Images
Chris Gregg
Based on slides by Chris Piech and Mehran Sahami
CS106A, Stanford University
• Final Reminder: Diagnostic is on Thursday
  – Takes place during class time
  – Please download BlueBook software well before the exam
  – Like an exam, but meant for you to gauge your understanding
    • It's also 10% of your course grade
  – Covers material through this past Thursday
  – Email Wil if you have a time conflict or are outside the Americas
• Handout #8: Image Reference Guide
  – We'll be talking through a lot of that today

• Handout #9: Assignment #3
  – Will be posted tomorrow (July 7th)
  – Due next week – start early!
  – More complex than Assignment #2
Global Variables: Bad Style

# Constant – visible to all functions
NUM_DAYS_IN_WEEK = 7

# Global variable – visible to all functions
balance = 0

def main():
    balance = int(input("Initial balance: "))
    while True:
        amount = int(input("Deposit (0 to quit): "))
        if amount == 0:
            break
        deposit(amount)

def deposit(amount):
    balance += amount

• Also, really BAD style
  – So bad, that Python won't even let you do it unless you basically add a command that says "I want to have bad style"
  – I'm not going to show you that command in Python
    – But, if you know it already, DON'T use it!
  – We're in polite company
Using Parameters: Good Style

Don't want using your toaster to impact your refrigerator!

```python
def main():
    balance = int(input("Initial balance: "))
    while True:
        amount = int(input("Deposit (0 to quit): "))
        if amount == 0:
            break
        balance = deposit(balance, amount)

def deposit(balance, amount):
    balance += amount
    return balance
```

Encapsulation Principle:
Data used by a function should be a parameter or encapsulated in function.
Learning Goals

1. Understanding how images are represented
2. Learning about the SimpleImage library
3. Writing code that can manipulate images
What is an Image?

- Image made of square pixels
  - Example: flower.png
- Each pixel has x and y coordinates in the image
  - The origin (0, 0) is at the upper-left corner
  - y increases going down, x increases going right
- Each pixel has single color encoded as 3 RGB values
  - R = red; G = green; B = blue
  - Each value represents brightness for that color (red, green, or blue)
  - Can set RGB values to make any color!
Pixels in an Image Close-Up

Pixel (1, 0):
red: 6 green: 250 blue: 7
(i.e. shade of green)

Pixel (4, 2):
red: 241 green: 252 blue: 23

Pixel (2, 3):
red: 247 green: 250 blue: 237
Working with Images: Pillow and the SimpleImage library
Installing Pillow

• Pillow is a version of the Python Imaging Library (PIL)
  – Nick Parlante built SimpleImage library using Pillow
  – You'll be using SimpleImage in this class
  – So, you need to install Pillow first

• To install Pillow, open PyCharm Terminal tab and type (note the capital P in Pillow):
  – On a PC:  
    ```bash
    py -m pip install Pillow
    ```
  – On a Mac:  
    ```bash
    python3 -m pip install Pillow
    ```
  – Will see something like:
    ```
    ...bunch of stuff...
    Successfully installed Pillow-7.1.2
    ```

• Handout #8: Image Reference Guide contains more information
Using SimpleImage Library

• In folders for assignment or lecture on images, there is a file `simpleimage.py`
  – This is the SimpleImage library

• To use the SimpleImage library in your code, include at the top of your program file:

  ```python
  from simpleimage import SimpleImage
  ```

• This is importing the SimpleImage module, so that it is accessible in the code you write
  – Similar to when you used `import random` to use random number generator library
Functions in SimpleImage Library

• Create a SimpleImage object by reading an image from file (jpg, png, gif, etc.) and store it in a variable.
  – Note: each SimpleImage object is made up of Pixel objects
  
    ```python
    my_image = SimpleImage(filename)
    ```

• Show the image on your computer.

    ```python
    my_image.show()
    ```

• We can manipulate an image by changing its pixels

• We can also create new images and set its pixels
Accessing Pixels in an Image

• We can use a new kind of loop called a "for-each" loop

• Recall basic for loop (using range):

```python
for i in range(num):
    # i will go from 0 to num - 1
    do_something()
```

• For-each loop:

```python
for item in collection:
    # Do something with item
```

• For-each loop with image:

```python
image = SimpleImage("flower.jpg")
for pixel in image:
    # Do something with pixel
```
image = SimpleImage("flower.jpg")
for pixel in image:
    # Body of loop
    # Do something with pixel

• Like variable i in for loop using range(), pixel is a variable that gets updated with each loop iteration.
• pixel gets assigned to each pixel object in the image in turn.

This code gets repeated once for each pixel in image.
Properties of Images and Pixels

• Each SimpleImage image has properties you can access:
  – Can get the width and height of image (values are in pixels)
    image.width, image.height

• Each pixel in an image also has properties:
  – Can get x, y coordinates of a pixel in an image
    pixel.x, pixel.y
  – Can get RGB values of a pixel
    pixel.red, pixel.green, pixel.blue
    • These are just integers between 0 and 255
    • Higher R, G, or B values means more of that color in pixel
  – Can also set pixel RGB values in an image to change it!
def darker(filename):
    """
    Reads image from file specified by filename.
    Makes image darker by halving red, green, blue values.
    Returns the darker version of image.
    """

    # Demonstrate looping over all the pixels of an image, 
    # changing each pixel to be half its original intensity.
    image = SimpleImage(filename)
    for pixel in image:
        pixel.red = pixel.red // 2
        pixel.green = pixel.green // 2
        pixel.blue = pixel.blue // 2
    return image
def red_channel(filename):
    
    """
    Reads image from file specified by filename. Changes the image as follows:
    For every pixel, set green and blue values to 0 yielding the red channel.
    Return the changed image.
    """
    
    image = SimpleImage(filename)
    for pixel in image:
        pixel.green = 0
        pixel.blue = 0
    return image
Let's take it out for a spin!
imageexamples.py
Greenscreening
What is Greenscreening?

• Like the movies (and Zoom backgrounds)
  – Have original image with areas that are "sufficiently green."
  – Replace "green" pixels with pixels from corresponding x, y locations in another image
What is Greenscreening?

• Like the movies (and Zoom backgrounds)
  – Have original image with areas that are "sufficiently green."
  – Replace "green" pixels with pixels from corresponding x, y locations in another image

INTENSITY_THRESHOLD = 1.6

def greenscreen(main_filename, back_filename):
    image = SimpleImage(main_filename)
    back = SimpleImage(back_filename)
What is Greenscreening?

• Like the movies (and Zoom backgrounds)
  – Have original image with areas that are "sufficiently green."
  – Replace "green" pixels with pixels from corresponding x, y locations in another image

INTENSITY_THRESHOLD = 1.6

def greenscreen(main_filename, back_filename):
    image = SimpleImage(main_filename)
    back = SimpleImage(back_filename)
    for pixel in image:
What is Greenscreening?

- Like the movies (and Zoom backgrounds)
  - Have original image with areas that are "sufficiently green."
  - Replace "green" pixels with pixels from corresponding x, y locations in another image

INTENSITY_THRESHOLD = \(1.6\)

```python
def greenscreen(main_filename, back_filename):
    image = SimpleImage(main_filename)
    back = SimpleImage(back_filename)
    for pixel in image:
        average = (pixel.red + pixel.green + pixel.blue) // 3  # See if this pixel is "sufficiently" green
        if pixel.green >= average * INTENSITY_THRESHOLD:
```
What is Greenscreening?

• Like the movies (and Zoom backgrounds)
  – Have original image with areas that are "sufficiently green."
  – Replace "green" pixels with pixels from corresponding x, y locations in another image

\[
\text{INTENSITY\_THRESHOLD} = 1.6
\]

```python
def greenscreen(main_filename, back_filename):
    image = SimpleImage(main_filename)
    back = SimpleImage(back_filename)
    for pixel in image:
        average = (pixel.red + pixel.green + pixel.blue) // 3
        # See if this pixel is "sufficiently" green
        if pixel.green >= average * INTENSITY\_THRESHOLD:
            # If so, overwrite pixel in original image with corresponding pixel from the back image.
            x = pixel.x
            y = pixel.y
            image.set_pixel(x, y, back.get_pixel(x, y))
    return image
```
Let's try it!
(But using red instead of green)
Mirroring an image
image = SimpleImage(filename)
width = image.width
height = image.height

for y in range(height):
    for x in range(width):
        pixel = image.get_pixel(x, y)
        # do something with pixel
def mirror_image(filename):
    image = SimpleImage(filename)
    width = image.width
    height = image.height

    # Create new image to contain mirror reflection
    mirror = SimpleImage.blank(width * 2, height)

    for y in range(height):
        for x in range(width):
            pixel = image.get_pixel(x, y)
            mirror.set_pixel(x, y, pixel)
            mirror.set_pixel((width * 2) - (x + 1), y, pixel)

    return mirror
I wanna see it!
def darker(filename):
    img = SimpleImage(filename)
    for px in img:
        px.red = px.red // 2
        px.green = px.green // 2
        px.blue = px.blue // 2
    return img

def darker(filename):
    img = SimpleImage(filename)
    for y in range(img.height):
        for x in range(img.width):
            px = img.get_pixel(x, y)
            px.red = px.red // 2
            px.green = px.green // 2
            px.blue = px.blue // 2
    return img

Nothing!
We only want to use nested for loops if we care about \( x \) and \( y \).
(Needed that for mirroring image.)
Learning Goals

1. Understanding how images are represented
2. Learning about the SimpleImage library
3. Writing code that can manipulate images