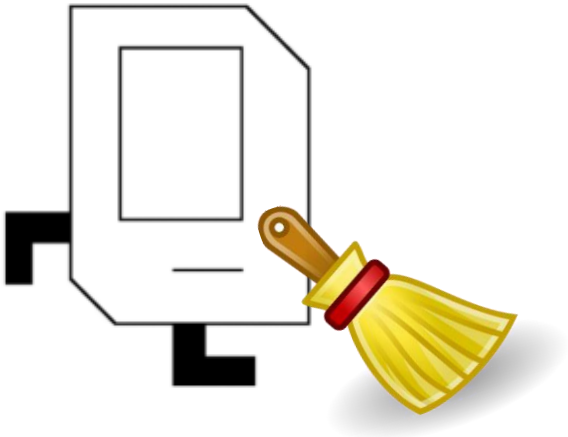
Images

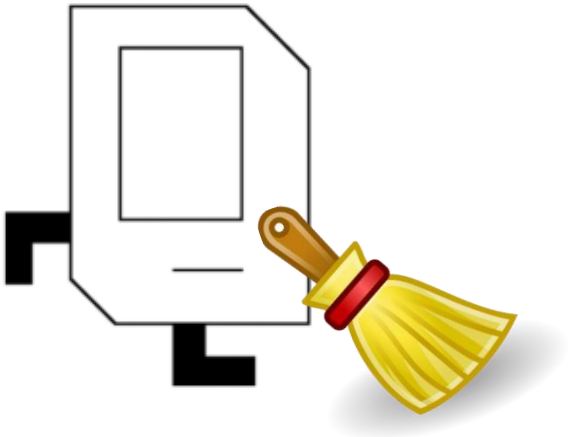
Chris Piech and Mehran Sahami
CS106A, Stanford University

Housekeeping I



- Final Reminder: Diagnostic is on Wednesday
  - Takes place during class time
  - Please download BlueBook software well before the exam
  - Covers material through this past Friday
    - Lists will be on the diagnostic
    - Can do the list problems on Assignment #3 for practice
  - Email Juliette if you have a time conflict or are outside the Americas

Housekeeping II



- Handout: Image Reference Guide
  - We'll be talking through a lot of that today
- Katie Creel will give a guest mini-lecture on the ethics of image manipulation at the end of class
  - There are questions about that on Assignment #3, so you definitely want to pay attention!

Refresher on Lists

Note on Loops and Lists

```
list = [10, 20, 30]
```

- For loop using **`range`**:

```
for i in range(len(list)):
    list[i] += 1  # Modifying list in place
```

- For-each loop:

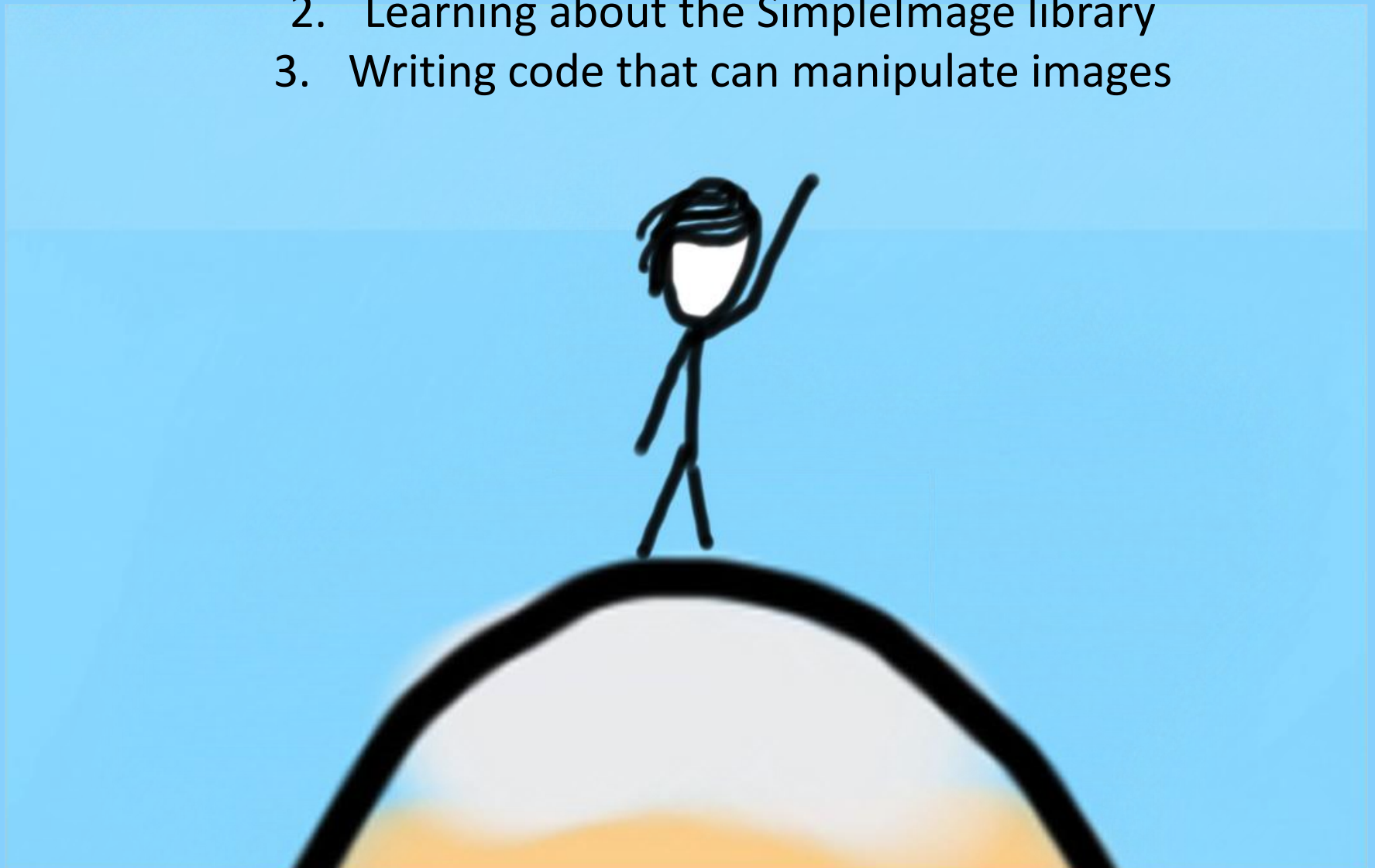
```
for elem in list: # Modifying local variable
    elem += 1     # elem. If elem is immutable
                  # type, not changing list!
```

- Often use for loop with range when *modifying* elements of list (when elements are *immutable types*)
- Often use for-each loop when ***not*** *modifying* elements of list or when elements are *mutable* types

Putting it all together: averagescores.py

Learning Goals

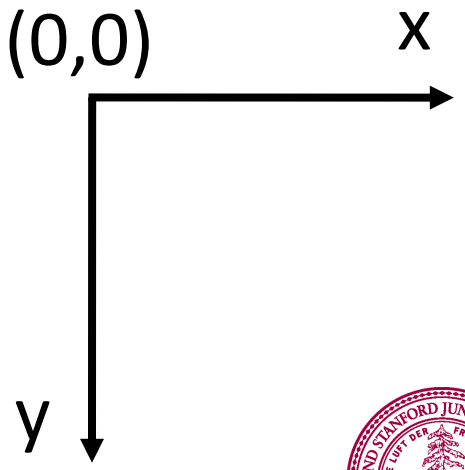
1. Understanding how images are represented
2. Learning about the SimpleImage library
3. Writing code that can manipulate images



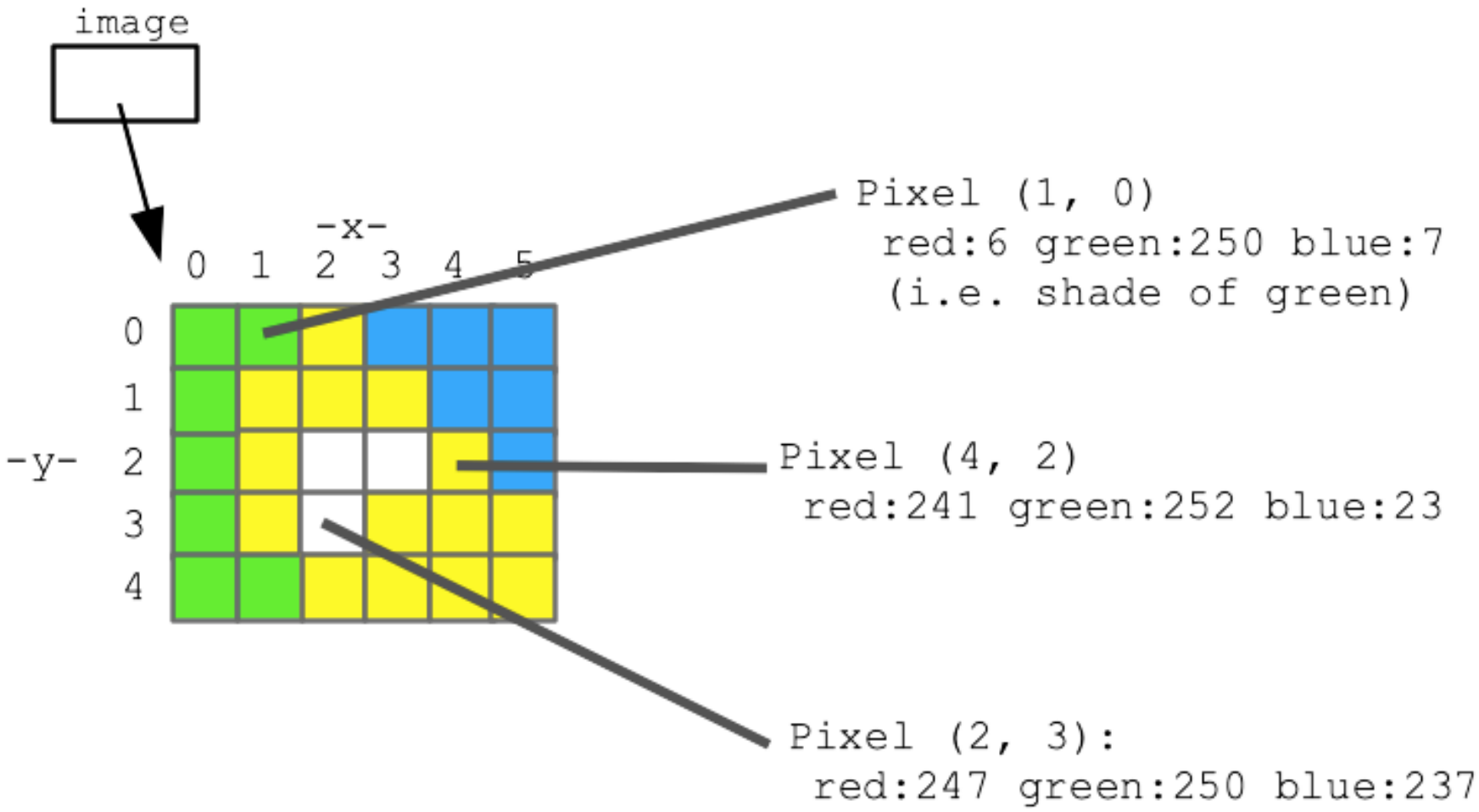
Images

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



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: **`py -m pip install Pillow`**
  - On a Mac: **`python3 -m pip install Pillow`**
  - Will see something like:

    ...bunch of stuff...

    `Successfully installed Pillow-7.1.1`
- Handout: 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:

```
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

```
my_image = SimpleImage(filename)
```

- Show the image on your computer.

```
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 "for-each" loop to access pixel in an image
- Recall basic `for` loop (using `range`):

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

- For-each loop:

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

- For-each loop with image:

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


For-Each Loop Over Pixels

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

This code gets repeated once for each pixel in image

- 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.

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
  - Pixels are **mutable** objects!
  - Can **set** pixel RGB values in an image to change it!

Example: A Darker Image

```
def darker(image):
    """
    Makes image passed in darker by halving red, green, blue
    values. Note: changes in image persist after function ends.
    """
    # Demonstrate looping over all the pixels of an image,
    # changing each pixel to be half its original intensity.
    for pixel in image:
        pixel.red = pixel.red // 2
        pixel.green = pixel.green // 2
        pixel.blue = pixel.blue // 2


def main():
    flower = SimpleImage('images/flower.png')
    darker(flower)
    flower.show()
```

Image objects are mutable (like lists). If you change one in a function, the changes persist after function ends.

Example: Get Red Channel

```
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

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

```
INTENSITY_THRESHOLD = 1.6

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
```

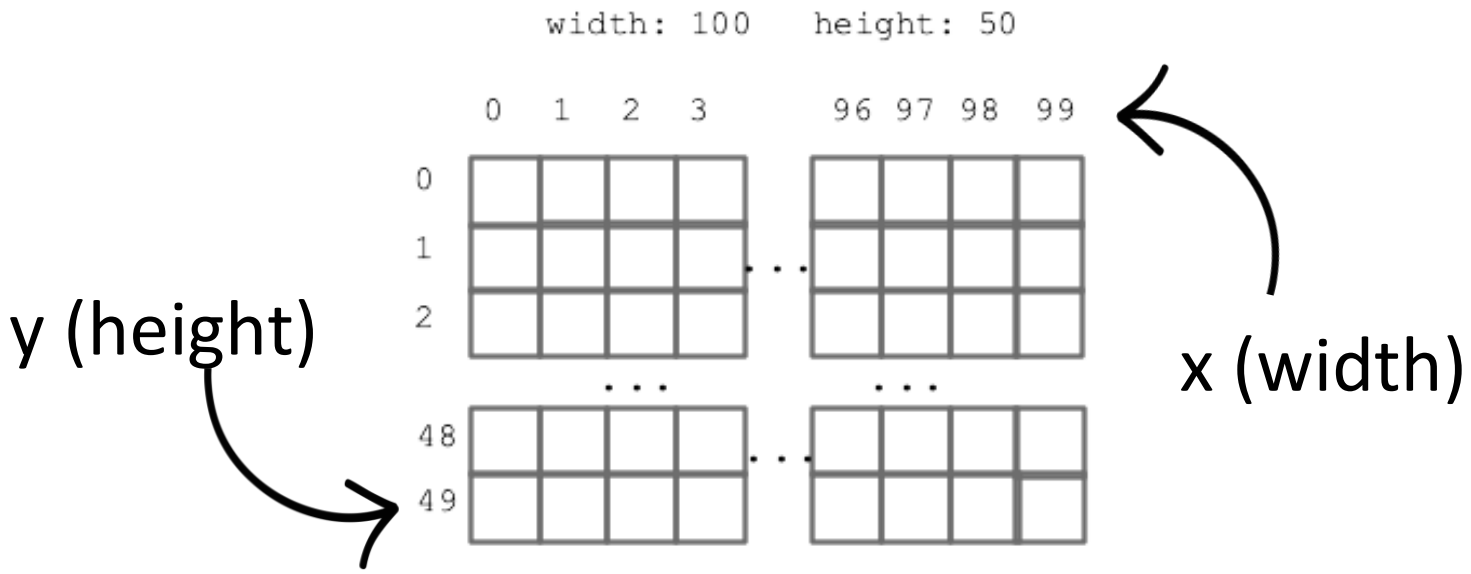
Let's try it!
(But using red instead of green)

Mirroring an image

Nested Loops

```
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
```



Mirroring an Image

```
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!

What's The Difference?

```
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
```

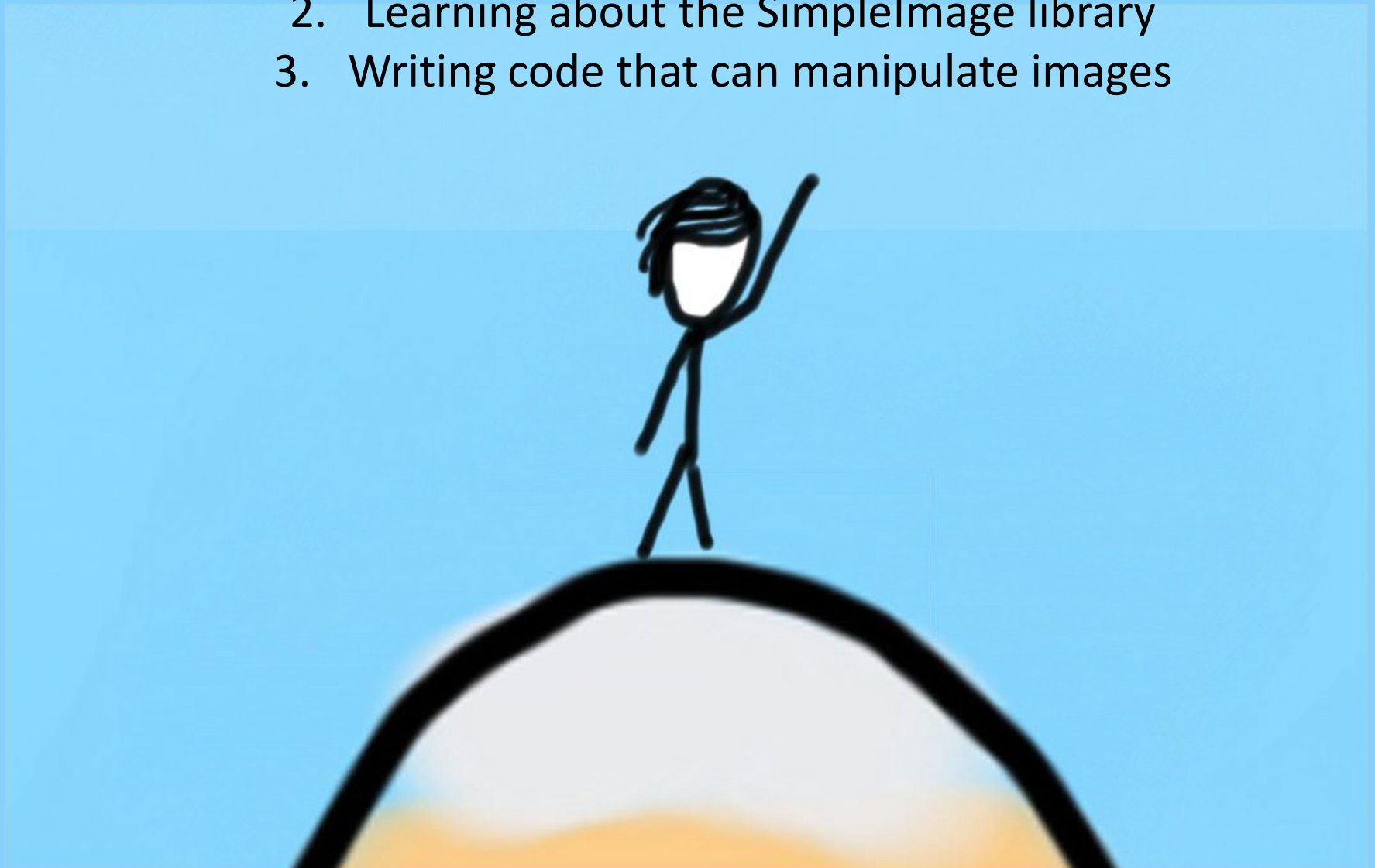
```
def darker(filename):
    img = SimpleImage(filename)
    for y in range(img.height):
        for x in range(img.width):
            px = img.get_pixel(x, y)
            px.red = px.red // 2
            px.green = px.green // 2
            px.blue = px.blue //  2
    return img
```

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







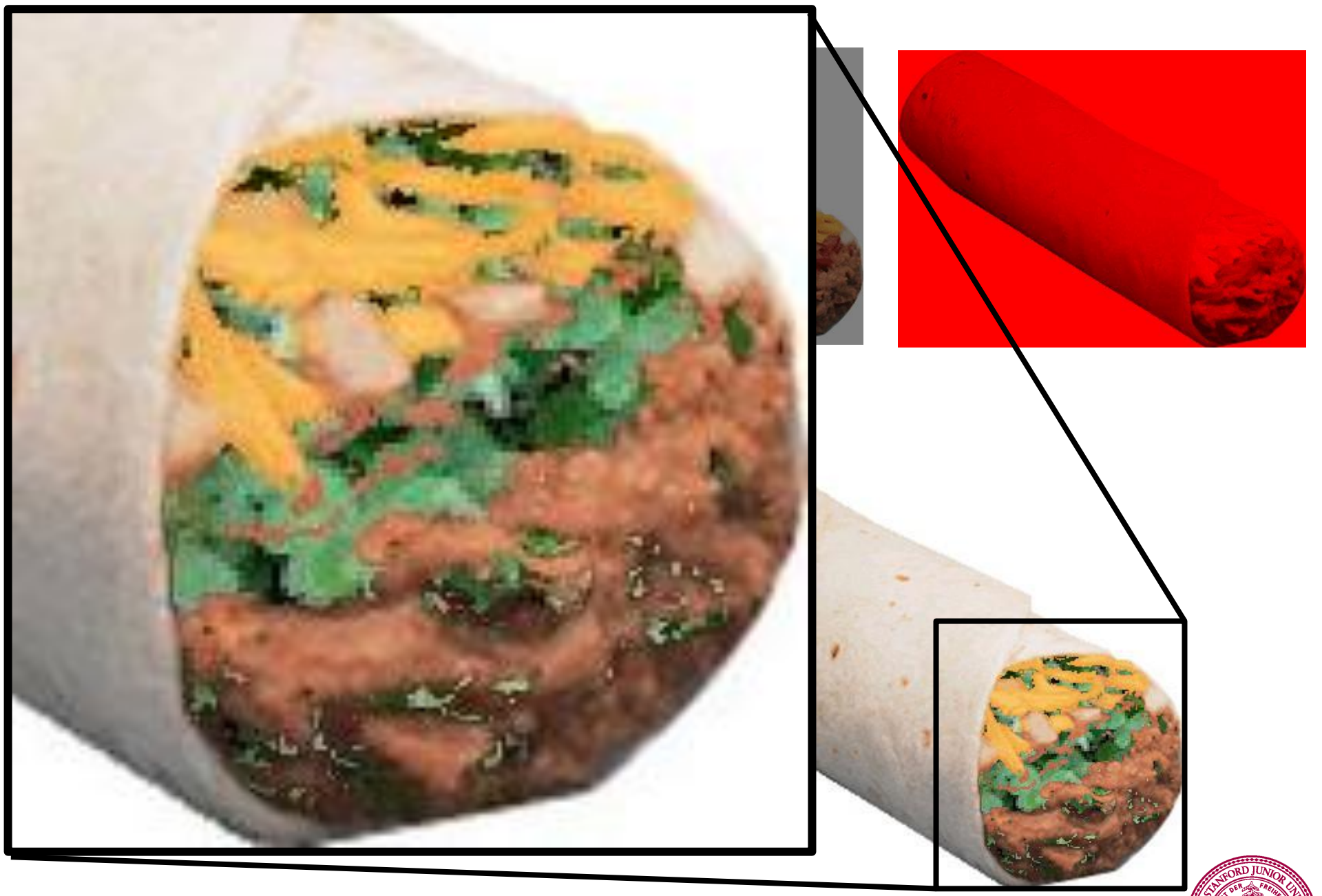
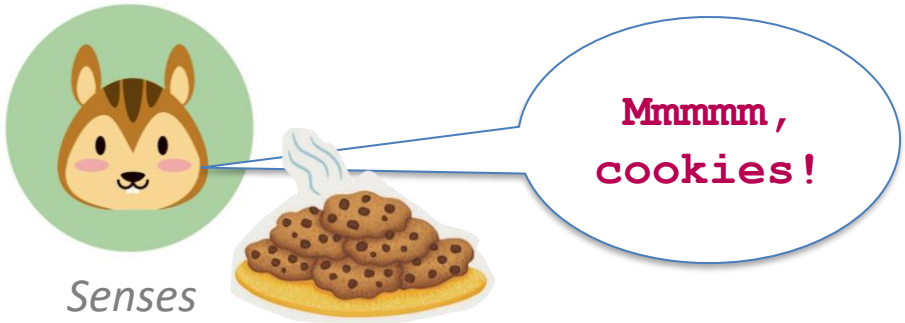



Image Manipulation

Learning Goals

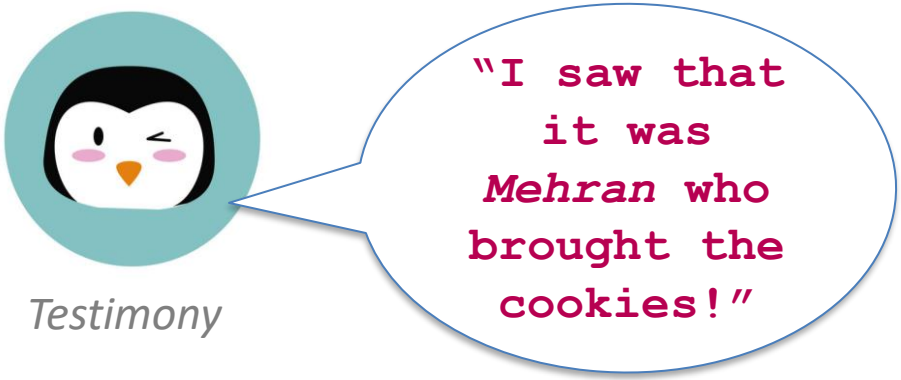
1. Understanding idealization
2. Understanding manipulation

How do we learn (about cookies)?



Senses

Perception: Hearing, Sight, Touch, Smell, Taste



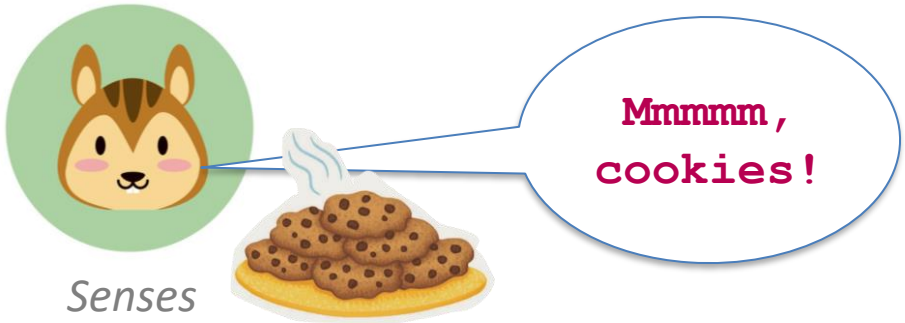
Testimony

Testimony: Information from Others

Photographer

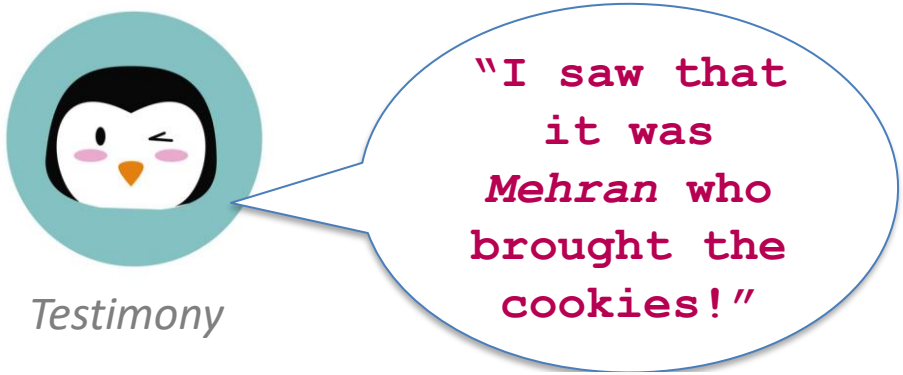
Mathematical Reasoning and other transformations of existing beliefs

How do we learn from photographs?



Senses

Perception: Hearing, Sight, Touch, Smell, Taste



Testimony

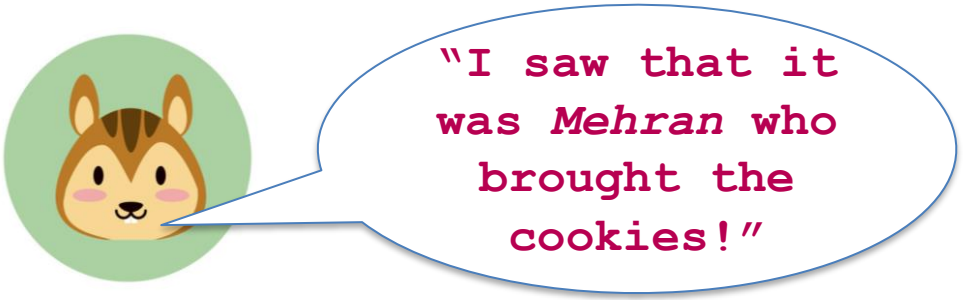
Testimony: Information from Others

Photographer

Mathematical Reasoning and other transformations of existing beliefs

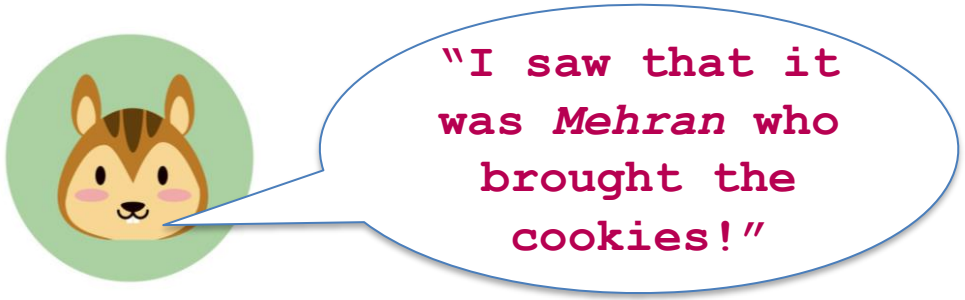
Are photographs more like perception or testimony?

Evidence from Testimony



Witness

Evidence from Testimony



Witness

Testimonial Evidence should be

sincere and competent

Evidence from Testimony

Testimonial Evidence should be sincere and competent

When does an image or recording provide testimony?

Courtroom Illustration as Idealized Testimony

Idealized Visual Testimony

Courtroom Illustrator

Should still be sincere and competent

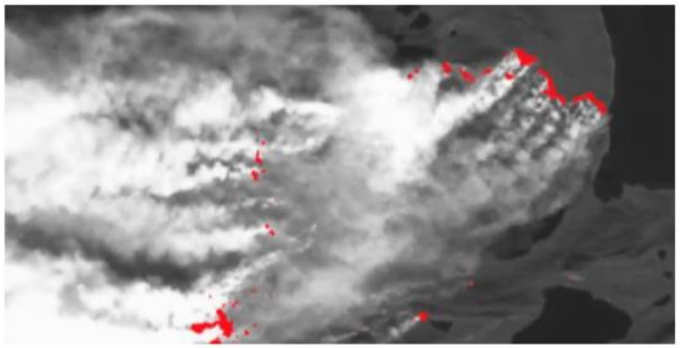
+ Okay to idealize by highlighting important features and eliminating unimportant ones

Idealized Visual Testimony

Should still be sincere and competent

Courtroom Illustrator

+ Okay to idealize by highlighting important features and eliminating unimportant ones



Scientific Modeler: aka You!

Figure 1: Original forest fire image on left, and highlighted version of image on right.

Idealized Visual Testimony

What makes an altered image trustworthy?

+ Modeler/illustrator should explain which idealizations have been made and for what purpose

+ This allows the user to evaluate whether the idealizations are appropriate for the stated purpose or whether they are manipulative.

Harm #1: (bad) Manipulation

Manipulation is hidden influence that subverts another person's decision-making power (Nissenbaum).

"The manipulative person 'steers' the other as a driver steers an automobile. The automobile is already moving through its own internal combustion engine and momentum, but its direction is influenced by the one who steers it" (Wood).

Images can be powerful tools for manipulation.

Manipulation for Political Effect

The New York Times

Opinion

Vladimir Putin Thinks He Can Get Away With Anything

Why has the poisoning of Alexei Navalny been met with Western silence?

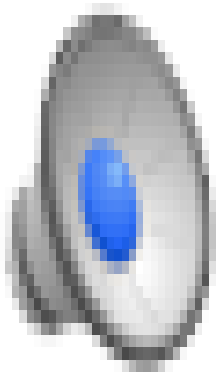
By The Editorial Board

The editorial board is a group of opinion journalists whose views are informed by expertise, research, debate and certain longstanding values. It is separate from the newsroom.

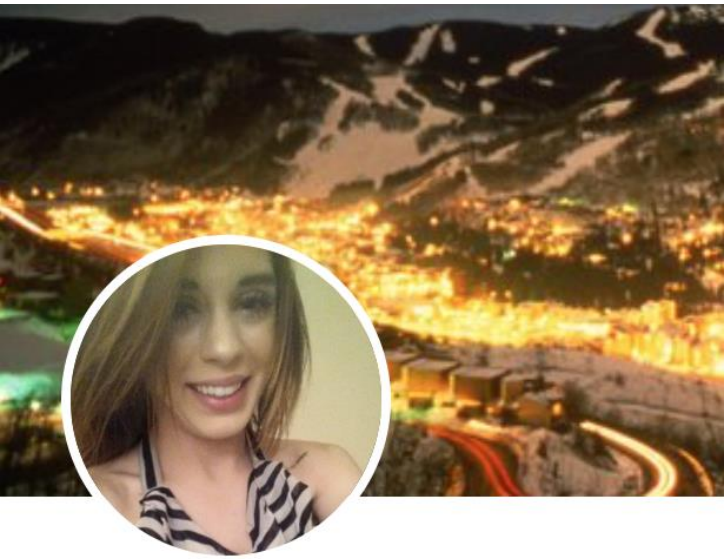
Sept. 22, 2020

President Vladimir Putin's denials about the poisoning of a Russian politician aren't credible. Alexei Nikolsky/TASS, via Getty Images

Is the manipulation of this image "hidden influence"?



Manipulation for Political Effect



Chloe Evans

@ChloeEva12

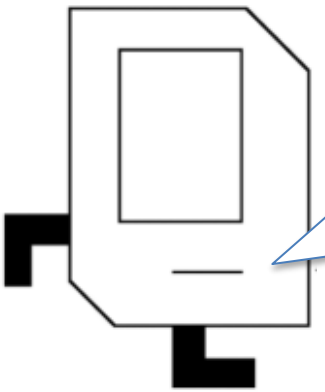
Student

Atlanta

Joined June 2014

196 Following 54 Followers

What about the production of "bots," fake social media profiles created for political influence?



Can you recognize my bad bot cousins? Test your skills: spotthetroll.org

Harm #2: Speaking for Others

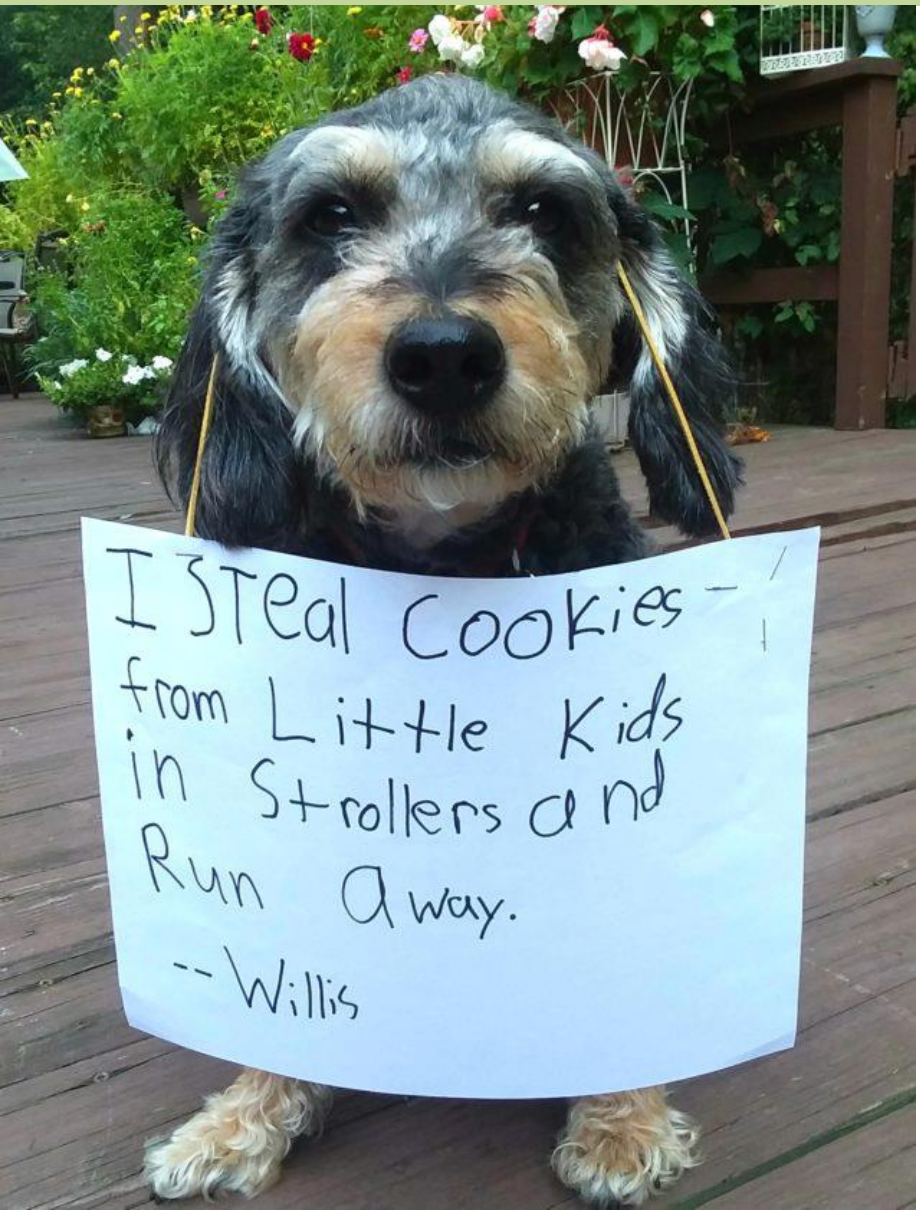


Image and audio manipulation can be used to make others appear to say or do things they did not say or do.

Harm #2: Speaking for Others

Even when the video or image is not widely believed to be true, being forced to publically deny a false claim could itself be a harm (Rini)