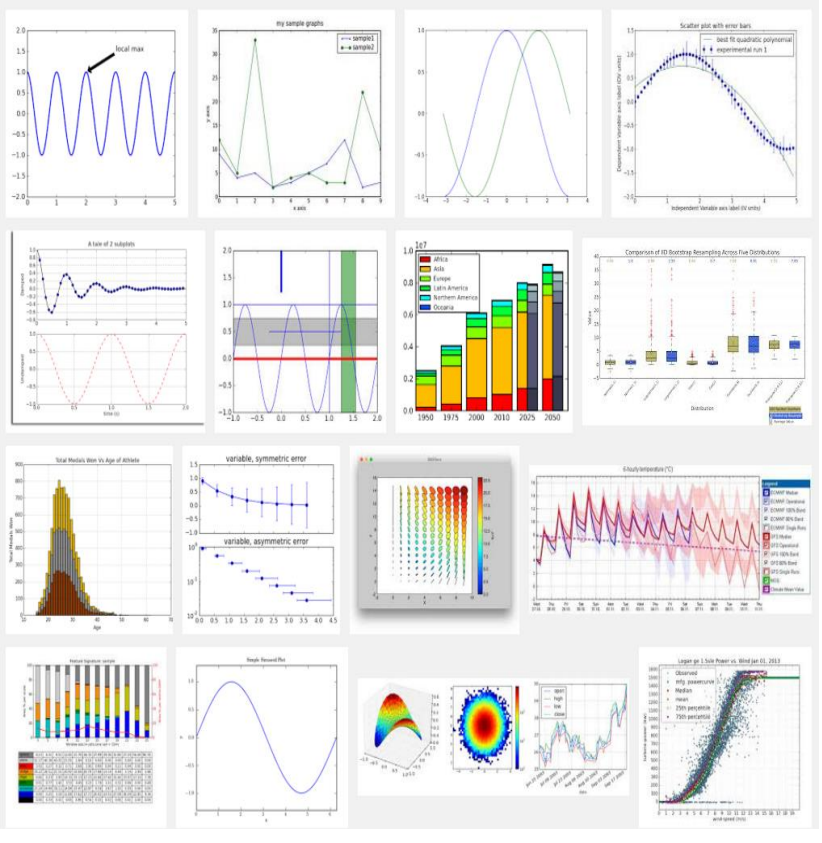
PFE 2

Dr Wojciech Palubicki

MATPLOTLIB


local max
my sample graphs
sample1
sample2
Scatter plot with error bars
best fit quadratic polynomial
experimental run 1
A tale of 2 subplots
Damped
Undamped
Africa
Asia
Europe
Latin America
Northern America
Oceania
1950
1975
2000
2010
2025
2050
Comparison of 3D Bootstrap Resampling Across Five Distributions
Value
Distribution
Total Medals Won Vs Age of Athlete
Total Medals Won
Age
variable, symmetric error
variable, asymmetric error
Hourly temperature (°C)
Observed
Median
mean
25th percentile
75th percentile

MATPLOTLIB

- Matplotlib is a package of python modules for data visualization
- It contains a module matplotlib.pyplot providing an interface the plotting library
- https://matplotlib.org/2.2.3/Matplotlib.pdf

Pyplot Example

```
1   import matplotlib.pyplot as plt
```

Pyplot Example

```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4]
4  y = [3, 7, 4.5, 8]
```

Pyplot Example

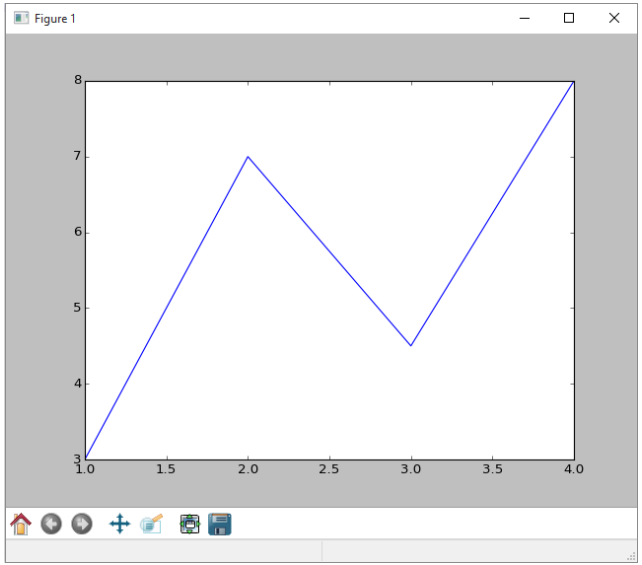
```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4]
4  y = [3, 7, 4.5, 8]
5
6  plt.plot(x, y)
```

Pyplot Example

```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4]
4  y = [3, 7, 4.5, 8]
5
6  plt.plot(x, y)
7
8  plt.show()
```

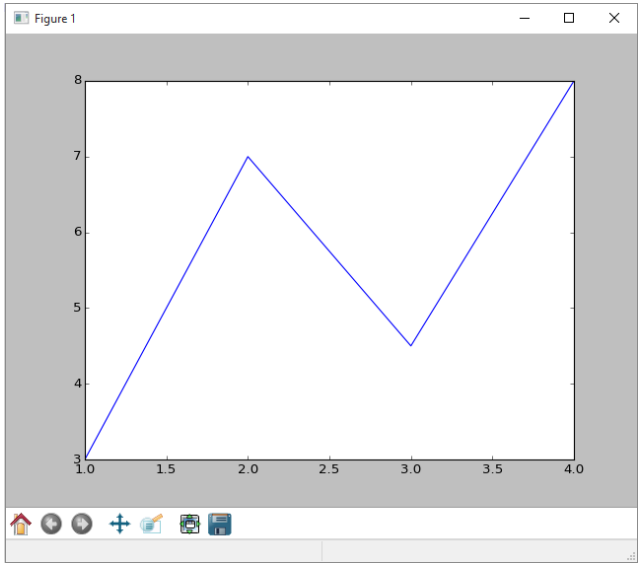
Pyplot Example

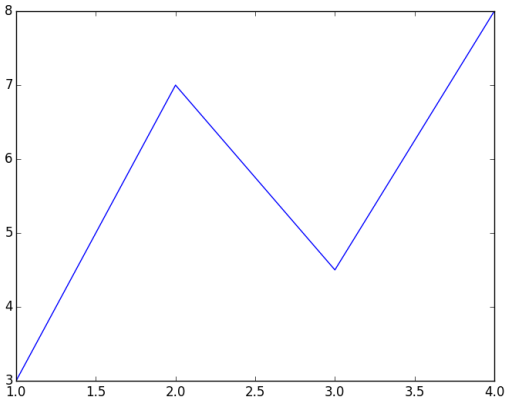
```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4]
4  y = [3, 7, 4.5, 8]
5
6  plt.plot(x, y)
7
8  plt.show()
```

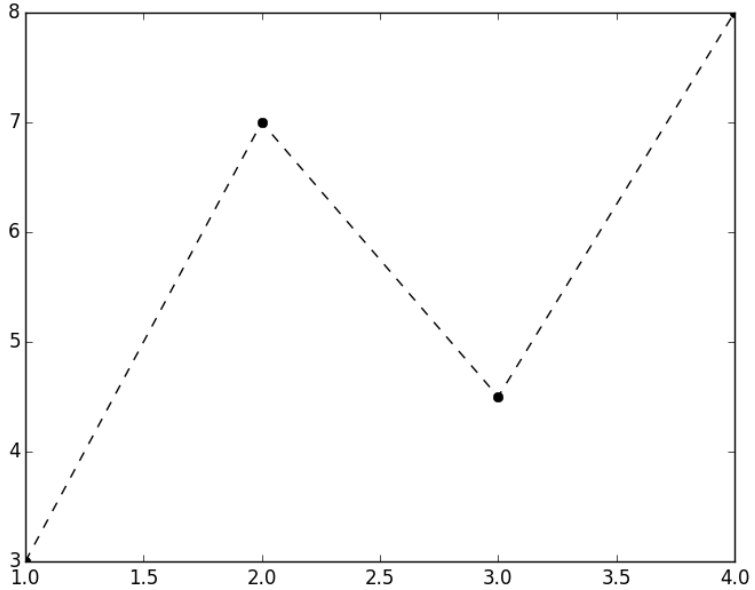


Pyplot Example

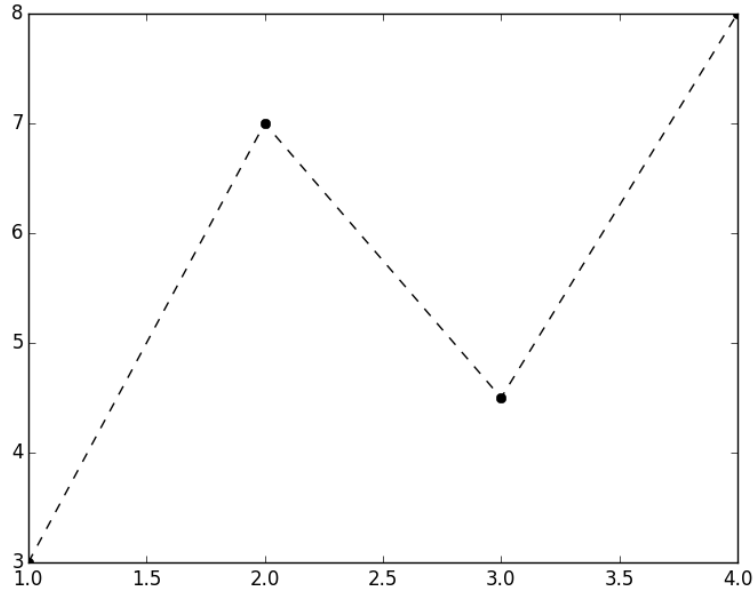
```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4]
4  y = [3, 7, 4.5, 8]
5
6  plt.plot(x, y)
7
8  plt.savefig('nazwa.pdf')
```








```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4]
4  y = [3, 7, 4.5, 8]
5
6  plt.plot(x, y, linestyle="dashed", marker="o", color="black")
7
8  plt.show()
```

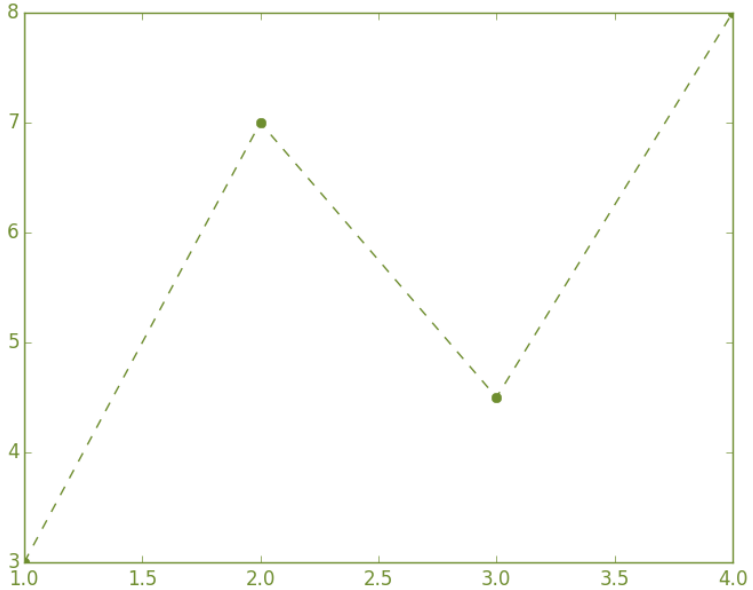



```
plt.plot(x, y, 'k--o')
```

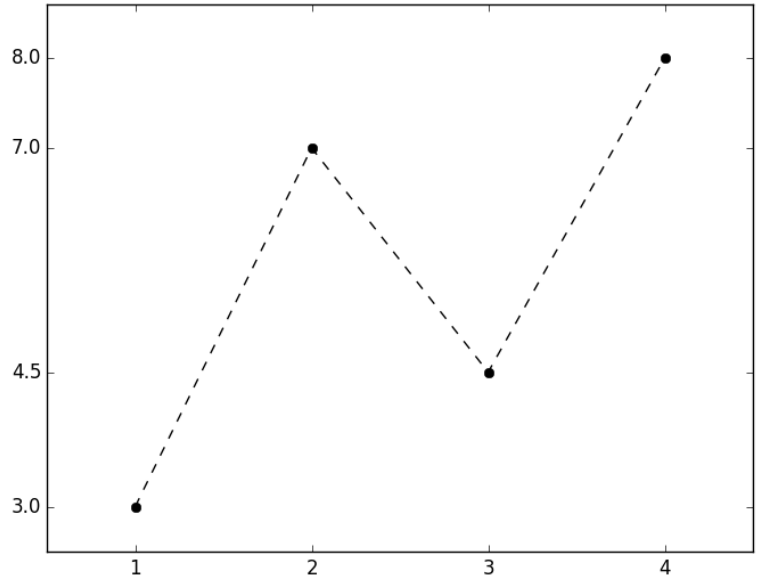
```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4]
4  y = [3, 7, 4.5, 8]
5
6  plt.plot(x, y, linestyle="dashed", marker="o", color="black")
7
8  plt.show()
```

plt.plot?

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

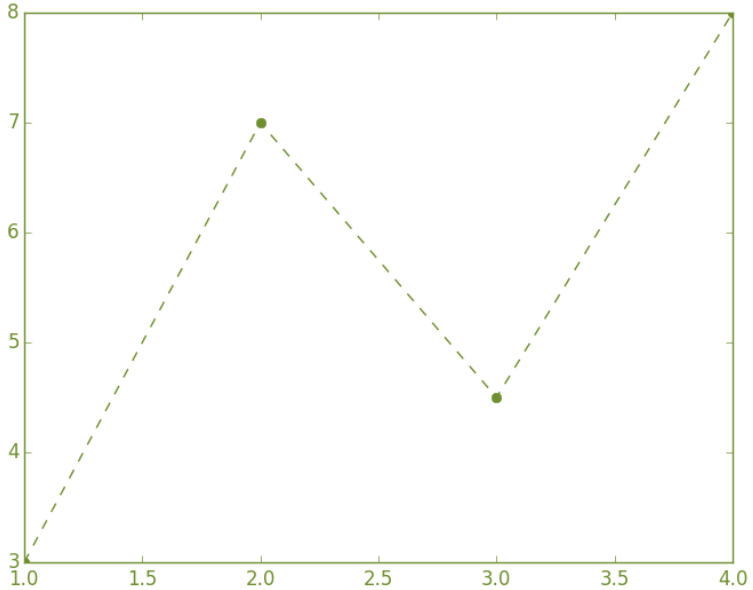


old

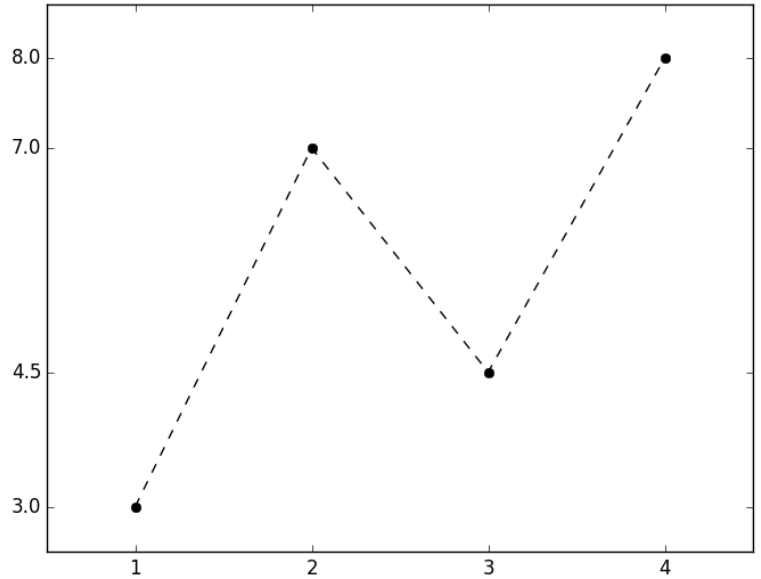


new

```
 8   plt.xlim(0.5,4.5)
 9   plt.xticks([1,2,3,4])
10
11   plt.ylim(2.5,8.6)
12   plt.yticks([3, 7, 4.5, 8])
```



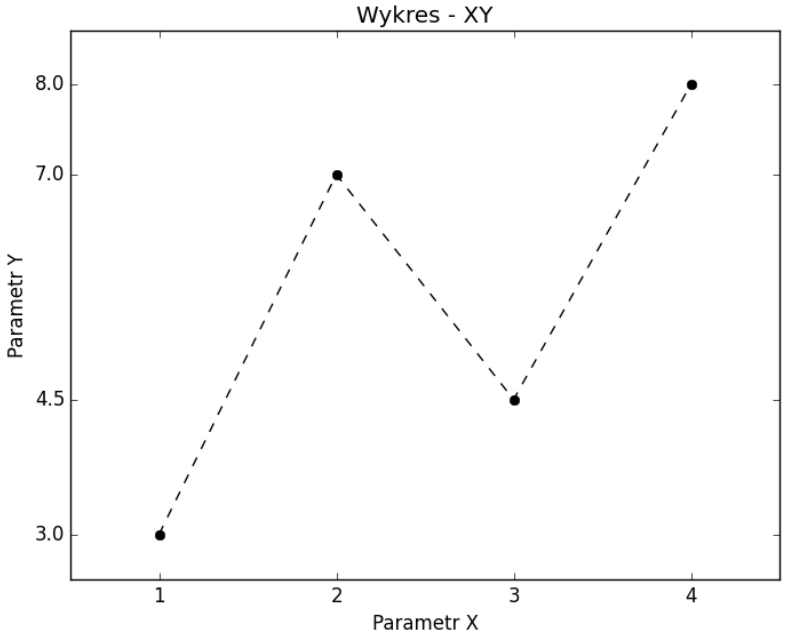
old



new

```
 8   plt.xlim(0.5,4.5)
 9   plt.xticks([1,2,3,4])
10
11   plt.ylim(2.5,8.6)
12   plt.yticks([3, 7, 4.5, 8])
```

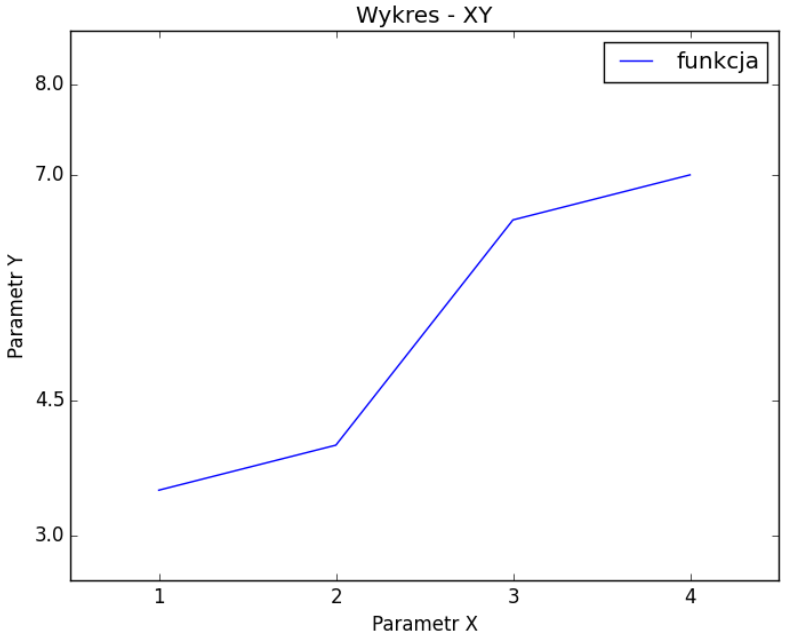
```
plt.axis([0,30,0,30])
```



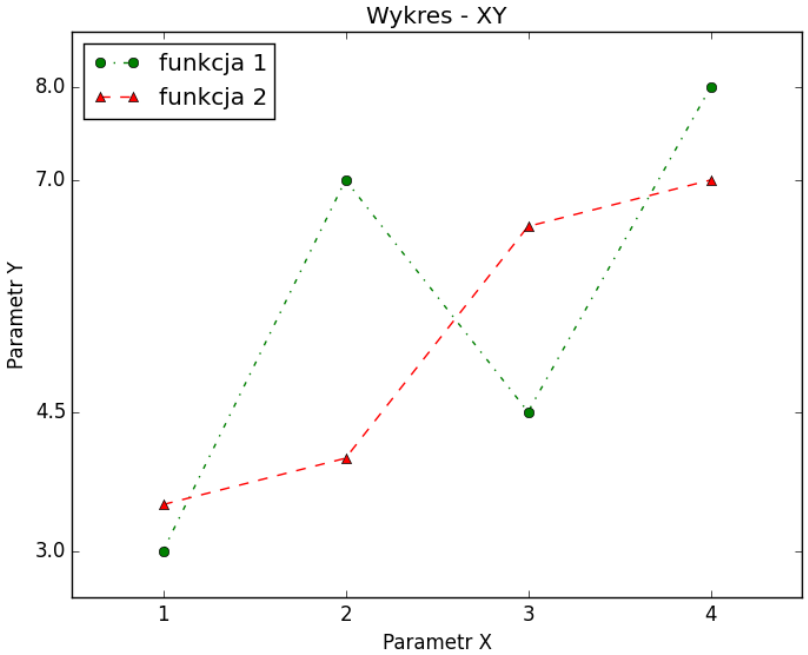

```
14    plt.xlabel("Parametr X")
15    plt.ylabel("Parametr Y")
16
17    plt.title("Wykres - XY")
```

Legend

```
19    plt.plot(x,y, label = 'funkcja')
20    plt.legend()
```



Example


Wykres - XY
funkcja 1
funkcja 2
8.0
7.0
4.5
3.0
Parametr Y
1
2
3
4
Parametr X

Exercise

- Reconstruct the graphs from the previous slides using your own list of data points. Save the graph as a .png file.

Matplotlib steps

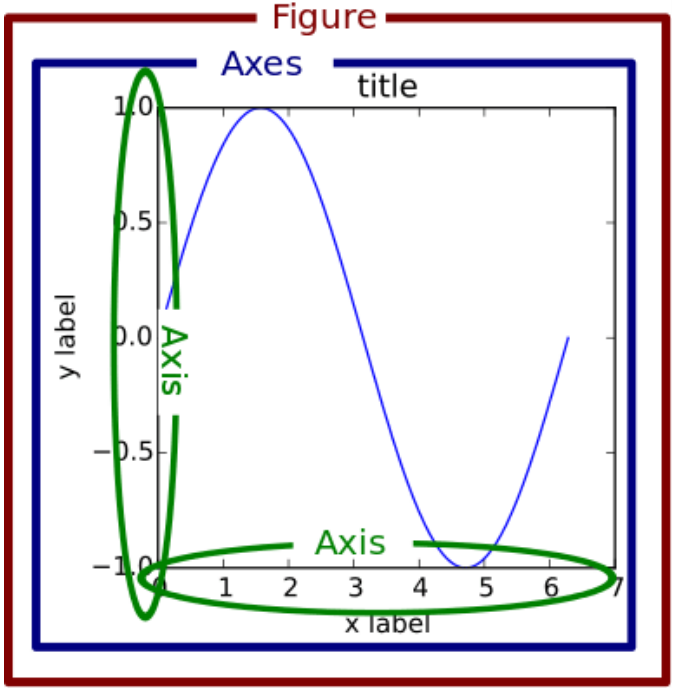
- Prepare data
- Create plot
- Choose plotting routine
- Customize plot
- Save plot
- Show plot

Pyplot vs. Object Oriented Matplotlib

```
plt.figure(1)

plt.subplot(111)

plt.plot(x, y, ‘k')
```

```
fig = plt.figure(1)

ax = fig.add_subplot(1,1,1)

ax.plot(x, y, ‘k')
```

Hierarchical Containers



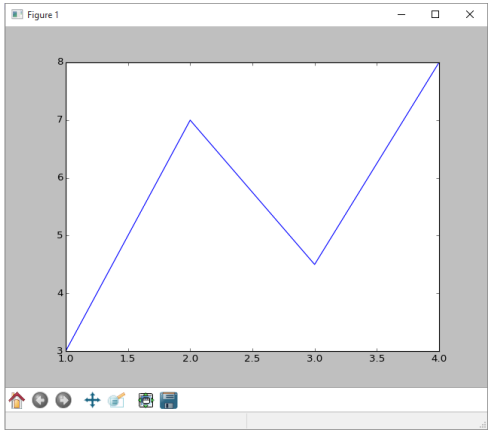
Hierarchies Example

```
fig, ax = plt.subplots() # make Figure and Axes which is
contained by fig

fig = plt.figure() # make Figure

ax = fig.add_subplot(1,1,1) # make Axes contained by fig
```

Figures

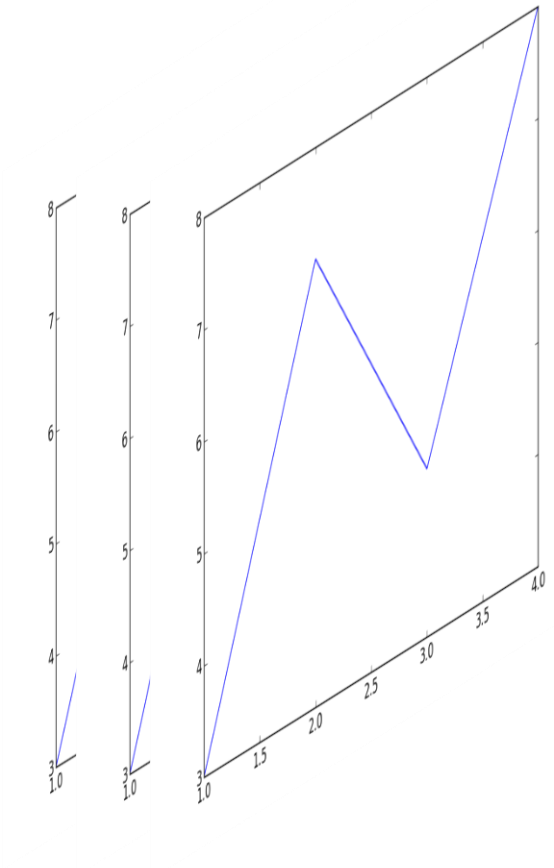
- Figures are indexed from 1: e.g. **figure** (num = 1) or **figure**(1)
- Figure coordinates are [0,1]x[0,1]

Argument	**Opis**
num	Figure number
figsize	Figure size
dpi	Figure resolution
facecolor	Background color



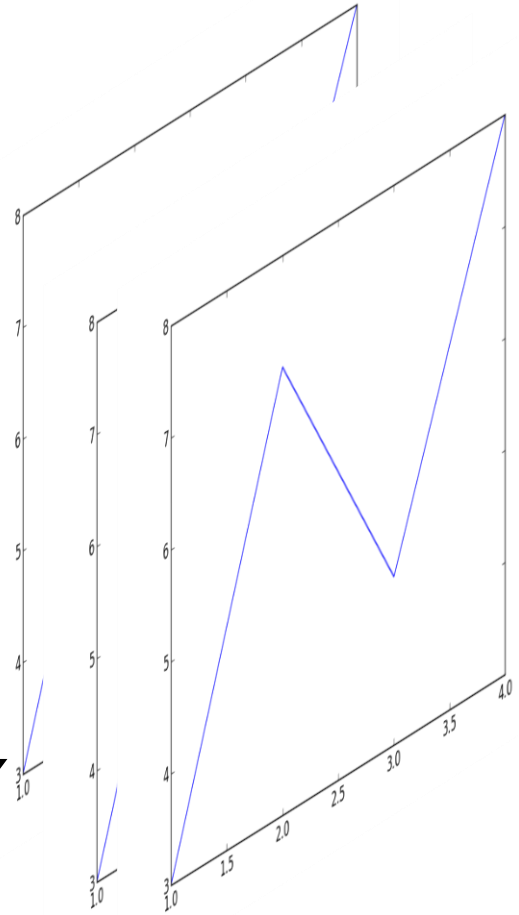
Figures

- Figure Buffer:



Figures

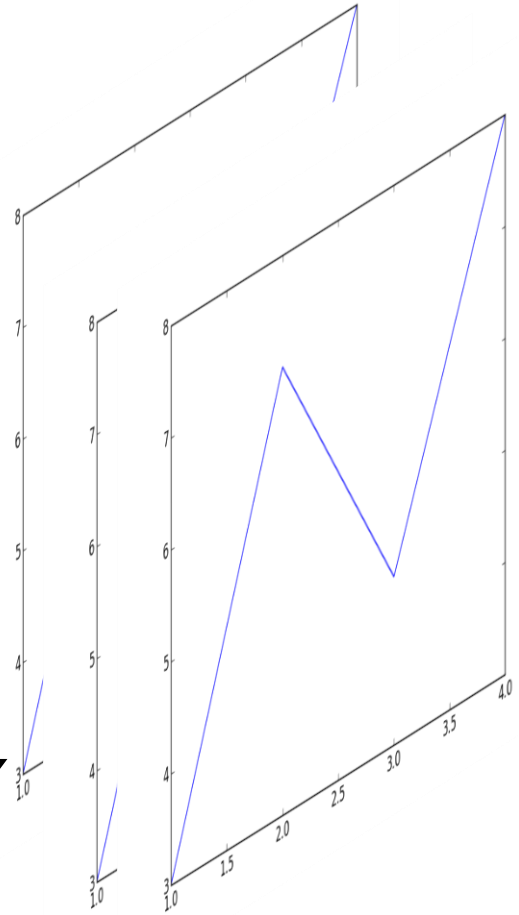
- Figure Buffer:



```
plt.figure(1)
```

Figures

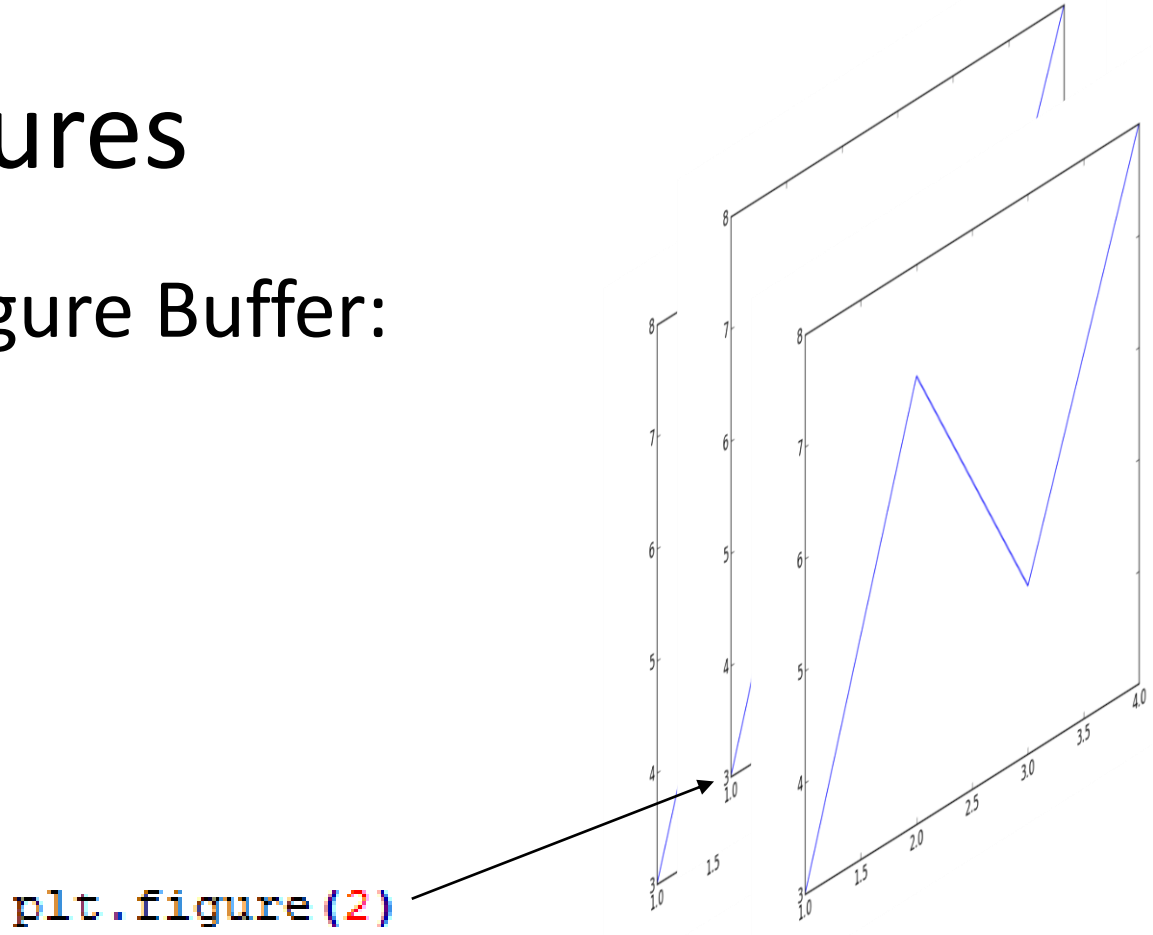
- Figure Buffer:




```
plt.figure(1)
plt.plot(x2, y2)
```

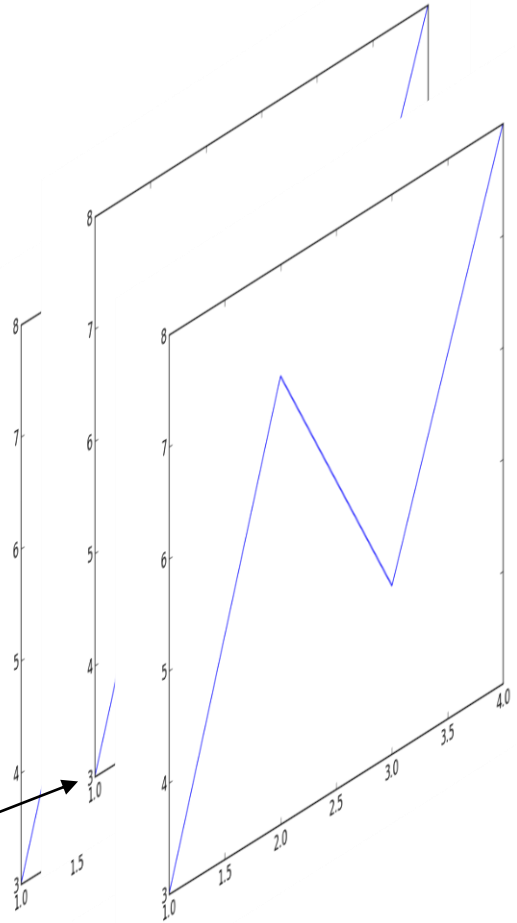
Figures

- Figure Buffer:



Figures

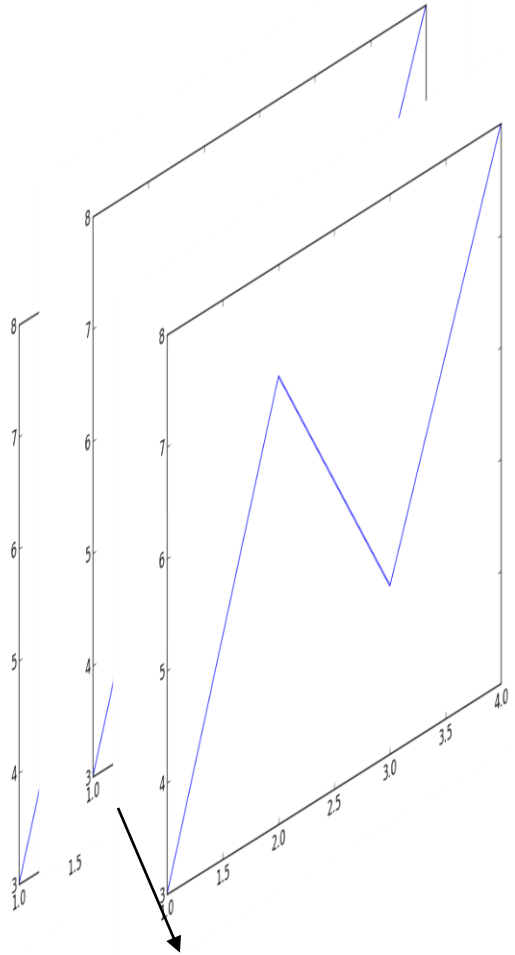
- Figure Buffer:



```
plt.figure(2)
plt.plot(x, y)
```

Figures

- Figure Buffer:



Reference to active figure: `plt.gcf()`

Figure Creation

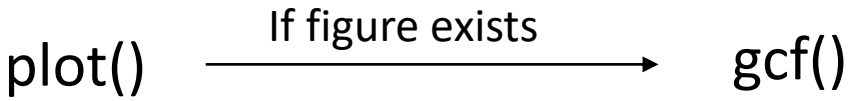
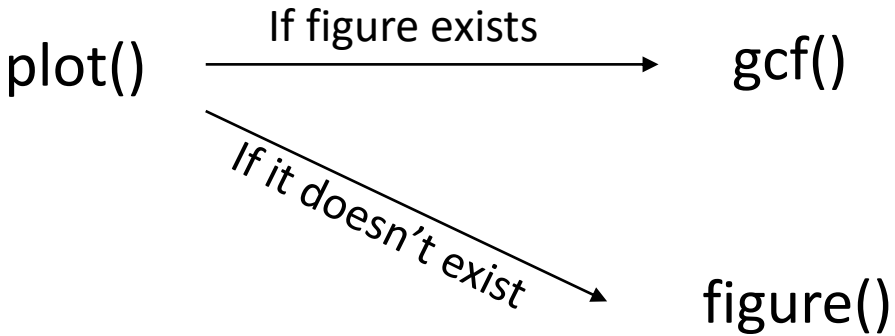
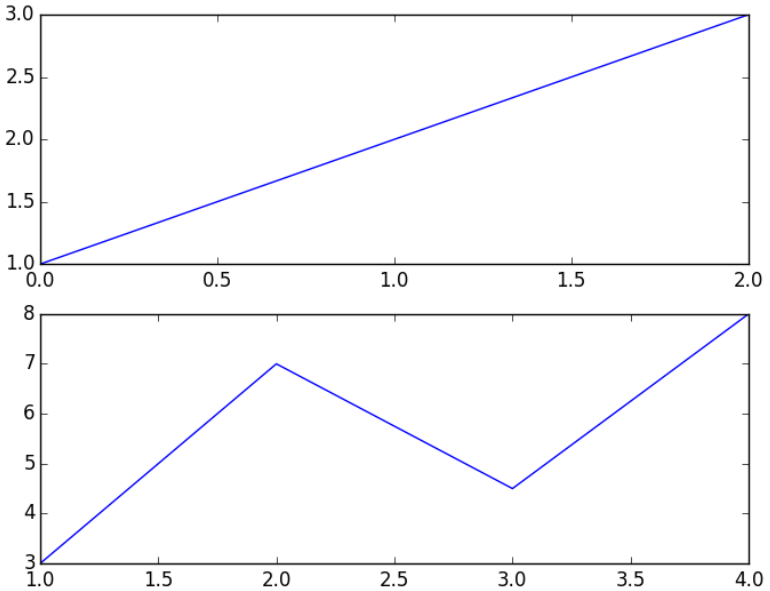


Figure Creation

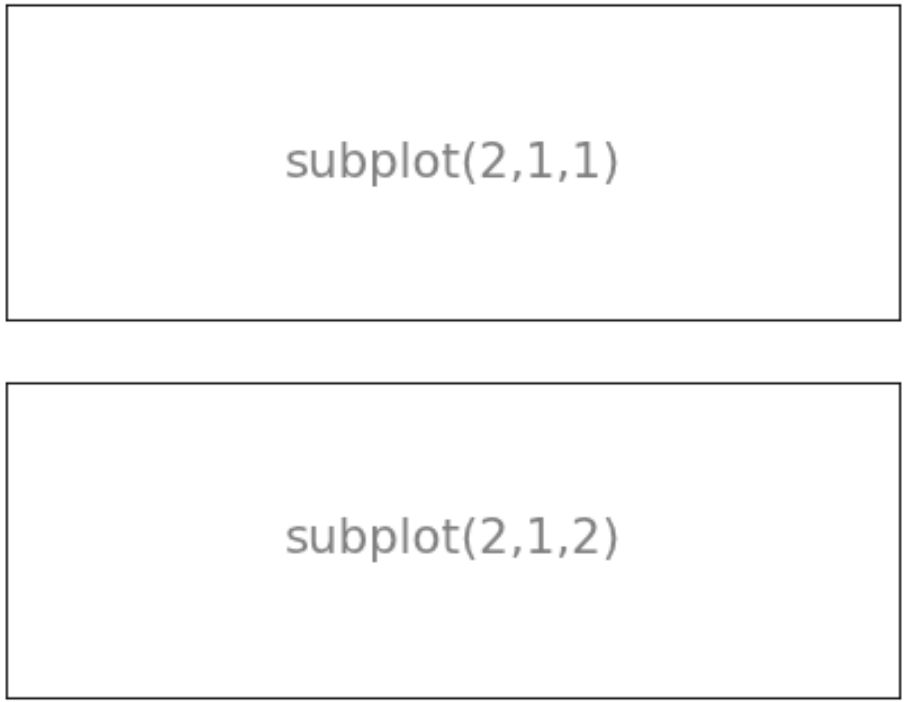


Subplots

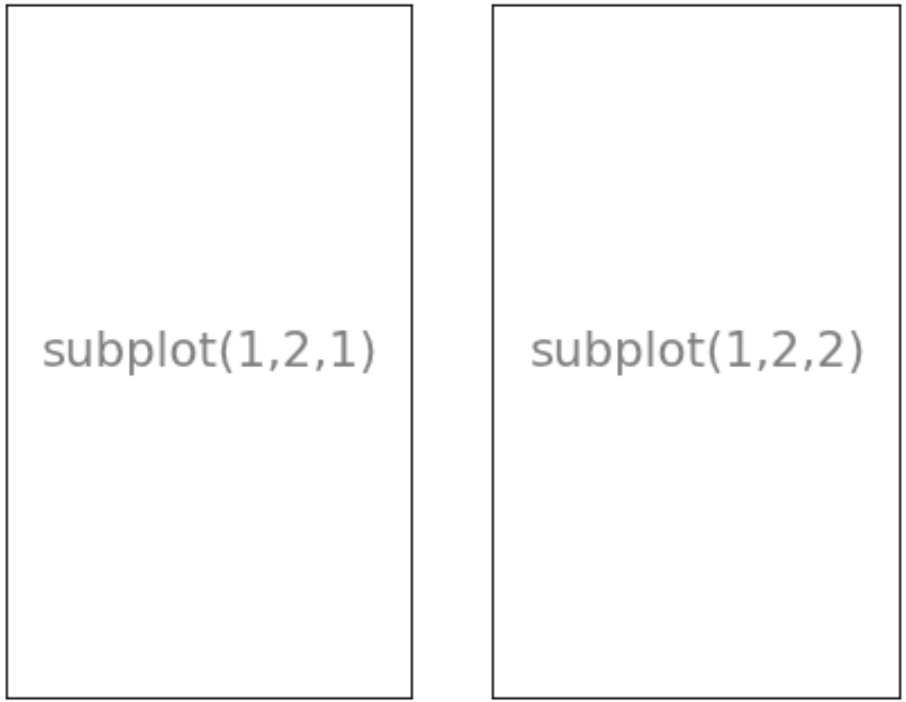
```
plt.figure(1)
plt.subplot(211)
plt.plot(...)
plt.subplot(212)
plt.plot(...)
```



Subplots



Subplots



Subplots

Pyplot is a state-machine interface

```
# matplotlib/pyplot.py
>>> def plot(*args, **kwargs):
...     """An abridged version of plt.plot()."""
...     ax = plt.gca()
...     return ax.plot(*args, **kwargs)

>>> def gca(**kwargs):
...     """Get the current Axes of the current Figure."""
...     return plt.gcf().gca(**kwargs)
```

Shell Verification

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1) # make a blank plotting area
print('fig.axes:', fig.axes)
print('ax.figure:', ax.figure)
print('ax.xaxis:', ax.xaxis)
print('ax.yaxis:', ax.yaxis)
print('ax.xaxis.axes:', ax.xaxis.axes)
print('ax.yaxis.axes:', ax.yaxis.axes)
print('ax.xaxis.figure:', ax.xaxis.figure)
print('ax.yaxis.figure:', ax.yaxis.figure)
print('fig.xaxis:', fig.xaxis)
```

Output

```
fig.axes: [<matplotlib.axes._subplots.AxesSubplot object at 0x00000215DB1E6320>]
ax.figure: Figure(640x480)
ax.xaxis: XAxis(80.000000,52.800000)
ax.yaxis: YAxis(80.000000,52.800000)
ax.xaxis.axes: AxesSubplot(0.125,0.11;0.775x0.77)
ax.yaxis.axes: AxesSubplot(0.125,0.11;0.775x0.77)
ax.xaxis.figure: Figure(640x480)
ax.yaxis.figure: Figure(640x480)
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-9-25857b24c47a> in <module>()
      9 print('ax.xaxis.figure:', ax.xaxis.figure)
     10 print('ax.yaxis.figure:', ax.yaxis.figure)
---> 11 print('fig.xaxis:', fig.xaxis)
     12

AttributeError: 'Figure' object has no attribute 'xaxis'
```

Container Relation Rules

- Figure references Axes but not Axis class
- Axes references both Figure and Axis classes
- Axis references both Axes and Figure classes
- Figure can contain multiple Axes objects because fig.axes is a list of Axes objects
- Axes can be contained by only a single Figure because ax.figure is not a list
- Axes can contain only one XAxis and one YAxis object
- XAxis and YAxis can be contained by only a single Axes and Figure object

Matplotlib API Layers

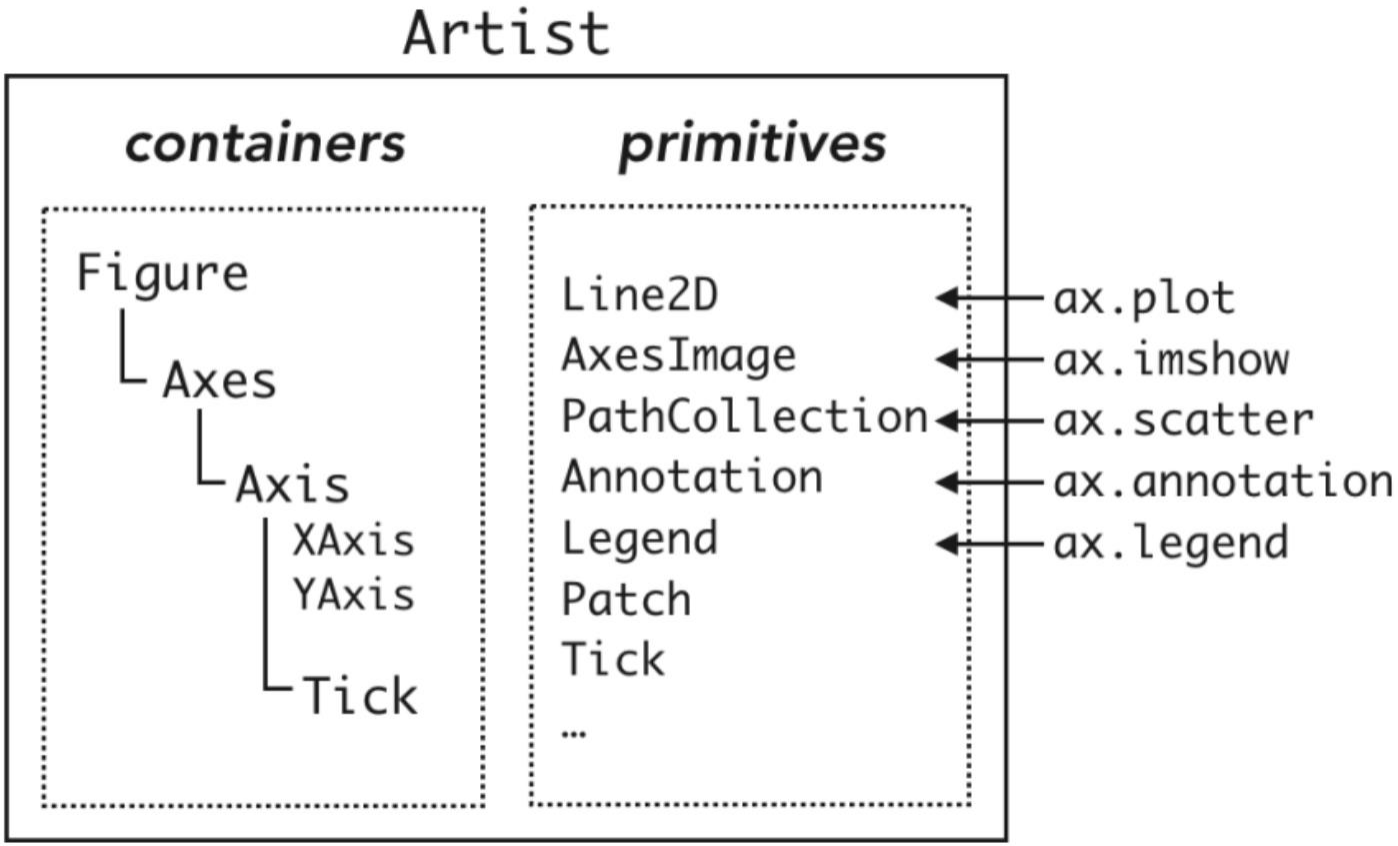
- the **matplotlib.backend_bases.FigureCanvas** is the area onto which the figure is drawn
- the **matplotlib.backend_bases.Renderer** is the object which specifies how to draw on the FigureCanvas
- and the **matplotlib.artist.Artist** is the object that specifies how to use a renderer to render onto the canvas.

Renderer

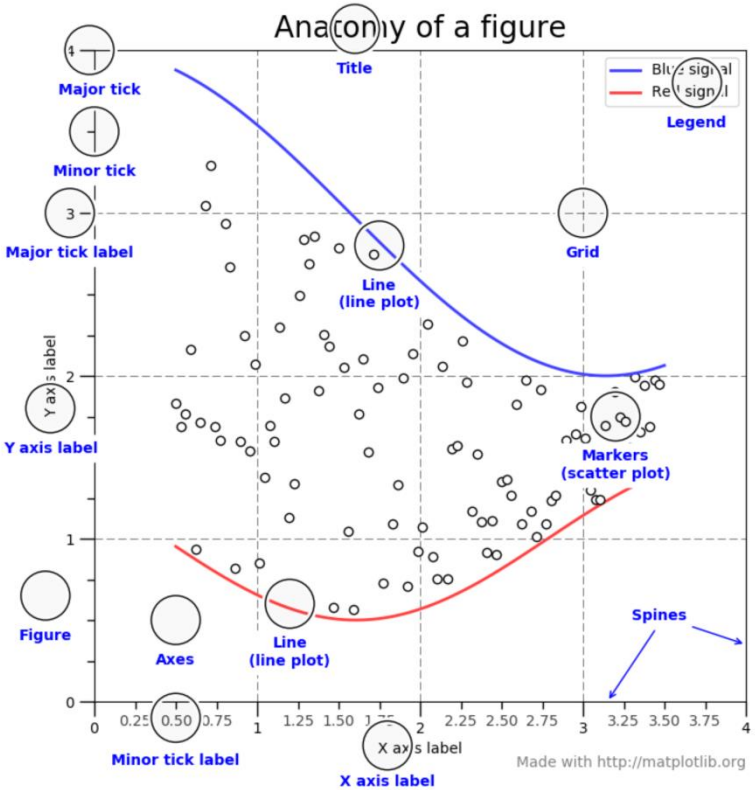
- An abstract base class to handle drawing/rendering operations.
- The following methods must be implemented in the backend for full functionality:
  - draw_path()
  - draw_image()
  - draw_gouraud_triangle()
- The following methods *should* be implemented in the backend for optimization reasons:
  - draw_text()
  - draw_markers()
  - draw_path_collection()
  - draw_quad_mesh()

Artist Objects

- There are two types of objects: **primitives** and **containers**
- Primitives represent standard graphical objects (Line2D, Rectangle, Text, AxesImage, etc.)
- Containers are data structures to store them in (Axis, Axes and Figure)


Artist
containers
primitives
Figure
Axes
Axis
XAxis
YAxis
Tick
Line2D
AxesImage
PathCollection
Annotation
Legend
Patch
Tick
…
ax.plot
ax.imshow
ax.scatter
ax.annotation
ax.legend

Artist Objects



https://matplotlib.org/examples/showcase/anatomy.html

Artist Objects

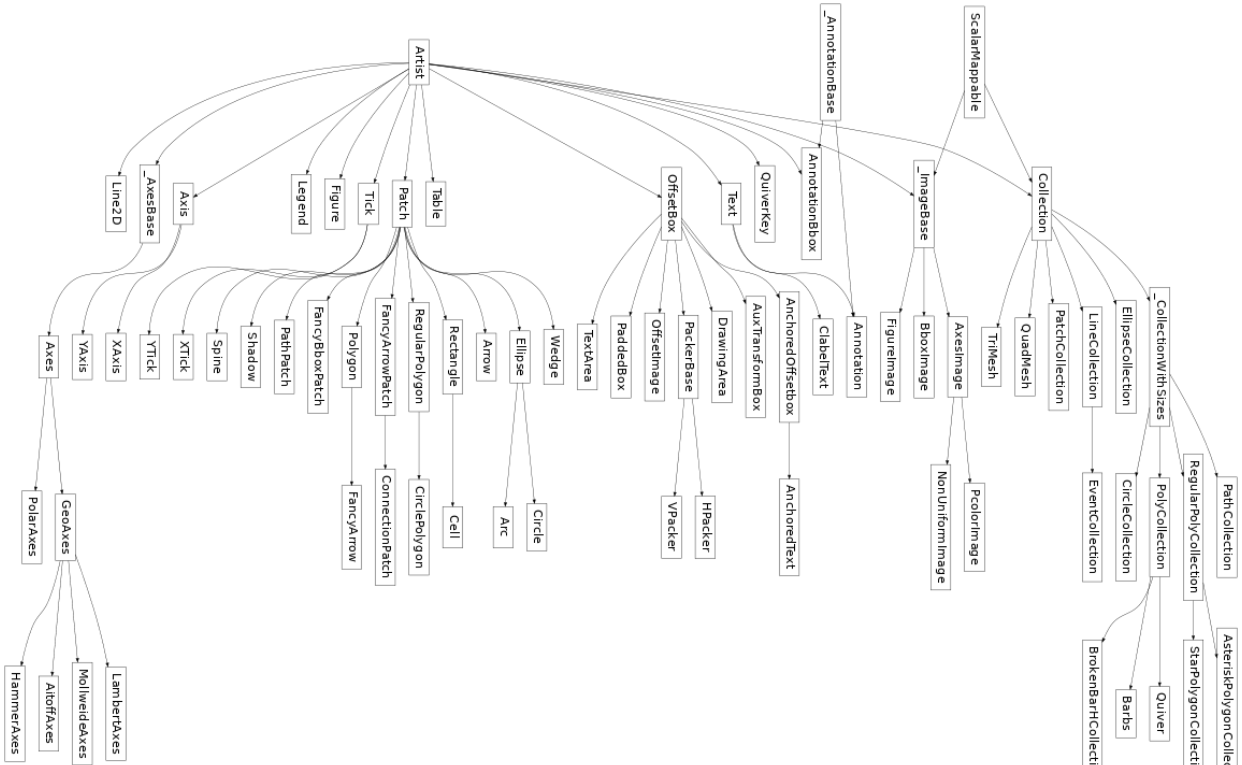


Figure attributes

Figure attribute	Description
axes	A list of Axes instances (includes Subplot)
patch	The Rectangle background
images	A list of FigureImages patches - useful for raw pixel display
legends	A list of Figure Legend instances (different from Axes.legends)
lines	A list of Figure Line2D instances (rarely used, see Axes.lines)
patches	A list of Figure patches (rarely used, see Axes.patches)
texts	A list Figure Text instances

Axes attributes

Axes attribute	Description
artists	A list of Artist instances
patch	Rectangle instance for Axes background
collections	A list of Collection instances
images	A list of AxesImage
legends	A list of Legend instances
lines	A list of Line2D instances
patches	A list of Patch instances
texts	A list of Text instances
xaxis	matplotlib.axis.XAxis instance
yaxis	matplotlib.axis.YAxis instance

```
...: fig = plt.figure()
...: ax = fig.add_subplot(1,1,1)
...: print('ax.lines before plot:\n', ax.lines) # empty
...: line1, = ax.plot(x, np.sin(x), label='1st plot') # add Line2D in ax.lines
...: print('ax.lines after 1st plot:\n', ax.lines)
...: line2, = ax.plot(x, np.sin(x+np.pi/8), label='2nd plot') # add another Line2D
...: print('ax.lines after 2nd plot:\n', ax.lines)
...: ax.legend()
...: print('line1:', line1)
...: print('line2:', line2)
```

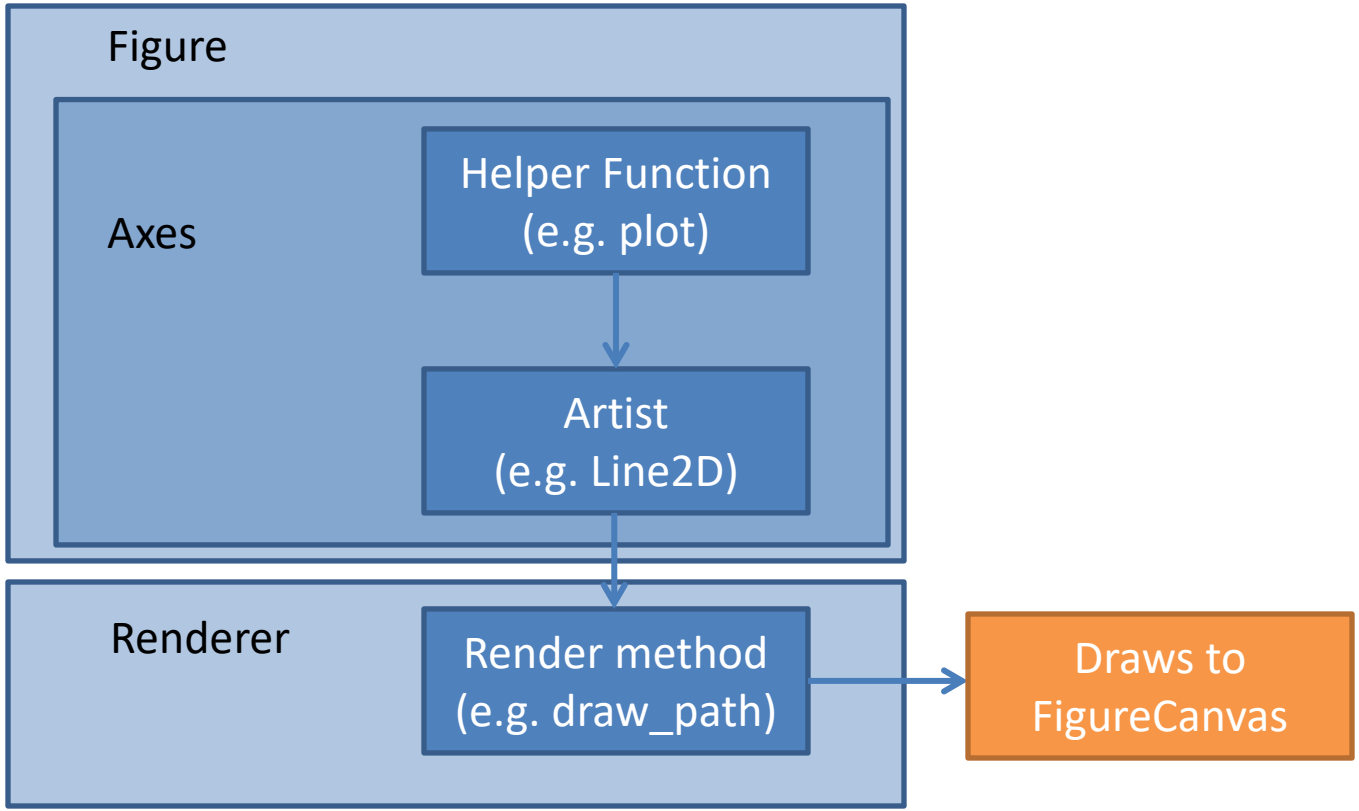
```
...: fig = plt.figure()
...: ax = fig.add_subplot(1,1,1)
...: print('ax.lines before plot:\n', ax.lines) # empty
...: line1, = ax.plot(x, np.sin(x), label='1st plot') # add Line2D in ax.lines
...: print('ax.lines after 1st plot:\n', ax.lines)
...: line2, = ax.plot(x, np.sin(x+np.pi/8), label='2nd plot') # add another Line2D
...: print('ax.lines after 2nd plot:\n', ax.lines)
...: ax.legend()
...: print('line1:', line1)
...: print('line2:', line2)
```

```
ax.lines before plot:
 []
ax.lines after 1st plot:
 [<matplotlib.lines.Line2D object at 0x00000215DB222AC8>]
ax.lines after 2nd plot:
 [<matplotlib.lines.Line2D object at 0x00000215DB222AC8>, <matplotlib.lines.Line2D object at
0x00000215DB222CF8>]
line1: Line2D(1st plot)
line2: Line2D(2nd plot)
```

Brief Summary of Axes Methods

Helper method	Artist	Container
ax.annotate - text annotations	Annotate	ax.texts
ax.bar - bar charts	Rectangle	ax.patches
ax.errorbar - error bar plots	Line2D and Rectangle	ax.lines and ax.patches
ax.fill - shared area	Polygon	ax.patches
ax.hist - histograms	Rectangle	ax.patches
ax.imshow - image data	AxesImage	ax.images
ax.legend - axes legends	Legend	ax.legends
ax.plot - xy plots	Line2D	ax.lines
ax.scatter - scatter charts	PolygonCollection	ax.collections
ax.text - text	Text	ax.texts

Matplotlib Process



Axis Containers

- The matplotlib.axis.Axis instances handle the drawing of the tick lines, the grid lines, the tick labels and the axis label.

Example

```
fig, ax = plt.subplots()
axis = ax.xaxis
axis.get_ticklocs()
```

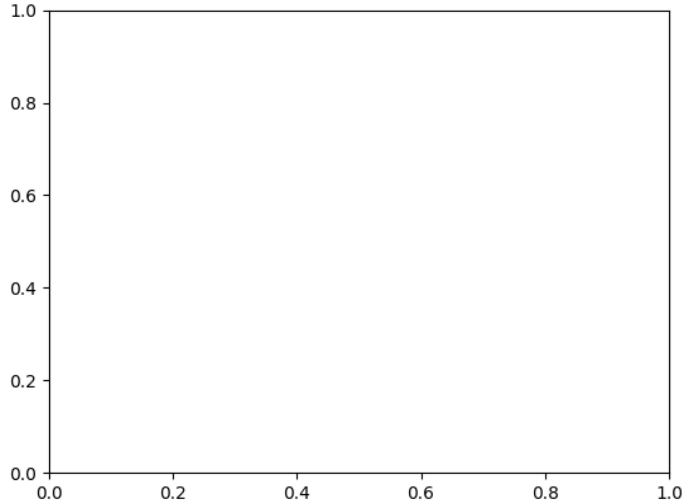
Out: array([0. , 0.2, 0.4, 0.6, 0.8, 1.])

```
axis.get_ticklabels()
```

Out: <a list of 6 Text major ticklabel objects>

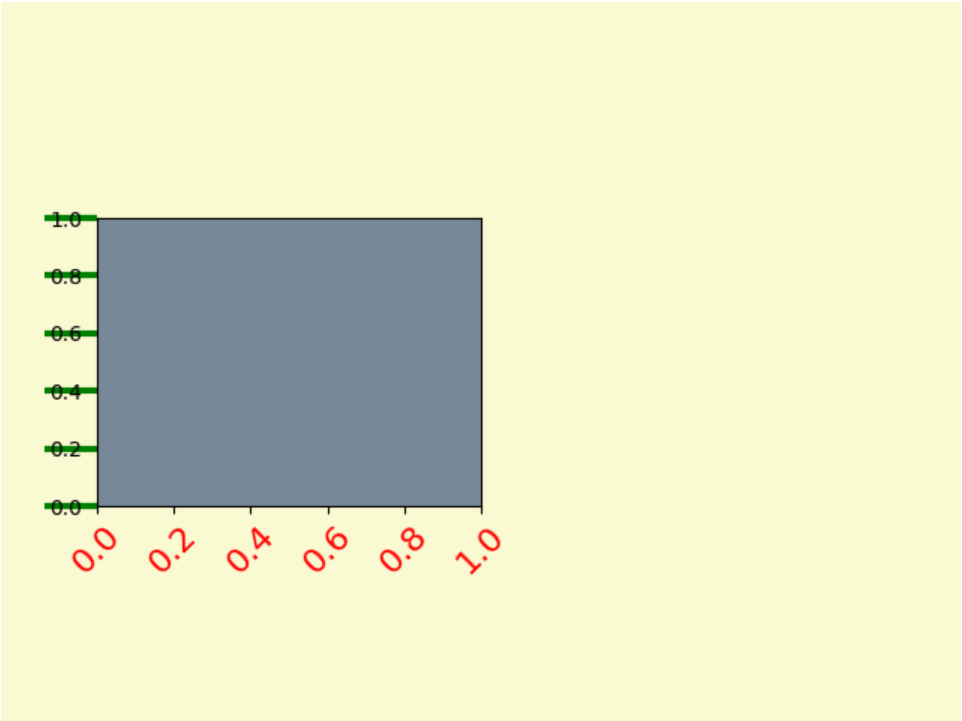
```
axis.get_ticklines()
```

Out: <a list of 12 Line2D ticklines objects>



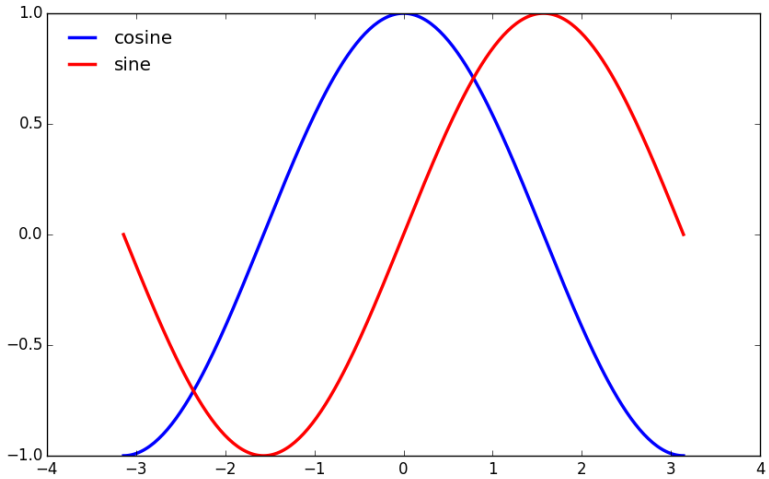
Example

```
fig = plt.figure()
rect = fig.patch  # a rectangle instance
rect.set_facecolor('lightgoldenrodyellow')

ax1 = fig.add_axes([0.1, 0.3, 0.4, 0.4])
rect = ax1.patch
rect.set_facecolor('lightslategray')


for label in ax1.xaxis.get_ticklabels():
    # label is a Text instance
    label.set_color('red')
    label.set_rotation(45)
    label.set_fontsize(16)

for line in ax1.yaxis.get_ticklines():
    # line is a Line2D instance
    line.set_color('green')
    line.set_markersize(25)
    line.set_markeredgewidth(3)

plt.show()
```



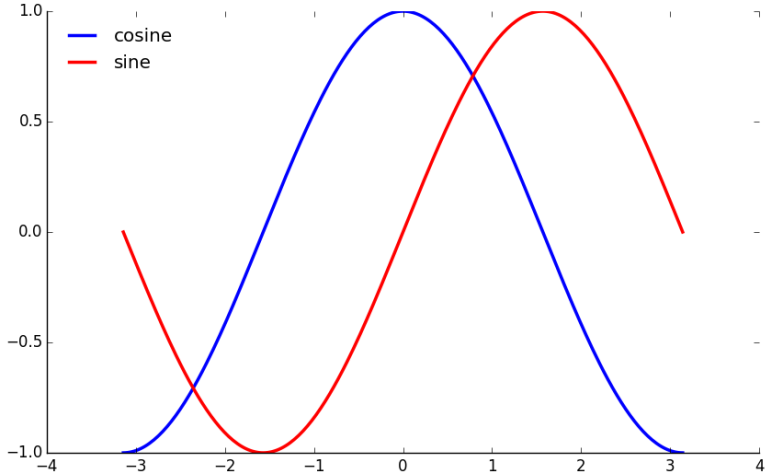
Axes: Moving spines

```
ax = plt.gca()
```



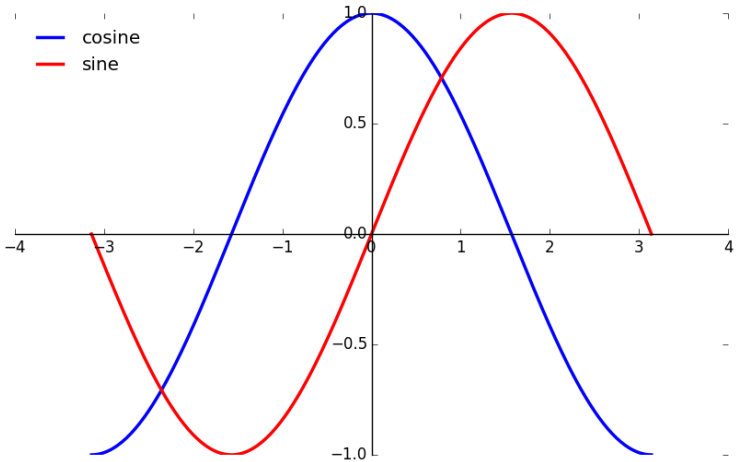
Axes: Moving spines

```
ax = plt.gca()

ax.spines['right'].set_color('none')

ax.spines['top'].set_color('none')
```



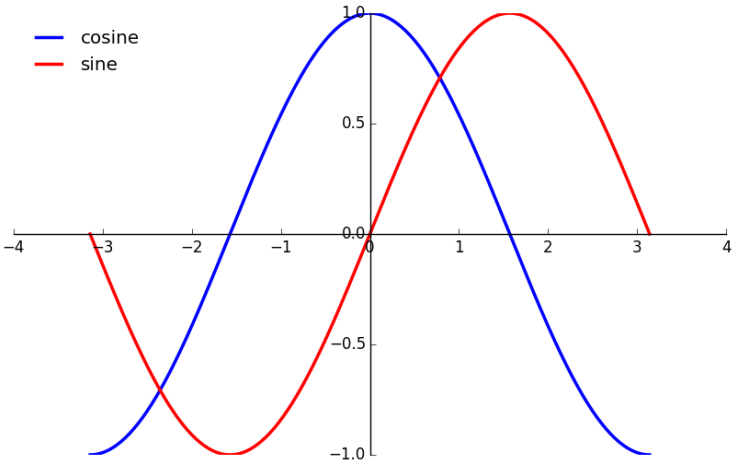
Axes: Moving spines

```
ax = plt.gca()

ax.spines['right'].set_color('none')

ax.spines['top'].set_color('none')

ax.spines['bottom'].set_position('center')

ax.spines['left'].set_position('center')
```



Axes: Moving spines

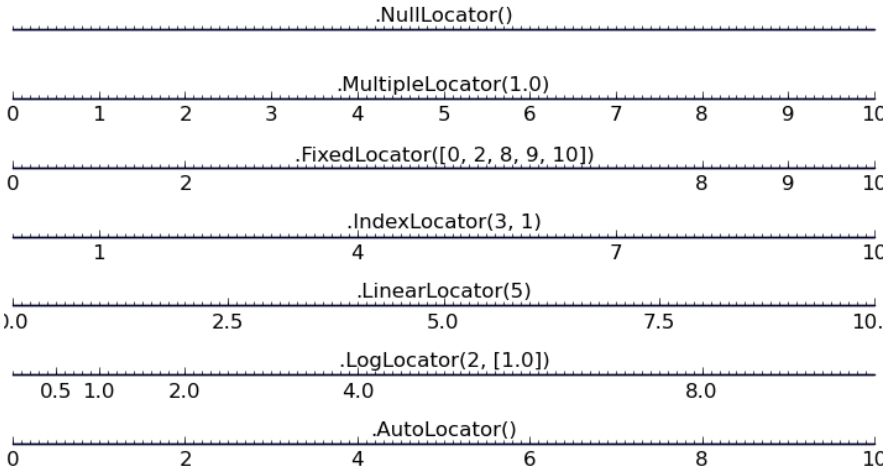
```
ax = plt.gca()

ax.spines['right'].set_color('none')

ax.spines['top'].set_color('none')

ax.spines['bottom'].set_position('center')

ax.spines['left'].set_position('center')

ax.xaxis.set_ticks_position('left')

ax.yaxis.set_ticks_position('bottom')
```



Tick locators

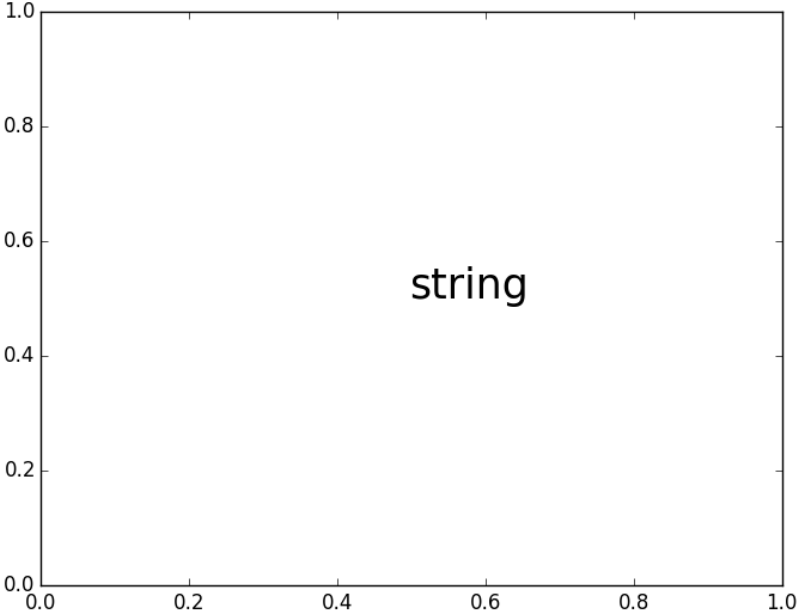
import matplotlib.ticker as tck

```
ax = plt.gca()

locator = tck.LogLocator()

ax.xaxis.set_major_locator((locator))
```



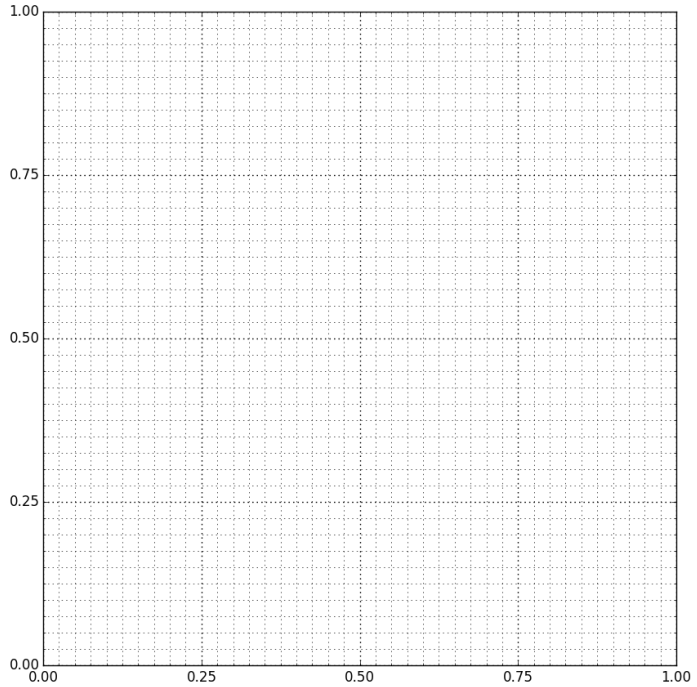
Text

- text(x, y, ’string’)



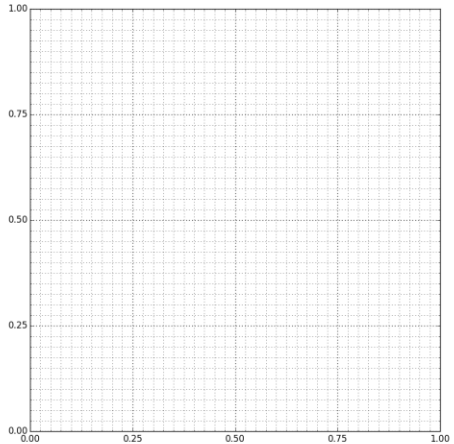
Grids

```
axes.grid(which='major', linewidth=1.0)
axes.grid(which='minor', linewidth=0.5)
```



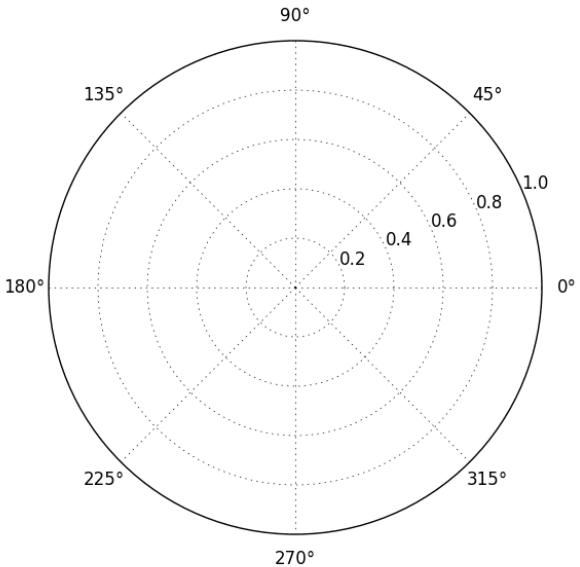
Grids

```
axes.xaxis.set_major_locator(plt.MultipleLocator(0.25))
axes.xaxis.set_minor_locator(plt.MultipleLocator(0.025))
axes.yaxis.set_major_locator(plt.MultipleLocator(0.25))
axes.yaxis.set_minor_locator(plt.MultipleLocator(0.025))

axes.grid(which='major', linewidth=1.0)
axes.grid(which='minor', linewidth=0.5)
```



Polar axes

```
ax = plt.axes(polar=True)
```

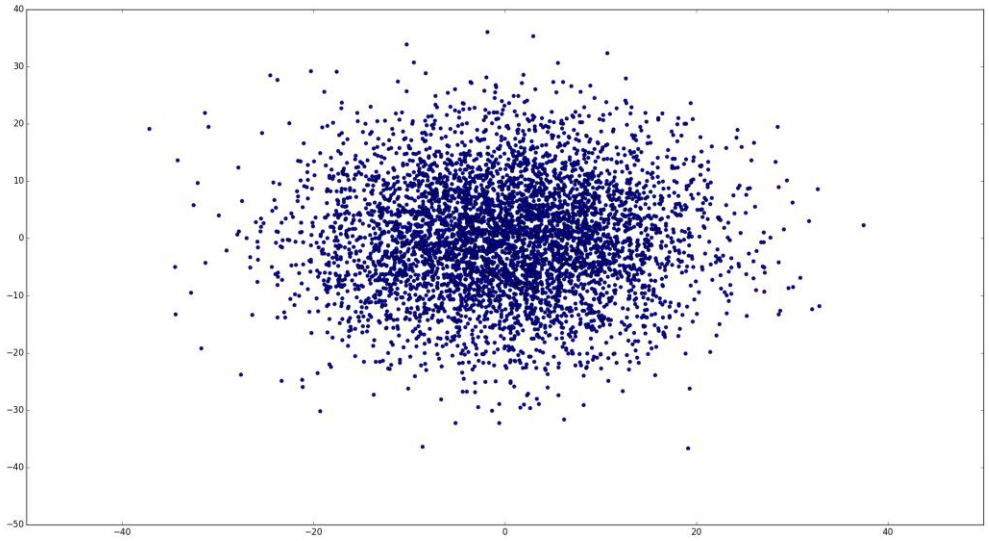


Function Format

```
def my_plotter(ax, data1, data2, param_dict):
    """
    A helper function to make a graph

    Parameters
    ----------
    ax : Axes
        The axes to draw to

    data1 : array
       The x data

    data2 : array
       The y data

    param_dict : dict
       Dictionary of kwargs to pass to ax.plot

    Returns
    -------
    out : list
        list of artists added
    """
    out = ax.plot(data1, data2, **param_dict)
    return out

# which you would then use as:

data1, data2, data3, data4 = np.random.randn(4, 100)
fig, ax = plt.subplots(1, 1)
my_plotter(ax, data1, data2, {'marker': 'x'})
```

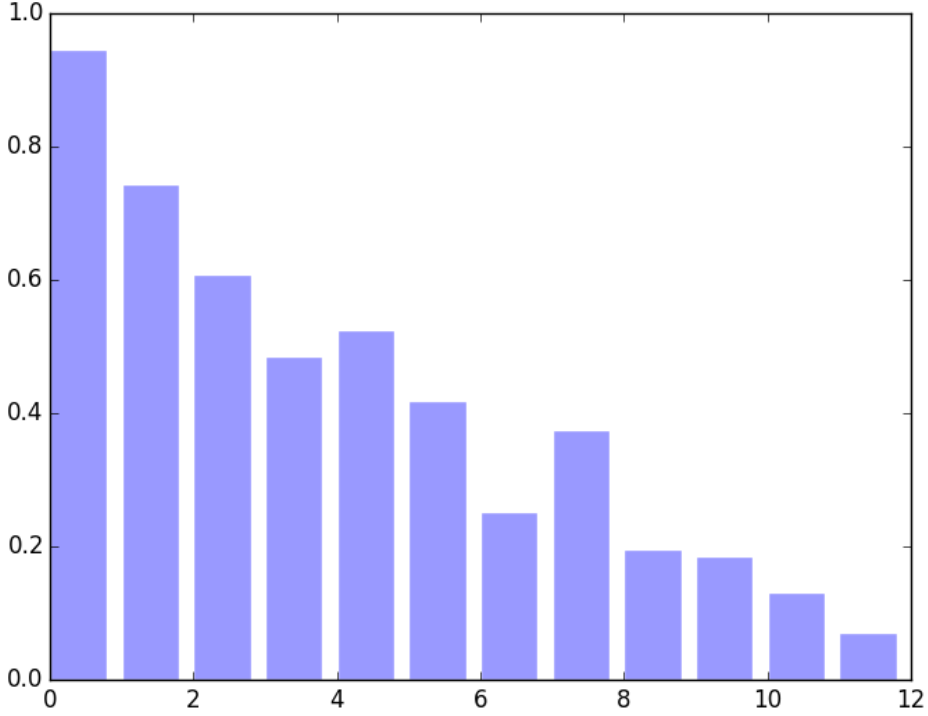
Scatter plots

- scatter(x, y)

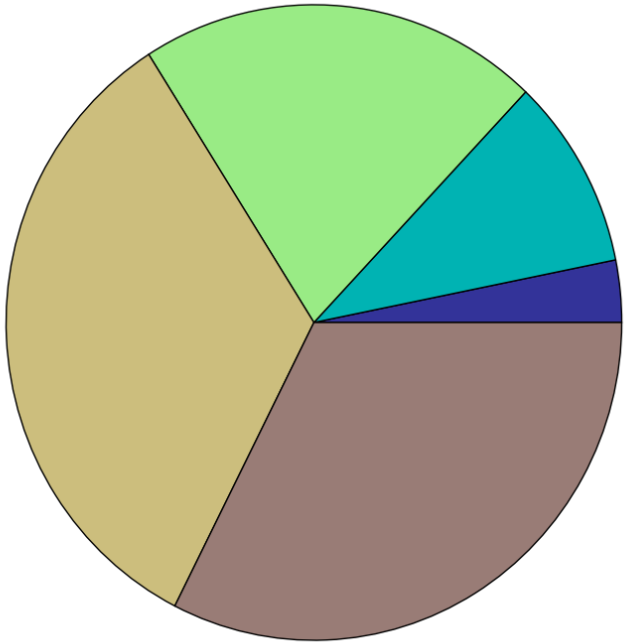


Bar plots

- bar(x, y)

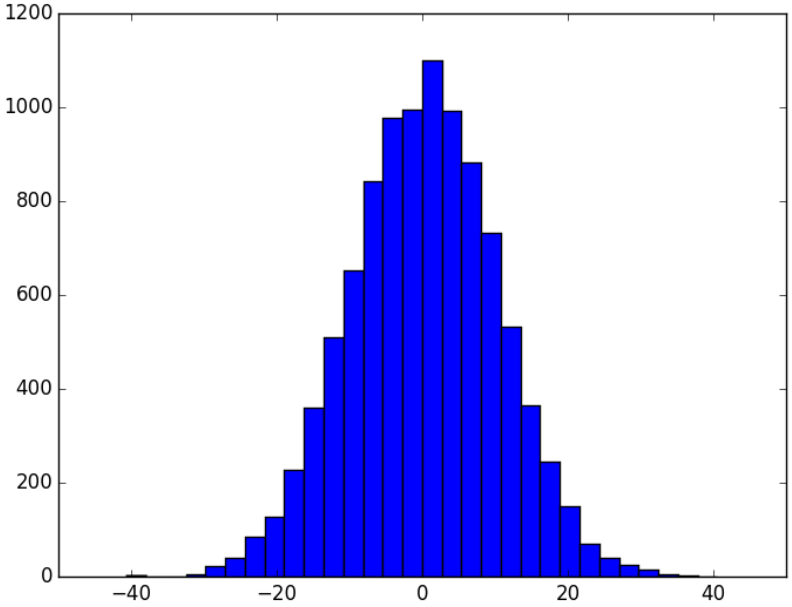


- pie(x)



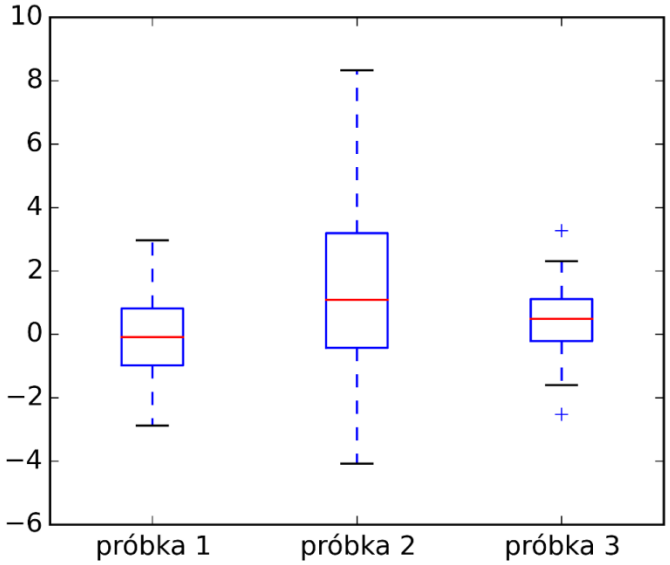
Histograms

- hist(data, bins=30)

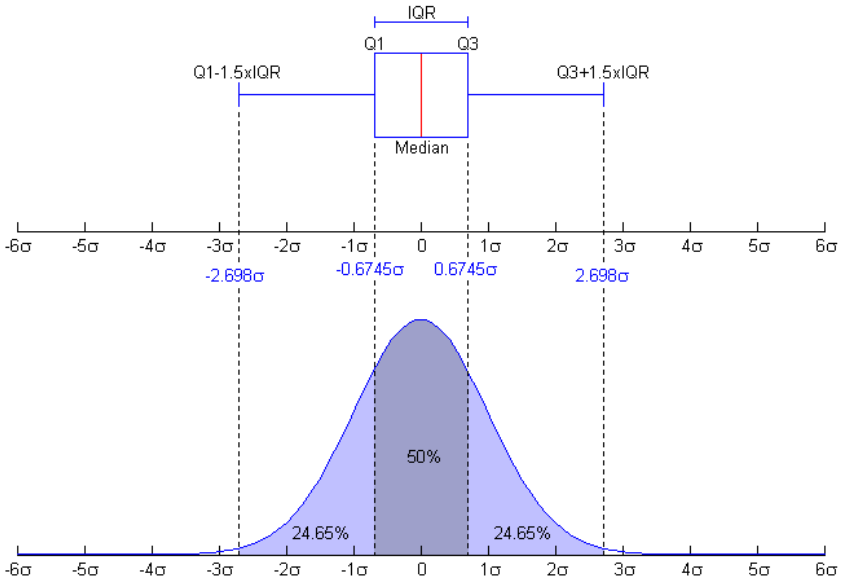


Boxplots

- boxplot((sample1, sample2, etc.))



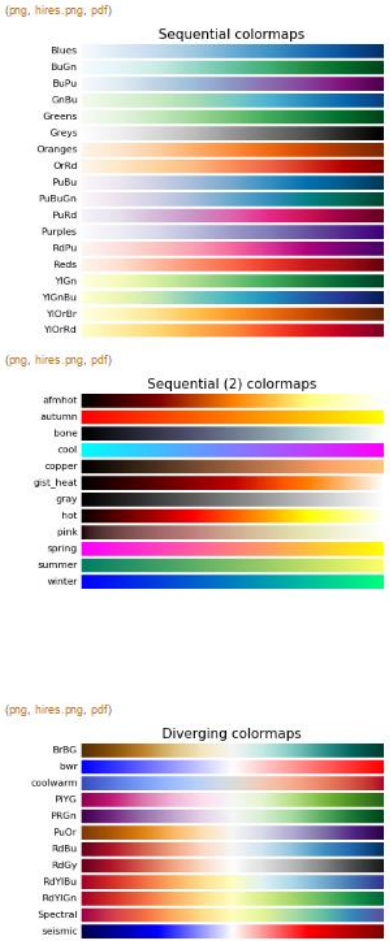
Boxplots


IQR
Q1
Q3
Q1-1.5xIQR
Q3+1.5xIQR
Median
-6σ
-5σ
-4σ
-3σ
-2σ
-1σ
0
1σ
2σ
3σ
4σ
5σ
6σ
-2.698σ
-0.6745σ
0.6745σ
2.698σ
50%
24.65%
24.65%

Color maps

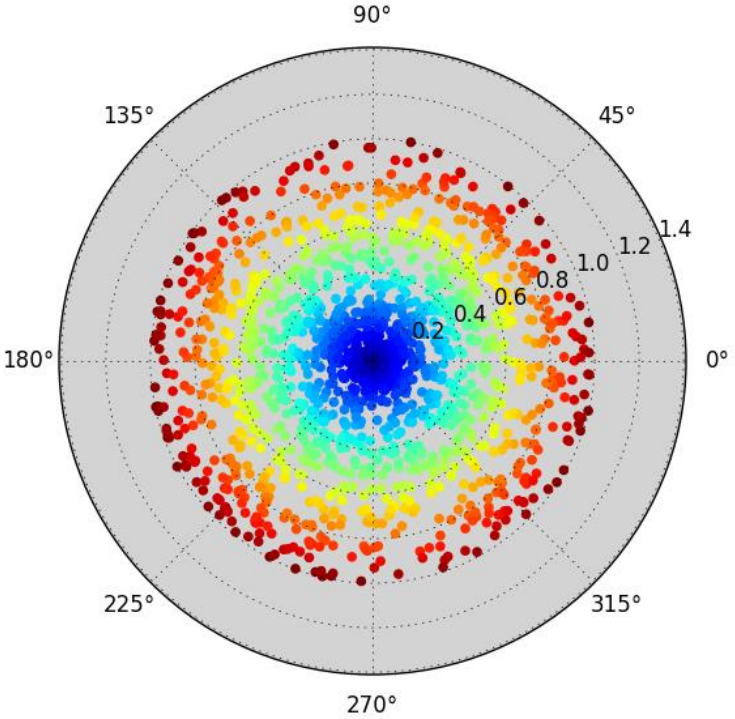
- from matplotlib import cm
- **`color = cm.copper(data)`**

- *# functions from* **cm** *return colors in* **rgba** *format (a 4-tuple list of numbers)*
- *#* **data** *is a list of single numbers from the range* [0.0, 1.0]



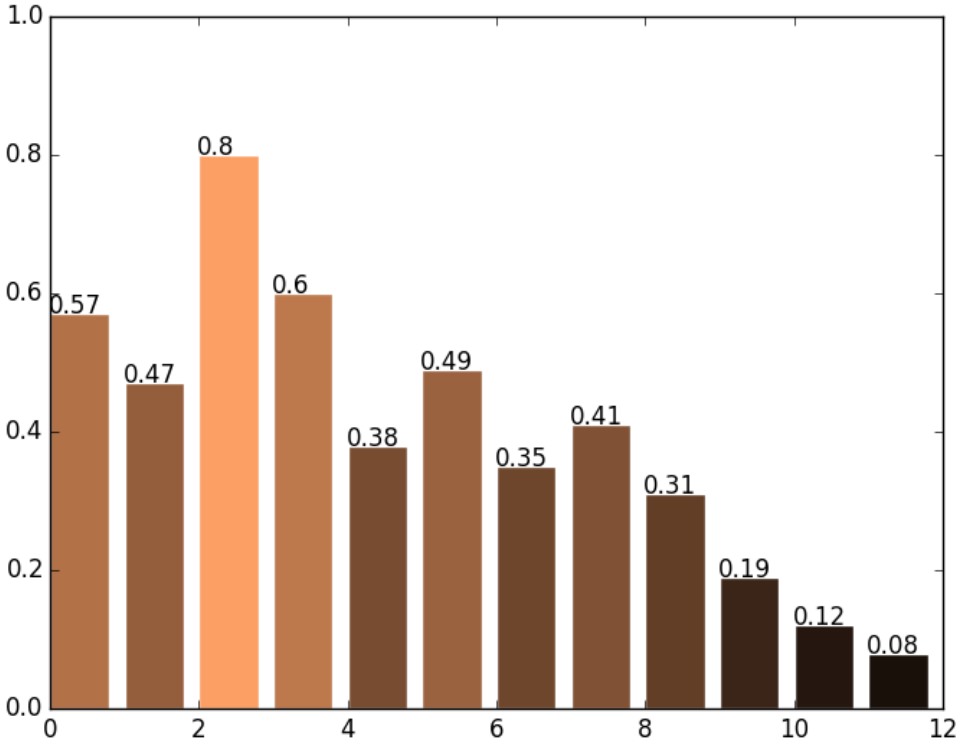
Exercise

- Dane
  - Reconstruct this plot using φ and **r** values in the data file (1st and 2nd line).
  - How to graph this plot in a Cartesian coordinate system?



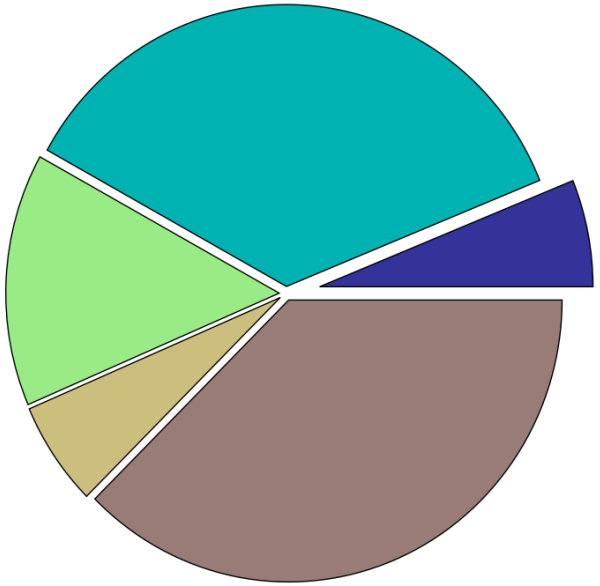
Exercise

- Dane
  - **zip**() is a function that concatenates two lists to a single list of tuples



Exercise

- Dane
  - Reconstruct this pie chart
  - What artist object types does pie return? How does this differ from plot and scatter functions?



Exercises

- Download the file (data2.dat) containing data that has to be visualized once as a normed cumulative and another time as a non-cumulative **histogram**. These histograms are to be plotted side by side in a single figure.
- Download the file (data3.dat) containing three data samples that have to be visualized as a **boxplot**. The figure aspect ratios should be 16:9.

Exercise

- Reconstruct this plot on a grid without tick numbers (use data points of your choice)