Homework 7

due Wed Nov 2 at start of class

`Homework7.pdf`

`HW7.ipynb` or `HW7.py`

download from Brightspace

`Visualization.ipynb`
`Images.zip`

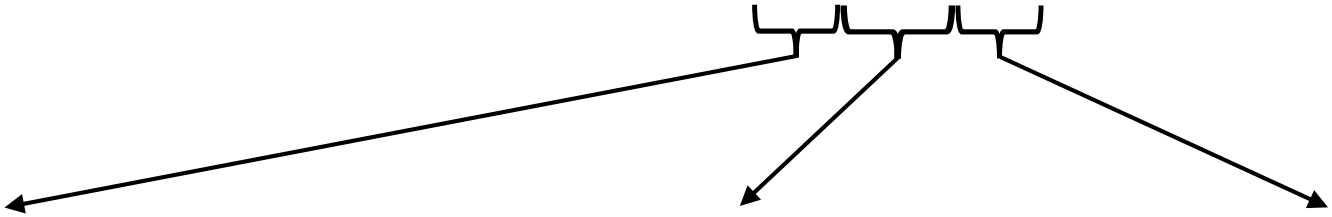
plt.subplots()

creates multiple plots (axes) on the same figure

```
fig, axs = plt.subplots(2, 2)
x = np.linspace(0, 10, 1000)
axs[0,0].plot(x, np.sin(x), 'r-')
axs[0,1].plot(x, np.cos(x), 'g-')
axs[1,0].plot(x, np.sqrt(x), 'b-')
axs[1,1].plot(x, np.log(x+1), 'b-')
```

line and symbol styles

```
ax.plot(x, x**4, 'b-.s')
```



color

`'b'`	blue
`'g'`	green
`'r'`	red
`'c'`	cyan
`'m'`	magenta
`'y'`	yellow
`'k'`	black
`'w'`	white

line style

`'-'`	solid
`':'`	dotted
`'--'`	dashed
`'-.'`	dash-dot

symbol style

`'.'`	dot
`'o'`	circle
`'s'`	square
`'v'`	upside-down triangle
`'^'`	triangle
`'<'`	left triangle
`'>'`	right triangle
`'8'`	octagon
`'p'`	pentagon
`'D'`	diamond

line and symbol styles

```
ax.plot(x, x**4, 'b-.s')
```

marker	symbol	description
"."		point
","		pixel
"o"		circle
"v"		triangle_down
"^"		triangle_up
"<"		triangle_left
">"		triangle_right
"1"		tri_down
"2"		tri_up
"3"		tri_left
"4"		tri_right
"8"		octagon
"s"		square
"p"		pentagon
"P"		plus (filled)
"*"		star
"h"		hexagon1
"H"		hexagon2
"+"		plus
"x"		x
"X"		x (filled)
"D"		diamond
"d"		thin_diamond
"|"		vline
"_"		hline

symbol style

- `'.'` dot
- `'o'` circle
- `'s'` square
- `'v'` upside-down triangle
- `'^'` triangle
- `'<'` left triangle
- `'>'` right triangle
- `'8'` octagon
- `'p'` pentagon
- `'D'` diamond

https://matplotlib.org/api/markers_api.html

line and symbol styles

```
ax.plot(x, x**2, color='sienna', linestyle='dashed', linewidth=4,
        marker='^', markersize=12)
```

R,**G**,**B** (proportion)

```
ax.plot(x, 1*x, color=(.3,.7,.5),  marker='^', markersize=12)
```

R,**G**,**B** (hex)

```
ax.plot(x, 2*x, color='#3f23a1',   marker='D', markersize=12)
```

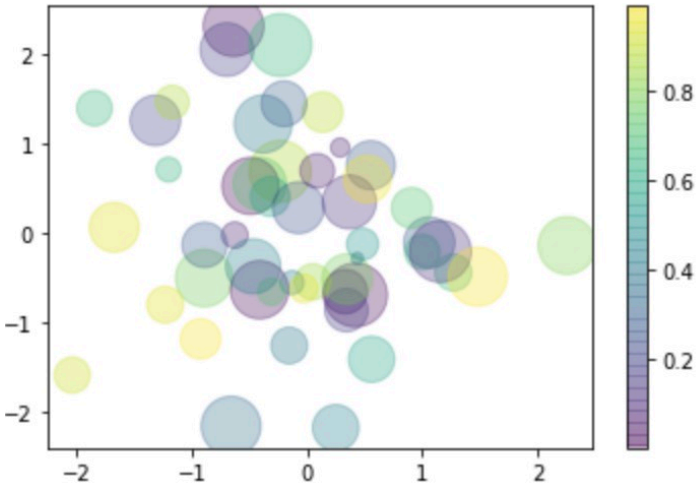
color name

```
ax.plot(x, 3*x, color='chocolate', marker='s', markersize=12)
```

https://matplotlib.org/3.3.1/tutorials/colors/colors.html

scatterplots

4-dimensional data plot




```
N = 50
x = R.randn(N)
y = R.randn(N)
colors = R.rand(N)
sizes = 1000 * R.rand(N)

pos = ax.scatter(x, y, c=colors, s=sizes,
                 alpha=0.3, cmap='viridis')

fig.colorbar(pos);
```

coords

color

size

transparent (0.0) to opaque (1.0)

colormap

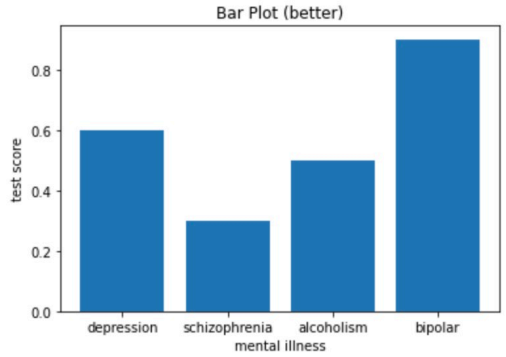
add a color bar

https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.scatter.html

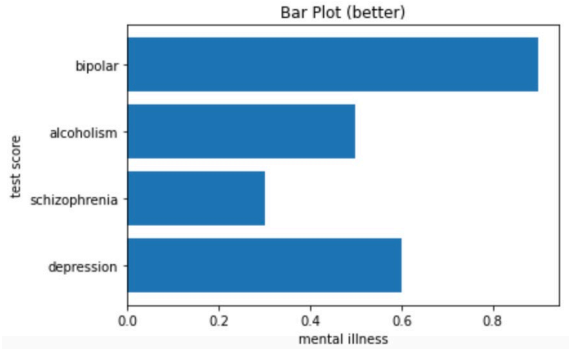
bar charts (categorical conditions)

```
illness = ['depression', 'schizophrenia',
           'alcoholism', 'bipolar']
score = [.6, .3, .5, .9]
```

```
ax.bar(illness, score)
```
vertical bar chart

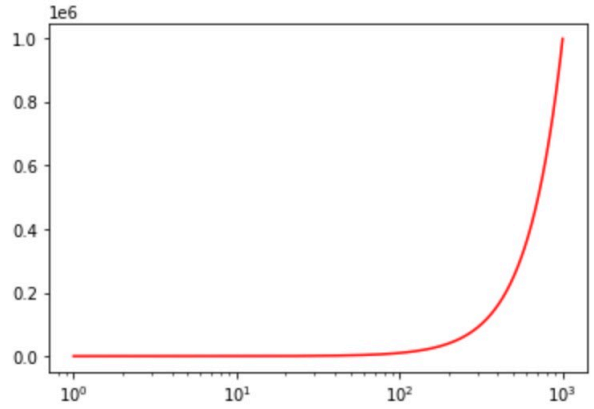



```
ax.barh(illness, score)
```
horizontal bar chart

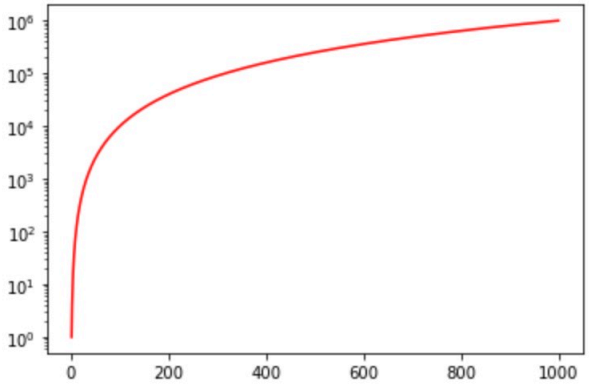


log axes

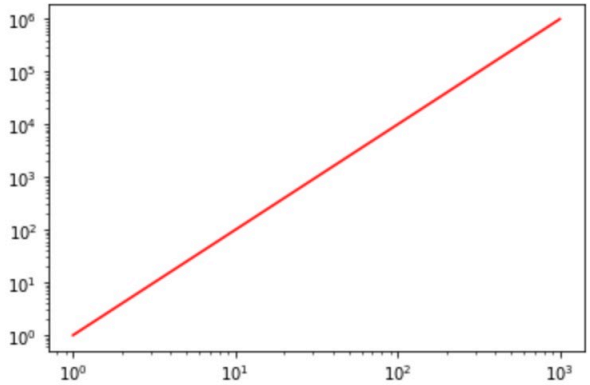
```
ax.semilogx(x, x**2, 'r-');
```




```
ax.semilogy(x, x**2, 'r-');
```

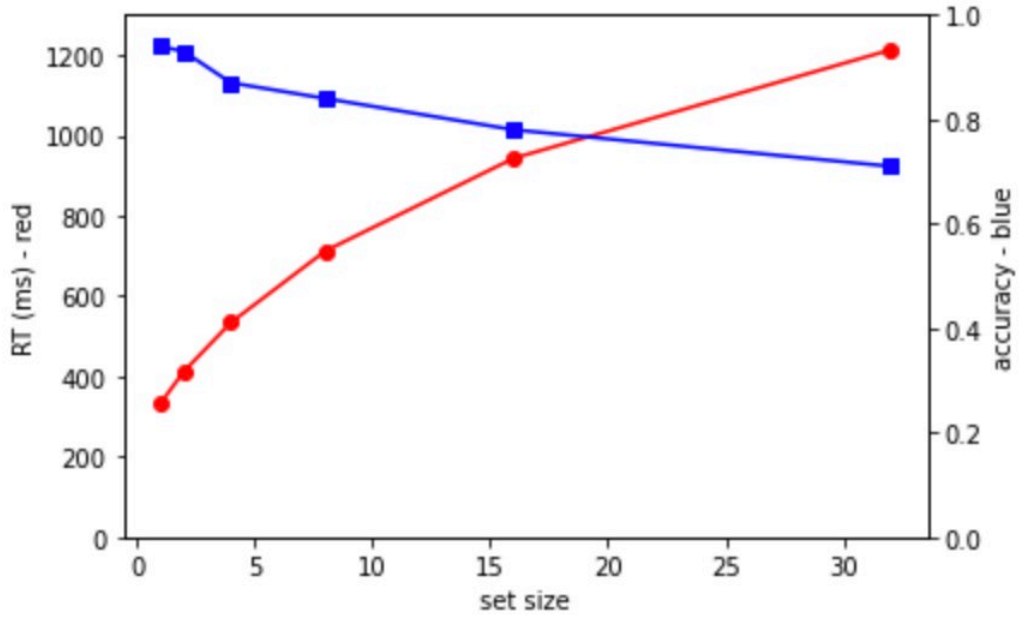



```
ax.loglog(x, x**2, 'r-');
```



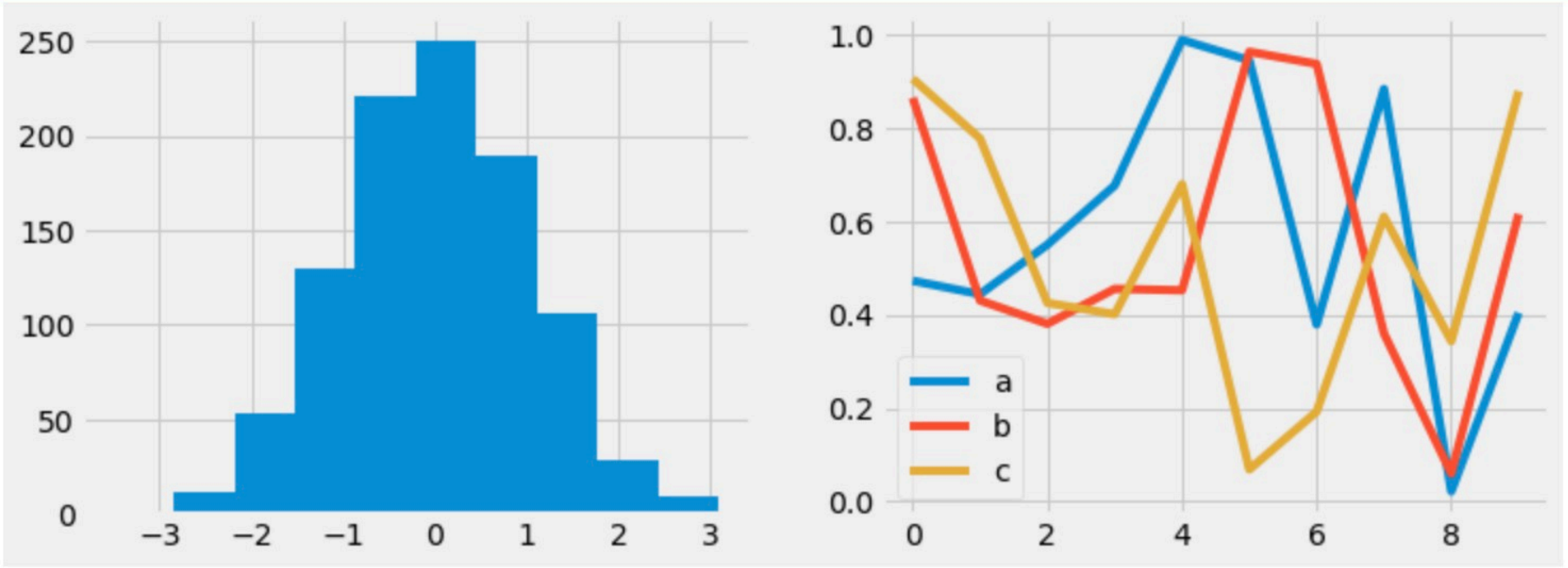
multiple axes on the same graph

```
fig = plt.figure()
ax1 = plt.axes()
ax2 = ax1.twinx()

ax1.plot(setsize, RT,  'r-o')
ax2.plot(setsize, acc, 'b-s')
```



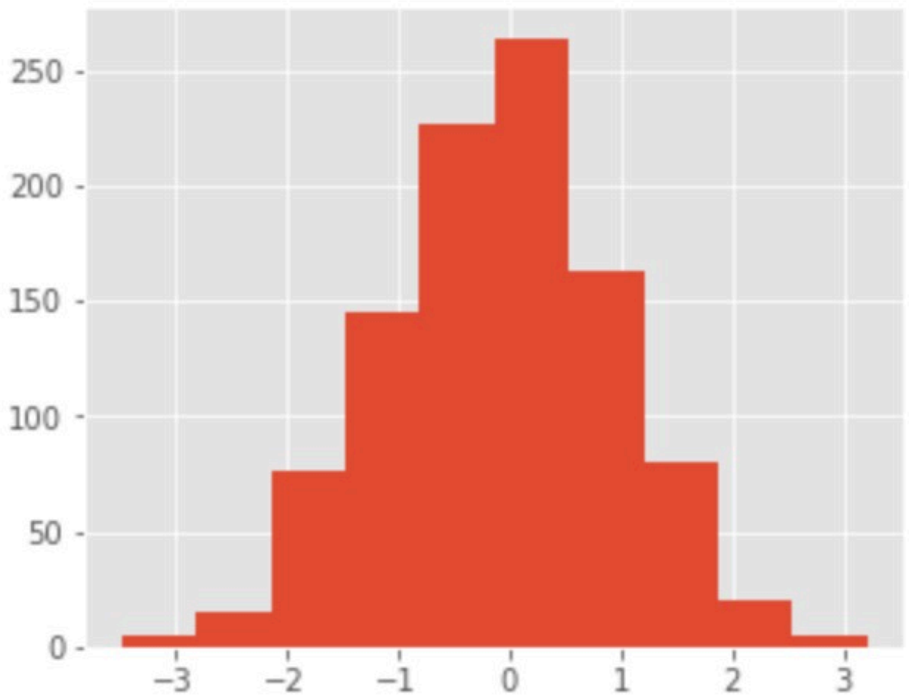
matplotlib styles

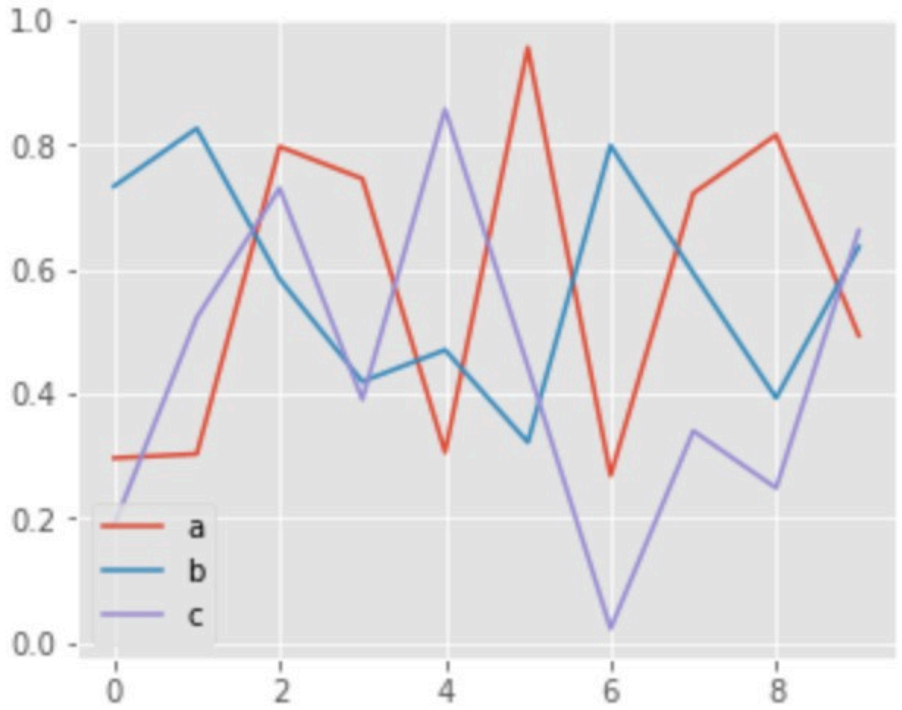
```
with plt.style.context('fivethirtyeight'):
    fig, ax = plt.subplots(1, 2, figsize=(11, 4))
    ax[0].hist(R.randn(1000))
    for i in range(3):
        ax[1].plot(R.rand(10))
        ax[1].legend(['a', 'b', 'c'], loc='lower left');
```



matplotlib styles

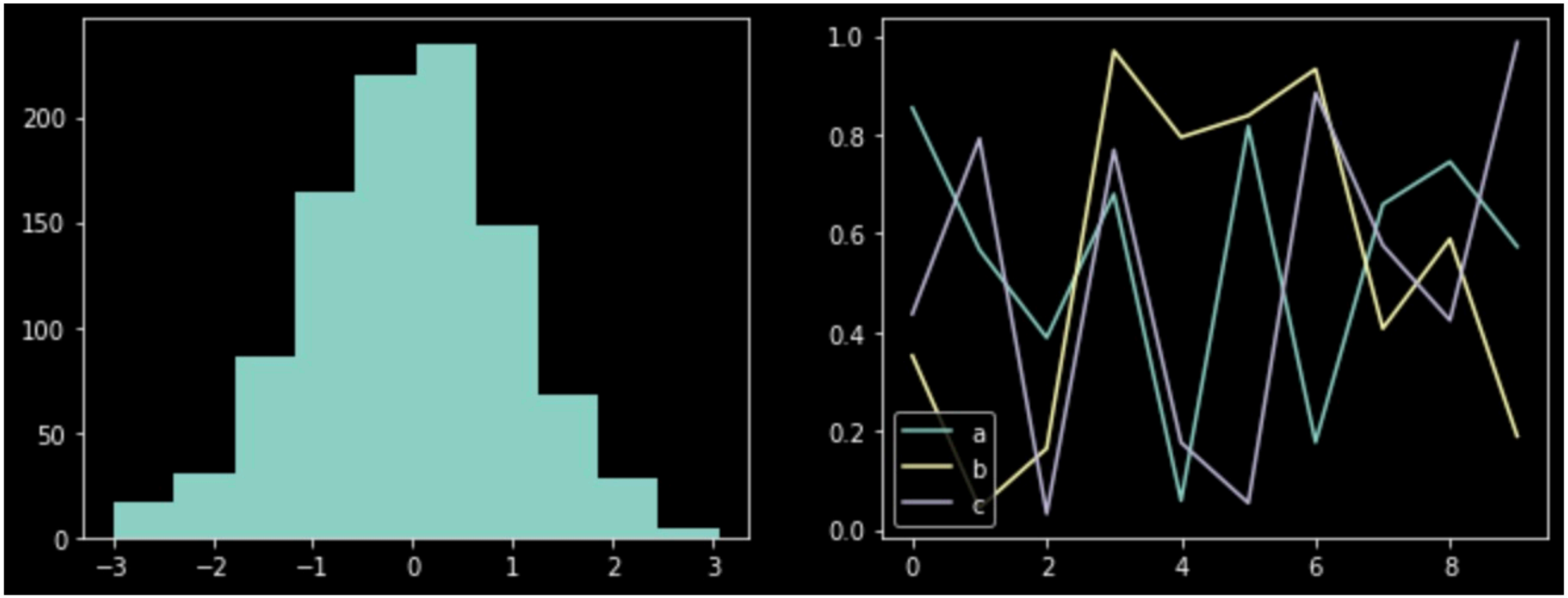
```
with plt.style.context('ggplot'):
    fig, ax = plt.subplots(1, 2, figsize=(11, 4))
    ax[0].hist(R.randn(1000))
    for i in range(3):
        ax[1].plot(R.rand(10))
        ax[1].legend(['a', 'b', 'c'], loc='lower left');
```





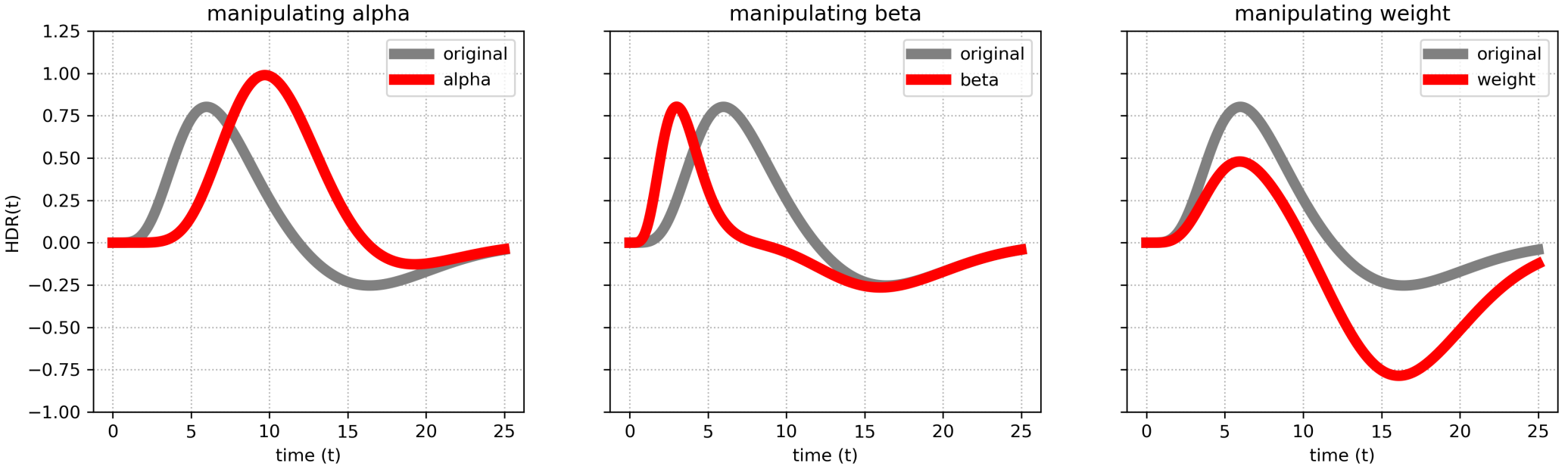
matplotlib styles

```
with plt.style.context('dark_background'):
    fig, ax = plt.subplots(1, 2, figsize=(11, 4))
    ax[0].hist(R.randn(1000))
    for i in range(3):
        ax[1].plot(R.rand(10))
        ax[1].legend(['a', 'b', 'c'], loc='lower left');
```



Homework 7

- manipulate parameters of HDR function
- reproduce this array of plots (subplots) as closely as possible (all the formatting) - you will need to look through the class slides and may need to look online for how to do some of the formatting pieces



3D plots

3D plots

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

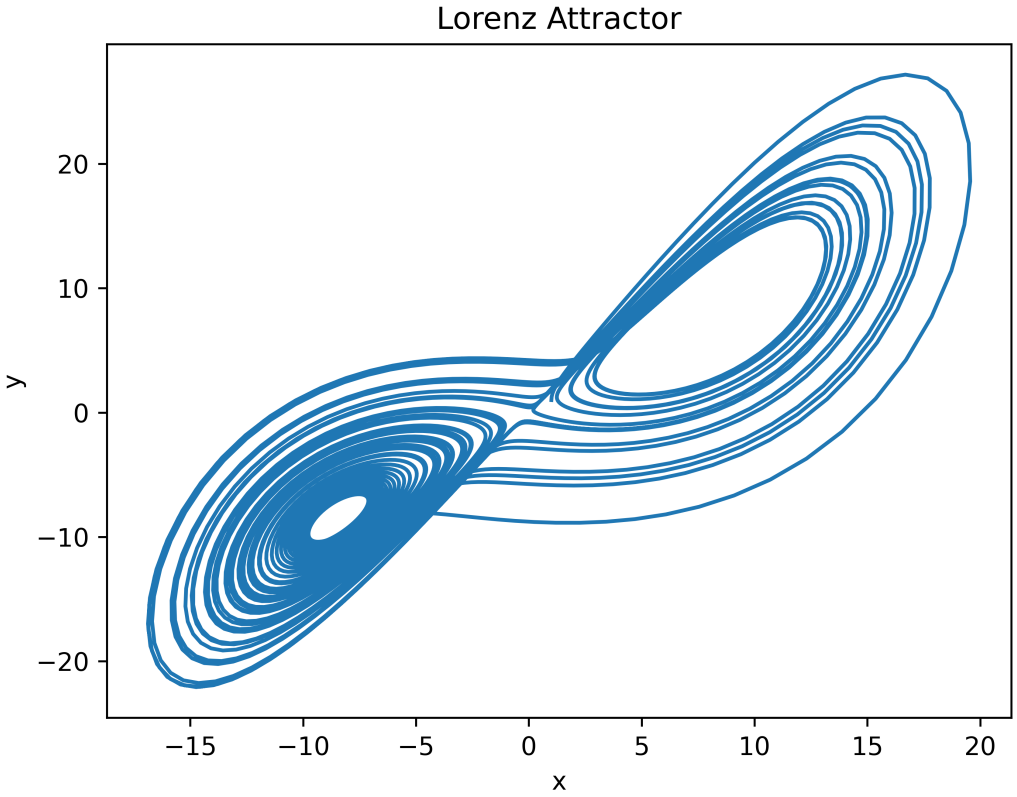
assume :

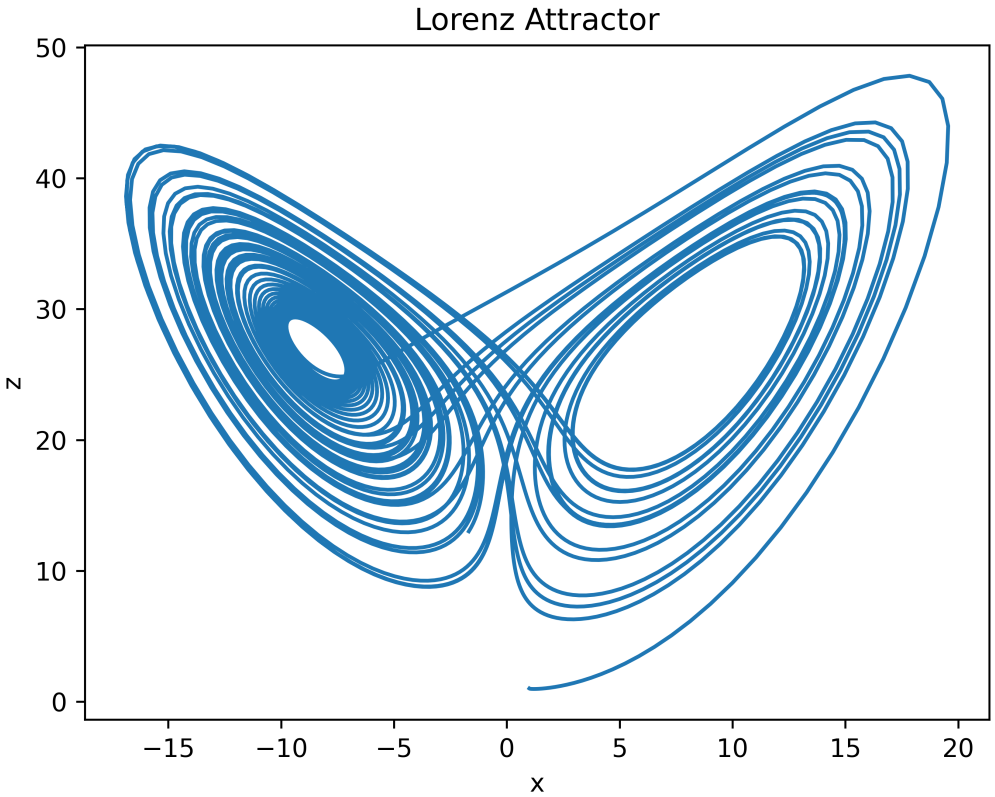
$\sigma = 10$

$\beta = 8/3$

$\rho = 28$

$x(0) = y(0) = z(0) = 1$





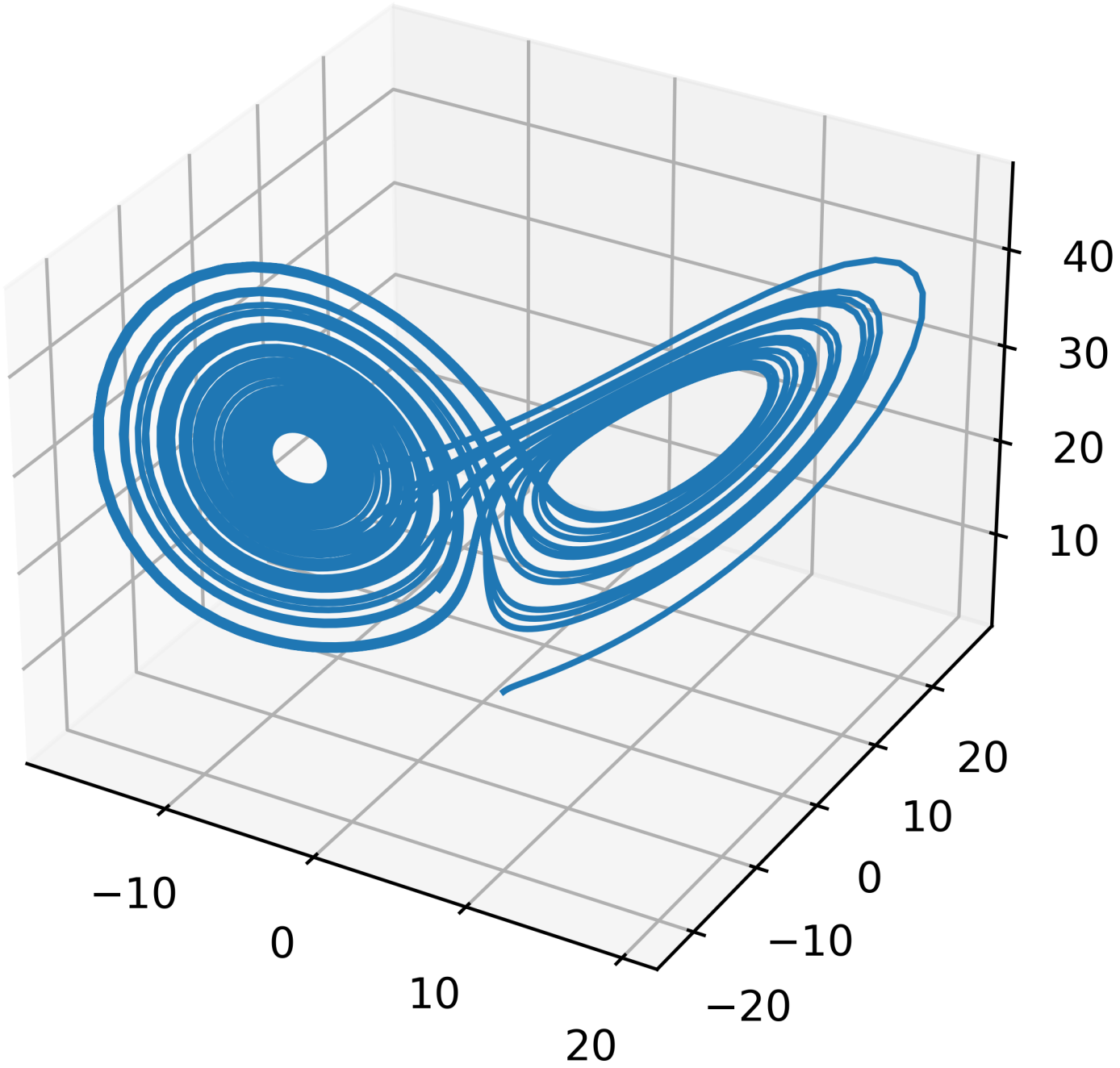
3D plots

```
from mpl_toolkits.mplot3d import Axes3D


fig = plt.figure()
ax = plt.axes(projection='3d')


x = x[:,0]
y = x[:,1]
z = x[:,2]
ax.plot(x, y, z)
```

https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html

https://matplotlib.org/stable/tutorials/toolkits/mplot3d.html

3D plots



3D plots

What is this?

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

3D plots

What do you think this is?

$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi \mid \Sigma \mid}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right)$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$$ **mean**

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$$ **covariance matrix**

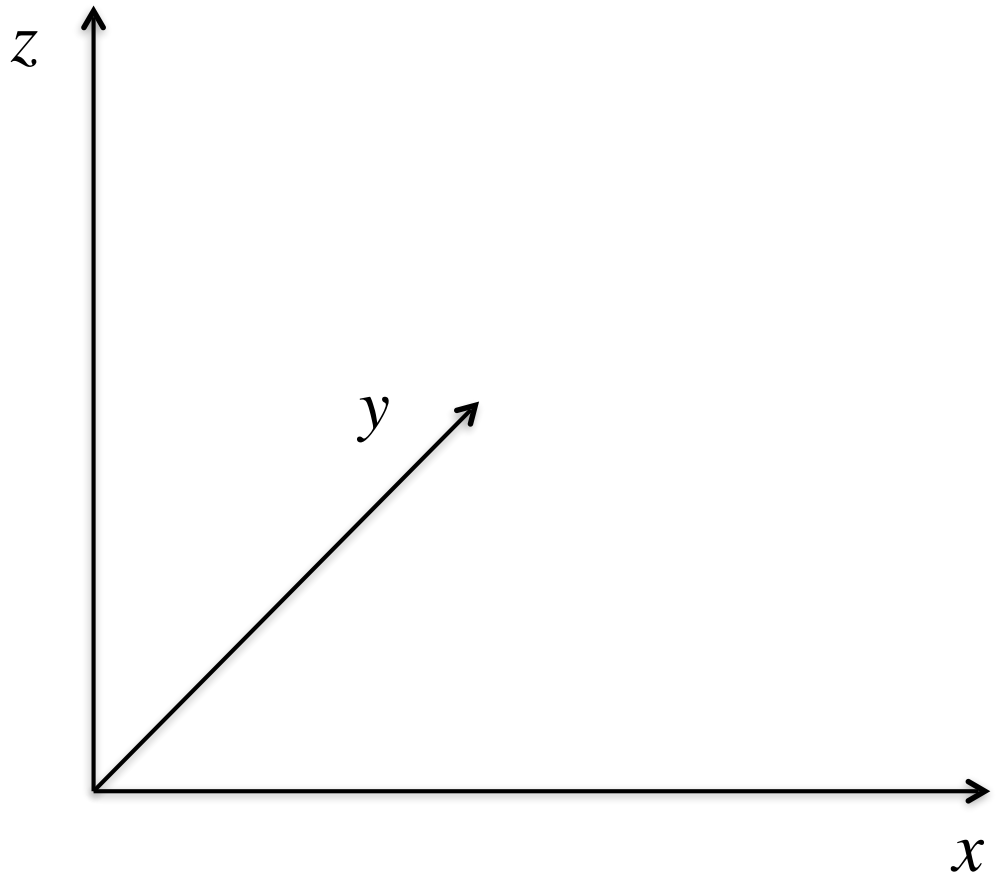
3D plots

multivariate normal distribution (pdf)

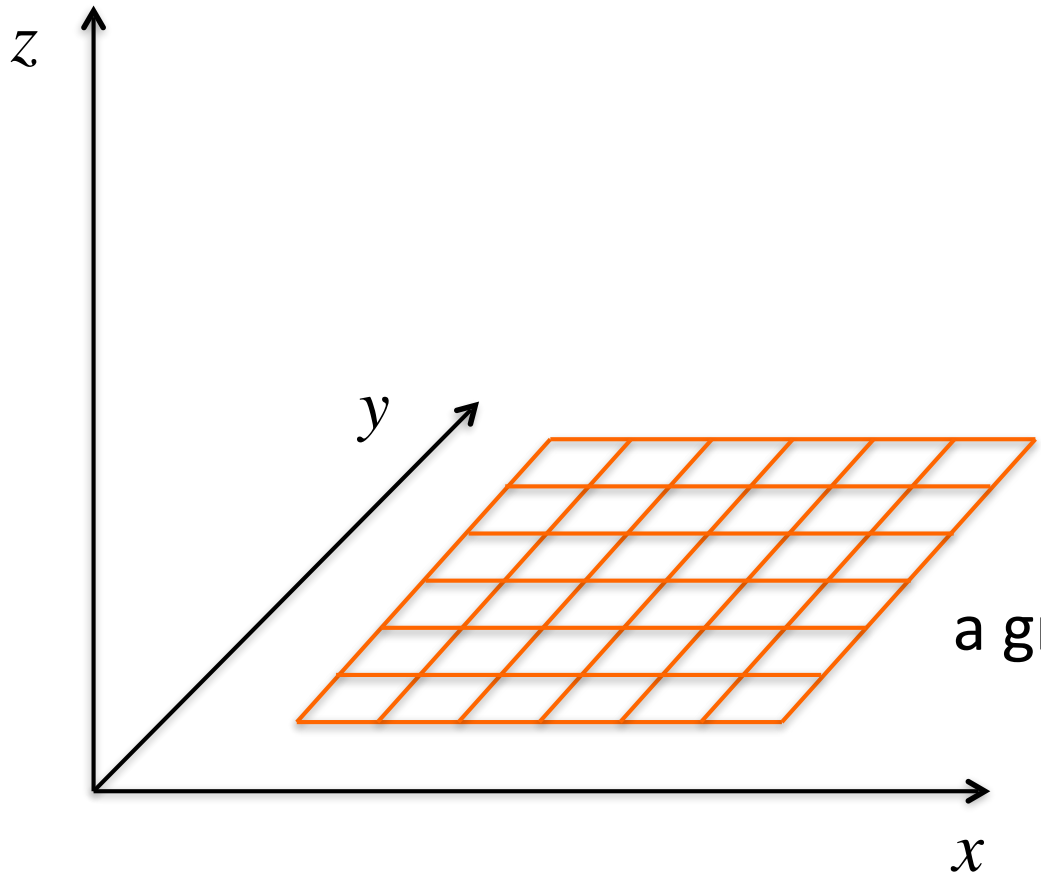
```
mx = 2; my = 1
sx = 1; sy = 2; r = -.7
mu = np.array([mx, my])
sig = np.array([[sx**2,   r*sx*sy],
                [r*sx*sy, sy**2]])

def my_mvnpdf(x, M, S):
    return (1 / np.sqrt((2 * np.pi *
              np.linalg.det(S)))) * np.exp(-(1/2) *
              (np.transpose(x-M) @ np.linalg.inv(S)
              @ (x-M)))
```

how can we plot this multivariate pdf?

$$z = f(x,y)$$

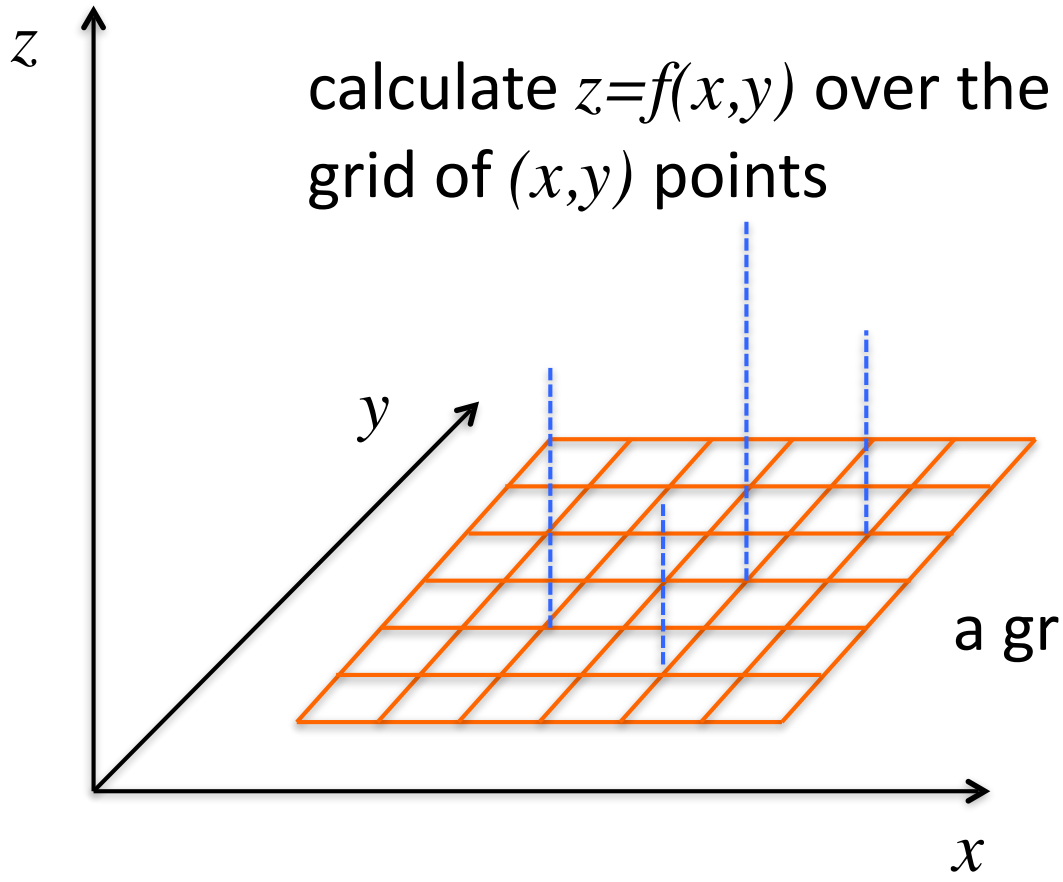


$$z = f(x,y)$$



a grid of (x,y) points

$$z = f(x,y)$$



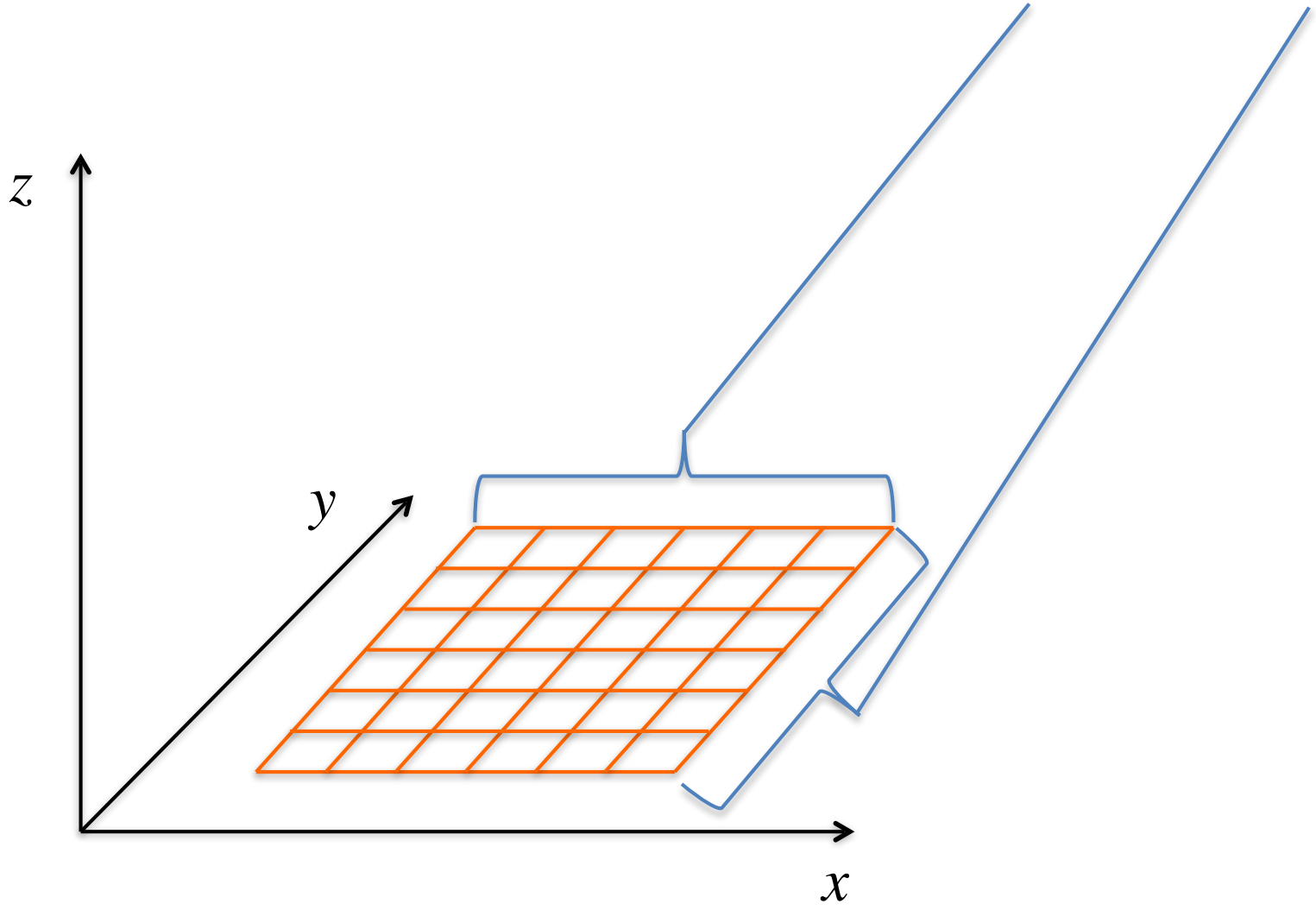
calculate $z=f(x,y)$ over the grid of (x,y) points

a grid of (x,y) points

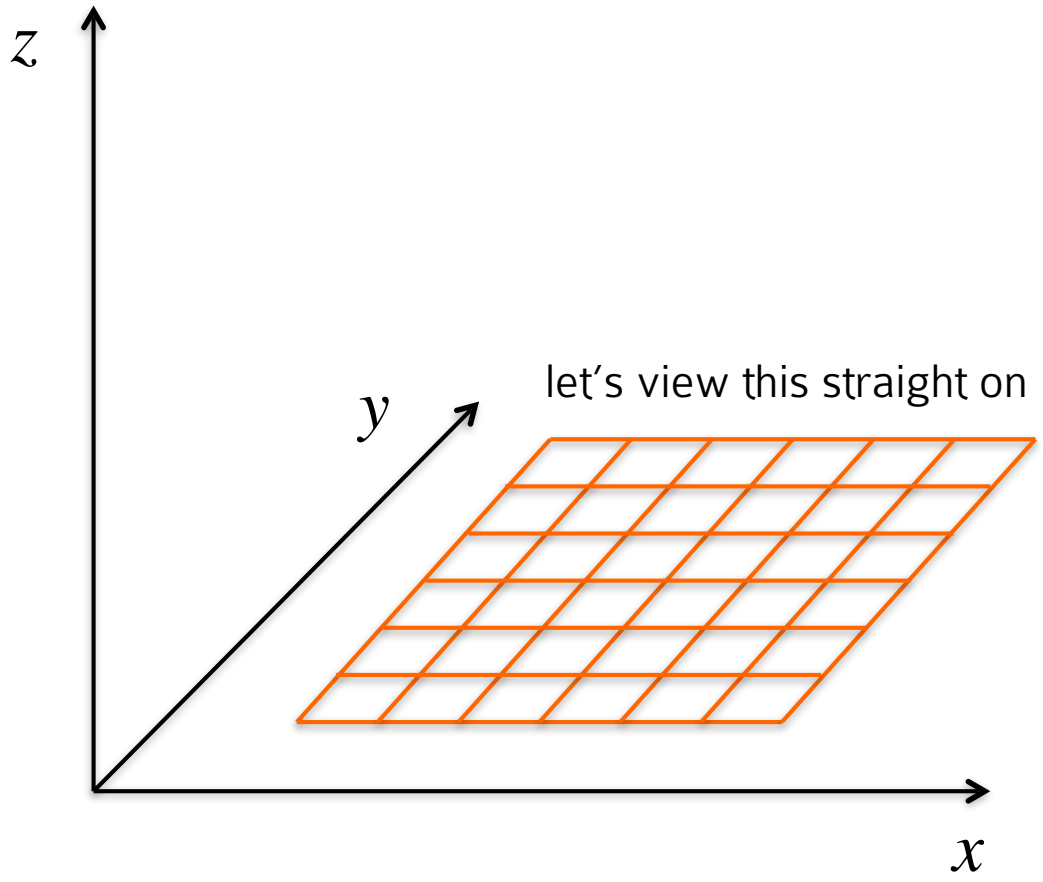
3D plots

```
inc = .25
xsteps = np.arange(-7, 7, inc)
ysteps = np.arange(-7, 7, inc)

X, Y = np.meshgrid(xsteps, ysteps)
```

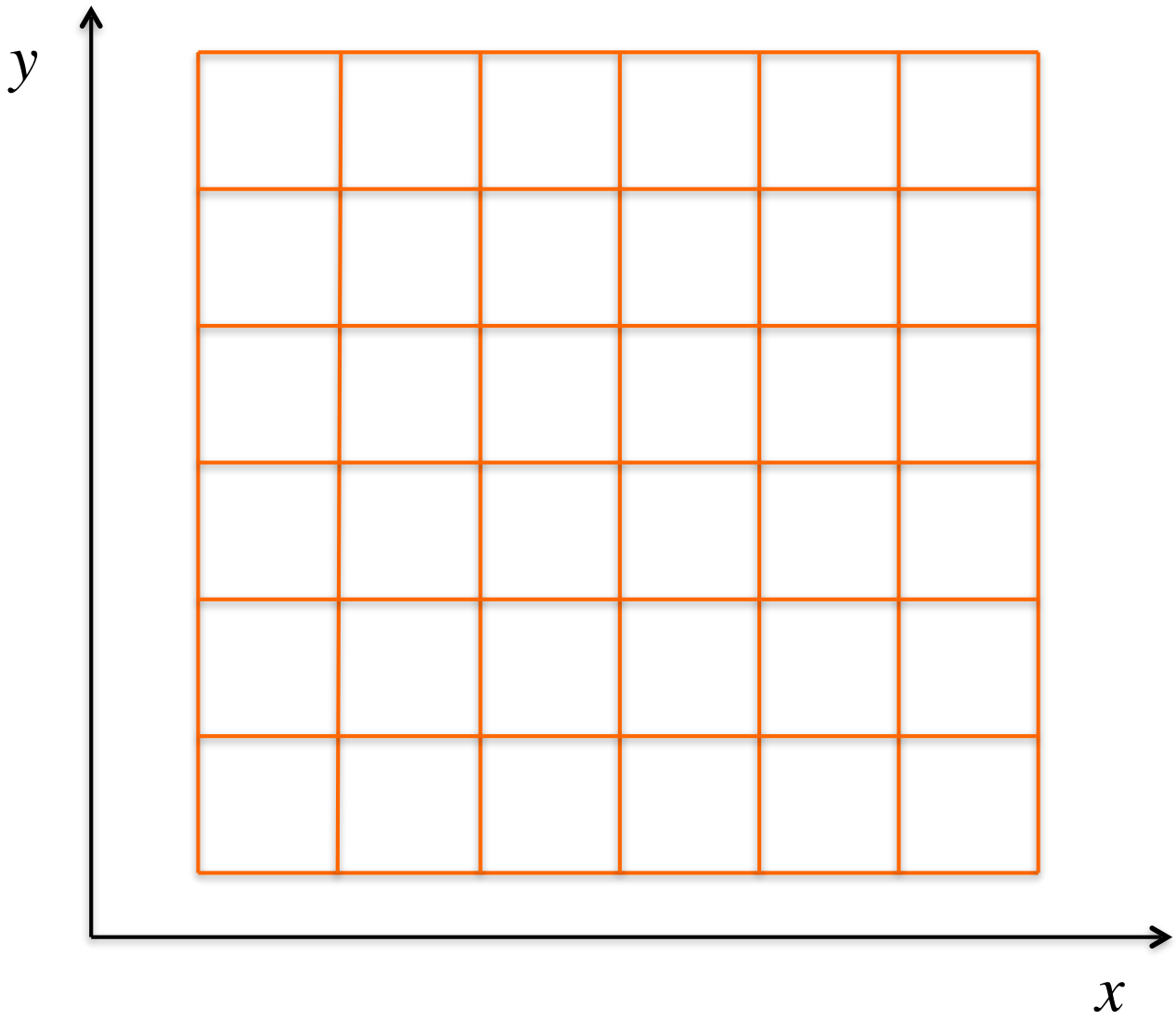
```
X, Y = np.meshgrid(xsteps, ysteps)
```



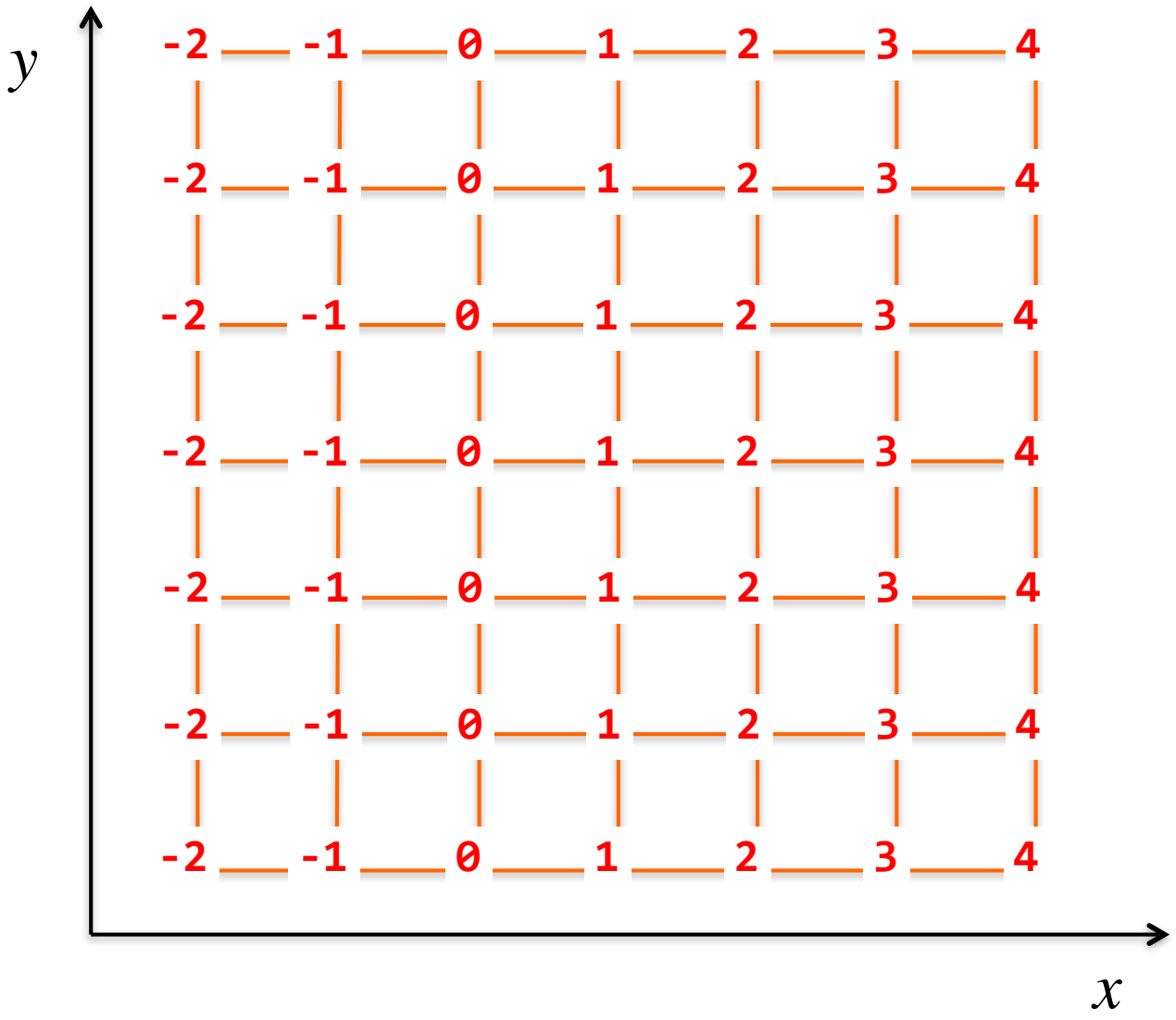
```
X, Y = np.meshgrid(xsteps, ysteps)
```



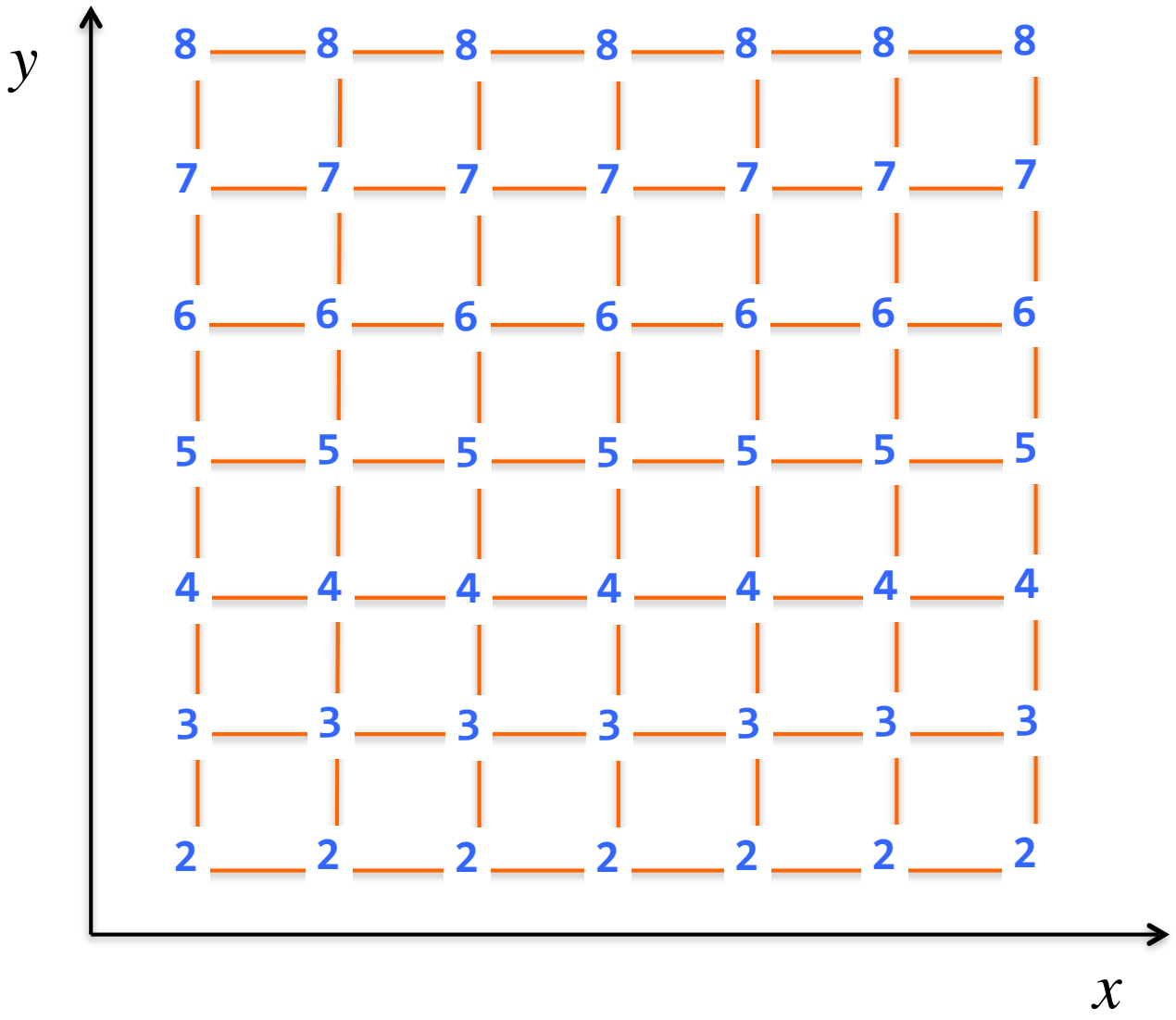
```
X, Y = np.meshgrid(xsteps, ysteps)
```



```
X, Y = np.meshgrid(xsteps, ysteps)
```



```
X, Y = np.meshgrid(xsteps, ysteps)
```



3D plots

```
X, Y = np.meshgrid(xsteps, ysteps)
```

How would we create Z, which has the same dimensions as X and Y?

3D plots

```
X, Y = np.meshgrid(xsteps, ysteps)
```

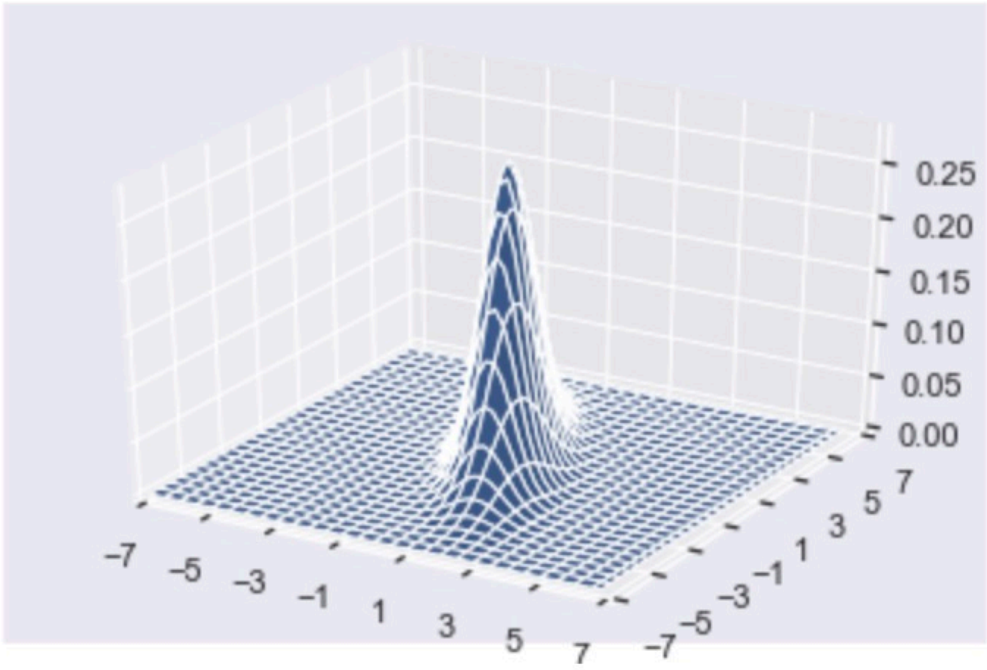
How would we create Z, which has the same dimensions as X and Y?

```
for i in range(len(xsteps)):
    for j in range(len(ysteps)):
        xy = np.array([X[i,j], Y[i,j]])
        Z[i,j] = my_mvnpdf(xy, mu, sig)
```

3D plots

surface plot

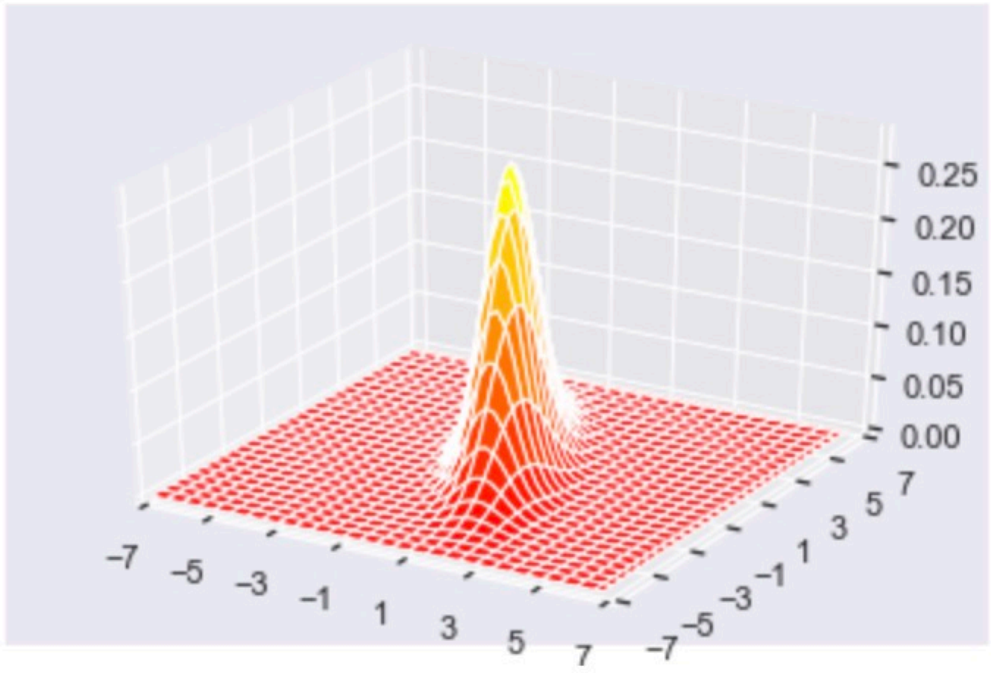
```
ax.plot_surface(X, Y, Z)
```

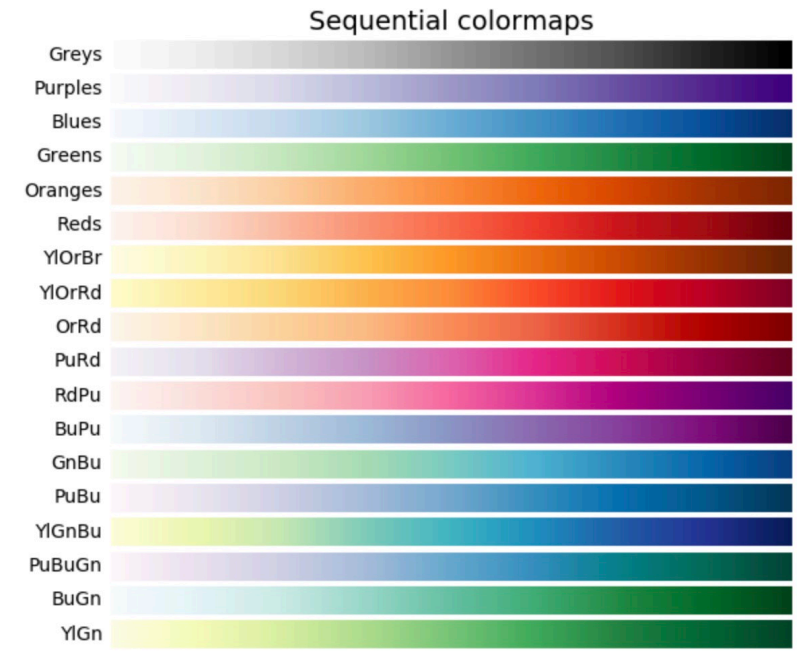


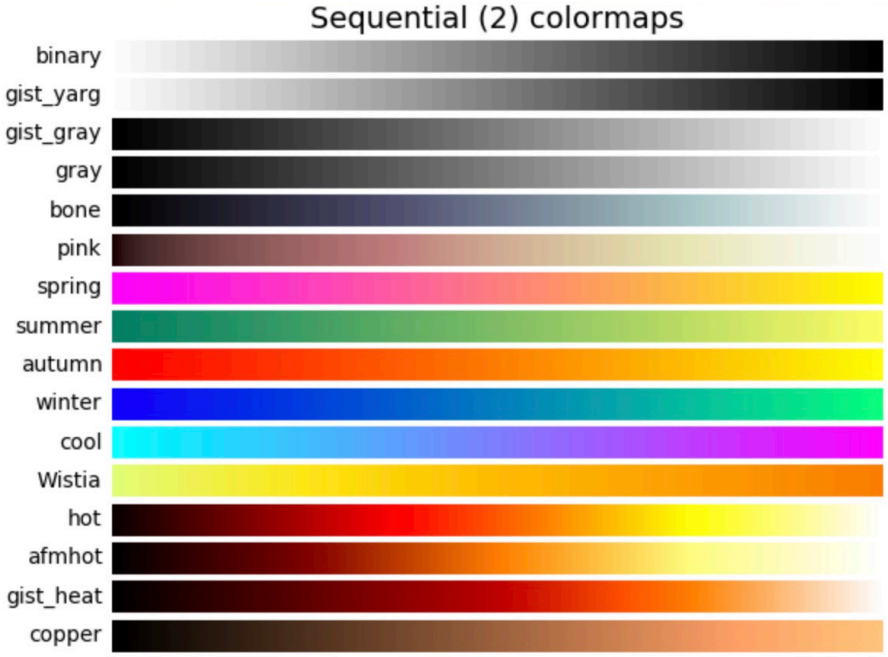
3D plots

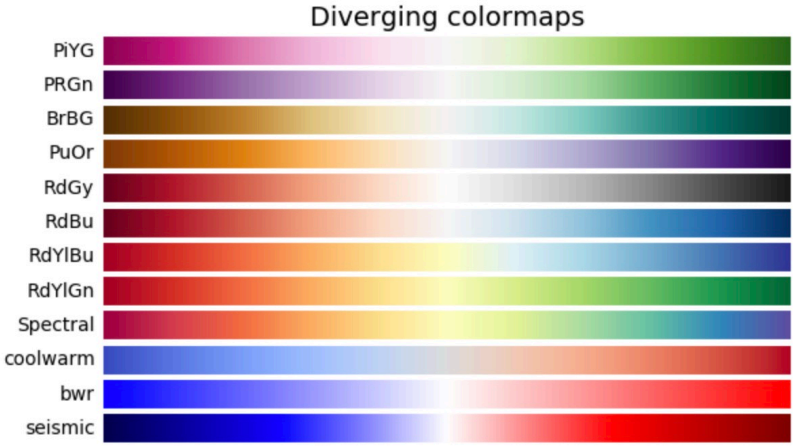
surface plot

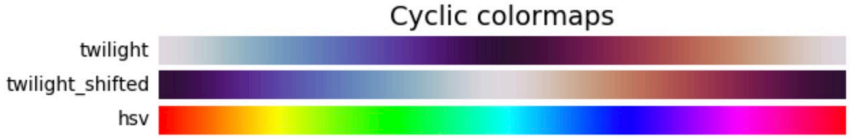
```
ax.plot_surface(X, Y, Z, cmap='autumn')
```









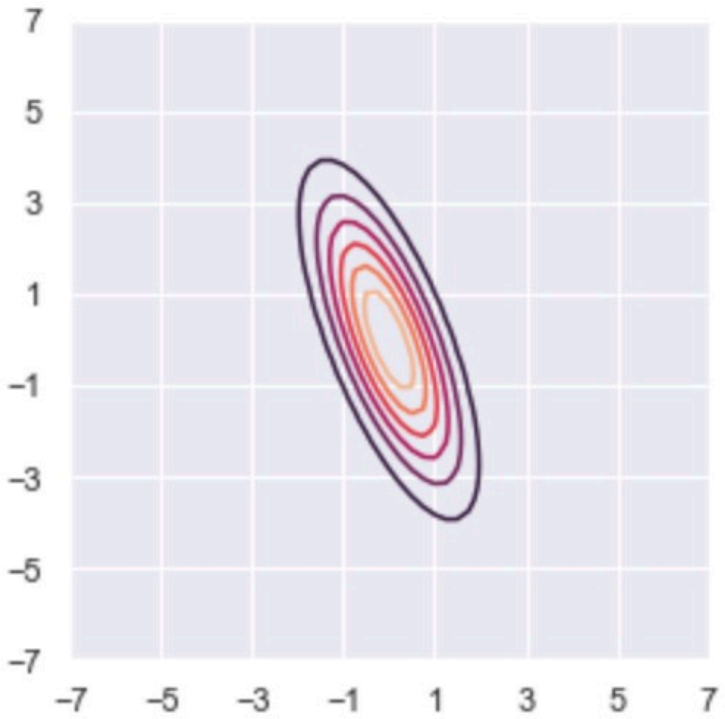


https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html

3D plots

contour plot

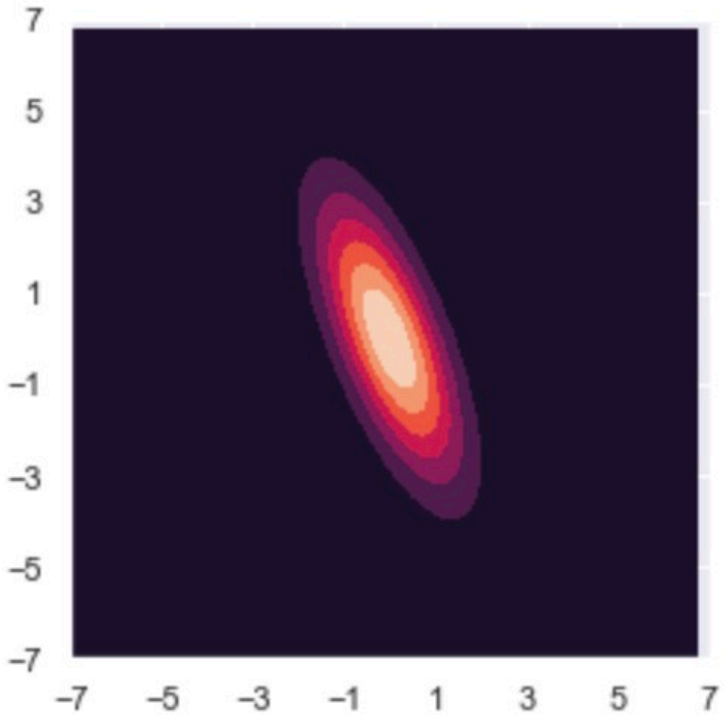
```
ax.contour(X, Y, Z)
```



3D plots

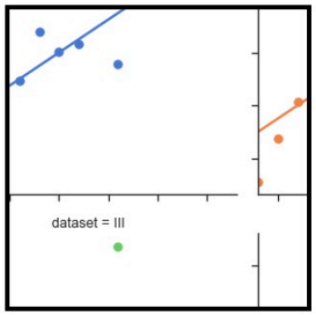
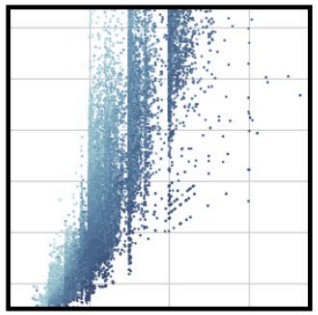
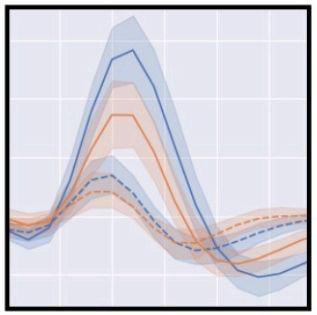
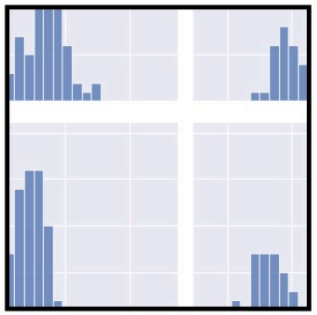
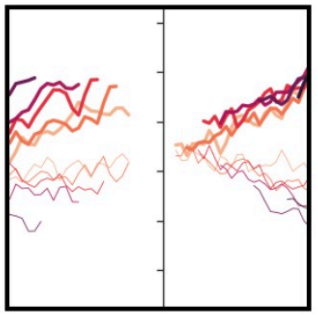
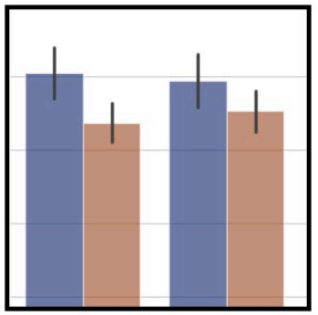
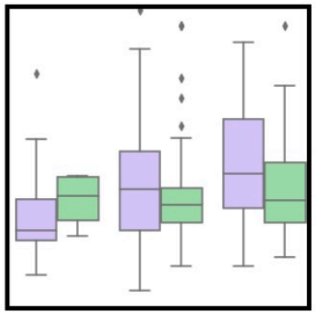
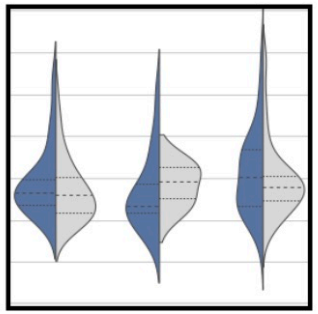
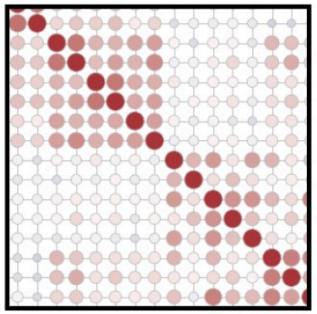
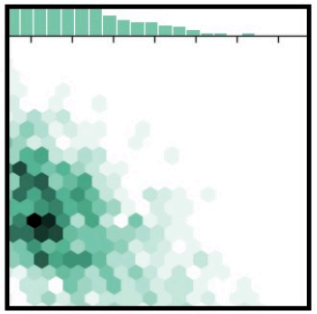
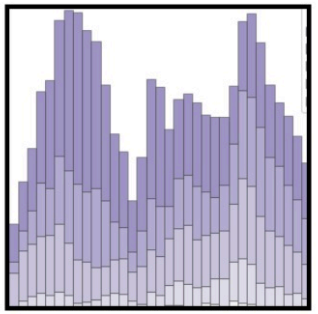
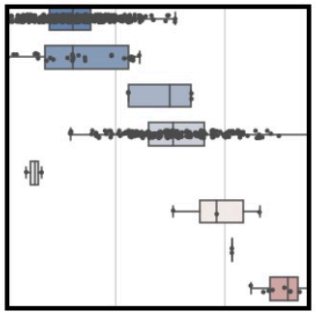
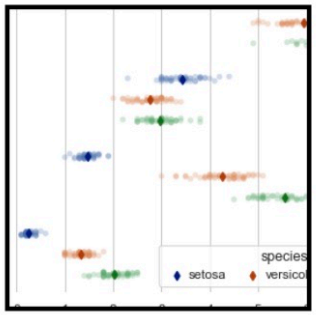
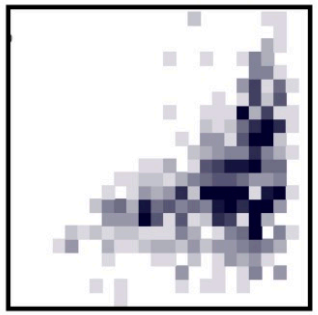
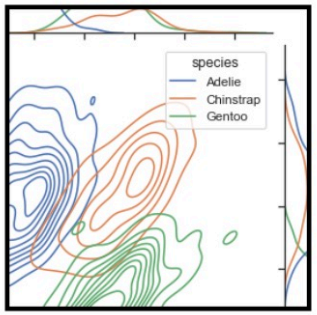
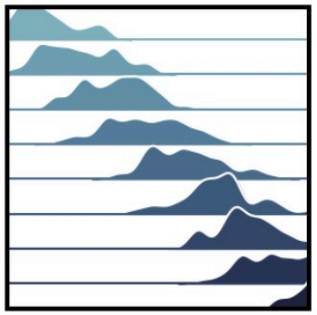
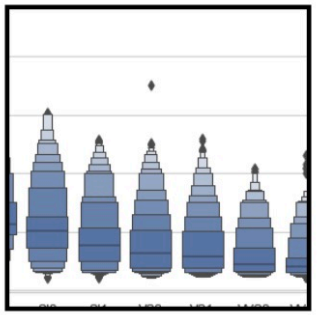
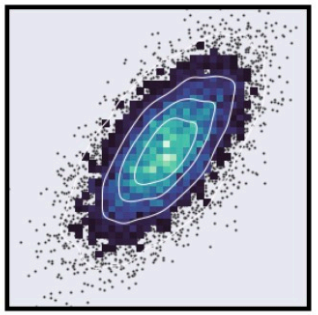
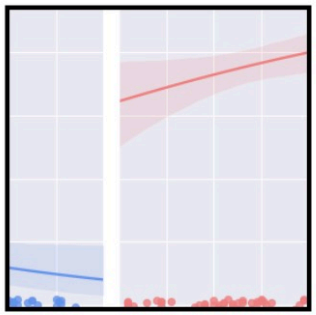
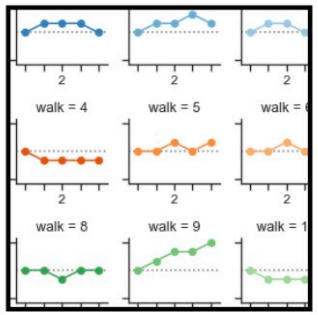
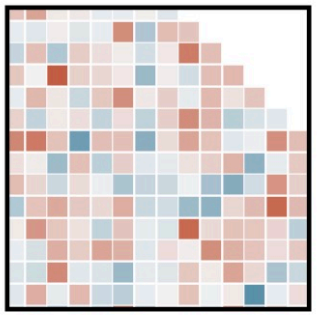
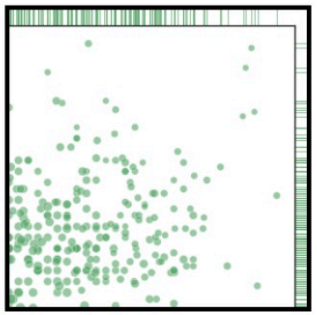
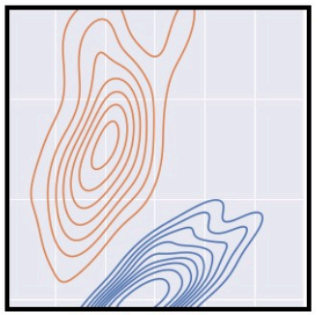
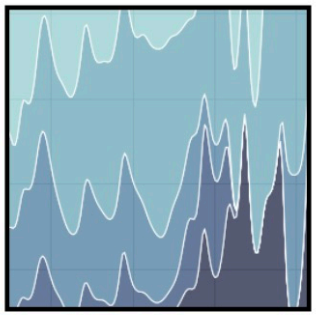
contour plot

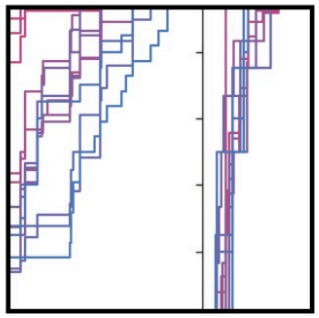
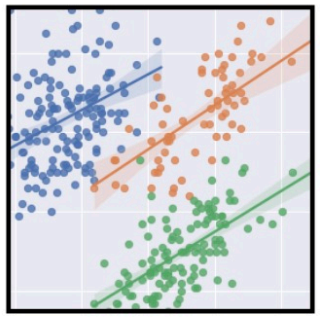
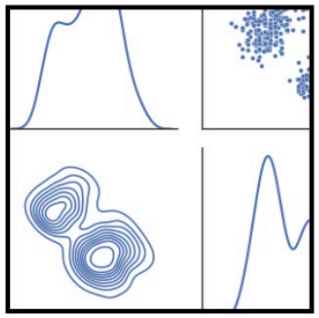
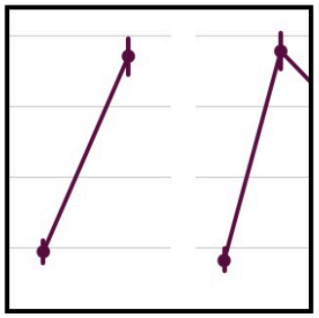
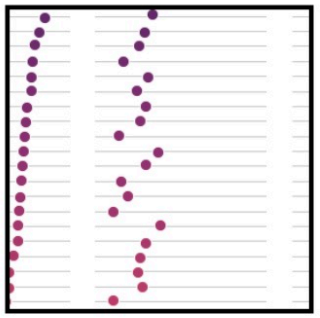
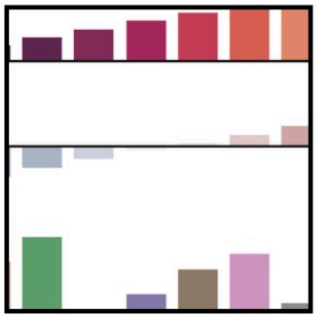
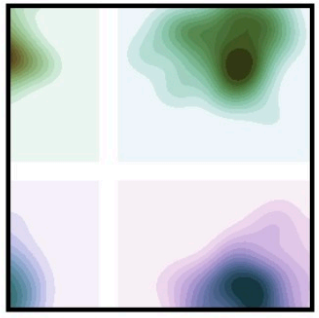
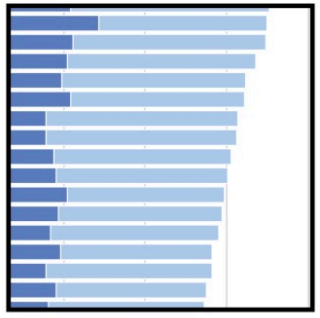
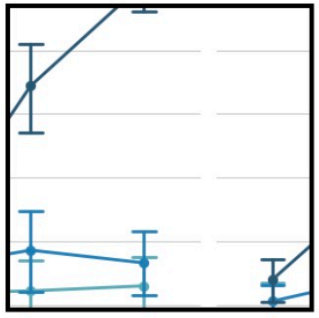
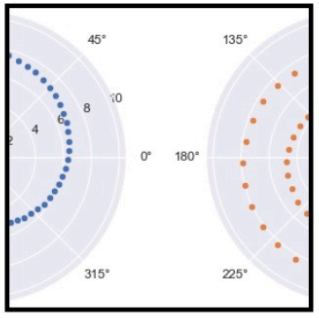
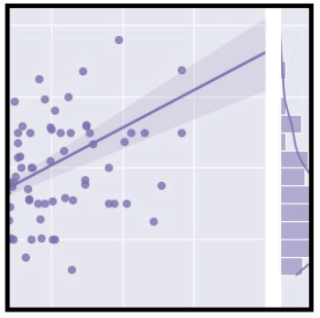
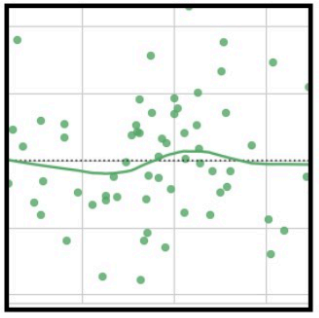
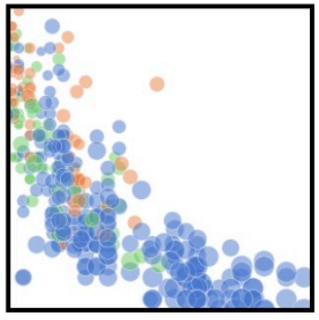
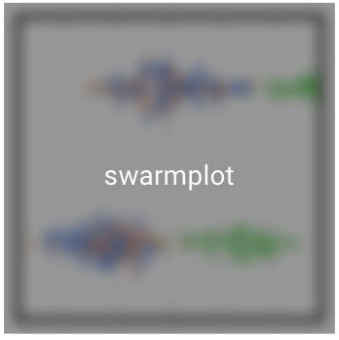
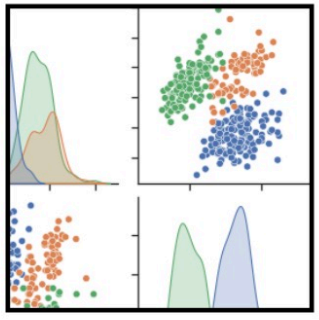
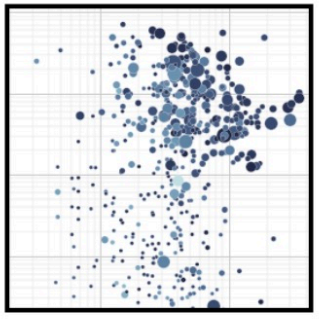
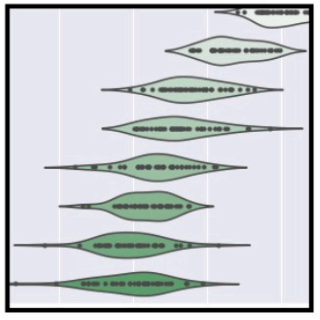
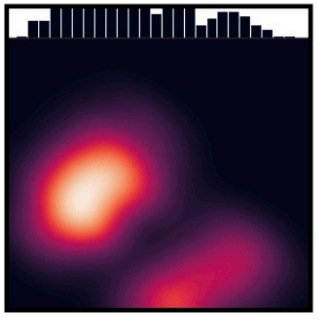
```
ax.contourf(X, Y, Z)
```

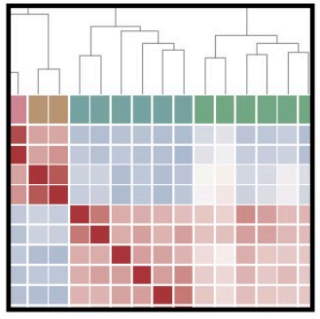
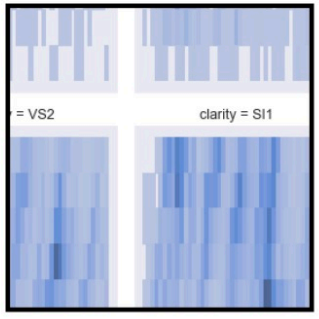
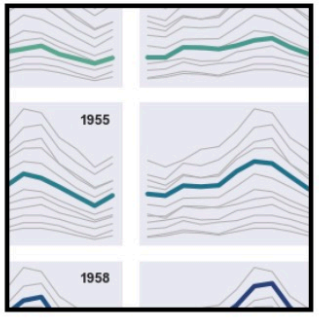
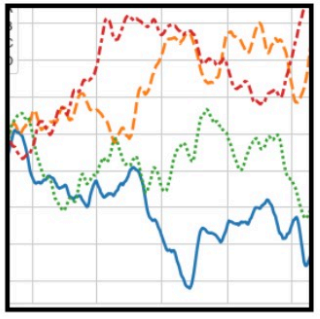
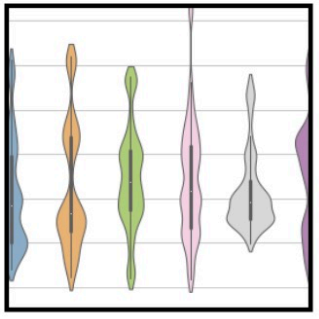


seaborn

https://seaborn.pydata.org


dataset = III












species
setosa


species
Adelie
Chinstrap
Gentoo





walk = 4
walk = 5
walk = 8
walk = 9















45°
135°
0°
180°
315°
225°




swarmplot





193 236 235 267 317 356 362
181 235 227 269 313 348 348
183 229 234 270 318 355 363
218 243 264 315 374 422 435
230 264 302 364 413 465 491
242 272 293 347 405 467 505
209 237 259 312 355 404 404
191 211 229 274 306 347 359
172 180 203 237 271 305 310


= VS2
clarity = SI1

1955
1958



Visualization.ipynb

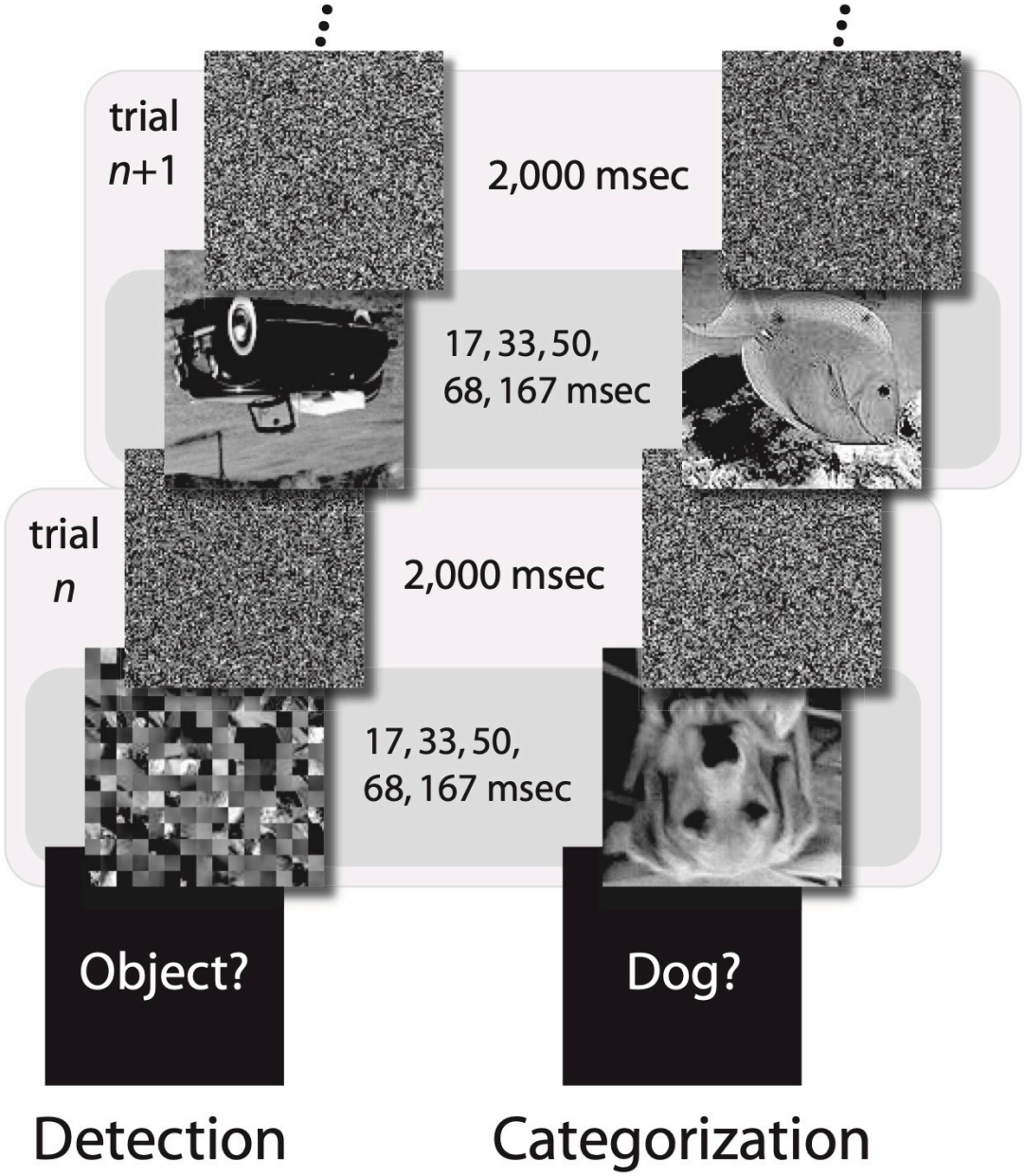
note that seaborn works with data in numpy arrays
or in pandas data frames (but most of the example
use pandas, which we may talk about later in the course)

Images and Signals

and some basics of image and signal processing

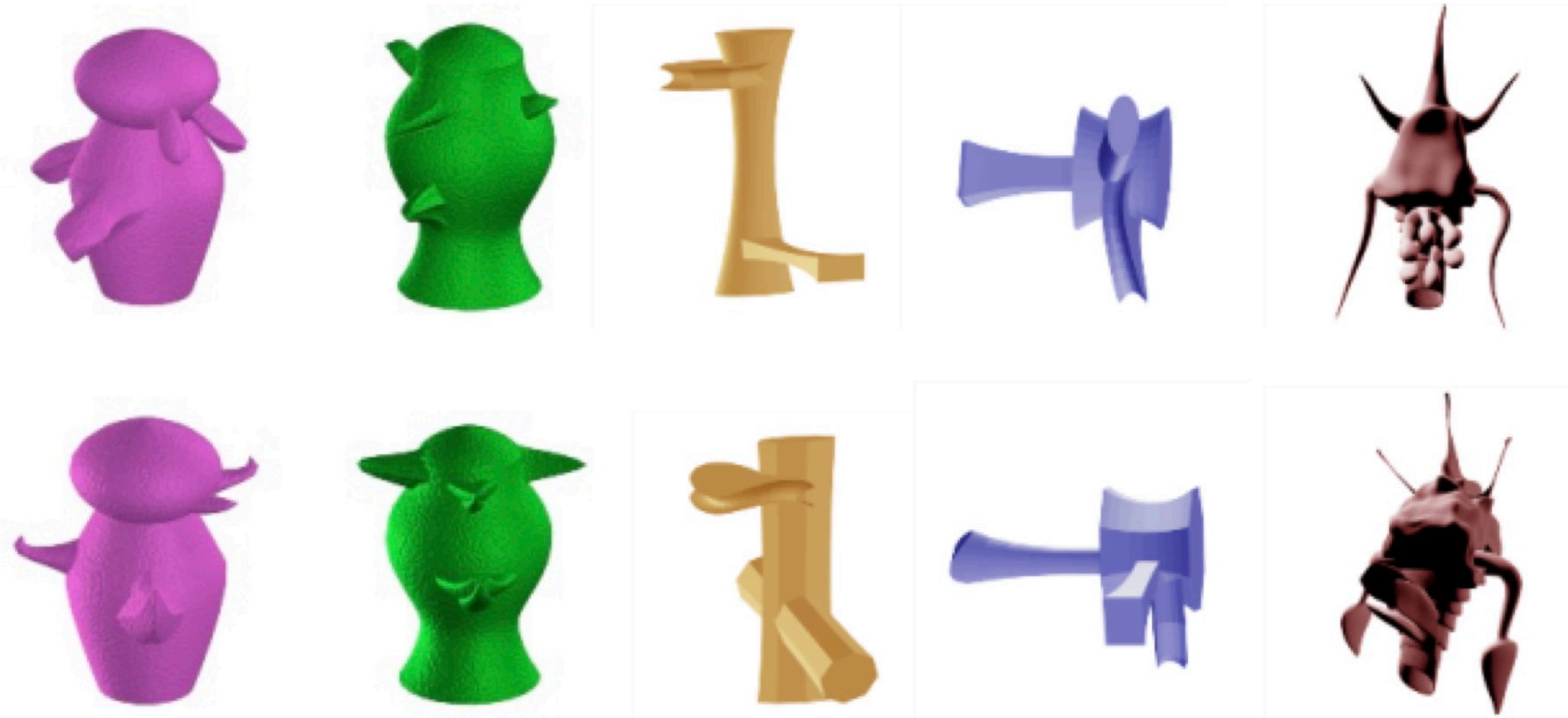
images in psychology and neuroscience

images as stimuli



images in psychology and neuroscience

images as stimuli

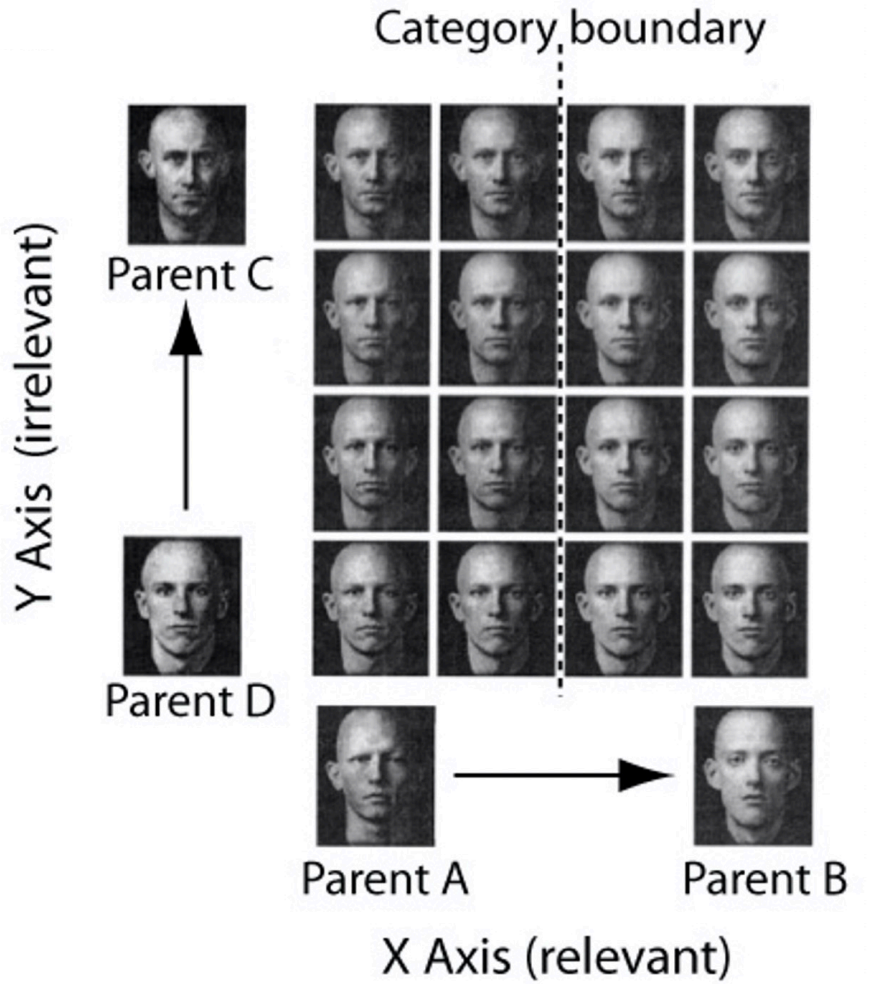


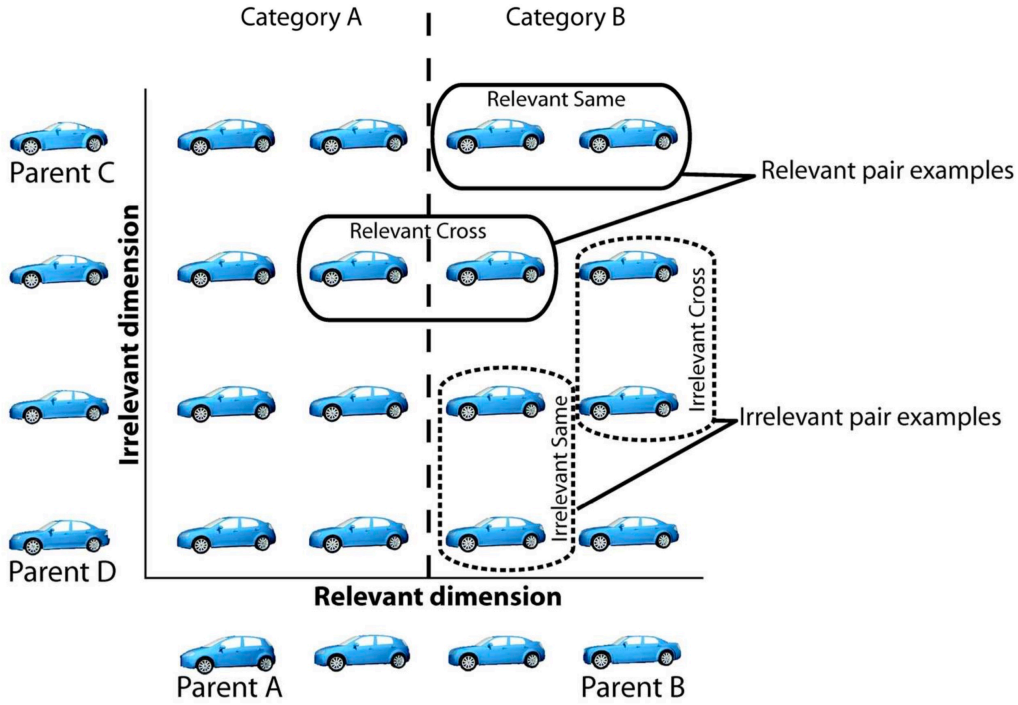
images in psychology and neuroscience

images as stimuli

images in psychology and neuroscience

images as stimuli



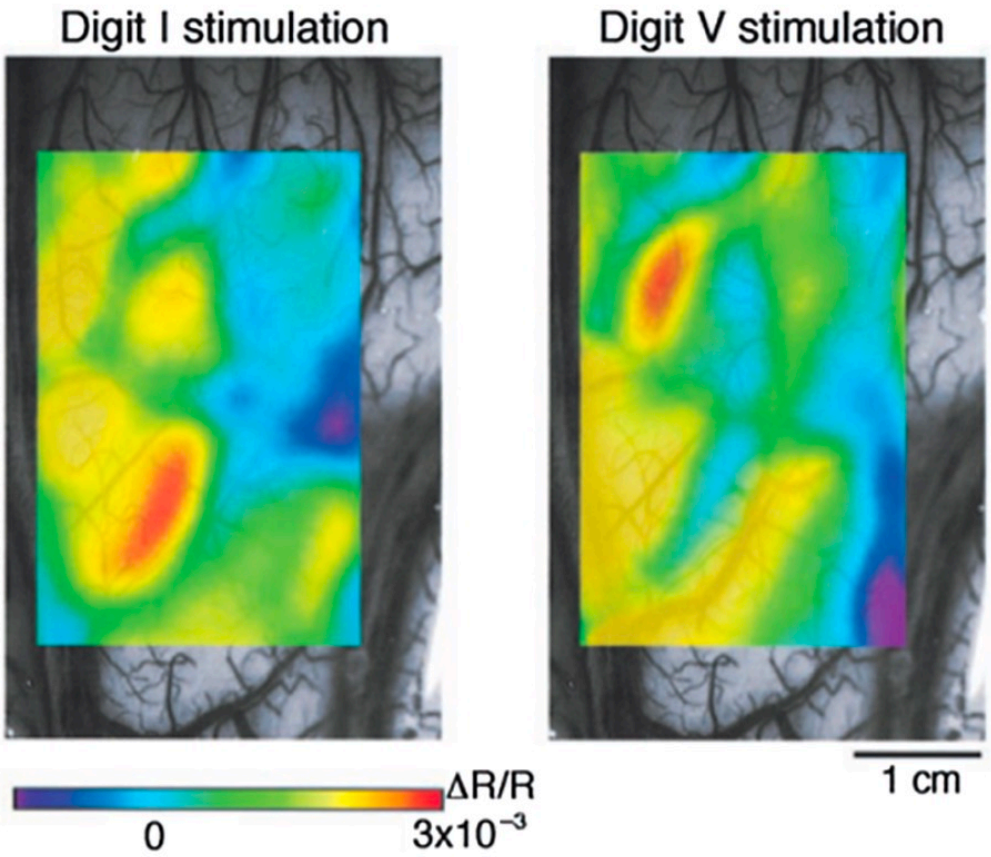


images in psychology and neuroscience

images as stimuli

images in psychology and neuroscience

images as data



images in psychology and neuroscience

images as data

images

matplotlib.image

- reads images into numpy arrays
- uses Pillow to load images

Pillow - an updated fork of PIL (Python Imaging Library)

https://pillow.readthedocs.io/en/stable/

https://python-pillow.org

images

```
import matplotlib.image as mpimg

im = mpimg.imread('Jordingray.bmp')
print(im.shape); print(type(im[0,0]))
```

512x512 uint8 (unsigned integer 8 bit)

some high-definition images can be 12 or 16 bit

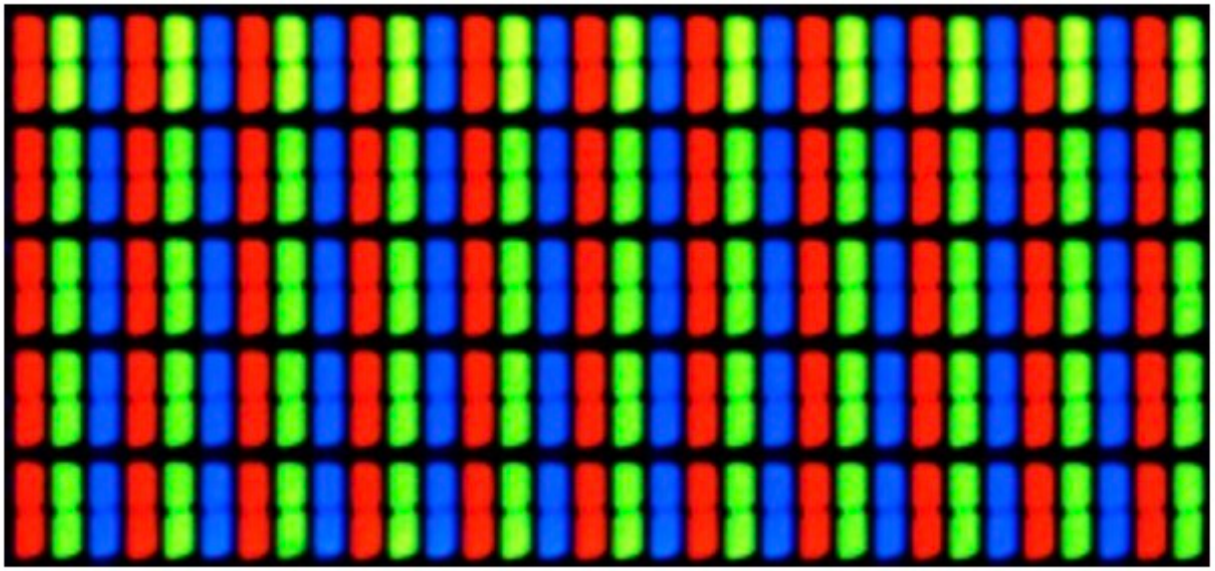
```
im = mpimg.imread('Jordin.bmp')
print(im.shape); print(type(im[0,0]))
```

512x512x3 uint8 (unsigned integer 8 bit)

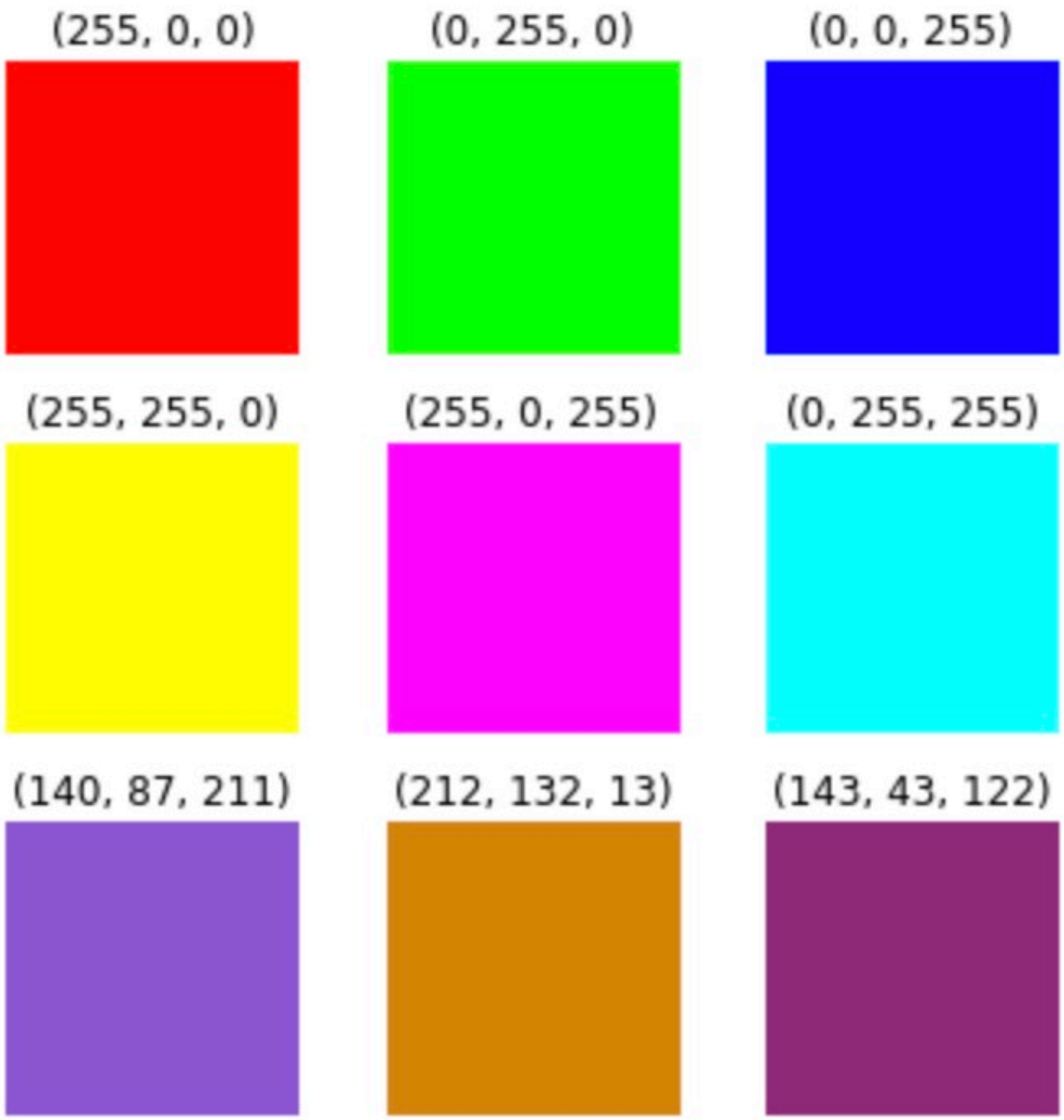
R, G, B

RGB

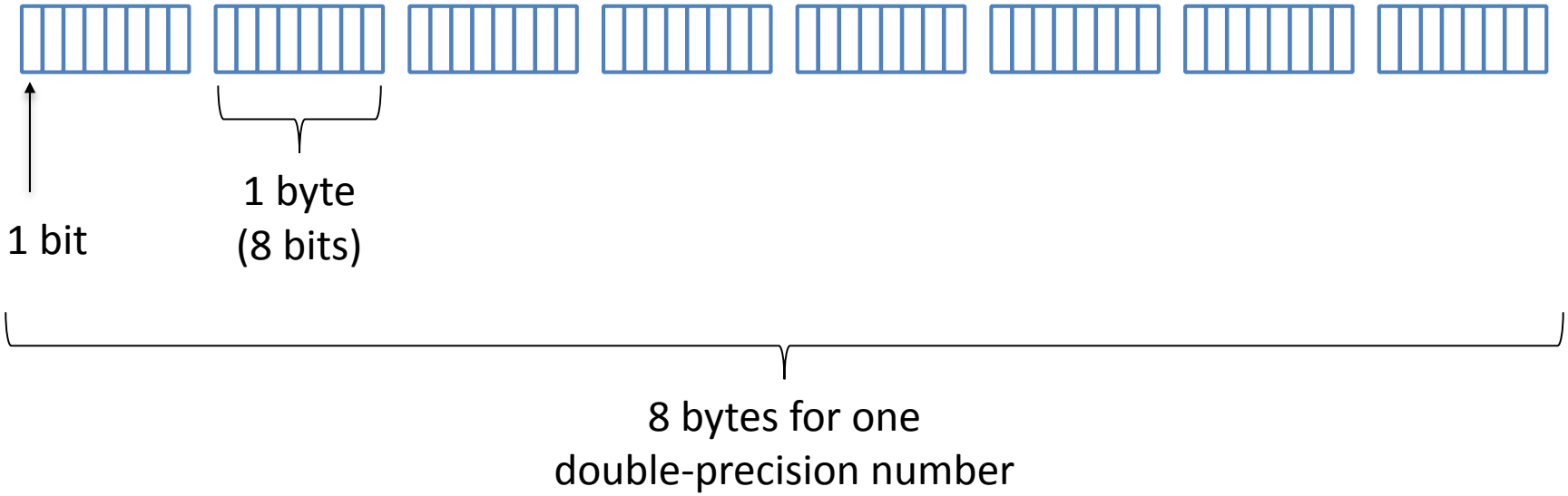
pixels on an LCD computer monitor



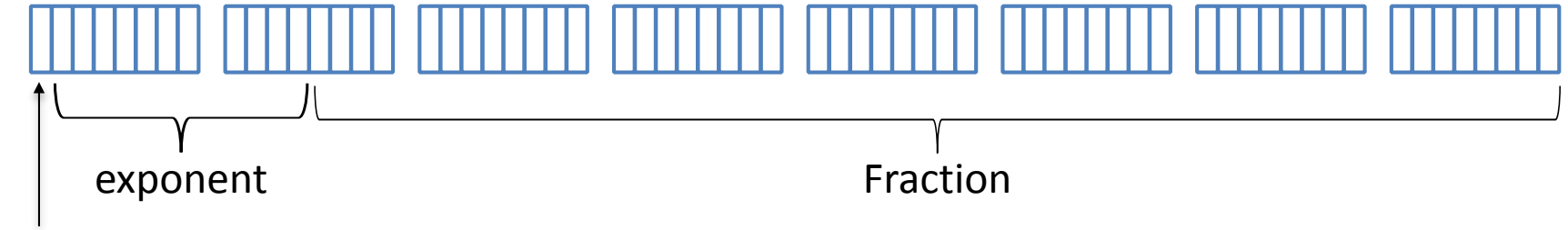
RGB


(255, 0, 0)
(0, 255, 0)
(0, 0, 255)
(255, 255, 0)
(255, 0, 255)
(0, 255, 255)
(140, 87, 211)
(212, 132, 13)
(143, 43, 122)

double-precision floating point

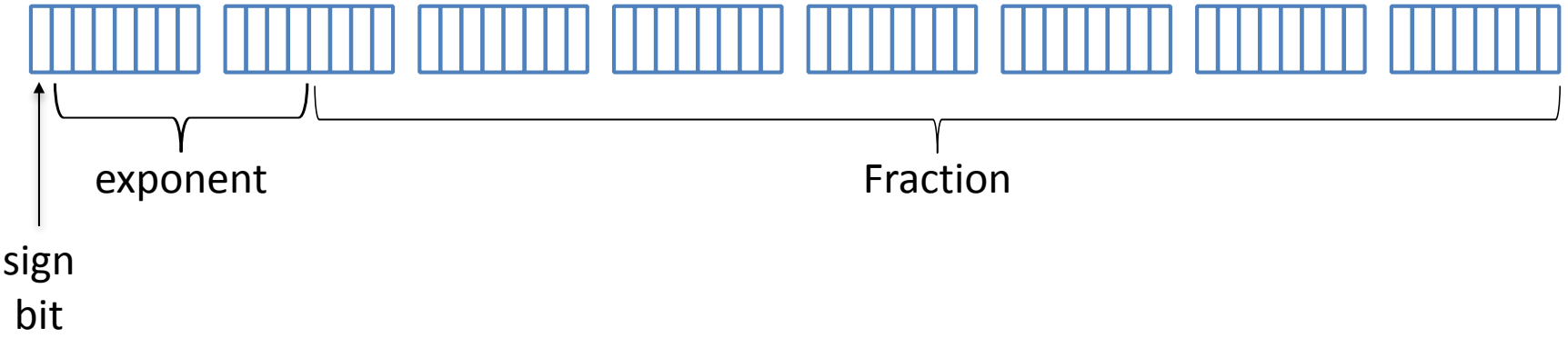


double-precision floating point

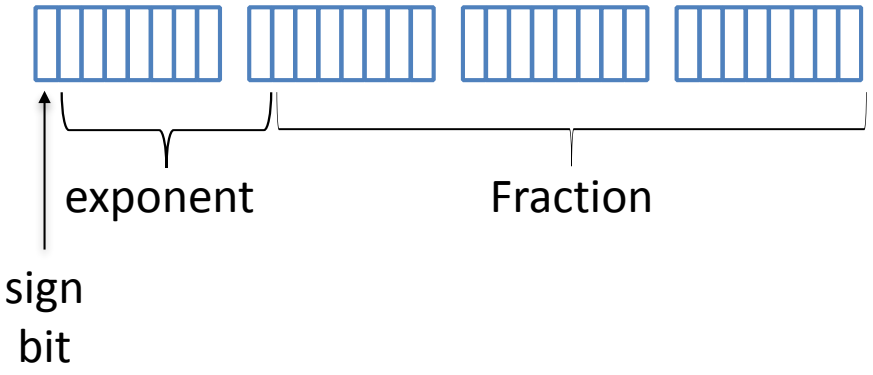


Bits	Usage
63	Sign (`0` = positive, `1` = negative)
62 to 52	Exponent, biased by `1023`
51 to `0`	Fraction `f` of the number `1.f`

double-precision floating point



single-precision floating point

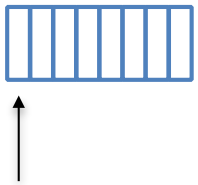


GPUs are often faster to compute single precision than double precision

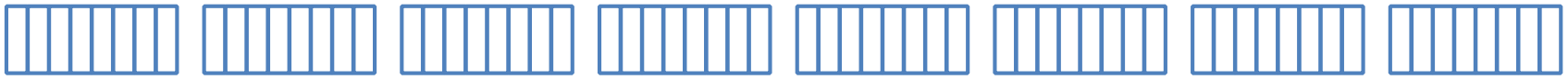
double-precision floating point

single-precision floating point

8-bit signed integer



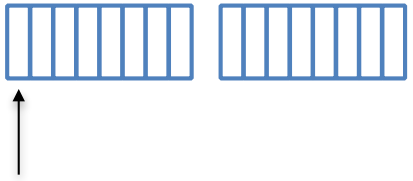
double-precision floating point



single-precision floating point

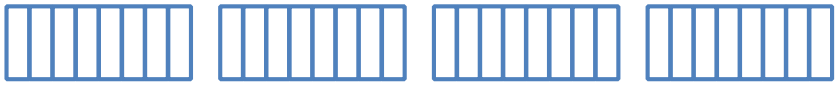
8-bit signed integer

16-bit signed integer



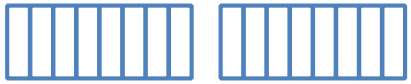
double-precision floating point

single-precision floating point



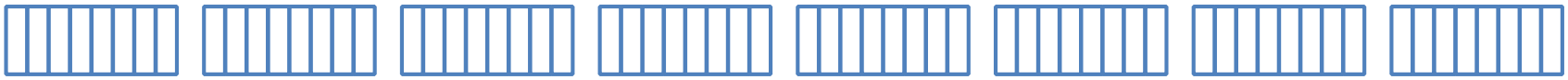
8-bit signed integer

16-bit signed integer



8-bit unsigned integer

64-bit unsigned integer



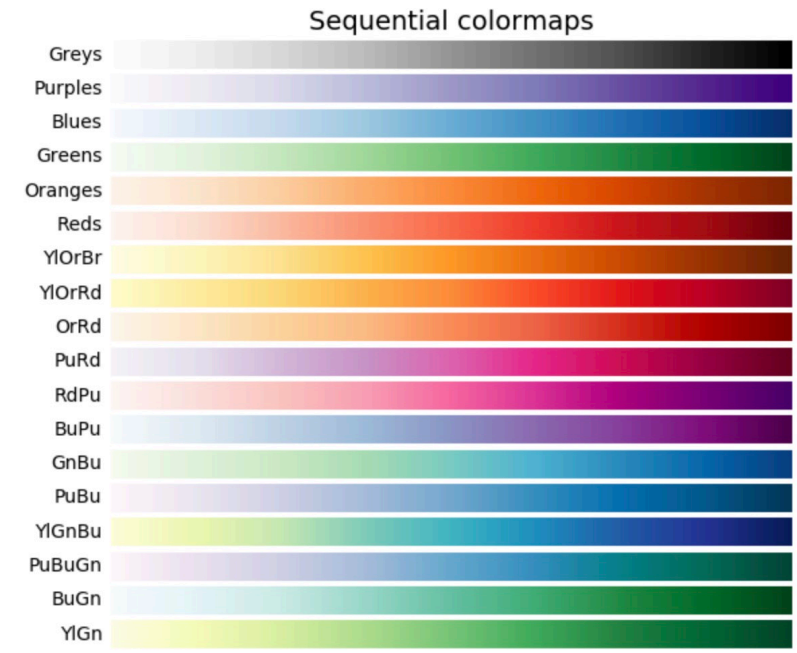
color maps

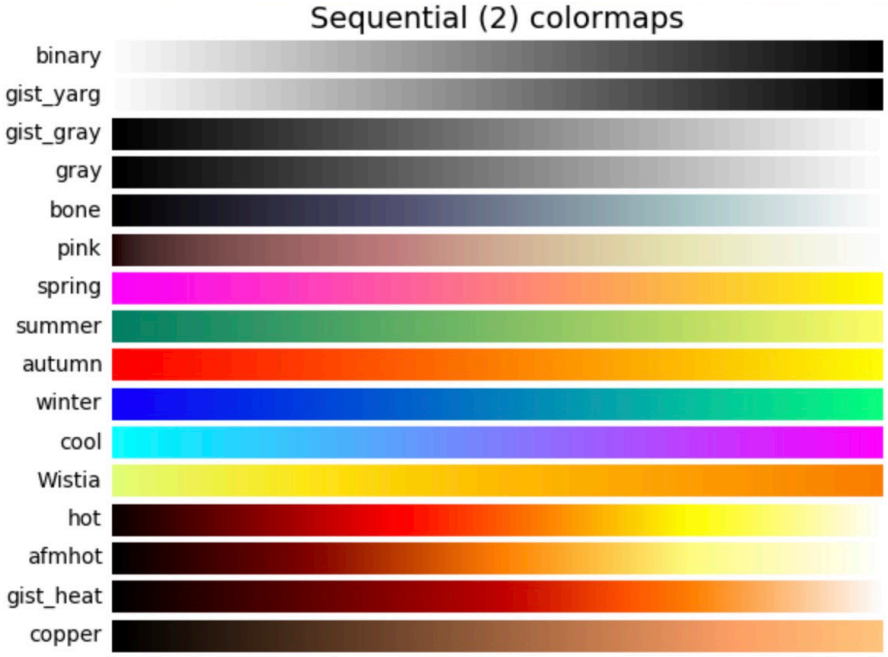
two-dimensional arrays ("grayscale") images can be colorized various ways using a color map

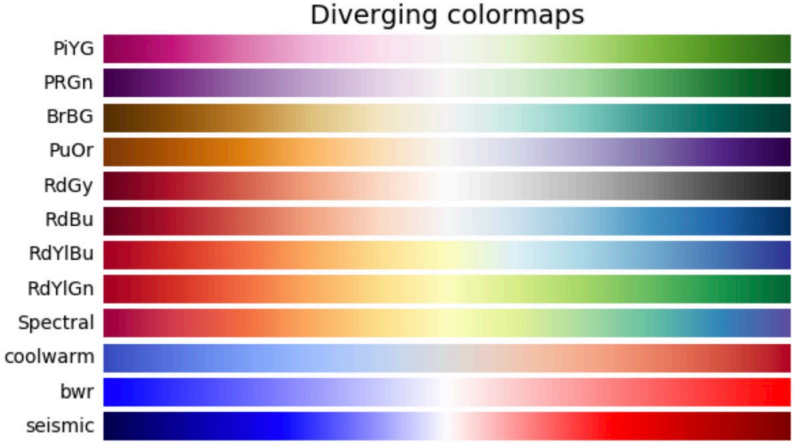
```
fig = plt.figure(); ax = plt.axes()
ax.imshow(im, cmap='copper')
ax.set_xticks([])
ax.set_yticks([]);
```

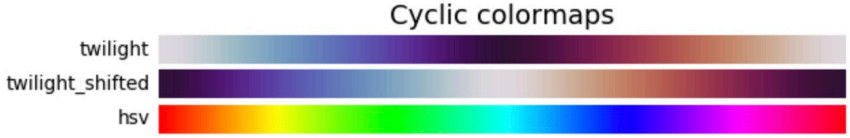
```
fig = plt.figure(); ax = plt.axes()
ax.imshow(im, cmap='twilight')
ax.set_xticks([])
ax.set_yticks([]);
```

does not change the image,
just how it is displayed









https://matplotlib.org/stable/tutorials/colors/colormaps.html

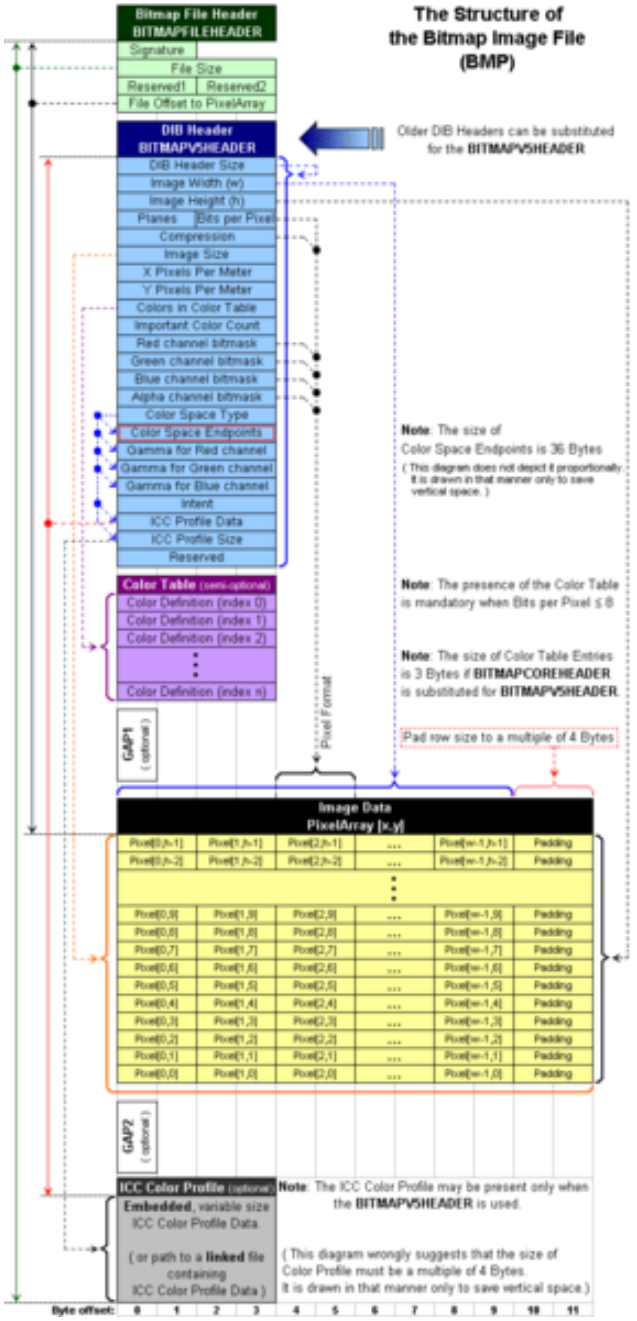
image formats

Images can be created and saved in various formats: jpeg, gif, tiff, bmp, png, etc. etc. etc.

Files in these formats have an internal structure that Python reads in order to reconstruct the actual image array.

The header specifies things like the image size, resolution, color map, and lots of other things.

headers are an example of "meta-data"
data about the data (in this case about image)



EXIF (Exchangeable Image File format) metadata


```
{'ExifVersion': b'0221',
 'ComponentsConfiguration': b'\x01\x02\x03\x00',
 'ShutterSpeedValue': 7.484742807323452,
 'DateTimeOriginal': '2018:03:18 09:39:15',
 'DateTimeDigitized': '2018:03:18 09:39:15',
 'ApertureValue': 2.2750072066878064,
 'BrightnessValue': 7.896821261073476,
 'ExposureBiasValue': 0.0,
 'MeteringMode': 5,
 'Flash': 16,
 'FocalLength': 2.87,
 'ColorSpace': 65535,
 'ExifImageWidth': 3088,
 'FocalLengthIn35mmFilm': 32,
 'SceneCaptureType': 0,
 'Make': 'Apple',
 'ExifImageHeight': 2320,
 'SubsecTimeOriginal': '367',
 'SubsecTimeDigitized': '367',
 'Model': 'iPhone 8 Plus',
 'Orientation': 1,
 'YCbCrPositioning': 1,
 'SensingMethod': 2,
 'ExposureTime': 0.00558659217877095,
 'XResolution': 72.0,
 'YResolution': 72.0,
 'FNumber': 2.2,
 'SceneType': b'\x01',
 'ExposureProgram': 2,
 'CustomRendered': 2,
 'ISOSpeedRatings': 20,
 'ResolutionUnit': 2,
 'ExposureMode': 0,
 'FlashPixVersion': b'0100',
 'WhiteBalance': 0,
 'Software': '11.2.6',
 'LensSpecification': (2.87, 2.87, 2.2, 2.2),
 'LensMake': 'Apple',
 'LensModel': 'iPhone 8 Plus front camera 2.87mm f/2.2',
 'DateTime': '2018:03:18 09:39:15',
 'ExifOffset': 198,
 'MakerNote': []}
```

image formats

One reason to care a bit about differences in image formats is that some formats lose information.

Uncompressed: full NxM image array is stored (many tiff and bmp files)

Lossless Compressed: image array is compressed, but when opened, the original is recovered perfectly (png and gif files)

Lossy Compressed: compressed and the original can only be recovered to some approximation (most jpg files)

Best Practices

NEVER use jpg images when those images are your data – use uncompressed or compressed lossless formats instead!

also if images come from a device yielding 16 bit resolution, you would never want to save them as 8 bit (compressed or uncompressed) JPG files

if the images are stimuli, like pictures to remember or identify, then it may not matter as much. Moreover, many image databases (and cameras) use jpg.