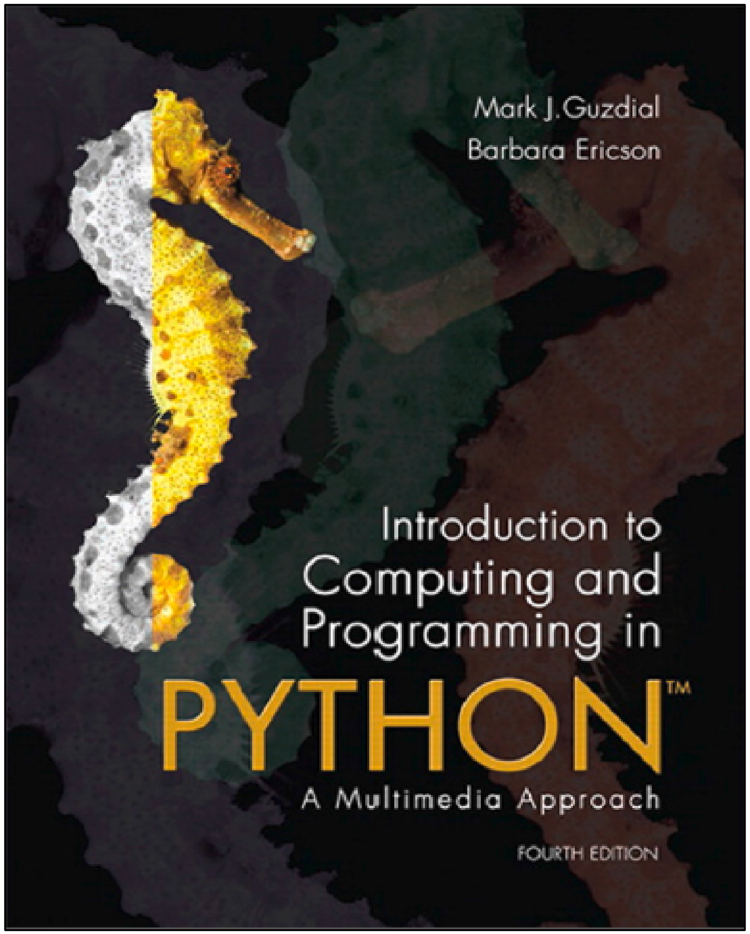
Introduction to Computing and Programming in Python™: A Multimedia Approach

Fourth Edition



Chapter 4

Modifying Pictures Using Loops

Learning Objectives (1 of 2)

4.1 To understand how images are digitized by taking advantage of limits in human vision.

4.2 To identify different models for color, including RGB, the most common one for computers.

4.3 To manipulate color values in pictures, like increasing or decreasing red values.

4.4 To convert a color picture to grayscale, using more than one method.

4.5 To negate a picture.

Learning Objectives (2 of 2)

4.6 To use a matrix representation in finding pixels in a picture.

4.7 To use the objects **pictures** and **pixels**.

4.8 To use iteration (with a for loop) for changing the color values of pixels in a picture.

4.9 To nest blocks of code within one another.

4.10 To choose between having a function **return** a value and just providing a **side effect**.

4.11 To determine the **scope** of a variable name.

We Perceive Light Different from How it Actually is

- Color is continuous
  - Visible light is in the wavelengths between 370 and 730 nanometers
    - That's 0.00000037 and 0.00000073 meters
- But we **perceive** light with color sensors that peak around 425 nm (blue), 550 nm (green), and 560 nm (red).
  - Our brain figures out which color is which by figuring out how much of each kind of sensor is responding
  - One implication: We perceive two kinds of "orange" — one that's **spectral** and one that's red + yellow (hits our color sensors just right)
  - Dogs and other simpler animals have only two kinds of sensors
    - They **do** see color. Just **less** color.

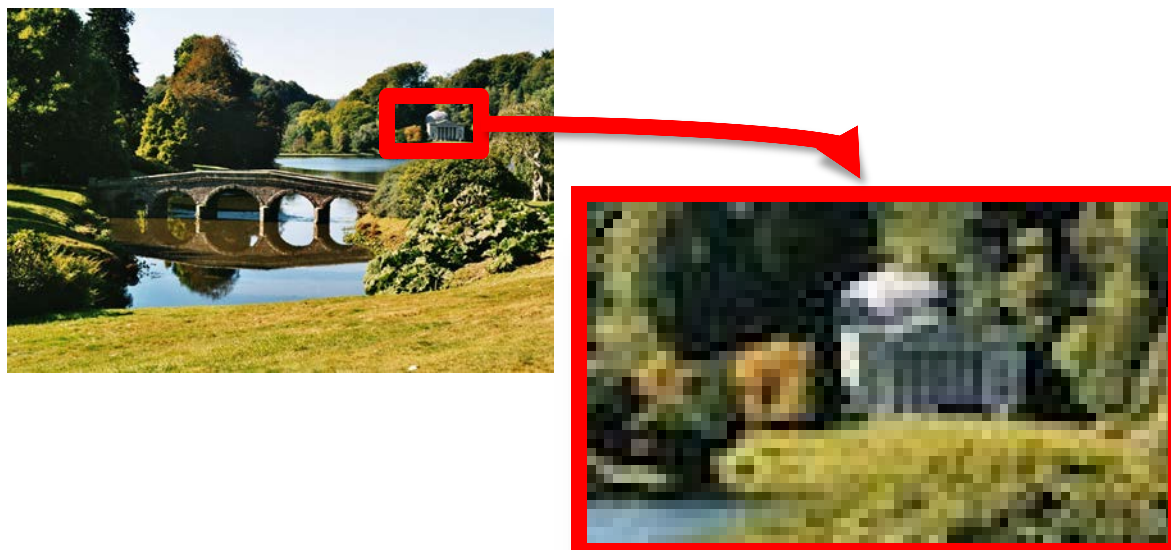
Luminance Vs. Color

- We perceive **borders** of things, motion, depth via **luminance**
  - Luminance is **not** the amount of light, but our **perception** of the amount of light.
  - We see blue as “darker” than red, even if same amount of light.
- Much of our luminance perception is based on comparison to backgrounds, not raw values.

Luminance is actually **color blind**. Completely different part of the brain does luminance vs. color.

Digitizing Pictures as Bunches of Little Dots

- We digitize pictures into lots of little dots
- Enough dots and it looks like a continuous whole to our eye
  - Our eye has limited resolution
  - Our background/depth **acuity** is particularly low
- Each picture element is referred to as a **pixel**

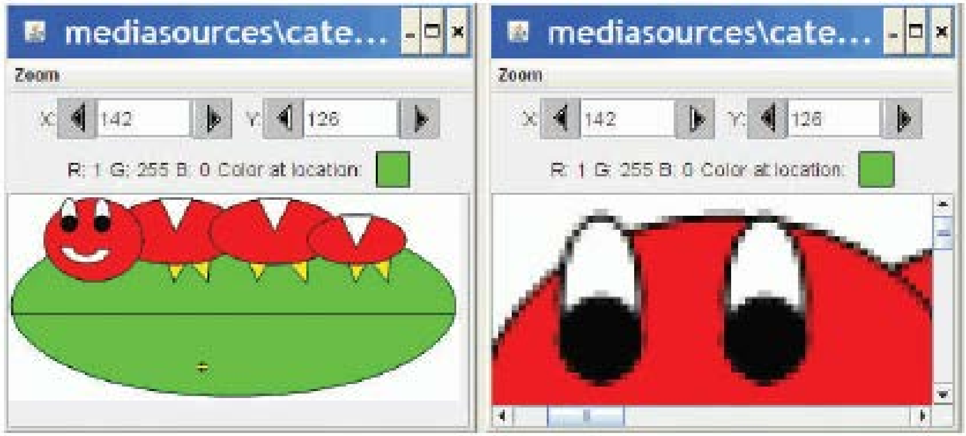


Pixels

- Pixels are **picture elements**
  - Each pixel object knows its **color**
  - It also knows where it is in its **picture**

```
>>> file = "c:/ip-book/mediasources/caterpillar.jpg"
>>> pict = makePicture(file)
>>> explore(pict)
```

When we zoom the picture to 500%, we can see individual pixels.



A Picture is a Matrix of Pixels

- It’s not a continuous line of elements, that is, an **array**
- A picture has two dimensions: Width and Height
- We need a two-dimensional array: a **matrix**

0	1	2	3
15	12	13	10

	0	1	2	3
0	15	12	13	10
1	9	7	2	1
2	6	3	9	10

Referencing a Matrix

- We talk about positions in a matrix as (x,y), or (horizontal, vertical)
- Element (1,0) in the matrix at left is the value 12
- Element (0,2) is 6

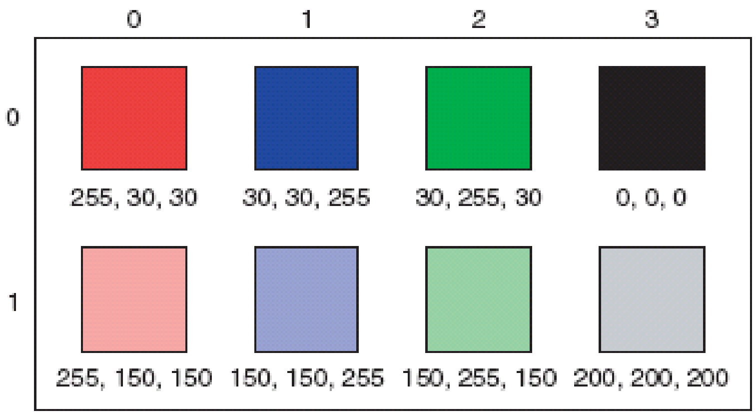
	0	1	2	3
0	15	12	13	10
1	9	7	2	1
2	6	3	9	10

Encoding Color

- Each pixel encodes color at that position in the picture
- Lots of encodings for color
  - Printers use CMYK: Cyan, Magenta, Yellow, and blacK.
  - Others use HSB for Hue, Saturation, and Brightness (also called HSV for Hue, Saturation, and Value)
- We’ll use the most common for computers
  - RGB: Red, Green, Blue

Encoding RGB

- Each component color (red, green, and blue) is encoded as a single byte
- Colors go from (0,0,0) to (255,255,255)
  - If all three components are the same, the color is in greyscale
    - (200,200,200) at (3,1)
  - (0,0,0) (at position (3,0) in example) is black
  - (255,255,255) is white



How Much Can We Encode in 8 Bits?

- Let's walk it through.
  - If we have one bit, we can represent two patterns: 0 and 1.
  - If we have two bits, we can represent four patterns:

    00, 01, 10, 11.
  - If we have three bits, we can represent eight patterns:

    $000, 001, 010, 011, 100, 101, 110, 111$
- General rule: In *n* bits, we can have 2^n patterns
  - In 8 bits, we can have 2^8 patterns, or 256
  - If we make one pattern 0, then the highest value we can represent is $2^8 - 1$ or 255

Is That Enough?

- We're representing color in $24(3*8)$ bits.
  - That's $16{,}777{,}216\ (2^{24})$ possible colors
  - Our eye can discern millions of colors, so it's probably pretty close
  - But the real limitation is the physical devices: We don't get 16 million colors out of a monitor
- Some graphics systems support 32 bits per pixel
  - May be more pixels for color, or an additional 8 bits to represent 256 levels of **translucence**

Size of Images

	320 x 240 image	640 x 480 image	1024 x 768 monitor
24 bit color	230,400 bytes	921,600 bytes	2,359,296 bytes
32 bit color	307,200 bytes	1,228,800 bytes	3,145,728 bytes