Compsci 101
Images, Tuples

Susan Rodger
October 25, 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 Thursday, Oct 27
- Assign 4 due Thursday, November 3
- Prelab 7 – do before lab this week
  - Some of it is practice for the upcoming exam
- Exam 2 in one week!
Exam 2 – in person – Tues, Nov 1

- Exam is in class on paper – 10:15am
  - Need pen or pencil
- See materials under 11/1 date
  - Exam 2 Reference sheet - part of exam
- Covers
  - topics /reading through today
  - APTs through APT4
  - Labs through Lab 7 (Parts 1 and 2)
  - Assignments through Assignment 3, parts of Assign 4 helpful

Exam 2 topics include …

- List, tuples, list comprehensions
- Loops – for loop, while loop, indexing with a loop
- Reading from a file
  - Converting data into a list of things
- Parallel lists
- Sets – solving problems
- Dictionaries – only reading them and understanding output, no problem solving
- No turtles, no images - but note we are practicing other concepts with images

Exam 2

- Exam 2 is your own work!
- No looking at other people’s exam
- You cannot use any notes, books, computing devices, calculators, or any extra paper
- Bring only a pen or pencil
- The exam has extra white space and has the Exam 2 reference sheet as part of the exam.

- Do not discuss any problems on the exam with others until it is handed back
Exam 2 – How to Study

• Practice writing code on paper!
• Rewrite an APT
• Try to write code from lecture from scratch
• Try to write code from lab from scratch
• Practice from old exams
• Put up old Sakai quizzes, but better to practice writing code
• Look at Exam 2 reference sheet when writing code!

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

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

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

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 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?

```python
from PIL import Image

if __name__ == '__main__':
    img = Image.open("images/bluedevil.png")
    img.show()
    print("type is: ", type(img))
    print("width %d height %d" % (img.width, img.height))
```

OUTPUT:

```
type is: <class 'PIL.PngImagePlugin.PngImageFile'>
width 397 height 337
```

String formatting with % operator

- Use formattted 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:
Xiao is 19 years old
7.864300 is a list [6, 7.8643, 2]
fav in [6, 7.8643, 2] is 7.86

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”
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"`

Objects are created from a class

```
x is a list object from the <class 'list'>
b and c are string objects from the <class 'str'>
```

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 a method
  - `img.width`
    - `img.width` is an attribute aka field/property
  - Note there are no ()’s, like a variable
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

```
# 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
```

Image Library Basics

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

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

Types

```
**Demo: Convert Color to Gray**

Process each pixel
Convert to gray

```
main

```if __name__ == '__main__':
    img = Image.open("images/easter_eggs.jpg")
    start = time.process_time()
    gray_img = grayByPixel(img,True)
    #gray_img = grayByData(img,True)
    end = time.process_time()
    img.show()
    gray_img.show()
    print("Time = %1.3f" % (end-start))

grayByPixel Function

```def grayByPixel(img, debug=False):
    width = img.width
    height = img.height
    new_img = img.copy()
    if debug:
        print("creating %d x %d image" % (width,height))
    for x in range(width):
        for y in range(height):
            (r,g,b) = img.getpixel((x,y))
            grays = getGray(r,g,b)
            new_img.putpixel((x,y),grays)
    return new_img```
getGray function

```python
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

Make Gray: Notice the Tuples!

```python
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?

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

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

Make Gray cont.

```python
def grayByPixel(img, debug=False):
    width = img.width
    height = img.height
    new_img = img.copy()
    if debug:
        print("creating %d x %d image" % (width,height))
    for x in range(width):
        for y in range(height):
            (r,g,b) = img.getpixel((x,y))
            grays = getGray(r,g,b)
            new_img.putpixel((x,y),grays)
```

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

- 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

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

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?
Summary of Image functions

- Many, many more

<table>
<thead>
<tr>
<th>Image function/method</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>im.show()</td>
<td>Display image on screen</td>
</tr>
<tr>
<td>im.save(&quot;foo.jpg&quot;)</td>
<td>Save image with filename</td>
</tr>
<tr>
<td>im.copy()</td>
<td>Return copy of im</td>
</tr>
<tr>
<td>im.getdata()</td>
<td>Return iterable pixel sequence</td>
</tr>
<tr>
<td>im.load()</td>
<td>Return Pixel collection indexed by tuple (x,y)</td>
</tr>
</tbody>
</table>