Toward understanding images

- Images are stored in several formats
  - .jpg lossy compression
  - .png lossless compression
  - .gif lossless
  - .tiff everything?

- For viewing and manipulating, pixels are R,G,B
  - image is a collection of pixels, (R,G,B), tuples in Python
  - rectangular images have width and height
  - use the Python Image library to manipulate these, lab8

Removing red from an image

- Using Image library in a simple way

```python
import Image
im = Image.open("vase.png")
im = im.convert("RGB")
pix = [(r,g,b) for (r,g,b) in im.getdata()]
im.putdata(pix)
im.show()
```

- What does this sequence of image operations do?
  - What are advantages of using the list comprehension?
  - Disadvantages?

Hiding in plain sight

- How can we "hide" information in an image
  - Why would we want to do this?

- Why is black specified by (255,255,255)?
  - 8-bits per pixel, 00000000, 00000001, 00000010, ...
  - Suppose we had 3-bppp, what are values?
  - Binary numbers: 0-7

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
</tbody>
</table>

- How much can we store in low-order 2-bits of 8pp?

Digits and Bits and Bytes, Oh My!

- 255 in binary is 11111111, 252 in binary is 11111100
  - This is 4*(255/4) = 4*63, now low order bits are clear!

```python
x = 3 # hide this
y = cleared(255) + 3  ➞  255
y = (255 >> 2) << 2
```

- 255 shifted right 2: 11111111 >> 2 is 00111111
- Then shift this left, 00111111 << 2 is 11111100

- A pixel is 32-bits, 4 8-bit values, R,G,B, and alpha
  - Do we really have to know this?