12.010 Computational Methods of Scientific Programming 2021

Lecture 8: Graphics

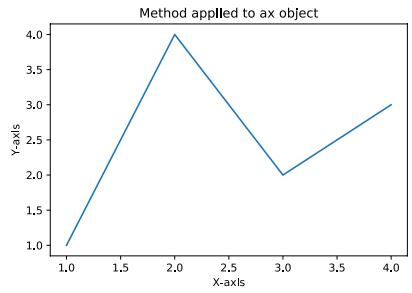
Summary

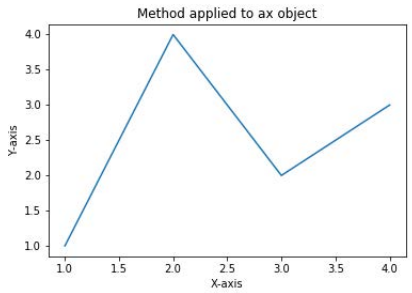
- Graphics: Types
  - Raster/image graphics
  - Vector graphics (postscript, PDF)
- Python Graphics modules
  - Examine different plot types and use of methods and objects

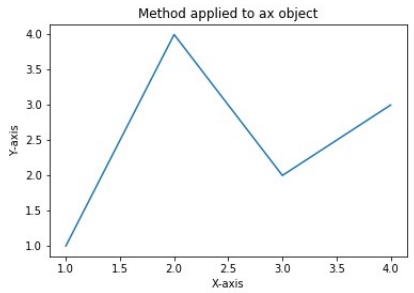
- The notebook for this class is Lec09_graphics.ipynb on Canvas site.

Types of graphics files

- Graphics files fall into two basic types:
  - Vector (ps, pdf, svg)
  - Raster or pixel files (png, jpeg, gif)
- PDF is the preferred method for publishing papers and reports because the resolution is maintained with scaling (and zooming of the figure). Examples below (PDF, PNG, JPEG 50% scaling). However, PC Word does not allow insert of PDF files, Mac does).

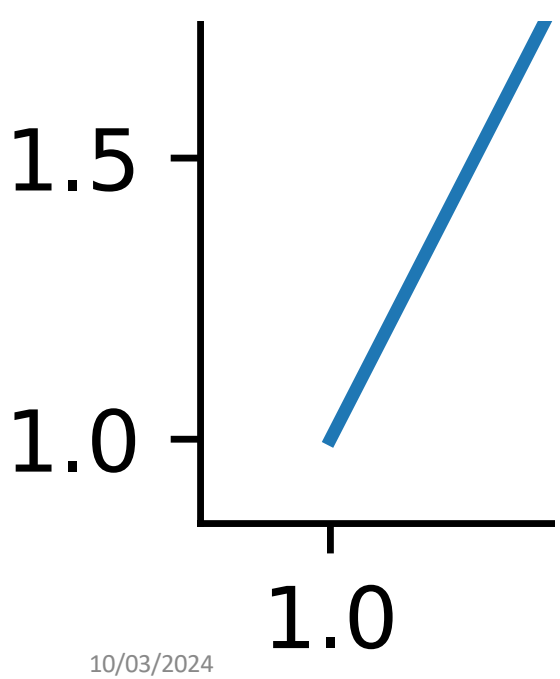


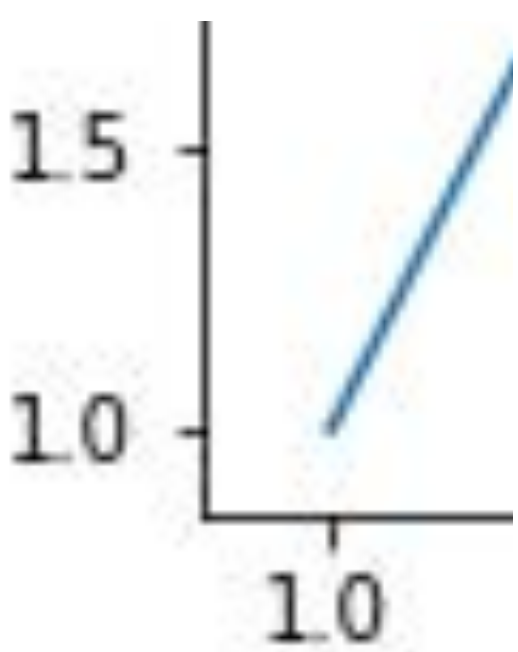


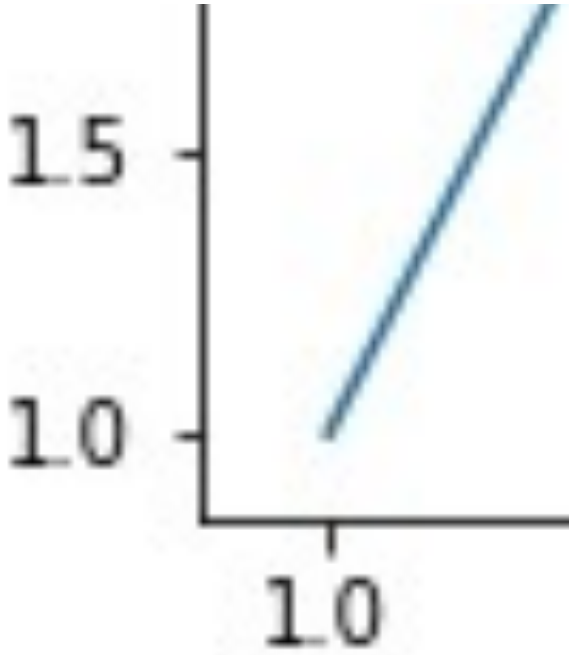


Now zoom and crop the same figures

- PDF, PNG, JPEG







File sizes

- For the simple figure here, file sizes are similar (file size in bytes)
  19156 Lec09_fig.jpg
  10640 Lec09_fig.pdf
  16232 Lec09_fig.png
- For graphics objects with lots of overlapping points and lines. PDF can be much larger than PNG because all points are retained in PDF (and postscript). PDF figures can be edited in Adobe Illustrator.
- PNG can work well if created at the size that it is needed.
- Could look at these figures in Preview to see differences as well. (The Lec09_graphics.ipynb generates the figures shown here).

Python methods for creating graphics*

- The Lec09_graphics.ipynb shows three different approaches, with the latter two being the preferred:
- Use %matplotlib magic to create interactive figures (we will come back to this later)
- Basic: Modules loading into the current namespace
  from matplotlib.pyplot import * (or list of functions).

```
# Method that looks like old style program.  Importing into current name space
# (methods are called as functions but could have issue with functions being
# with different imports)
# we only import specific methods here. Normally * would be used to get all
# funtions
from matplotlib.pyplot import plot, title, xlabel, ylabel, show
plot([1, 2, 3, 4], [1, 4, 2, 3])
title("from matplotlib.pyplot import")
xlabel("X-axis")
ylabel("Y-axis")
show()
```

Use of object methods

- import matplotlib.pyplot as plt
- Different methods are applied with plt to current figure and axes.
- With this approach ‘plot’ could be different functions from different modules. pyplot is loaded into its own namespace.

```
# This next method is one of the more common peferred methods
# using a separate namespace for the modules
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
plt.title("import matplotlib.pyplot as plt")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

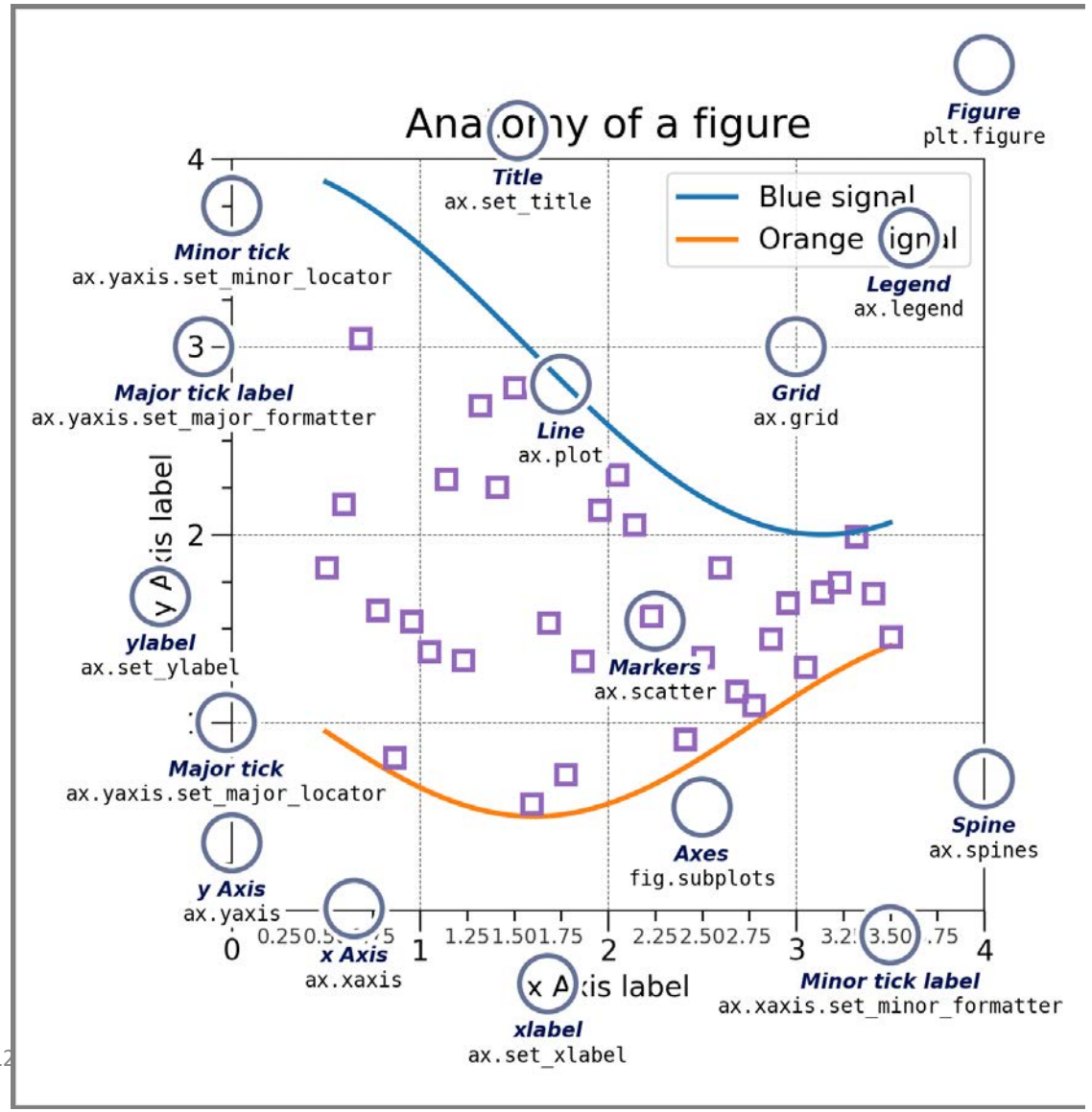
Use of object oriented (OO) method

- Similar to the previous approach but figure and axis objects are created, and methods applied to the object.
- Very similar to the previous slide, but some methods have different names

```
import matplotlib.pyplot as plt
# Create new Figure with black background
# Could Add: figsize=(8, 8) in figure creation.
fig = plt.figure()
# Add a subplot (could use multiple)
ax = plt.subplot()
#fig, ax = plt.subplots()
#fig.figsize=(8, 8)
# Now plot methods are applied to the ax object
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Method applied to ax object")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis");  # Adding ; stops output
```

Figure components

- Parts of figures



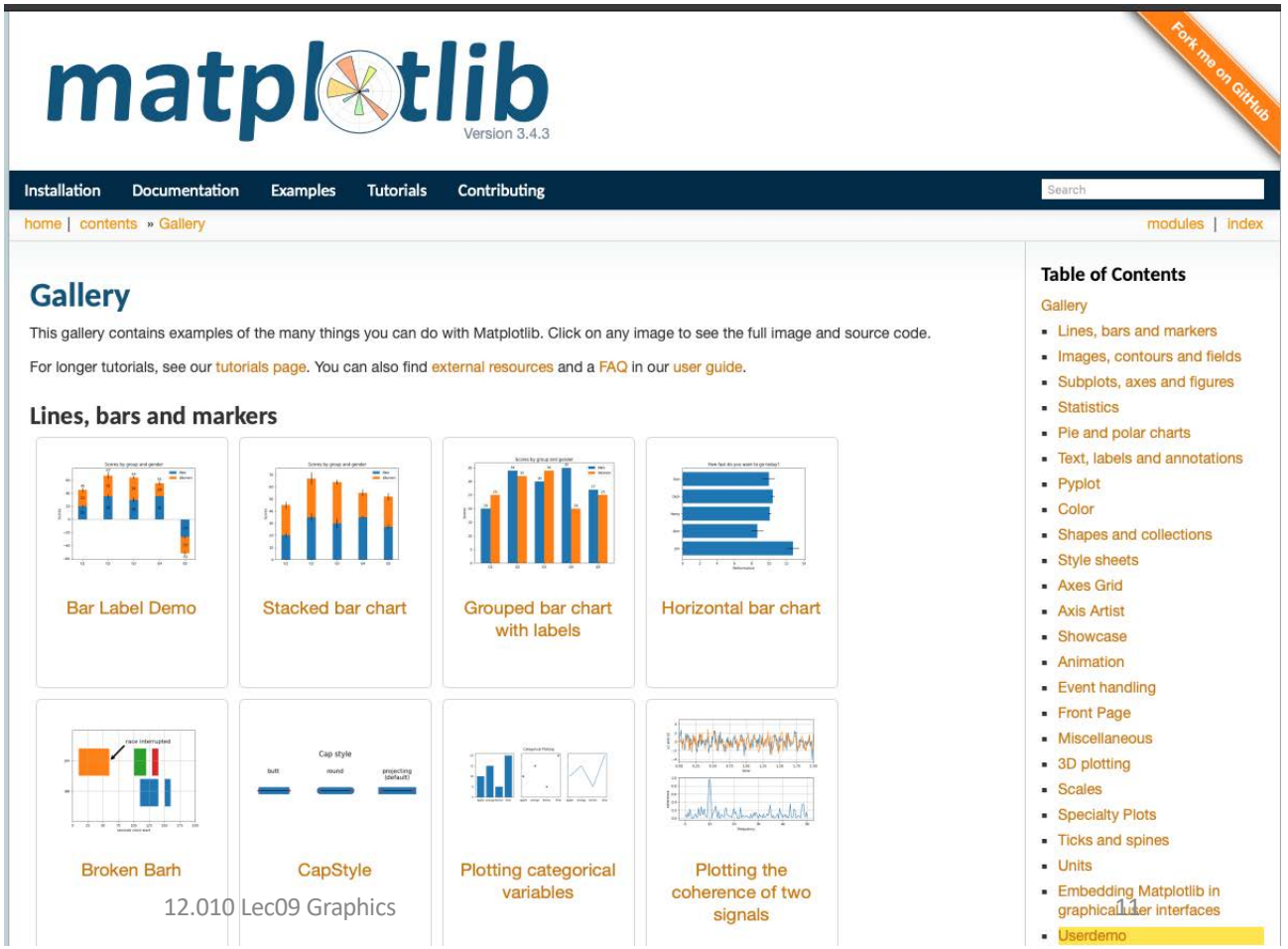
From: https://matplotlib.org/stable/tutorials/introductory/usage.html

Controlling and setting matplotlib style*

- There are multiple ways to control the style of matplotlib plots
- Options can be included in the plot/axis, etc calls
- For the overall style applied to all plots, use rcParams. These can be set with a configuration file or in code (better for exporting code)
- Full documentation at: https://matplotlib.org/stable/api/matplotlib_configuration_api.html#matplotlib.rcParams
- Create a style file and plt.style.use('my.style')
- Set with matplotlib.rcParams['lines.linewidth'] = 2
- Restore defaults: matplotlib.rc_file_defaults and matplotlib.style.use('default')
- matplotlib.rcParams.keys() to see all keys,

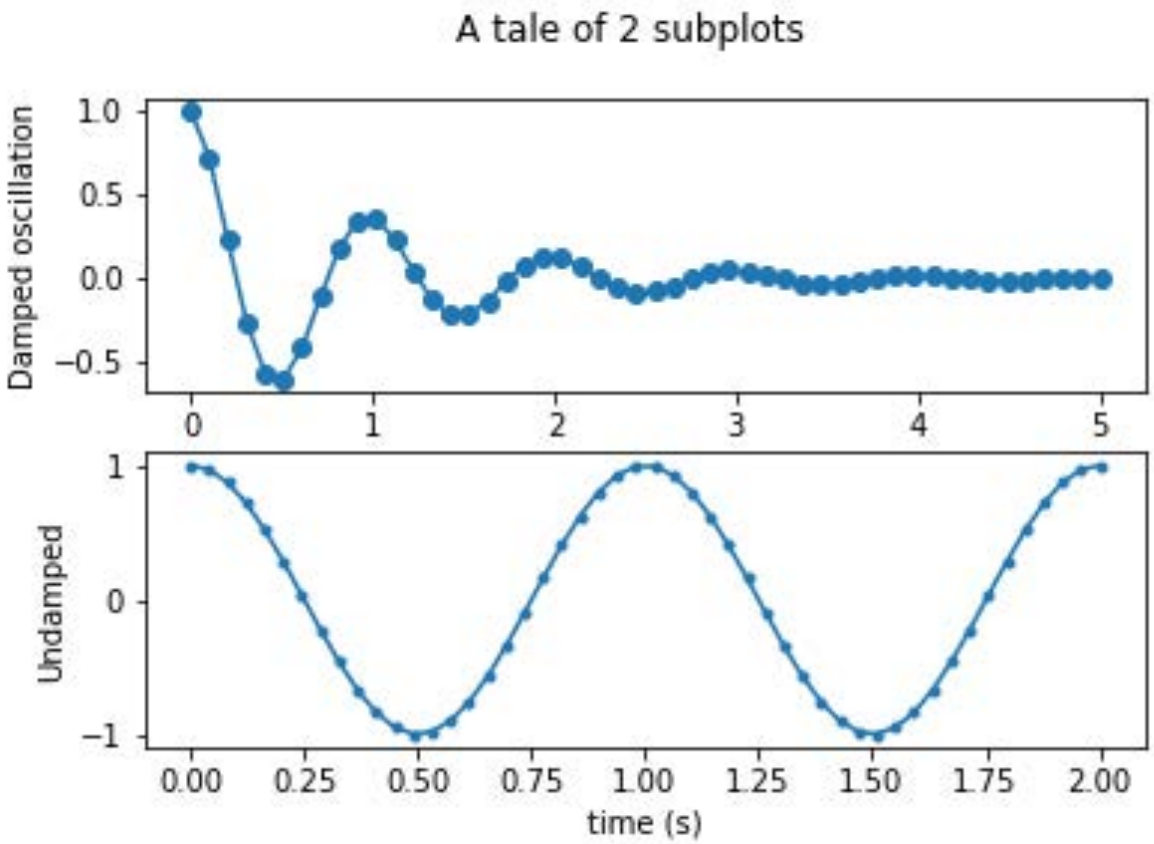
Documentation and examples

- URL https://matplotlib.org/stable/gallery/index.html
- Look down Examples to see the type of plot you want. Py and ipynb codes are available, which can be modified for your specific problem.
- We show some examples



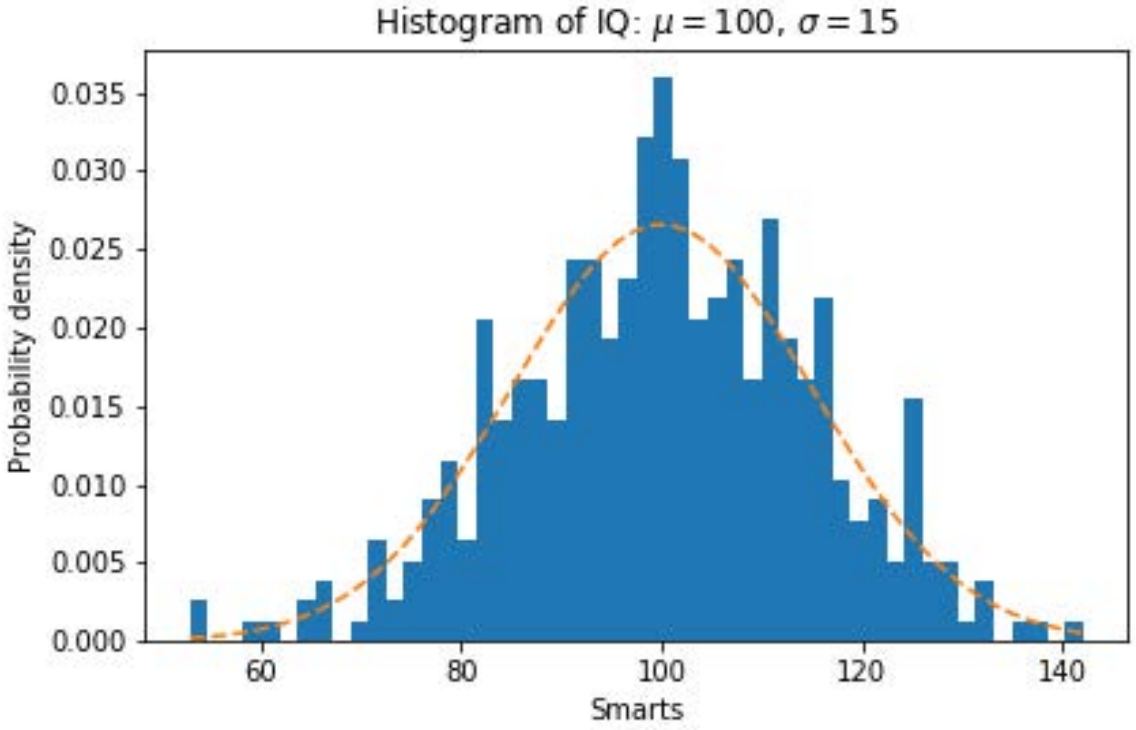
Tutorial from Matplotlib*

- Subplots: 2 approaches in Lec09_graphics.ipynb



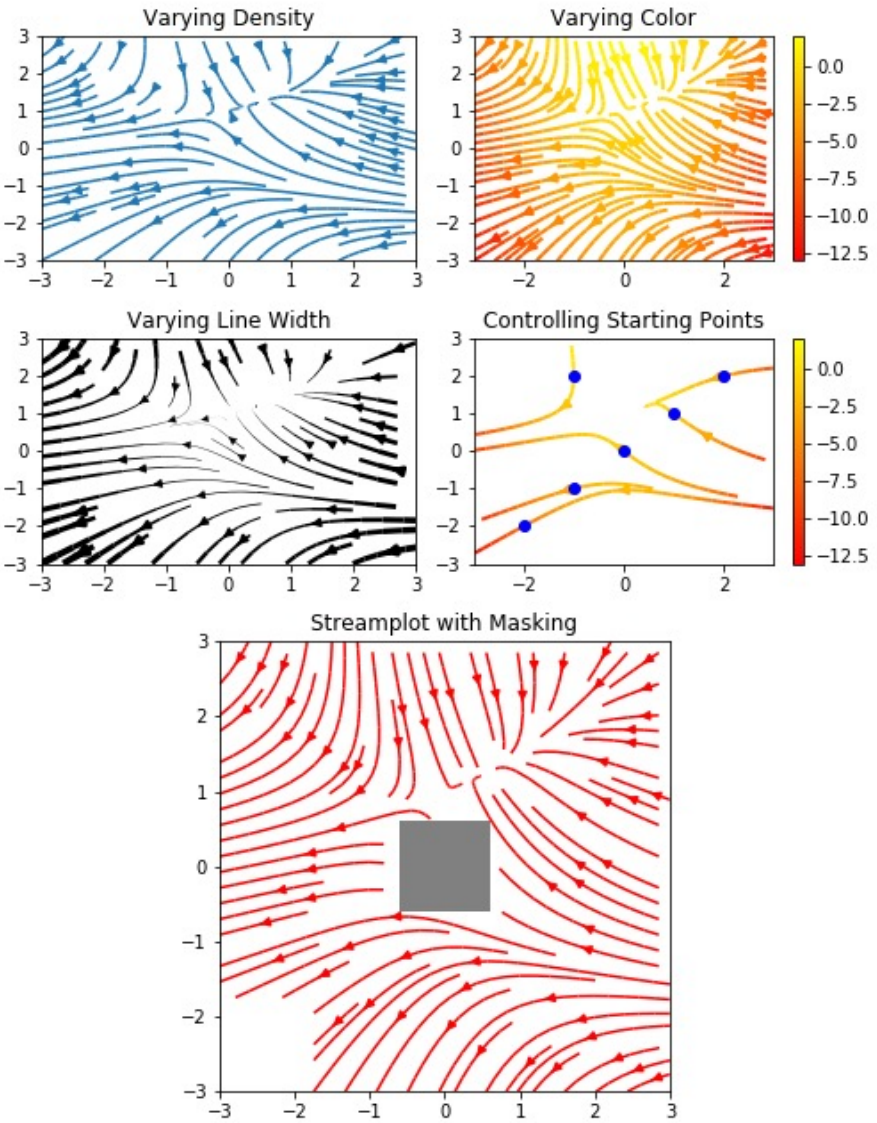
Histogram

- Random + expectation



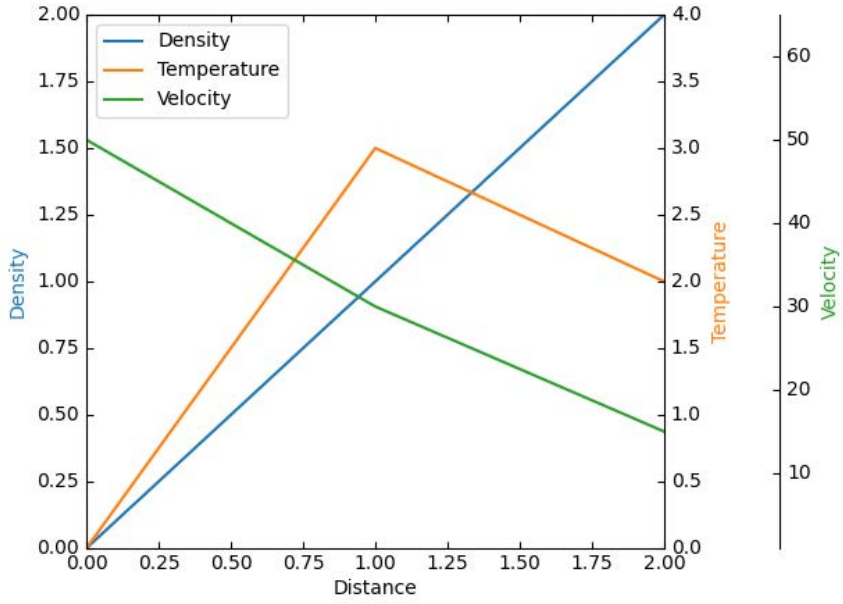
Stream plots

- Combination of plots



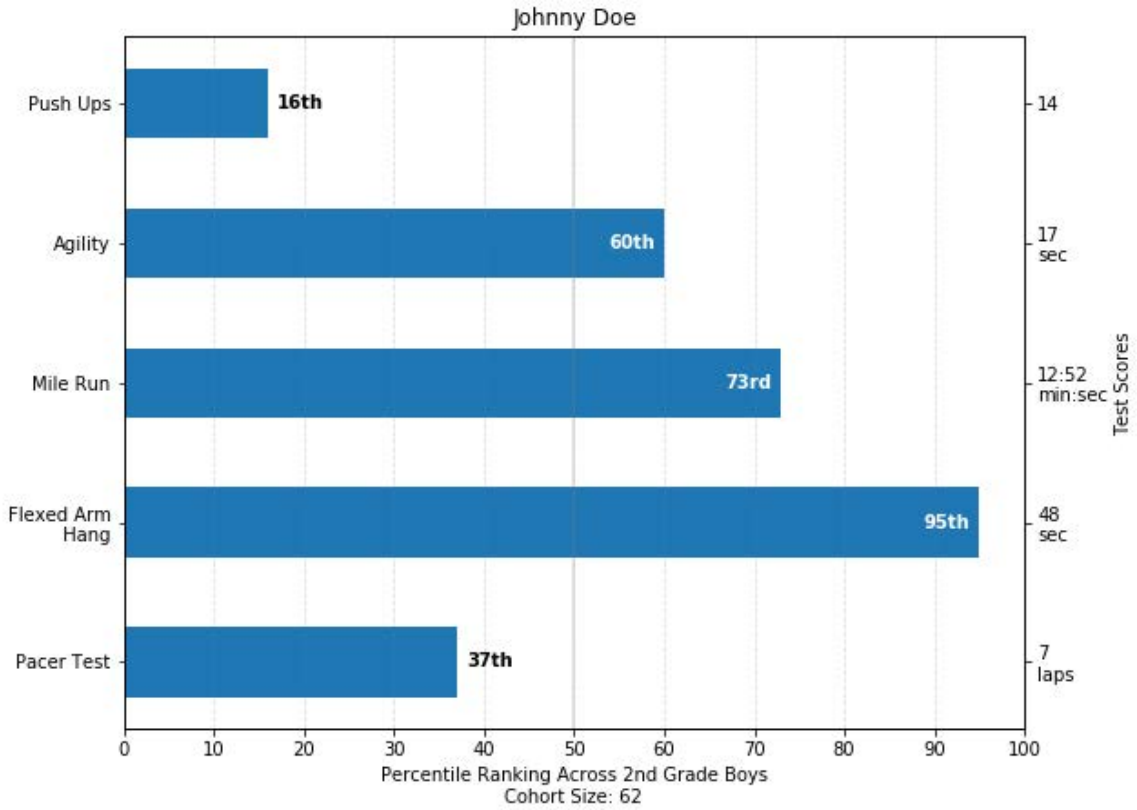
Multiple and offset axes

- Original case https://matplotlib.org/stable/gallery/axisartist/demo_parasite_axes.html#sphx-glr-gallery-axisartist-demo-parasite-axes-py



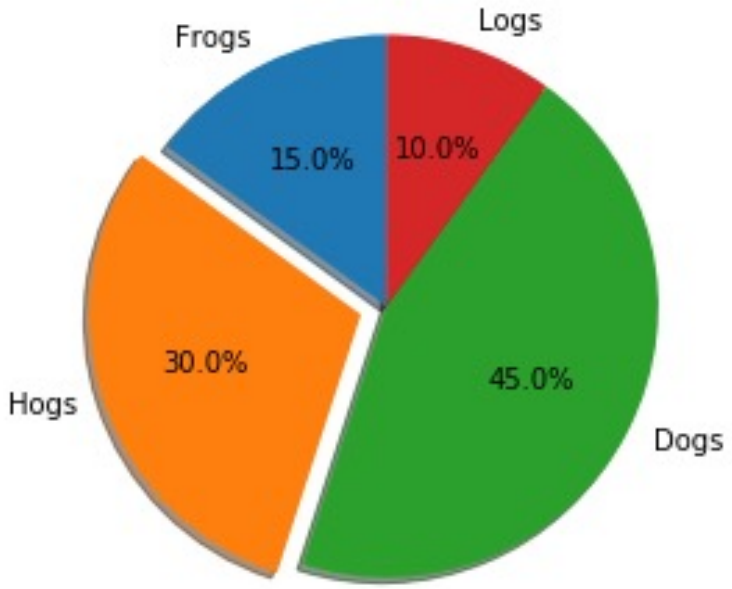
Bar graphs

- Many additional features



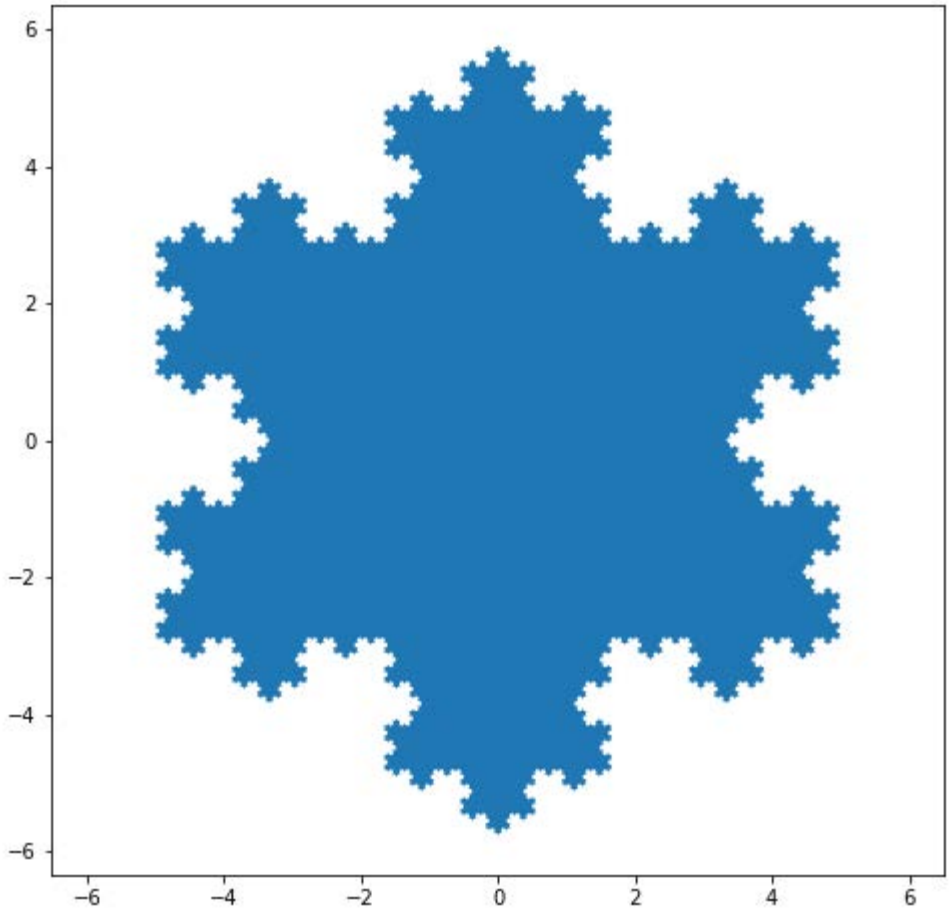
Pie Charts

- With exploded segment



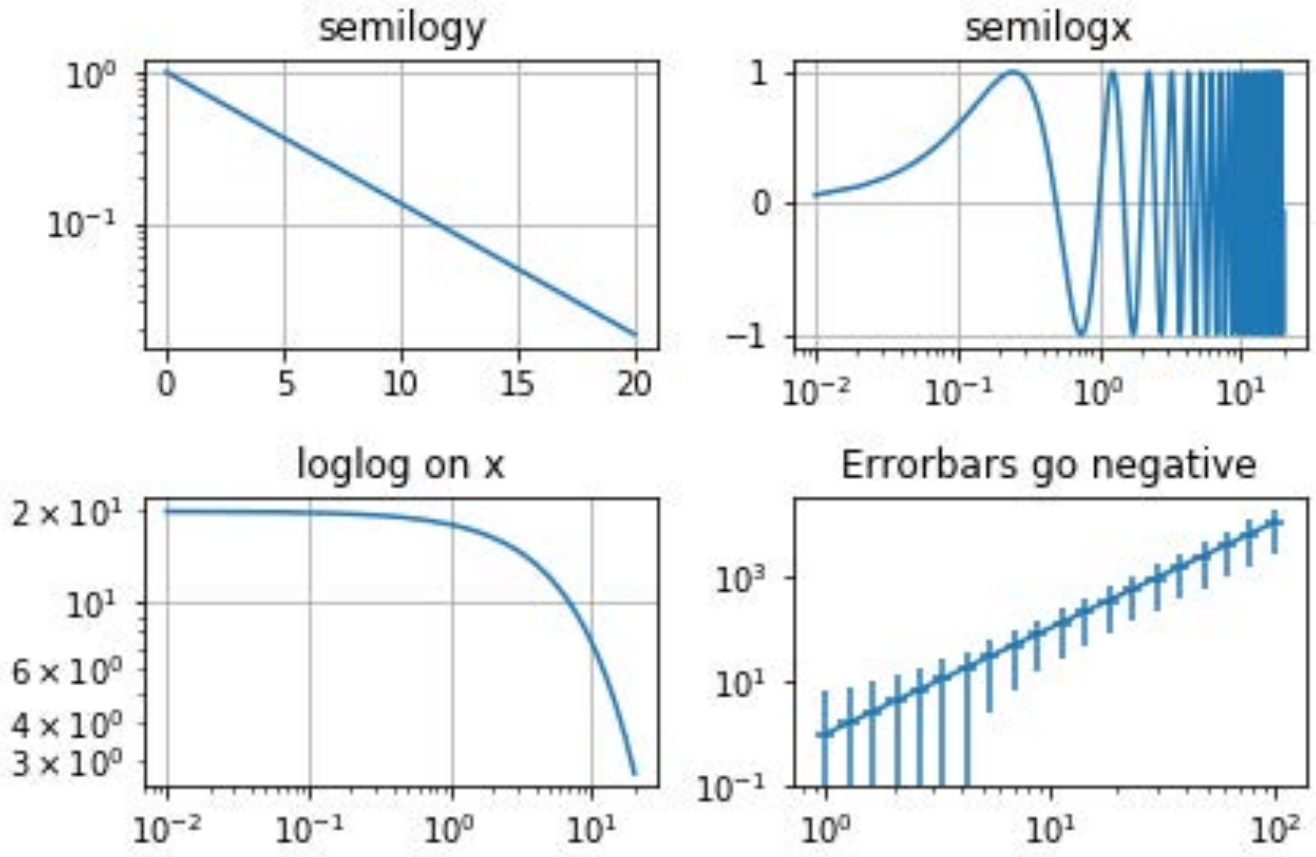
Filled polygon

- More options in notebook



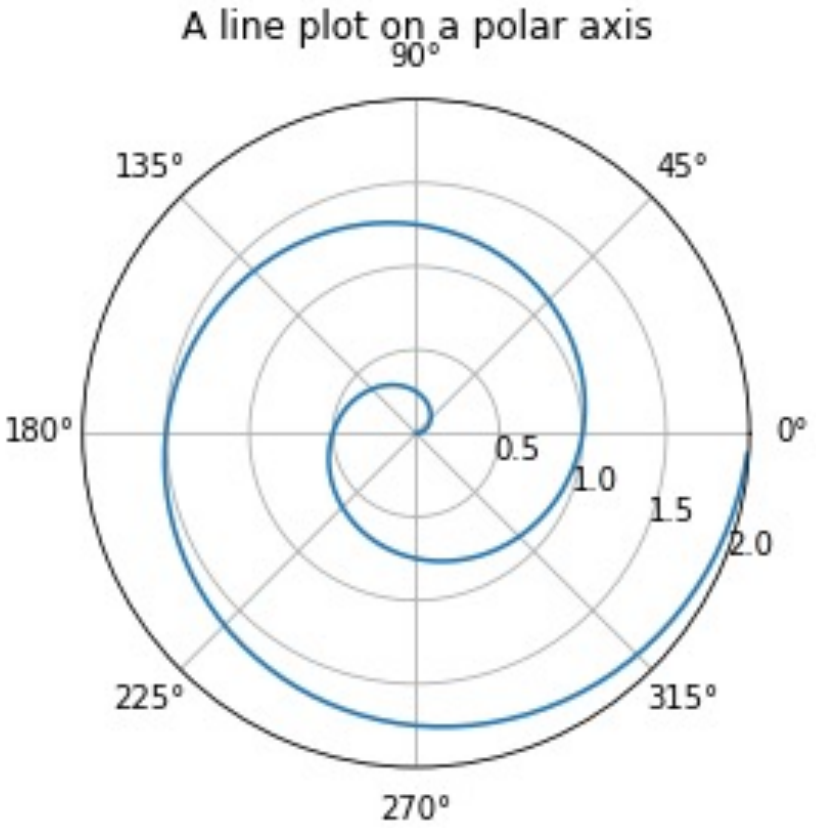
Log plots

- Some issues with tutorial code



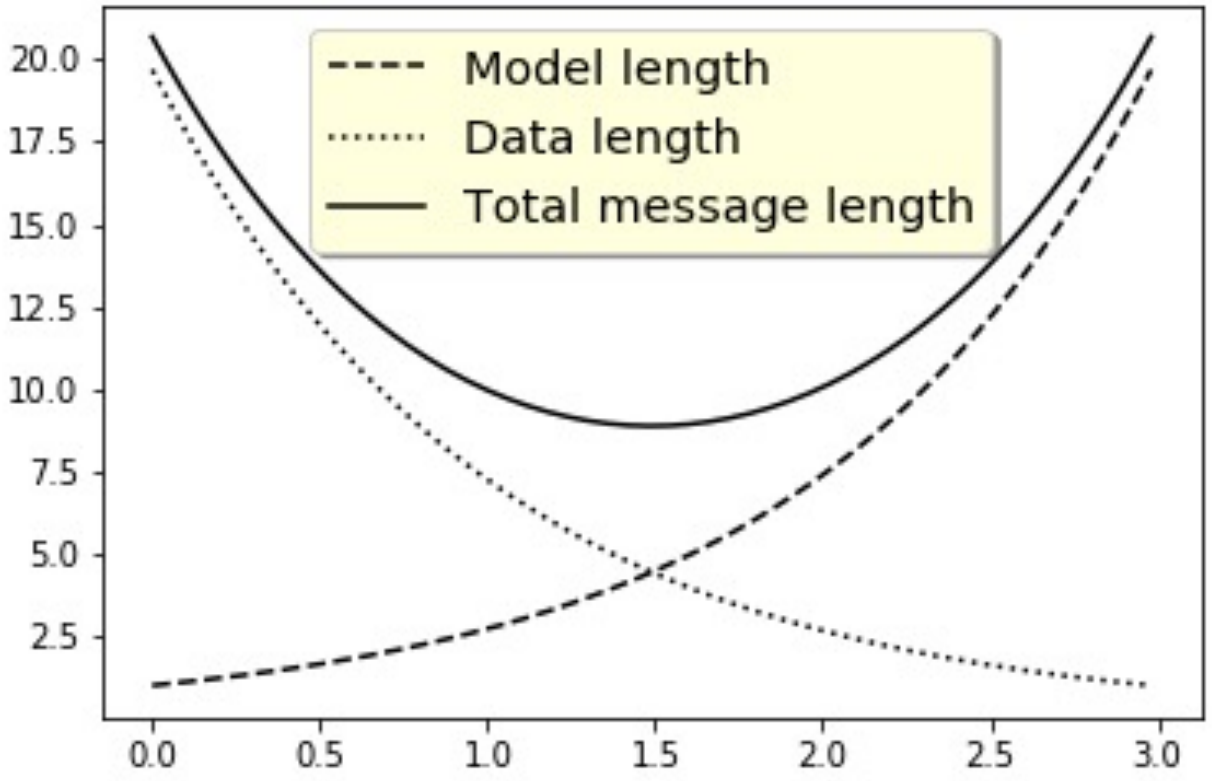
Polar plot

- Spiral



Legends

- Set colors



Summary

- Lots of good examples but be careful of the version of matplotlib – not all work with anaconda installation.
  - In some cases this is due to differences between versions
  - One case bug in current code or change between versions that no longer works.
- The next discussion is on Graphical User Interfaces (GUI) and interaction with plots. It is not straightforward and version and installation-dependent.

Math rendering

- Examples: Renders in the notebook, but figure save is blank. Examine why this is the case in the notebook.
- Issue related to show() usage.

MIT OpenCourseWare

https://ocw.mit.edu

12.010 Computational Methods of Scientific Programming, Fall 2024