WWU
MÜNSTER

Python in GIS

matplotlib

living.knowledge

ifgi
Institut für Geoinformatik
Universität Münster

Learning goals:

After this lesson you should be able to do the following with matplotlib:

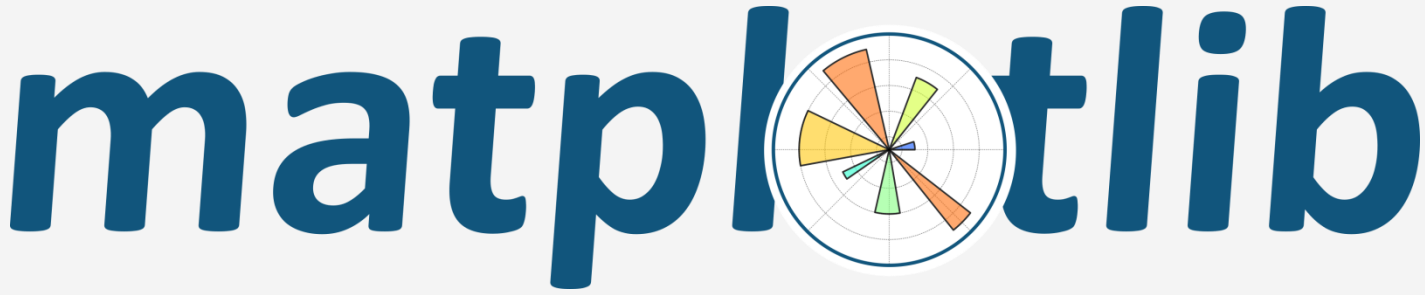
- Plot non-spatial data
- Show plots in interactive modus
- Create subplots
- Plot vector and raster data
- Build legends

Acknowledgements:

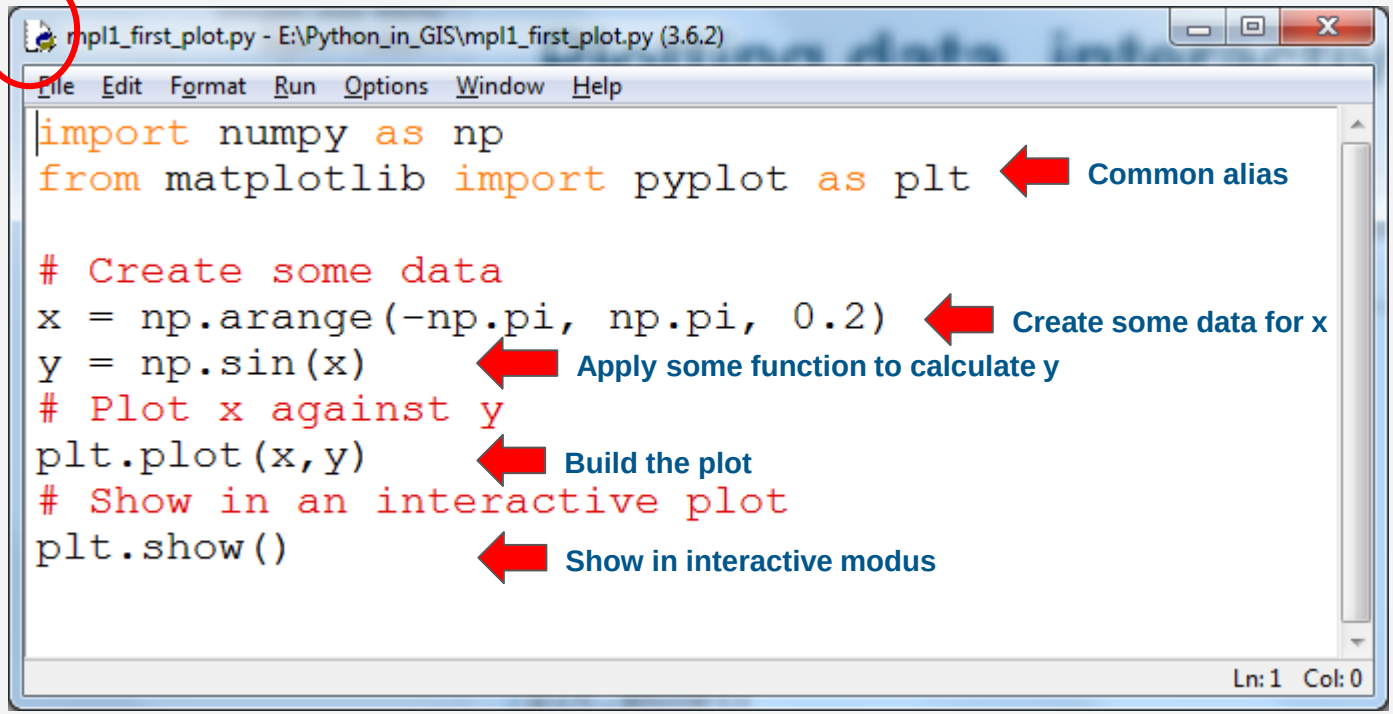
- Matplotlib documentation: http://matplotlib.org/
- Examples in the gallery: http://matplotlib.org/gallery.html

What is matplotlib?

- A Python plotting library
- Producing figures in a variety of image file formats
- Can be used in Python scripts, the IPython shell, and jupyter notebook
- Already installed in QGIS python and ArcGIS python



Plotting data, interactive visualization (1)



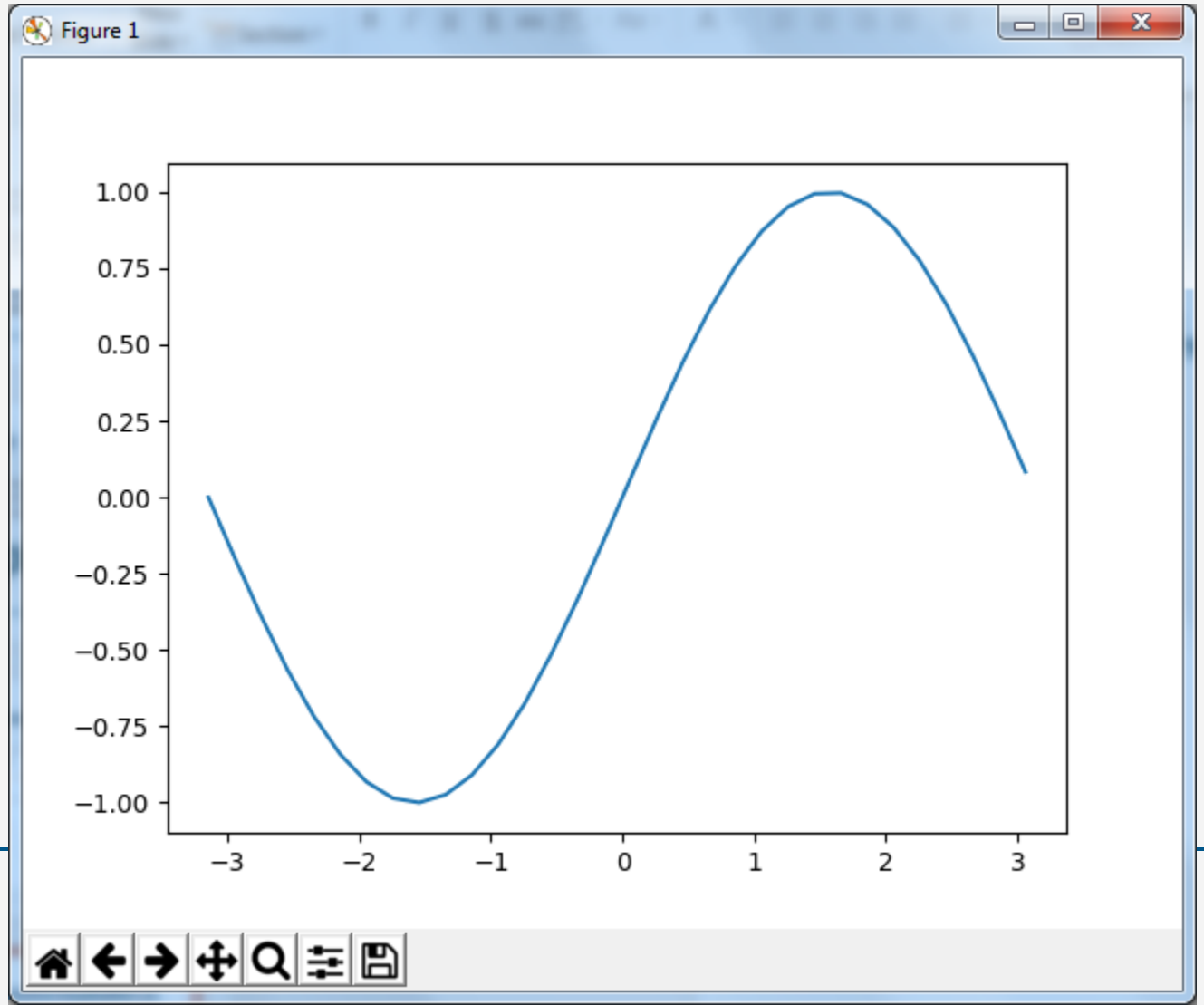
WWU MÜNSTER

Plotting data, interactive visualization (2)

Interactive plot opens

Works also inside QGIS

and ArcGIS



Plotting within QGIS, ISSUE

QGIS also comes with matplotlib, but some **plugins** block tkinter, a library needed by matplotlib.

Solution from https://github.com/gem/oq-irmt-qgis/issues/224:

1. uncheck the plugin(s) in the plugin manager
2. restart QGIS

Multiple plots in one figure (1)

mpl2_subplots.py - E:\Python_in_GIS\mpl2_subplots.py (3.6.2)

File Edit Format Run Options Window Help

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

# Create some data
x1 = np.arange(-np.pi, np.pi, 0.2)
y1 = np.sin(x1)
x2 = np.arange(-np.pi, 0, 0.2)
y2 = np.cos(x2)

# Initiate a figure
# A figure with two subplots
f, axarr = plt.subplots(2)
# subplots are stored in an array
line = axarr[0].plot(x1, y1)
axarr[0].set_title('plot 1')
axarr[1].plot(x2, y2)
axarr[1].set_title('plot 2')
plt.show()
```

Ln: 12 Col: 25

Two series, second is shorter

Figure instance and axis instance(s), i.e. subplots

title of the plot

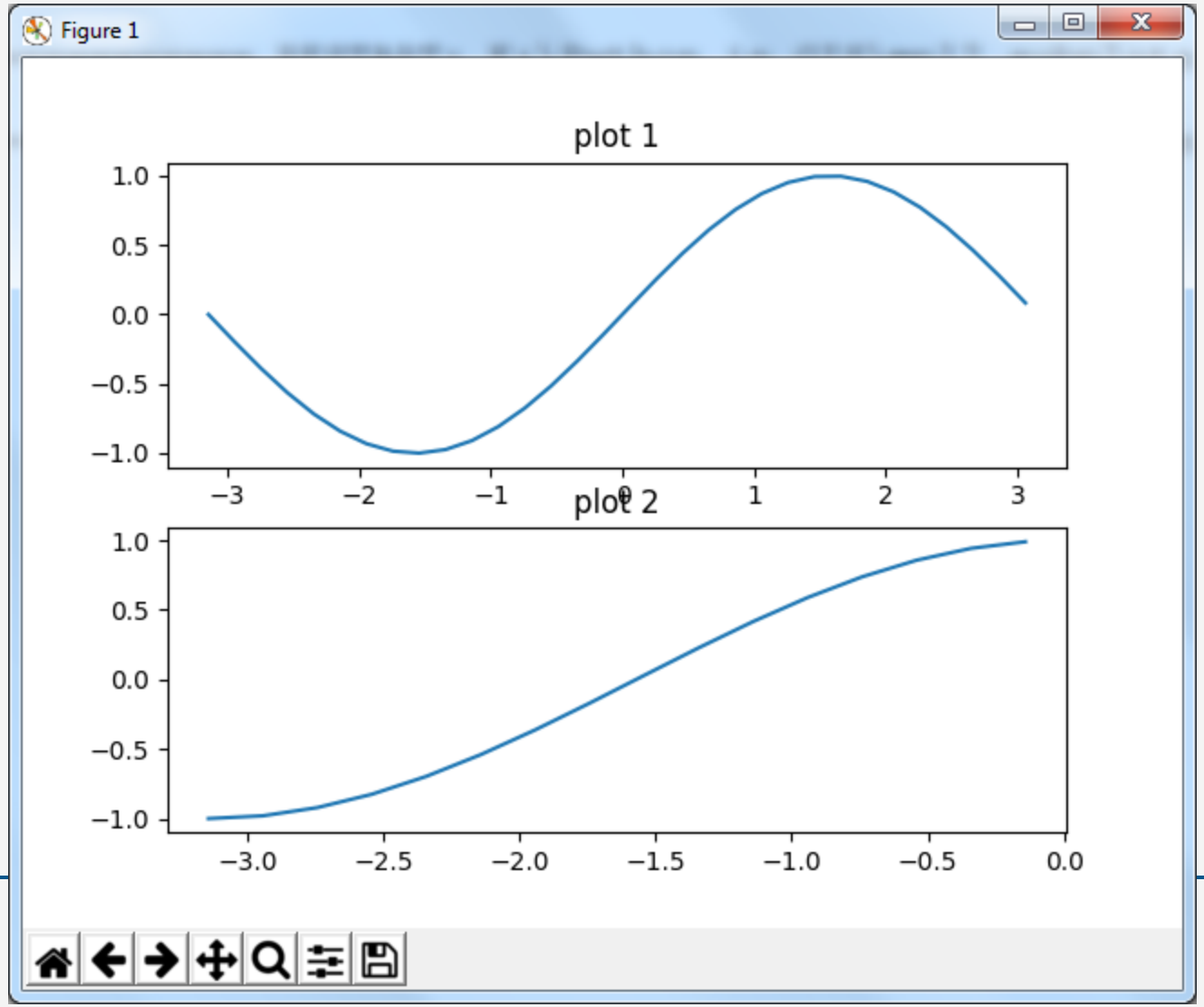
Multiple plots in one figure (2)

By default plots on

top of each other

Plot 1 at the top

Representation of x-axis



Multiple plots in one figure (3)

mpl2_subplots.py - E:\Python_in_GIS\mpl2_subplots.py (3.6.2)

File Edit Format Run Options Window Help

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

# Create some data
x1 = np.arange(-np.pi, np.pi, 0.2)
y1 = np.sin(x1)
x2 = np.arange(-np.pi, 0, 0.2)
y2 = np.cos(x2)

# Initiate a figure
# A figure with two subplots
f, axarr = plt.subplots(2, sharex=True)
# subplots are stored in an array
line = axarr[0].plot(x1, y1)
axarr[0].set_title('plot 1')
axarr[1].plot(x2, y2)
axarr[1].set_title('plot 2')
plt.show()
```

Same x-axis scale

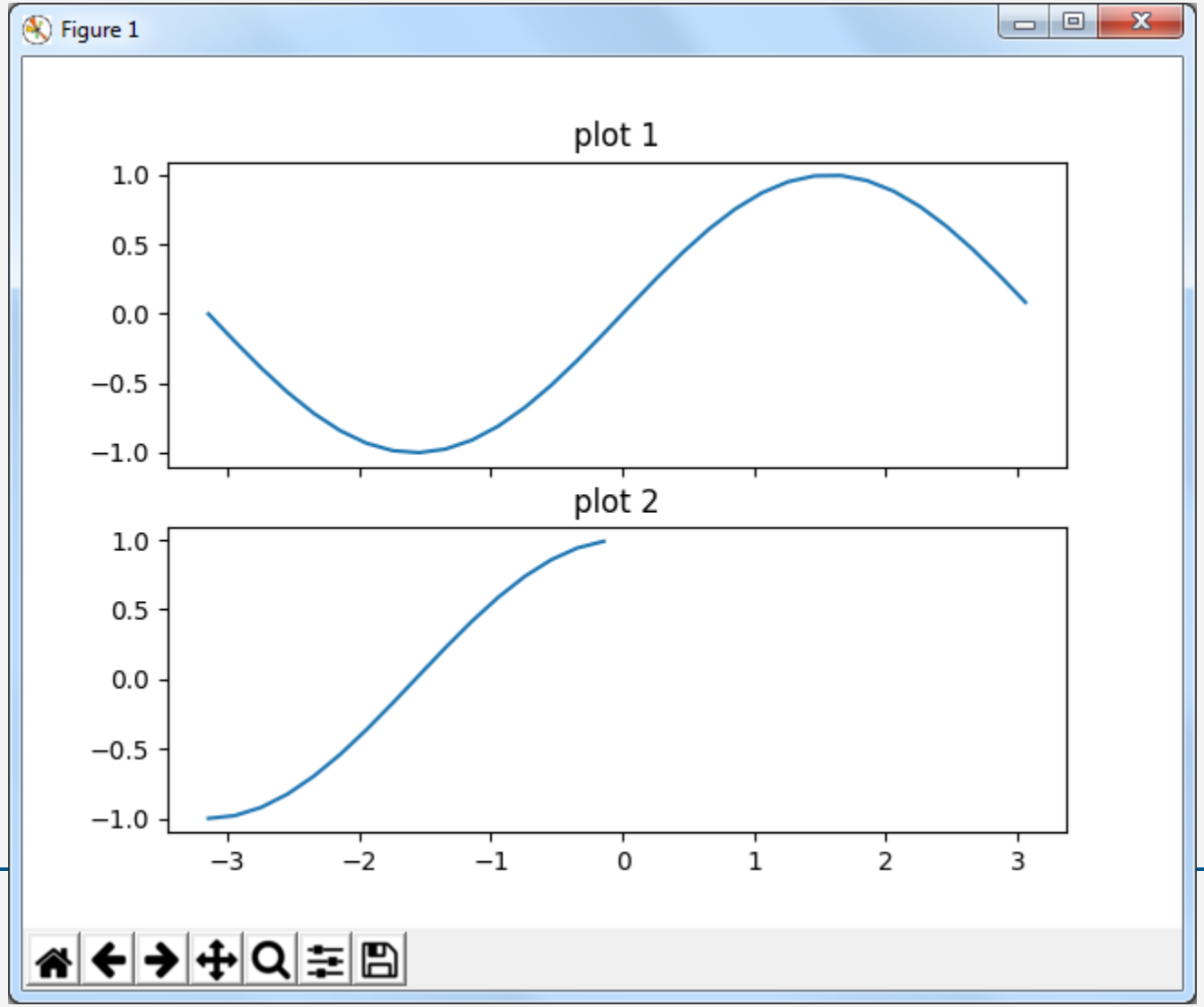
Ln: 18 Col: 10

Multiple plots in one figure (4)

Now x-axis is shared

Easier to compare plots

Note that only the bottom subplot has the labels now



Multiple plots in one figure (5)

mpl2_subplots.py - E:\Python_in_GIS\mpl2_subplots.py (3.6.2)

File Edit Format Run Options Window Help

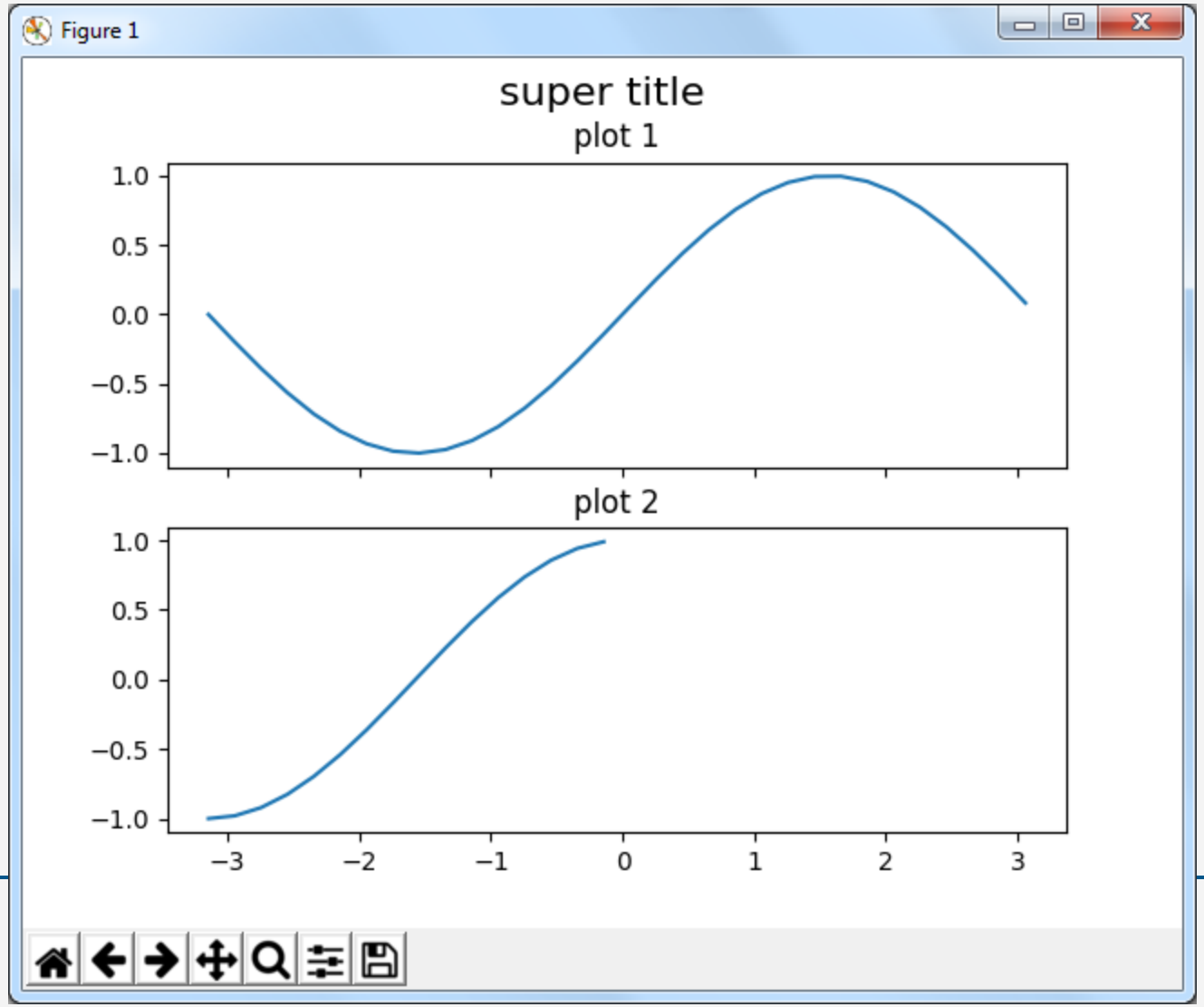
```
import numpy as np
from matplotlib import pyplot as plt

# Create some data
x1 = np.arange(-np.pi, np.pi, 0.2)
y1 = np.sin(x1)
x2 = np.arange(-np.pi, 0, 0.2)
y2 = np.cos(x2)

# Initiate a figure
# A figure with two subplots
f, axarr = plt.subplots(2, sharex=True)
# subplots are stored in an array
line = axarr[0].plot(x1, y1)
axarr[0].set_title('plot 1')
axarr[1].plot(x2, y2)
axarr[1].set_title('plot 2')
f.suptitle('super title', fontsize=16)
plt.show()
```

Ln: 19 Col: 10

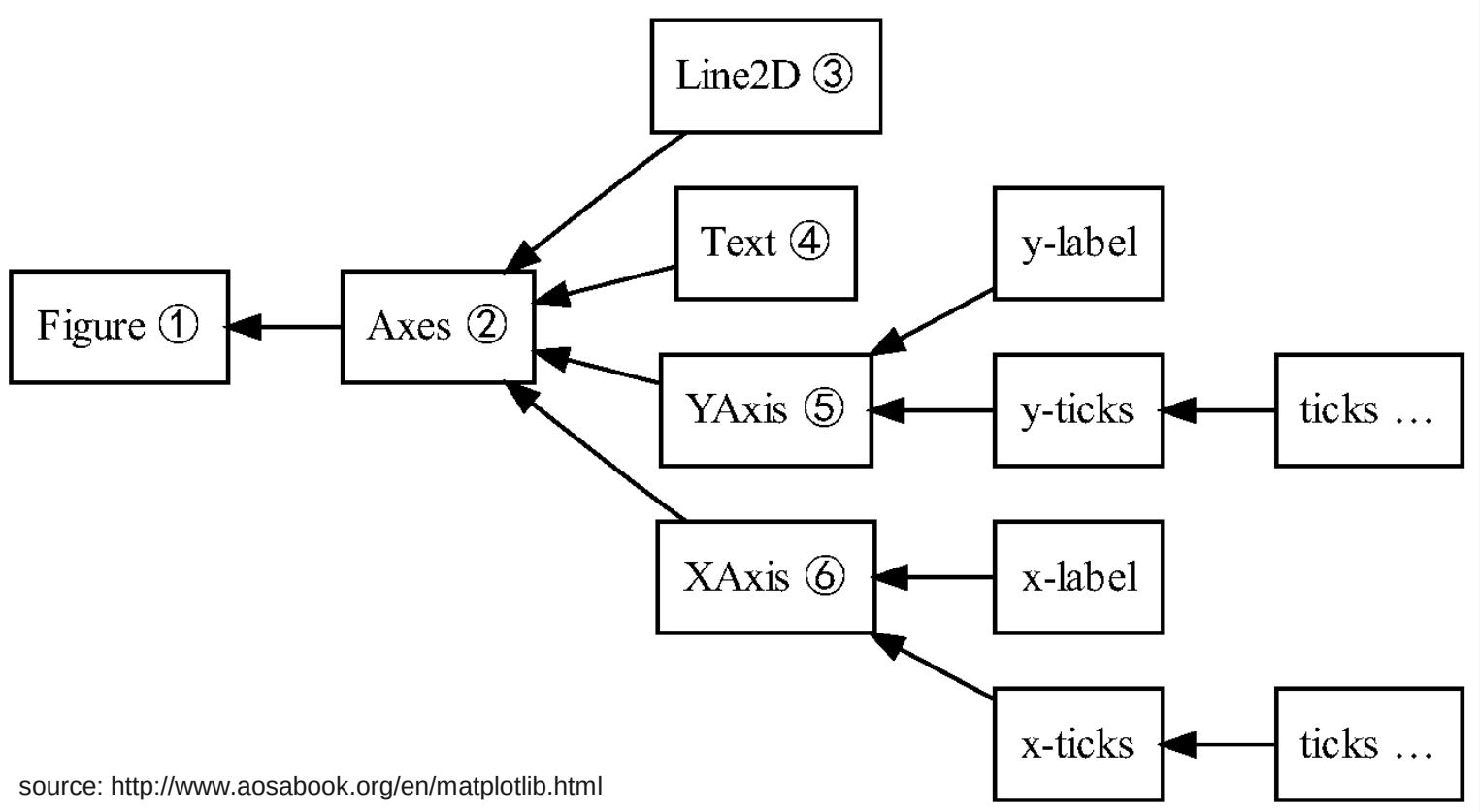
Title of the whole figure, font style and size can be customized

Multiple plots in one figure (6)

Title above all plots

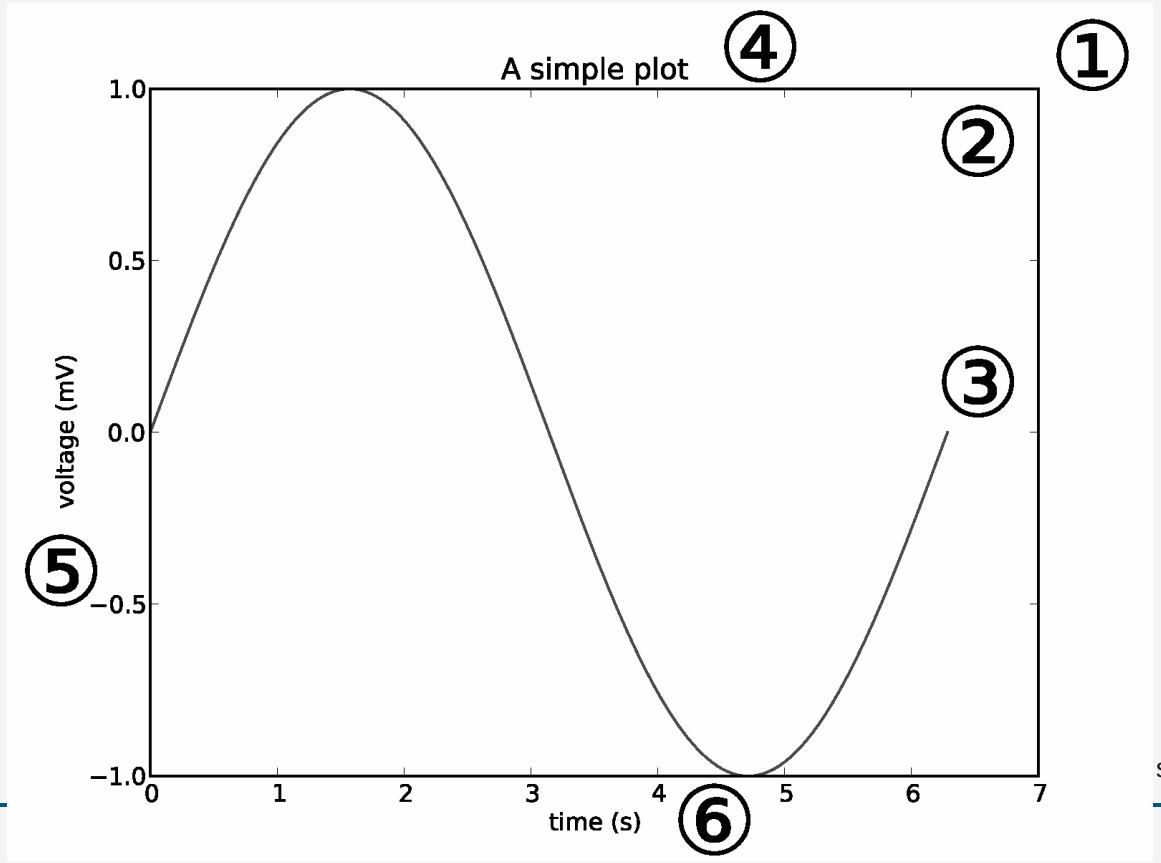


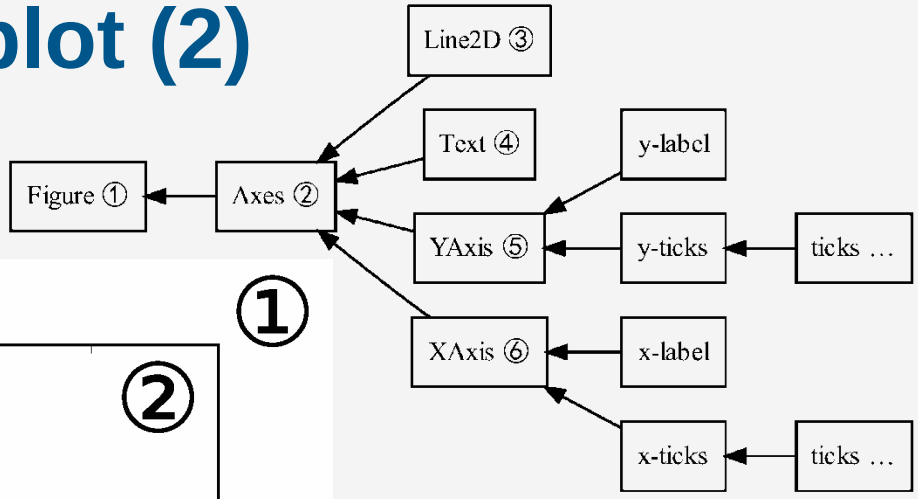
The structure of a plot (1)



source: http://www.aosabook.org/en/matplotlib.html

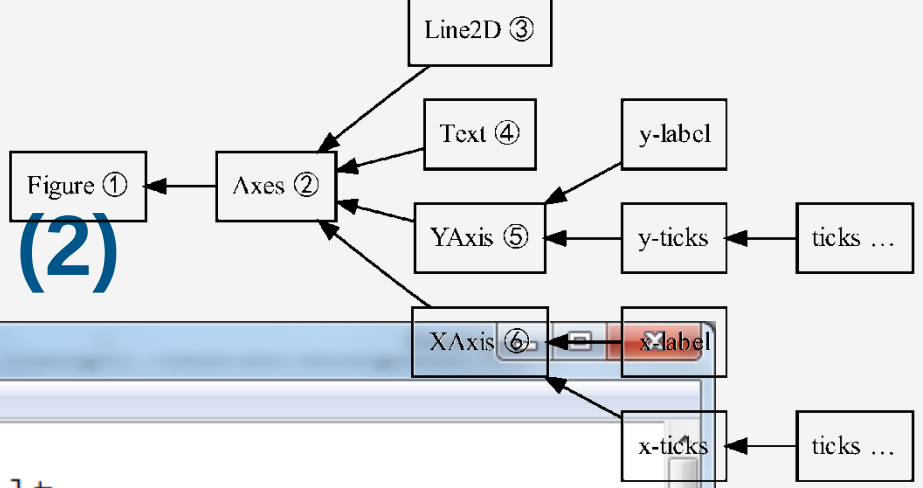
The structure of a plot (2)





source: http://www.aosabook.org/en/matplotlib.html

The structure of a plot (2)



mpl2_subplots.py - E:\Python_in_GIS\mpl2_subplots.py (3.6.2)

File Edit Format Run Options Window Help

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

# Create some data
x1 = np.arange(-np.pi, np.pi, 0.2)
y1 = np.sin(x1)
x2 = np.arange(-np.pi, 0, 0.2)
y2 = np.cos(x2)

# Initiate a figure
# A figure with two subplots
f, axarr = plt.subplots(2, sharex=True)
# subplots are stored in an array
line = axarr[0].plot(x1, y1)
print(line)
axarr[0].set_title('plot 1')
axarr[1].plot(x2, y2)
axarr[1].set_title('plot 2')
f.suptitle('super title', fontsize=16)
plt.show()
```

Ln: 15 Col: 11

Python 3.6.2 Shell

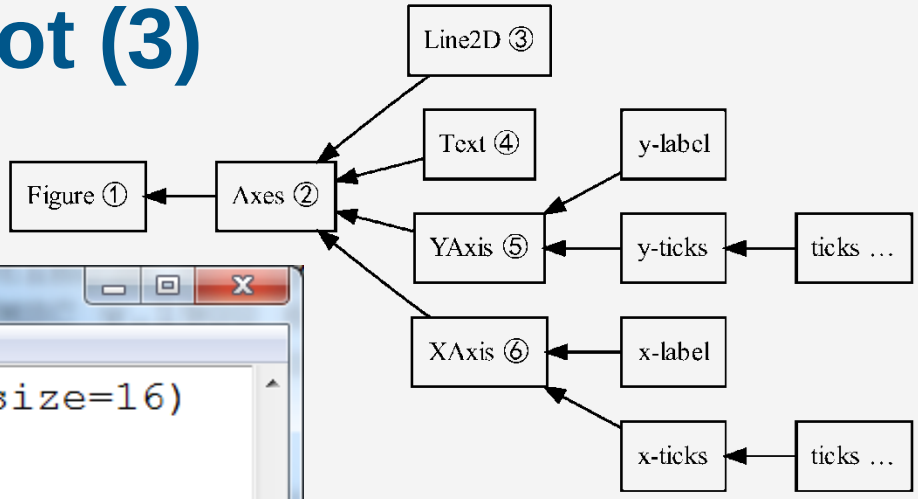
File Edit Shell Debug Options Window Help

```
Python 3.6.2 |Continuum Analytics, Inc.| (default, Jul 20
2017, 12:30:02) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more inform
ation.
>>>
================== RESTART: E:\Python_in_GIS\mpl2_subplots.
py ==================
[<matplotlib.lines.Line2D object at 0x0000000005686E80>]
>>>
```

Ln: 6 Col

Catch the line object in a variable to see the type

The structure of a plot (3)



Axes labels

mpl2_subplots.py - E:\Python_in_GIS\mpl2_subplots.py (3.6.2)

File Edit Format Run Options Window Help

```
f.suptitle('super title', fontsize=16)

# Axis label
axarr[1].set_xlabel('x')
axarr[0].set_ylabel('sin(x)')
axarr[1].set_ylabel('cos(x)')
plt.show()
```

Ln: 20 Col: 0

Why x-label only in plot 2?

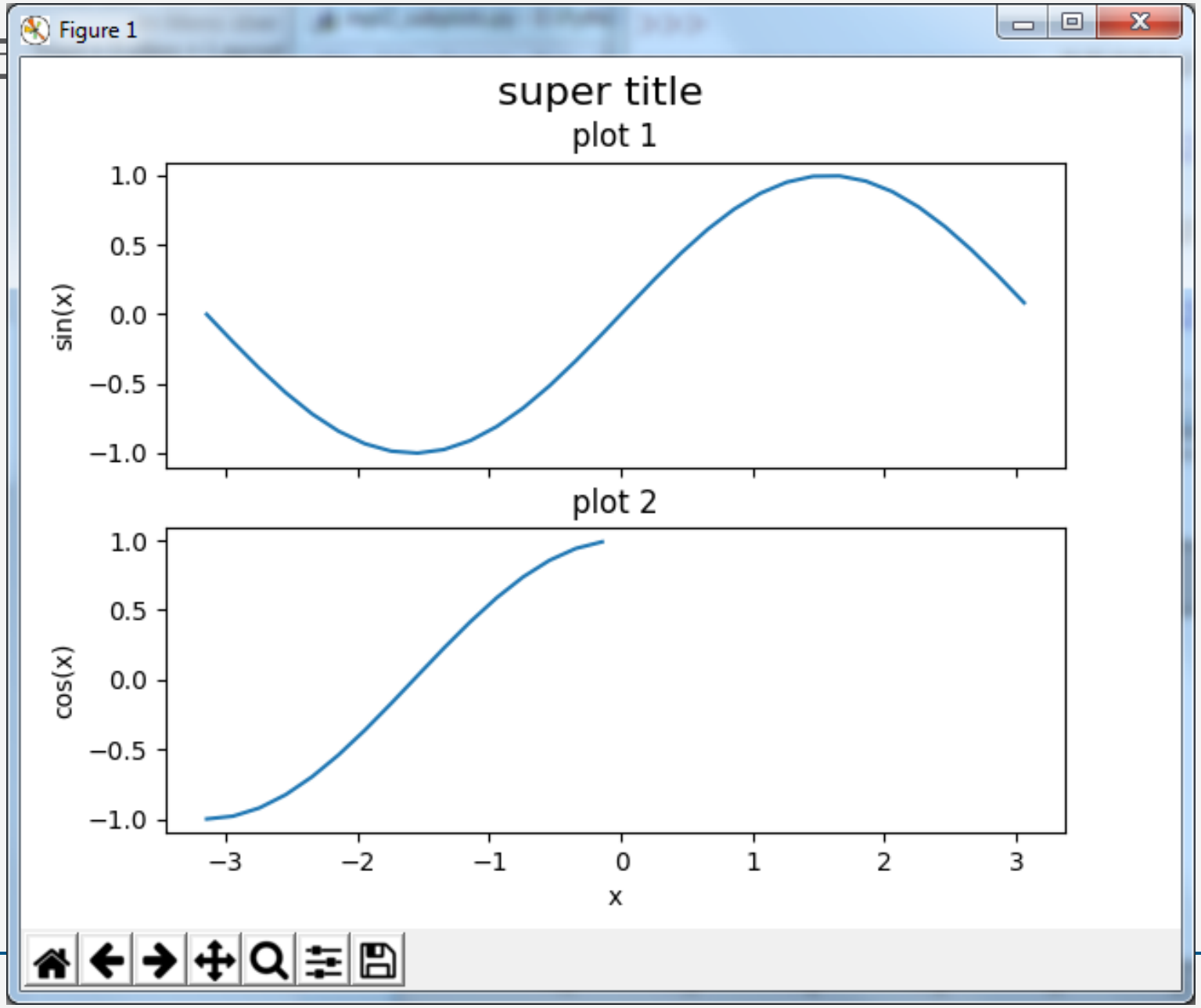
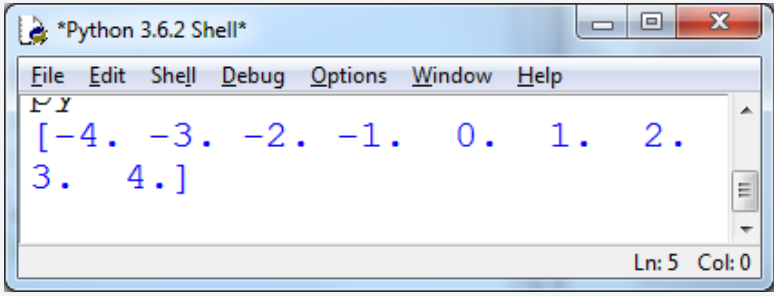


Figure 1
super title
plot 1
sin(x)
1.0
0.5
0.0
−0.5
−1.0
plot 2
cos(x)
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
x

The structure of a plot (4)

TeX markup in any matplotlib text string by $.....$

See: https://matplotlib.org/users/mathtext.html



mpl2_subplots.py - E:\Python_in_GIS\mpl2_subplots.py (3.6.2)

File Edit Format Run Options Window Help

```
# Axis ticks and tick labels
xticks = axarr[1].get_xticks()
print(xticks)
new_ticks = np.arange(-np.pi, np.pi + 0.1, 0.25 * np.pi)
new_labels = [r"$-\pi$", r"$-\frac{3}{4}\pi$",
              r"$-\frac{1}{2}\pi$", r"$-\frac{1}{4}\pi$",
              "$0$", r"$\frac{1}{4}\pi$",
              r"$\frac{1}{2}\pi$", r"$\frac{3}{4}\pi$",
              r"$2\pi$"]
axarr[1].set_xticks(new_ticks)
axarr[1].set_xticklabels(new_labels)

plt.show()
```

Ln: 38 Col: 10

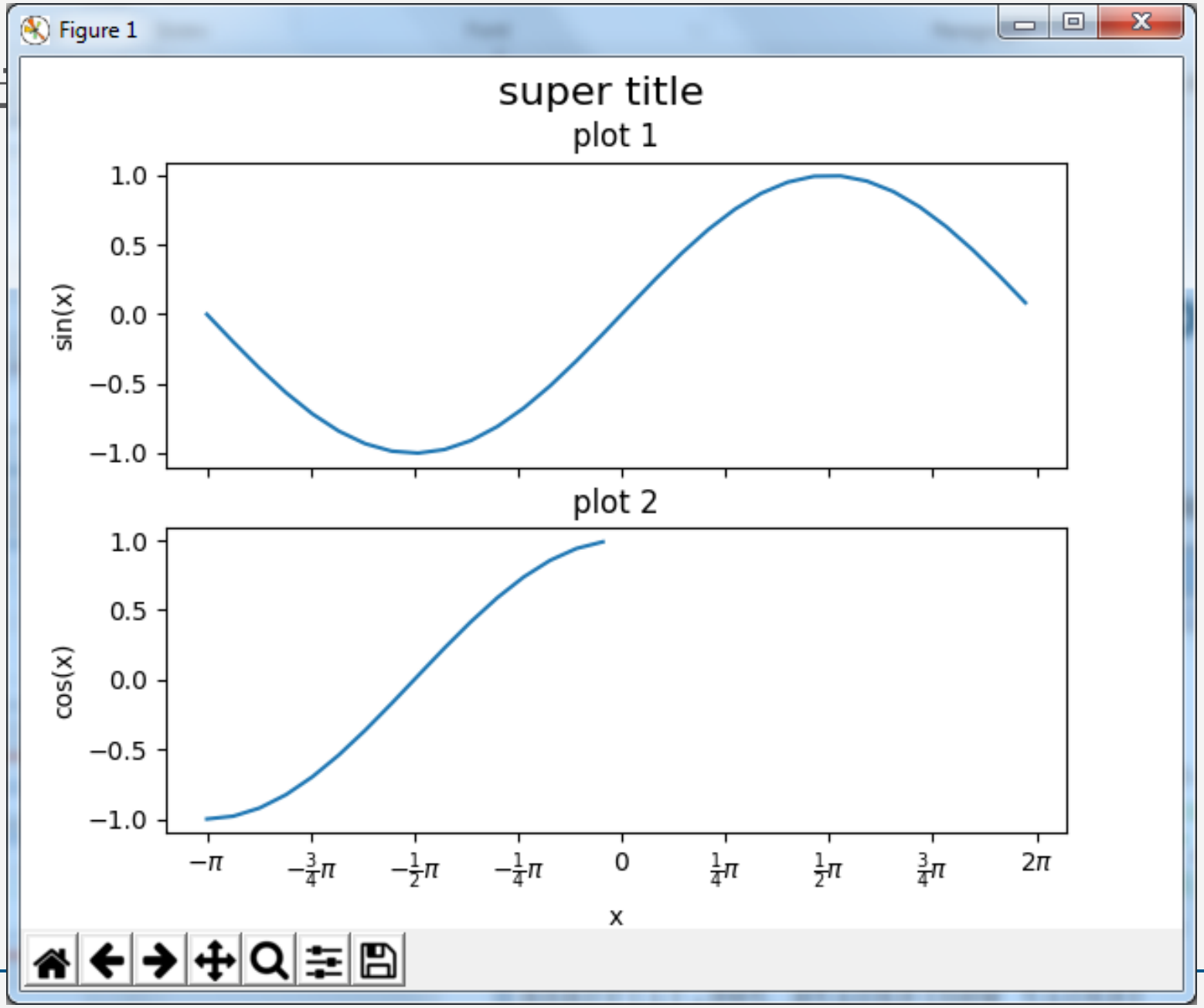
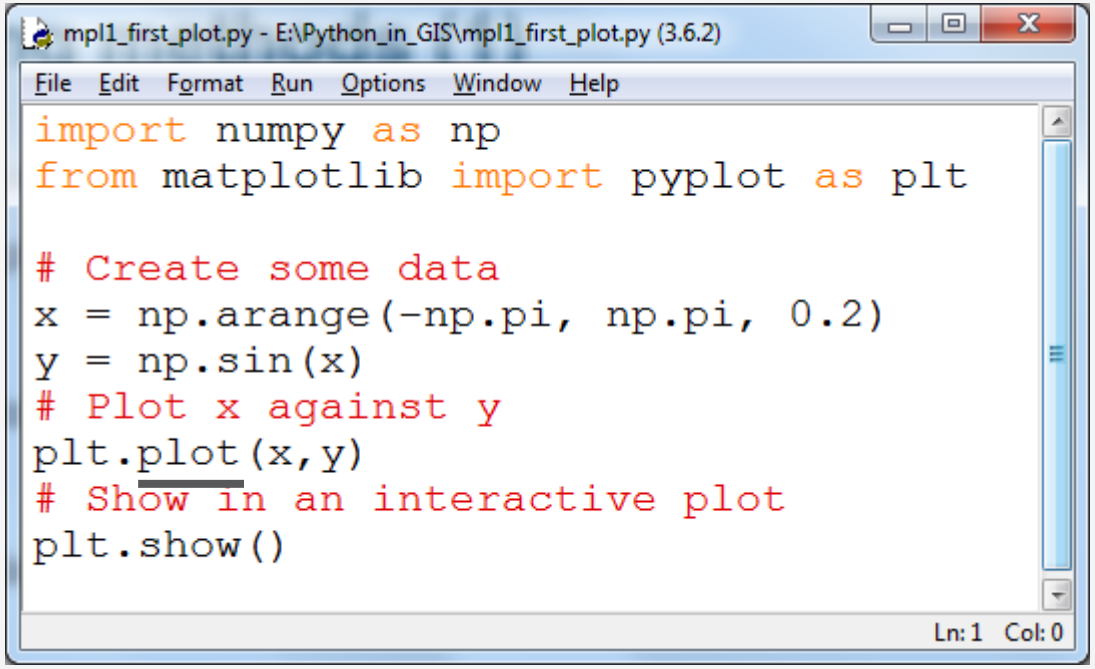


Figure 1
super title
plot 1
sin(x)
1.0
0.5
0.0
−0.5
−1.0
plot 2
cos(x)
1.0
0.5
0.0
−0.5
−1.0
−π
−3/4π
−1/2π
−1/4π
0
1/4π
1/2π
3/4π
2π
x

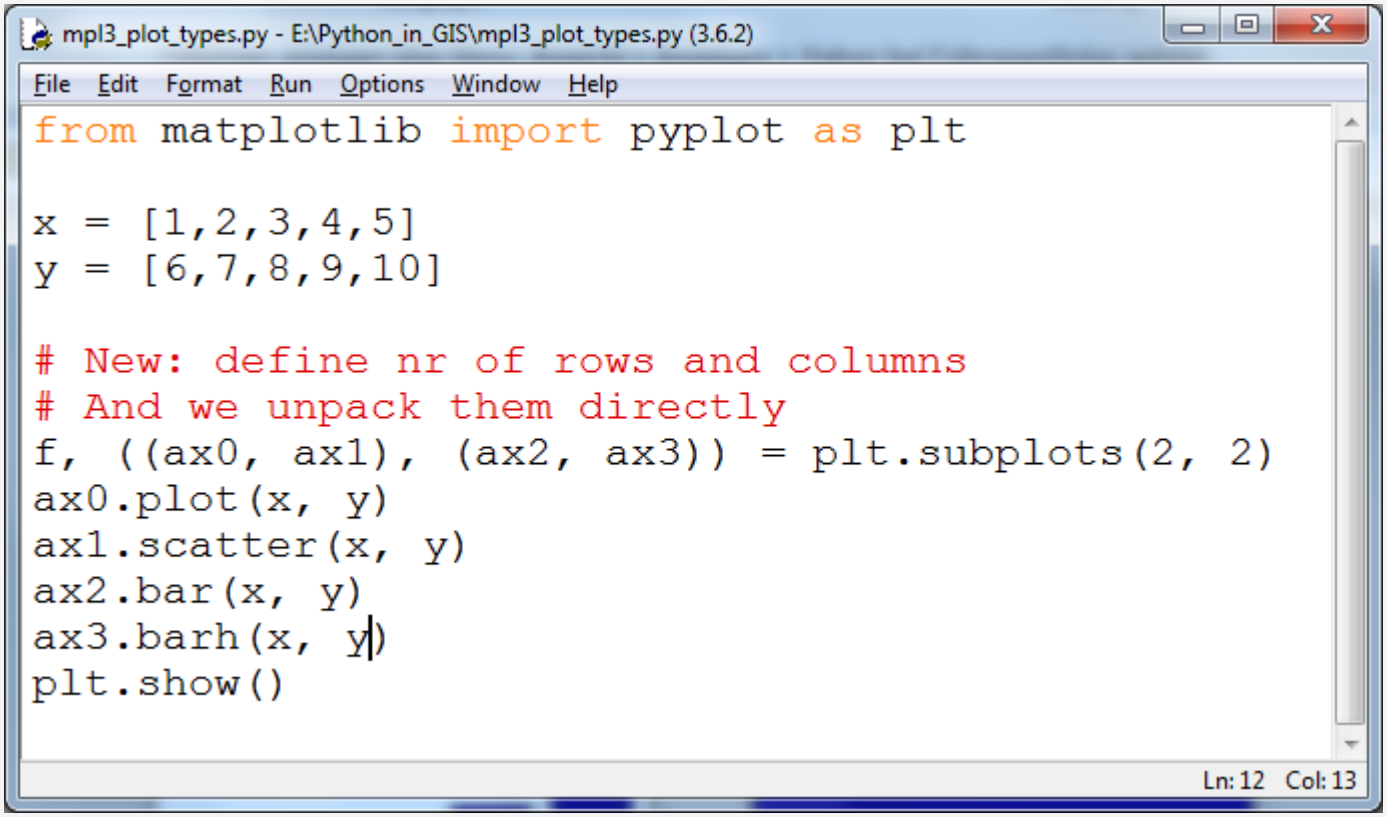
Other plotting methods (1)

Besides ‘plot’, you can use other command for different kinds of plots:

- ‘bar’ or ‘barh’ for bar plots
- ‘hist’ for histograms
- ‘scatter’ for scatter plots
- ‘pie’ for pie plots
- ‘contour’ for contour lines
- ….and more



Other plotting methods (2)



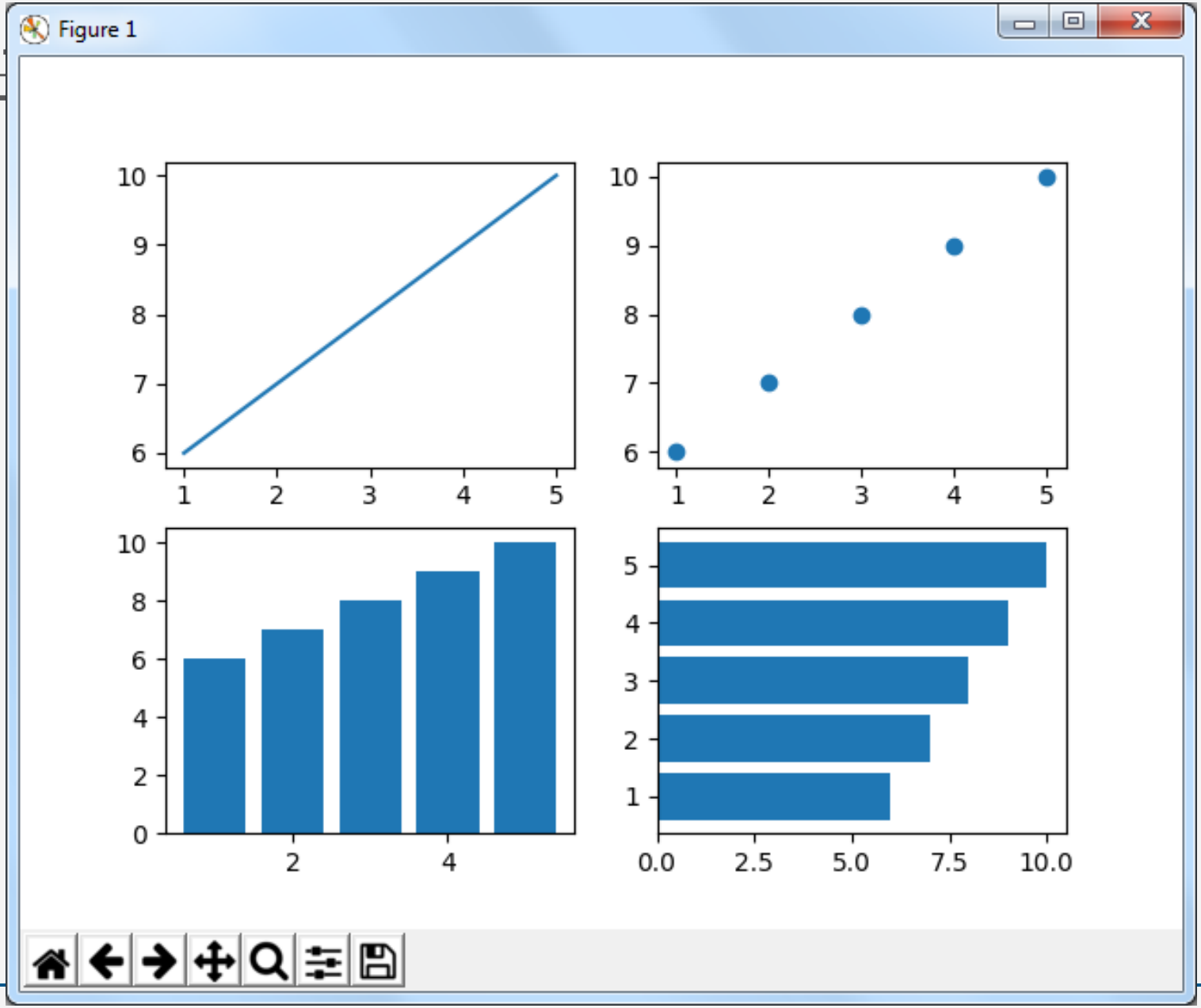


Figure 1
10
9
8
7
6
1
2
3
4
5
10
9
8
7
6
1
2
3
4
5
10
8
6
4
2
0
2
4
5
4
3
2
1
0.0
2.5
5.0
7.5
10.0

Exercise #1

You are given the following land use data about a region:

Land use type	Area (ha)
urban	1000
cropland	20000
pasture	7000
forest	3000

Make a script with two subplots:

- The left-hand subplot showing these data as a bar chart
- The left-hand subplot showing these data as a pie chart

Colors and drawing styles (1)

Line and marker styles:

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

Colors and drawing styles (2)

source: http://www.scipy-lectures.org/intro/matplotlib/matplotlib.html

Colors and drawing styles (3)

Colors:

- Simple
- Or any legal html name, e.g.:

  color = 'red'

  color = 'burlywood'

  color = 'chartreuse‘
- Or using html hex string:

  color = '#eeefff'
- Or you can an R,G,B tuple, each in [0,1].

  color=(0.1843, 0.3098, 0.3098)

Simple:

character	color
‘b’	blue
‘g’	green
‘r’	red
‘c’	cyan
‘m’	magenta
‘y’	yellow
‘k’	black
‘w’	white

Colors and drawing styles (4)

Color maps:

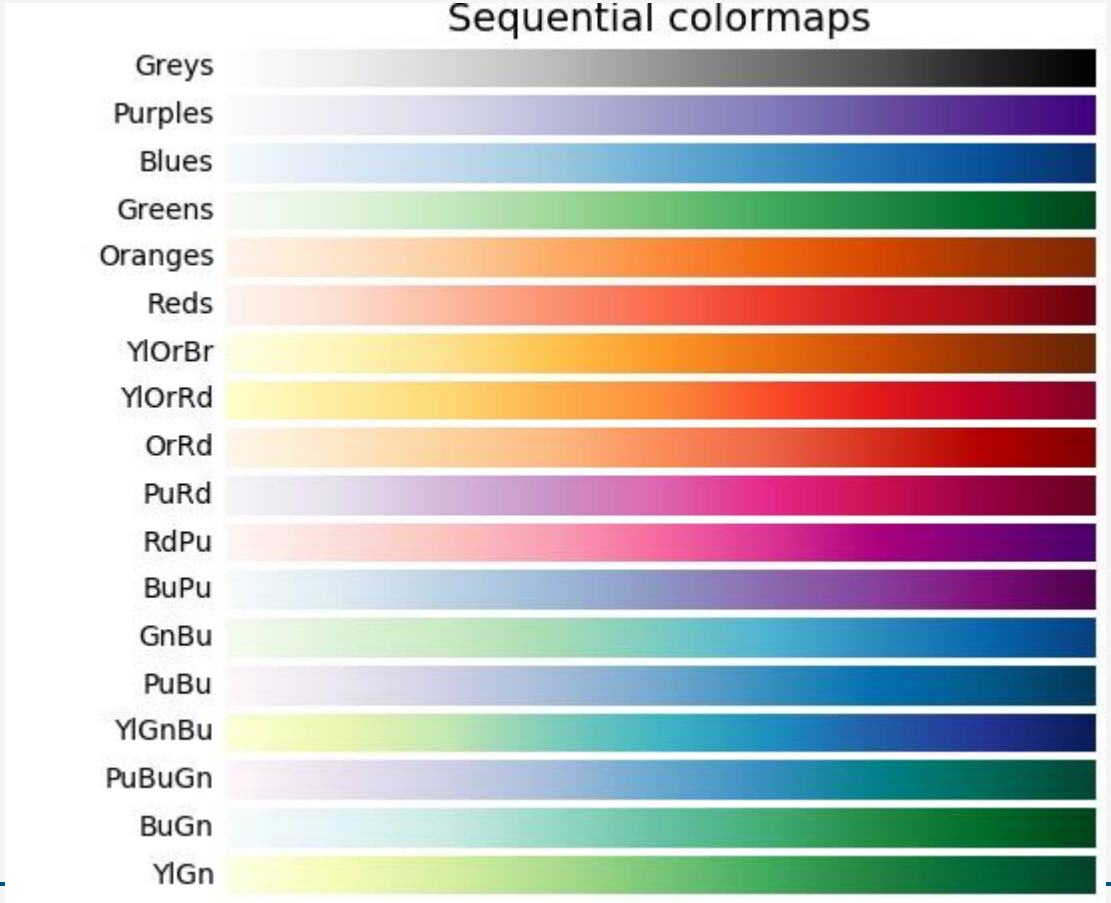
Work with:

Scatter

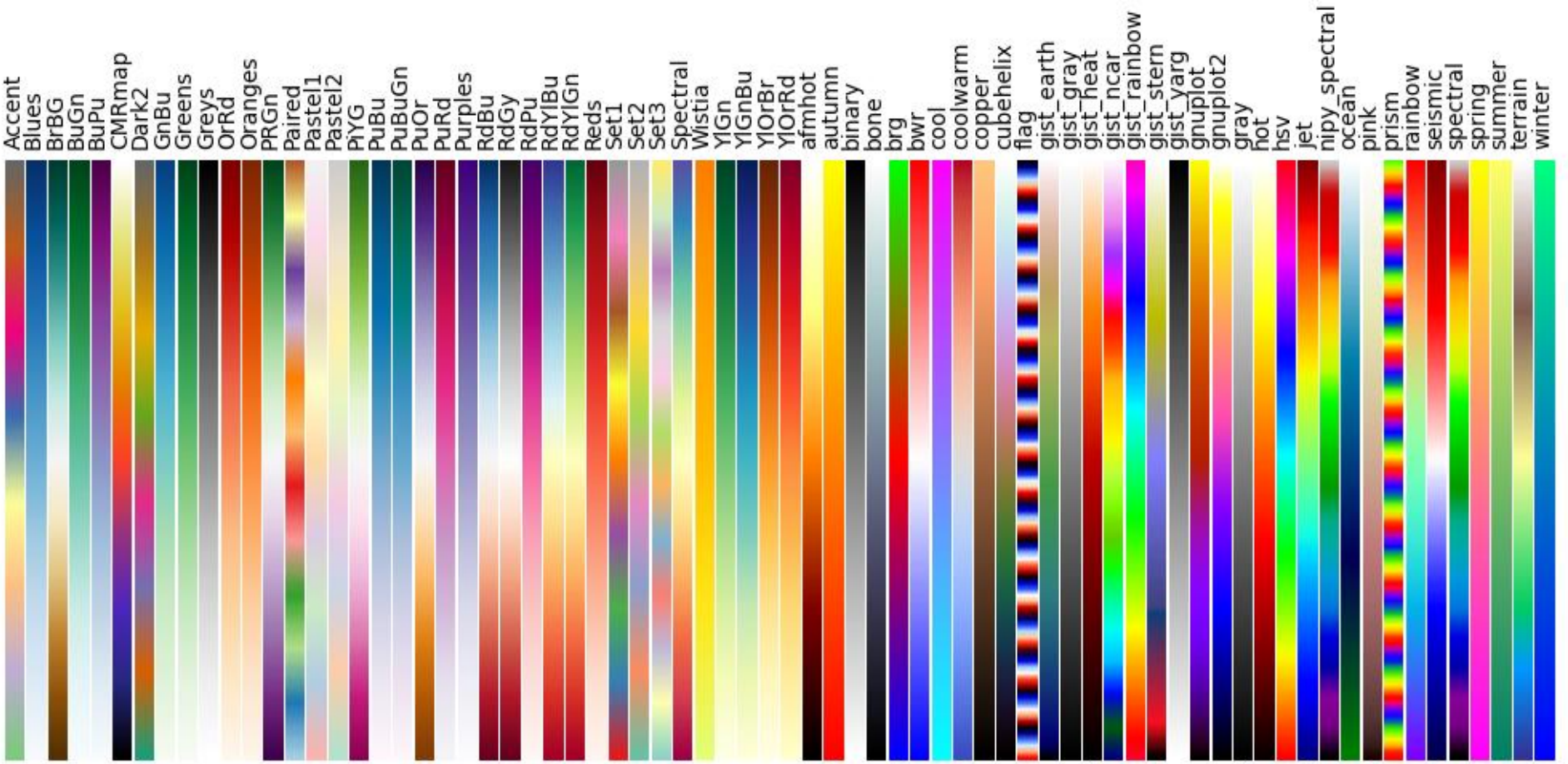
Contour

..

But not with plot!



Colors and drawing styles (5)



source: http://www.scipy-lectures.org/intro/matplotlib/matplotlib.html

Colors and drawing styles (5)

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

mpl3_plot_types.py - E:\Python_in_GIS\mpl3_plot_types.py (3.6.2)

File Edit Format Run Options Window Help

```
from matplotlib import pyplot as plt

x = [1,2,3,4,5]
y = [6,7,8,9,10]

# New: define nr of rows and columns
# And we unpack them directly
f, ((ax0, ax1), (ax2, ax3)) = plt.subplots(2, 2)
ax0.plot(x, y, 'r--')
ax1.scatter(x, y, c=y, cmap='bwr', s = 35)
ax2.bar(x, y, color='k')
ax3.barh(x, y, color='y')
plt.show()
```

Ln: 7 Col: 8

Combine color and line / marker style

Attribute for the color, color map, size

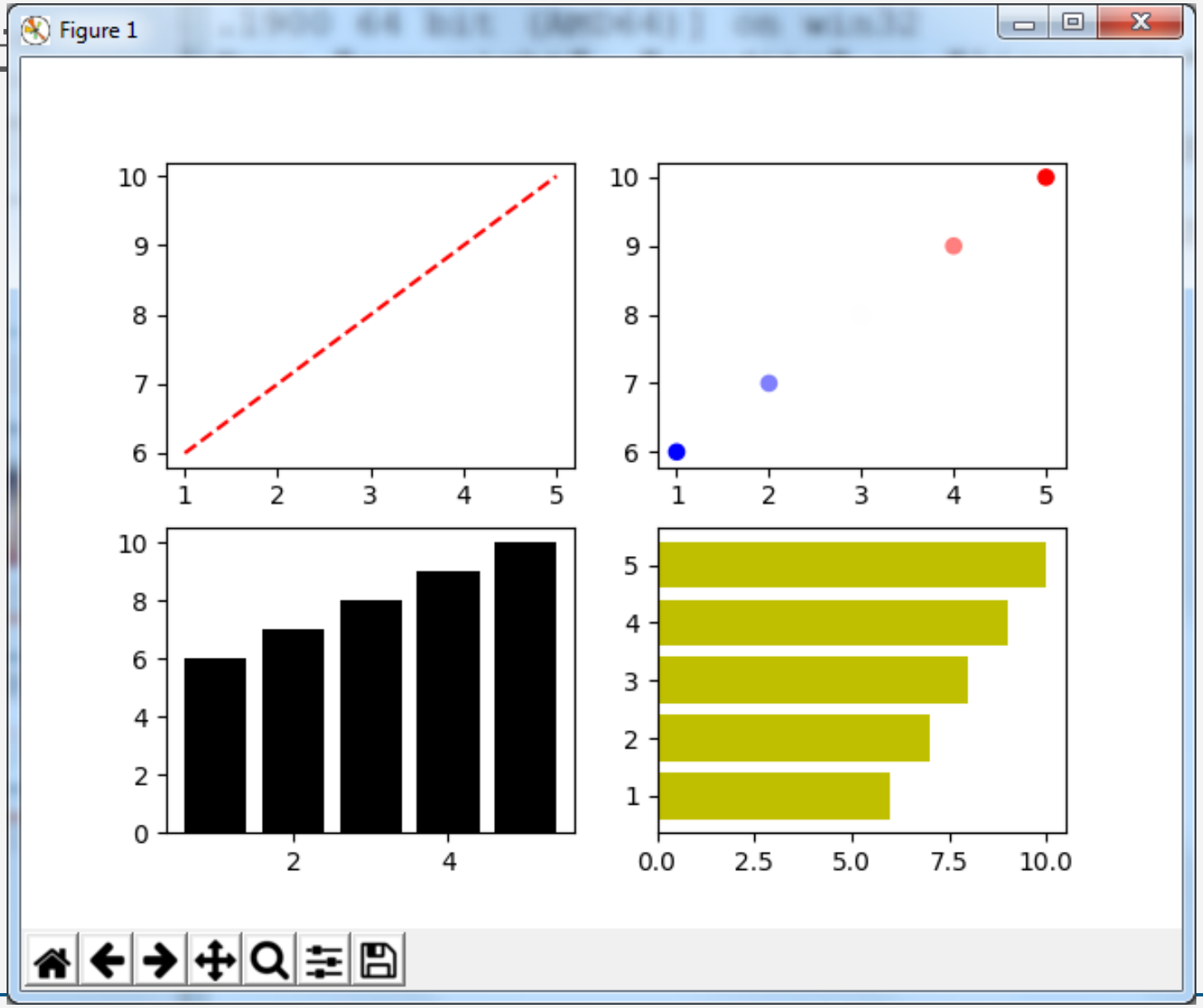


Figure 1
10
9
8
7
6
1
2
3
4
5
10
9
8
7
6
1
2
3
4
5
10
8
6
4
2
0
2
4
5
4
3
2
1
0.0
2.5
5.0
7.5
10.0

Plotting a map using a shapefile (1)

mpl4_vector_layer.py

```
1  import matplotlib.pyplot as plt
2  import ogr
3  import os
4
5  #############
6  # Reading a shapefile
7  in_path = os.path.join('C:\\', 'Users', 'verstege', \
8  'Documents', 'education', 'python_in_GIS', '2017_2018',
9  'data', 'shapefiles', 'track_reprojected.shp')
10
11 # Get the correct driver and open file for reading
12 driver = ogr.GetDriverByName('ESRI Shapefile')
13 track = driver.Open(in_path, 0)
14
15 # Get the layer
16 layer = track.GetLayer(0)
17
```

Using ogr, so swiched to QGIS!

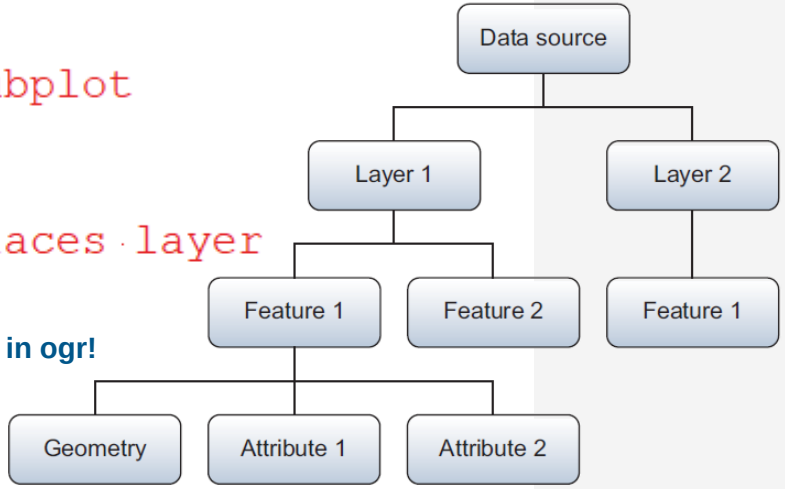
Open the shapefile for reading

Plotting a map using a shapefile (2)

mpl4_vector_layer.py

```
17
18  # Create the plot
19  # Not really necessary with one subplot
20  f, axarr = plt.subplots(1)
21
22  # Access single features in the places layer
23  # And plot them
24  for feat in layer:
25      pt = feat.geometry()
26      x = pt.GetX()
27      y = pt.GetY()
28      # No indexing for axarr, because only one subplot
29      axarr.plot(x, y, 'ro')
30
31  plt.show()
32
33
```

Remember structure in ogr!



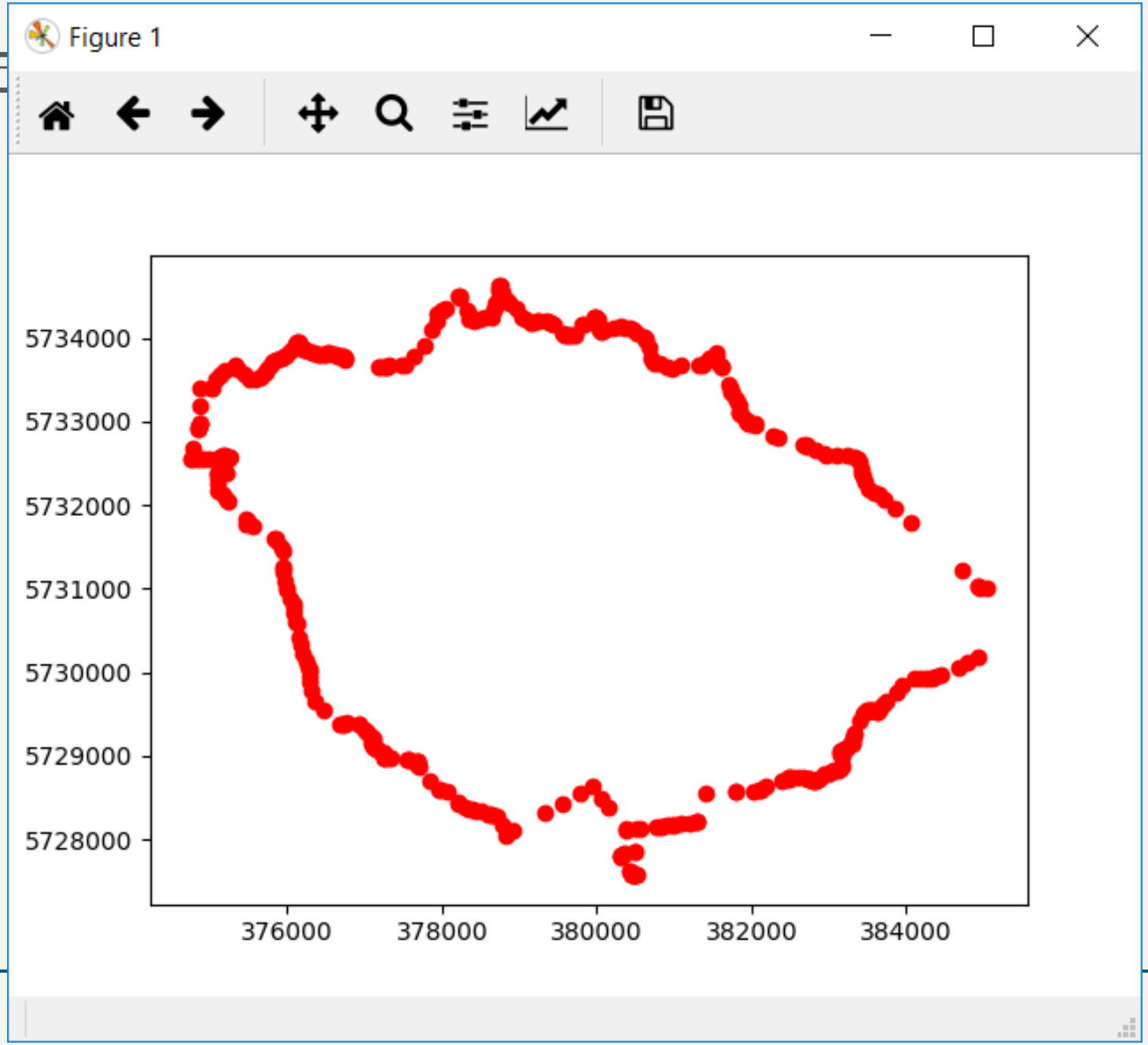


Figure 1
5734000
5733000
5732000
5731000
5730000
5729000
5728000
376000
378000
380000
382000
384000

Plotting a map using a shapefile (4)

mpl4_vector_layer.py

```
18  # Create the plot
19  # Not really necessary with one subplot
20  f, axarr = plt.subplots(1)
21
22  # Access single features in the places layer
23  # And plot them
24  for feat in layer:
25      pt = feat.geometry()
26      x = pt.GetX()
27      y = pt.GetY()
28      # No indexing for axarr, because only one subplot
29      axarr.plot(x, y, 'ro')
30
31  # Making the axes' units equal
32  plt.axis('equal')
33  plt.show()
34
```

Same horizontal and vertical scales, necessary for maps

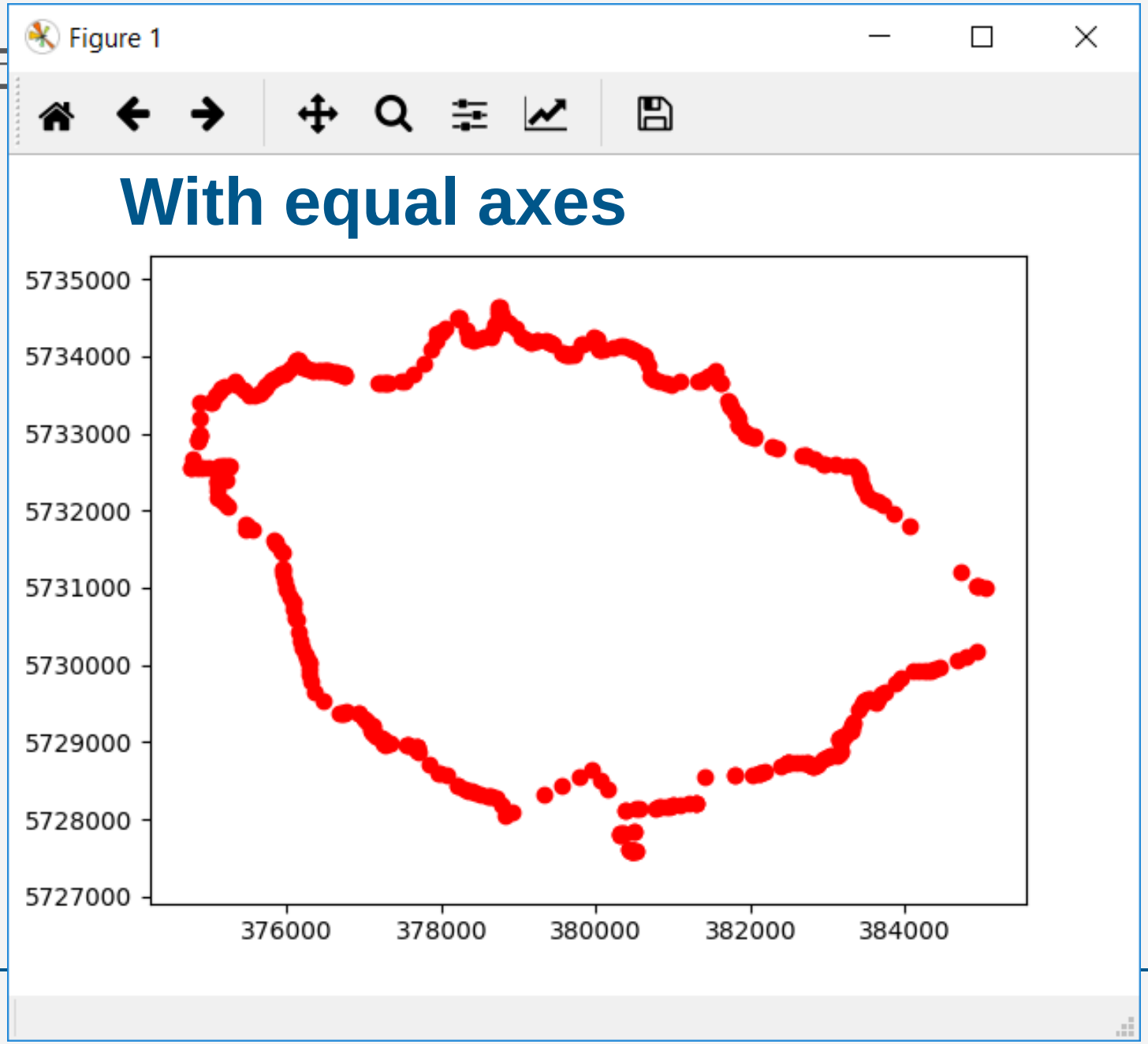


Figure 1
With equal axes
5735000
5734000
5733000
5732000
5731000
5730000
5729000
5728000
5727000
376000
378000
380000
382000
384000

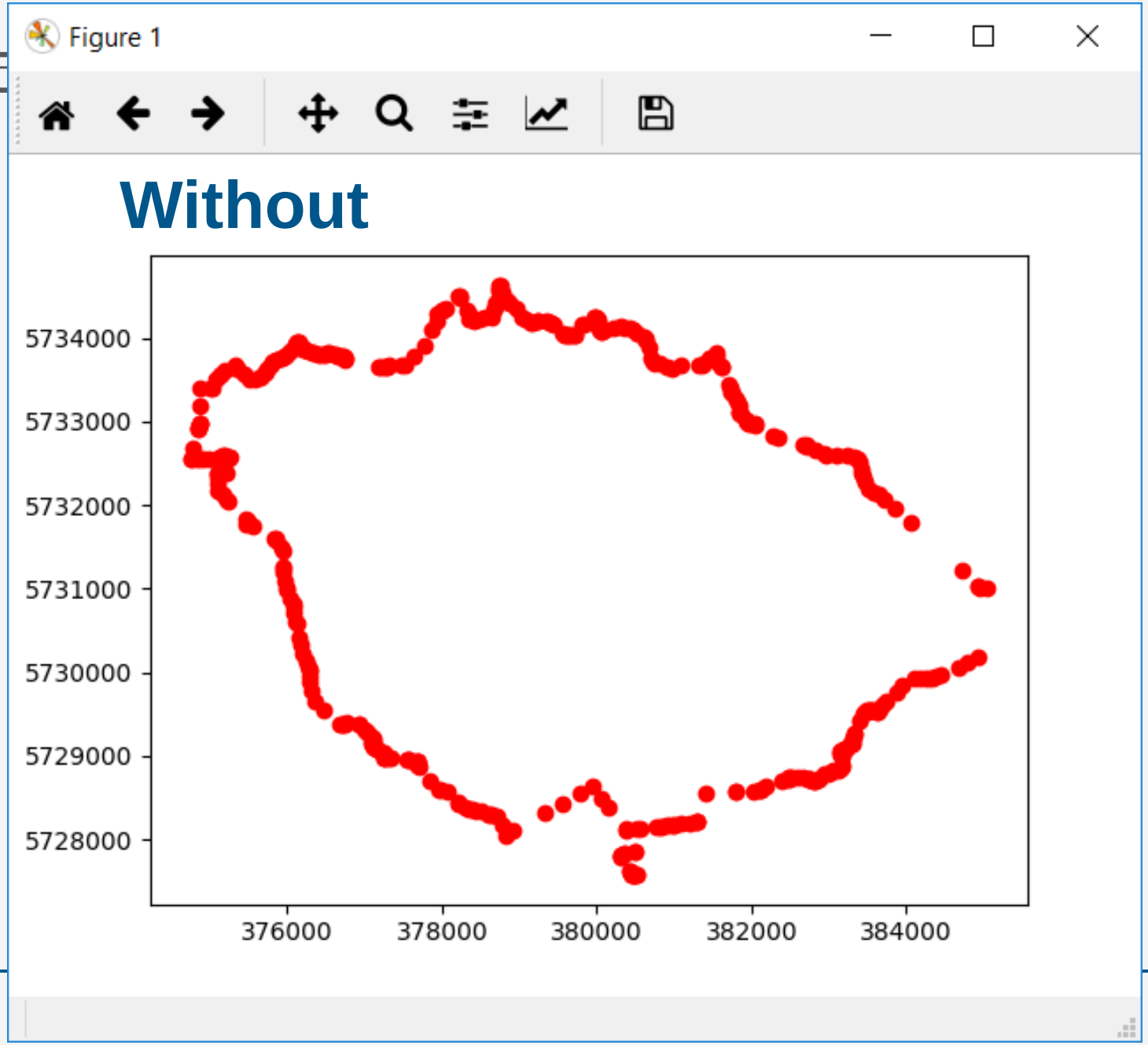


Figure 1
Without
5734000
5733000
5732000
5731000
5730000
5729000
5728000
376000
378000
380000
382000
384000

Exercise #2

In the data folder there is a shapefile gps_track_projected.shp

Make a script that:

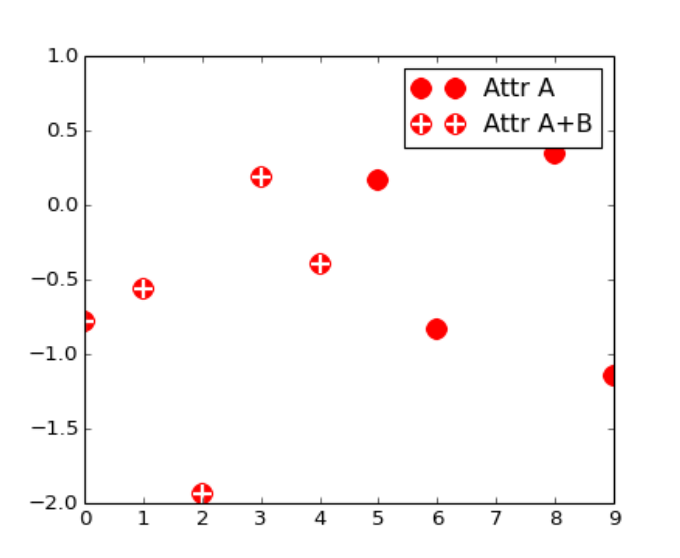
- Plots the shapefile
- Such that the color of the markers (the circles) corresponds to the elevation from the ‘ele’ field of the shapefile

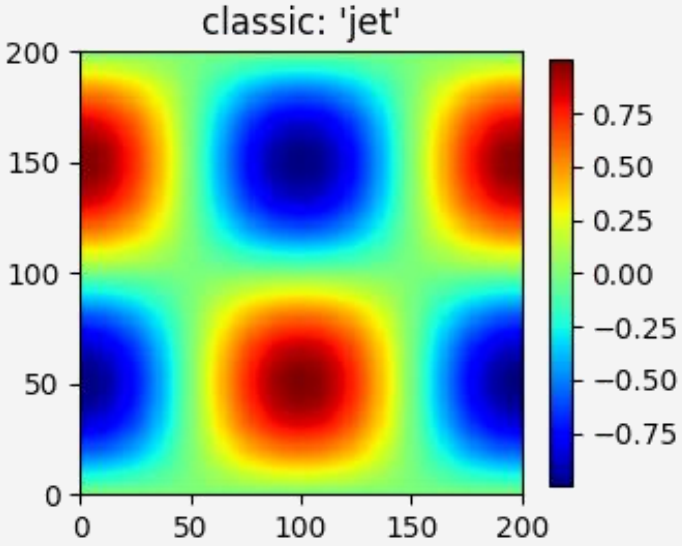
Note: you have not yet learned how to make a legend, so you do not have to do that.

Legend types

Two types of legends in matplotlib (or in general)

- With items - for categorical data
- A color ramp (colormap) - for continuous data





Building a legend with items (1)

See also: https://matplotlib.org/users/legend_guide.html

mpl5.py

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  x = np.arange(0,6,1)
5  y1 = np.arange(0,3,0.5)
6  y2 = np.arange(0,12,2)
7
8  f, axarr = plt.subplots(1)
9  one = axarr.plot(x, y1, 'r--', label='gradual line')
10 two = axarr.plot(x, y2, 'b:', label='steep line')
11 # legend with location, customized
12 axarr.legend()
13 plt.show()
```

Need a label for legend

Place the legend

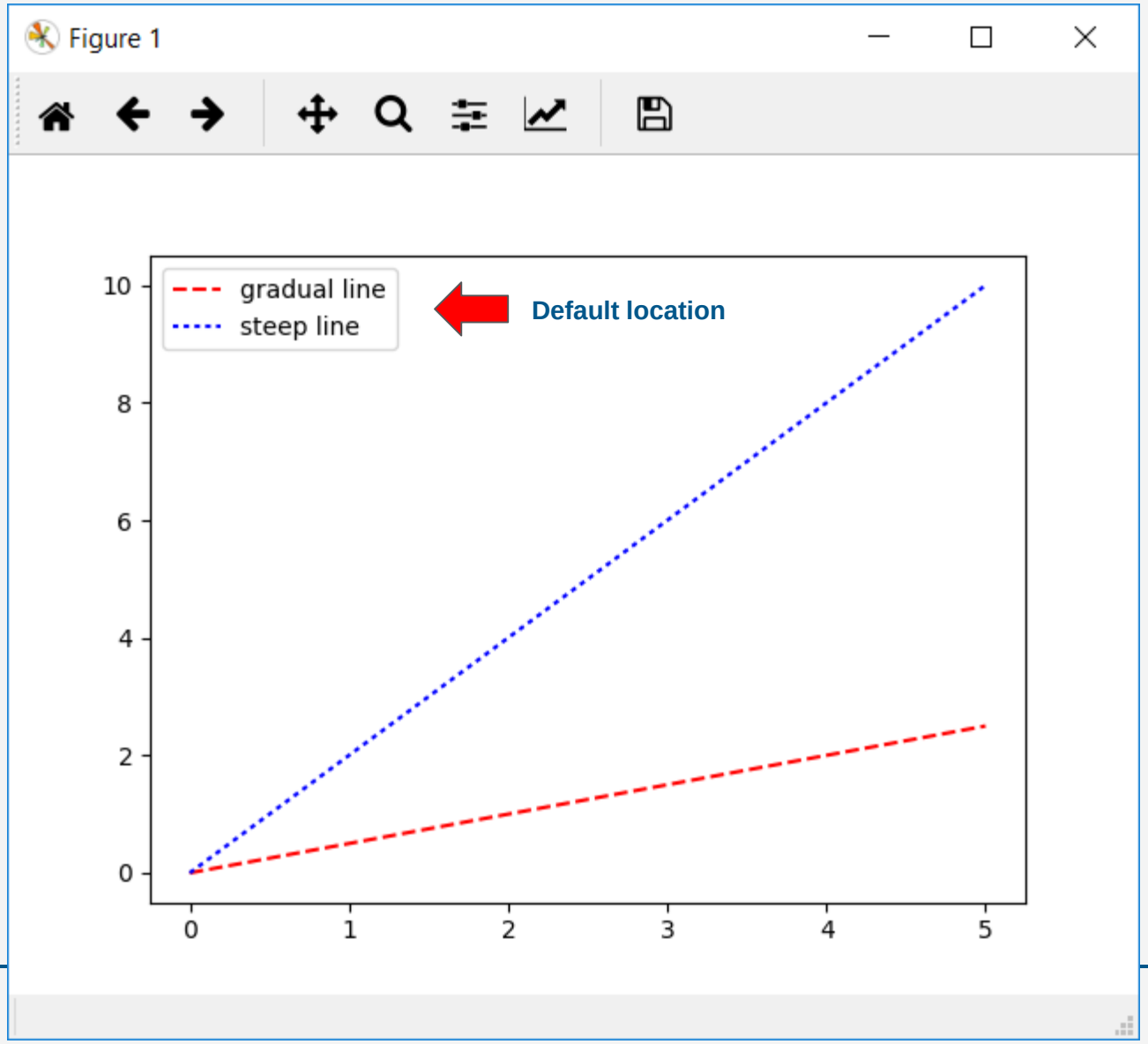
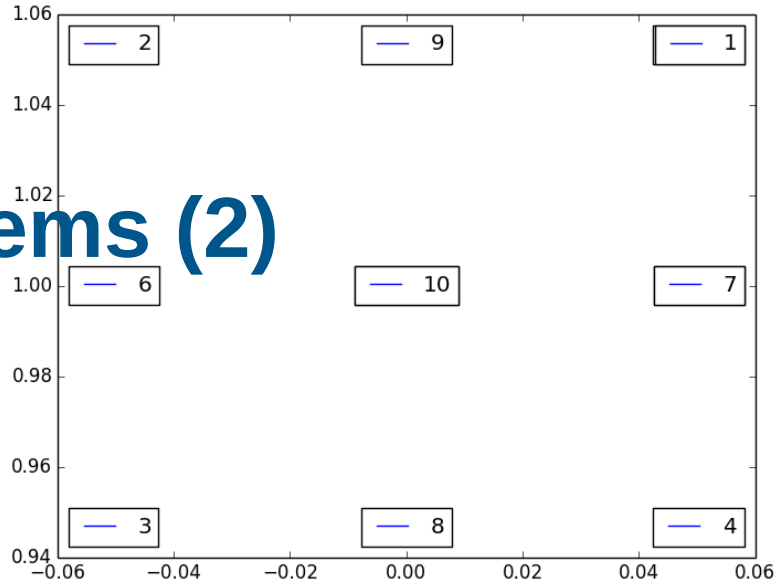


Figure 1
gradual line
steep line
Default location
10
8
6
4
2
0
0
1
2
3
4
5

Building a legend with items (2)

mpl5.py

```
1  from matplotlib import pyplot as
2  import numpy as np
3
4  x = np.arange(0,6,1)
5  y1 = np.arange(0,3,0.5)
6  y2 = np.arange(0,12,2)
7
8  f, axarr = plt.subplots(1)
9  one = axarr.plot(x, y1, 'r--', label='gradual line')
10 two = axarr.plot(x, y2, 'b:', label='steep line')
11 # legend with location, customized
12 axarr.legend(loc=7)
13 plt.show()
14
```



source: https://stackoverflow.com/questions/28521744/error-of-adding-a-legend-for-a-plot-in-python-3-2-matplotlib

Location String	Location Code
‘best’	0
‘upper right’	1
‘upper left’	2
‘lower left’	3
‘lower right’	4
‘right’	5
‘center left’	6
‘center right’	7
‘lower center’	8
‘upper center’	9
‘center’	10

source:https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.legend

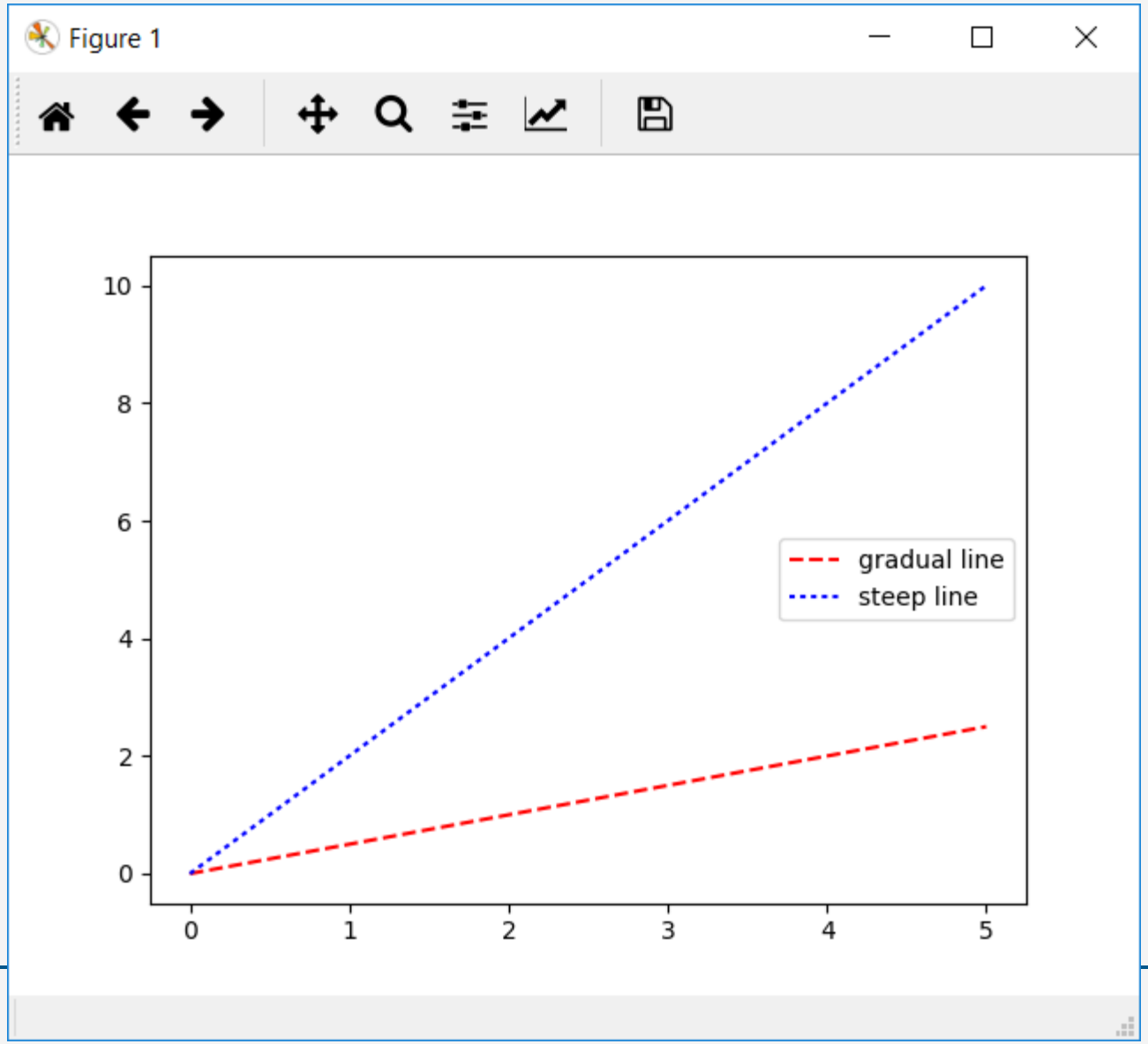


Figure 1
10
8
6
4
2
0
0
1
2
3
4
5
gradual line
steep line

Building a legend with items (3)

Many things adjustable: https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.legend

mpl5.py

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  x = np.arange(0,6,1)
5  y1 = np.arange(0,3,0.5)
6  y2 = np.arange(0,12,2)
7
8  f, axarr = plt.subplots(1)
9  one = axarr.plot(x, y1, 'r--', label='gradual line')
10 two = axarr.plot(x, y2, 'b:', label='steep line')
11 # legend with location, customized
12 axarr.legend(loc=2, bbox_to_anchor=(0,1.1), ncol=2)
13 plt.show()
14
```

Coordinates relative to loc 2 and nr of columns in legend

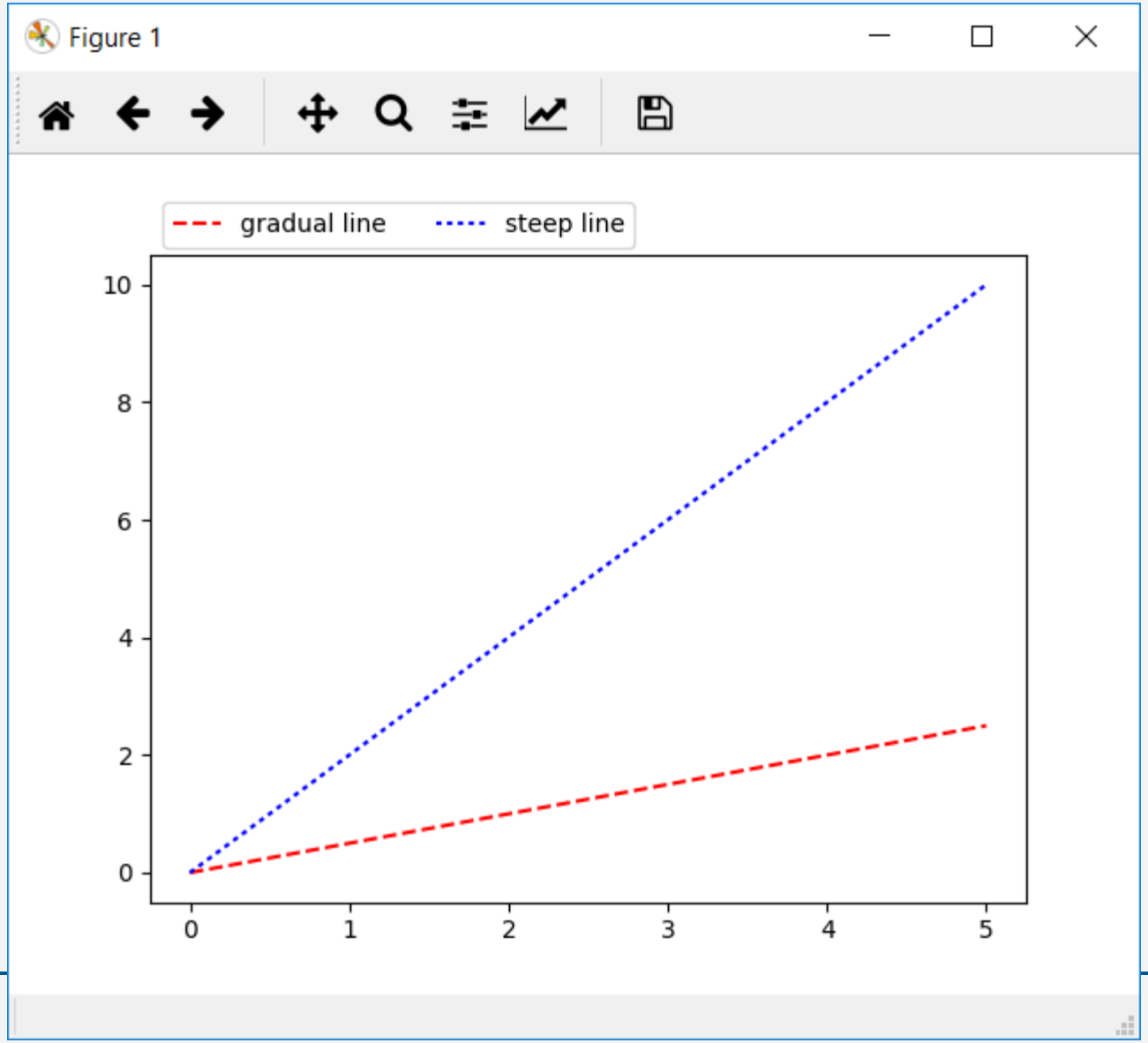


Figure 1
gradual line
steep line
10
8
6
4
2
0
0
1
2
3
4
5

Building a colormap legend (1)

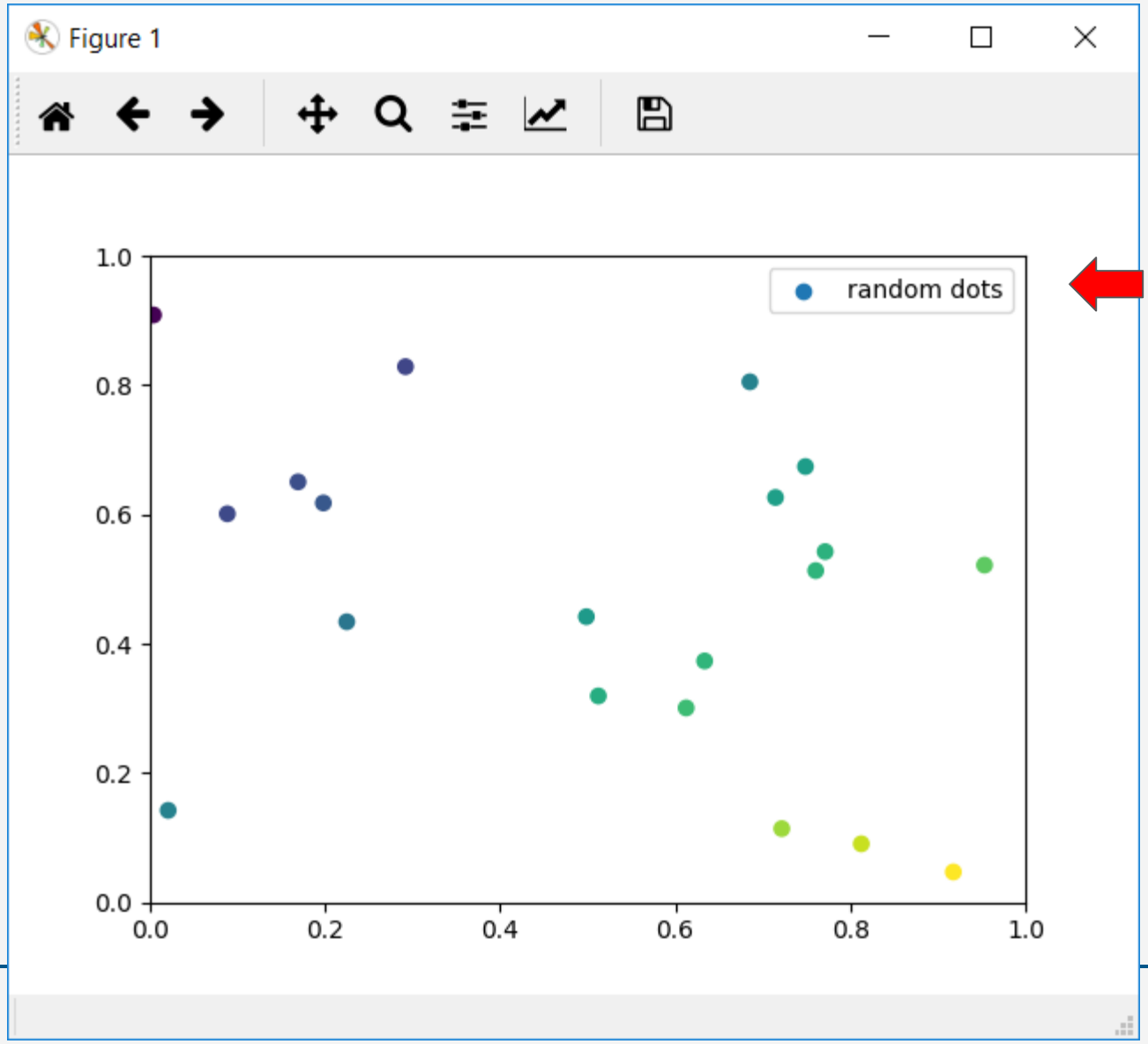
mpl6.py

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  np.random.seed(seed=10)
5  x = np.random.random(20)
6  y = np.random.random(20)
7
8  f, axarr = plt.subplots(1)
9  # scatter polt of random x and y values
10 # colored by the difference between the pairs
11 scat = axarr.scatter(x,y, c=x-y, s=35, \
12                      label='random dots')
13 axarr.set_xlim([0,1])
14 axarr.set_ylim([0,1])
15 axarr.legend()
16 plt.show()
```

This is to be able to make the same figure for you each time

Two numpy arrays of random values [0,1]

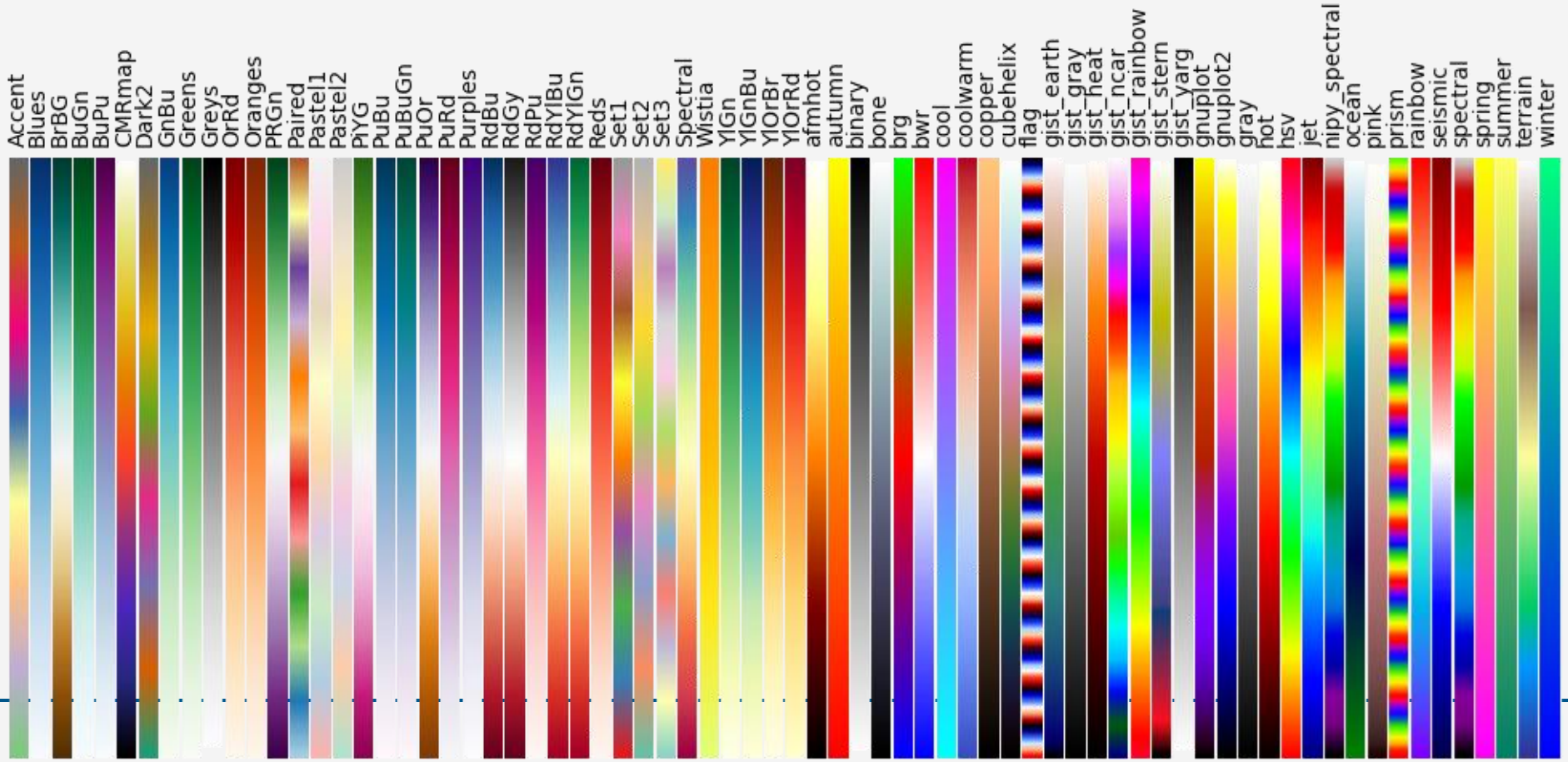
Colored by difference
(this has no meaning here)



Legend not meaningful

Building a colormap legend (2)

Remember from last week:



Building a colormap legend (3)

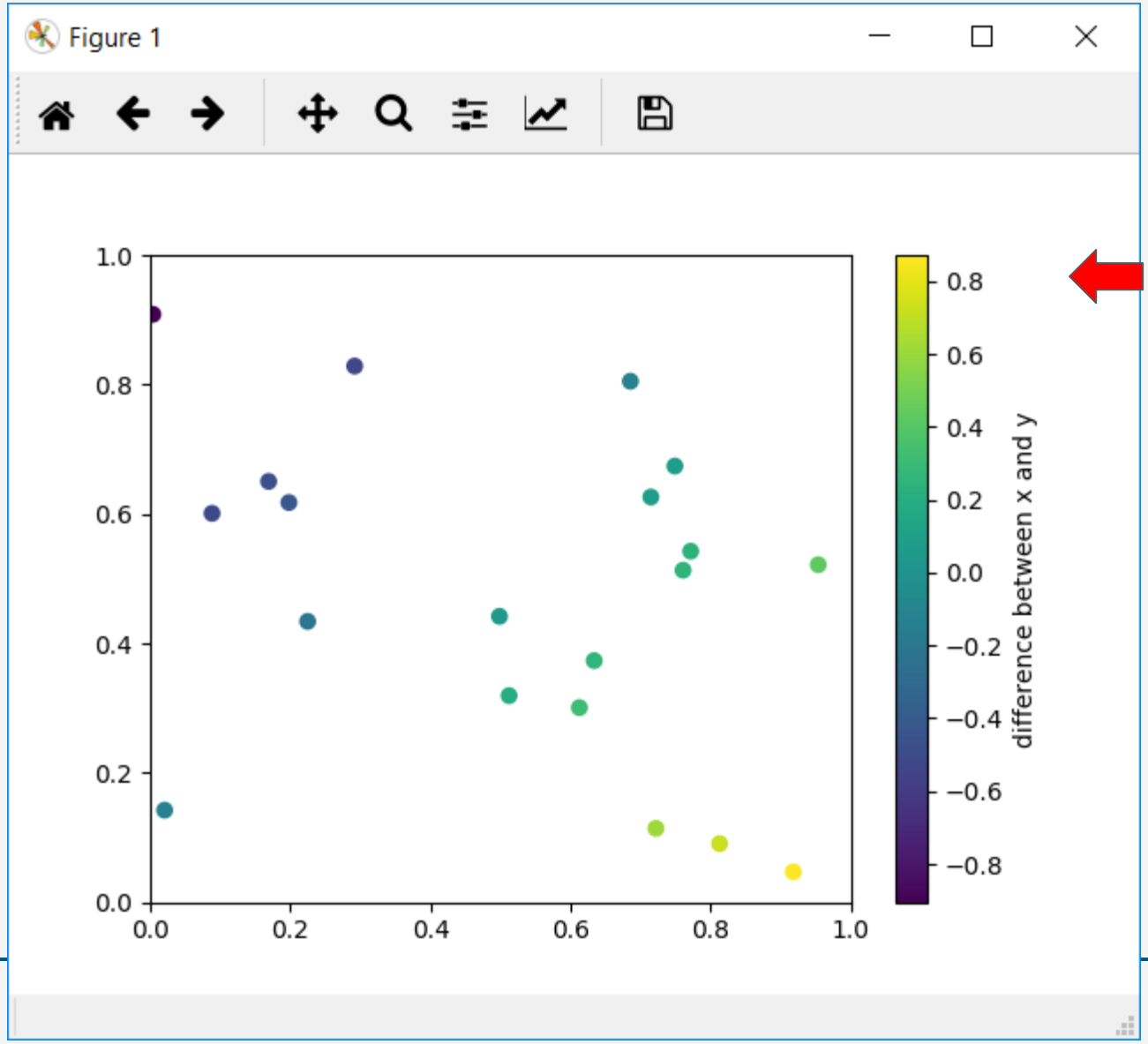
mpl6.py

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  np.random.seed(seed=10)
5  x = np.random.random(20)
6  y = np.random.random(20)
7
8  f, axarr = plt.subplots(1)
9  # scatter polt of random x and y values
10 # colored by the difference between the pairs
11 scat = axarr.scatter(x,y, c=x-y, s=35, \
12                       label='random dots')
13 axarr.set_xlim([0,1])
14 axarr.set_ylim([0,1])
15 # colormap legend
16 cb = plt.colorbar(scat, spacing='proportional')
17 cb.set_label('difference between x and y')
18 plt.show()
19
```

Catch the scatter dots into an object

Create a color ramp using this object

Label it



What is the theoretical maximum?

Building a colormap legend (4)

mpl6.py

```
1  from matplotlib import pyplot as plt
2  from matplotlib import colors as cls
3  import numpy as np
4
5  np.random.seed(seed=10)
6  x = np.random.random(20)
7  y = np.random.random(20)
8  total_range = cls.Normalize(vmin=-1.0, vmax=1.0)
9
10 f, axarr = plt.subplots(1)
11 # scatter polt of random x and y values
12 # colored by the difference between the pairs
13 scat = axarr.scatter(x,y, c=x-y, s=35, \
14                      label='random dots', \
15                      norm=total_range)
16 axarr.set_xlim([0,1])
17 axarr.set_ylim([0,1])
18 # colormap legend normalized
19 cb = plt.colorbar(scat, spacing='proportional')
20 cb.set_label('difference between x and y')
21 plt.show()
```

Loading the colors library

Maxima of differences in theory

Give the range object to the scatter method, colors are adapted

Colormap methods same as before

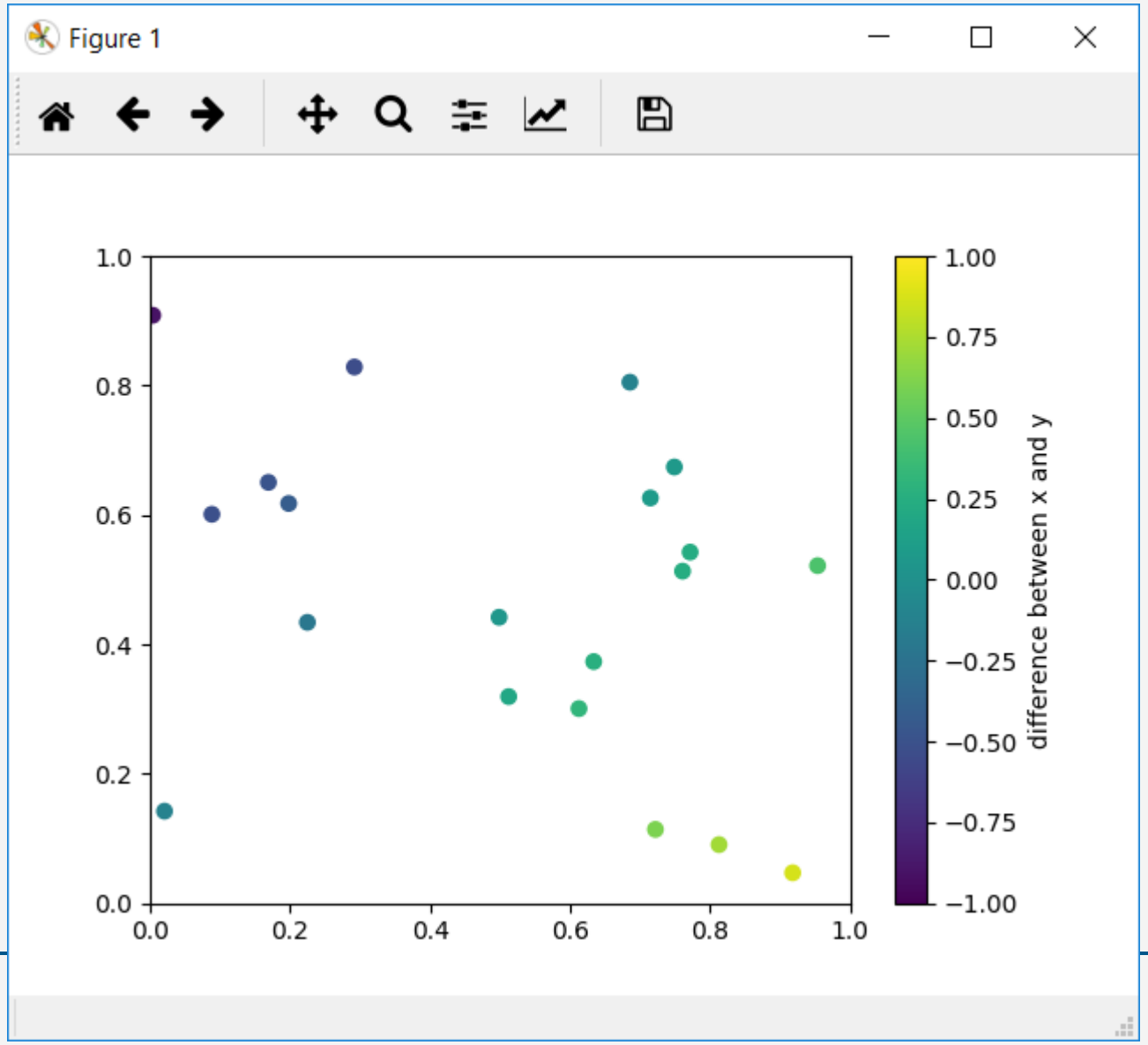


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
difference between x and y

Organizing subplots on the canvas

subplots_adjust(left=**None**, bottom=**None**, right=**None**, top=**None**, wspace=**None**, hspace=**None**)

- *left* = 0.125: the left side of the subplots of the figure
- *right* = 0.9: the right side of the subplots of the figure
- *bottom* = 0.1: the bottom of the subplots of the figure
- *top* = 0.9: the top of the subplots of the figure
- *wspace* = 0.2: the amount of width reserved for blank space between subplots, expressed as a fraction of the average axis width
- *hspace* = 0.2: the amount of height reserved for white space between subplots, expressed as a fraction of the average axis height

Called on the figure (f.subplots_adjust(….))

Saving a plot to disk (1)

savefig(fname, dpi=**None**, facecolor='w', edgecolor='w', orientation='portrait', papertype=**None**, format=**None**, transparent=**False**, bbox_inches=**None**, pad_inches=0.1, frameon=**None**)

- *fname*: A string containing a path to a filename

Most important arguments:

- *dpi*: The resolution in dots per inch
- *orientation*: [‘landscape’ | ‘portrait’], not supported on all backends
- *format*: One of the file extensions supported by the active backend. Most backends support png, pdf, ps, eps and svg

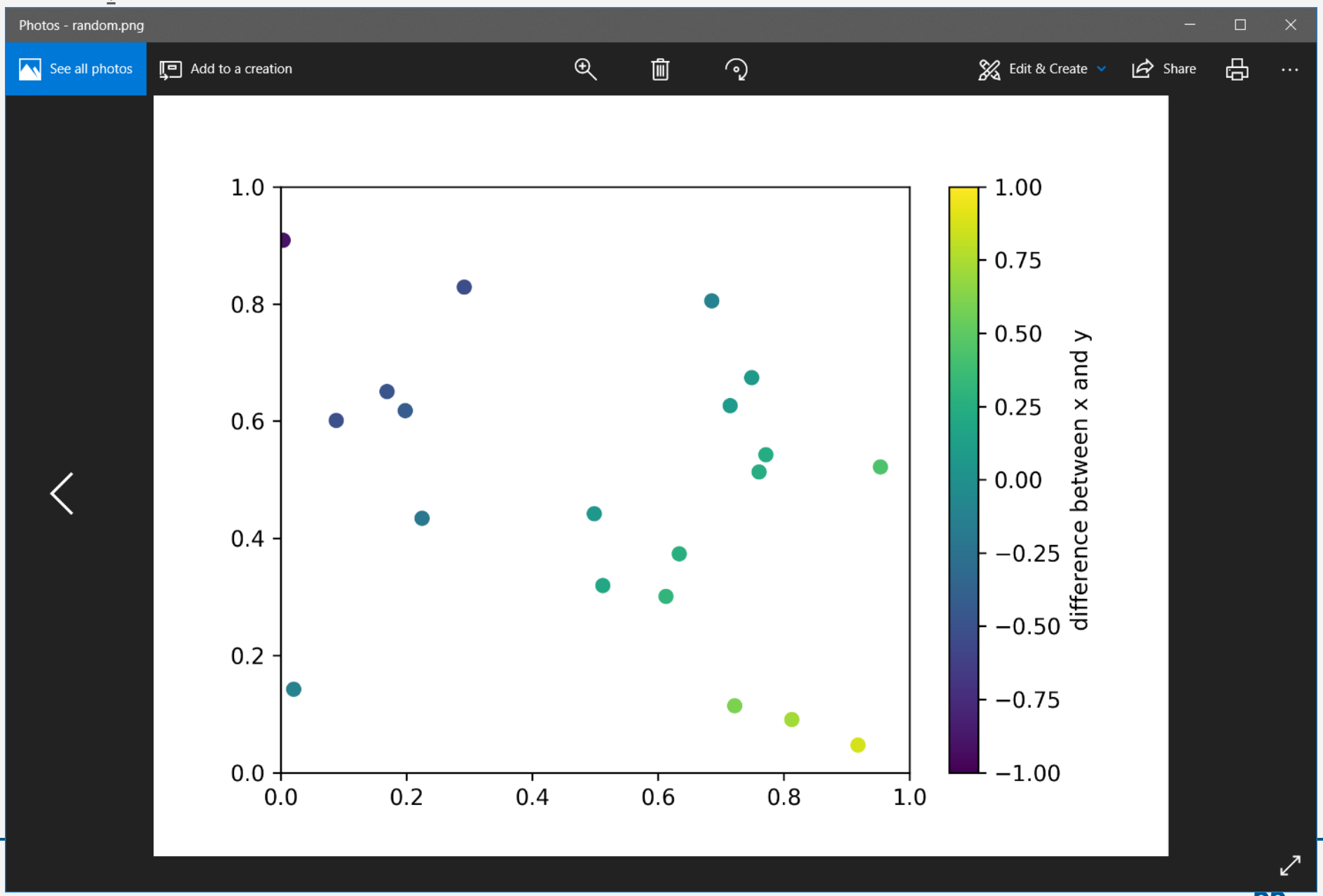
Called on the pyplot lib (plt.savefig(…))

Saving a plot to disk (2)

mpl6.py

```
13  scat = axarr.scatter(x,y, c=x-y, s=35, \
14                       label='random dots', \
15                       norm=total_range)
16  axarr.set_xlim([0,1])
17  axarr.set_ylim([0,1])
18  # colormap legend normalized
19  cb = plt.colorbar(scat, spacing='proportional')
20  cb.set_label('difference between x and y')
21  # Save the figure to disk
22  data_dir = os.path.join('C:\\', 'Users', 'verstege', \
23  'Documents', 'education', 'python_in_GIS', '2017_2018',
24  'data')
25  filename = os.path.join(data_dir, 'random.png')
26  plt.savefig(filename, dpi=300, format='png')
27
```

Do not forget to include extension in the filename


Photos - random.png
See all photos
Add to a creation
Edit & Create
Share
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
difference between x and y

Plotting raster maps (1)

mpl7.py

```
1  import gdal
2  from matplotlib import pyplot as plt
3  import numpy as np
4  import os
5
6  # path to the raster
7  data_dir = os.path.join('C:\\', 'Users', 'verstege', \
8  'Documents', 'education', 'python_in_GIS', '2017_2018',
9  'data')
10 in_fn = os.path.join(data_dir, 'rasters', 'clipped_dem.tif')
11
12 # open the raster and read out the data in a numpy array
13 rast_data_source = gdal.Open(in_fn)
14 in_band = rast_data_source.GetRasterBand(1)
15 data = in_band.ReadAsArray()
```

Open the raster as learned in the gdal lecture

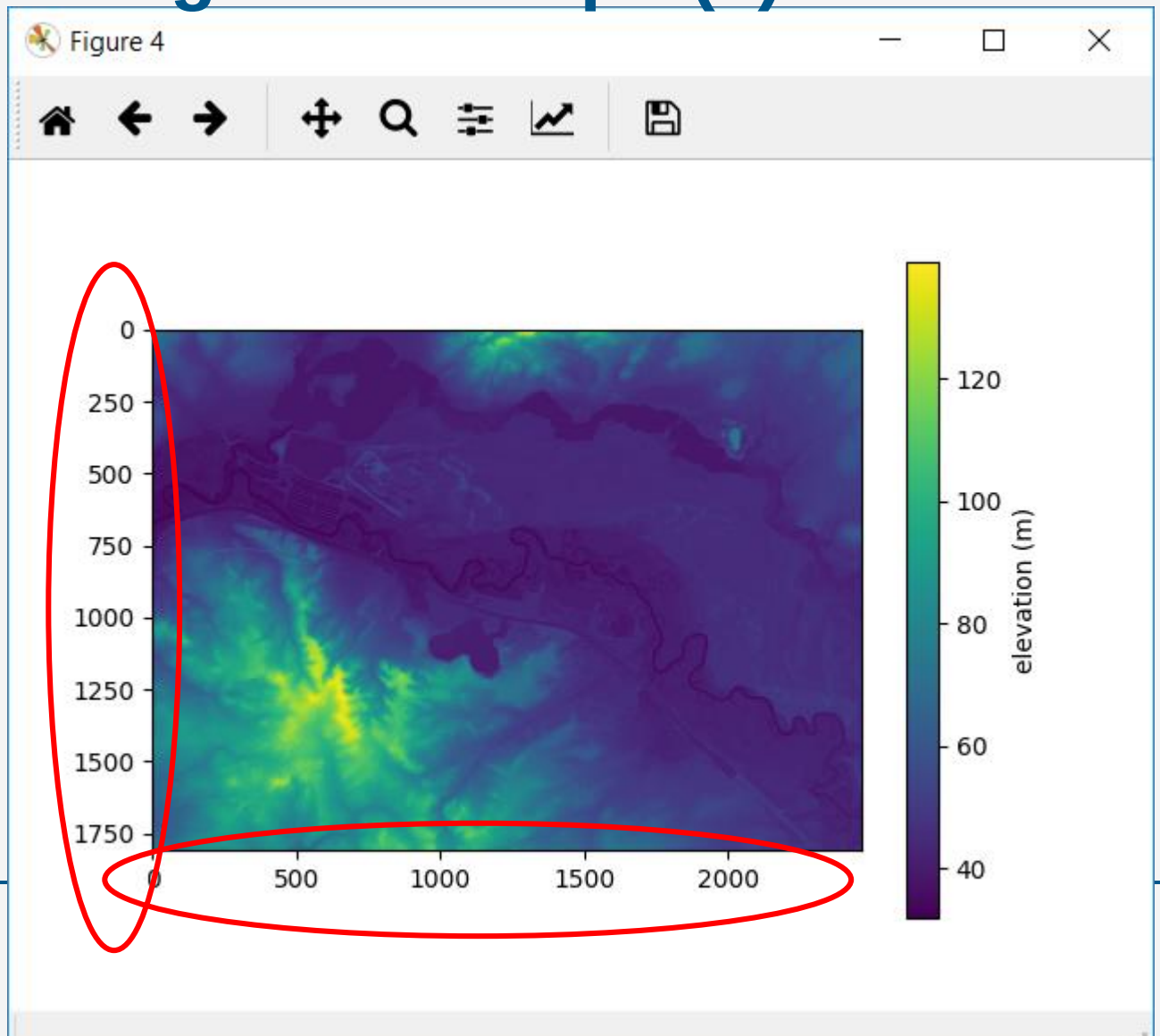
Plotting raster maps (2)

mpl7.py

```
25  # create the figure
26  f, axarr = plt.subplots(1)
27  # use imshow to plot the raster
28  im = axarr.imshow(data)
29  # add the legend
30  cb = plt.colorbar(im, spacing='proportional')
31  cb.set_label('elevation (m)')
32  plt.show()
```

imshow is the matplotlib function to plot pixel images, and thus raster maps

Plotting raster maps (2)



Plotting raster maps (3)

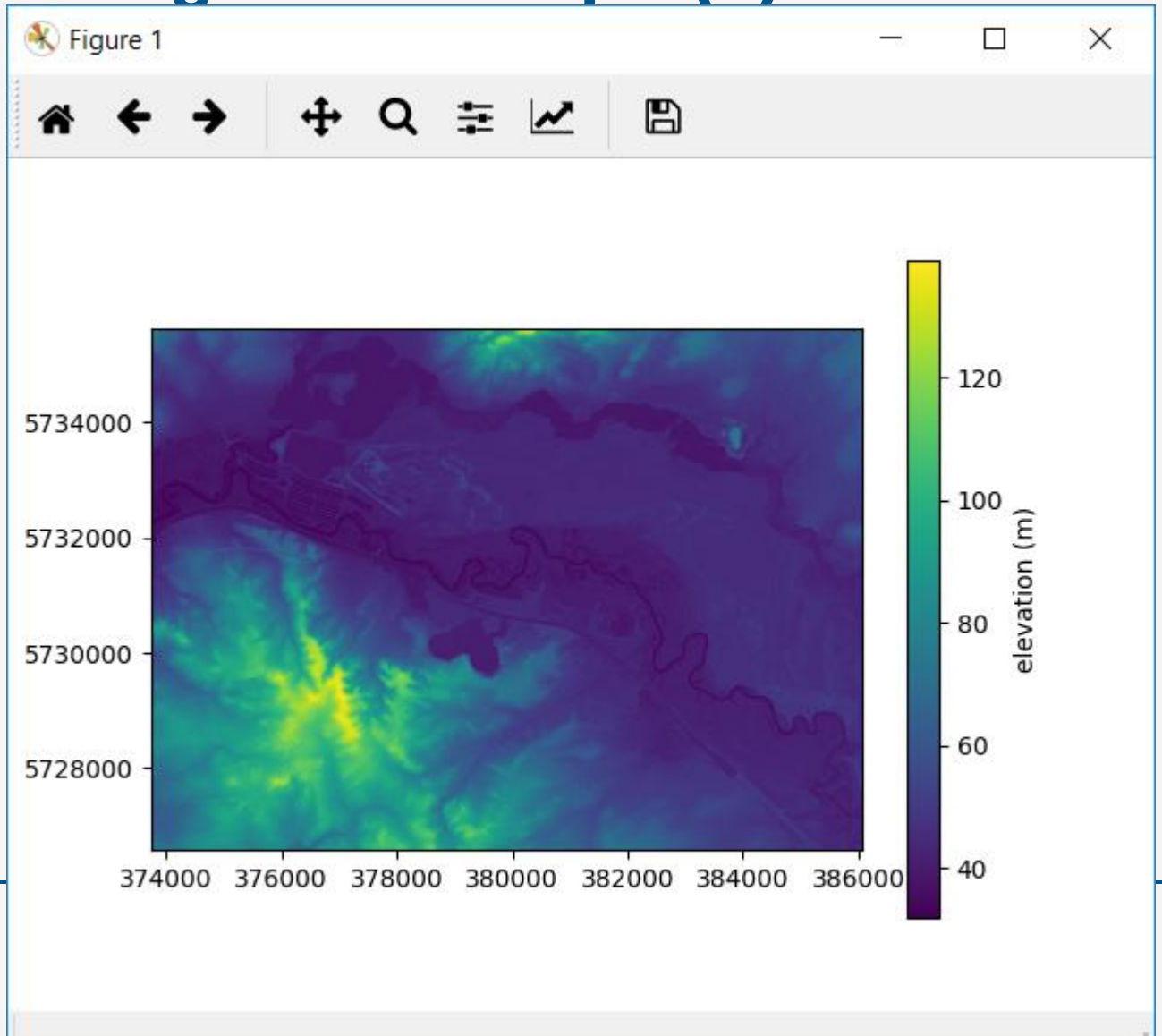
mpl7.py

```
16
17  geoTransform = rast_data_source.GetGeoTransform()
18  minx = geoTransform[0]
19  maxy = geoTransform[3]
20  maxx = minx + geoTransform[1] * \
21           rast_data_source.RasterXSize
22  miny = maxy + geoTransform[5] * \
23           rast_data_source.RasterYSize
24
25  # create the figure
26  f, axarr = plt.subplots(1)
27  # use imshow to plot the raster
28  im = axarr.imshow(data, extent=(minx, maxx, miny, maxy))
29  # add the legend
30  cb = plt.colorbar(im, spacing='proportional')
31  cb.set_label('elevation (m)')
32  plt.show()
33
```

After opening the raster, we calculate the raster extent, using gdal metadata for:
- coordinates of the origin
- nr of rows and columns
- raster cell size(s)

Pass extent to imshow

Plotting raster maps (3)



Exercise #3

In the data folder there is alse a raster clipped_dem.tif

Extend your previous script, such that:

- The clipped raster is in the background
- It includes a legend for both the raster and the shapefile
- There is enough space in the figure on all sides to see the axis labels (if you have this problem)
- It saves the figure to disk