Problem: Given list of words, find word with most vowels

• Example:
  – Given ['dog', 'cat', 'gerbil', 'elephant']
  – 'elephant' has 3 vowels, the most

• To solve – nested loops:
  – Loop over words in list
    • For each word: Loop over characters in word

---

Announcements

• Reading and RQ13 due next time
• Assignment 5 due Thursday
• APT 5 due on today, APT 6 out and due after spring break

• This week:
  – Nested loops, tuples, images and more with sets

---

```python
def wordWithMostVowels(words):
    maxcnt = 0
    maxword = ""
    cnt = 0
    for word in words:
        for letter in word:
            if isVowel(letter):
                cnt += 1
            if cnt > maxcnt:
                maxcnt = cnt
                maxword = word
    return maxword
```
Problem 2 – Given two lists of names, print a list of pairs of names in which the two names are the same length

• A = ['mo', 'ted', 'bill']
• B = ['billie', 'jes', 'bo']

To solve
– for name in A:
  for name in B:
    Check length
    print pair

mo, bo
ted jes

Tuples

• Like a list, but cannot change them
  – Define them with “,”,
    (5, 7, 8) or 5, 7, 8
• Use most list operations on them
  – they are a type of list
  – But immutable
• Examples

Example

x = (4, 6, 8)
y = 9, 5, 6
print x
print y
print x[1]
print y[1]
y[0] = 2
z = ([5,6], [7,8])
print z

print z
z[0][1] = 12
print z
z[0].append(4)
print z
z[0].remove(5)
z[0].remove(12)
z[0].remove(4)
print z
print z
Crossword Plagiarism

bit.ly/crossword-0308 - from fivethirtyeight.com

**EXAMPLE OF "SHADOY"**

Answers in white are the same.

<table>
<thead>
<tr>
<th>Original Publication</th>
<th>Repeater</th>
<th>No. of Puzzles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>USA Today</td>
<td>537</td>
</tr>
<tr>
<td>USA Today</td>
<td>Universal</td>
<td>102</td>
</tr>
<tr>
<td>New York Times</td>
<td>Universal</td>
<td>04</td>
</tr>
<tr>
<td>New York Times</td>
<td>USA Today</td>
<td>28</td>
</tr>
<tr>
<td>Chicago Tribune</td>
<td>Universal</td>
<td>15</td>
</tr>
<tr>
<td>Los Angeles Times</td>
<td>USA Today</td>
<td>14</td>
</tr>
<tr>
<td>Los Angeles Times</td>
<td>Universal</td>
<td>12</td>
</tr>
</tbody>
</table>

Image Processing

• What's real, what's Photoshopped
  – Learn more at http://bit.ly/1Psi0hG, we'll do very basic stuff in class and lab, next assignment too!
Example: convert color to gray scale

![Color to Gray Example]

Process each pixel
Convert to gray

Example: convert blue to green

![Blue to Green Example]

Process each pixel
Convert blue ones to green
Is this like red-eye removal?

Need new concepts and Image library

- Red, Green, Blue color model
  - Triples of (R,G,B) are processed as Python tuples.
  - *Let's study tuples!*

- Images can be very big, what's 4K display?
  - $4,096 \times 2,160 = 8,847,360$ pixels, 8Mb at least
  - Creating huge lists takes up memory
  - Sometimes only need one pixel at-a-time
  - *Let's study generators!*

- Need new concepts and Image library
  - Red, Green, Blue color model
    - Additive model, each pixel specified by (r,g,b) triple, values of each between 0-255
    - White is (255,255,255) and Black is (0,0,0)
  - Images stored as sequence of (r,g,b) tuples, typically with more data/information too
    - 256 values, represented as 8 bits, $2^8 = 256$
    - 32 bits per pixel (with alpha channel)
    - In Python we can largely ignore these details!
Image library: Two ways to get pixels

• Each pixel is a *tuple* in both models
  – Like a list, indexable, but *immutable*
    - pix = (255, 0, 0)
      - What is pix?, pix[0]?, What is pix[5]?
  • Invert a pixel: by subscript or named tuple
    – Access by assignment to variables!
      
      \[
      \text{npx} = (255-\text{pix}[0], 255-\text{pix}[1], 255-\text{pix}[2])
      \]
      
      \[
      (r, g, b) = \text{pix}
      \quad \text{npx} = (255-r, 255-g, 255-b)
      \]

Let's look at GrayScale.py

• Key features we see
  – Import Image library, use API by example
  – Image.open creates an image object

• Image functions for Image object im
  – im.show(), displays image on screen
  – im.save("xy"), saves with filename
  – im.copy(), returns image that's a copy
  – im.load(), [x,y] indexable pixel collection
  – im.getdata(), iterable pixel collection
  • Let's look at two ways to process pixels!

Image Library: open, modify, save

• Image.open can open most image files
  – .png, .jpg, .gif, and more
  – Returns an image object, so store in variable of type Image instance
  – Get pixels with im.getdata() or im.load()

• Image.new can create a new image, specify color model "RGB" and size of image
  – Add pixels with im.putdata()

• These belong to Image package

im.getdata(), accessing pixels

• Returns something *like* a list
  – Use: for pix in im.getdata():
  – Generates pixels on-the-fly, can't slice or index unless you use list(im.getdata())
  – Structure is called a Python generator!
  – Saves on storing all pixels in memory if only accessed one-at-a-time

• See usage in GrayScale.py, note how used in list comprehension, like a list!
Alternate: Still Tuples and Pixels

- The `im.getdata()` function returns list-like iterable
  - Can use in list comprehension, see code
  - Use `.putdata()` to store again in image

```
def makeGray(pixel):
    r,g,b = pixel
    gray = (r+g+b)/3
    return (gray,gray,gray)
```

Making Tuples and Generators

- Overuse and abuse of parentheses
  - To create a tuple, use parentheses
    ```python
    for pix in im.getdata():
        (r,g,b) = pix
        npx = (255-r,255-g,255-b)
    ```
  - To create a generator use parentheses as though creating a list comprehension!
    ```python
    [2*n for n in range(10000)]
    [2*n for n in range(10000)]
    ```

Questions about Image Code

```
bit.ly/101sp16-0308-3
```

`im.load()`, accessing pixels

- Returns something that can be indexed `[x,y]`
  - Only useful for accessing pixels by x,y coords
- Object returned by `im.load()` is ...
  - Use `pix[x,y]` to read and write pixel values
- Note: this is NOT a generator

```
pix = im.load()
tup = pix[0,0]
pix[1,1] = (255,255,0)
```
Lab 7

• You’ll create new images
  – Invert
  – Solarize
  – Darken
  – Brighten
  – etc