File Formats

A **multimedia** file(audio, video, image) is simply a sequence of 0's and 1s.

In the previous lecture, with character encoding, this sequence can be translated into text.

> For example, ASCII, UTF-8. In ASCII, 65 is an A, etc.

Encoding such as ASCII, however, doesn't specify, for example, font color.

> HTML provides one way of specifying font, background color.

But a Word document(file ends in .doc and .docx) doesn't use HTML.

> .doc is called a **filename extension**, indicates file's format.

The purpose of a file format is to define a standardized way of representing information.

File Formats

A file's format simply tells your computer how the bits of a files should be interpreted.

> Those 0s and 1s could represent an image, a sound, or a video, but without any kind of rules for interpreting a file's bits, they're essentially meaningless.

A file's extension usually indicates the format of the file, but isn't a definitive answer.

> Microsoft word can open both important.doc to important.whatever

> a PPM file for example is a text file if open with Notepad or TextEdit but is an image if open with GIMP.

Popular image file formats: BMP, JPEG, PNG, GIF.

> All of these acronyms are just different ways of representing images using binary data.

Bitmap

An image is 2D grid of squares(**bitmap**), where each square is filled with only one color; this is kind of structure is called a **raster graphics image.**

Each of these small squares, called a **pixel**, can be filled with exactly one color.

Computers use **additive color mixing** to produce colors.

> primary colors are red, green and blue(RGB)

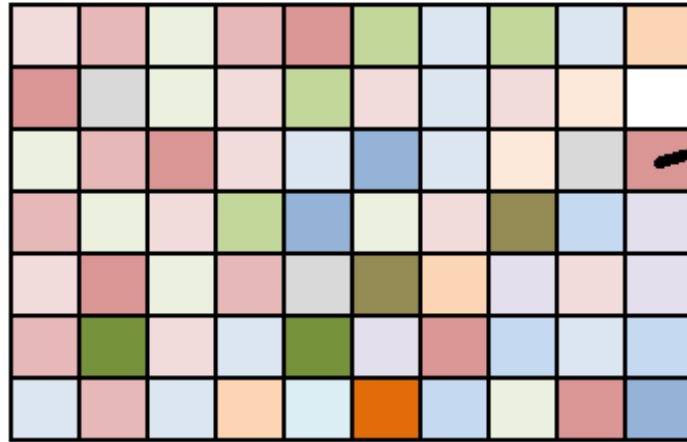Typically, bitmaps use 24-bit(3 bytes) for colors.
- 8 bits for each of the three primary colors.
- each color has values 0 to 255.
- 24-bit equals approximately 16.8 million colors.

R 93%	R 35%	R 90%
G 93%	G 35%	G 90%
B 93%	B 16%	B 0%
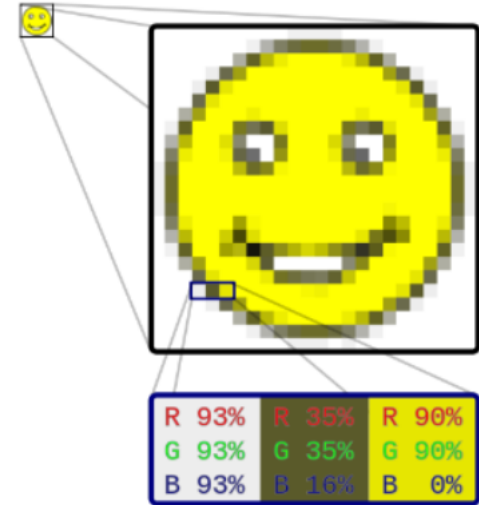
RGB Model

Encoding Data in Binary

RGB(218,150,149)

R = 11011010
G = 10010110
B = 10010101

Bitmap

Colors often are represented in hexadecimal.

Each color is stored using 8 bits, need six hexadecimal digits to express a 24-bit color. (a hexadecimal is 4 bits.)

As a convention, hexadecimal colors are prefixed with either the hash character (#) or sometimes (0x)

  the color #FF0000 (red) is a lot of red, no green, and no blue, while the color #0088FF (sky blue) is no red, a bit of green, and a lot of blue

.bmp is a bitmap file format.

  The first two bytes of any bitmap file are the same: the magic number 0x42 0x4D. 0x42 is the same as the decimal number 66, while 0x4D is the same as the decimal number 77.

  then some **metadata**(data that describes a file's data) such as width, height, size.

  then the bytes that describe the pixels of the image.

Resolution

Raster graphics image has to be made up of a grid of squares, but how big should each square be?

the **resolution** of an image, or the amount of information stored in the image

- same image size but smaller pixel size equals higher resolution(more detail) and therefore larger number of pixels in the image.

- for example, the left image below is 16 pixels by 16 pixels resolution but the right image is a higher resolution 512x512.

Lossless Compression

What would the bitmap data look like for the image below?
> a lot of the bytes are exactly the same!

Is there a way to encode the image with an abbreviation for "500 black pixels" rather than listing each pixel individually?

CompuServe develops GIF, or the Graphics Interchange Format, in 1987.
> can be pronounced "GIF" or "JIF"

Unlike bitmaps, GIFs are compressed,
> can represent exactly the same information
> as some bitmaps using a smaller number of bits.
> essentially, if two pixels that are horizontally
> adjacent are exactly the same, then GIF
> compresses the data.

This method is called **lossless compression**, i.e, no data is lost.

Besides images, RAR, ZIP, GZIP, LZW compress any file.

Lossy Compression

While lossless compression preserves all of a file's data, **lossy compression** throws away some data in the interest of compressing the file even more.

Effective lossy compression should preserve the essential data. You use effective lossy compression every time you text someone!

Hey wat r u doin tmrw? I wntd 2 go 2 c Ben.(43 characters)

Hey, what are you doing tomorrow? I wanted to go to see Ben.(57 characters)

(compressed by about 25%.)

(you can "probs" do even better.)

GIF uses lossless compression

JPEG(Joint Photographic Experts Group), for example, uses lossy compression.

PNG(Portable Network Graphic) is another popular format.

the original creators of PNG actually picked the acronym for "PNG is Not GIF" as a reaction to some of the dubious licensing issues with the GIF format.

Comparison

Like GIF, PNG uses lossless compression, and like JPEG, uses 24 bits for color.

Another feature of the PNG format is support for alpha, or transparency. BMP and JPEG, on the other hand, don't have a channel devoted to **transparency**.

GIF does have the distinct advantage of supporting animation achieved by repeatedly showing a series of frames.

Comparison

Summary of these formats. The alpha channel is one reason .png files are popular in game images(Sprites).

Name	Extension	Compression	Color	Alpha
Bitmap	.bmp	No	24-bit	No
GIF	.gif	Lossless	8-bit	Yes
JPEG	.jpg, .jpeg	Lossy	24-bit	No
PNG	.png	Lossless	24-bit	Yes

Vector Graphics

If a raster graphics image is scaled larger, the image will appear pixelated.

Vector graphics can scale without losing quality.

Unlike raster graphics, vector graphics doesn't involve turning an image into a grid and storing the values of individual pixels.

Instead, vector graphics stores images using mathematics.

For example, let's say we want to represent a circle. In raster graphics, we'd create a grid of squares and then trace out a circle by filling in squares with some color.

Using vector graphics, we'd instead say that the equation for a square looks something like $x^2 + y^2 = r^2$.

We can create a raster graphic simply by picking a size for the image, then using the equation to figure out which pixels should be colored.

This representation doesn't depend on any pixels. **If we want to create a larger circle, we can just pick a larger value for r, and our equation will create a larger circle without any loss in quality.**

Scalable Vector Graphics

Scalable Vector Graphics(SVG) is an example of a vector graphic.

Notice the difference below between a JPEG and a SVG after some rescaling.

In the powerpoint version of this slides, you can play aroundwith the SVG on the right.

$$\sqrt{b^2 - 4ac}$$

$$\sqrt{b^2 - 4ac}$$

Numpy

Python is currently one of most popular languages. It has many optimized libraries for working with data.

Numpy is a Python library that provides a high-performing multidimensional array object(matrices) and mathematical operations to work with these arrays.

```python
import numpy as np

a = np.array([1,2,3,4])

b = np.array([[1,2,3,4], [5,6,7,8]])

print(a.shape)   # (4,)

print(b.shape)   # (2, 4)

print(b.dtype)  # int64
```

Numpy arrays can only store data of a single type and are super fast.

Python lists can hold objects of different types and are very slow.

Numpy

```python
import numpy as np
a = np.array([[1,2,3,4],
              [5,6,7,8],
              [9,10,11,12]])
print(a[0, 0])    # 1
print(a[1, 3]     # 8
```

Note the use of commas, if a was a 2D Python list: print(a[0][0])

Numpy uses commas(tuples) for indexing.

```python
print(a[:, 1:3])    # all rows, columns 1 and 2.
[[ 2, 3],
 [ 6, 7],
 [10, 11]]
```

Similar to Python lists, slicing works with Numpy arrays!

Matplotlib

Matplotlib is built on top of Numpy and provides plotting capabilities. It is recommended that you use the Jupyter Notebook when working with these libraries.

```python
import matplotlib.pyplot as plt
img = plt.imread("flower.jpg")
print(img.shape)
(859, 840, 3)
```

rows
(height)

columns
(width)

RGB channel

Matplotlib

In Matplotlib, the Figure object is used to contain one or more sets of Axes objects. Data is plotted within a given set of axes.
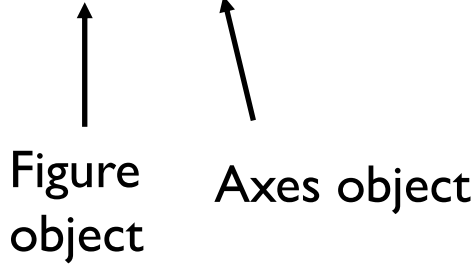
The **subplots()** function can be used to create a figure along with a specified layout of axes.

```python
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
```

Figure
object

Axes object

By default, subplots will create a figure with a single set of axes; calling it will return the figure object and its axes object in a tuple.

imshow()

```python
import matplotlib.pyplot as plt
img = plt.imread("flower.jpg")
print(img.shape)    # (859, 840, 3)

fig, ax = plt.subplots()
ax.imshow(img)
```

imshow() displays the 2D grid of pixels as an image.

Out[102]: <matplotlib.image.AxesImage at 0x123f8b518>
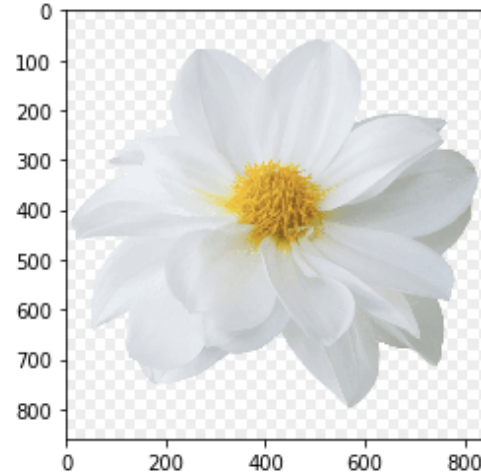
Pixels

```python
import matplotlib.pyplot as plt
img = plt.imread("flower.jpg")
print(img.shape)    # (859, 840, 3)

fig, ax = plt.subplots()
ax.imshow(img)

# (0,0) pixel, slicing all colors
print(img[0,0,:])
array([255, 255, 255], dtype=uint8)
```

Out[102]: <matplotlib.image.AxesImage at 0x123f8b518>
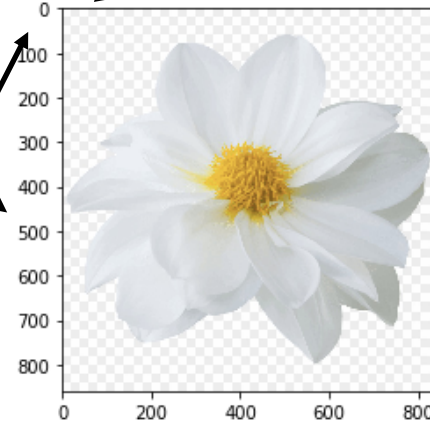
The $(0, 0)$ pixel is at the top left corner of the image.

Decomposing an Image

What would you guess the R, G and B components of the yellow pixel at the center of the flower?

Since red + green is yellow. We should expect high components in the R and G channels and a low number for the B channel.

```
print(img[400,400,:])
array([185, 135, 0], dtype=uint8)
```

Out[102]: <matplotlib.image.AxesImage at 0x123f8b518>
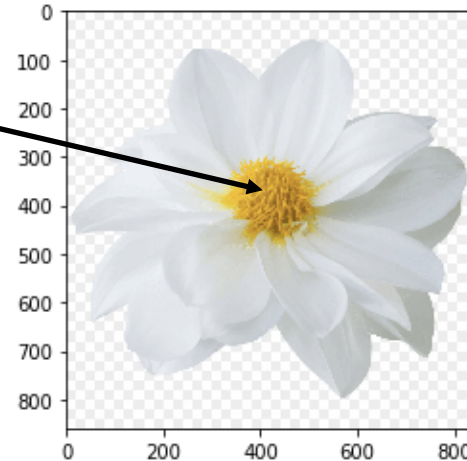
Decomposing an Image

How about a pixel on one of the white petals?

Since red + green + blue is white. We should expect high components in all R, G and B channels.

```
print(img[200,400,:])
array([218, 222, 225], dtype=uint8)
```
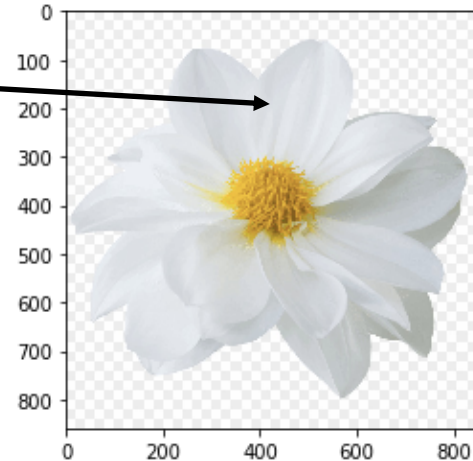
Decomposing an Image

Let write code to decompose our flower image into its individual components.

```python
import matplotlib.pyplot as plt
img = plt.imread("flower.jpg")

# same shape as img, all zeroes
img_red = np.zeroes(img.shape, dtype="uint8")
# extract all rows, all columns, but only the Red channel
img_red[:, :, 0] = img[:, :, 0]
```

Similarly for the G and B channels.

Decomposing an Image

Let's generalize this and put our code into a loop to extract all three components, create three set of axes on Matplotlib and plot all components!

Creates 3 Axes objects using 1 row and 3 columns layout.

```python
import matplotlib.pyplot as plt
img = plt.imread("flower.jpg")


fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(15,5))
for c in range(3):
    temp_img = np.zeros(img.shape, dtype="uint8")
    temp_img[:,:,c] = img[:,:,c]
    axs[c].imshow(temp_img)
```

(width, height) of Figure object
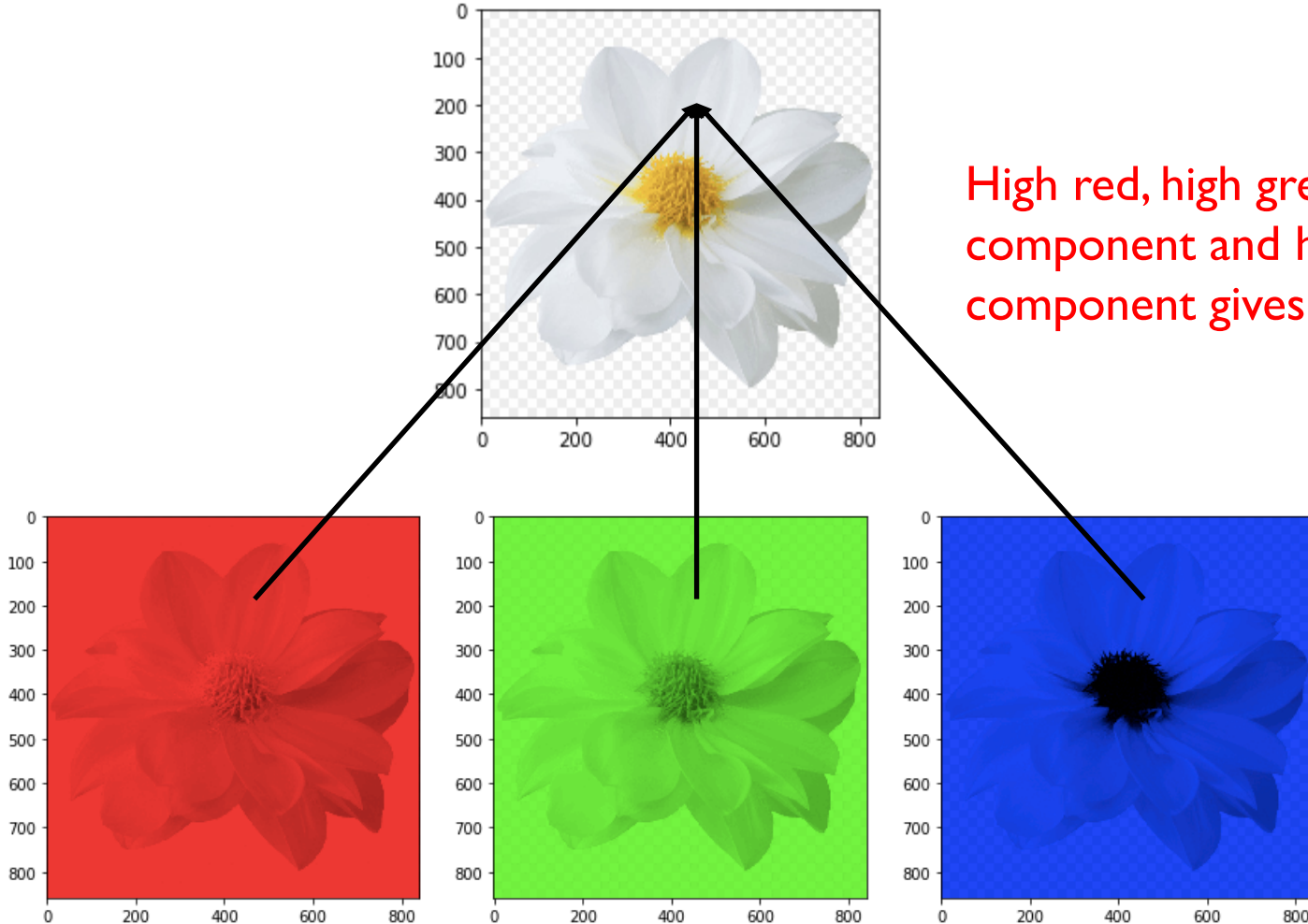
See the next slide for the output.

Decomposing an Image

This output is generated from the previous slide of code in Jupyter Notebook. Pretty good for just a few lines of Python code!

Sum of Its Parts

High red, high green component and no blue component gives yellow!
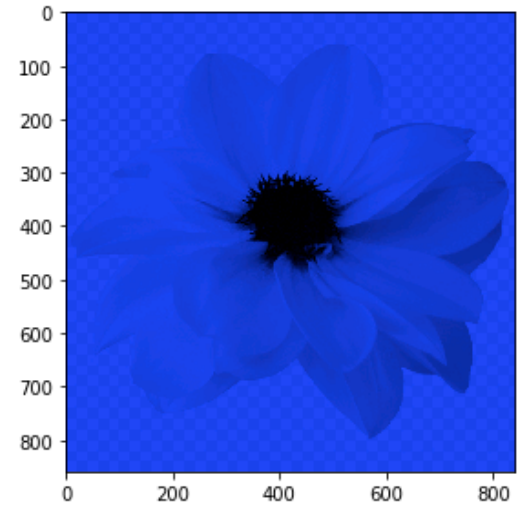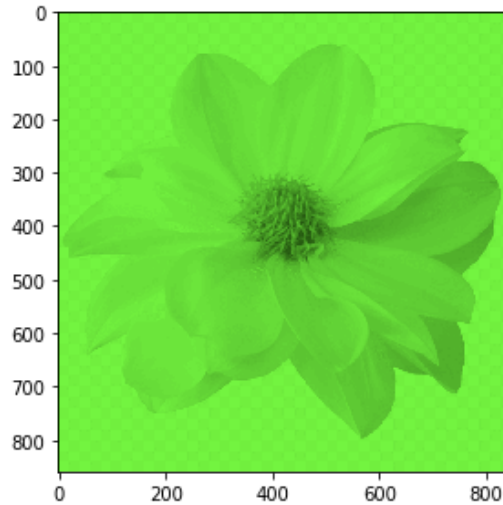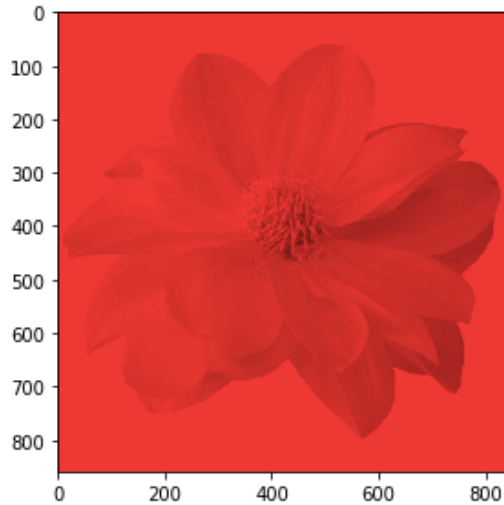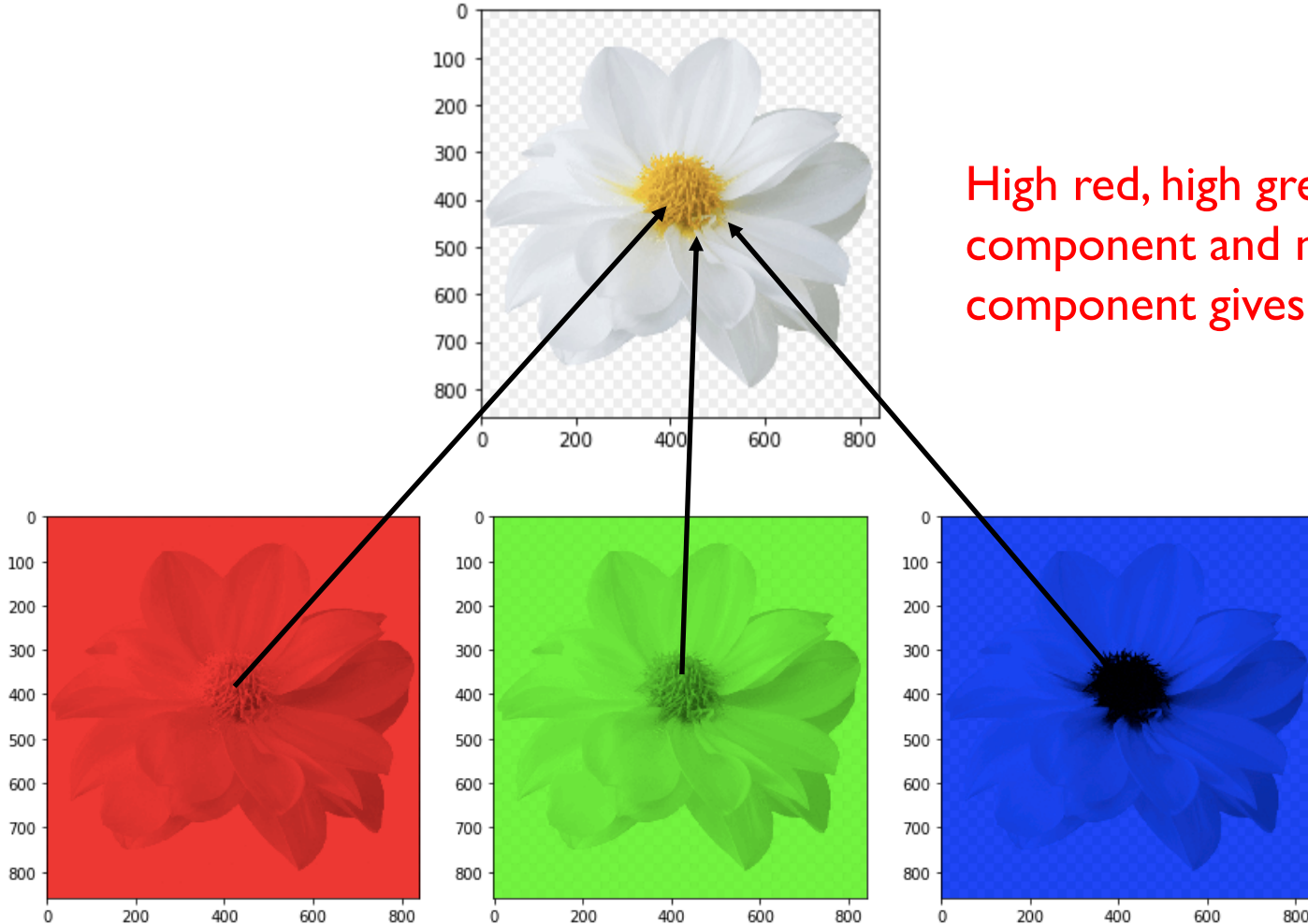
Sum of Its Parts

High red, high green component and high blue component gives white!

RGB to Grayscale

Many image software allows you to convert your image to a black and white version. This amounts to convert the tuple (R, G, B) to one grayscale number.

Two common algorithms found in some software uses the **average method** or the **luminosity method**. (See, for example, the free, open source, image manipulation program called GIMP).

The average method simply averages the R, G, B components:

$$(R + G + B) / 3$$

The luminosity method computes a weighted average taking into account human perception of lightness(for example, human are more sensitive to green, so green has a larger weight):
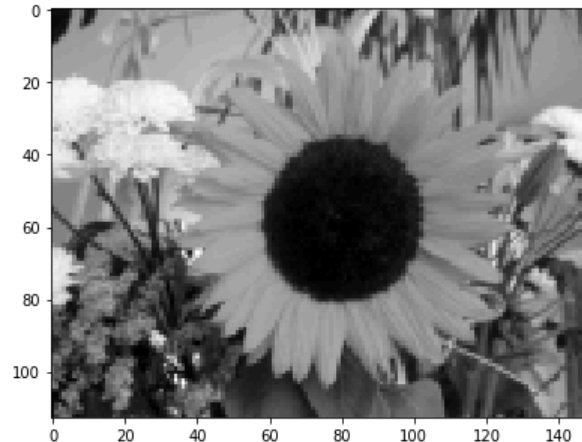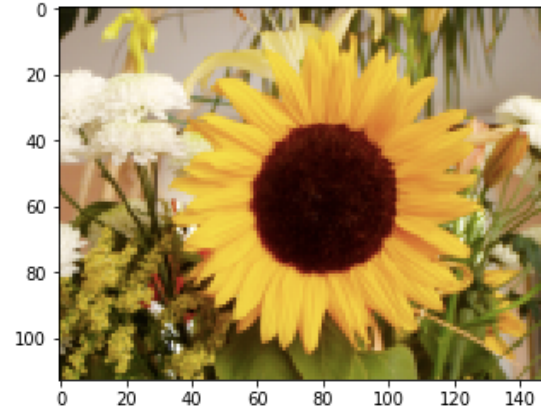
$$0.21 * R + 0.72 * G + 0.07 * B$$

RGB to Grayscale

In your homework, you'll write Python code to do the following conversion to grayscale.

The left image uses the average method, the right uses the luminosity method.

The luminosity is the default method in GIMP.
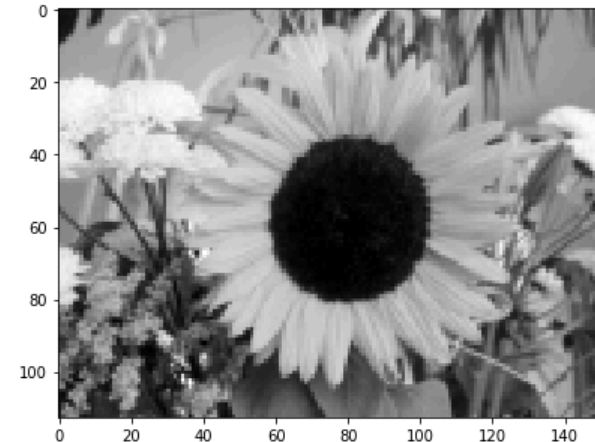
Tint

To tint an image is to mix its colors with white. This will increase the lightness of the image.

In your lab, you will write a Python function, which takes an image and a percentage value as a parameter. Setting 'percentage' to 0 will not change the image, setting it to one means that the image will be completely whitened.

For example, suppose a pixel with RGB components of [0.80, 0.60, 0.40]. Tinting it by 25% means that the pixel is now [0.85, 0.70, 0.55].
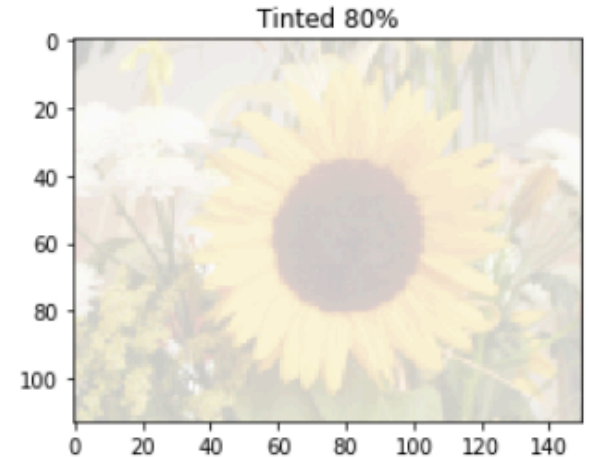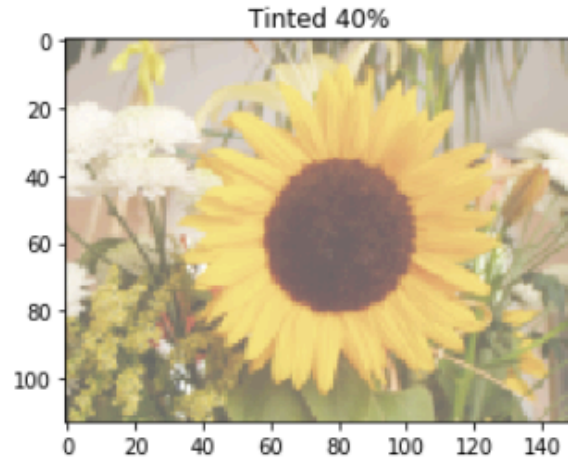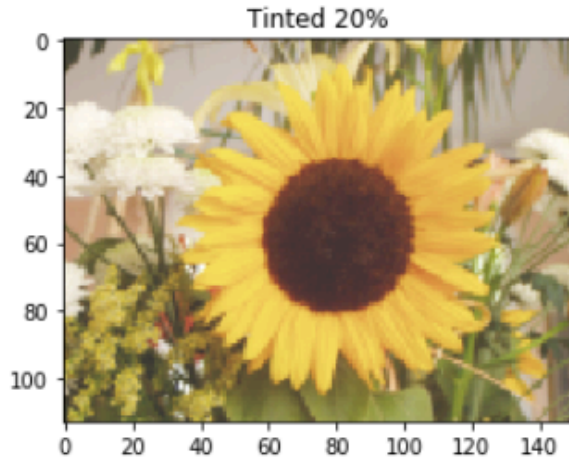
References

1) Part of this lecture is taken from a lecture from an OpenCourseWare course below.

Computer Science E-1 at Harvard Extension School

Understanding Computers and the Internet

by Tommy MacWilliam.

2) The formulas for converting RGB to grayscale can be found in the GIMP documentation: https://docs.gimp.org/2.6/en/gimp-tool-desaturate.html