Python初級數據分析員證書

（五）進階Python數據分析及可視化技巧

10. Matplotlib套件
Part 3 - Review

10. Matplotlib – Part 3 - Review

Review Summary

- Basic concept
- Pyplot
- Axes plot
- Customizing style
- Legend
- Color/colormap
- Complex and semantic figure composition
- Text in Matplotlib

Matplotlib Interface

Matplotlib has two major application interfaces, or styles of using the library:

- An explicit "Axes" interface that uses methods on a Figure or Axes object to create other Artists, and build a visualization step by step.
- An implicit "pyplot" interface that keeps track of the last Figure and Axes created, and adds Artists to the object it thinks the user wants.

Figure Structure

The figure categorized into few parts

- Figure: a canvas to draw plots, can contain multiple subplot
- Axes: can contain several Axes, and each of them contain their own x-label and y-label
- Axis: number of line or object for graph limit
- x-label, y-label
- title

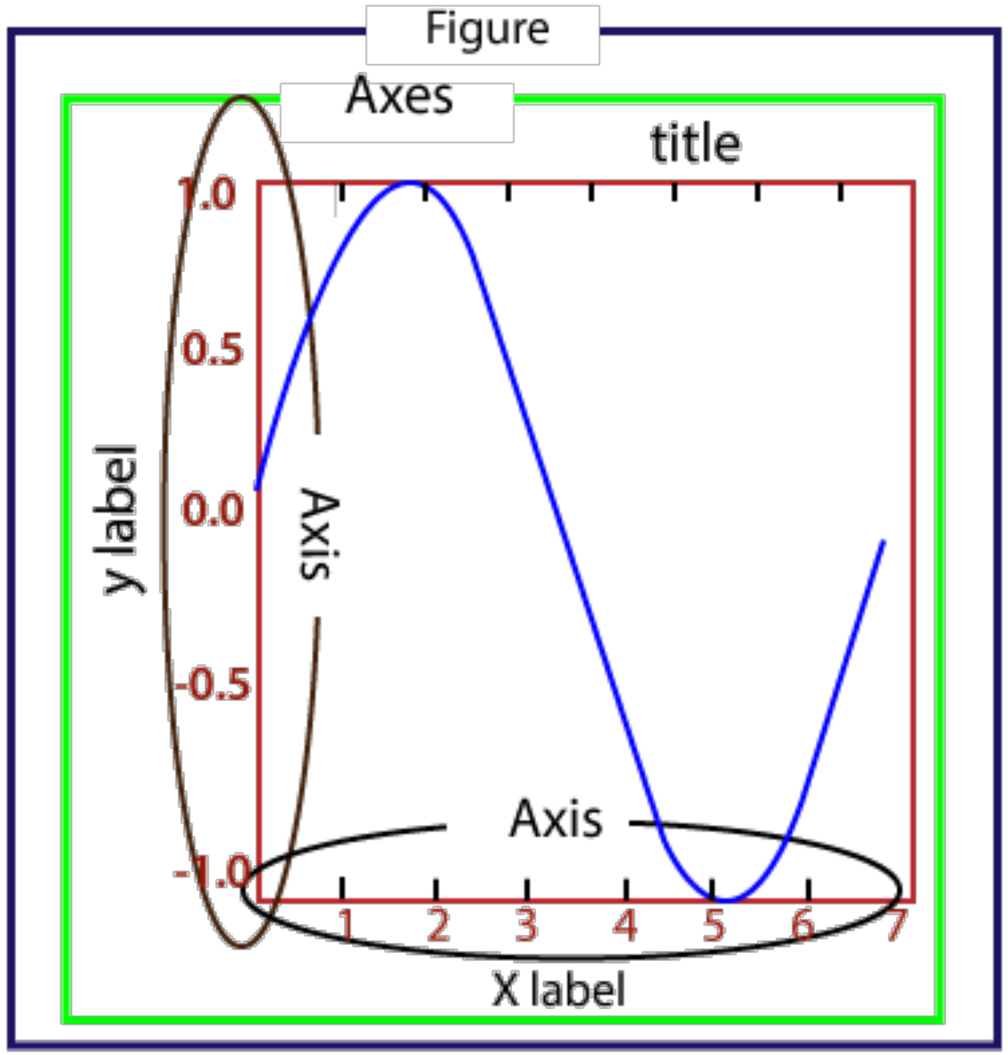
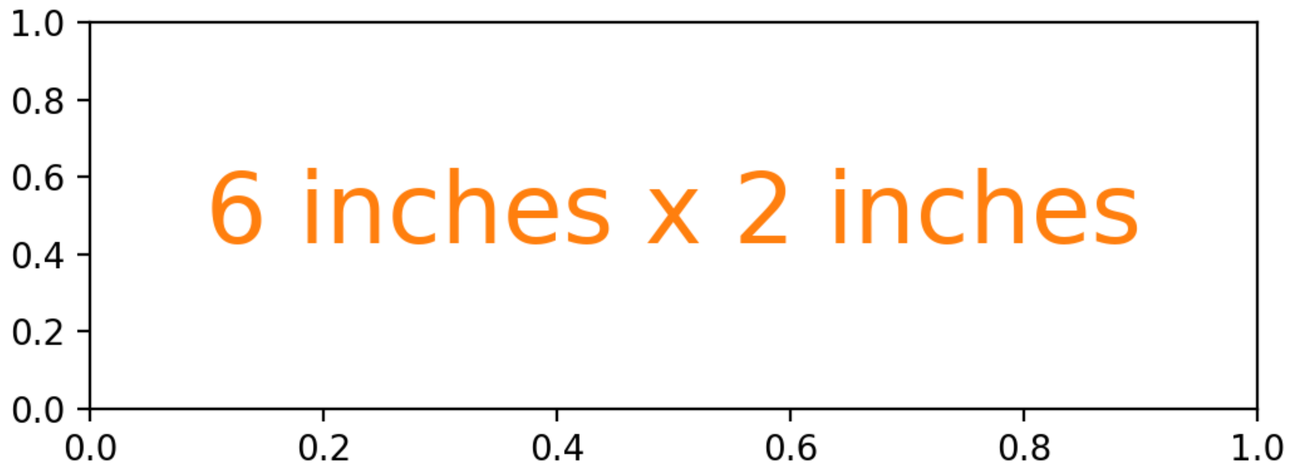


Figure Size

```
import matplotlib.pyplot as plt

text_kwargs = dict(ha='center', va='center', fontsize=28, color='C1')

plt.subplots(figsize=(6, 2))
plt.text(0.5, 0.5, '6 inches x 2 inches', **text_kwargs)
plt.show()
```



Setup

Setup title, axes label, and legend

To change your plot title and axes labels, you can follow one of the following approaches, depending of which container of which you want to make use:

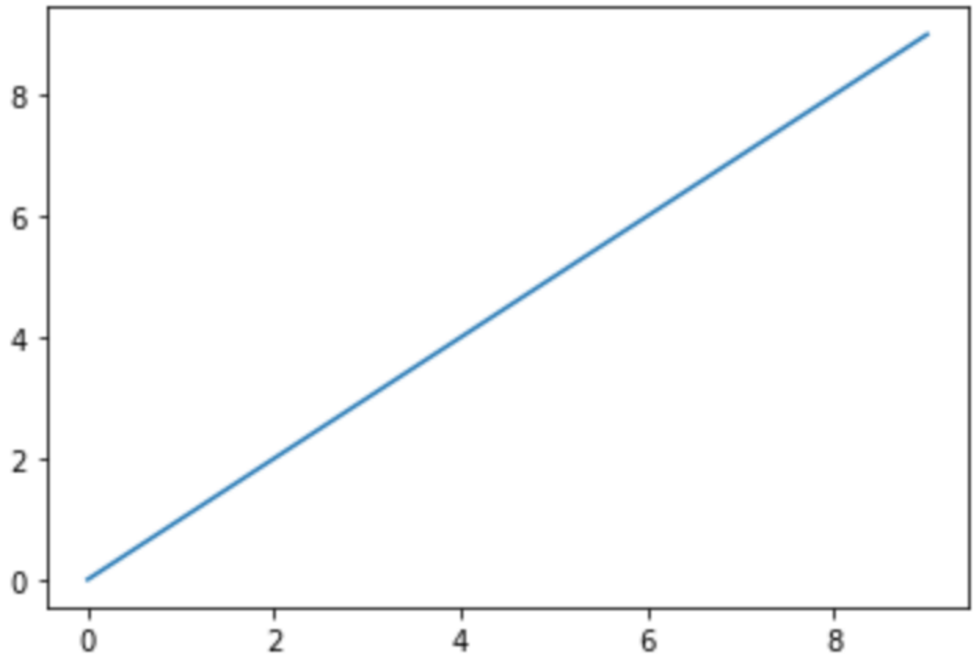
- The easiest way to set these things right is by using ax.set(title="A title", xlabel="x", ylabel="y") or ax.set_xlim(), ax.set_ylim() or ax.set_title().
- If you want to work with the figure, you might also resort to fig.suptitle() to add a title to your plot.
- If you're making use of the default settings that the package has to offer, you might want to use plt.title(), plt.xlabel(), plt.ylabel()

Use ax.legend() to denote the items in colors

Line Chart

```
import matplotlib.pyplot as plt

data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

plt.plot(data)
```

```
[<matplotlib.lines.Line2D at 0x10ea5f850>]
```



The core and basic elements of a chart

Simple Line Chart

```
import matplotlib.pyplot as plt

x = ['Dec01,2022','Dec02,2022','Dec05,2022','Dec06,2022','Dec07,2022']
y = [4076.57, 4071.70, 3998.84, 3941.26, 3933.92]
plt.plot(x, y)

plt.ylabel('closing index')
plt.title('S&P 500 index')
plt.show()
```

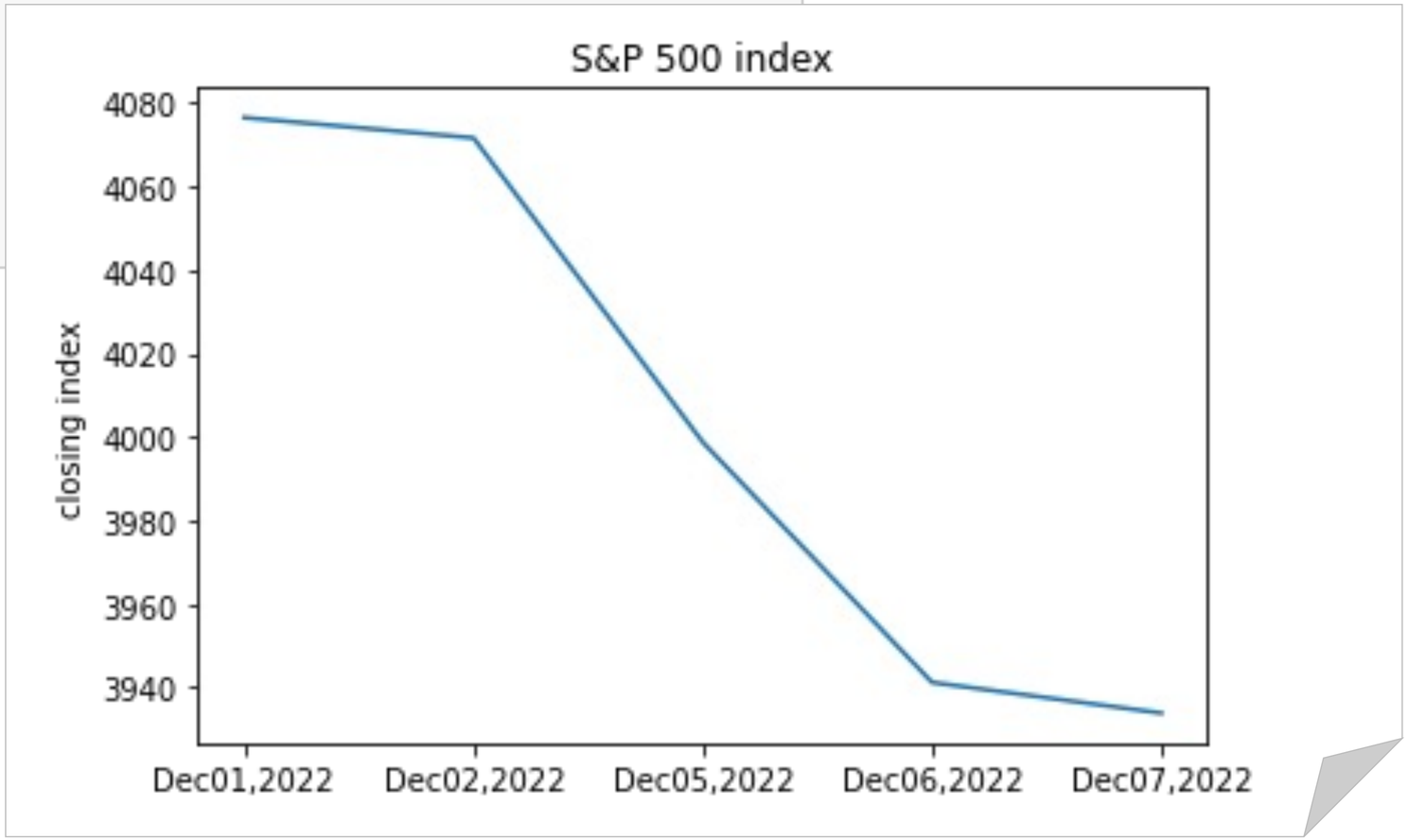


Figure and subplot - 1

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
```

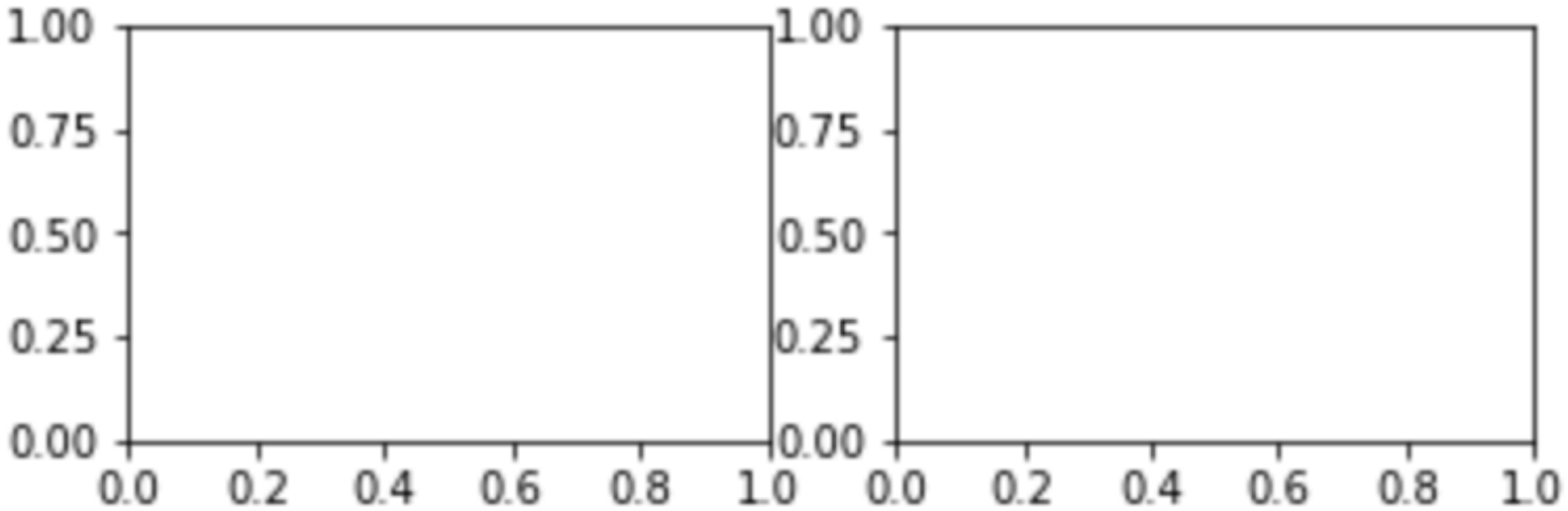


Figure and subplot - 2

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(2, 1, 1)
ax2 = fig.add_subplot(2, 1, 2)
```

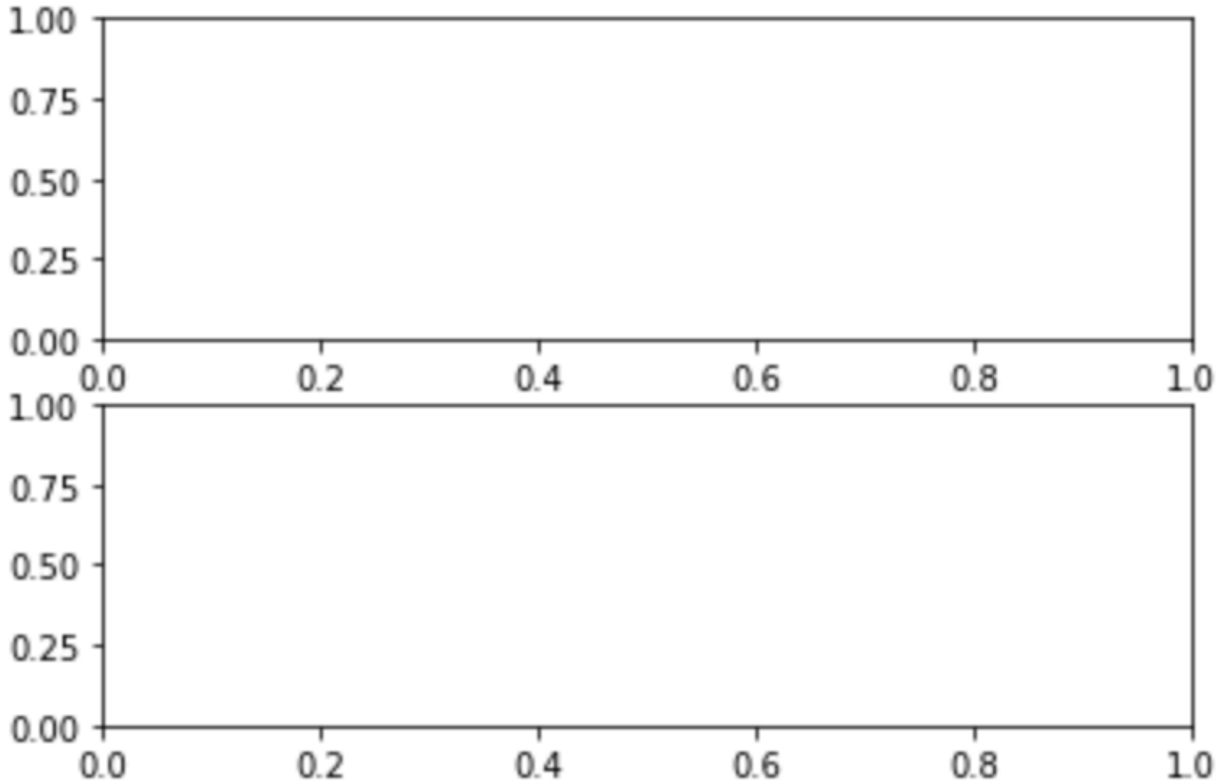
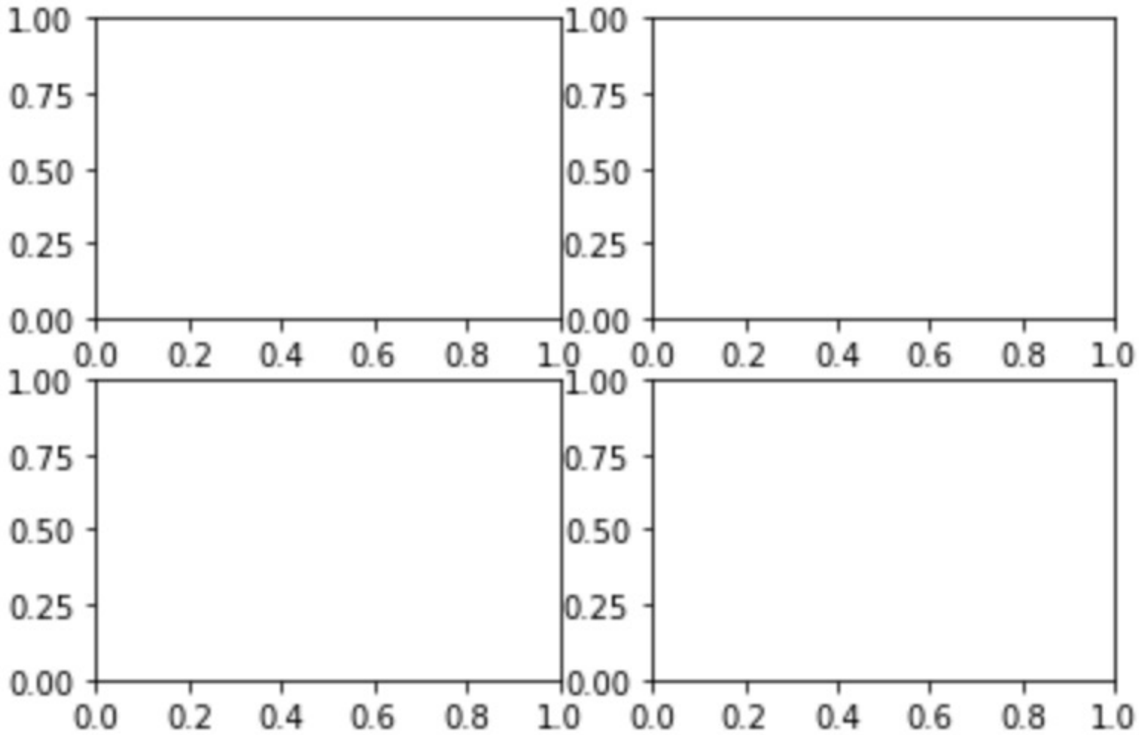


Figure and subplot - 3

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax3 = fig.add_subplot(2, 2, 4)
```



Pyplot.subplot

Pyplot.subplots option

Argument	Description
`nrows`	Number of rows of subplots
`ncols`	Number of columns of subplots
`sharex`	All subplots should use the same x-axis ticks (adjusting the `xlim` will affect all subplots)
`sharey`	All subplots should use the same y-axis ticks (adjusting the `ylim` will affect all subplots)
`subplot_kw`	Dict of keywords passed to `add_subplot` call used to create each subplot
`**fig_kw`	Additional keywords to `subplots` are used when creating the figure, such as `plt.subplots(2, 2, figsize=(8, 6))`

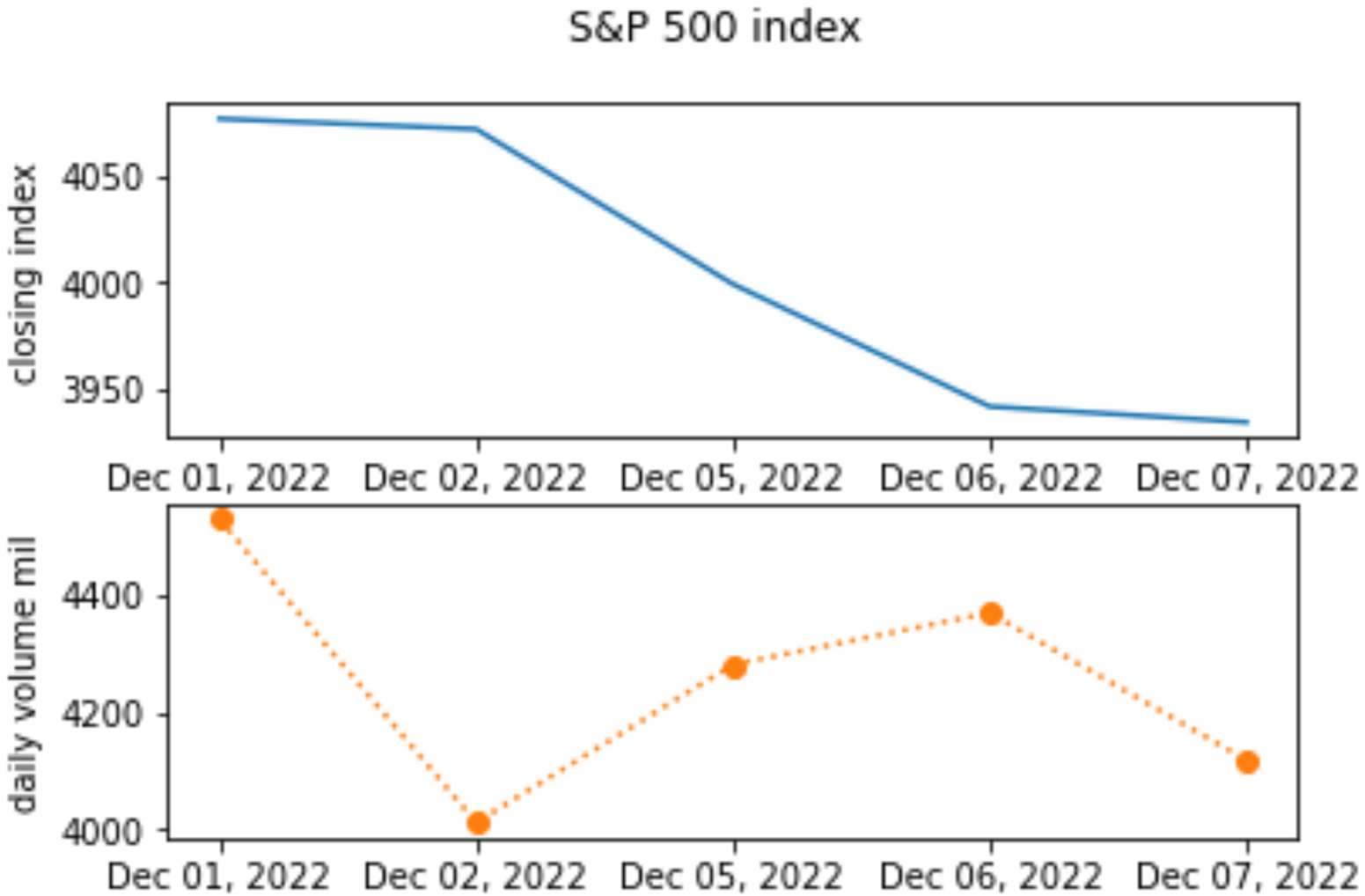
Pyplot.subplot

Line Chart with Subplot

```
import matplotlib.pyplot as plt

dates = ['Dec 01, 2022', 'Dec 02, 2022', 'Dec 05, 2022',
         'Dec 06, 2022', 'Dec 07, 2022']
indx = [4076.57, 4071.70, 3998.84, 3941.26, 3933.92]
vol = [4527, 4012, 4280, 4368, 4118]

plt.figure()
plt.subplot(211)
plt.plot( dates, indx)
plt.ylabel('closing index')

plt.subplot(212)
plt.plot( dates, vol, 'tab:orange',
         marker = 'o', linestyle = 'dotted')
plt.ylabel('daily volume mil')

plt.suptitle('S&P 500 index')
plt.show()
```

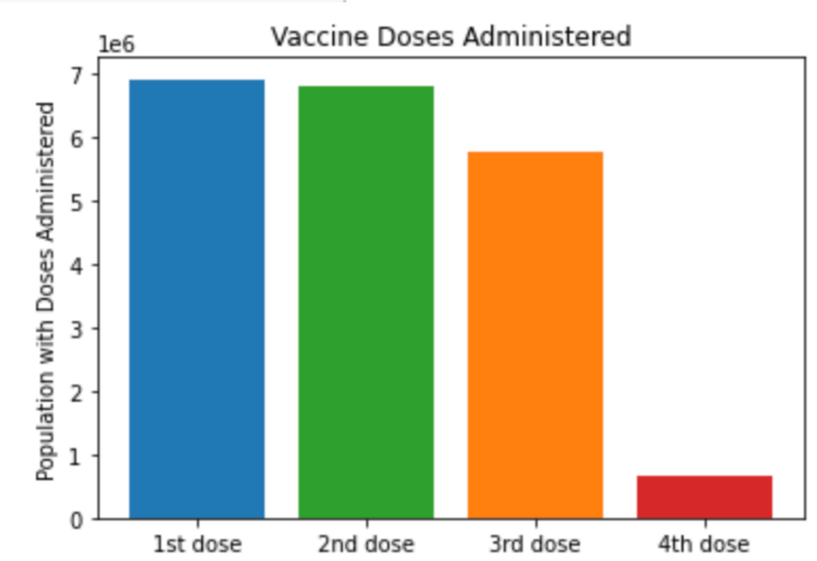
Pyplot.subplot

Line Chart with Subplot



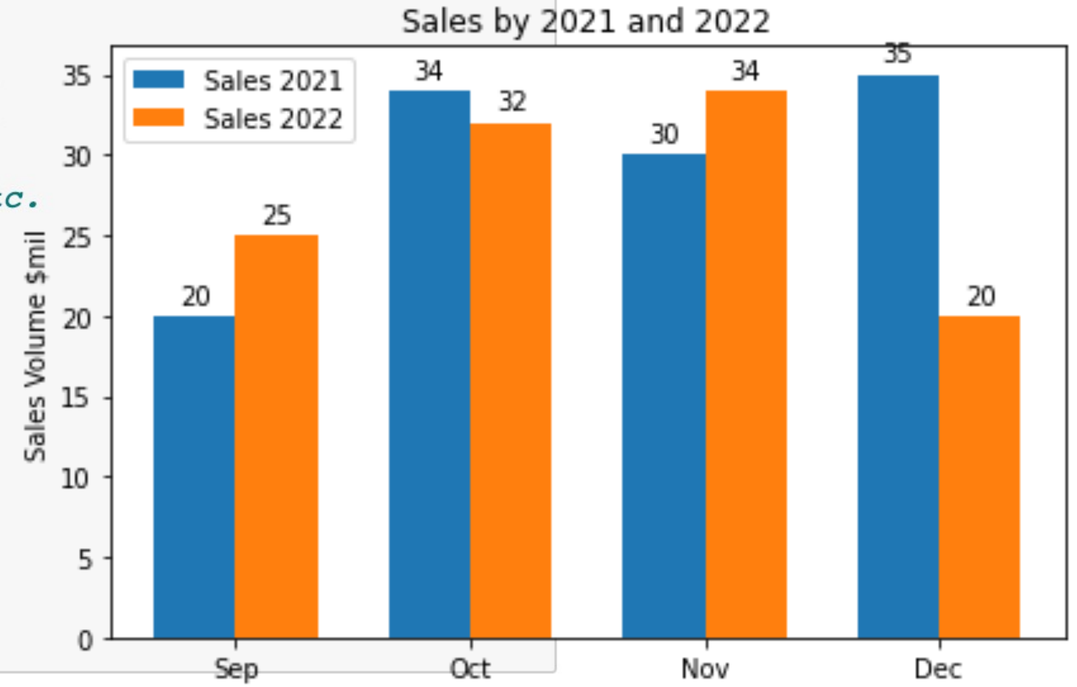
Bar Chart

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

vaccine = ['1st dose', '2nd dose', '3rd dose', '4th dose']
counts = [6904243, 6779247, 5764953, 675118]
bar_colors = ['tab:blue', 'tab:green', 'tab:orange', 'tab:red']

ax.bar(vaccine, counts, label=bar_labels, color=bar_colors)

ax.set_ylabel('Population with Doses Administered')
ax.set_title('Vaccine Doses Administered')

plt.show()
```



Grouped bar chart with label and legend

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


labels = ['Sep', 'Oct', 'Nov', 'Dec']
sales_2021 = [20, 34, 30, 35]
sales_2022 = [25, 32, 34, 20]

x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, sales_2021, width, label='Sales 2021')
rects2 = ax.bar(x + width/2, sales_2022, width, label='Sales 2022')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Sales Volume $mil')
ax.set_title('Sales by 2021 and 2022')
ax.set_xticks(x, labels)
ax.legend()

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

plt.show()
```

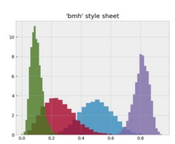


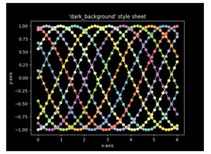
Preset

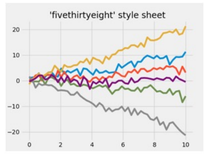
Plotting with preset Style Sheet

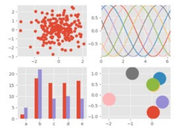
- There are numbers of predefined style sheet in Matplotlib
- plt.style.use()

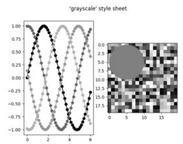
Style sheets

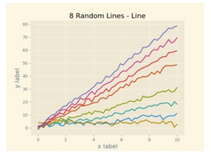

Bayesian Methods for Hackers style sheet


Dark background style sheet


FiveThirtyEight style sheet


ggplot style sheet


Grayscale style sheet


Solarized Light stylesheet

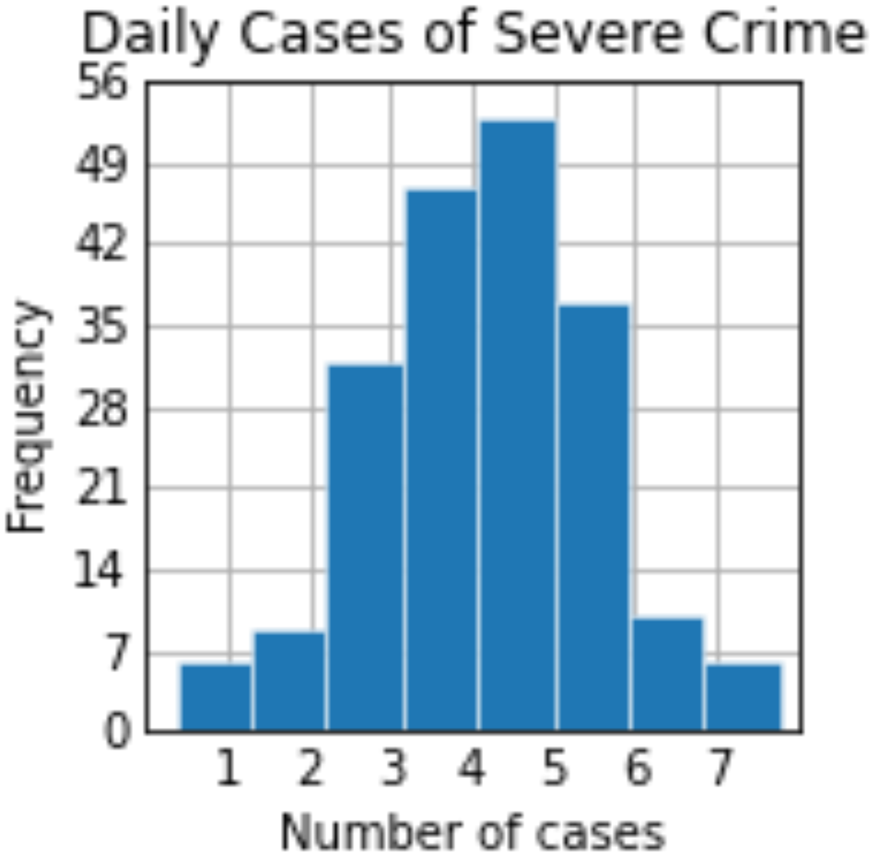

Style sheets reference

```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-
nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight',
'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind',
'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid', 'seaborn-dee
p', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-paste
l', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white',
'seaborn-whitegrid', 'tableau-colorblind10']
```

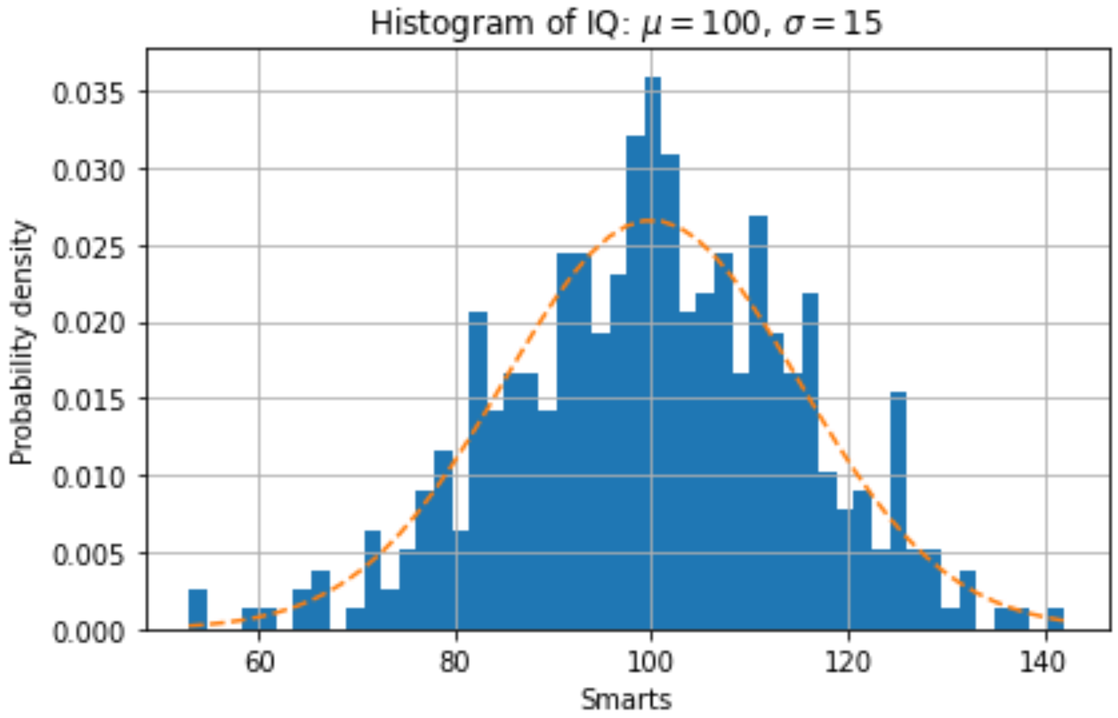
Histogram

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

plt.style.use('_mpl-gallery')

# make data
np.random.seed(1)
x = 4 + np.random.normal(0, 1.5, 200)

# plot:
fig, ax = plt.subplots()

ax.hist(x, bins=8, linewidth=0.5, edgecolor="white")

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 56), yticks=np.linspace(0, 56, 9))

ax.set_title('Daily Cases of Severe Crime')
ax.set_xlabel('Number of cases')
ax.set_ylabel('Frequency')
plt.show()
```



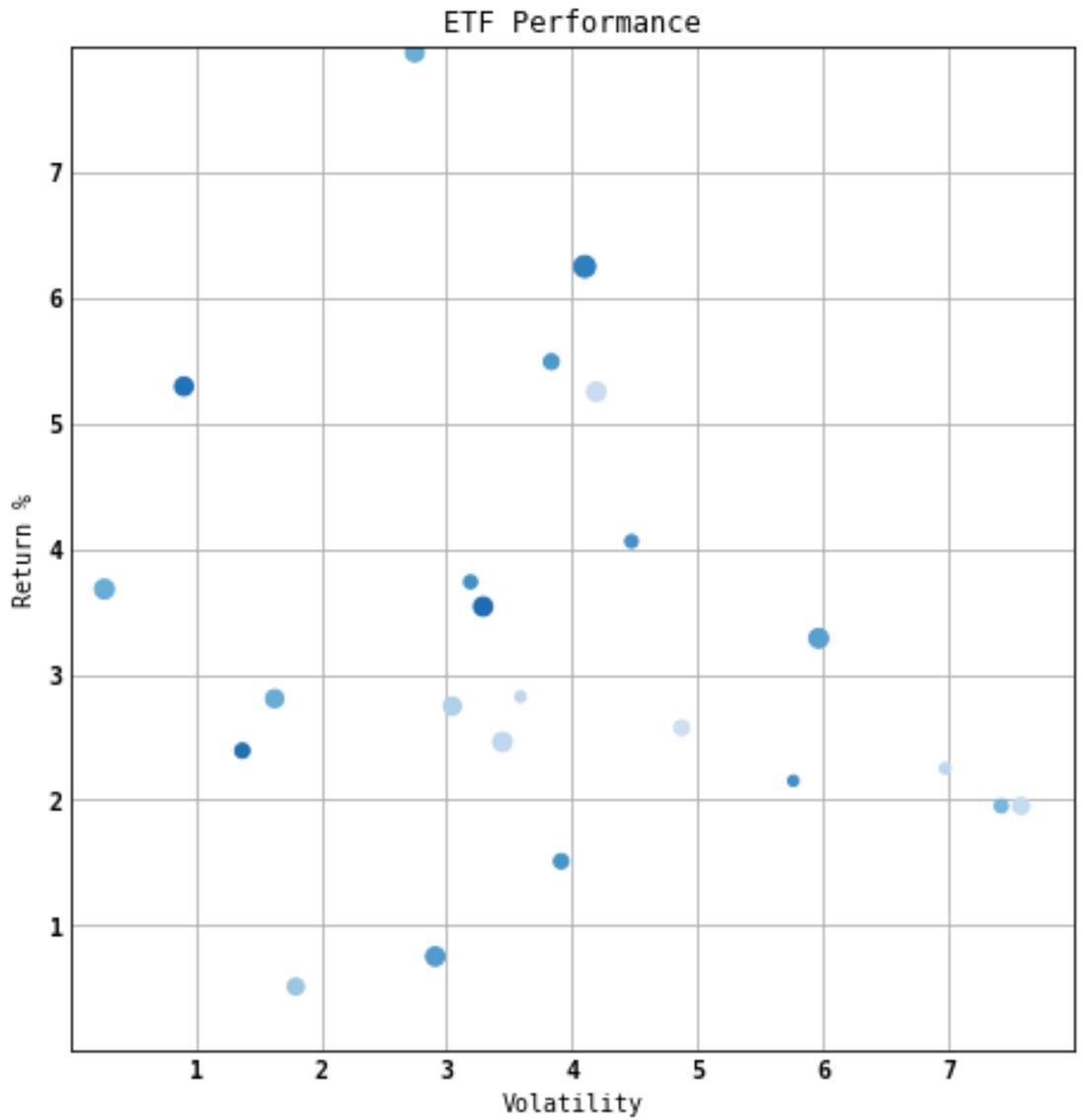
Histogram

```
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.grid()
ax.set_title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

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

plt.show()
```



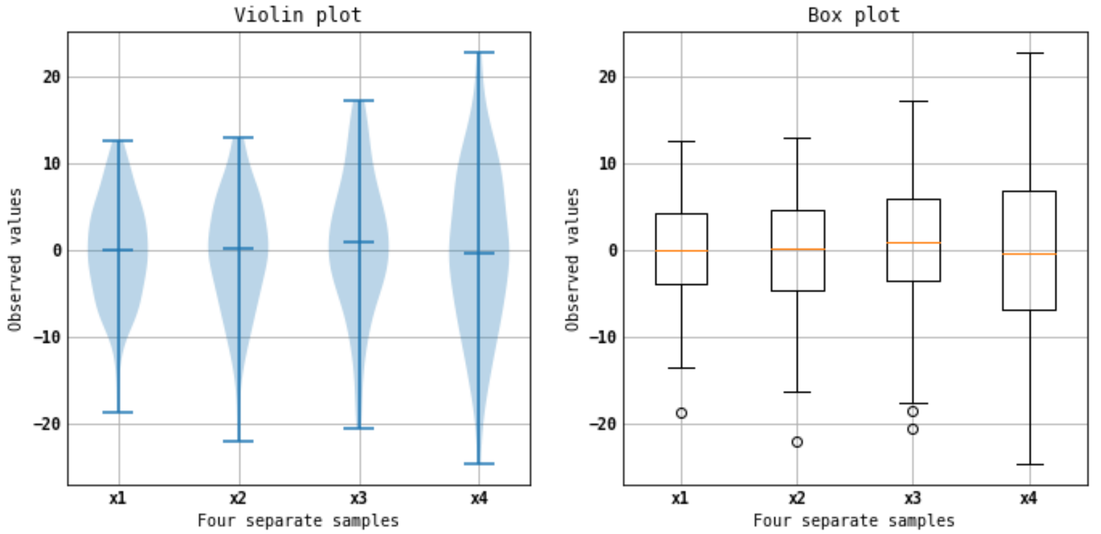
Scatter plot

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

plt.style.use('_mpl-gallery')

# make the data
np.random.seed(3)
x = 4 + np.random.normal(0, 2, 24)
y = 4 + np.random.normal(0, 2, len(x))
# size and color:
sizes = np.random.uniform(15, 80, len(x))
colors = np.random.uniform(15, 80, len(x))

# plot
fig, ax = plt.subplots(figsize=(6, 6))

ax.scatter(x, y, s=sizes, c=colors, vmin=0, vmax=100)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

ax.set_xlabel('Volatility')
ax.set_ylabel('Return %')
ax.set_title(r'ETF Performance')
plt.show()
```



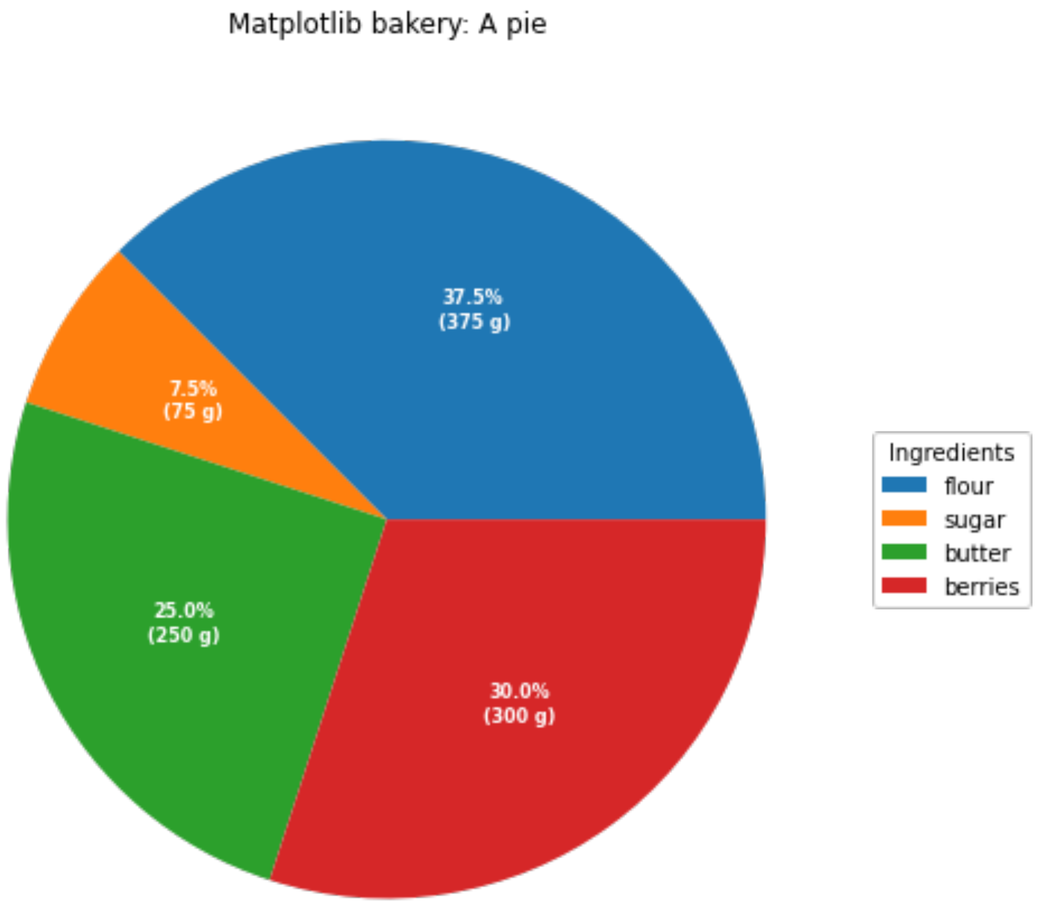
Box plot and Violin plot

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

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(9, 4))

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

# generate some random test data
all_data = [np.random.normal(0, std, 100) for std in range(6, 10)]

# plot violin plot
axs[0].violinplot(all_data,
                  showmeans=False,
                  showmedians=True)
axs[0].set_title('Violin plot')

# plot box plot
axs[1].boxplot(all_data)
axs[1].set_title('Box plot')

# adding horizontal grid lines
for ax in axs:
    ax.yaxis.grid(True)
    ax.set_xticks([y + 1 for y in range(len(all_data))],
                  labels=['x1', 'x2', 'x3', 'x4'])
    ax.set_xlabel('Four separate samples')
    ax.set_ylabel('Observed values')

plt.show()
```



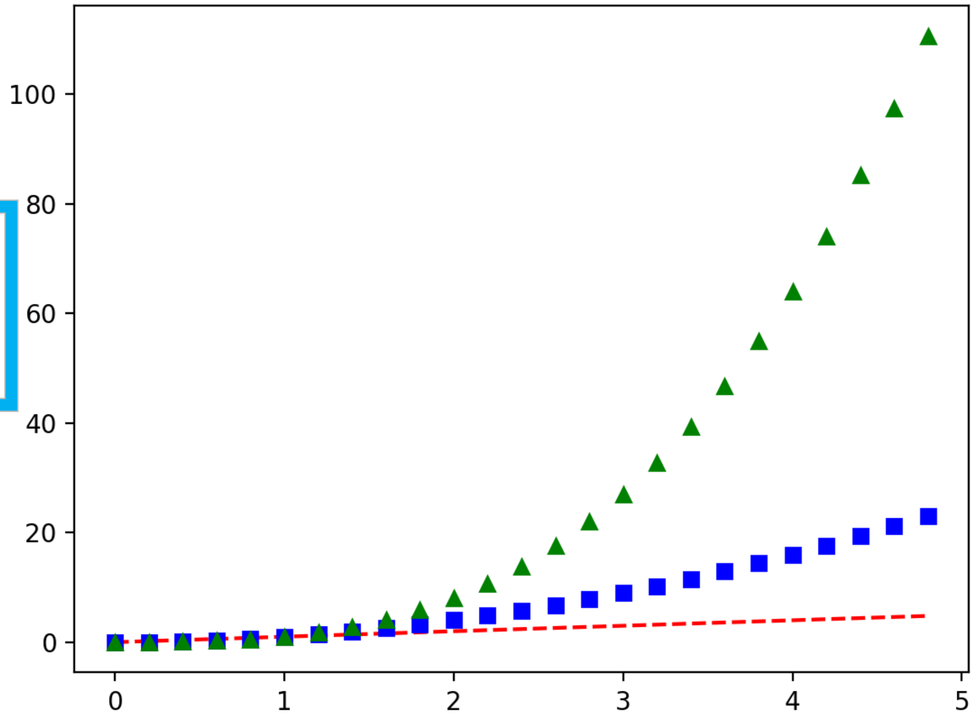
Pie Chart

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

fig, ax = plt.subplots(figsize=(8, 6), subplot_kw=dict(aspect="equal"))

recipe = ["375 g flour",
          "75 g sugar",
          "250 g butter",
          "300 g berries"]

data = [float(x.split()[0]) for x in recipe]
ingredients = [x.split()[-1] for x in recipe]

def func(pct, allvals):
    absolute = int(np.round(pct/100.*np.sum(allvals)))
    return "{:.1f}%\n({:d} g)".format(pct, absolute)

wedges, texts, autotexts = ax.pie(data, autopct=lambda pct: func(pct, data),
                                  textprops=dict(color="w"))

ax.legend(wedges, ingredients,
          title="Ingredients",
          loc="center left",
          bbox_to_anchor=(1, 0, 0.5, 1))

plt.setp(autotexts, size=8, weight="bold")

ax.set_title("Matplotlib bakery: A pie")

plt.show()
```



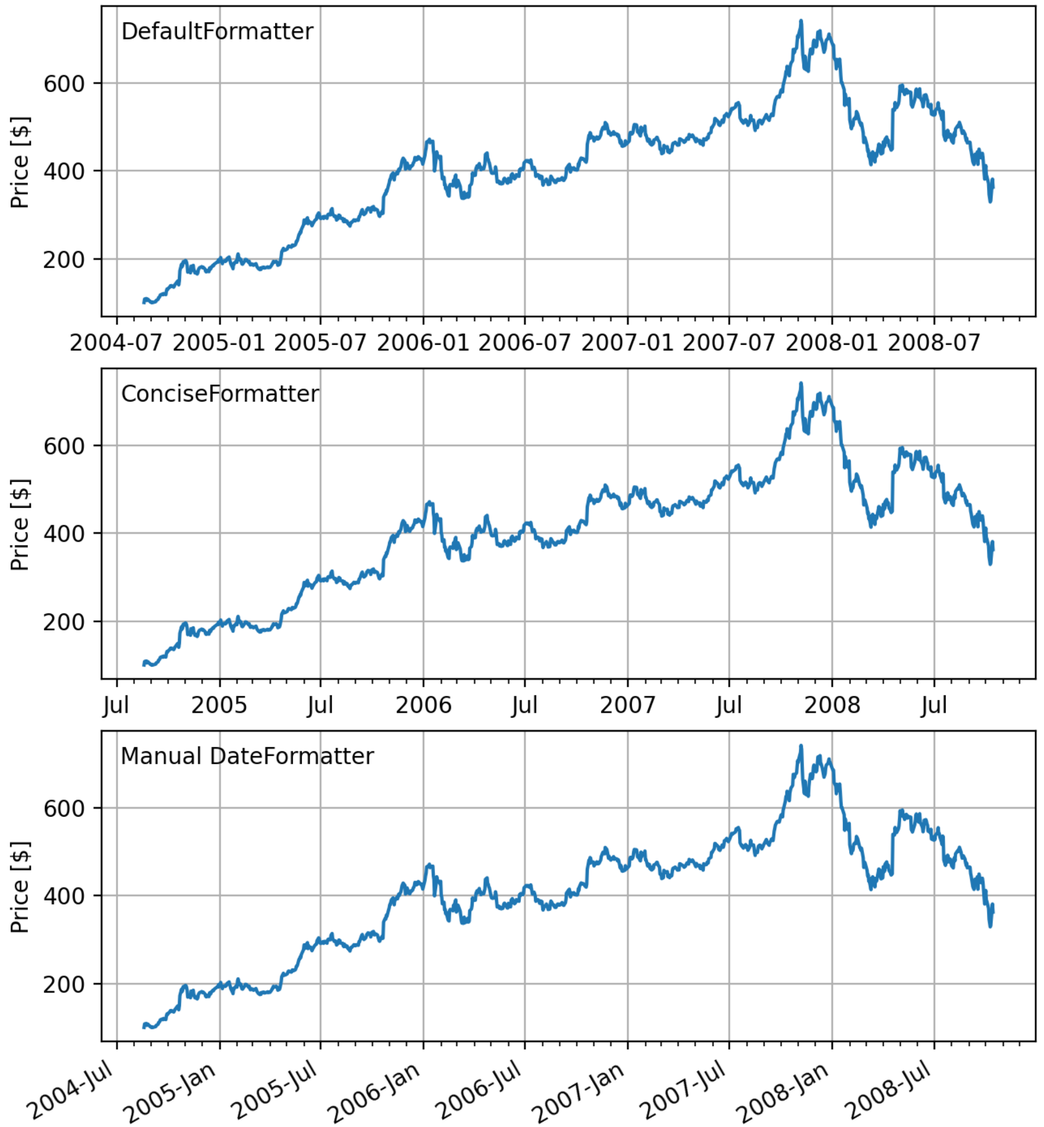
Line formatting

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

figure(figsize=(8, 6), dpi=80)

# 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()
```



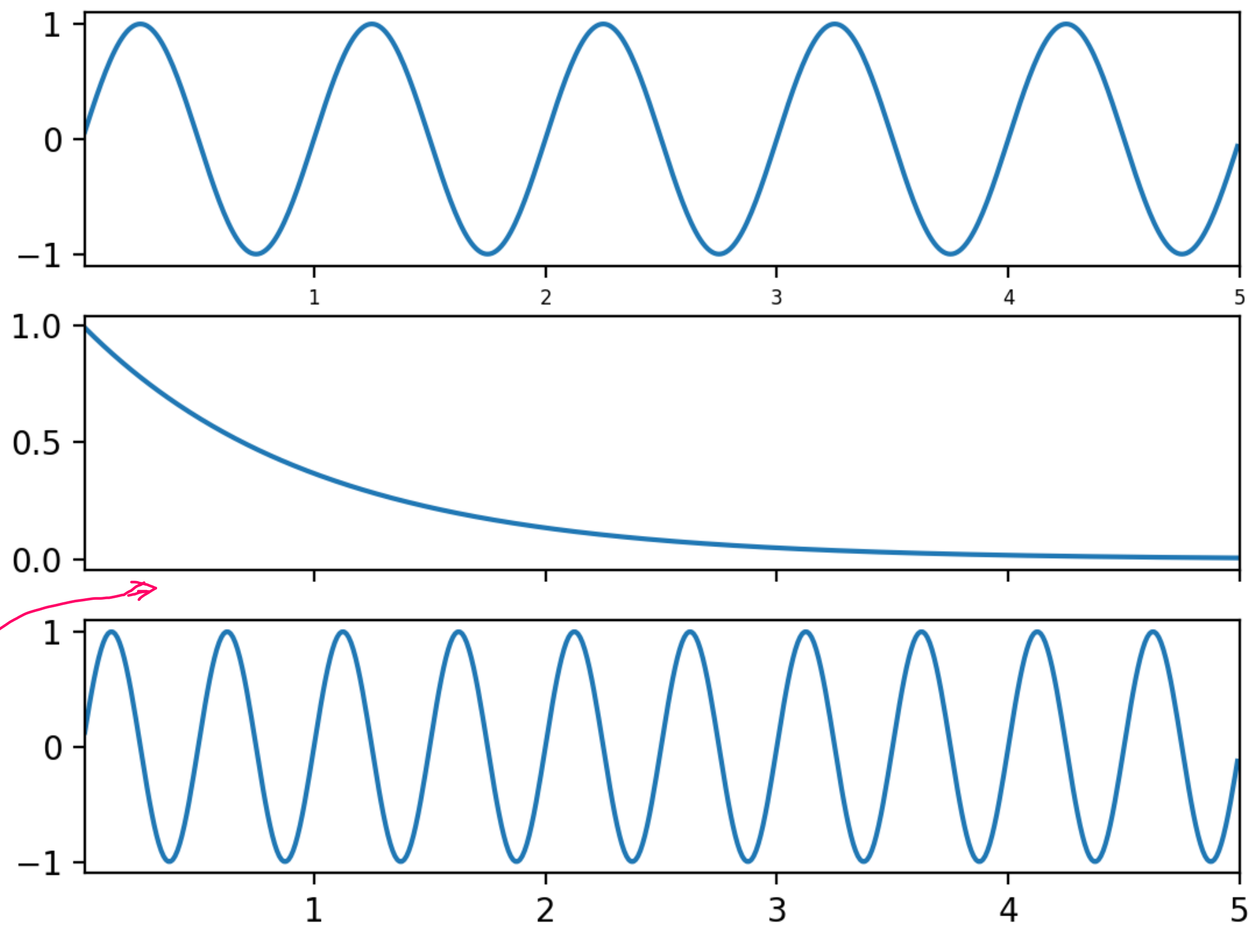
Date Tick label

```python
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.cbook as cbook

# Load a numpy record array from yahoo csv data with fields date, open, high,
# low, close, volume, adj_close from the mpl-data/sample_data directory. The
# record array stores the date as an np.datetime64 with a day unit ('D') in
# the date column.
data = cbook.get_sample_data('goog.npz', np_load=True)['price_data']

fig, axs = plt.subplots(3, 1, figsize=(6.4, 7), constrained_layout=True)
# common to all three:
for ax in axs:
    ax.plot('date', 'adj_close', data=data)
    # Major ticks every half year, minor ticks every month,
    ax.xaxis.set_major_locator(mdates.MonthLocator(bymonth=(1, 7)))
    ax.xaxis.set_minor_locator(mdates.MonthLocator())
    ax.grid(True)
    ax.set_ylabel(r'Price [\$]')

# different formats:
ax = axs[0]
ax.set_title('DefaultFormatter', loc='left', y=0.85, x=0.02, fontsize='medium')

ax = axs[1]
ax.set_title('ConciseFormatter', loc='left', y=0.85, x=0.02, fontsize='medium')
ax.xaxis.set_major_formatter(
    mdates.ConciseDateFormatter(ax.xaxis.get_major_locator()))

ax = axs[2]
ax.set_title('Manual DateFormatter', loc='left', y=0.85, x=0.02,
             fontsize='medium')
# Text in the x axis will be displayed in 'YYYY-mm' format.
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%b'))
# Rotates and right-aligns the x labels so they don't crowd each other.
for label in ax.get_xticklabels(which='major'):
    label.set(rotation=30, horizontalalignment='right')

plt.show()
```



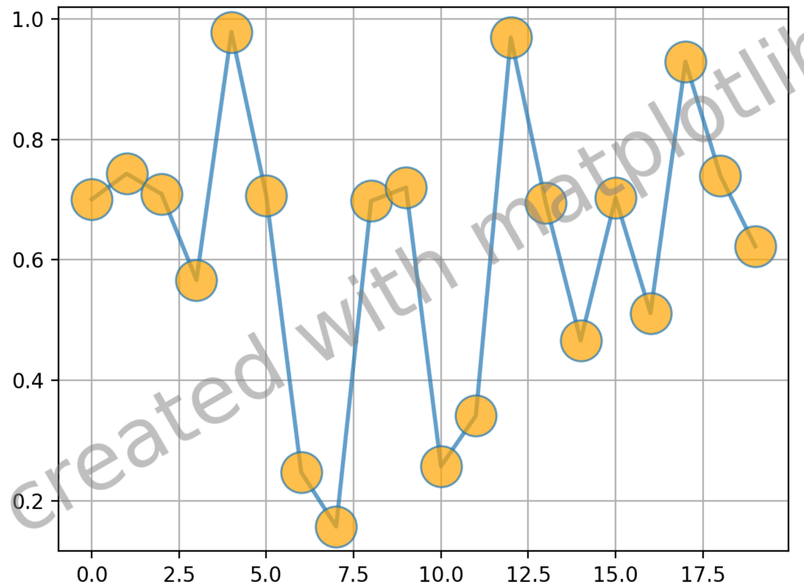
Share label

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

t = np.arange(0.01, 5.0, 0.01)
s1 = np.sin(2 * np.pi * t)
s2 = np.exp(-t)
s3 = np.sin(4 * np.pi * t)

ax1 = plt.subplot(311)
plt.plot(t, s1)
plt.tick_params('x', labelsize=6)

# share x only
ax2 = plt.subplot(312, sharex=ax1)
plt.plot(t, s2)
# make these tick labels invisible
plt.tick_params('x', labelbottom=False)

# share x and y
ax3 = plt.subplot(313, sharex=ax1, sharey=ax1)
plt.plot(t, s3)
plt.xlim(0.01, 5.0)
plt.show()
```



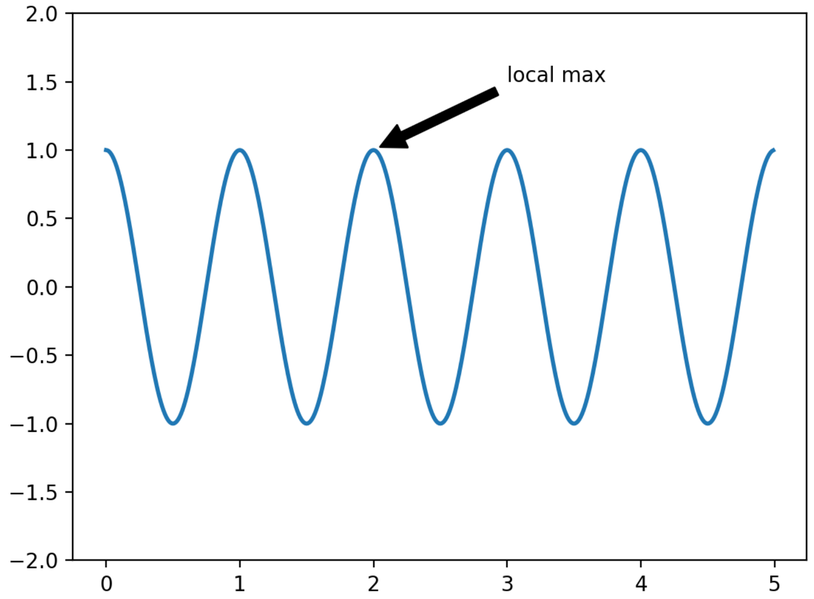
Watermark

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

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


fig, ax = plt.subplots()
ax.plot(np.random.rand(20), '-o', ms=20, lw=2, alpha=0.7, mfc='orange')
ax.grid()

ax.text(0.5, 0.5, 'created with matplotlib', transform=ax.transAxes,
        fontsize=40, color='gray', alpha=0.5,
        ha='center', va='center', rotation=30)

plt.show()
```

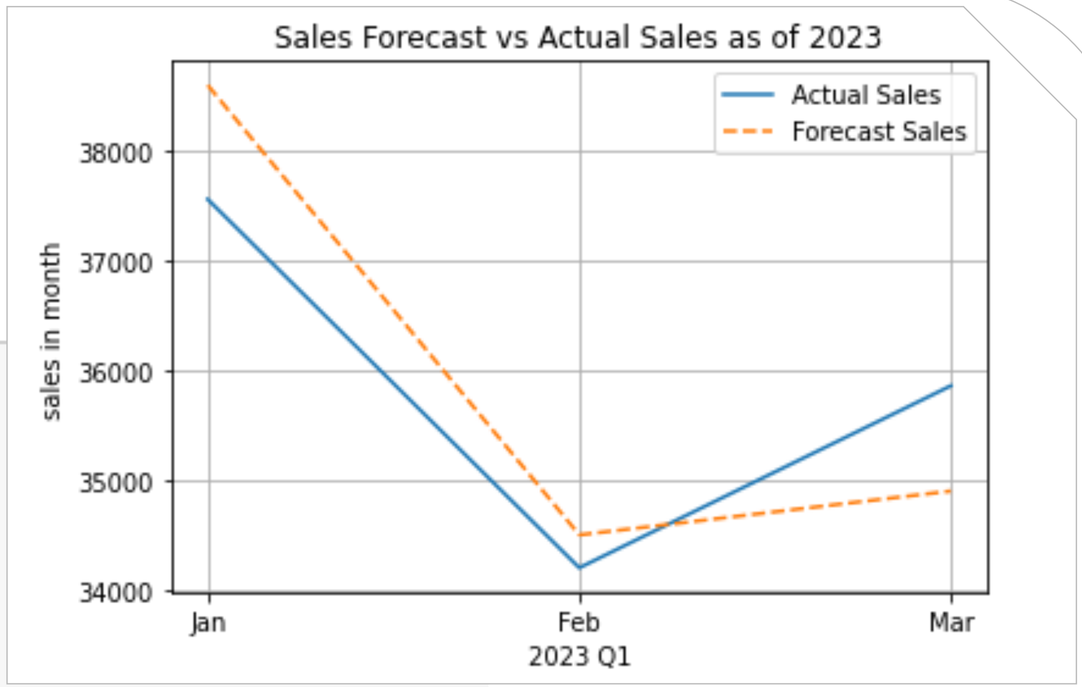


Annotate a plot

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

fig, ax = plt.subplots()

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = ax.plot(t, s, lw=2)

ax.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
            arrowprops=dict(facecolor='black', shrink=0.05),
            )
ax.set_ylim(-2, 2)
plt.show()
```



Typical line plot




```python
import matplotlib.pyplot as plt

month = ['Jan','Feb','Mar']
actual_sales = [37560, 34200, 35860 ]
forecast_sales = [38600, 34500, 34900]

plt.plot(month, actual_sales, label='Actual Sales')
plt.plot(month, forecast_sales, '--', label='Forecast Sales')
plt.ylabel('sales in month')
plt.xlabel('2023 Q1')
plt.grid(True)
plt.legend()
plt.title('Sales Forecast vs Actual Sales as of 2023')
```

Conclusion

- Matplotlib is a large visualization library, and cope with many variety of science fields
- It is evolving and extending function by professionals
- Find out more plots, information and documentation on its website matplotlib.org

Reference

Appendix I

Types of Plot

No	Function & Description
1	**Bar** Make a bar plot.
2	**Barh** Make a horizontal bar plot.
3	**Boxplot** Make a box and whisker plot.
4	**Hist** Plot a histogram.
5	**hist2d** Make a 2D histogram plot.
6	**Pie** Plot a pie chart.
7	**Plot** Plot lines and/or markers to the Axes.
8	**Polar** Make a polar plot..
9	**Scatter** Make a scatter plot of x vs y.
10	**Stackplot** Draws a stacked area plot.
11	**Stem** Create a stem plot.
12	**Step** Make a step plot.
13	**Quiver** Plot a 2-D field of arrows.

Axis Functions

No	Function & Description
1	**Axes** Add axes to the figure.
2	**Text** Add text to the axes.
3	**Title** Set a title of the current axes.
4	**Xlabel** Set the x axis label of the current axis.
5	**Xlim** Get or set the x limits of the current axes.
6	**Xscale** .
7	**Xticks** Get or set the x-limits of the current tick locations and labels.
8	**Ylabel** Set the y axis label of the current axis.
9	**Ylim** Get or set the y-limits of the current axes.
10	**Yscale** Set the scaling of the y-axis.
11	**Yticks** Get or set the y-limits of the current tick locations and labels.

Figures Function

No	Function & Description
1	**Figtext** Add text to figure.
2	**Figure** Creates a new figure.
3	**Show** Display a figure.
4	**Savefig** Save the current figure.
5	**Close** Close a figure window.

Reference

Appendix II

Color Code

Character	Color
‘b’	Blue
‘g’	Green
‘r’	Red
‘b’	Blue
‘c’	Cyan
‘m’	Magenta
‘y’	Yellow
‘k’	Black
‘b’	Blue
‘w’	White

Line Style

Character	Description
‘-‘	Solid line
‘__‘	Dashed line
‘-.’	Dash-dot line
‘:’	Dotted line
‘H’	Hexagon marker

Marker Code

Character	Description
‘.’	Point marker
‘o’	Circle marker
‘x’	X marker
‘D’	Diamond marker
‘H’	Hexagon marker
‘s’	Square marker
‘+’	Plus marker