Computer Vision, Part 1

CST 205

What is computer vision?

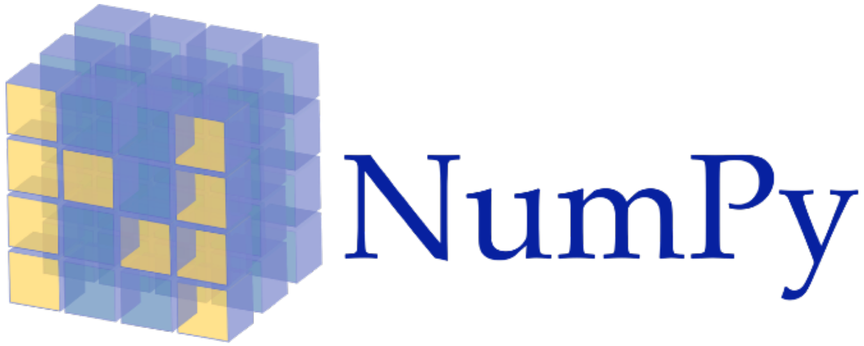
- **Computer vision** is the automated extraction of information from images.
- “Information” can refer to:
    - 3d models
    - camera position
    - object detection and recognition

Computer vision tools

- **OpenCV**
  - Open source C++ Computer Vision Library
  - Originally developed by Intel
  - Emphasis on real-time applications that use multi-core and GPU processing
  - Free for both academic and commercial use.
  - Includes more than 2,000 algorithms for processing image data.

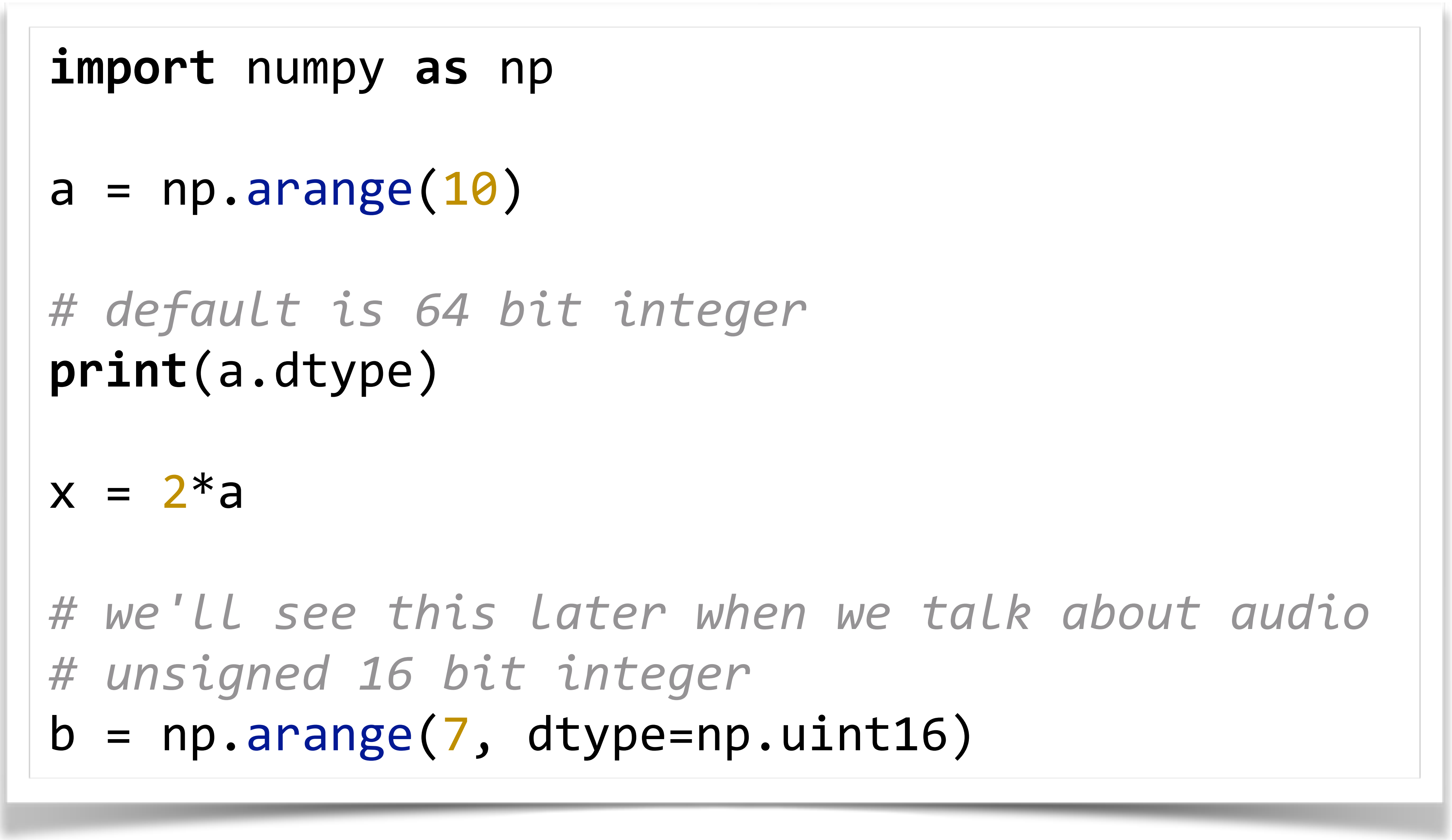
Computer vision tools (cont.)

- **NumPy**
  - Useful for vector and matrix representations and operations



  - Most of NumPy written in C … very fast!
  - `ndarray`: N-dimensional array
    - stored more efficiently and more performant
    - operations can work directly on ndarray (without use of loops)
  - All `OpenCV` array structures are converted to and from NumPy arrays

NumPy Example



```
import numpy as np

a = np.arange(10)

# default is 64 bit integer
print(a.dtype)

x = 2*a

# we'll see this later when we talk about audio
# unsigned 16 bit integer
b = np.arange(7, dtype=np.uint16)
```

Installation

- If you don’t already have NumPy:
  - `pip install numpy`

- `OpenCV` has Python bindings. The name of the library is OpenCV Python. To install:
  - `pip install opencv-python`

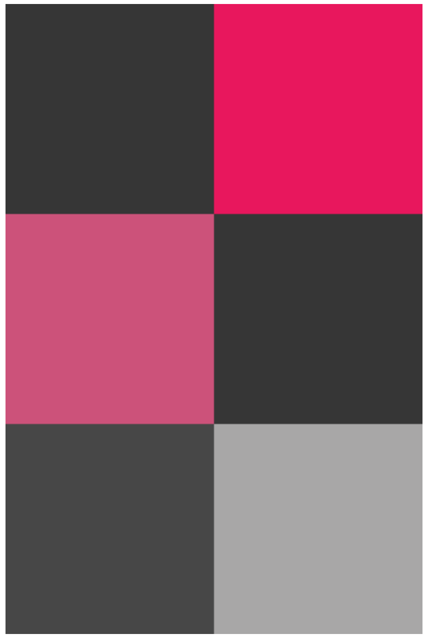
Displaying images

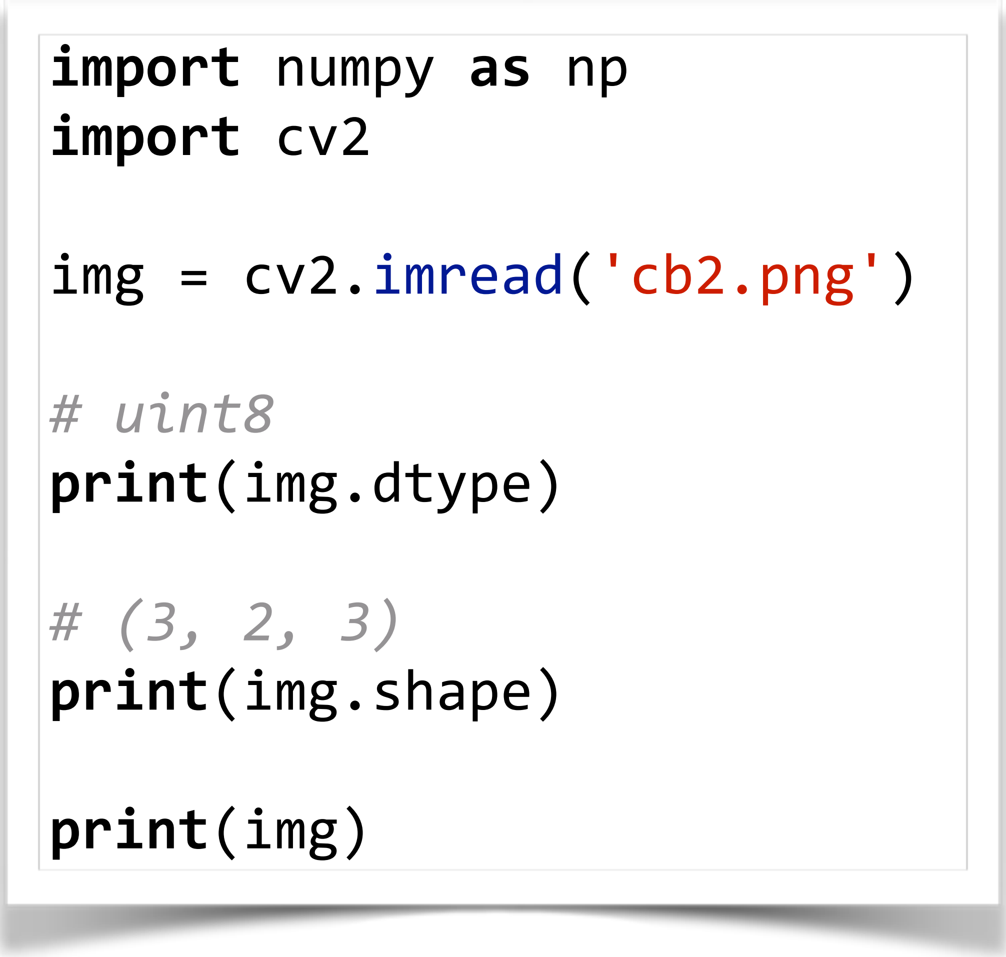
- `OpenCV` provides the `highgui` module
  - High Level GUI and Media I/O
- As with Pillow, we can get image attributes

The `imread()` method

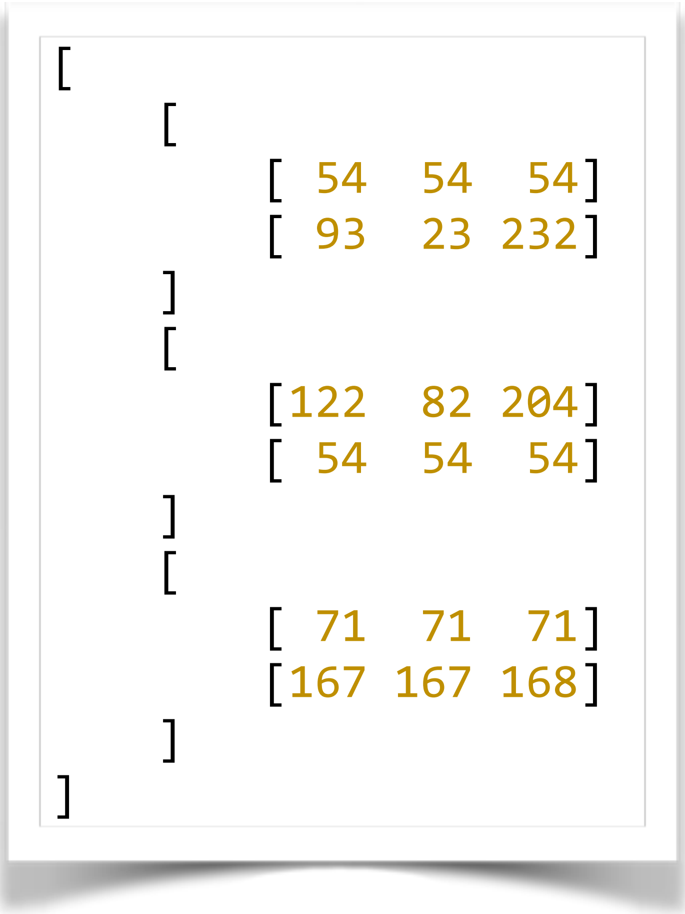
- `imread()` stores the image as a NumPy `ndarray`
- `imread()` supports a variety of image formats
- `imread()` stores colors in BGR format

imread() example





```
import numpy as np
import cv2

img = cv2.imread('cb2.png')

# uint8
print(img.dtype)

# (3, 2, 3)
print(img.shape)

print(img)
```



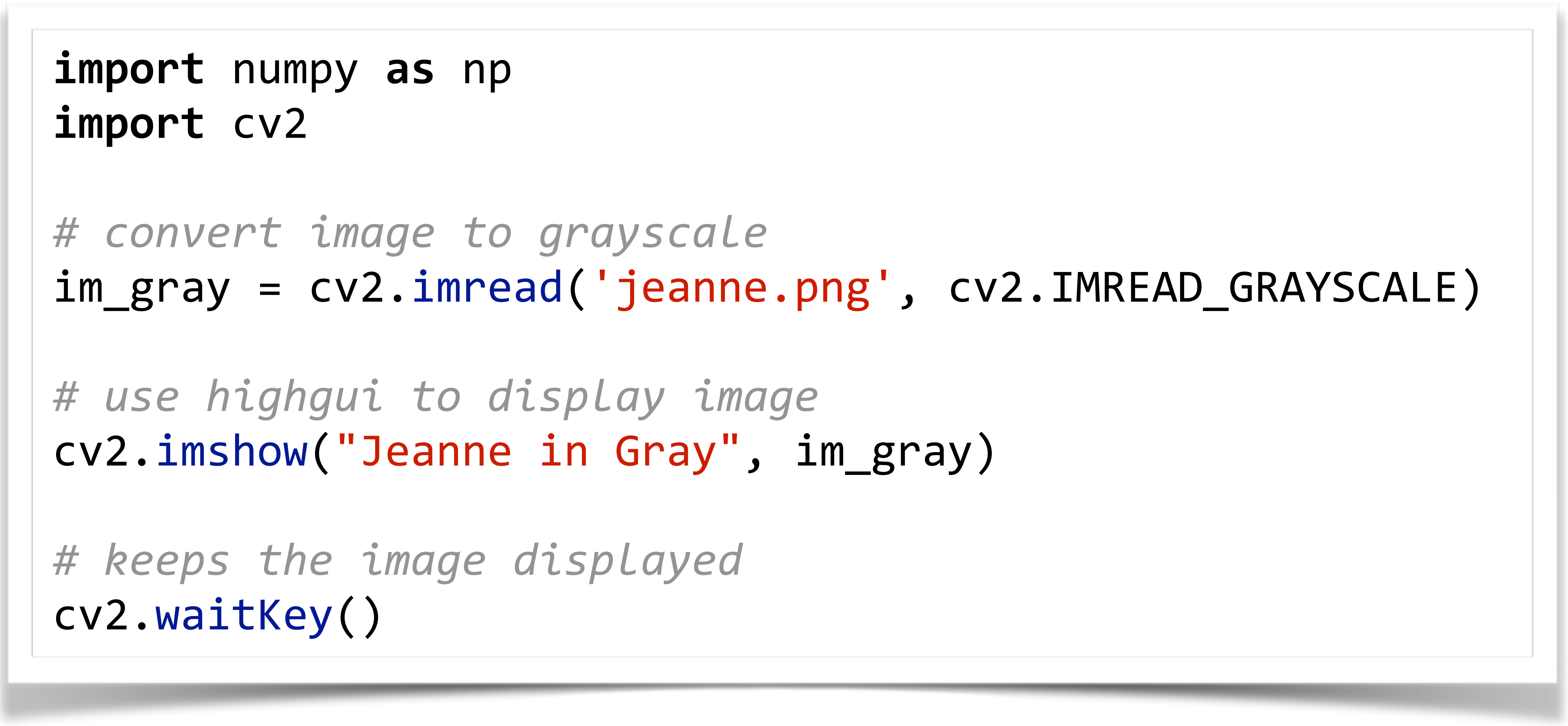
```
[
    [
        [ 54  54  54]
        [ 93  23 232]
    ]
    [
        [122  82 204]
        [ 54  54  54]
    ]
    [
        [ 71  71  71]
        [167 167 168]
    ]
]
```

Convert to grayscale

- Later, when we use classifiers to identify objects in images, we will always first convert the image to grayscale.

    - `imread()` can take a second argument, for grayscale it is `cv2.IMREAD_GRAYSCALE`

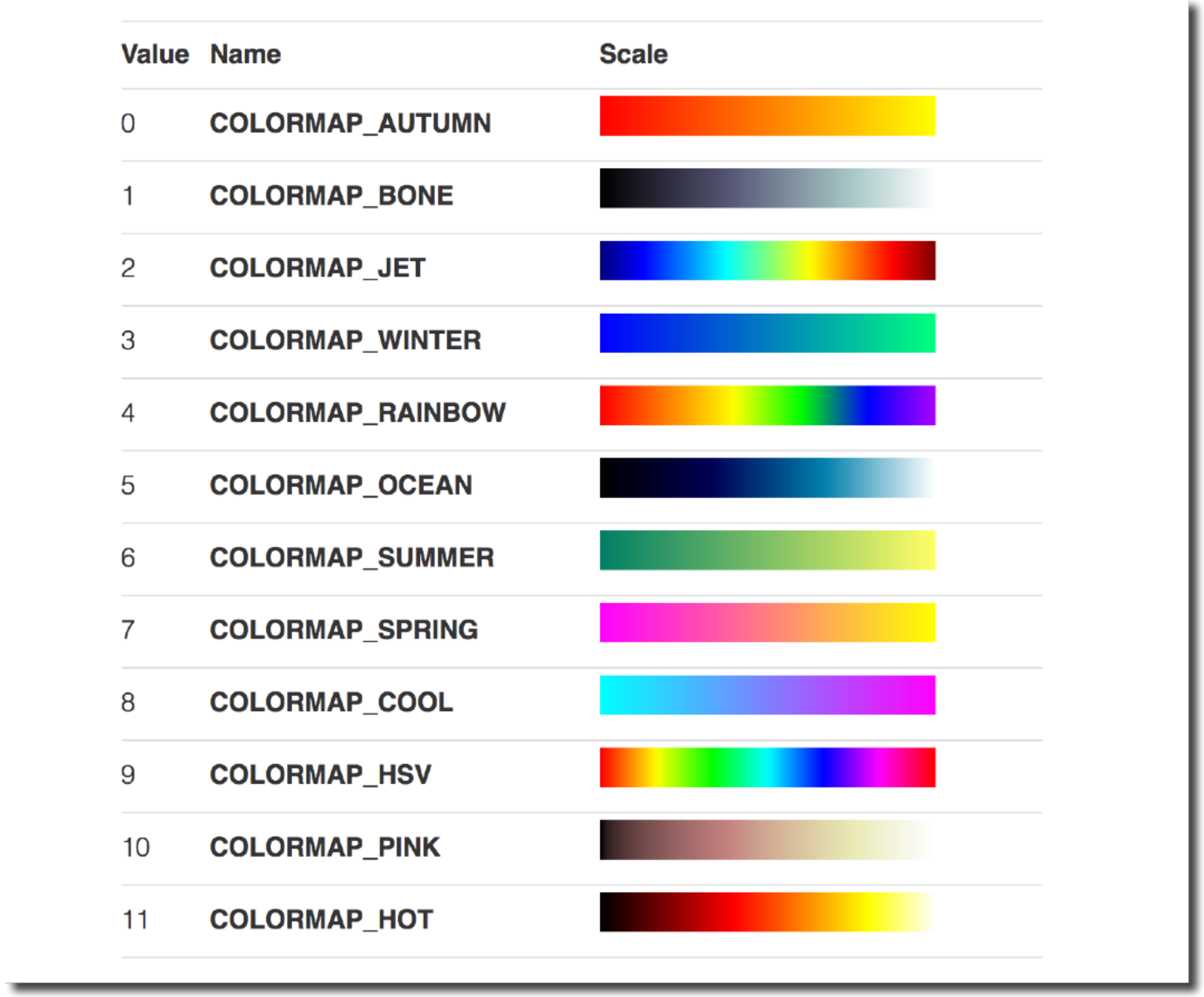
Example



```python
import numpy as np
import cv2

# convert image to grayscale
im_gray = cv2.imread('jeanne.png', cv2.IMREAD_GRAYSCALE)

# use highgui to display image
cv2.imshow("Jeanne in Gray", im_gray)

# keeps the image displayed
cv2.waitKey()
```

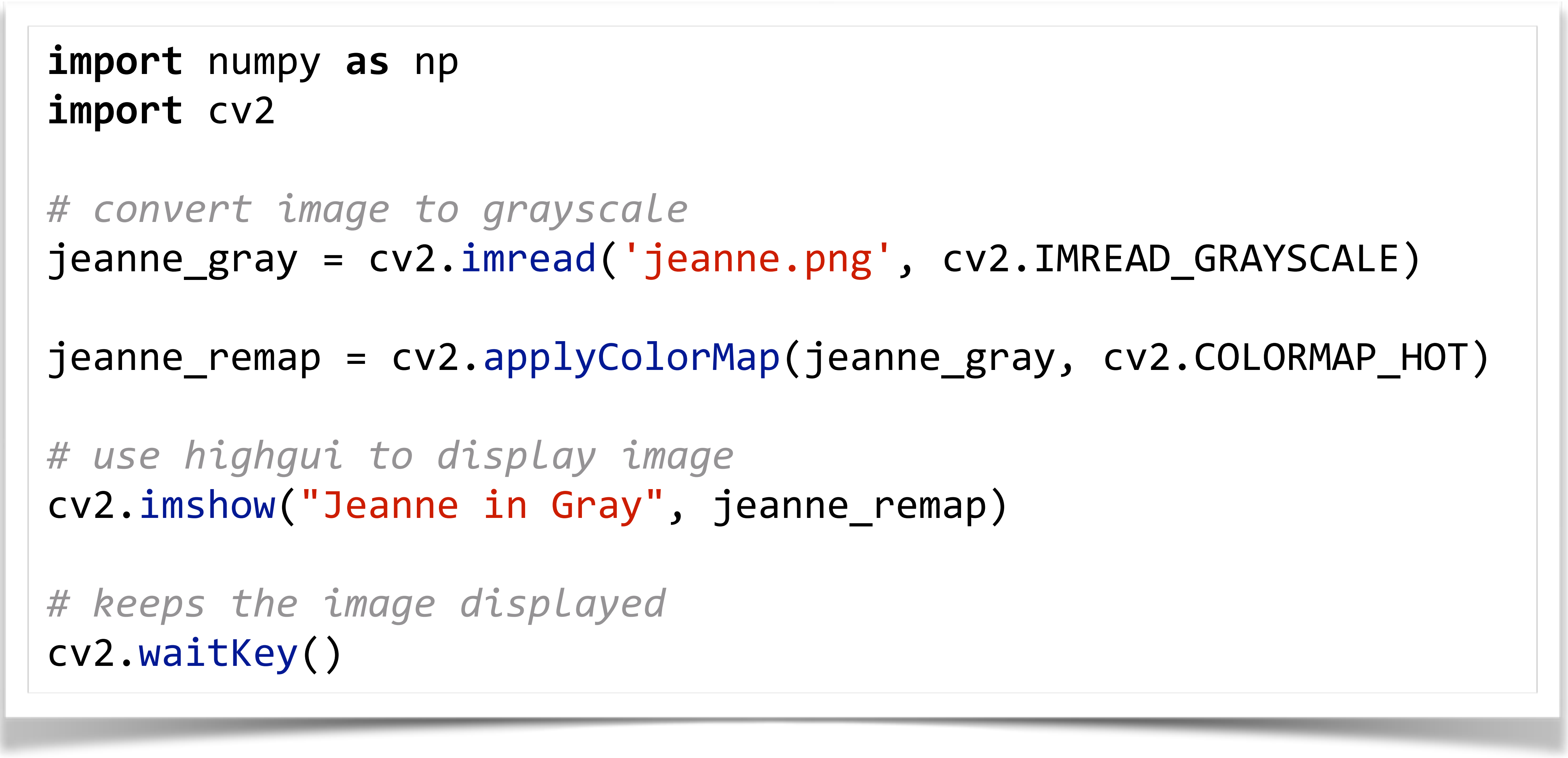
Color Maps

- OpenCV comes with various colormaps to enhance the visualization, use `applyColorMap()`



Value	Name	Scale
0	**COLORMAP_AUTUMN**	
1	**COLORMAP_BONE**	
2	**COLORMAP_JET**	
3	**COLORMAP_WINTER**	
4	**COLORMAP_RAINBOW**	
5	**COLORMAP_OCEAN**	
6	**COLORMAP_SUMMER**	
7	**COLORMAP_SPRING**	
8	**COLORMAP_COOL**	
9	**COLORMAP_HSV**	
10	**COLORMAP_PINK**	
11	**COLORMAP_HOT**	

ColorMap Example

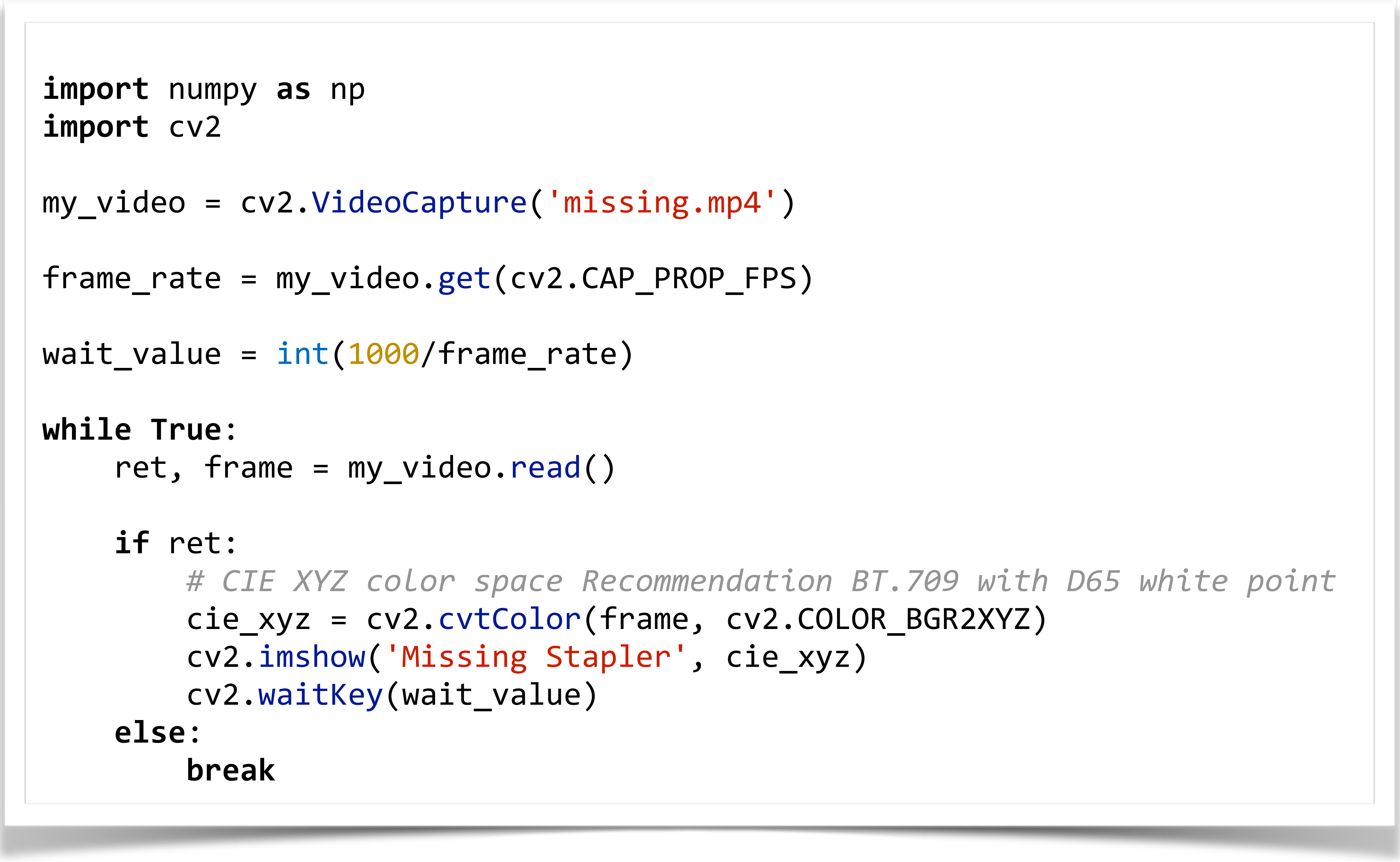


```python
import numpy as np
import cv2

# convert image to grayscale
jeanne_gray = cv2.imread('jeanne.png', cv2.IMREAD_GRAYSCALE)

jeanne_remap = cv2.applyColorMap(jeanne_gray, cv2.COLORMAP_HOT)

# use highgui to display image
cv2.imshow("Jeanne in Gray", jeanne_remap)

# keeps the image displayed
cv2.waitKey()
```

Real-time video manipulation

- OpenCV's `VideoCapture` class
  - `read()` method returns a tuple: `(return, frame)`
    - first value is a success flag, and the second is the image (as an `ndarray`).
    - each frame is an image that we can display with `imshow()`

Example



```python
import numpy as np
import cv2

my_video = cv2.VideoCapture('missing.mp4')

frame_rate = my_video.get(cv2.CAP_PROP_FPS)

wait_value = int(1000/frame_rate)

while True:
    ret, frame = my_video.read()

    if ret:
        # CIE XYZ color space Recommendation BT.709 with D65 white point
        cie_xyz = cv2.cvtColor(frame, cv2.COLOR_BGR2XYZ)
        cv2.imshow('Missing Stapler', cie_xyz)
        cv2.waitKey(wait_value)
    else:
        break
```