

Compsci 101, Nested Data

Owen Astrachan

Kristin Stephens-Martinez

March 1, 2018

L is for ...

- **Loops**
 - While, For, Nested – Iteration!
- **Library**
 - Where we find APIs and Implementations
- **Logic**
 - The Boolean Heart of ...
- **Linux**
 - The OS that runs the world?



M is for ...

- **Machine Learning**
 - Math, Stats, Compsci: learning at scale
- **Microsoft, Mozilla, Macintosh**
 - Software that changed the world?
- **Meme**
 - I don't always use M-words, ...



PFtFDoM

- Transform Big Ideas
 - Main ideas and learning objectives
 - What the program does, what you'll do
- Code that reflects structure: nested lists/loops
 - Files as sequence of lines or characters
 - Lines as sequence of words or characters
 - Images as sequence of pixels

Transform

- Reading and writing files
 - We've seen how to read, writing is similar
 - Open, read, and close
 - Open, write, and close
- Apply a function to every word in a file
 - Respect lines, so saved file has similar structure
 - Rmv vwls and onvert-cay ig-pay English-way

Concepts in Starter Code

- Global variables
 - Generally avoided, but often necessary
 - Accessible in all module functions
- FileDialog and tkinter
 - API and libraries for building UI and UX
 - Using these is often fraught with ...
- Docstrings for understanding!

Global Variables

- Access, modifying/writing, using/reading
 - Accessible in each function, not parameters
 - Initialized at beginning or in main program block
- Indicate variable is ***global*** within each function
 - Required if global variable modified/changed
 - Good practice to do always: code reading

Tkinter and FileDialog

- This library and API is useful for creating GUIs
 - Difficult and not always about the big picture
 - Debugging can be frustrating
 - Tedium of making things right versus exultation in creating wonderful programs!
- If you don't see the rocket-ship? Oops
 - What happens when you run Transform?

Developing a New Transform

Learning to program can be both enlightening and frustrating

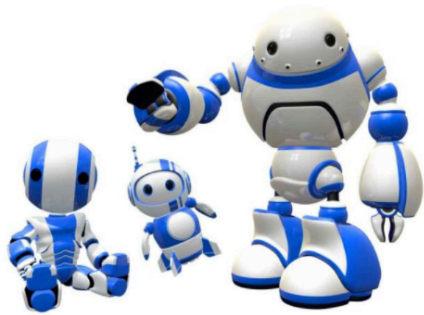
- How do we do this? How do we debug?
 - Write module `Shuffleizer.py` -- function `encrypt`
 - Extra Challenge: write `decrypt`



sh  *ffle*

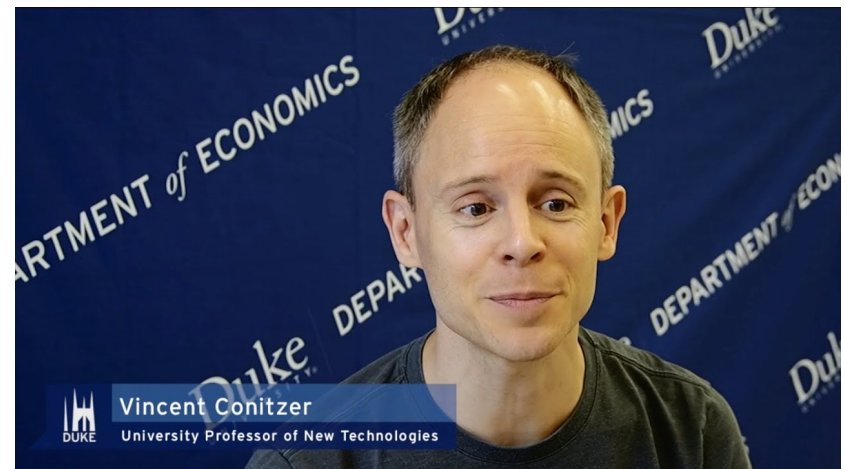
Vince Conitzer

THE EVOLUTION OF AI: CAN MORALITY BE PROGRAMMED?



- Computers and Thought Award
- Computational Social Choice
 - Duke Professor of Philosophy, Economics, Computer Science

To solve these problems, and to help figure out exactly how morality functions and can (hopefully) be programmed into an AI, the team is combining the methods from computer science, philosophy, and psychology “That’s, in a nutshell, what our project is about,” Conitzer asserts.



Code Examples that Follow

- Modules LetterCounter.py, GrayScale.py
 - Folders data (text files) images (images)
- For Eclipse project:
 - <https://www2.cs.duke.edu/courses/spring18/compsci101/code/March1-Code.zip>
- For Browsing:
 - <https://www2.cs.duke.edu/courses/spring18/compsci101/code/march1/>

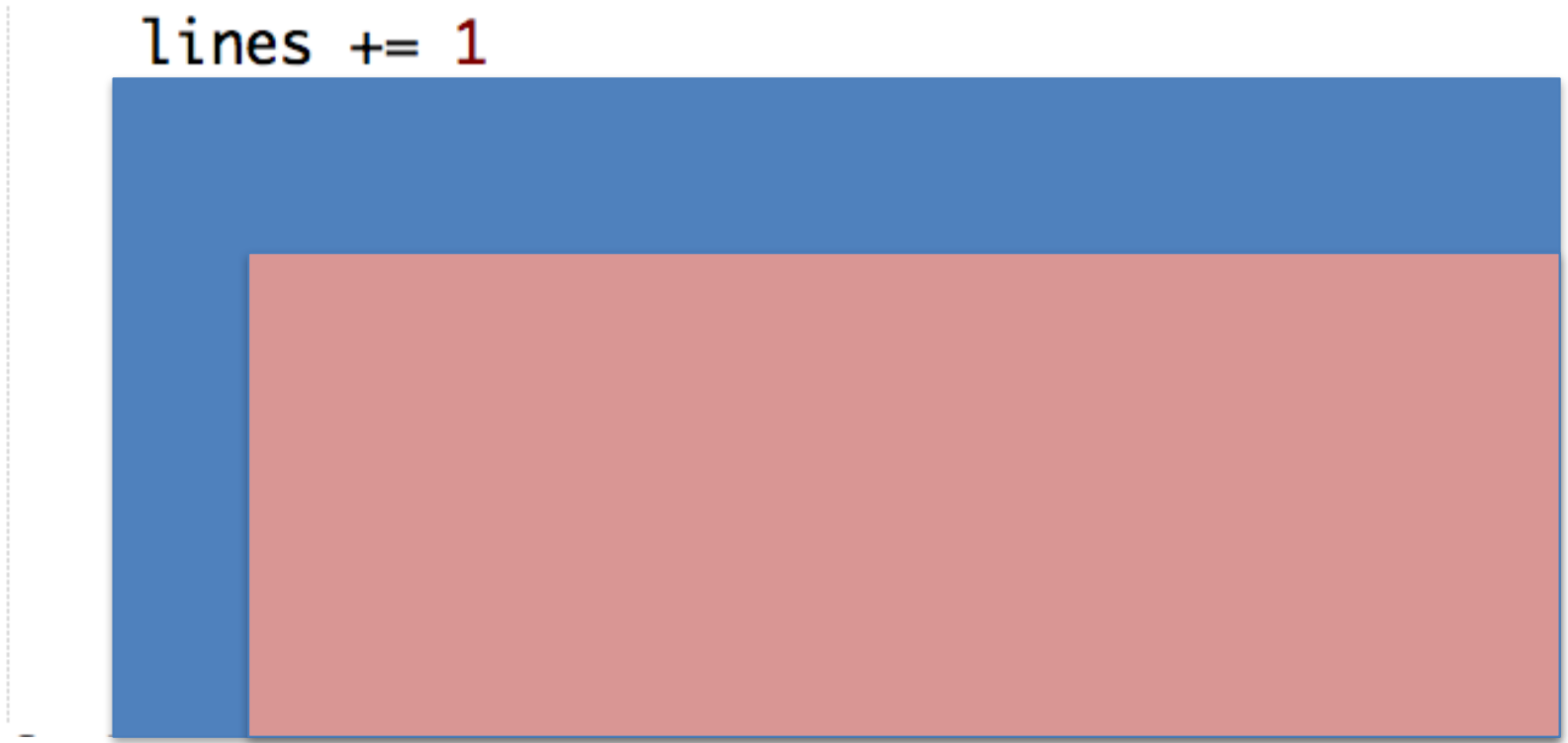
What is a File?

- It depends! Which is often a good answer
 - On what does it depend?
- A text file is a sequence of lines
 - A line is a sequence of words
 - A word is a sequence of characters
 - A character is a sequence of 0's and 1's
- Alternatively: a text file is a sequence of characters
 - Some characters delineate lines, some words



Counting Lines, Words, Chars

```
14 alph = "abcdefghijklmnopqrstuvwxyz"
15 for line in f:
16     lines += 1
17
18
19
20
21
22
23
24
```



LetterCounter.py

- When analyzing or constructing nested loops
 - Each inner loop finishes completely before next iteration of out loop
 - Think of inner loop as a "helper function": call it, when it's done, continue
- Inner-most loop "finishes", then starts again on next iteration of outer loop
 - Doesn't "remember" that it executed before

File is a sequence of characters

- How do we know where line ends? Word ends?

```
35     alph = "abcdefghijklmnopqrstuvwxyz"
36     for ch in f.read():
37         ch = ch.lower()
38         if ch == '\n':
39             lines += 1
40         elif ch.isalpha():
41             letters[alph.index(ch)] += 1
42         else:
43             letters[-1] += 1
..
```

Advantages of Views?

- How many words: "**the big brown fox**"
 - How many characters?
 - When string split to list on whitespace, ...
- Ultimately all files are bit-sequences: 0's and 1's
 - Rarely is this an ideal way to view the file
 - Perhaps for compression?
 - Can 0101010111111 be music? Video? Text?

Moving Between Views

- `f.readlines()` returns list of strings, one/line
`["this is a test\n", "it is only a test\n"]`
 - Same as read by code: `for line in f:`
 - Each string has a `"\n"` newline character at end

- What are these list comprehensions?

```
x = [w.split() for w in f.readlines()]
```

```
y = sum([len(zz) for zz in x])
```

```
z = sum([' '.join(zz) for zz in x])
```

WOTO

<http://bit.ly/101spring18-march1-1>

THE -FILES

Image Processing

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

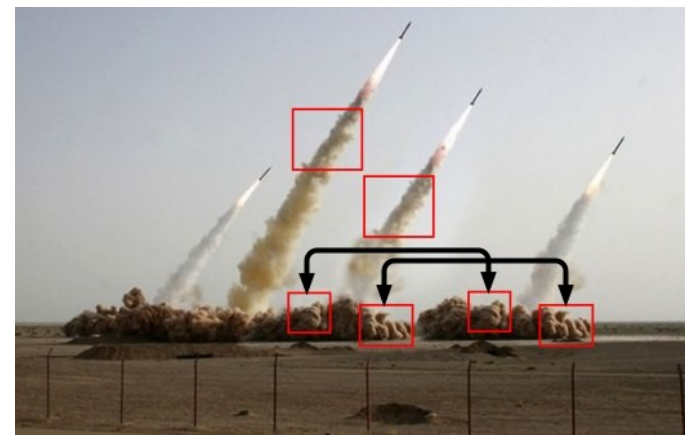
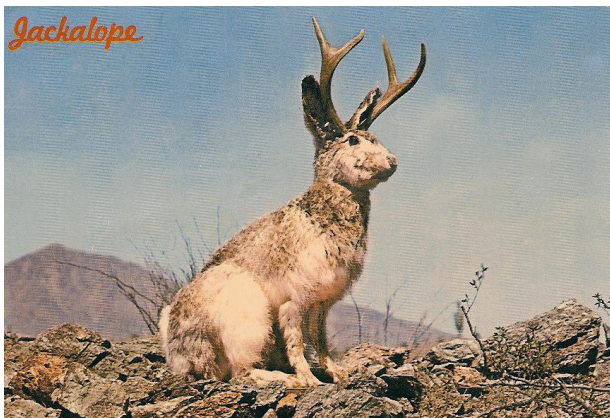


Image Library

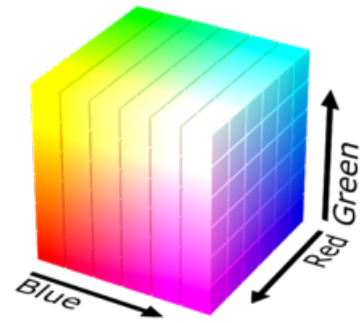
- PIL: Python Image Library -> Pillow
 - `python3 -m pip install pillow`
- Library has extensive API, far more than we need
 - Concepts often apply to every image library
 - Realized in Python-specific code/functions

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
```

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 statement, by what type?
 - In code below what type is pixels?

```
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.l
33     return new_img
```

Summary of Image functions

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

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

Make Green?

- Instead of `getGray(r,g,b)` what about `getGreen`?
 - If pixel isn't green, make it green
 - If $\text{red} < 200$, make pixel $(0,200,0)$
- How can we change `grayByData` function?
 - What's similar to `Transform`?



WOTO

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

