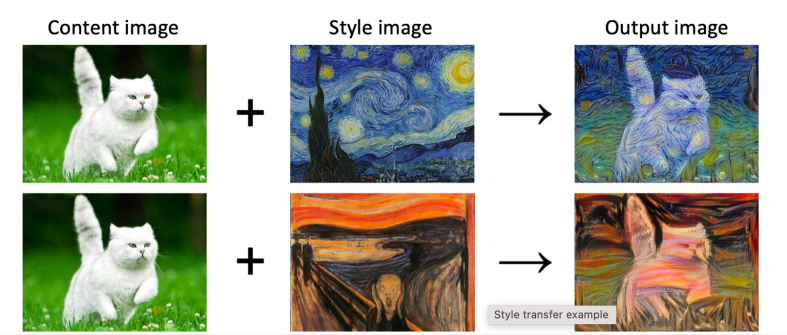
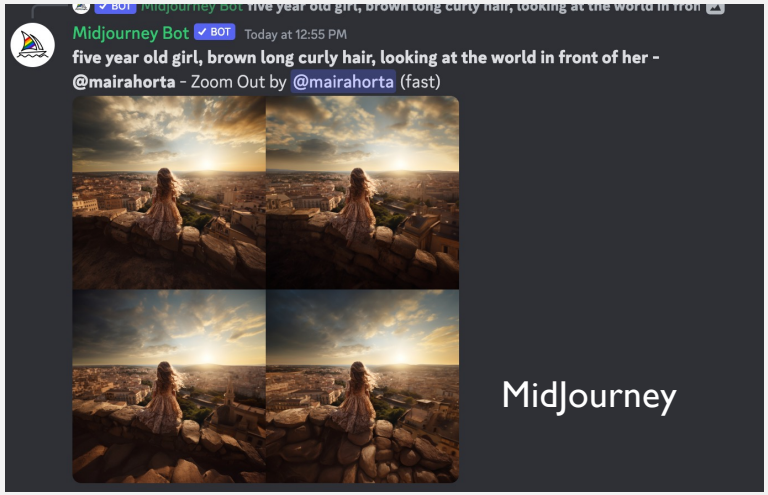
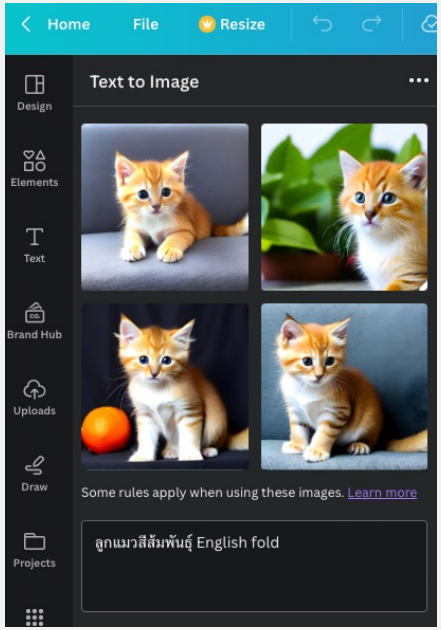
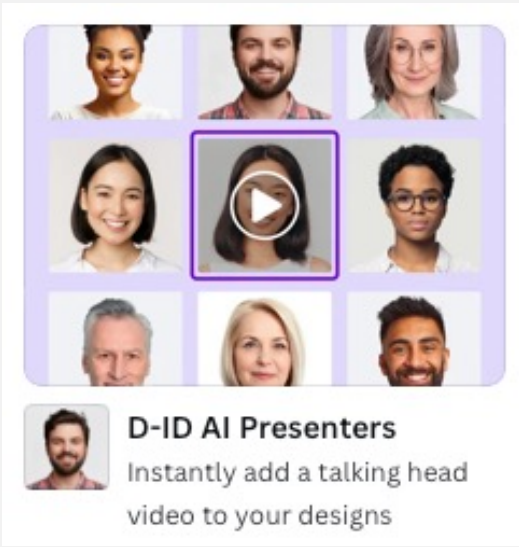
IMAGE GENERATIVE AI







Canva

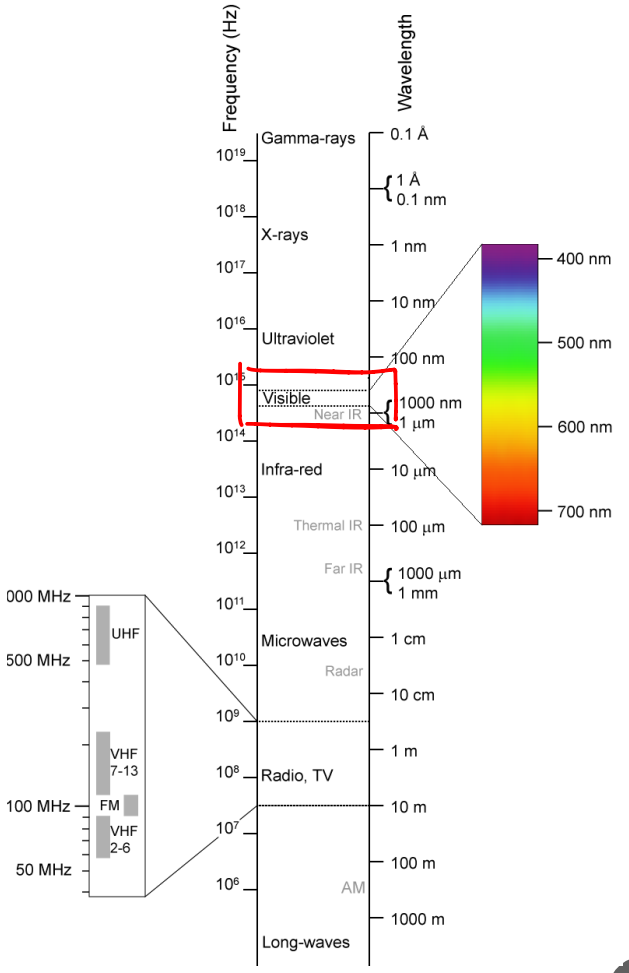


Canva/Heygen

EXAMPLES OF FIELDS THAT USE DIGITAL IMAGE PROCESSING

EXAMPLES OF FIELDS THAT USE DIGITAL IMAGE PROCESSING

- To categorize by the sources of the image
  - Electromagnetic Energy Spectrum → X-ray and visual bands
  - Others: Acoustic, ultrasonic and electronic (electron beams used in microscopy) image from these band
  - Synthetic images for modeling and visualization



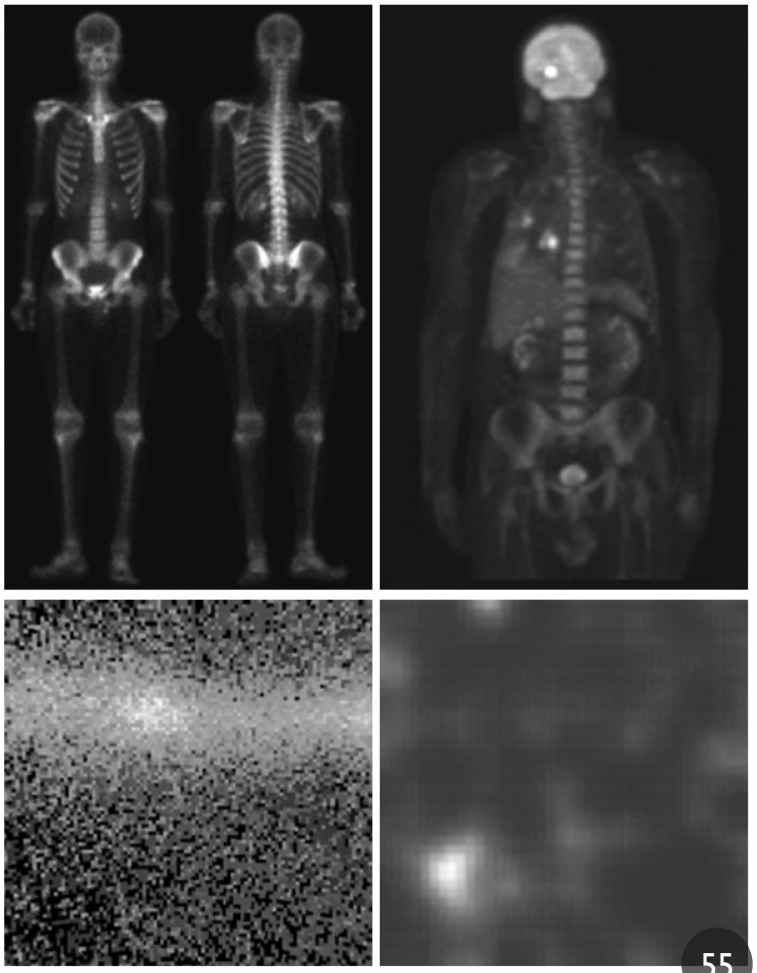
Images from Gonzalez & Woods, Digital Image Processing, second edition

GAMMA-RAY IMAGING

- Nuclear medicine and astronomical observations
  - (a) Locate bone pathology, such as infections, tumors
  - (b) Positron emission tomography (PET) using radioactive isotope
  - (c) Cygnus Loop in gamma ray bands
  - (d) gamma radiation from a valve in a nuclear reactor

a b
c d

FIGURE 1.6 Examples of gamma-ray imaging. (a) Bone scan. (b) PET image. (c) Cygnus Loop. (d) Gamma radiation (bright spot) from a reactor valve. (Images courtesy of (a) G.E. Medical Systems, (b) Dr. Michael E. Casey, CTI PET Systems, (c) NASA, (d) Professors Zhong He and David K. Wehe, University of Michigan.)



แผ่นชิงสี : แสงตกกระทบ

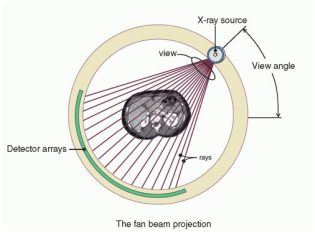
PET scan machine from
https://www.healthline.com/health/pet-scan

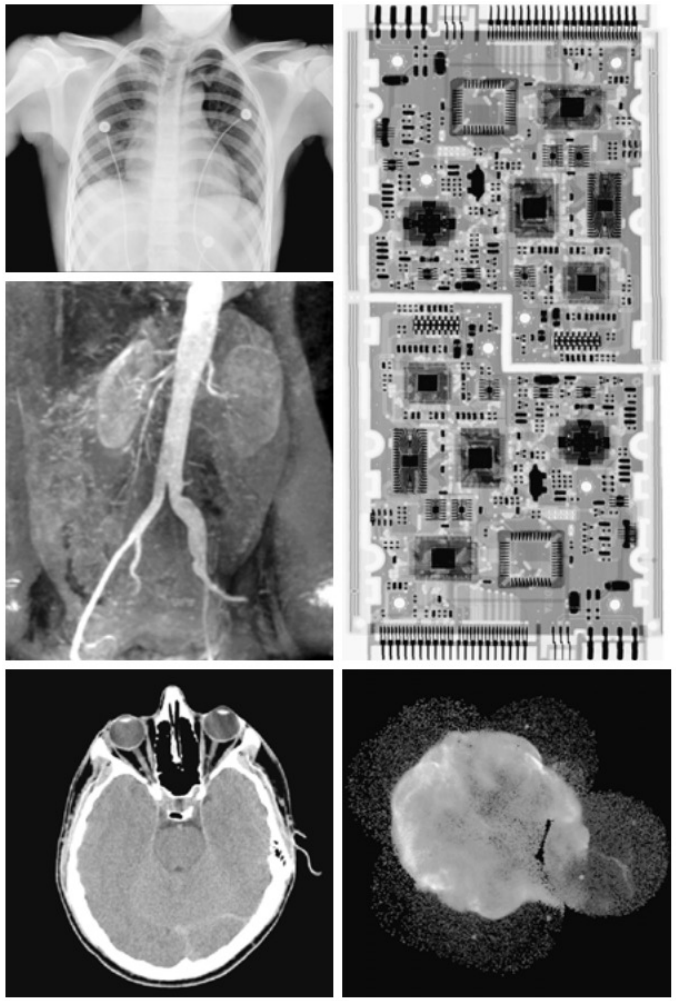
Images from Gonzalez & Woods, Digital Image Processing, second edition

X-RAY IMAGING

- Medical diagnostics, astronomy, industrial imaging
- X-ray generated by replacing patient between X-ray source and a film (phosphor screen) sensitive to X-ray energy
- Angiography (Contrast enhancement of blood vessels)
- Computerized axial tomography (CAT) – slice taken perpendicularly through the patient -> 3D rendition

ดูดซึม : ดูดซึมกลับ





a b c d e

FIGURE 1.7 Examples of X-ray imaging. (a) Chest X-ray. (b) Aortic angiogram. (c) Head CT. (d) Circuit boards. (e) Cygnus Loop. (Images courtesy of (a) and (c) Dr. David R. Pickens, Dept. of Radiology & Radiological Sciences, Vanderbilt University Medical Center, (b) Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, (d) Mr. Joseph E. Pascente, Lixi, Inc., and (e) NASA.)

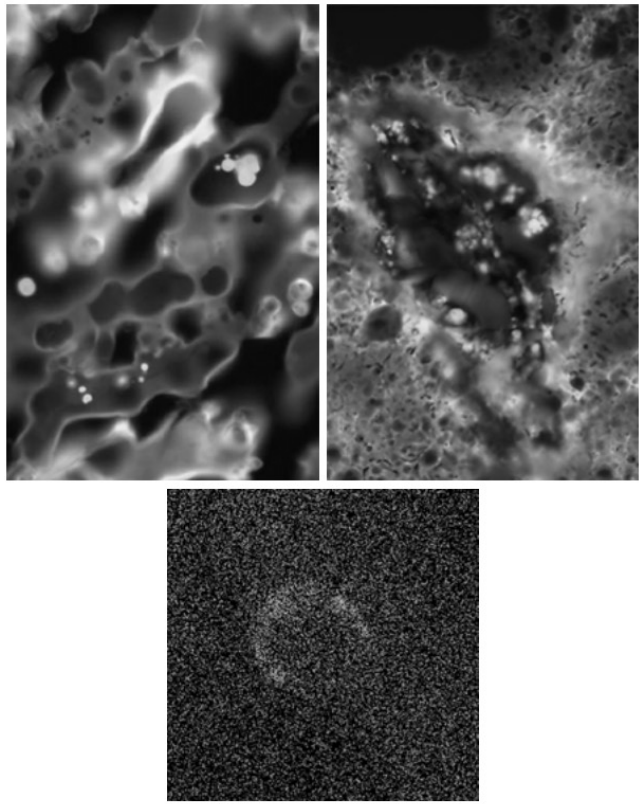
Images from Gonzalez & Woods, Digital Image Processing, second edition
https://en.wikipedia.org/wiki/CT_scan
https://radiologykey.com/computed-tomography-15/

IMAGING IN ULTRAVIOLET BAND

- Industrial inspection, microscopy, lasers, biological imaging, astronomical observations
- Fluorescent Microscopy – studying materials, such as corn smut (disease of the corn)
- Cygnus Loop again in Ultraviolet band

a b
c

FIGURE 1.8
Examples of ultraviolet imaging.
(a) Normal corn.
(b) Smut corn.
(c) Cygnus Loop.
(Images courtesy of (a) and (b) Dr. Michael W. Davidson, Florida State University, (c) NASA.)

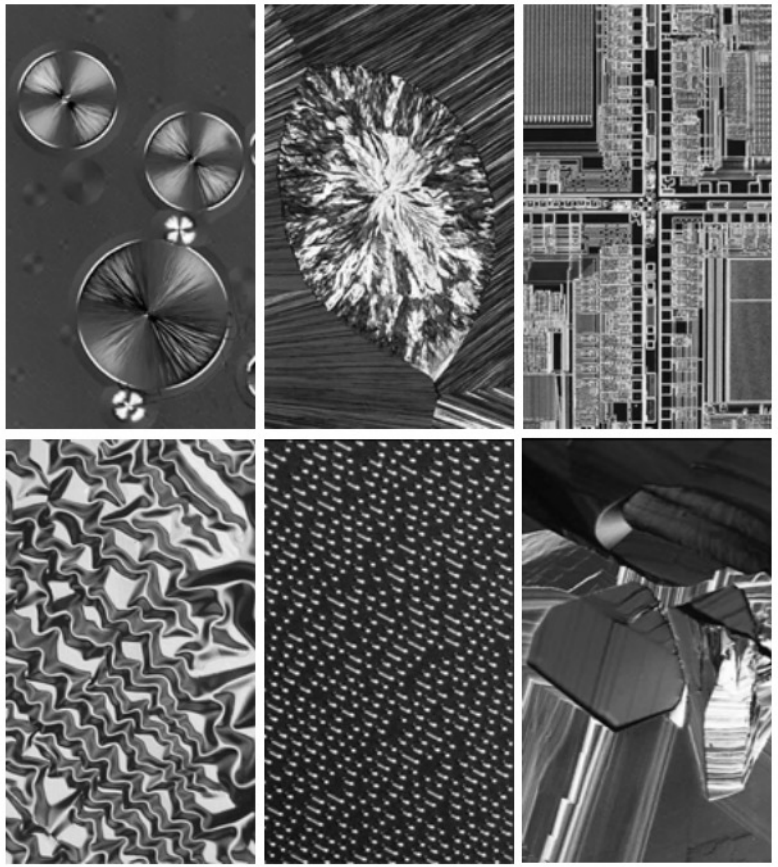


Images from Gonzalez & Woods, Digital Image Processing, second edition

IMAGING IN VISIBLE AND INFRARED BANDS

- Light microscopy imaging
  - Phamaceuticals and microinspection to materials characterization

https://www.carlroth.com



a b c
d e f

FIGURE 1.9 Examples of light microscopy images. (a) Taxol (anticancer agent), magnified 250×. (b) Cholesterol—40×. (c) Microprocessor—60×. (d) Nickel oxide thin film—600×. (e) Surface of audio CD—1750×. (f) Organic superconductor—450×. (Images courtesy of Dr. Michael W. Davidson, Florida State University.)

Images from Gonzalez & Woods, Digital Image Processing, second edition

IMAGING IN VISIBLE AND INFRARED BANDS

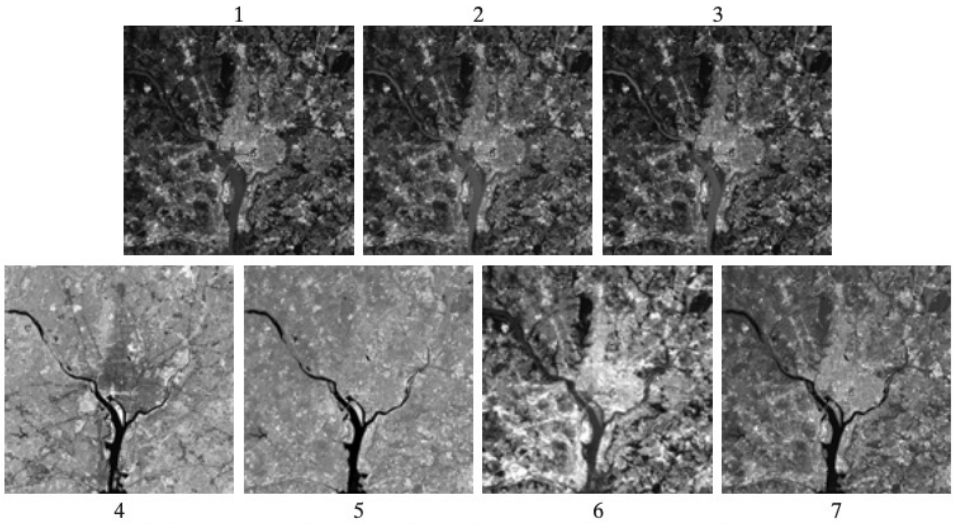


FIGURE 1.10 LANDSAT satellite images of the Washington, D.C. area. The numbers refer to the thematic bands in Table 1.1. (Images courtesy of NASA.)

Images from Gonzalez & Woods, Digital Image Processing, second edition

- Remote sensing – Thematic bands in NASA's LANDSAT satellite
  - Multispectrum imaging

Thematic bands of NASA are as LANDSAT satellite.

Band No.	Name	Wavelength (μm)	Characteristics and use
1	Visible blue	0.45–0.52	Maximum water penetration
2	Visible green	0.52–0.60	Good for measuring plant vigor
3	Visible blue	0.63–0.69	Vegetation discrimination
4	Near infrared	0.76–0.90	Biomass and shoreline mapping
5	Middle infrared	1.55–1.75	Moisture content of soil
6	Thermal infrared	10.4–12.5	Soil moisture, thermal mapping
7	Middle infrared	2.08–2.35	Mineral mapping

IMAGING IN VISIBLE AND INFRARED BANDS

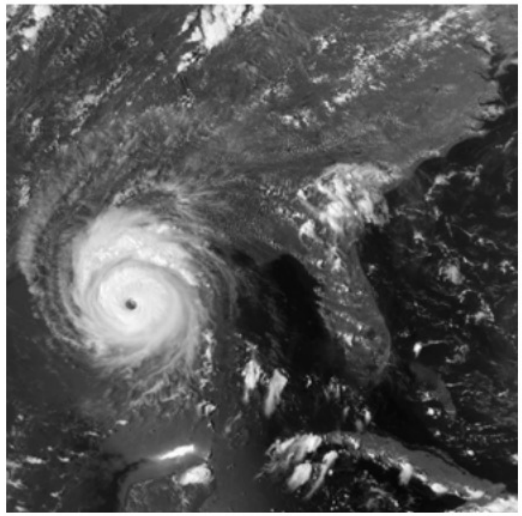


FIGURE 1.11 Multispectral image of Hurricane Andrew taken by NOAA GEOS (Geostationary Environmental Operational Satellite) sensors. (Courtesy of NOAA.)

Images from Gonzalez & Woods, Digital Image Processing, second edition

- Weather observation – multispectral imaging
  - Sensors in visible and infrared bands
- Nighttime lights of the world dataset (next page)

IMAGING IN VISIBLE AND INFRARED BANDS

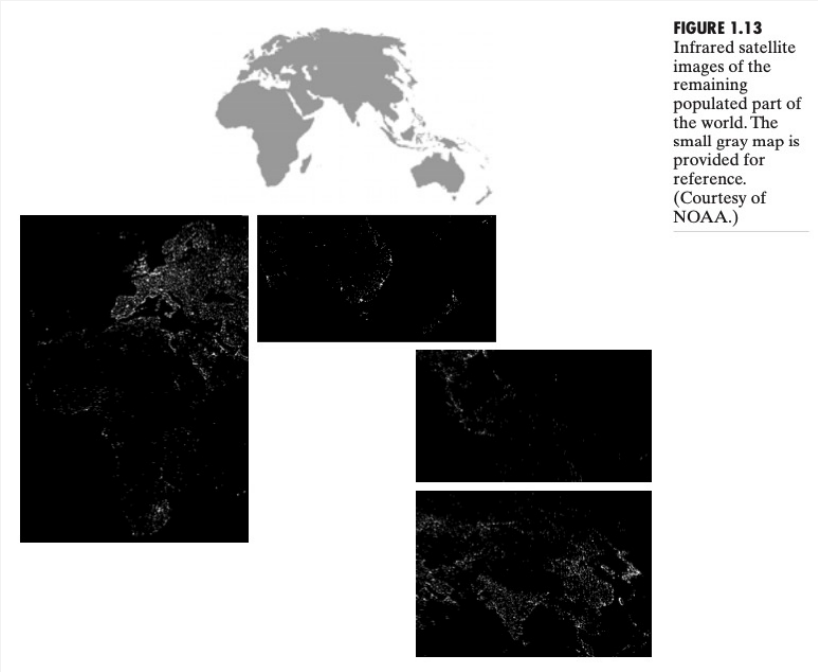


FIGURE 1.13 Infrared satellite images of the remaining populated part of the world. The small gray map is provided for reference. (Courtesy of NOAA.)

Part of the *Nighttime Lights of the World* dataset

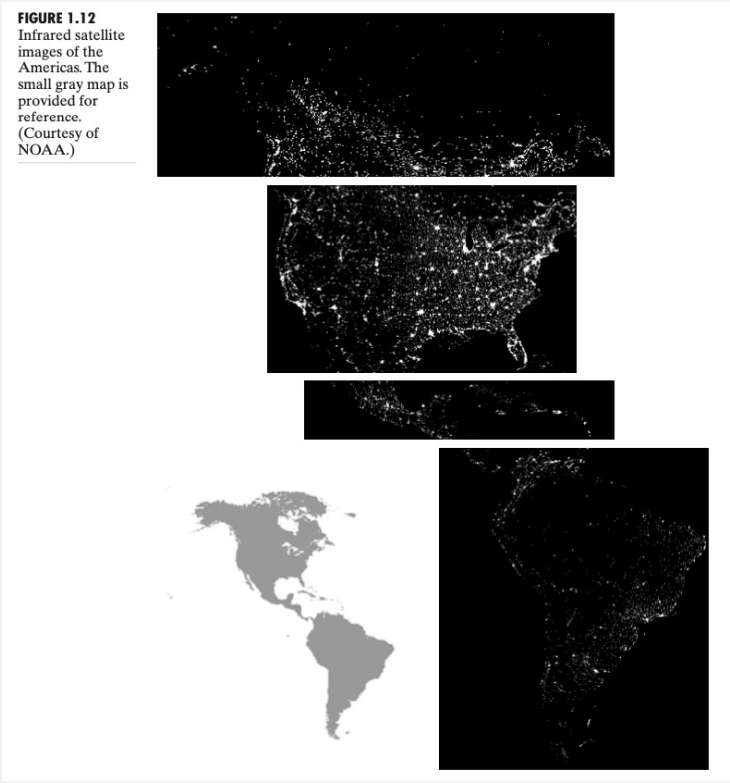


FIGURE 1.12 Infrared satellite images of the Americas. The small gray map is provided for reference. (Courtesy of NOAA.)

Images from Gonzalez & Woods, Digital Image Processing, second edition

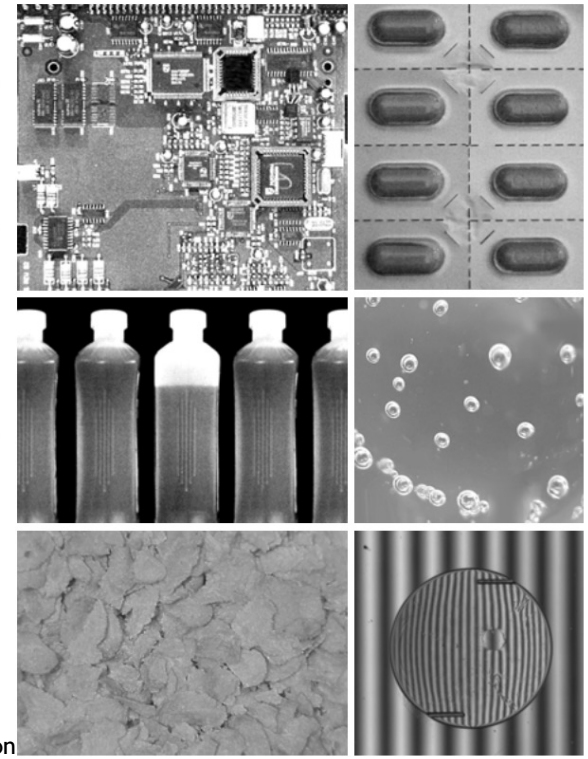
IMAGING IN VISIBLE AND INFRARED BANDS

a b
c
d

FIGURE 1.15
Some additional examples of imaging in the visual spectrum. (a) Thumb print. (b) Paper currency. (c) and (d). Automated license plate reading. (Figure (a) courtesy of the National Institute of Standards and Technology. Figures (c) and (d) courtesy of Dr. Juan Herrera, Perceptics Corporation.)

a b
c d
e f

FIGURE 1.14
Some examples of manufactured goods often checked using digital image processing. (a) A circuit board controller. (b) Packaged pills. (c) Bottles. (d) Bubbles in clear-plastic product. (e) Cereal. (f) Image of intraocular implant. (Fig. (f) courtesy of Mr. Pete Sites, Perceptics Corporation.)



Images from Gonzalez & Woods, Digital Image Processing, second edition

MICROWAVE BAND

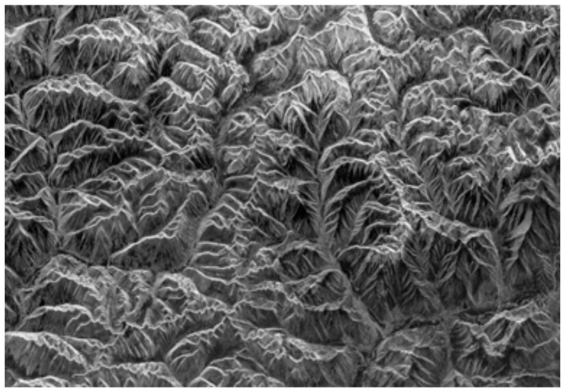
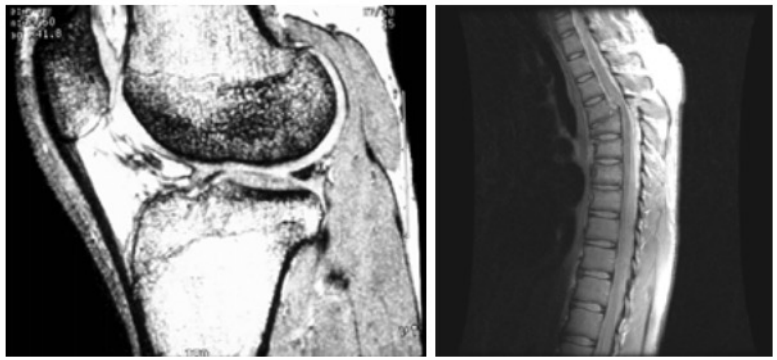


FIGURE 1.16 Spaceborne radar image of mountains in southeast Tibet. (Courtesy of NASA.)

- Radar – to collect data over virtually any region anytime regardless of weather lighting conditions
- See through vegetation, ice and dry sand
- Explore inaccessible regions of Earth's surface

Images from Gonzalez & Woods, Digital Image Processing, second edition

IMAGING IN RADIO BAND



a b

FIGURE 1.17 MRI images of a human (a) knee, and (b) spine. (Image (a) courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, and (b) Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

medical

- Magnetic Resonance Imaging (MRI) – placing a patient in a powerful magnet and passes radio waves through his/her body in short pulses

Images from Gonzalez & Woods, Digital Image Processing, second edition

IMAGING IN OTHER MODALITIES

- Imaging using sound – geological exploration, industry and medicine

low freq sound wave

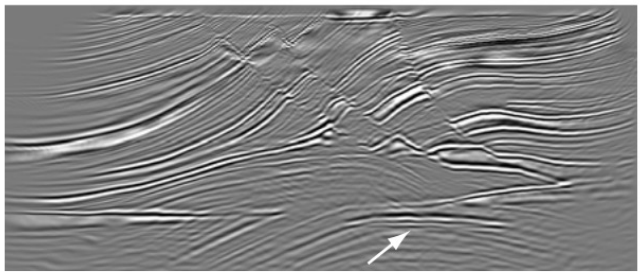
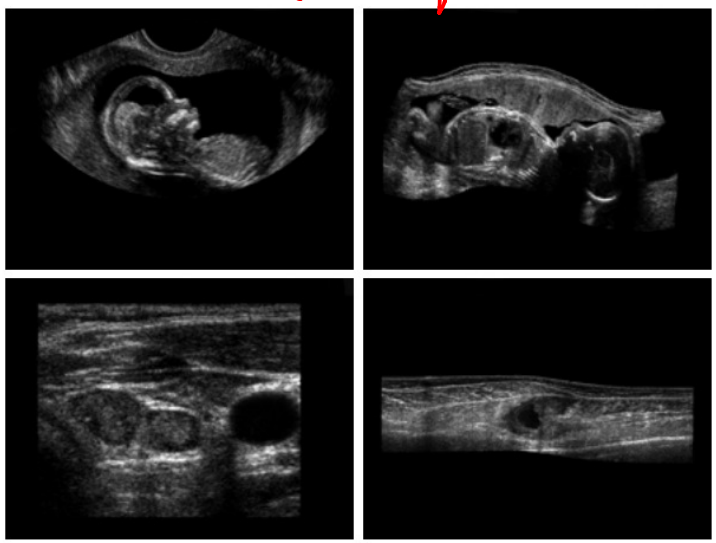


FIGURE 1.19 Cross-sectional image of a seismic model. The arrow points to a hydrocarbon (oil and/or gas) trap. (Courtesy of Dr. Curtis Ober, Sandia National Laboratories.)

High freq sound wave



a b
c d

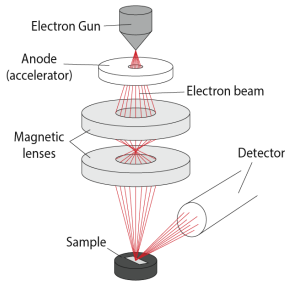
FIGURE 1.20 Examples of ultrasound imaging. (a) Baby. (2) Another view of baby. (c) Thyroids. (d) Muscle layers showing lesion. (Courtesy of Siemens Medical Systems, Inc., Ultrasound Group.)

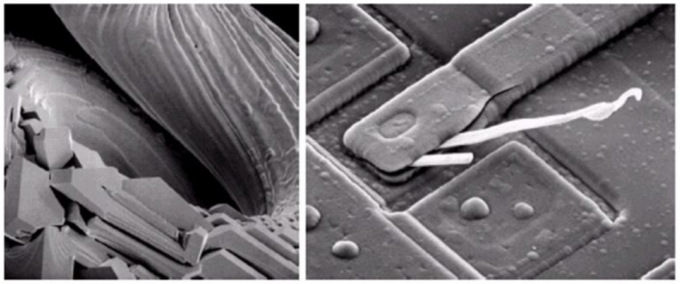
more safe than electromagnetic wave

Images from Gonzalez & Woods, Digital Image Processing, second edition

OTHER IMAGING MODALITIES

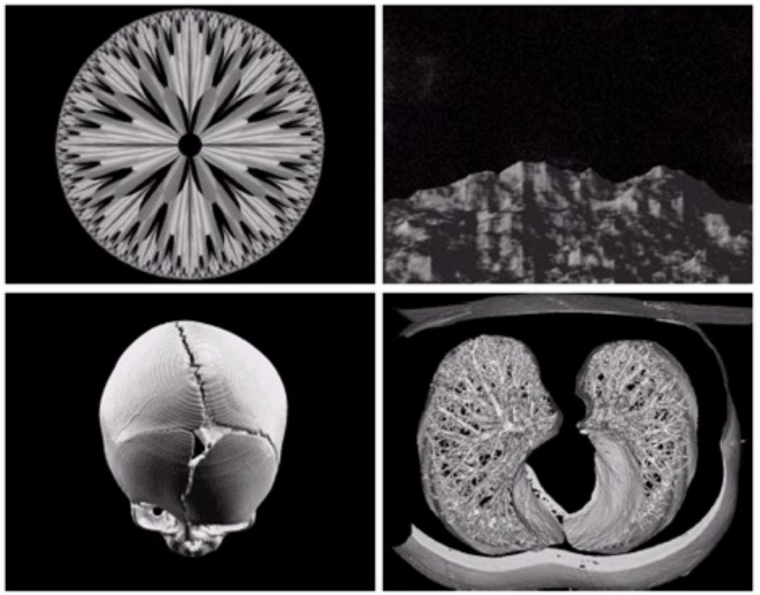
- Electron microscopy
- Computer-generated image





a b

FIGURE 1.21 (a) 250× SEM image of a tungsten filament following thermal failure. (b) 2500× SEM image of damaged integrated circuit. The white fibers are oxides resulting from thermal destruction. (Figure (a) courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene; (b) courtesy of Dr. J. M. Hudak, McMaster University, Hamilton, Ontario, Canada.)



a b
c d

FIGURE 1.22
(a) and (b) Fractal images. (c) and (d) Images generated from 3-D computer models of the objects shown. (Figures (a) and (b) courtesy of Ms. Melissa D. Binde, Swarthmore College, (c) and (d) courtesy of NASA.)

Images from - Gonzalez & Woods, Digital Image Processing, second edition
- https://www.eng-atoms.msm.cam.ac.uk/RoyalSocDemos/SEM

REFERENCES

- Chapter 1, Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison- Wesley

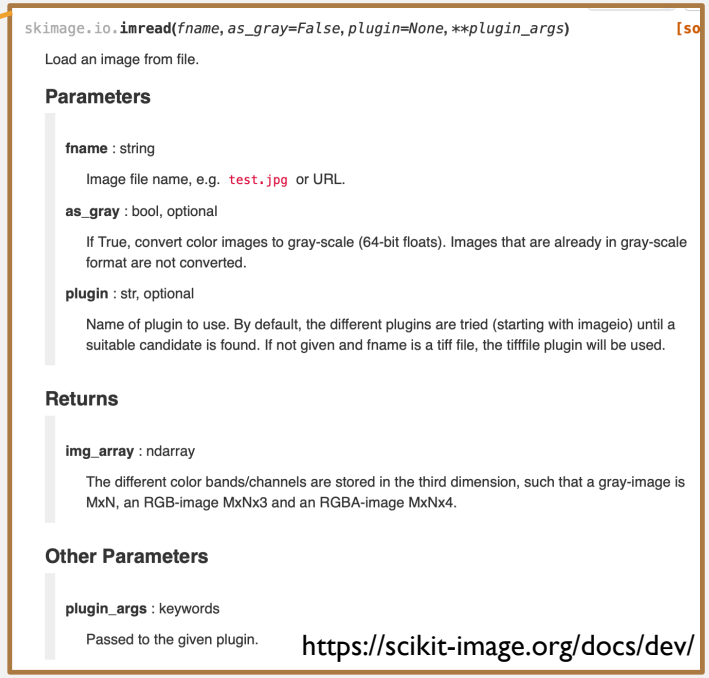
EXAMPLE #1: READ AND SHOW IMAGE

Warm up!

- Read and show an image
  - ***scikit-image*** is a collection of algorithms for image processing.

```
from skimage import io
img = io.imread("kitty.jpg")
io.imshow(img)
io.show()

import matplotlib.pyplot as plt
plt.imshow(img)
plt.show()
```



skimage.io.**imread**(*fname*, *as_gray=False*, *plugin=None*, ***plugin_args*) [so

Load an image from file.

Parameters

fname : string

Image file name, e.g. `test.jpg` or URL.

as_gray : bool, optional

If True, convert color images to gray-scale (64-bit floats). Images that are already in gray-scale format are not converted.

plugin : str, optional

Name of plugin to use. By default, the different plugins are tried (starting with imageio) until a suitable candidate is found. If not given and fname is a tiff file, the tifffile plugin will be used.

Returns

img_array : ndarray

The different color bands/channels are stored in the third dimension, such that a gray-image is MxN, an RGB-image MxNx3 and an RGBA-image MxNx4.

Other Parameters

plugin_args : keywords

Passed to the given plugin.

https://scikit-image.org/docs/dev/

GET PIXEL VALUES

```
from skimage import io
import numpy as np

img = io.imread("kitty.png")

img[0,0]
array([184, 169, 148], dtype=uint8)

img[0,0,0:2]
array([184, 169], dtype=uint8)
```

NumPy is the fundamental package for scientific computing in Python.
At the core of the ***NumPy*** package, is the ***ndarray*** object.
http://numpy.org

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
https://matplotlib.org

matplotlib.pyplot.imshow

`matplotlib.pyplot.`imshow(*X*, *cmap=None*, *norm=None*, *aspect=None*, *interpolation=None*, *alpha=None*, *vmin=None*, *vmax=None*, *origin=None*, *extent=None*, ***, *filternorm=True*, *filterrad=4.0*, *resample=None*, *url=None*, *data=None*, ***kwargs*) [source]

Display data as an image, i.e., on a 2D regular raster.

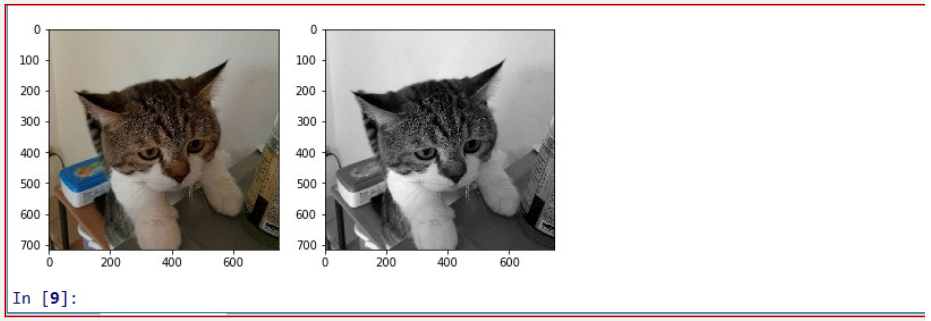
EXERCISE #1 READ AND SHOW IMAGE

Diary #1

- Compared RGB image and gray image

```
from skimage import io, color
import matplotlib.pyplot as plt
img = io.imread("kitty.png")
gray = color.rgb2gray(_____________)

fig = plt.figure(figsize=(8,4))
fig.add_subplot(1,2,1)
plt.imshow(img)
fig.add_subplot(1,2,2)
plt.imshow(_____________________________)
io.show()
```



rgb2gray

skimage.color.**rgb2gray**(*rgb*, *, *channel_axis=- 1*) [source]

Compute luminance of an RGB image.

Parameters

rgb : (…, 3, …) array_like

The image in RGB format. By default, the final dimension denotes channels.

Returns

out : ndarray

The luminance image - an array which is the same size as the input array, but with the channel dimension removed.

Raises

ValueError

If *rgb* is not at least 2-D with shape (…, 3, …).

https://scikit-image.org/docs/dev/

QUESTIONS

- What is your image size?
- What is your image data type?
- What is value at `kitty image at [50,50,2]`?
- Which one is Red/Green/blue?
- What is the maximum value of each RGB and what color is that?
- What is the minimum value of each RGB and what color is that?

EXERCISE #2 READ AND SHOW IMAGE USING CV2

- Read an image using opencv

```
import cv2
image2 = cv2.imread("kitty.jpg")

plt.imshow(image2)
plt.show()
```

- What do you see?
- Add this code:

```
image2 = cv2.cvtColor(image2,cv2.COLOR_BGR2RGB)
```

EXAMPLE #3 DICOM FILE

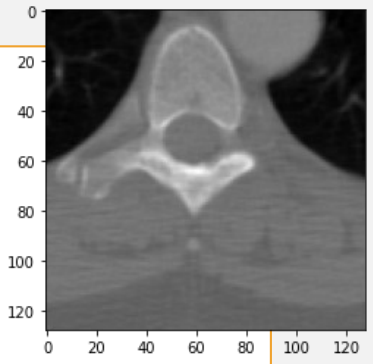
- Digital Imaging and Communications in Medicine (DICOM) format
  - Image and patient's data

```python
import matplotlib.pyplot as plt
from pydicom import dcmread
from pydicom.data import get_testdata_file

fpath = get_testdata_file('CT_small.dcm')
ds = dcmread(fpath)

print(f"Patient's Name...: {ds.PatientName}")
print(f"Patient ID.......: {ds.PatientID}")
print(f"Modality.........: {ds.Modality}")
print(f"Study Date.......: {ds.StudyDate}")
print(f"Image size.......: {ds.Rows} x {ds.Columns}")
print(f"Pixel Spacing....: {ds.PixelSpacing}")

# plot the image using matplotlib
plt.imshow(ds.pixel_array, cmap=plt.cm.gray)
plt.show()
```



pydicom

https://pydicom.github.io/pydicom/stable/auto_examples/index.html

- What is the image size?
- What is the datatype of pixel_array ?
- Min/max?
- Try with "MR_small.dcm"

clear - clear console
who – get current variables
%reset – clear all variables

Python For Data Science *Cheat Sheet*

NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com

NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

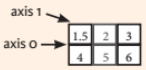
Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

1D array

2D array



3D array

Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4))                  Create an array of zeros
>>> np.ones((2,3,4),dtype=np.int16)  Create an array of ones
>>> d = np.arange(10,25,5)           Create an array of evenly
                                     spaced values (step value)
>>> np.linspace(0,2,9)               Create an array of evenly
                                     spaced values (number of samples)
>>> e = np.full((2,2),7)             Create a constant array
>>> f = np.eye(2)                    Create a 2X2 identity matrix
>>> np.random.random((2,2))          Create an array with random values
>>> np.empty((3,2))                  Create an empty array
```

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

```
>>> np.int64       Signed 64-bit integer types
>>> np.float32     Standard double-precision floating point
>>> np.complex     Complex numbers represented by 128 floats
>>> np.bool        Boolean type storing TRUE and FALSE values
>>> np.object      Python object type
>>> np.string_     Fixed-length string type
>>> np.unicode_    Fixed-length unicode type
```

Inspecting Your Array

```
>>> a.shape          Array dimensions
>>> len(a)           Length of array
>>> b.ndim           Number of array dimensions
>>> e.size           Number of array elements
>>> b.dtype          Data type of array elements
>>> b.dtype.name     Name of data type
>>> b.astype(int)    Convert an array to a different type
```

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

```
>>> g = a - b                        Subtraction
  array([[-0.5,  0. ,  0. ],
         [-3. , -3. , -3. ]])
>>> np.subtract(a,b)                 Subtraction
>>> b + a                            Addition
  array([[ 2.5,  4. ,  6. ],
         [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)                      Addition
>>> a / b                            Division
  array([[ 0.66666667,  1.        ,  1.        ],
         [ 0.25      ,  0.4       ,  0.5       ]])
>>> np.divide(a,b)                   Division
>>> a * b                            Multiplication
  array([[  1.5,   4. ,   9. ],
         [  4. ,  10. ,  18. ]])
>>> np.multiply(a,b)                 Multiplication
>>> np.exp(b)                        Exponentiation
>>> np.sqrt(b)                       Square root
>>> np.sin(a)                        Print sines of an array
>>> np.cos(b)                        Element-wise cosine
>>> np.log(a)                        Element-wise natural logarithm
>>> e.dot(f)                         Dot product
  array([[ 7.,  7.],
         [ 7.,  7.]])
```

Comparison

```
>>> a == b                           Element-wise comparison
  array([[False,  True,  True],
         [False, False, False]], dtype=bool)
>>> a < 2                            Element-wise comparison
  array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)             Array-wise comparison
```

Aggregate Functions

```
>>> a.sum()                          Array-wise sum
>>> a.min()                          Array-wise minimum value
>>> b.max(axis=0)                    Maximum value of an array row
>>> b.cumsum(axis=1)                 Cumulative sum of the elements
>>> a.mean()                         Mean
>>> b.median()                       Median
>>> a.corrcoef()                     Correlation coefficient
>>> np.std(b)                        Standard deviation
```

Copying Arrays

```
>>> h = a.view()          Create a view of the array with the same data
>>> np.copy(a)            Create a copy of the array
>>> h = a.copy()          Create a deep copy of the array
```

Sorting Arrays

```
>>> a.sort()              Sort an array
>>> c.sort(axis=0)        Sort the elements of an array's axis
```

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]                  Select the element at the 2nd index
  3
>>> b[1,2]                Select the element at row 1 column 2
  6.0                     (equivalent to b[1][2])
```

Slicing

```
>>> a[0:2]                Select items at index 0 and 1
  array([1, 2])
>>> b[0:2,1]              Select items at rows 0 and 1 in column 1
  array([ 2.,  5.])
>>> b[:1]                 Select all items at row 0
  array([[1.5, 2., 3.]])  (equivalent to b[0:1, :])
>>> c[1,...]              Same as [1,:,:]
  array([[[ 3.,  2.,  1.],
          [ 4.,  5.,  6.]]])
>>> a[ : :-1]             Reversed array a
  array([3,  2,  1])
```

Boolean Indexing

```
>>> a[a<2]                Select elements from a less than 2
  array([1])
```

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]   Select elements (1,0),(0,1),(1,2) and (0,0)
  array([ 4. ,  2. ,  6. ,  1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]   Select a subset of the matrix's rows
  array([[ 4. ,  5. ,  6. ,  4. ],  and columns
         [ 1.5,  2. ,  3. ,  1.5],
         [ 4. ,  5. ,  6. ,  4. ],
         [ 1.5,  2. ,  3. ,  1.5]])
```

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)      Permute array dimensions
>>> i.T                      Permute array dimensions
```

Changing Array Shape

```
>>> b.ravel()                Flatten the array
>>> g.reshape(3,-2)          Reshape, but don't change data
```

Adding/Removing Elements

```
>>> h.resize((2,6))          Return a new array with shape (2,6)
>>> np.append(h,g)           Append items to an array
>>> np.insert(a, 1, 5)       Insert items in an array
>>> np.delete(a,[1])         Delete items from an array
```

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)   Concatenate arrays
  array([ 1,  2,  3, 10, 15, 20])
>>> np.vstack((a,b))               Stack arrays vertically (row-wise)
  array([[ 1. ,  2. ,  3. ],
         [ 1.5,  2. ,  3. ],
         [ 4. ,  5. ,  6. ]])
>>> np.r_[e,f]                     Stack arrays vertically (row-wise)
>>> np.hstack((e,f))               Stack arrays horizontally (column-wise)
  array([[ 7.,  7.,  1.,  0.],
         [ 7.,  7.,  0.,  1.]])
>>> np.column_stack((a,d))         Create stacked column-wise arrays
  array([[ 1, 10],
         [ 2, 15],
         [ 3, 20]])
>>> np.c_[a,d]                     Create stacked column-wise arrays
```

Splitting Arrays

```
>>> np.hsplit(a,3)           Split the array horizontally at the 3rd
  [array([1]),array([2]),array([3])]  index
>>> np.vsplit(c,2)           Split the array vertically at the 2nd index
[array([[[ 1.5,  2. ,  1. ],
         [ 4. ,  5. ,  6. ]]]),
 array([[[ 3.,  2.,  3.],
         [ 4.,  5.,  6.]]])]
```

NUMPY

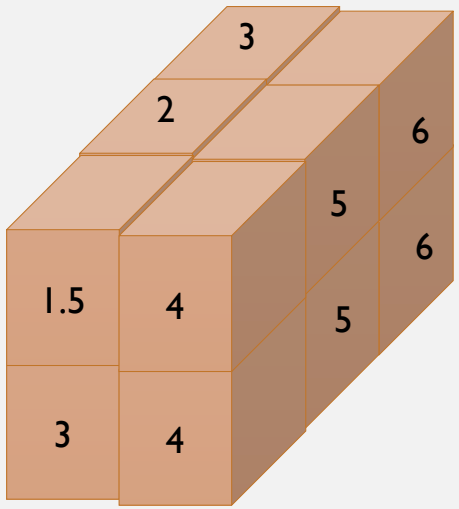
Warm up!

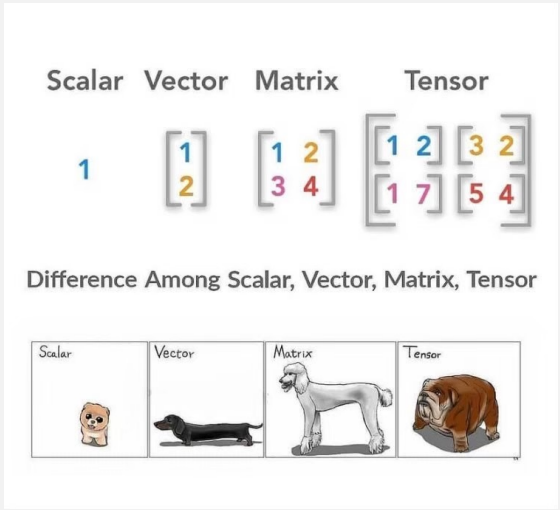
```
c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]], dtype = float)
array([[[ 1.5,  2. ,  3. ],
        [ 4. ,  5. ,  6. ]],

       [[ 3. ,  2. ,  1. ],
        [ 4. ,  5. ,  6. ]]])

c[0,0]
Out[31]: array([ 1.5,  2. ,  3. ])

c[0,1]
Out[32]: array([ 4.,  5.,  6.])
```





EXERCISE #4

- Create 150 x 150 black image

```
import numpy as np
import matplotlib.pyplot as plt

a = np.zeros((150,150),dtype=int)
plt.imshow(a,cmap='gray')
plt.show()
```

- Create an image of 100x100 which has red color on the half left and cyan color on the half right and display the image