Compsci 101
Images, Tuples, Sets
Part 1 of 2

Susan Rodger
March 11, 2021

N is for …

• Nested Loops
  • All pairs, all pixels, all 2D structures
• None
  • Default value for functions if no return
• Newline
  • The "\n" in a line

PFTD

• Images & Tuples cont.
• Sets and APTs

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 (a, b, c)

Let's run it first!

---

grayByPixel Function

```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)
    return new_img
```

getGray function

```python
def getGray(r, g, b):
    gray = int(0.21*r + 0.71*g + 0.07*b)
    return (gray,gray,gray)
```
If you name __ == '__main__':
    img = Image.open("images/easter.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))

Richard Stallman

- MacArthur Fellowship (Genious grant)
- ACM Grace Murray Hopper award
- Started GNU – Free Software Foundation (1983)
  - GNU Compiler Collection
  - GNU Emacs

Python Sets

- Set – unordered collection of distinct items
  - Unordered – can look at them one at a time, but cannot count on any order
  - Distinct - one copy of each
- Operations on sets:
  - Modify: add, clear, remove
  - Create a new set: difference(-), intersection(&), union (|), symmetric_difference(^)
  - Boolean: issubset <=, issuperset >=
- Can convert list to set, set to list
  - Great to get rid of duplicates in a list
List vs Set

- **List**
  - Ordered, 3rd item, can have duplicates
  - Example: \( x = [4, 6, 2, 4, 5, 2, 4] \)
- **Set**
  - No duplicates, no ordering
  - Example: \( y = \text{set}(x) \)
- **Both**
  - Add, remove elements
  - Iterate over all elements

List and Set, Similarities/Differences

<table>
<thead>
<tr>
<th>Function for List</th>
<th>Function for Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding element</td>
<td>( x\text{.append}(elt) )</td>
</tr>
<tr>
<td>Size of collection</td>
<td>( \text{len}(x) )</td>
</tr>
<tr>
<td>Combine collections</td>
<td>( x + y )</td>
</tr>
<tr>
<td>Iterate over</td>
<td>( \text{for elt in } x: )</td>
</tr>
<tr>
<td>Element membership</td>
<td>( \text{elt in } x )</td>
</tr>
<tr>
<td>Index of an element</td>
<td>( x\text{.index}(elt) )</td>
</tr>
</tbody>
</table>

- Lists are ordered and indexed, e.g., has a first or last
- Sets are **not** ordered, very fast, e.g., \( \text{if elt in } x \)
Creating and changing a set

```python
colorList = ['red', 'blue', 'red', 'red', 'green']
colorSet = set(colorList)
smallList = list(colorSet)
colorSet.clear()
colorSet.add("yellow")
colorSet.add("red")
colorSet.add("blue")
colorSet.add("yellow")
colorSet.remove("yellow")
```

```python
smallList = ['red', 'green', 'blue']
colorSet = set(["purple", "red", "blue"])"order?
```

Set Operations – Union and Intersection

```python
UScolors = set(["red", "white", "blue"])
dukeColors = set(["blue", "white", "black"])
```

```python
print dukeColors | UScolors
print dukeColors & UScolors
```

Set Operations - Difference

```python
UScolors = set(["red", "white", "blue"])
dukeColors = set(["blue", "white", "black"])
```

```python
set(['blue', 'black', 'white', 'red'])
set(['blue', 'white'])
```
Set Operations - Difference

UScolors = set(['red', 'white', 'blue'])
dukeColors = set(['blue', 'white', 'black'])

print dukeColors - UScolors
print UScolors - dukeColors

set(['black'])
set(['red'])

Set Operations – Symmetric Difference

UScolors = set(['red', 'white', 'blue'])
dukeColors = set(['blue', 'white', 'black'])

print dukeColors ^ UScolors
print UScolors ^ dukeColors

Python Set Operators

- Using sets and set operations often useful
- A | B, set union
  - Everything
- A & B, set intersection
  - Only in both
- B – A, set difference
  - In B and not A
- A ^ B, symmetric diff
  - Only in A or only in B