (variables associated with class)
  - Functions and fields access these properties, e.g., **`im.width`**, **`im.format`**, and more
- Pixels are formed as triples (255,255,255), (r,g,b)
  - In Python these are tuples: immutable sequence

Types

```
img = Image.open("images/bluedevil.png")
print(type(img.format))
```

```
img = Image.open("images/eastereggs.jpg")
print(type(img.format))
```

WOTO-2 Classes
http://bit.ly/101s22-0303-2

Demo: Convert Color to Gray

Process each pixel
Convert to gray

main

```
36  if __name__ == '__main__':
37      img = Image.open("images/easterеggs.jpg")
38      start = time.process_time()
39      gray_img = grayByPixel(img,True)
40      #gray_img = grayByData(img,True)
41      end = time.process_time()
42      img.show()
43      gray_img.show()
44      print("Time = %1.3f" % (end-start))
```

grayByPixel Function

```
13  def grayByPixel(img, debug=False):
14      width = img.width
15      height = img.height
16      new_img = img.copy()
17      if debug:
18          print("creating %d x %d image" % (width,height))
19      for x in range(width):
20          for y in range(height):
21              (r,g,b) = img.getpixel((x,y))
22              grays = getGray(r,g,b)
23              new_img.putpixel((x,y),grays)
24      return new_img
```

getGray function

```
12  def getGray(r,g,b):
13      gray = int(0.21*r + 0.71*g + 0.07*b)
14      return (gray,gray,gray)
```

WOTO-3 GrayScale
http://bit.ly/101s22-0303-3

Make Gray: Notice the Tuples!

```
13  def grayByPixel(img, debug=False):
14      width = img.width
15      height = img.height
16      new_img = img.copy()
17      if debug:
18          print("creating %d x %d image" % (width,height))
19      for x in range(width):
20          for y in range(height):
21              (r,g,b) = img.getpixel((x,y))
22              grays = getGray(r,g,b)
23              new_img.putpixel((x,y),grays)
```

How does this code make a grey image?

New stuff here, what and where?

Revisiting nested Loops

- What is printed here? y varies first
  - Value of x as inner loop iterates?

```
>>> for x in range(5):
...     for y in range(3):
...         print(x, y)
```

```
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
3 0
3 1
3 2
4 0
4 1
4 2
```

Why is the first column have the number repeated like that?
What if the print became:
`print(y, x)`?

Make Gray cont.

```
13  def grayByPixel(img, debug=False):
14      width = img.width
15      height = img.height
16      new_img = img.copy()
17      if debug:
18          print("creating %d x %d image" % (width,height))
19      for x in range(width):
20          for y in range(height):
21              (r,g,b) = img.getpixel((x,y))
22              grays = getGray(r,g,b)
23              new_img.putpixel((x,y),grays)
```

If stop code halfway, what half of image is gray?

Tuple

Tuple

Nested Loops

Tuple

How many parameters does putpixel have?

Accessing Individual Pixels is Inefficient

- Accessing each one one-at-a-time is inefficient
  - Python can do better "under the hood"

- PIL provides a function **`img.getdata()`**
  - Returns list-like object for accessing all pixels
  - Similar to how file is a sequence of characters
  - Symmetry: **`img.putdata(sequence)`**

Processing all Pixels at Once

- Treat **`img.getdata()`** as list, it's not quite a list
  - Iterable: object use in "`for … in …`" loop

```
27  def grayByData(img, debug=False):
28      pixels = [getGray(r,g,b) for (r,g,b) in img.getdata()]
29      new_img = Image.new("RGB", img.size)
30      new_img.putdata(pixels)
```

Think: An image is 2D and putdata(seq) takes a 1D sequence. How did we get an image?

Hint: What type are the elements in the list comprehension?

Hint: What do we know about the length of that sequence and the sequence putdata(…) needs?

GrayByData

```
27  def grayByData(img, debug=False):
28      pixels = [getGray(r,g,b) for (r,g,b) in img.getdata()]
29      new_img = Image.new("RGB", img.size)
30      new_img.putdata(pixels)
31      if debug:
32          print("created %d x %d gray image" % (img.width,img.height))
33      return new_img
```

Summary of Image functions

- Many, many more
  - http://bit.ly/pillow-image

Image function/method	Purpose
`im.show()`	Display image on screen
`im.save("foo.jpg")`	Save image with filename
`im.copy()`	Return copy of im
`im.getdata()`	Return iterable pixel sequence
`im.load()`	Return Pixel collection indexed by tuple (x,y)

WOTO-4 More on Images
http://bit.ly/101s22-0303-4