Compsci 101, Nested Data

Owen Astrachan
Kristin Stephens-Martinez
March 6, 2018

`N` is for …

- **Nested Loops**
  - All pairs, all pixels, all 2D structures
- **Next**
  - Getting the next "chunk" from a generator
- **Newline**
  - The "\n" in a line

Plan for This Week

- Images as a sequence of pixels and more
  - PIL and Pillow: The Python Image Library
  - Using functions, understanding tradeoffs
  - Preparing to be able to do more

- Concepts inherent in Image library
  - Tuple: immutable but indexable and list-like
  - Generator: too expensive to get all at one time

Transform Aside

- Let's look at ***Vowelizer.decrypt***
  - Global variables
  - When is **`lowerwords`** read? How often?
  - Where is **`novowels`** stored and initialized?
  - How does list comprehension work?

- Taking time to read and understand code
  - Sometimes just calling/using isn't enough

Vowelizer Take-aways

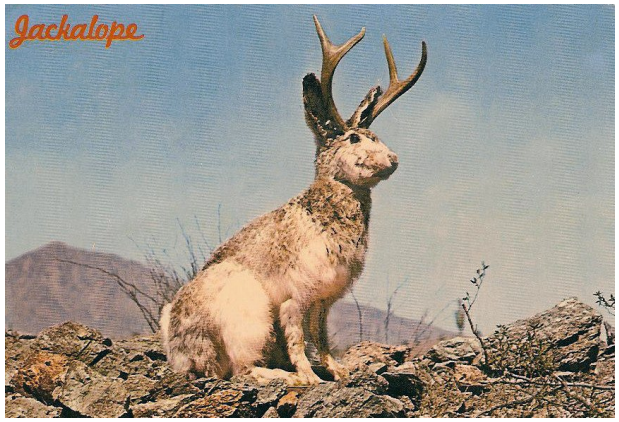
- Global variables can be hard to reason about
  - Where do they get initialized and used?
  - Use the global keyword as a reminder

- List comprehensions are powerful short-cuts
  - Dissect in pieces: sequence, if, element

- Read code and ask questions!

Image Processing

- Convert image into format for manipulating the image
  - Visualization, Sharpening, Restoration, Recognition, Measurement, more
  - Resizing, Red-eye Removal, more



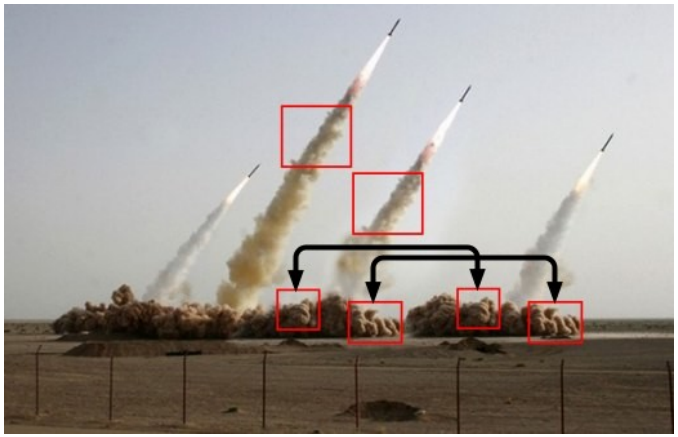


Image Library

- PIL: Python Image Library -> Pillow
  - To install run the command below: terminal
  - 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

March6-Code, 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.show()
11    print("width %d, height %d" % (img.width, img.height))
```

What is a class in Python?

- BluePrint/Code/Factory for creating objects
  - We've used int, float, str
  - **`<class 'int'>, <class list'>`**
  - Changed moving from Python2 to Python3

- Use ./dot notation to access object's innards
  - **`"Hello".lower()`** is function aka a method
  - **`img.width`** is an attribute aka field

Image Library Basics

- Library can create/open images in different formats, e.g., .png, .jpg, .gif, …

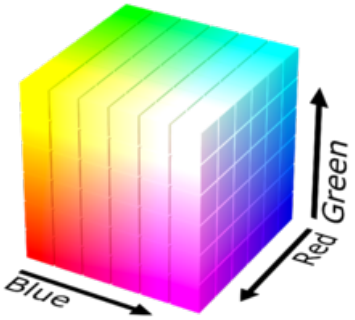
- Images have properties: width, height, type, color-model, and more
  - Functions and fields access these properties, e.g., **`im.width`**, **`im.format`**, and more

- Pixels are formed as triples (255,255,255), (r,g,b)
  - In Python these are tuples: immutable sequence

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



Example: Convert Color to Gray

Process each pixel
Convert to gray

First View of Image for Grayscale

- 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

- Tuple is immutable, indexed sequence (...)

Tuple: What and Why?

- Similar to a list in indexing starting at 0
  - Can store any type of element
  - Can iterate over

- Cannot mutate/change/add
  - Efficient because it can't be altered

- Consider **`x = (5,6)`** and **`y = ([1,2],3.14)`**
  - **`x[0] = 7`** is … **`y[0].append(5)`** is …

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)
24
25     return new_img
```

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
```

Accessing Pixels is Inefficient

- Not quite true: accessing each one one-at-a-time
  - Python can do better "under the hood"
  - Similar to indexing code: check and call index

- 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 of for in statement
  - What type is **`pixels`**? **`img.getdata()`**

```
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.h
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)

Make Green?

- Instead of **`getGray(r,g,b)`** -- getGreen?
  - If pixel isn't green, make it green
    - If red < 200, make pixel (0,200,0)

- How can we change grayByData function?
  - What's similar to Transform?

WOTO

http://bit.ly/101spring18-march1-2

Melissa Groisman, Duke 2001

- English and Compsci
- CMU Information Management
- U. Miami Law School

Ayanga Okpokowuruk, Duke 2009

- Charlotte, NC
- History and Compsci

Eleanor Mehlenbacher, 2016

- IDM Music and Computer Science
- Red Hat
- CSURF: Cloud based peer instruction

Generator

- Generate values one-at-a-time rather all-at-once
  - **`a = [x*x for x in range(1000) ]`**
  - **`b = (x*x for x in range(1000) )`**

- What are types? What are values? How to use?
  - Don't store 1000 ints to access 1000 ints
  - What does that even mean?
  - **`sys.getsizeof(…)`** method

Memory v Runtime

- Tradeoffs are important in some contexts
  - When data are small? Doesn't matter
  - What is Computer Science reminder

```
>>> x = (n*n for n in range(3000))
>>> y = [n*n for n in range(3000)]
>>> print(sys.getsizeof(x))
88
>>> print(sys.getsizeof(y))
26736
>>>
```

Comprehensions

- List comprehension
  - `[w for w in words if len(w) > 20]`
- Set comprehension
  - `{w for w in words if len(w) > 20}`
- Generator
  - `(w for w in words if len(w) > 20)`

Under the hood

- Creating a generator with code is straightforward
  - We will see how, but we won't leverage this
  - Efficiency not a large concern in 101
  - We're going along for a ride ☺