Matplotlib

Topics to cover…

- Fundamentals of `matplotlib`
- `pyplot` Basics
- Basic Plots

Fundamentals of Matplotlib

matplotlib

Version 3.3.3

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python
- https://matplotlib.org

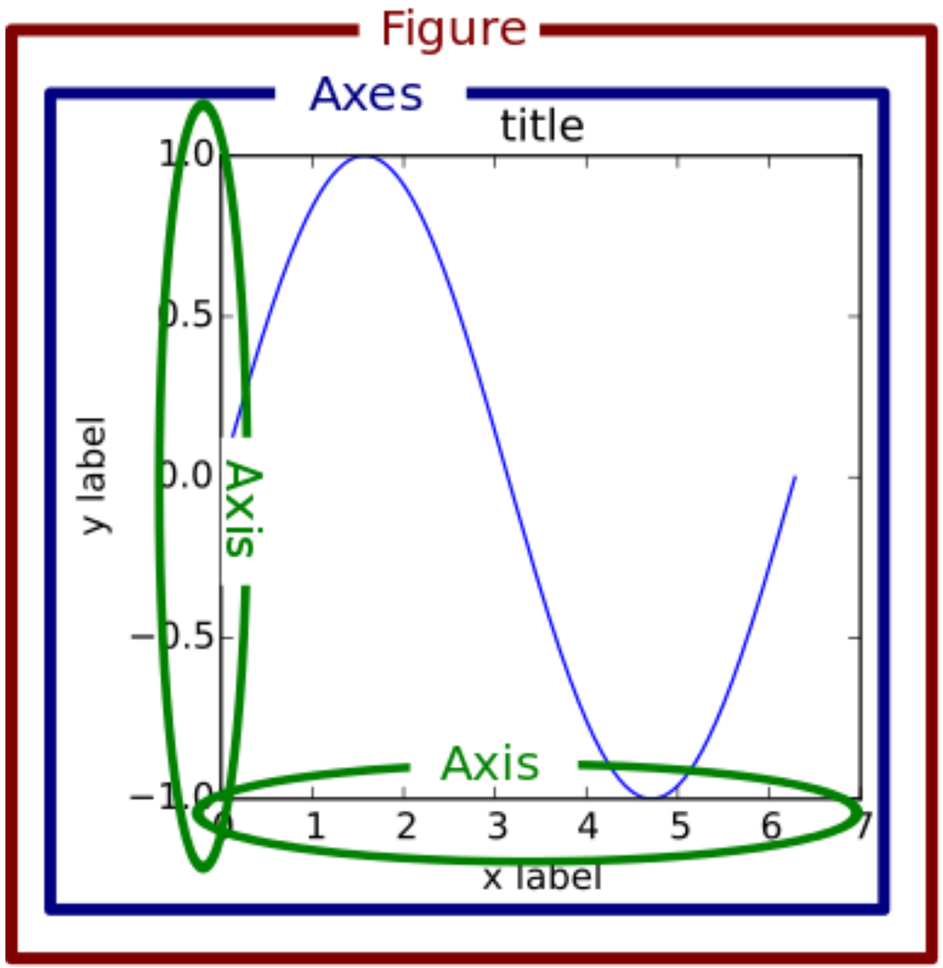
General Concepts

- Everything in Matplotlib is organised as a hierarchical structure
- Each plot is encapsulated in a **`Figure`** object
  - The top-level container of the visualization
  - Can have multiple axes, which are basically individual plots inside this top-level container

Components of Plots

- The two main components of a plot are as follows:
  - **Figure:**
    - An outermost container and is used as a canvas to draw on
    - Can draw multiple plots within it
    - Holds any number of Axes object
    - Can configure the Title
  - **Axes**:
    - An actual plot, or subplot, depending on whether you want to plot single or multiple visualizations
    - Its sub-objects include the x and y axis, spines, and legends

Parts of a `Figure`



Figure

- The figure keeps track of all the child Axes and the **canvas**
- A figure can have any number of Axes, but to be useful should have at least one

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

Axes

- This is what you think of as ‘a plot’
- A given figure can contain many Axes, but a given `Axes` object can only be in one `Figure`
- The Axes contains two (or three in the case of 3D) `Axis` objects (be aware of the difference between **Axes** and **Axis**) which take care of the data limits (the data limits can also be controlled via the `set_xlim()` and `set_ylim()` `Axes` methods)
- Each `Axes` has a title (set via `set_title()`), an x-label (set via `set_xlabel()`), and a y-label set via `set_ylabel()`)

subplots()

```
fig = plt.figure()
axs = fig.subplots(nrows=2, ncols=2)

axs.flat[0].annotate('I am ax1', (0.5, 0.5),
                xycoords='axes fraction', va='center', ha='center')
axs.flat[1].annotate('I am ax2', (0.5, 0.5),
                xycoords='axes fraction', va='center', ha='center')
axs.flat[2].annotate('I am ax3', (0.5, 0.5),
                xycoords='axes fraction', va='center', ha='center')
axs.flat[3].annotate('I am ax4', (0.5, 0.5),
                xycoords='axes fraction', va='center', ha='center')

plt.tight_layout()
```

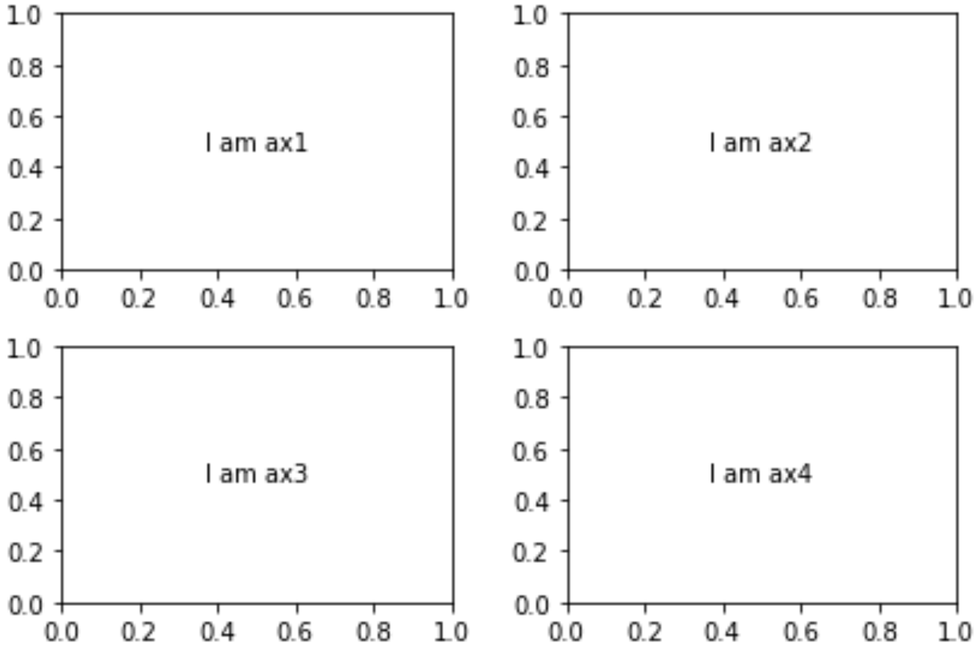


image source: https://towardsdatascience.com/the-many-ways-to-call-axes-in-matplotlib-2667a7b06e06

add_subplot()

```
fig = plt.figure()
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

ax1.annotate('I am ax1', (0.5, 0.5),
             xycoords='axes fraction', va='center', ha='center')
ax2.annotate('I am ax2', (0.5, 0.5),
             xycoords='axes fraction', va='center', ha='center')
ax3.annotate('I am ax3', (0.5, 0.5),
             xycoords='axes fraction', va='center', ha='center')
ax4.annotate('I am ax4', (0.5, 0.5),
             xycoords='axes fraction', va='center', ha='center')
plt.tight_layout()
```

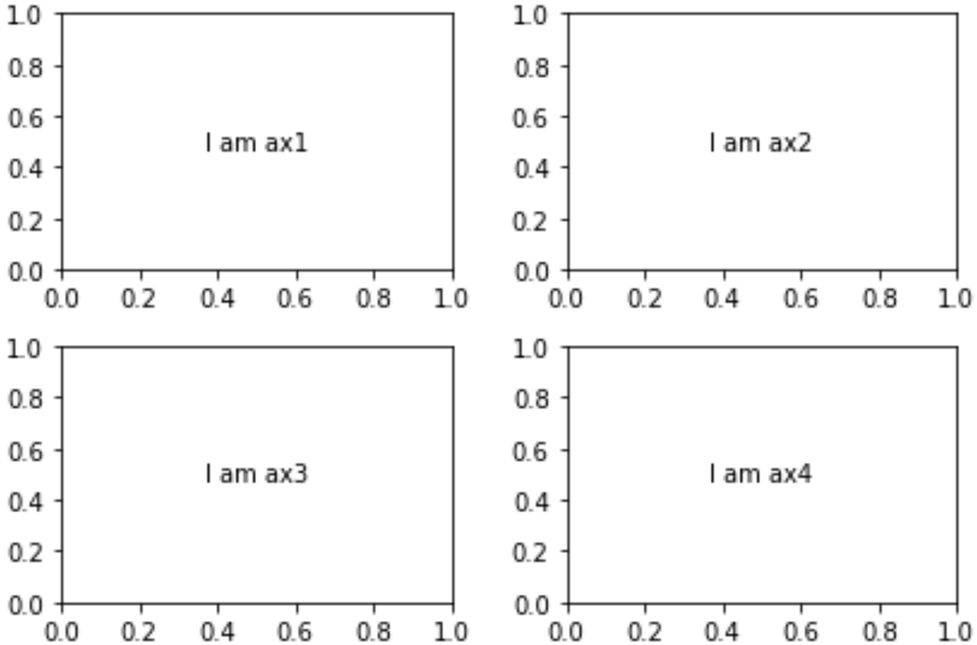


image source: https://towardsdatascience.com/the-many-ways-to-call-axes-in-matplotlib-2667a7b06e06

add_axes()

```
fig = plt.figure()
ax1 = fig.add_axes([0, 0.6, 0.5, 0.5])
ax2 = fig.add_axes([0.6, 0.6, 0.5, 0.5])
ax3 = fig.add_axes([0, 0, 0.5, 0.5])
ax4 = fig.add_axes([0.6, 0, 0.5, 0.5])

ax1.annotate('I am ax1', (0.5, 0.5),
               xycoords='axes fraction', va='center', ha='center')
ax2.annotate('I am ax2', (0.5, 0.5),
               xycoords='axes fraction', va='center', ha='center')
ax3.annotate('I am ax3', (0.5, 0.5),
               xycoords='axes fraction', va='center', ha='center')
ax4.annotate('I am ax4', (0.5, 0.5),
               xycoords='axes fraction', va='center', ha='center')

plt.show()
```

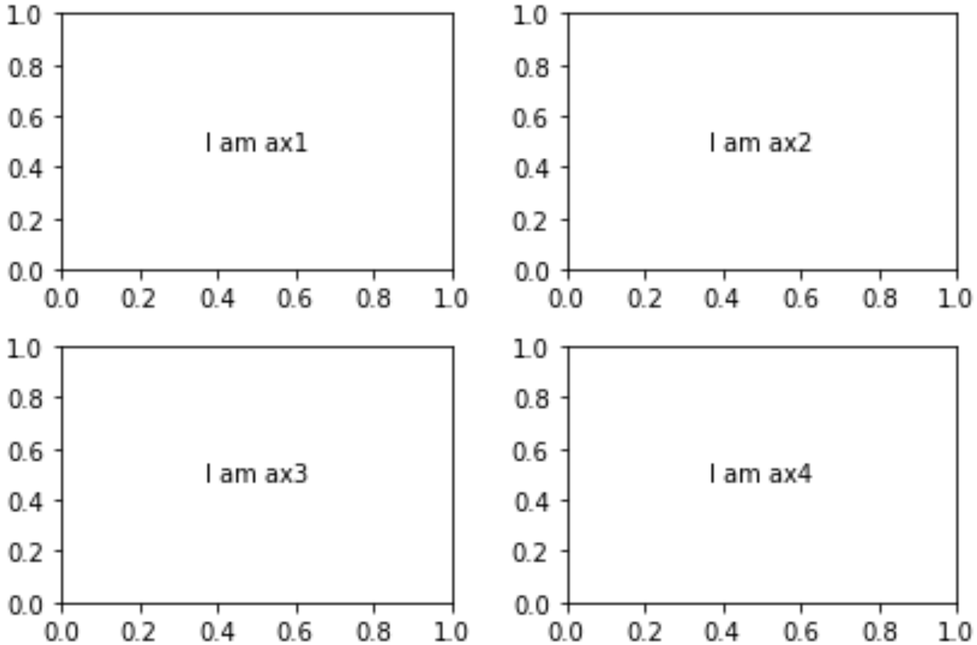


image source: https://towardsdatascience.com/the-many-ways-to-call-axes-in-matplotlib-2667a7b06e06

Axis

- For setting the graph limits and generating the ticks (the marks on the axis) and ticklabels (strings labeling the ticks)
- The location of the ticks is determined by a `Locator` object and the ticklabel strings are formatted by a `Formatter`. The combination of the correct `Locator` and `Formatter` gives very fine control over the tick locations and labels

“Anatomy” of a Figure object

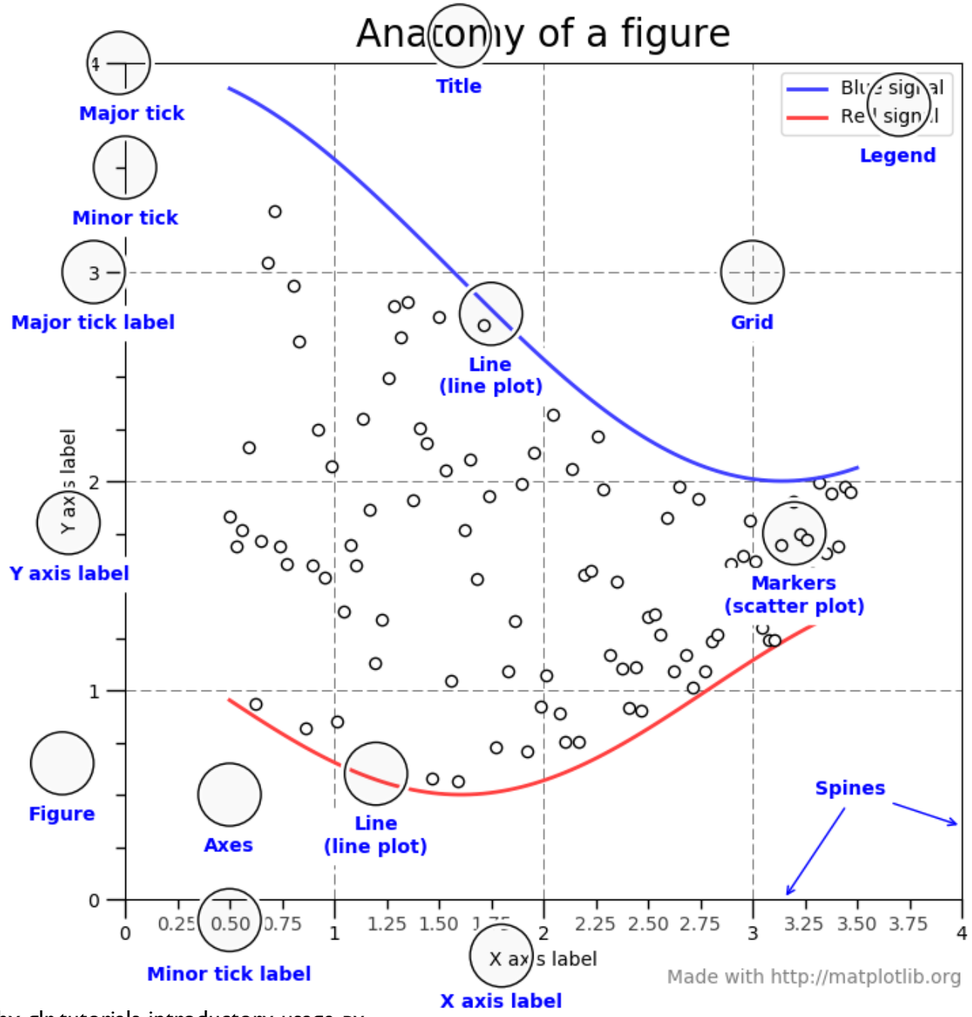


image source: https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py

“Anatomy” of a Figure object

- **Spines**: Lines connecting the axis tick marks
- **Title**: Text label of the whole Figure object
- **Legend**: They describe the content of the plot
- **Grid**: Vertical and horizontal lines used as an extension of the tick marks
- **X/Y axis label**: Text label for the X/Y axis below the spines
- **Minor tick**: Small value indicators between the major tick marks
- **Minor tick label**: Text label that will be displayed at the minor ticks
- **Major tick**: Major value indicators on the spines
- **Major tick label**: Text label that will be displayed at the major ticks
- **Line**: Plotting type that connects data points with a line
- **Markers**: Plotting type that plots every data point with a defined marker

Pyplot basics

Pyplot

- **Pyplot** contains a simpler interface for creating visualizations, which allows the users to plot the data without explicitly configuring the **Figure** and **Axes** themselves
- Matplotlib is the whole package; `matplotlib.pyplot` is a module in matplotlib
- They are implicitly and automatically configured to achieve the desired output
- ```
  import matplotlib.pyplot as plt
  ```
  ```

Format strings

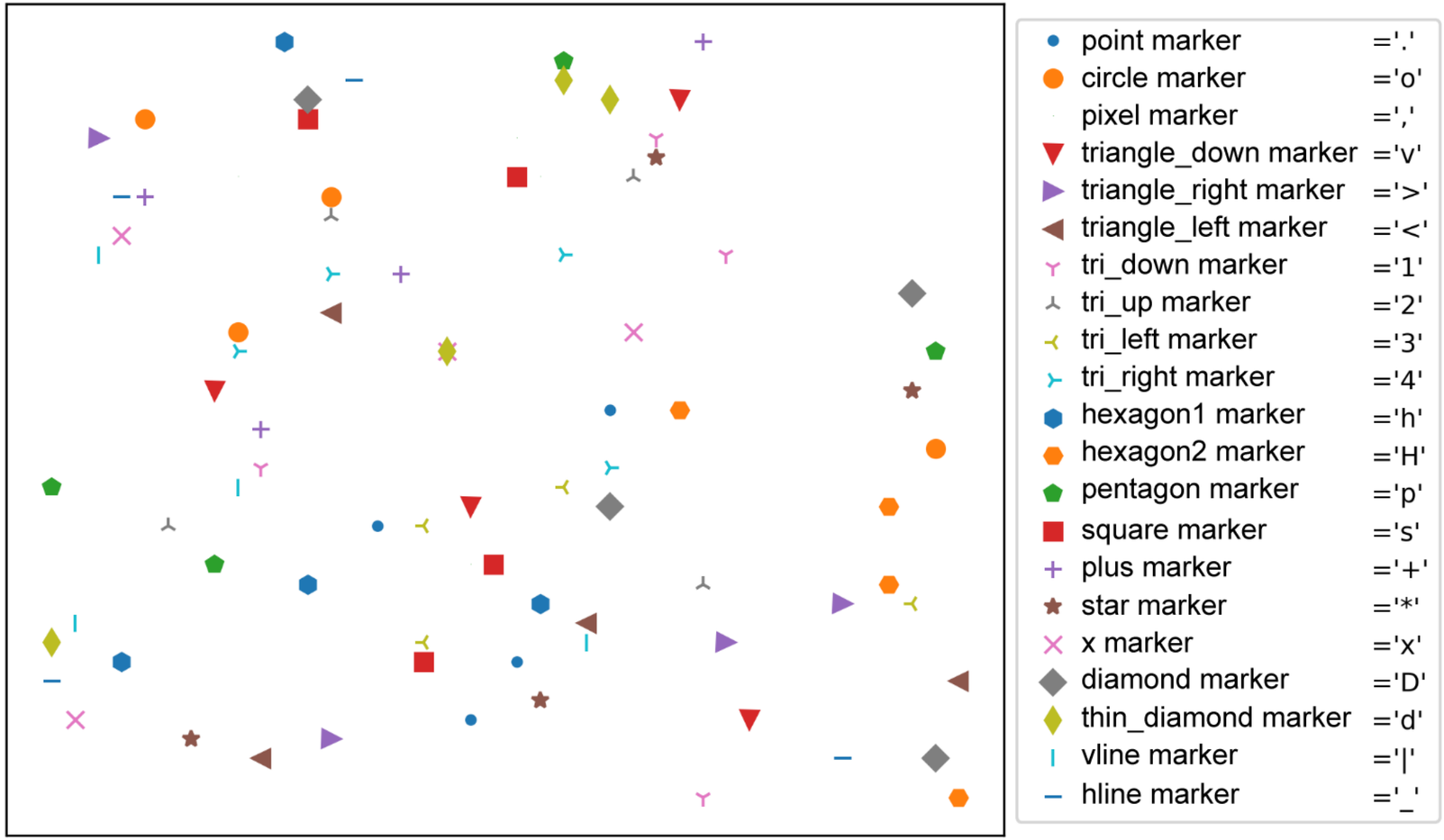
- A format string is specified as `[color][marker][line]`, where each item is optional
- To specify colors, marker types and line styles

Color

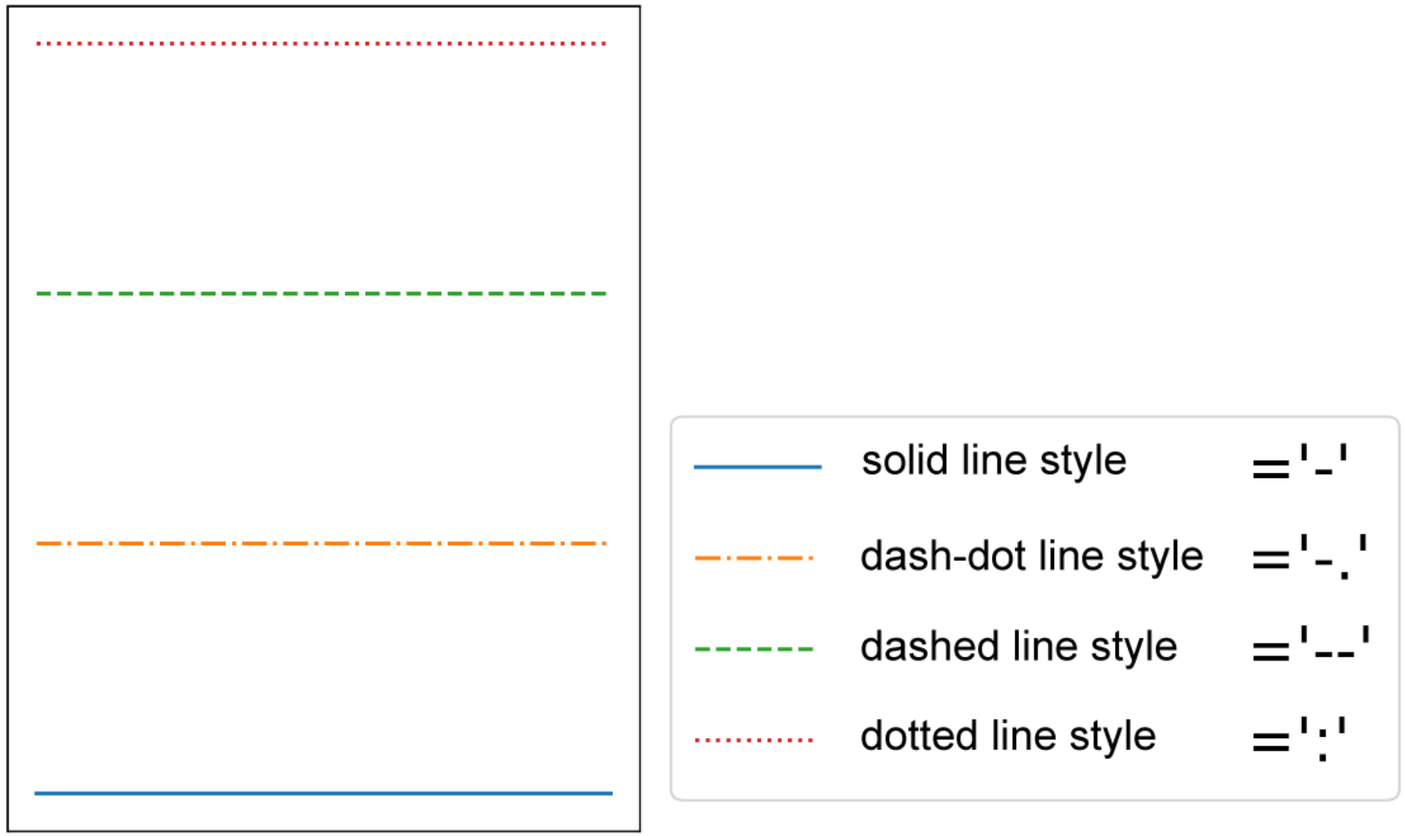
- Matplotlibs recognises the following format:
  - RGB or RGBA float tuples
    - e.g., `(0.2,0.4,0.3)` or `(0.2,0.4,0.3,0.5)`
  - RGB or RGBA hex strings
    - e.g., `'#0F0F0F'` or `'#0F0F0F0F'`

Format	Colors
'b'	blue
'r'	red
'g'	green
'm'	magenta
'c'	cyan
'k'	black
'w'	white
'y'	yellow

Marker


point marker ='.'
circle marker ='o'
pixel marker =','
triangle_down marker ='v'
triangle_right marker ='>'
triangle_left marker ='<'
tri_down marker ='1'
tri_up marker ='2'
tri_left marker ='3'
tri_right marker ='4'
hexagon1 marker ='h'
hexagon2 marker ='H'
pentagon marker ='p'
square marker ='s'
plus marker ='+'
star marker ='*'
x marker ='x'
diamond marker ='D'
thin_diamond marker ='d'
vline marker ='|'
hline marker ='_'

Line

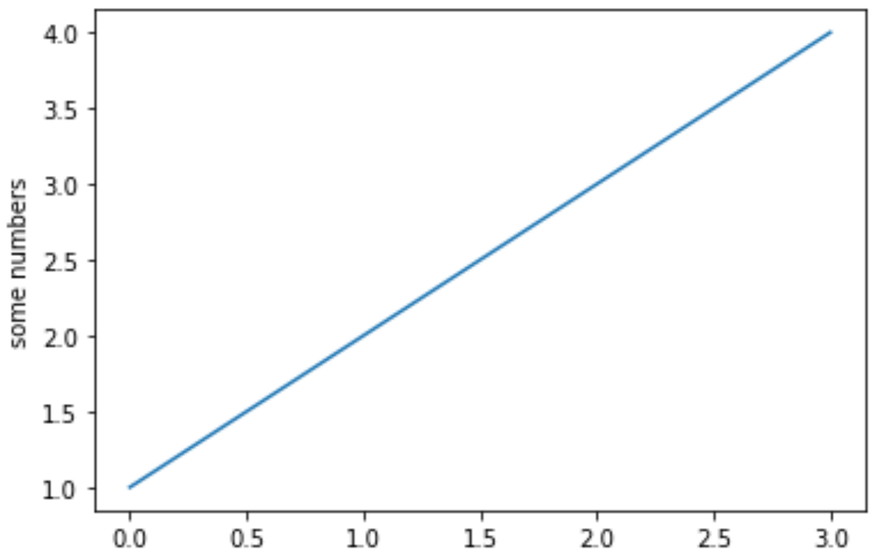

solid line style ='-'
dash-dot line style ='-.'
dashed line style ='--'
dotted line style =':'

Plotting data points

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

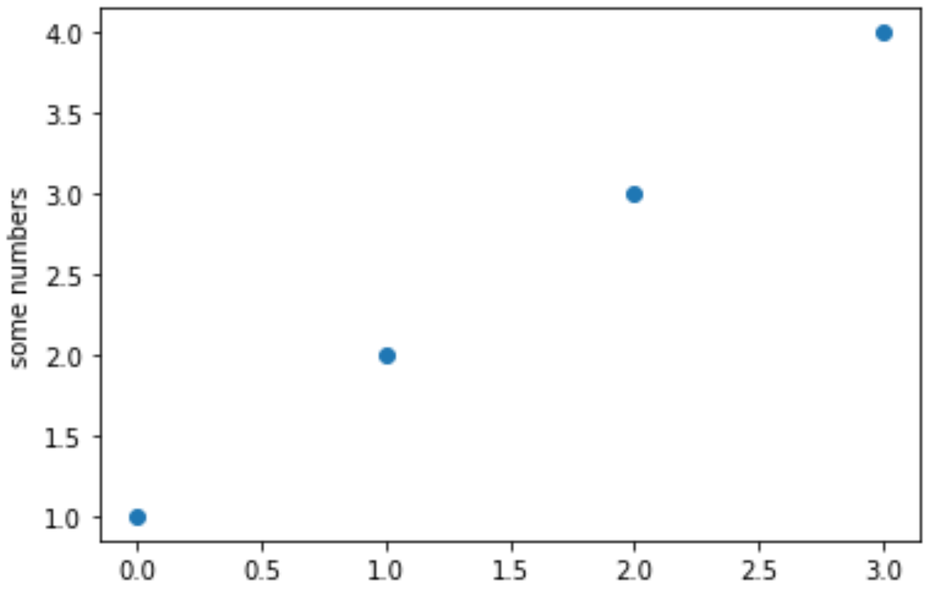
- `plt.plot([x], y, [fmt])`
  - Plot data points as lines and/or markers
- `plt.show()` is used to display a Figure or multiple Figures

```
# Plot data points as lines (default)
plt.plot([1,2,3,4])
plt.ylabel("some numbers")
plt.show()
```

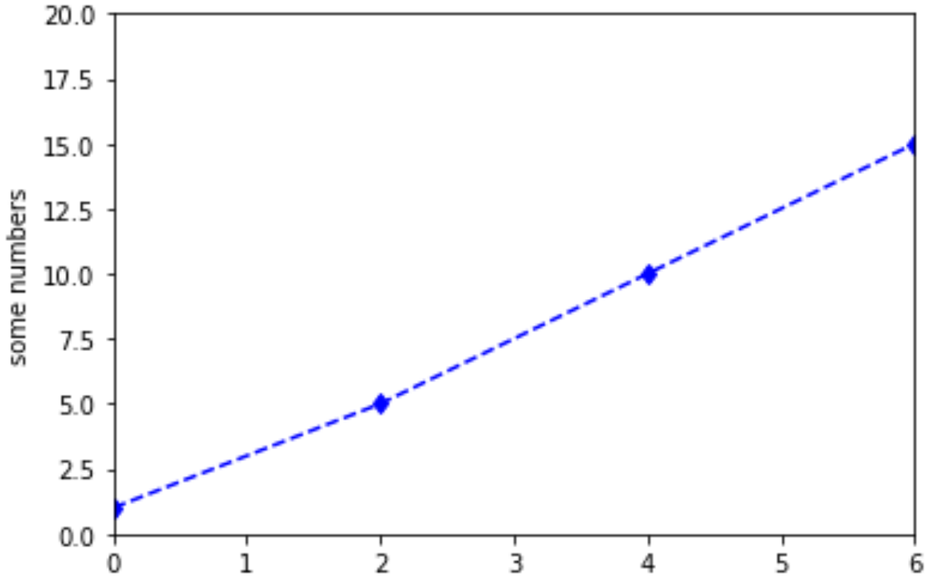


Plotting data points

```
# Plot data points as points
plt.plot([1,2,3,4], 'o')
plt.ylabel("some numbers")
plt.show()
```

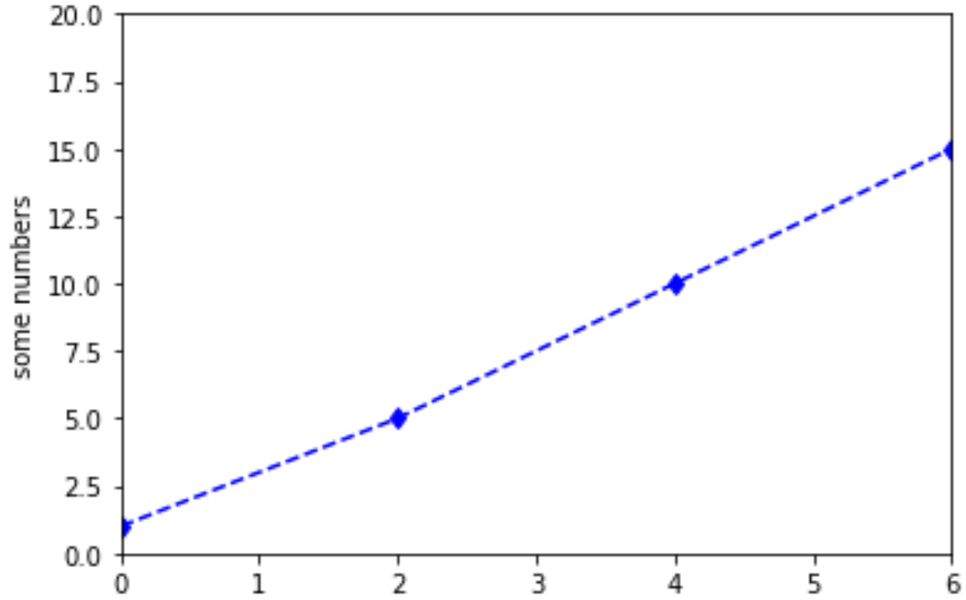



```
# Specifying color, marker and line
# Ploting x vs y by providing two list
plt.plot([0,2,4,6], [1,5,10,15], 'bd--')
plt.ylabel("some numbers")
# Set the range of x- and y-axis
plt.axis([0,6,0,20])
plt.show()
```



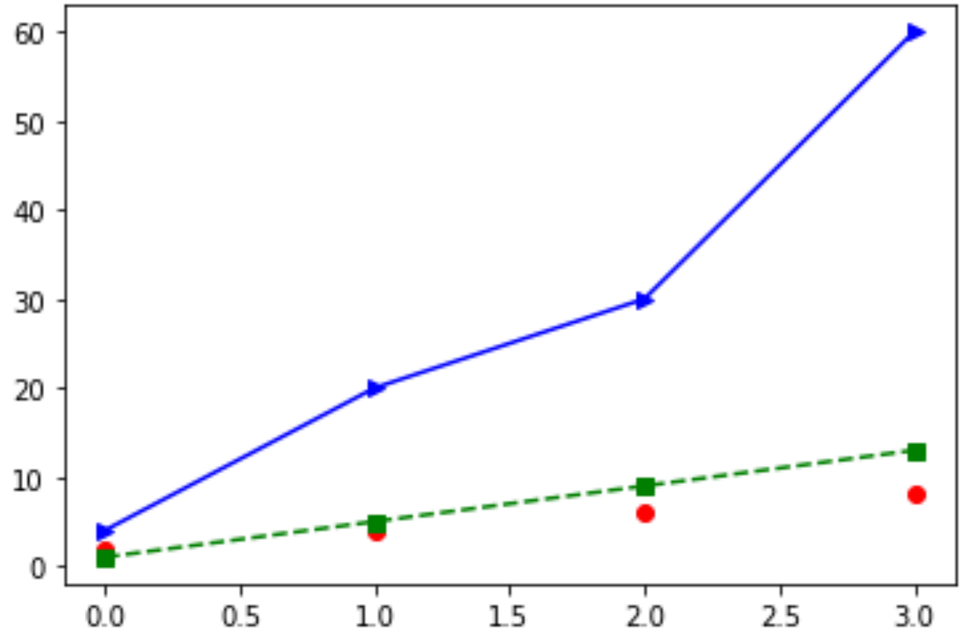
Plotting data points

```
plt.plot([0,2,4,6], [1,5,10,15], 'bd--')
plt.ylabel("some numbers")
# Set the range of x- and y-axis
plt.axis([0,6,0,20])
plt.show()
```



Multiple plots

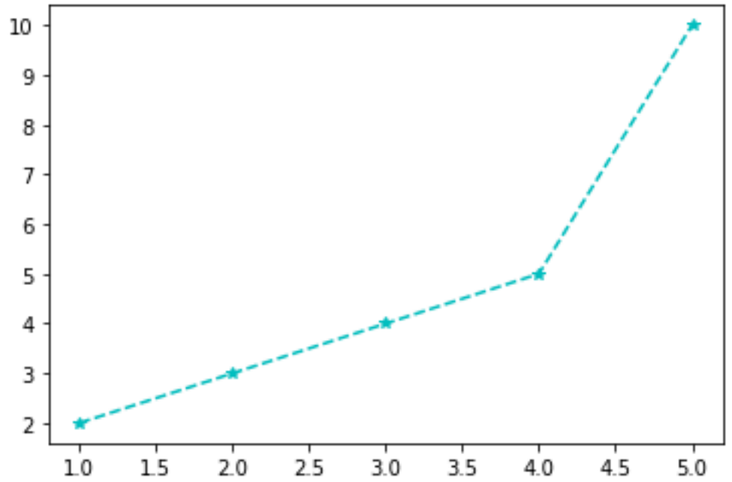
```
# Ploting multiple data pairs
plt.plot([2, 4, 6, 8], 'or', [1, 5, 9, 13], 'sg--', [4, 20, 30, 60], '-b>')
plt.show()
```



Saving figures

- `plt.savefig(fname)` saves the current Figure

```
# Create a Figure object
plt.figure()
# Plot
plt.plot([1,2,3,4,5],[2,3,4,5,10], '*c--')
# Save as a file named 'lineplot.png'
plt.savefig('lineplot.png', dpi=300, bbox_inches='tight') #bbox_inches='tight' removes outer write margins
```



Title, labels, limits

```
# Create a Figure object
fig = plt.figure()

# Add two subplots at a 1x2 grid of the Figure object
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)

# Plot on the Axes 'ax'
ax1.plot([0,2,4,6,8,10], [1,5,10,100,23,3], 'yH--')
ax2.plot([0,2,4,20,8,10], 'gD-')

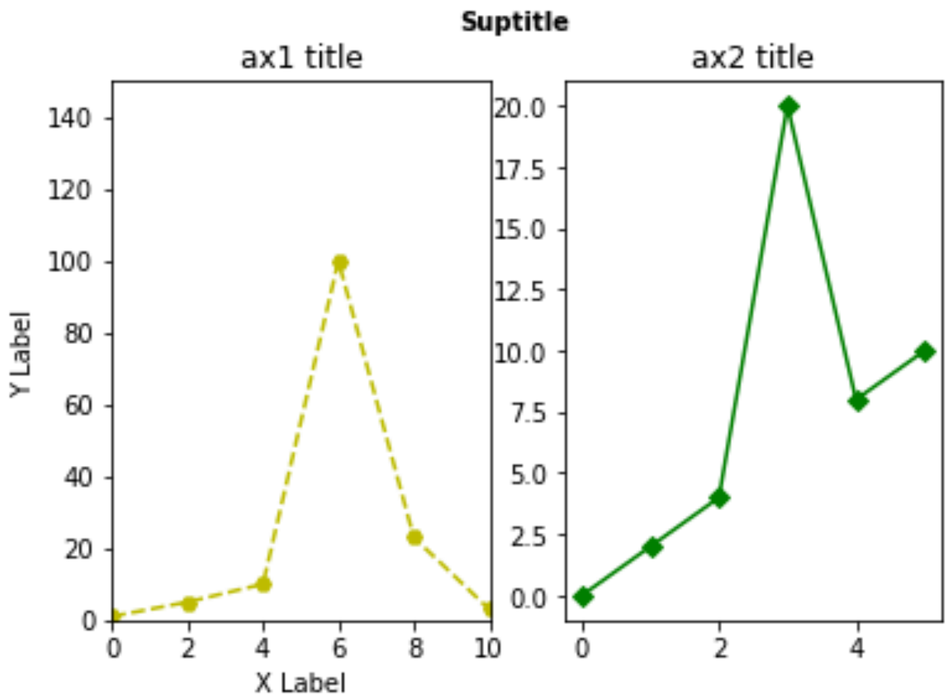
# Set labels
ax1.set_xlabel('X Label')
ax1.set_ylabel('Y Label')

# Set Limits
ax1.set_xlim(0,10)
ax1.set_ylim(0,150)

# Set a figure title
fig.suptitle('Suptitle', fontsize=10, fontweight='bold')

# Set axes title
ax1.set_title('ax1 title')
ax2.set_title('ax2 title')

plt.show()
```



Annotate

```
# Create a Figure object
fig = plt.figure()

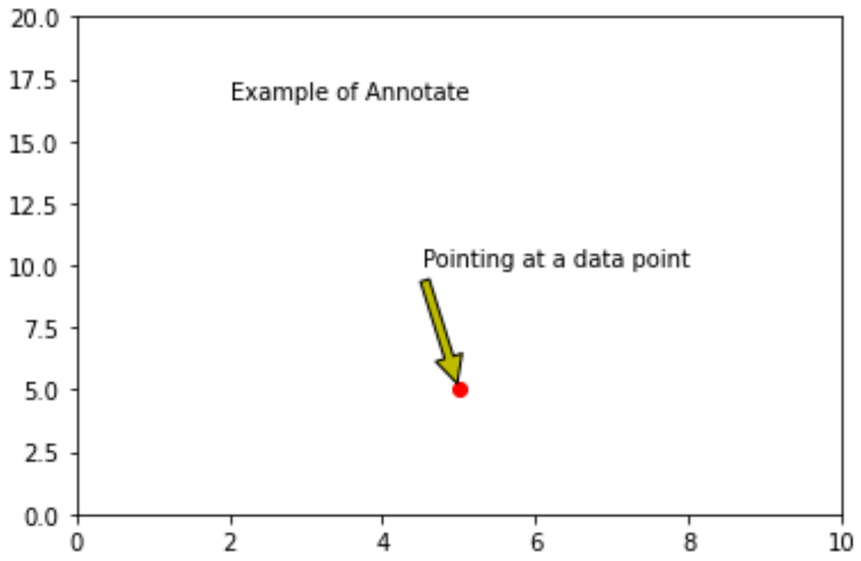
# Add one subplots at a 1x1 grid of the Figure object
ax1 = fig.add_subplot(1,1,1)

ax1.plot([5],[5],'ro')
ax1.set_xlim(0,10)
ax1.set_ylim(0,20)

# Add the text 'Example of Annotate' at the coordinate (2, 17)
# ha (horizontal alignment): center, left, right
# va (vertical alignment): center, top, bottom
ax1.annotate('Example of Annotate', xy=(2,17), ha='left', va='center')

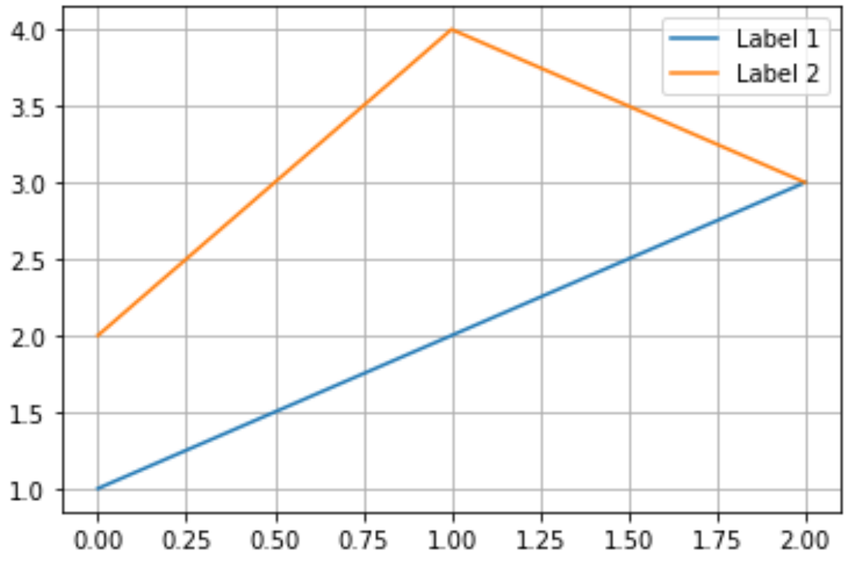
# Add the text 'Pointing at a data point' at the coordinate (8, 10)
# point to a data point at (5,5)
ax1.annotate('Pointing at a data point', xy=(5,5), xytext=(8,10), ha='right',
             arrowprops=dict(facecolor='y', shrink=0.05))

plt.show()
```



Legand, grid

```
plt.plot([1, 2, 3], label='Label 1')
plt.plot([2, 4, 3], label='Label 2')
plt.legend()
plt.grid()
plt.show()
```



References

- Part of this slide set is prepared or/and extracted from the following resources:
  - Mario Dobler and Tim Gromann (2019): “Data Visualization with Python: Create an impact with meaningful data insights using interactive and engaging visuals”, Packt Publishing
  - Matplotlib: https://matplotlib.org/
- This set of slides is for teaching purpose only