ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

Chương 3: Lập trình Python cho KHDL - Trực quan hoá dữ liệu với Matplotlib

TS. Phạm Công Thắng
Khoa Công nghệ thông tin

Tài liệu tham khảo

- https://matplotlib.org/
- https://matplotlib.org/stable/Matplotlib.pdf
- Jake VanderPlas, Python Data Science Handbook, O'Reilly Media, Inc., 548 pages, 2016.
- Peter Bruce, Andrew Bruce, Practical Statistics for Data Scientists: 50 Essential Concepts - 1st Edition, O'Reilly Media; 1st edition, 90 pages, 2017
- Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists 1st Edition, 392 pages, 2016
- Lillian Pierson, Data Science For Dummies - 2nd Edition, John Wiley & Sons Inc., 385 pages, 2017.
- David R. Anderson, Dennis J. Sweeney, Thomas A. Williams, Statistics for Business and Economics, South-Western College Pub., 880 pages, 2010.
- Các nguồn internet khác.

Matplotlib: Visualization with Python

- Matplotlib: Visualization with Python
  - Trực quan hóa bằng PythonMatplotlib là một thư viện để tạo các hình ảnh trực quan tĩnh, động và tương tác bằng Python.

Matplotlib

- Matplotlib để trực quan hóa bằng Python:
  - Matplotlib là một thư viện trực quan hóa dữ liệu đa dạng được xây dựng dựa trên NumPy và được thiết kế để hoạt động với thư viện SciPy một cách rộng lớn hơn.
  - Được John Hunter đề xuất vào năm 2002, là một bản vá cho IPython để cho phép vẽ biểu đồ kiểu MATLAB tương tác thông qua gnuplot từ dòng lệnh Ipython: gói Matplotlib ra đời, với phiên bản 0.1 được phát hành vào năm 2003.
  - Một trong những tính năng quan trọng nhất của Matplotlib là khả năng thích ứng tốt với nhiều hệ điều hành và các nền tảng đồ họa.

Matplotlib

- Matplotlib có thể được cài đặt từ thư mục nguồn một cách đơn giản :

  python -m pip install

- Matplotlib cần các điều kiện rang buộc sau:
  - Python (>= 3.6)
  - NumPy (>= 1.15)
  - setuptools
  - cycler (>= 0.10.0)
  - dateutil (>= 2.1)
  - kiwisolver (>= 1.0.0)
  - Pillow (>= 6.2)
  - pyparsing (>=2.0.3)

Matplotlib

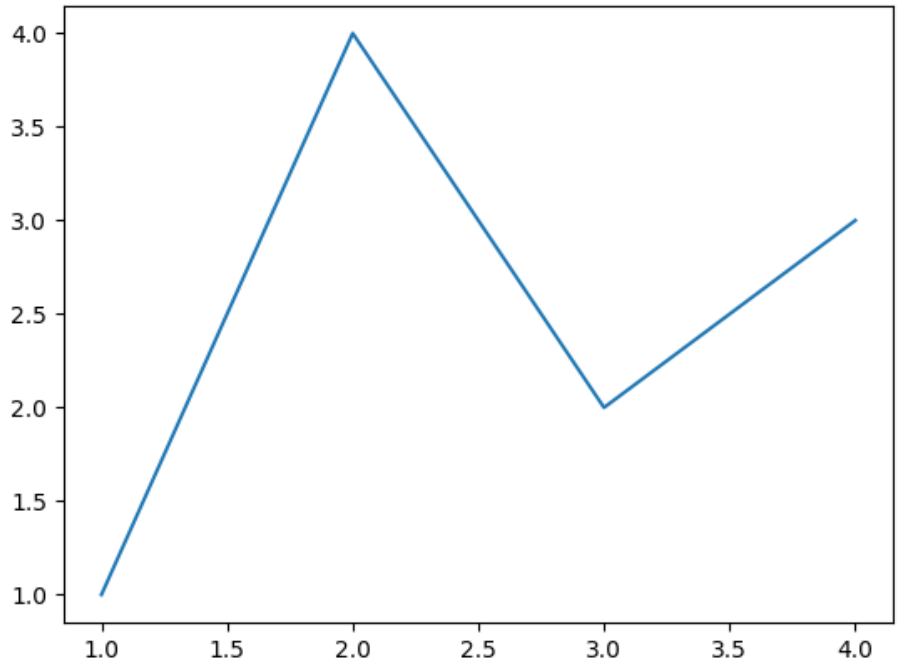
- Sử dụng cơ bản bắt đầu với Matplotlib.

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

- Matplotlib vẽ biểu đồ dữ liệu trên các Hình (Figures): windows, Jupyter widgets, …
  - Mỗi Hình có thể chứa một hoặc nhiều Trục (tức là một khu vực mà các điểm có thể được biểu diễn theo tọa độ xy (hoặc theta-r trong một cực plot hoặc xyz trong một biểu đồ 3D, v.v.).
  - Cách đơn giản nhất để tạo một hình với các trục là sử dụng pyplot.subplots. Sau đó, chúng ta có thể sử dụng Axes.plot để vẽ một số dữ liệu trên các trục

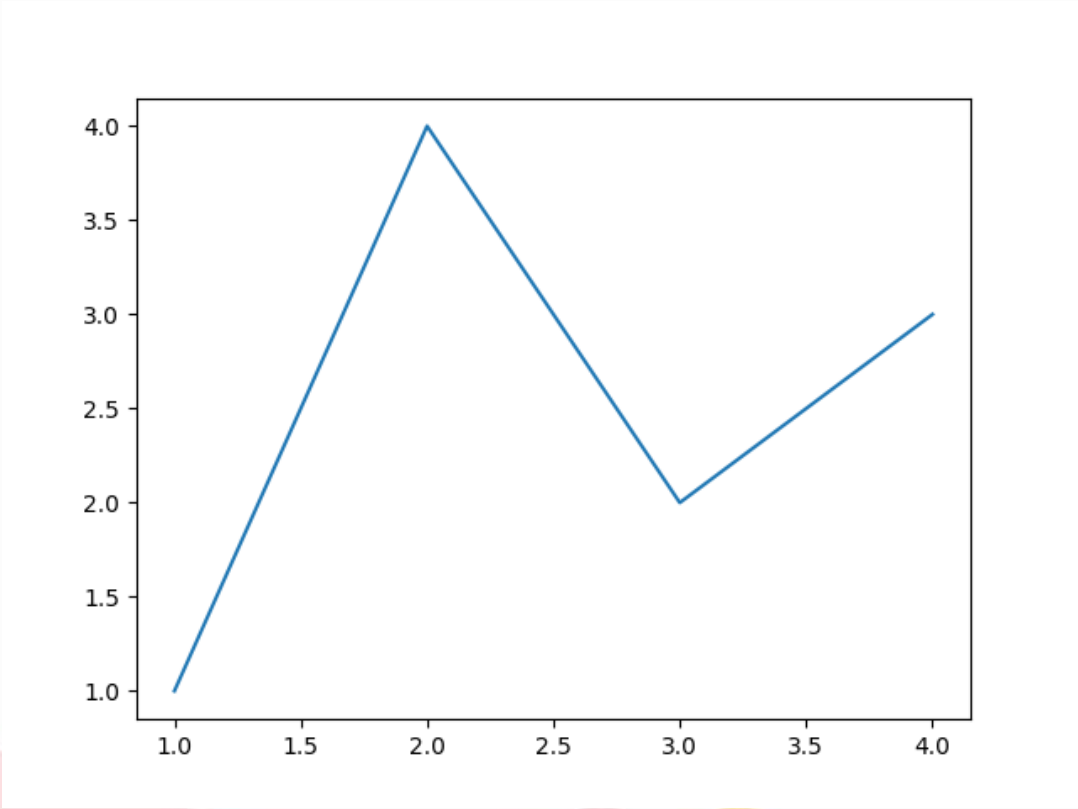
Matplotlib

```
fig, ax = plt.subplots() # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Plot some data on the axes.
```



Matplotlib

plt.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Matplotlib plot.



Matplotlib

- Axes chứa hai (hoặc ba trong trường hợp là 3D) đối tượng Axis (lưu ý sự khác biệt giữa Axes và Axis) quan tâm đến các giới hạn dữ liệu (các giới hạn dữ liệu cũng có thể được kiểm soát thông qua axes.Axes.set_xlim() và các phương thức axes.Axes.set_ylim() .
- Mỗi Axes có một tiêu đề (đặt qua set_title ()), một nhãn x-label (đặt qua set_xlabel ()) và một nhãn y-label được đặt qua set_ylabel ()).

```
fig = plt.figure() # an empty figure with no Axes
fig, ax = plt.subplots() # a figure with a single Axes
fig, axs = plt.subplots(2, 2) # a figure with a 2x2 grid of Axes
```

Matplotlib

- Phân loại đầu vào cho các hàm vẽ biểu đồ
  - Tất cả các hàm vẽ biểu đồ đều lấy *numpy.array* hoặc *numpy.ma.masked_array* làm đầu vào.
  - Các lớp ‘array-like' như dữ liệu pandas và numpy.matrix có thể hoạt động như dự kiến hoặc không.
    - Tốt nhất là chuyển đổi chúng thành các đối tượng numpy.array trước khi vẽ biểu đồ.

Matplotlib

- Ví dụ: chuyển đổi thành pandas.DataFrame

```
a = pandas.DataFrame(np.random.rand(4, 5), columns = list('abcde'))
a_asarray = a.values
```

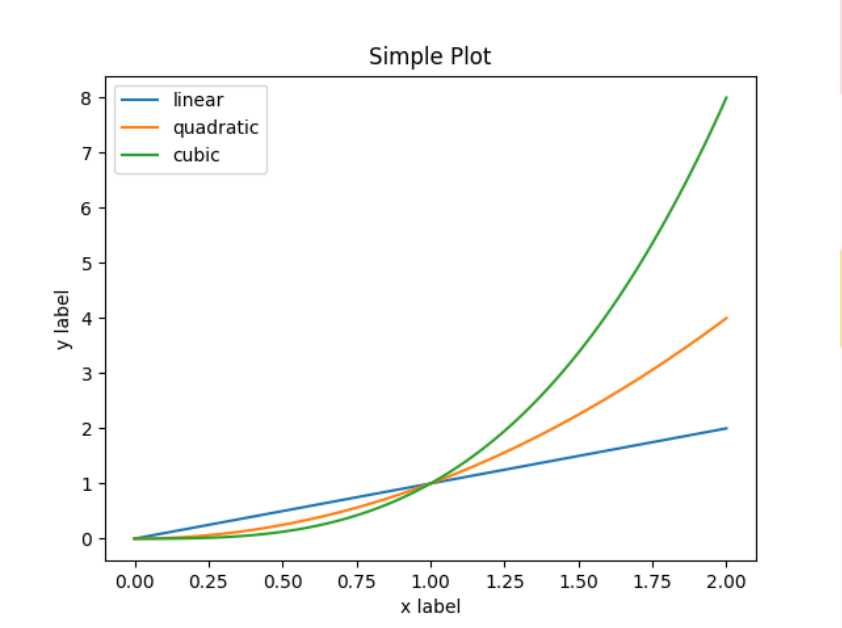
- Và chuyển đổi thành numpy.matrix

```
b = np.matrix([[1, 2], [3, 4]])
b_asarray = np.asarray(b)
```

Matplotlib

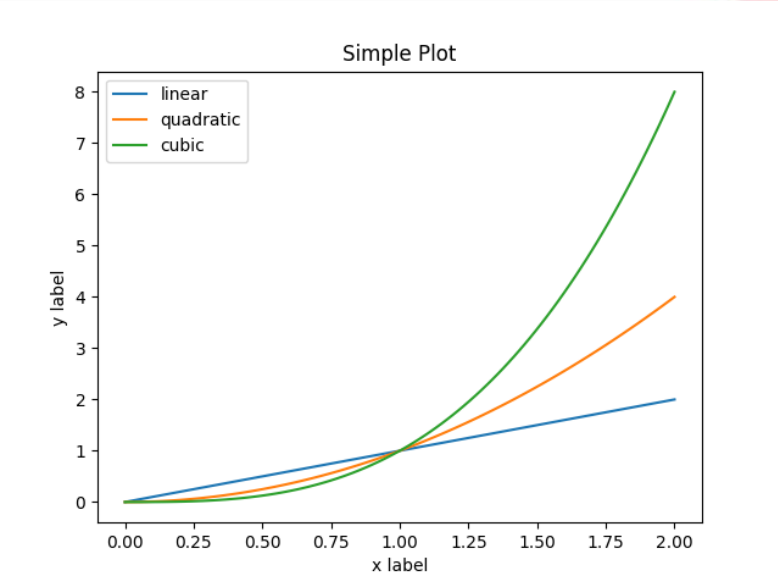
- pyplot interface

```
x = np.linspace(0, 2, 100)

# Note that even in the OO-style, we use `.pyplot.figure` to create the figure.
fig, ax = plt.subplots()  # Create a figure and an axes.
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend()  # Add a legend.
```



Matplotlib

```
x = np.linspace(0, 2, 100)

plt.plot(x, x, label='linear')  # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```

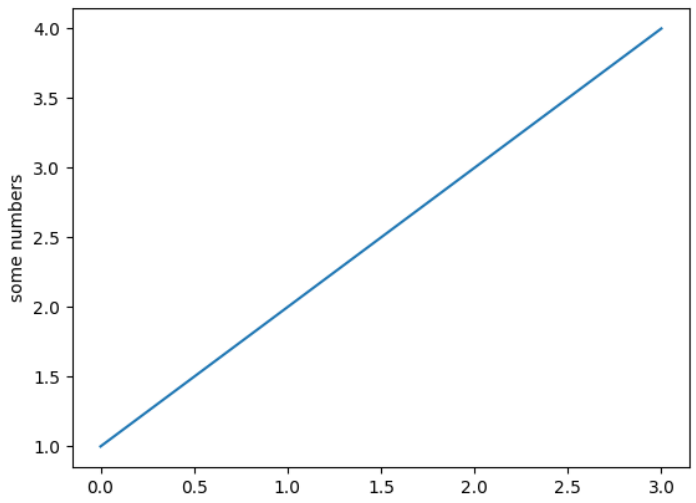


Pyplot tutorial

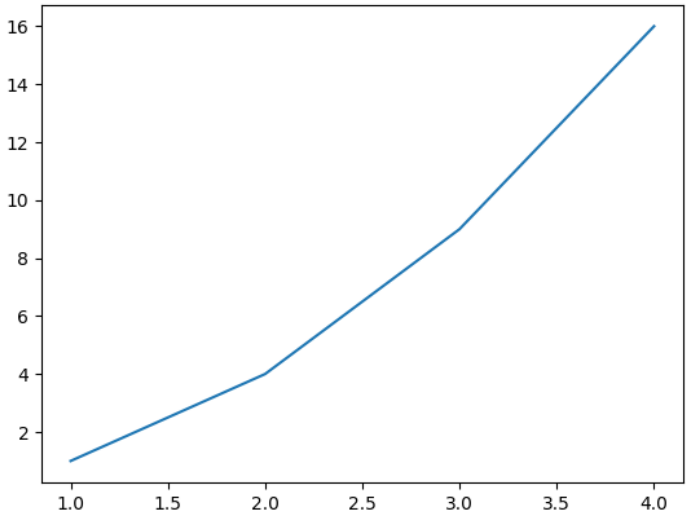
- matplotlib.pyplot
  - Một tập hợp các hàm làm cho matplotlib hoạt động giống như MATLAB.
  - Mỗi hàm pyplot thực hiện một số thay đổi đối với một hình (figure): tạo một hình, tạo vùng vẽ đồ thị trong một hình, vẽ một số đường trong vùng vẽ đồ thị, trang trí đồ thị bằng nhãn, v.v.

Pyplot

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()
```

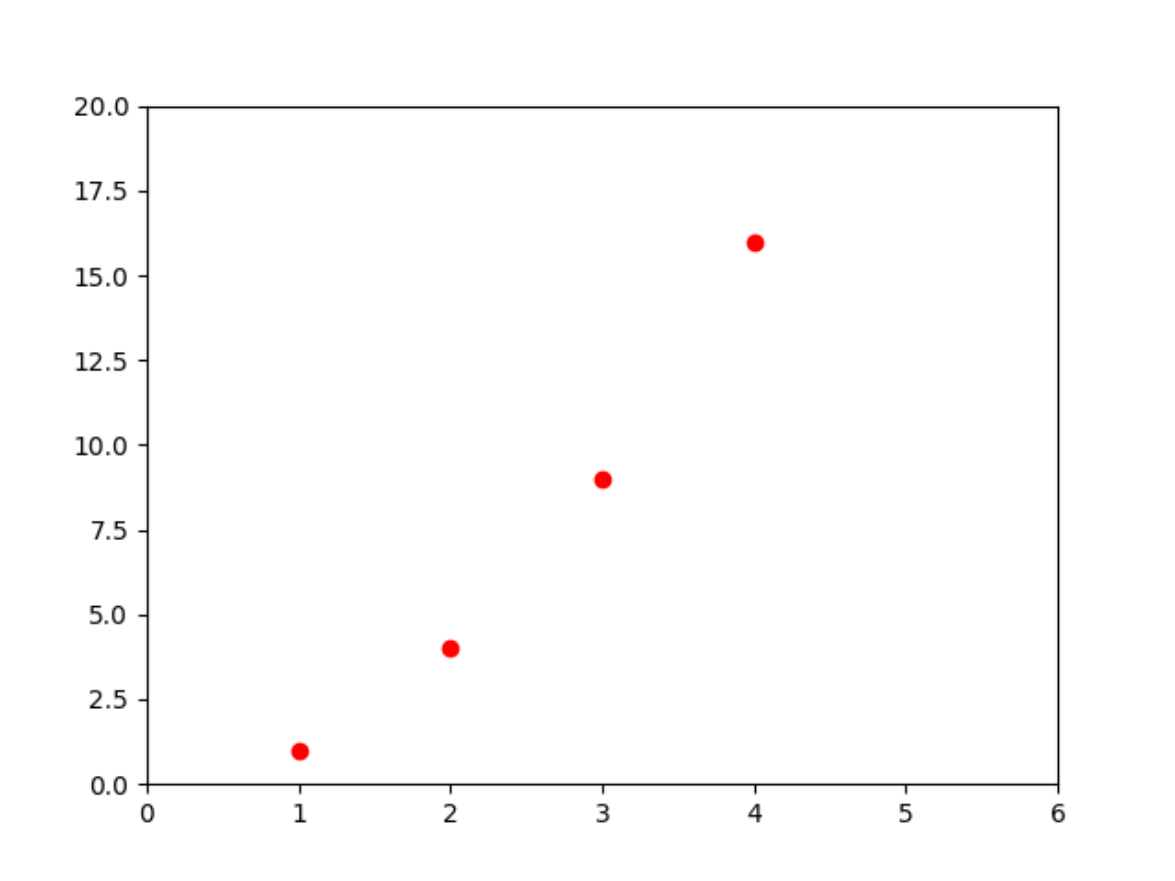



```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
```



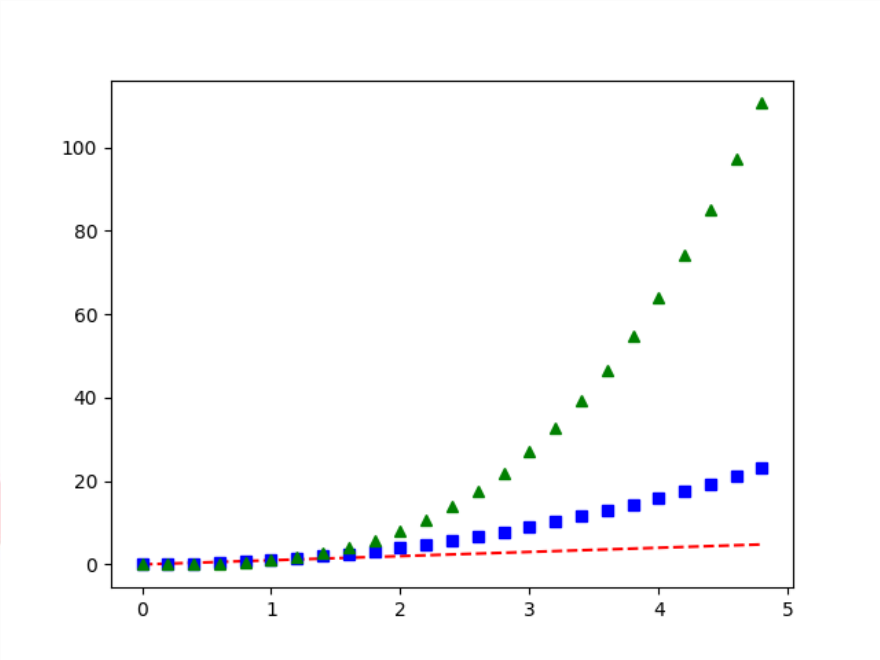
Pyplot - style of plot

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



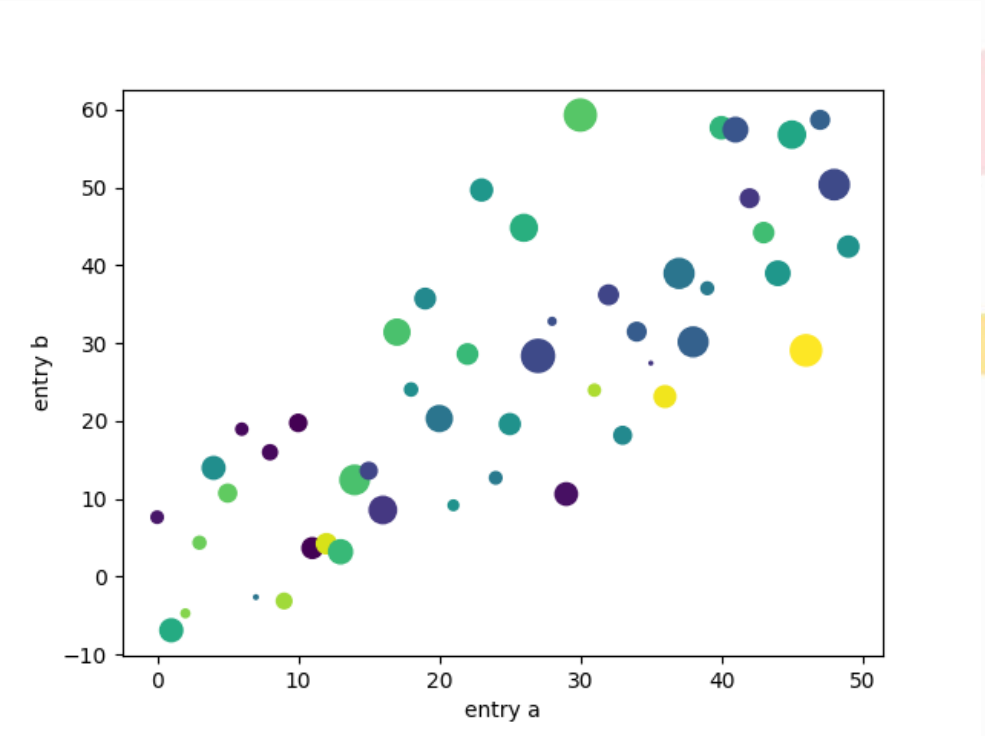
Pyplot - style of plot

```
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



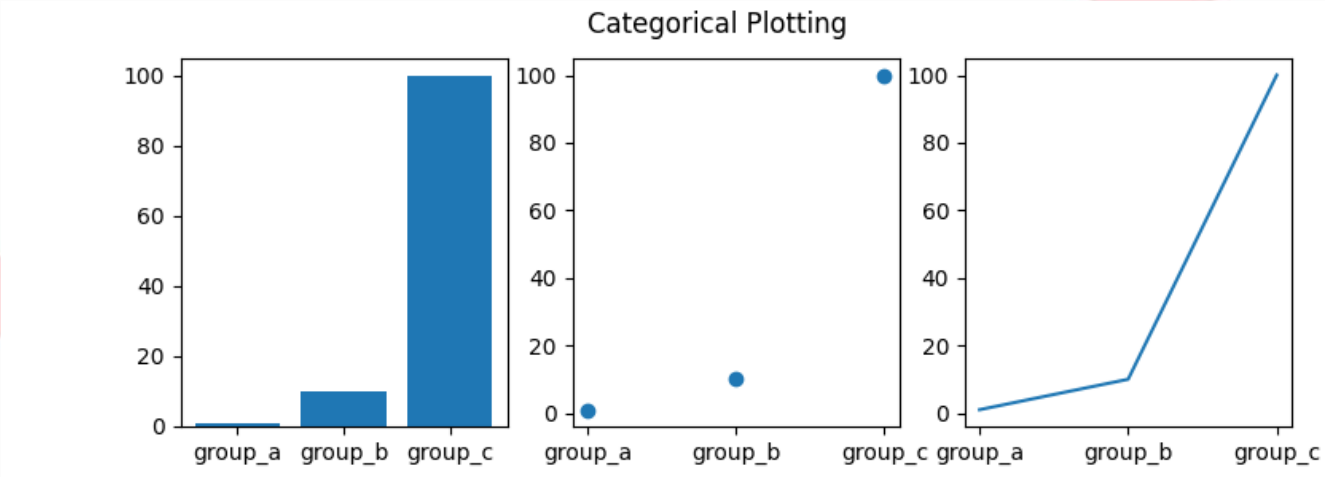
Plotting with keyword strings

```
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```



Plotting with categorical variables

```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(names, values)
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```



Controlling line properties

- Các dòng có nhiều thuộc : inewidth, dash style, antialiase … (matplotlib.lines.Line2D)

```
plt.plot(x, y, linewidth=2.0)
```

```
line, = plt.plot(x, y, '-')
line.set_antialiased(False) # turn off antialiasing
```

```
lines = plt.plot(x1, y1, x2, y2)
# use keyword args
plt.setp(lines, color='r', linewidth=2.0)
# or MATLAB style string value pairs
plt.setp(lines, 'color', 'r', 'linewidth', 2.0)
```

Line2D properties

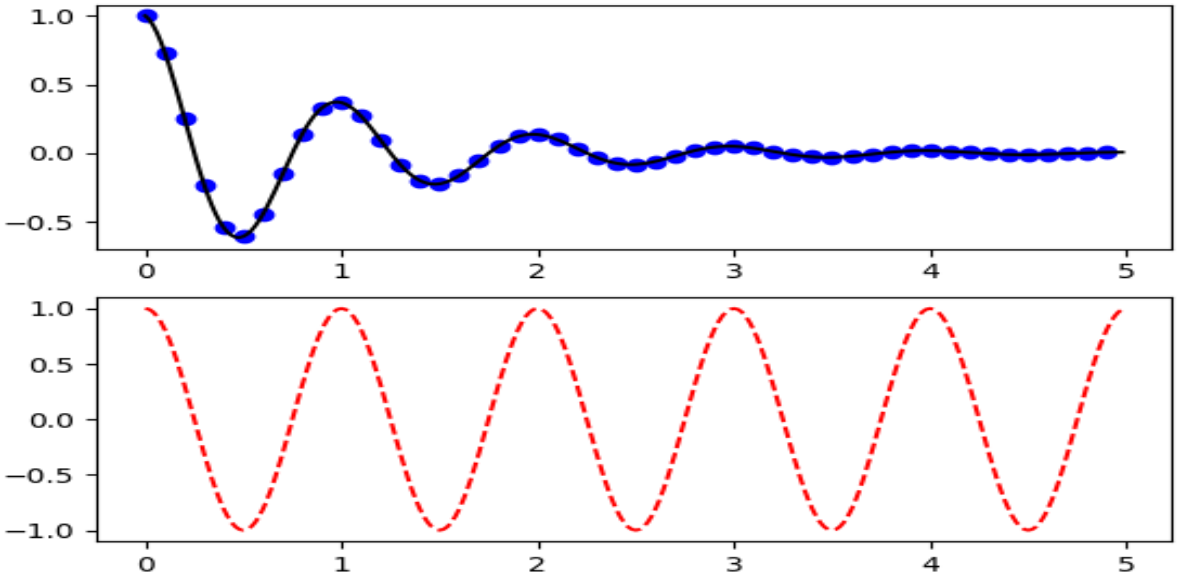
Property	Value Type
alpha	float
animated	[True | False]
antialiased or aa	[True | False]
clip_box	a matplotlib.transform.Bbox instance
clip_on	[True | False]
clip_path	a Path instance and a Transform instance, a Patch
color or c	any matplotlib color
contains	the hit testing function
dash_capstyle	['butt' | 'round' | 'projecting']
dash_joinstyle	['miter' | 'round' | 'bevel']
dashes	sequence of on/off ink in points
data	(np.array xdata, np.array ydata)
figure	a matplotlib.figure.Figure instance
label	any string
linestyle or ls	['-' | '--' | '-.' | ':' | 'steps' | ...]
linewidth or lw	float value in points

Line2D properties

Property	Value Type
marker	['+' | ',' | '.' | '1' | '2' | '3' | '4']
markeredgecolor or mec	any matplotlib color
markeredgewidth or mew	float value in points
markerfacecolor or mfc	any matplotlib color
markersize or ms	float
markevery	[None | integer | (startind, stride)]
picker	used in interactive line selection
pickradius	the line pick selection radius
solid_capstyle	['butt' | 'round' | 'projecting']
solid_joinstyle	['miter' | 'round' | 'bevel']
transform	a matplotlib.transforms.Transform instance
visible	[True | False]
xdata	np.array
ydata	np.array
zorder	any number

Working with multiple figures and axes

```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)
t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```



Working with multiple figures and axes

```
import matplotlib.pyplot as plt
plt.figure(1)            # the first figure
plt.subplot(211)             # the first subplot in the first figure
plt.plot([1, 2, 3])
plt.subplot(212)             # the second subplot in the first figure
plt.plot([4, 5, 6])


plt.figure(2)            # a second figure
plt.plot([4, 5, 6])       # creates a subplot(111) by default

plt.figure(1)            # figure 1 current; subplot(212) still current
plt.subplot(211)             # make subplot(211) in figure1 current
plt.title('Easy as 1, 2, 3') # subplot 211 title
```

Logarithmic and other nonlinear axes

- matplotlib.pyplot không chỉ hỗ trợ các thang đo trục tuyến tính mà còn hỗ trợ các thang đo logarit và logit.

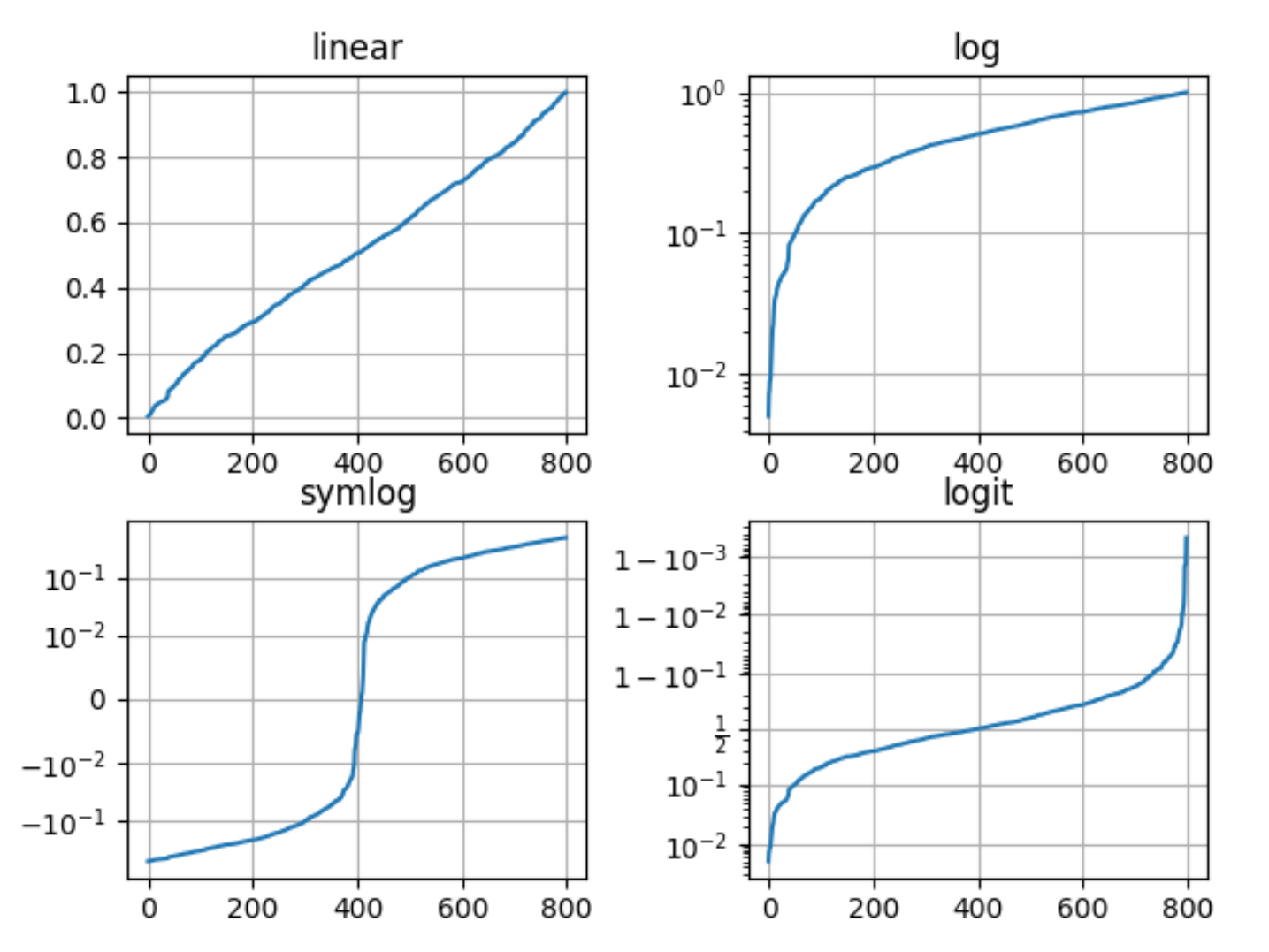
plt.xscale (‘log')

Logarithmic and other nonlinear axes

```
# Fixing random state for reproducibility
np.random.seed(19680801)
# make up some data in the open interval (0, 1)
y = np.random.normal(loc=0.5, scale=0.4, size=1000)
y = y[(y > 0) & (y < 1)]
y.sort()
x = np.arange(len(y))
# plot with various axes scales
plt.figure()
# linear
plt.subplot(221)
plt.plot(x, y)
plt.yscale('linear')
plt.title('linear')
plt.grid(True)
# log
plt.subplot(222)
plt.plot(x, y)
plt.yscale('log')
plt.title('log')
plt.grid(True)
```

```
# symmetric log
plt.subplot(223)
plt.plot(x, y - y.mean())
plt.yscale('symlog', linthresh=0.01)
plt.title('symlog')
plt.grid(True)

# logit
plt.subplot(224)
plt.plot(x, y)
plt.yscale('logit')
plt.title('logit')
plt.grid(True)
# Adjust the subplot layout, because the logit one
may take more space
# than usual, due to y-tick labels like "1 - 10^{-3}"
plt.subplots_adjust(top=0.92, bottom=0.08,
left=0.10, right=0.95, hspace=0.25,
                    wspace=0.35)

plt.show()
```

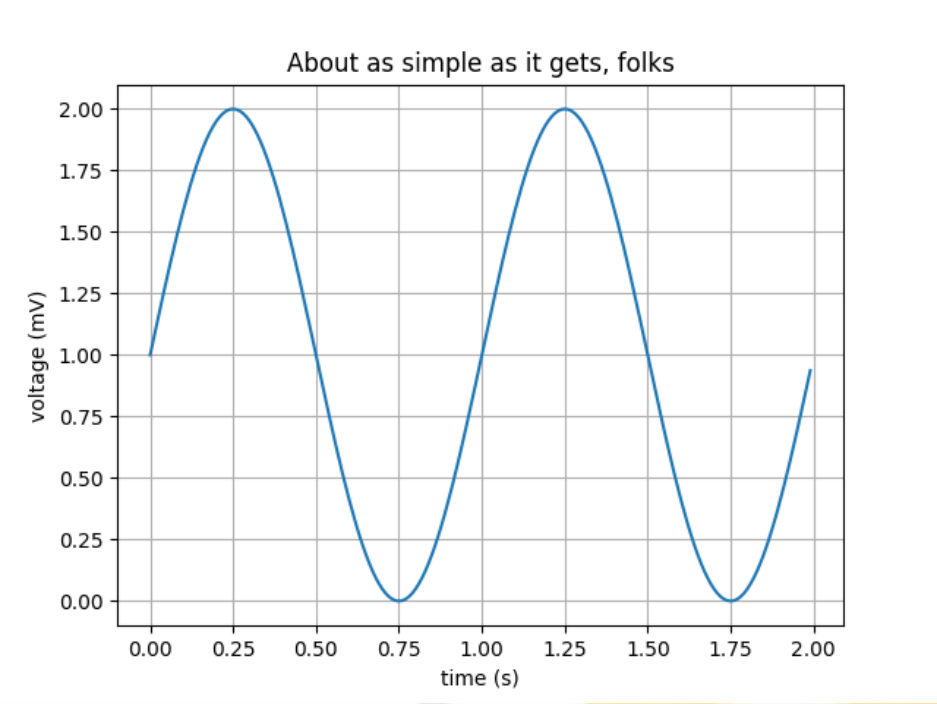
Logarithmic and other nonlinear axes



Sample plots in Matplotlib

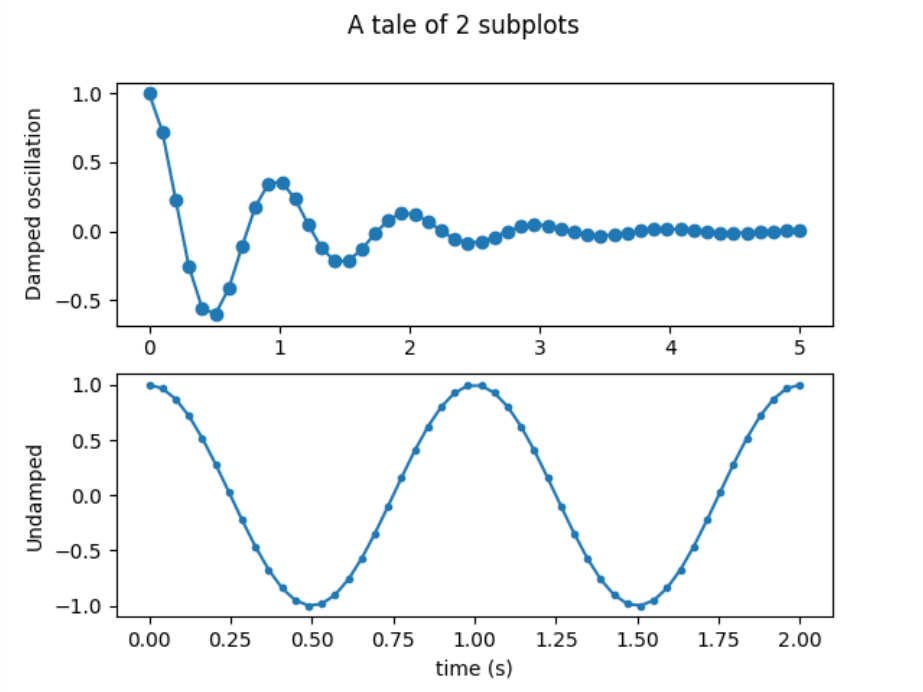
- Simple Plot

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

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)',
ylabel='voltage (mV)',
      title='About as simple as it
gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```



Sample plots in Matplotlib

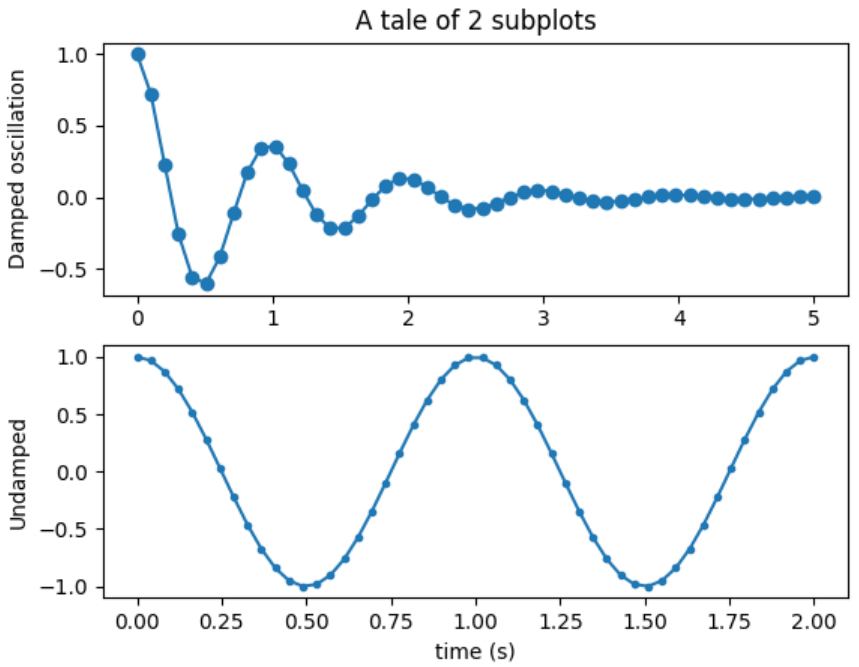
```
import numpy as np
import matplotlib.pyplot as plt
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)

y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

fig, (ax1, ax2) = plt.subplots(2, 1)
fig.suptitle('A tale of 2 subplots')

ax1.plot(x1, y1, 'o-')
ax1.set_ylabel('Damped oscillation')

ax2.plot(x2, y2, '.-')
ax2.set_xlabel('time (s)')
ax2.set_ylabel('Undamped')

plt.show()
```



Sample plots in Matplotlib

- Alternative Method For Creating Multiple Plots

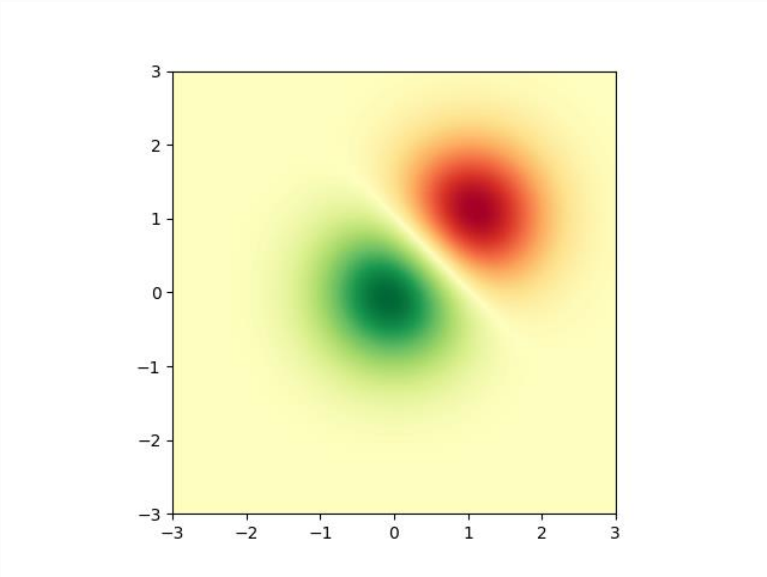
```
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)

y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'o-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')

plt.subplot(2, 1, 2)
plt.plot(x2, y2, '.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')

plt.show()
```



Sample plots in Matplotlib

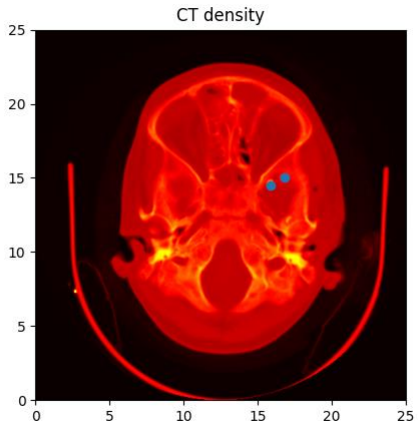
- Matplotlib có thể hiển thị hình ảnh (giả sử các kích thước chiều ngang cách đều nhau) bằng cách sử dụng hàm imshow ().

```
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import matplotlib.cbook as cbook
from matplotlib.path import Path
from matplotlib.patches import PathPatch

# Fixing random state for reproducibility
np.random.seed(19680801)
First we'll generate a simple bivariate normal distribution.
delta = 0.025
x = y = np.arange(-3.0, 3.0, delta)
X, Y = np.meshgrid(x, y)
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X - 1)**2 - (Y - 1)**2)
Z = (Z1 - Z2) * 2
fig, ax = plt.subplots()
im = ax.imshow(Z, interpolation='bilinear', cmap=cm.RdYlGn,
            origin='lower', extent=[-3, 3, -3, 3],
            vmax=abs(Z).max(), vmin=-abs(Z).max())

plt.show()
```



Sample plots in Matplotlib

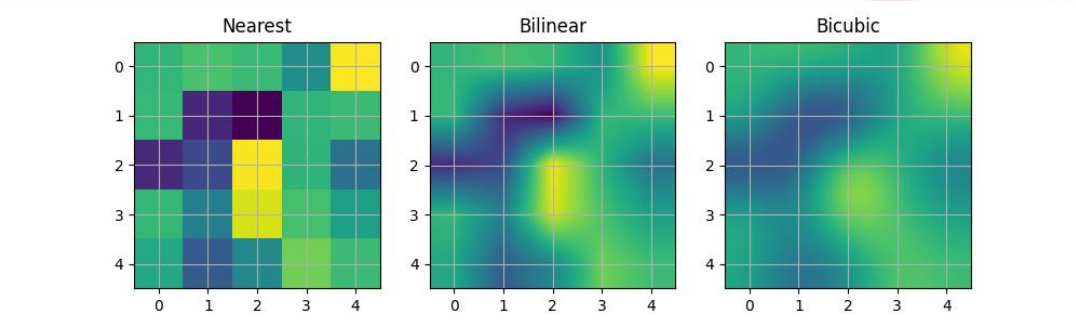
```
# A sample image
with cbook.get_sample_data('ada.png') as image_file:
    image = plt.imread(image_file)
fig, ax = plt.subplots()
ax.imshow(image)
ax.axis('off')  # clear x-axis and y-axis

# And another image
w, h = 512, 512
with cbook.get_sample_data('ct.raw.gz') as datafile:
    s = datafile.read()
A = np.frombuffer(s, np.uint16).astype(float).reshape((w, h))
A /= A.max()
fig, ax = plt.subplots()
extent = (0, 25, 0, 25)
im = ax.imshow(A, cmap=plt.cm.hot, origin='upper', extent=extent)
markers = [(15.9, 14.5), (16.8, 15)]
x, y = zip(*markers)
ax.plot(x, y, 'o')

ax.set_title('CT density')

plt.show()
```



Sample plots in Matplotlib

- Interpolating images

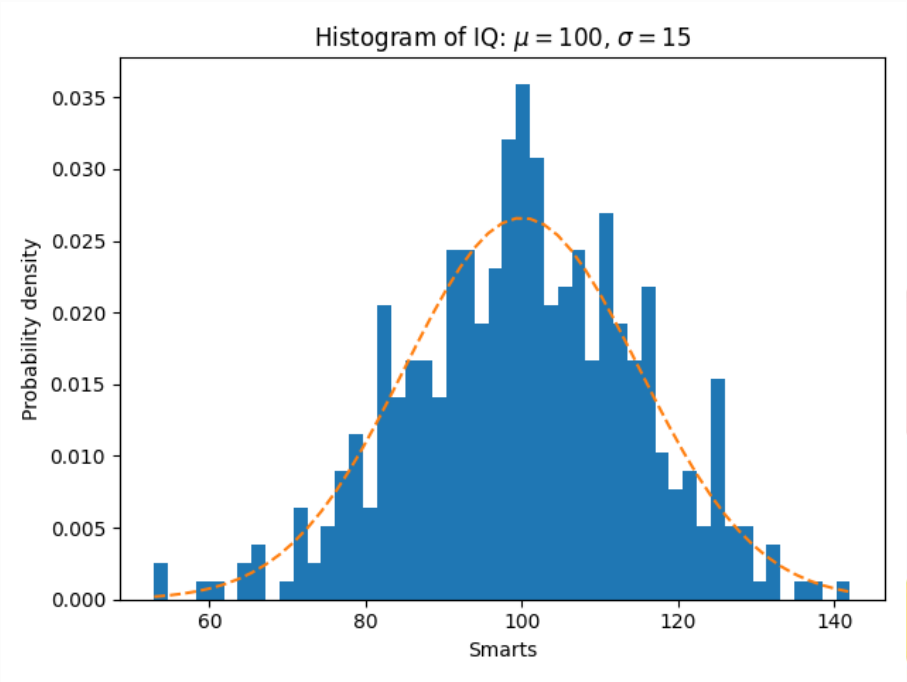
```
A = np.random.rand(5, 5)

fig, axs = plt.subplots(1, 3, figsize=(10, 3))
for ax, interp in zip(axs, ['nearest', 'bilinear',
'bicubic']):
    ax.imshow(A, interpolation=interp)
    ax.set_title(interp.capitalize())
    ax.grid(True)

plt.show()
```



Sample plots in Matplotlib

- Histograms

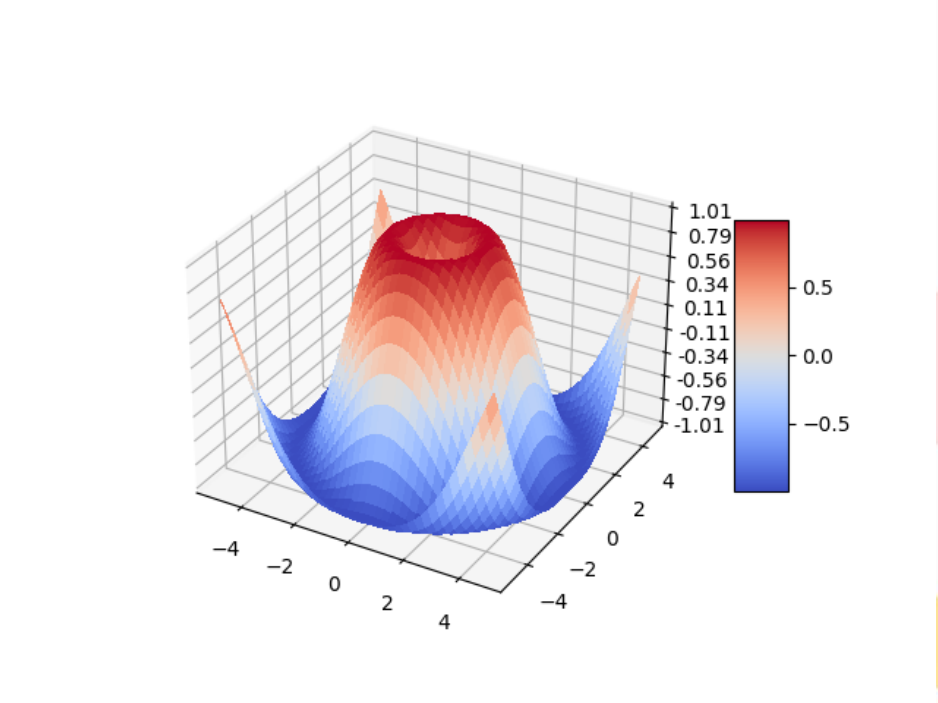
```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(19680801)
# example data
mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(437)
num_bins = 50
fig, ax = plt.subplots()
# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=True)
# add a 'best fit' line
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
     np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
ax.set_title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```



Sample plots in Matplotlib

- Three-dimensional plotting

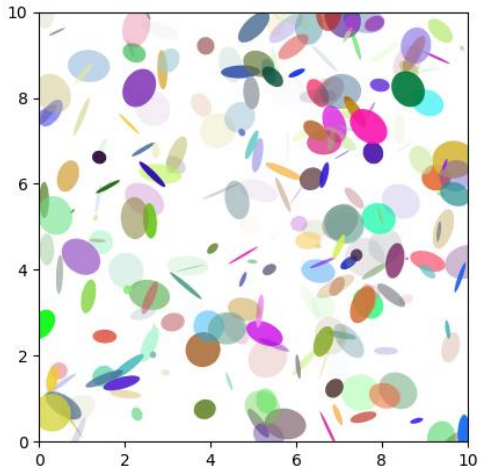
```
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
import numpy as np

fig, ax = plt.subplots(subplot_kw={"projection": "3d"})

# Make data.
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
# A StrMethodFormatter is used automatically
ax.zaxis.set_major_formatter('{x:.02f}')

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```



Sample plots in Matplotlib

- Ellipse Demo

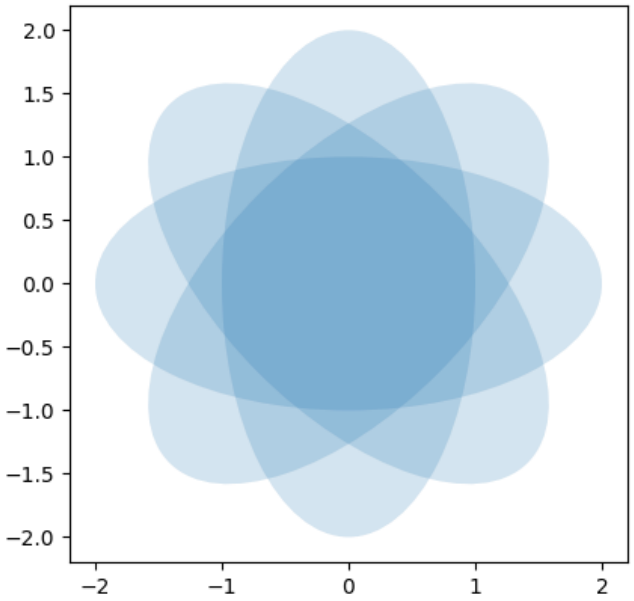
```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import Ellipse


# Fixing random state for reproducibility
np.random.seed(19680801)

NUM = 250

ells = [Ellipse(xy=np.random.rand(2) * 10,
            width=np.random.rand(), height=np.random.rand(),
            angle=np.random.rand() * 360)
        for i in range(NUM)]

fig, ax = plt.subplots(subplot_kw={'aspect': 'equal'})
for e in ells:
    ax.add_artist(e)
    e.set_clip_box(ax.bbox)
    e.set_alpha(np.random.rand())
    e.set_facecolor(np.random.rand(3))

ax.set_xlim(0, 10)
ax.set_ylim(0, 10)

plt.show()
```



Sample plots in Matplotlib

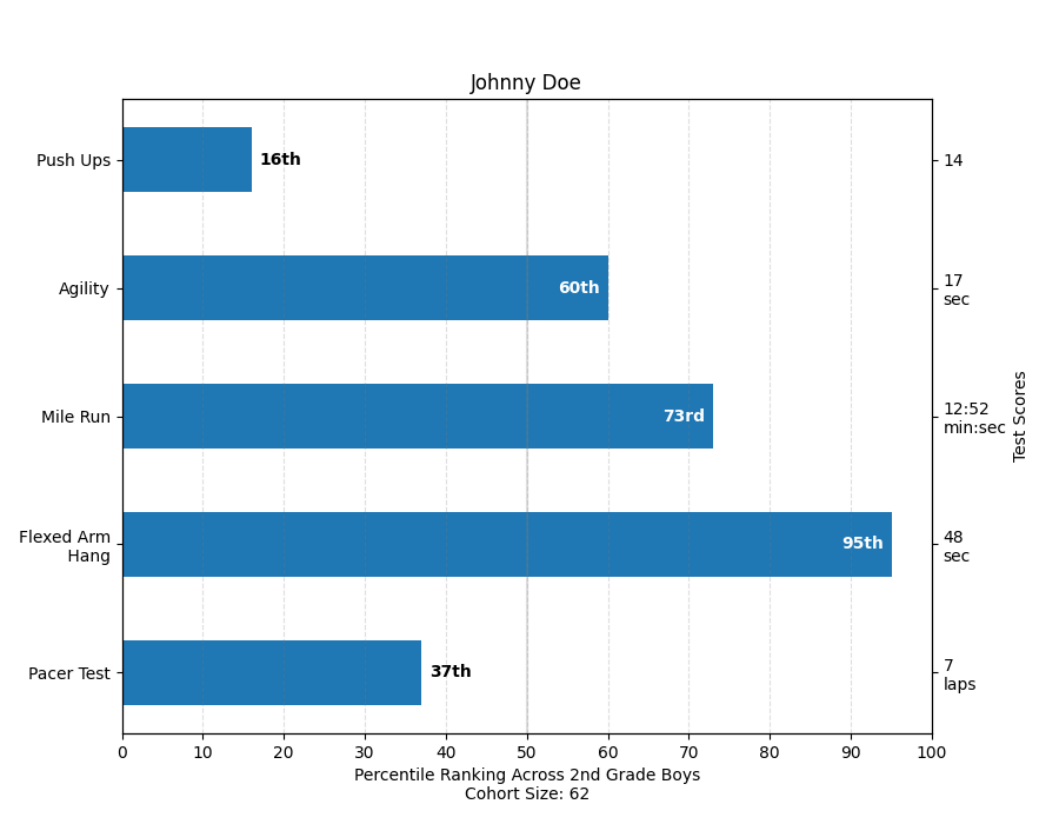
- Ellipse Rotated

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import Ellipse

angle_step = 45  # degrees
angles = np.arange(0, 360, angle_step)

ax = plt.subplot(aspect='equal')

for angle in angles:
    ellipse = Ellipse((0, 0), 4, 2, angle=angle, alpha=0.1)
    ax.add_artist(ellipse)

plt.xlim(-2.2, 2.2)
plt.ylim(-2.2, 2.2)

plt.show()
```



Sample plots in Matplotlib

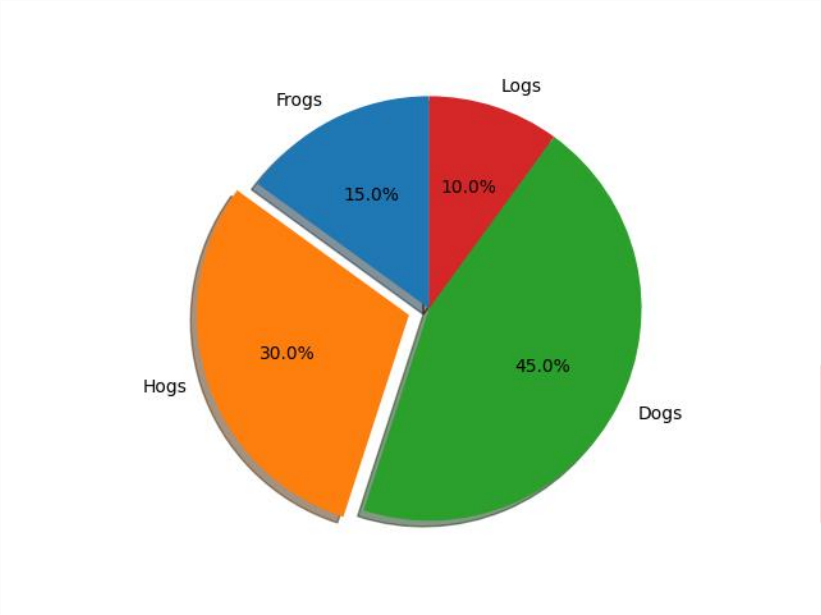
- Bar charts (code: https://matplotlib.org/stable/gallery/statistics/barchart_demo.html)



Sample plots in Matplotlib

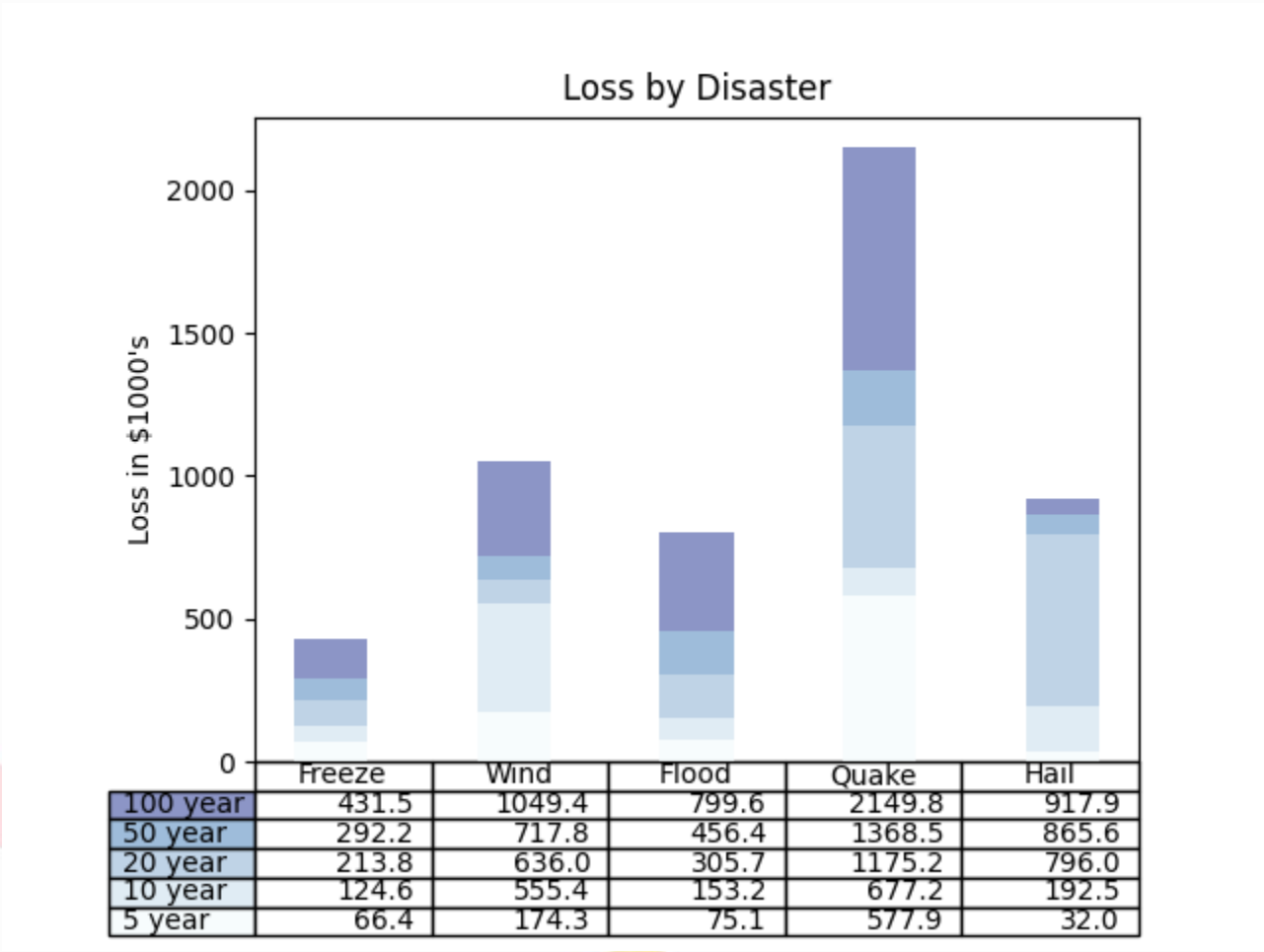
- Basic pie chart

```
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-
clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a
circle.

plt.show()
```



Sample plots in Matplotlib

- Table Demo (code: https://matplotlib.org/stable/gallery/misc/table_demo.html)

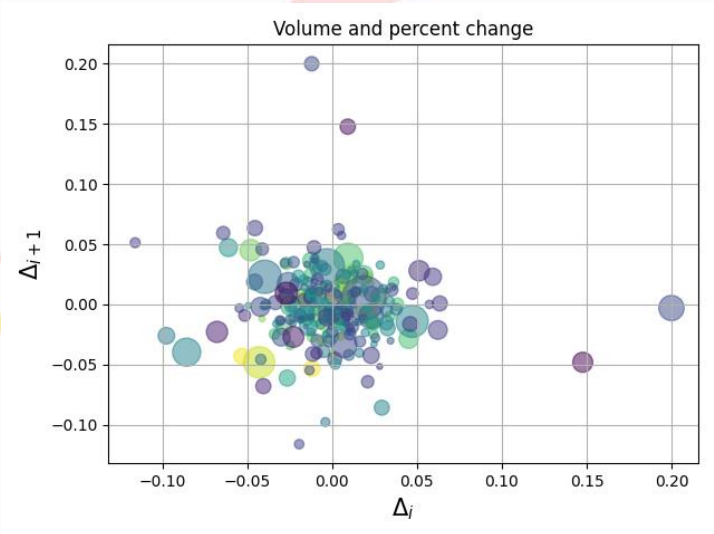


	Freeze	Wind	Flood	Quake	Hail
100 year	431.5	1049.4	799.6	2149.8	917.9
50 year	292.2	717.8	456.4	1368.5	865.6
20 year	213.8	636.0	305.7	1175.2	796.0
10 year	124.6	555.4	153.2	677.2	192.5
5 year	66.4	174.3	75.1	577.9	32.0

Sample plots in Matplotlib

- Scatter plots

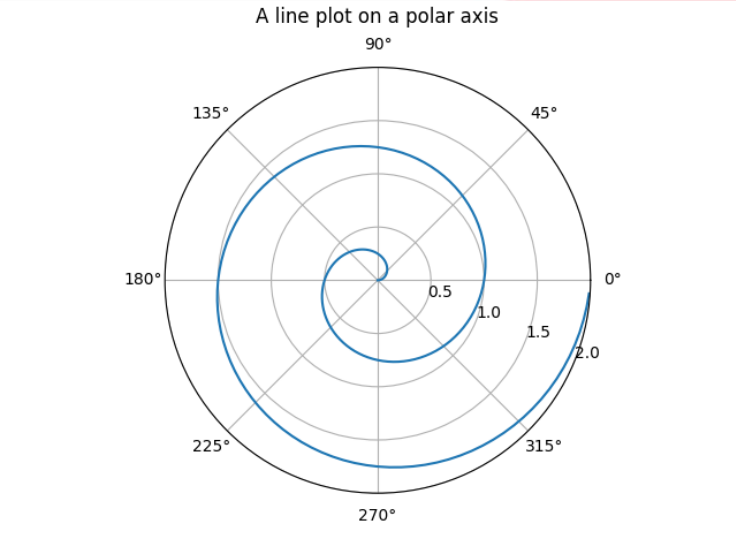
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cbook as cbook
# Load a numpy record array from yahoo csv data with fields date, open, close,
# volume, adj_close from the mpl-data/example directory. The record array
# stores the date as an np.datetime64 with a day unit ('D') in the date column.
price_data = (cbook.get_sample_data('goog.npz', np_load=True)['price_data']
              .view(np.recarray))
price_data = price_data[-250:]  # get the most recent 250 trading days
delta1 = np.diff(price_data.adj_close) / price_data.adj_close[:-1]
# Marker size in units of points^2
volume = (15 * price_data.volume[:-2] / price_data.volume[0])**2
close = 0.003 * price_data.close[:-2] / 0.003 * price_data.open[:-2]
fig, ax = plt.subplots()
ax.scatter(delta1[:-1], delta1[1:], c=close, s=volume, alpha=0.5)
ax.set_xlabel(r'$\Delta_i$', fontsize=15)
ax.set_ylabel(r'$\Delta_{i+1}$', fontsize=15)
ax.set_title('Volume and percent change')
ax.grid(True)
fig.tight_layout()
plt.show()
```



Sample plots in Matplotlib

- Polar plots

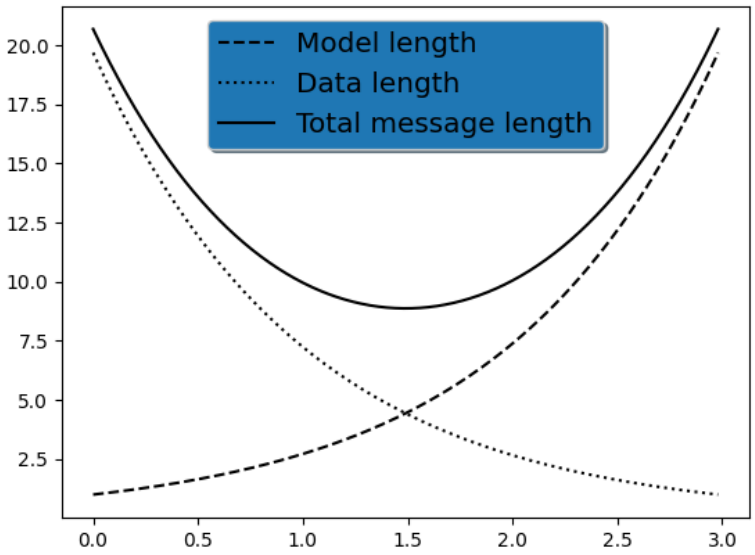
```
import numpy as np
import matplotlib.pyplot as plt


r = np.arange(0, 2, 0.01)
theta = 2 * np.pi * r

fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
ax.plot(theta, r)
ax.set_rmax(2)
ax.set_rticks([0.5, 1, 1.5, 2])  # Less radial ticks
ax.set_rlabel_position(-22.5)  # Move radial labels away from plotted line
ax.grid(True)

ax.set_title("A line plot on a polar axis", va='bottom')
plt.show()
```



Sample plots in Matplotlib

- Legends

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

# Make some fake data.
a = b = np.arange(0, 3, .02)
c = np.exp(a)
d = c[::-1]

# Create plots with pre-defined labels.
fig, ax = plt.subplots()
ax.plot(a, c, 'k--', label='Model length')
ax.plot(a, d, 'k:', label='Data length')
ax.plot(a, c + d, 'k', label='Total message length')

legend = ax.legend(loc='upper center', shadow=True,
fontsize='x-large')

# Put a nicer background color on the legend.
legend.get_frame().set_facecolor('C0')
```



Sample plots in Matplotlib

- Subplot example

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

np.random.seed(19680801)
data = np.random.randn(2, 100)

fig, axs = plt.subplots(2, 2, figsize=(5, 5))
axs[0, 0].hist(data[0])
axs[1, 0].scatter(data[0], data[1])
axs[0, 1].plot(data[0], data[1])
axs[1, 1].hist2d(data[0], data[1])

plt.show()
```