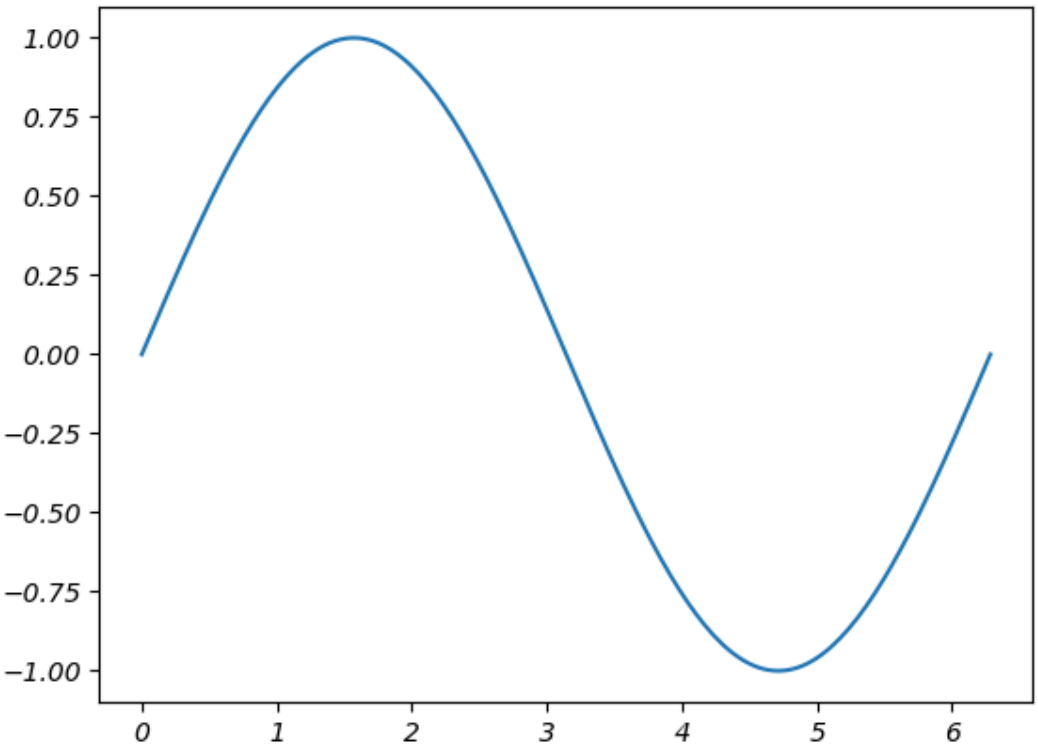
Introduction to matplotlib

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
## A simple example
```

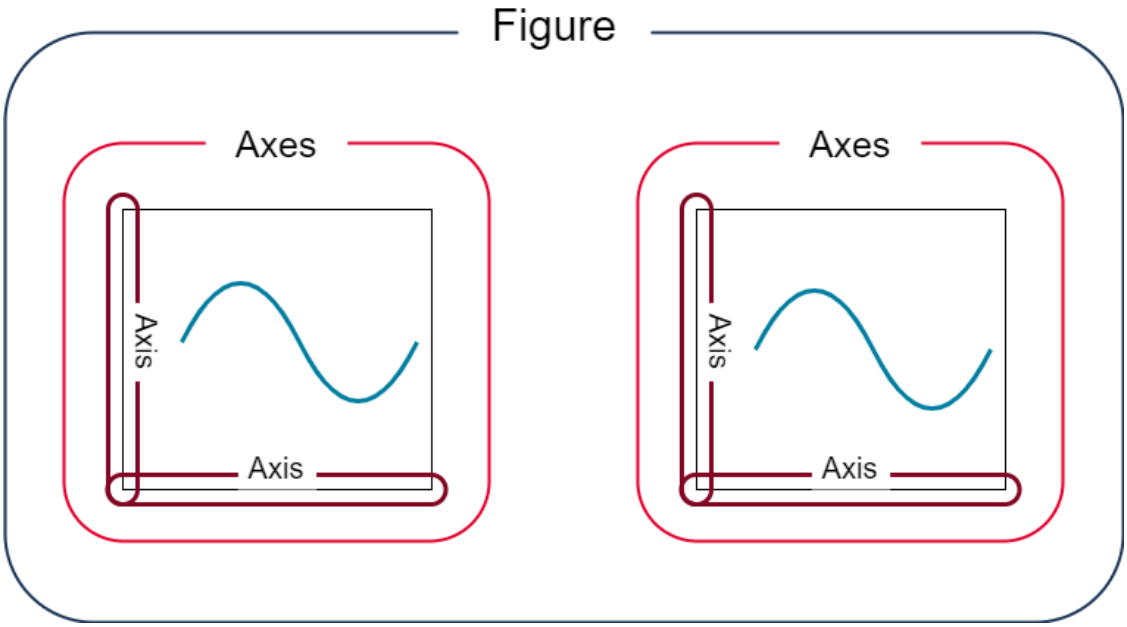
In [3]:
```
# Define data
x = np.linspace(0, 2 * np.pi, 200)
y = np.sin(x)
```

In [4]:
```
# Create Figure and Axes objects
fig, ax = plt.subplots()

# Add a line
ax.plot(x, y);
```



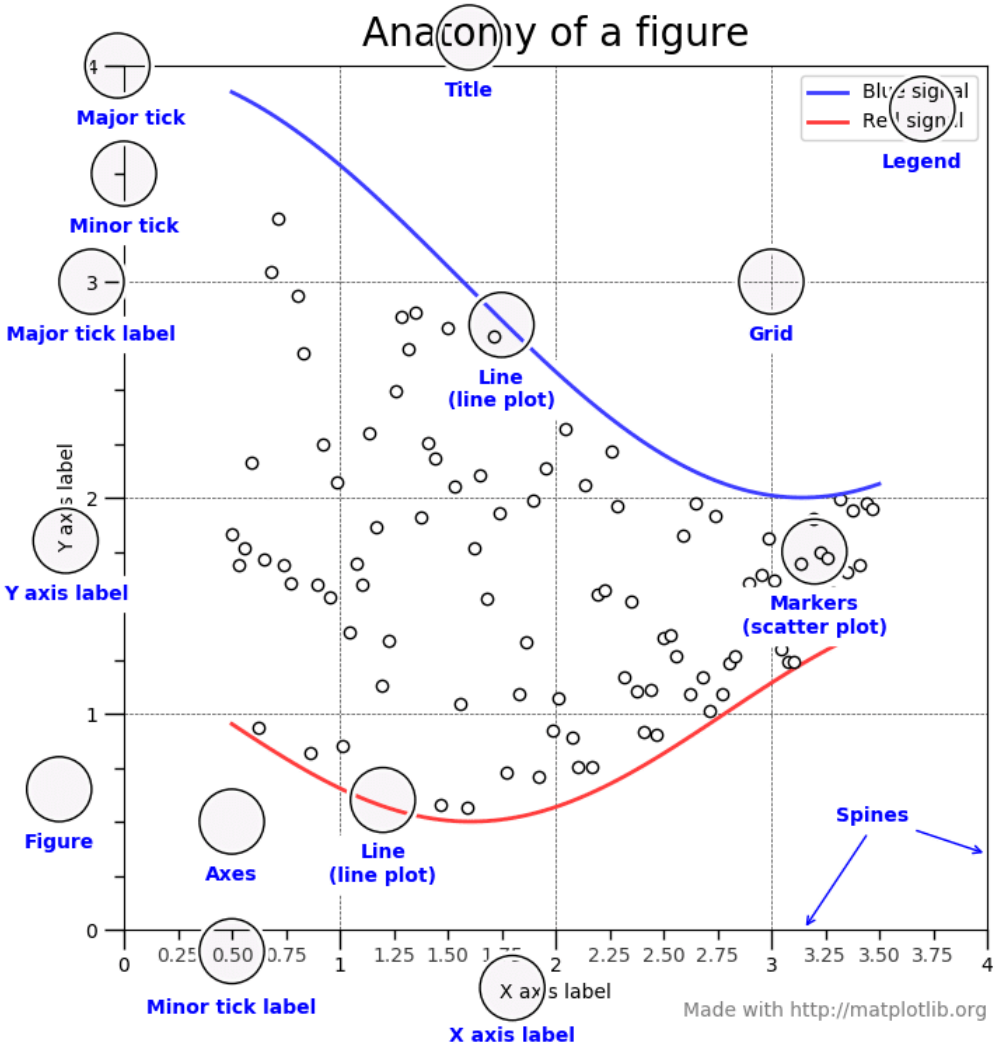
Components of a figure

- The **Figure** object is the top level container of all graphical elements
- The **Axes** object belongs to a Figure object and is the space where we plot data
- The **Axis** object belongs to an Axes object and can be categorized as XAxis or YAxis



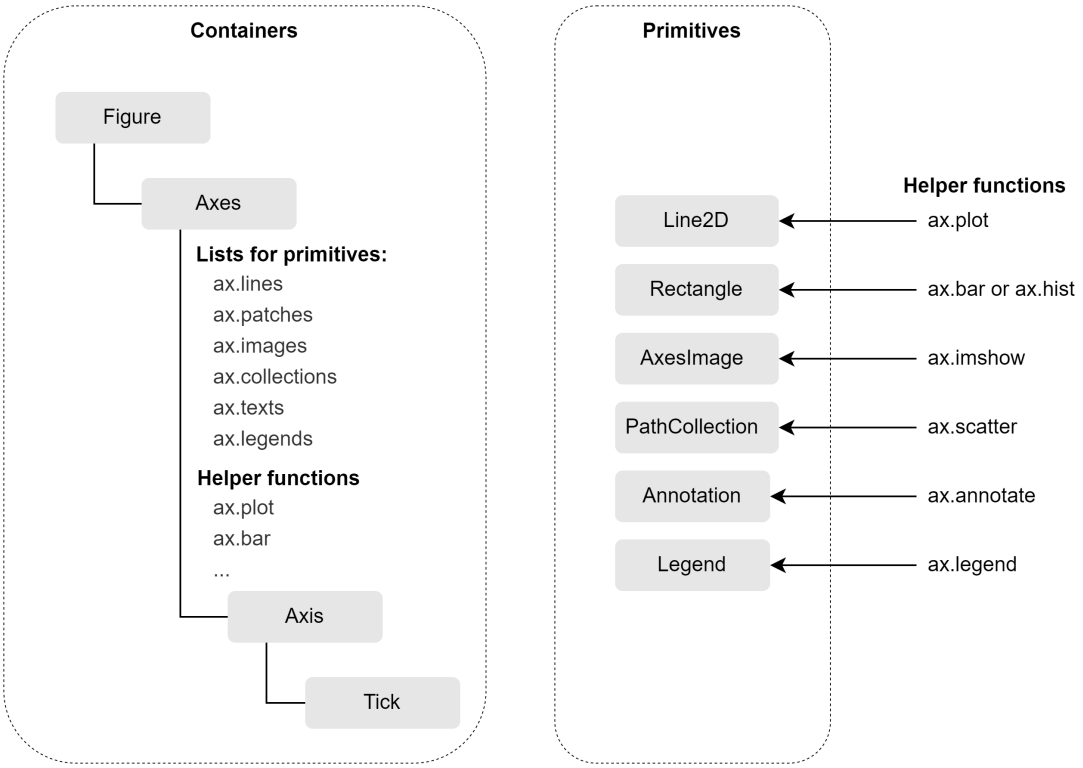
The Artist object

Everything in your plot is an **Artist** object. This includes text, lines, markers and even Figure, Axes, and Axis objects.



Two types of Artist objects

- **Containers**: hold and manage collections of graphical elements
- **Primitives**: represent a graphical element that is drawn directly on an Axes within a figure (e.g., line, markers, text, patches, images)



Coding styles

Explicit style (recommended)

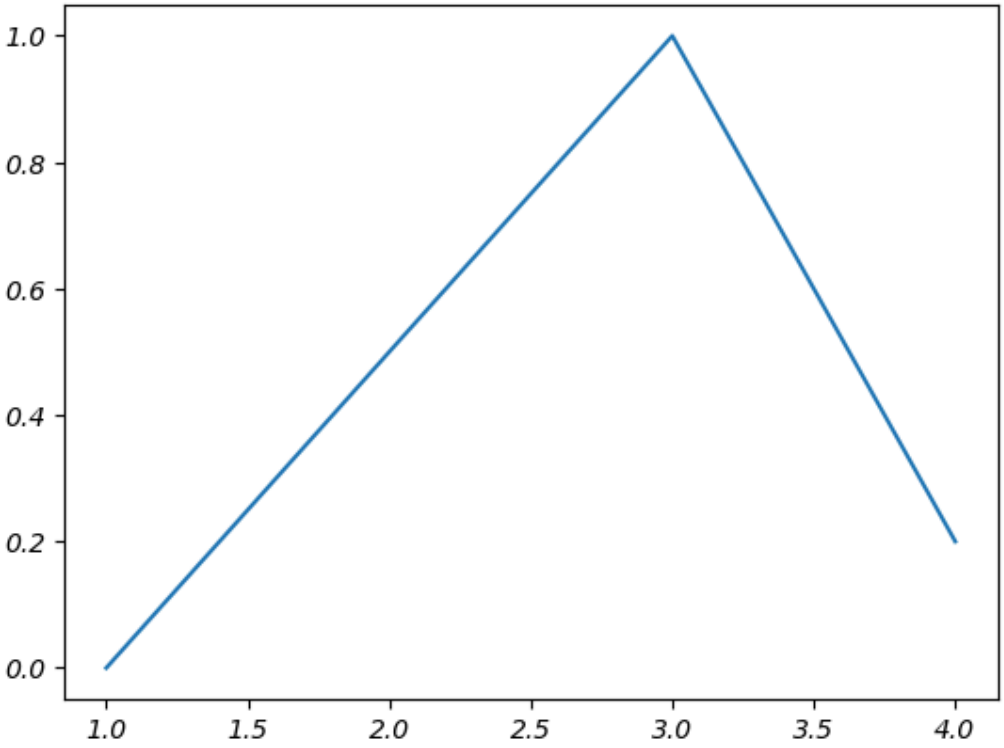
- Explicitly create Figures and Axes, and call helper functions on them to create other Artists
- Build a visual step by step
- This style is ideal for customizing and fine-tuning a visual

Implicit style:

- Use pyplot functions without explicitly creating the Figure and Axes objects
- This style is ideal for doing interactive work or simple visuals
- The current Figure can be retrieved using **plt.gcf()**
- The current Axes can be retrieved using **plt.gca()**

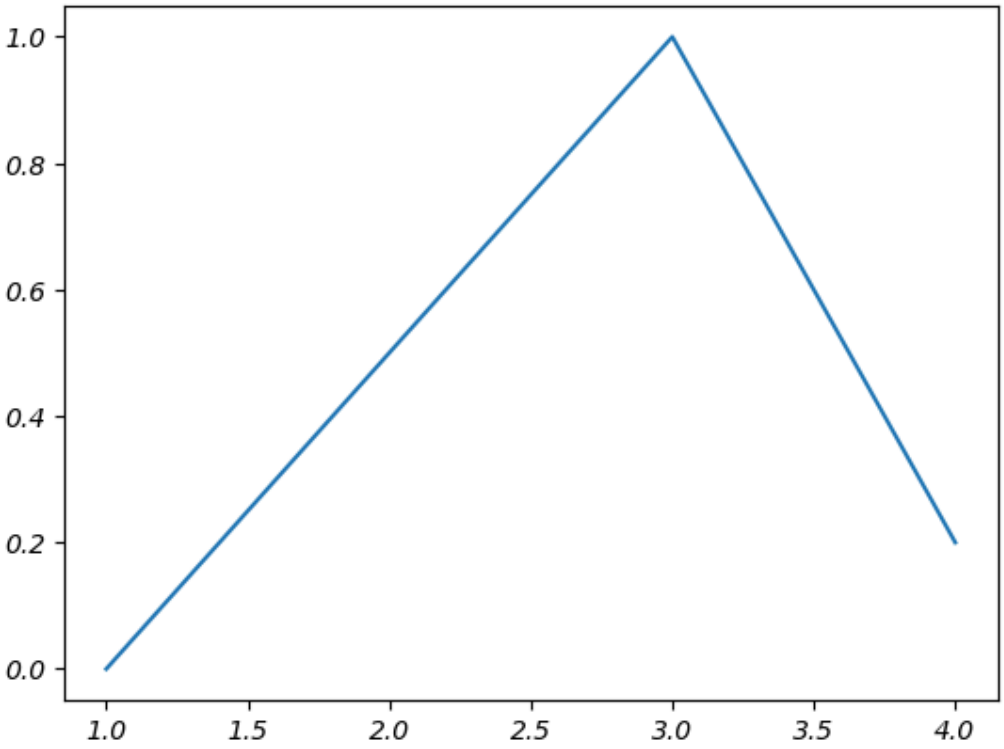
Explicit style (recommended)

In [5]:
```
# Create Figure object
fig = plt.figure()

# Add Axes object
ax = fig.subplots()

# Create line
ax.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2]);
```



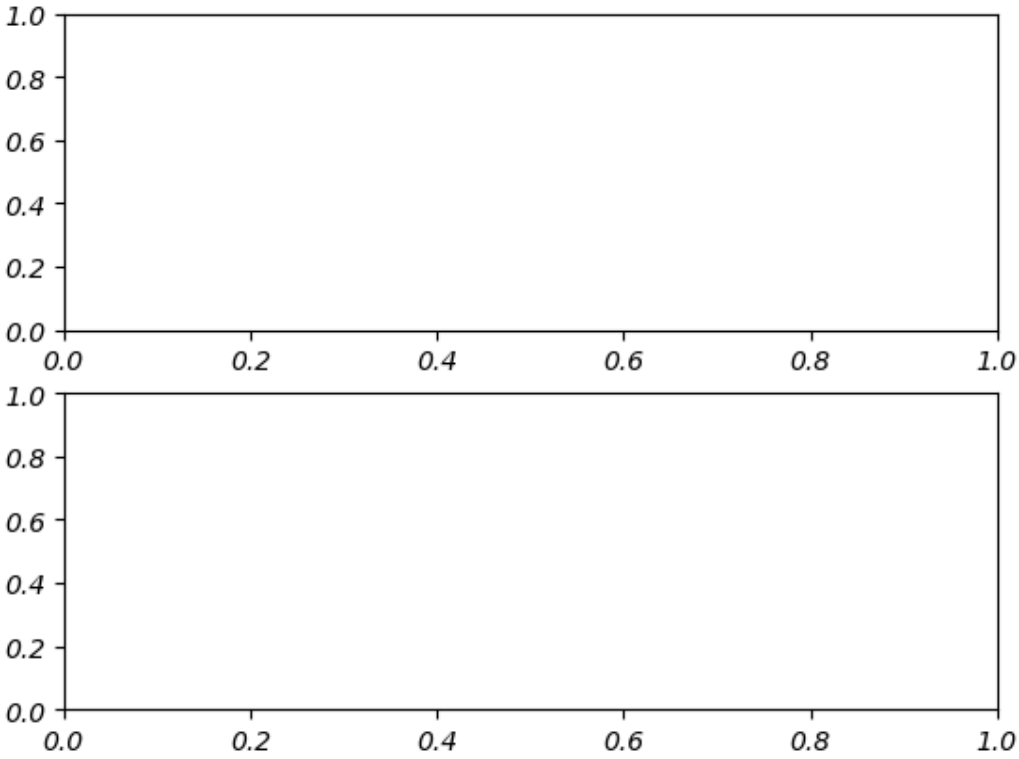
Implicit style

In [6]:
```
# Create Figure and Axes objects implicitly by using the plt.plot function
plt.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2]);
```

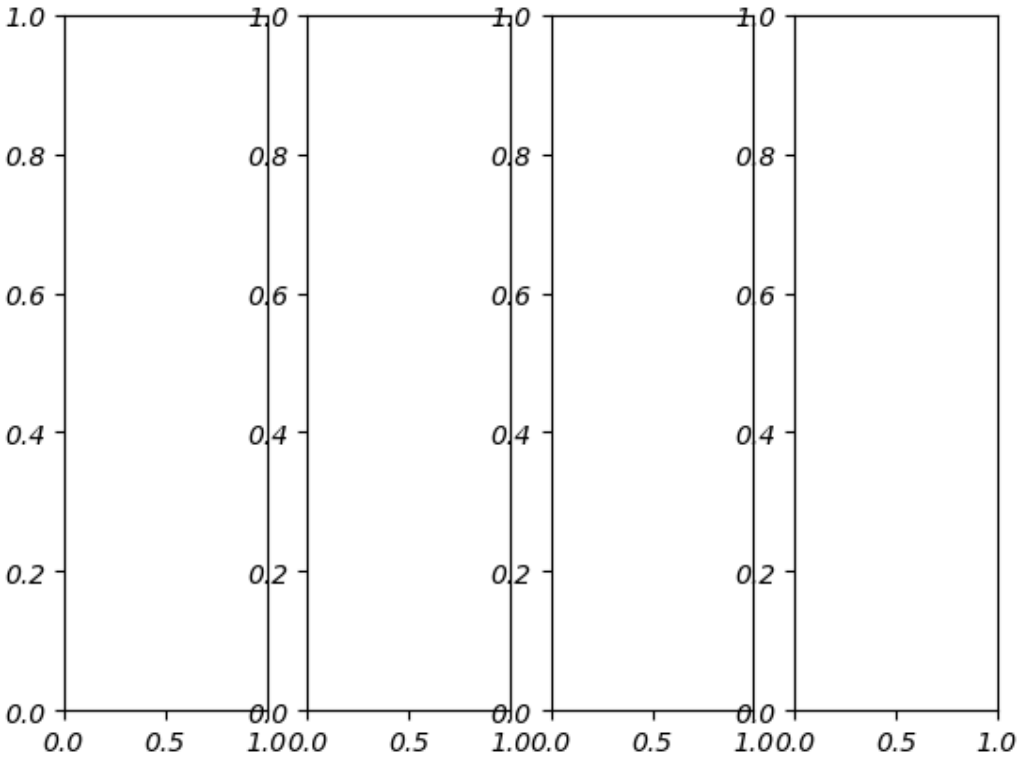


1) How to create multiple Axes

In [7]:
```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=2, ncols=1)
```

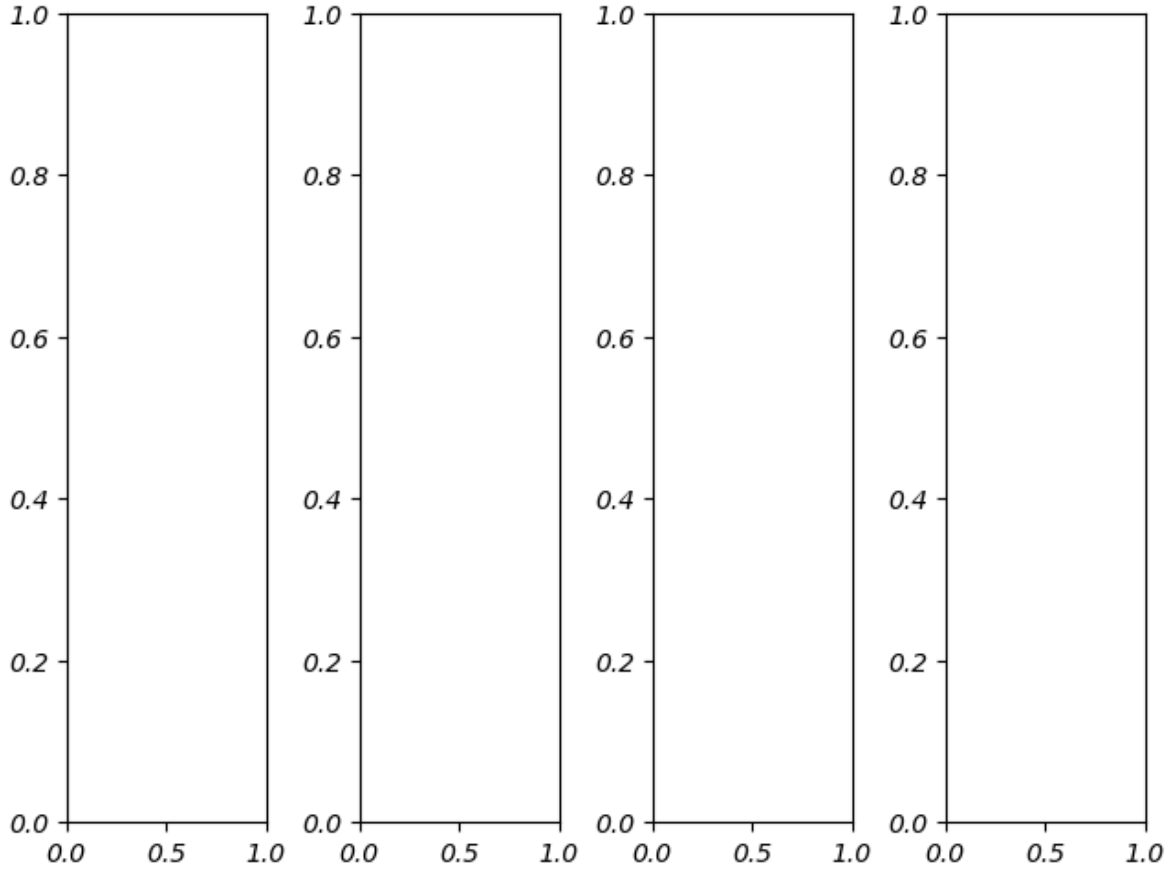


In [8]:
```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=1, ncols=4)
```



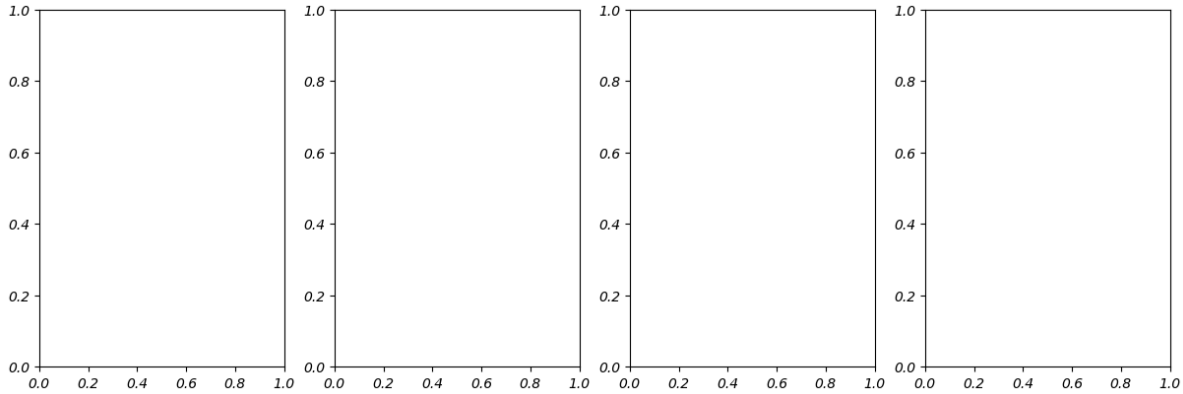
The **constrained** layout automatically adjusts subplots and decorations like legends and colorbars so that they fit in the figure window while still preserving, as best they can, the logical layout requested by the user.

In [9]:
```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=1, ncols=4, layout='constrained')
```



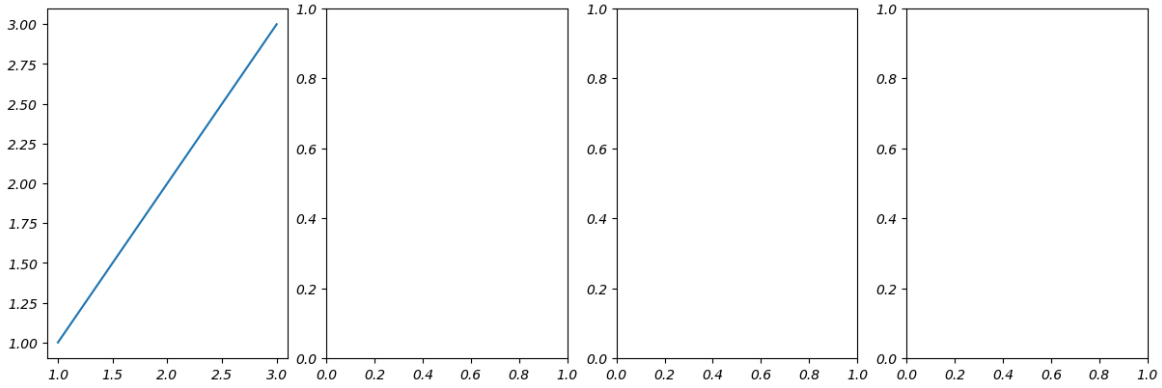
2) How to add Artists to containers

In [10]:
```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=1, ncols=4, figsize=(12, 4), layout='constrained')
```



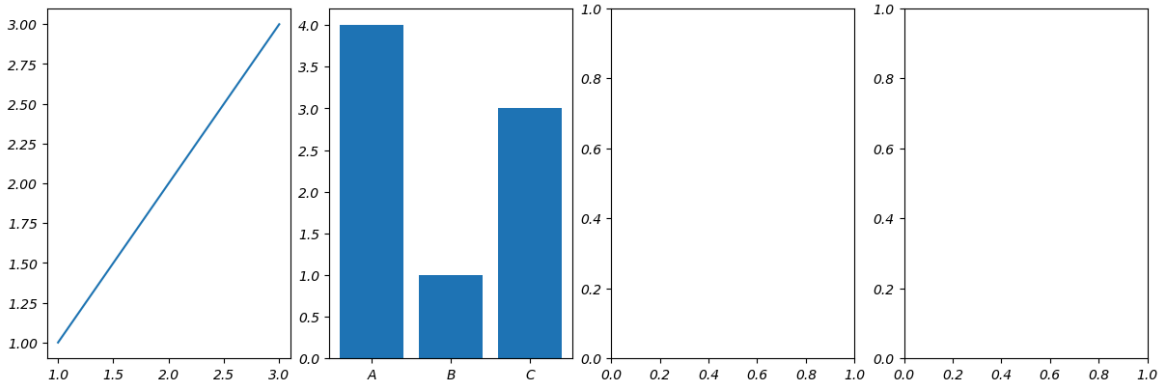
In [11]:
```
# Add lines to first Axes object
axs[0].plot([1, 2, 3], [1, 2, 3])

# Display the figure
fig
```

Out[11]:



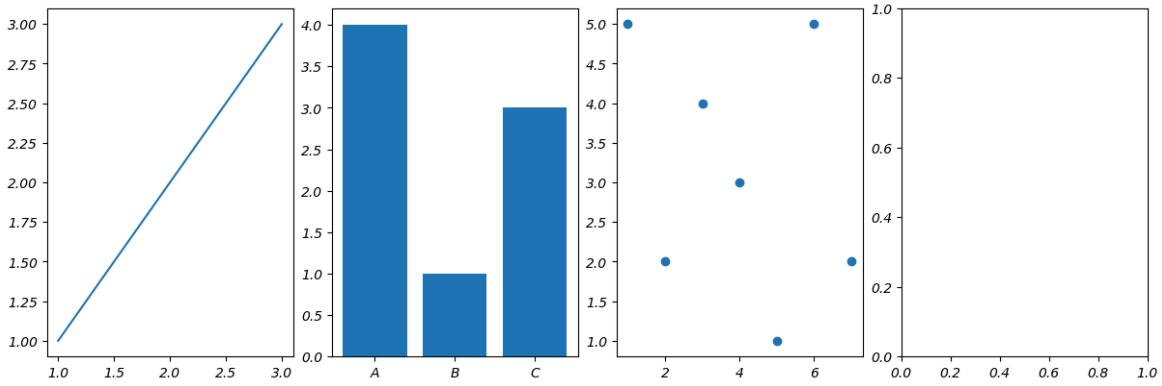
In [12]:
```
# Add bars to second Axes object
axs[1].bar(['A', 'B', 'C'], [4, 1, 3])

# Display the figure
fig
```

Out[12]:



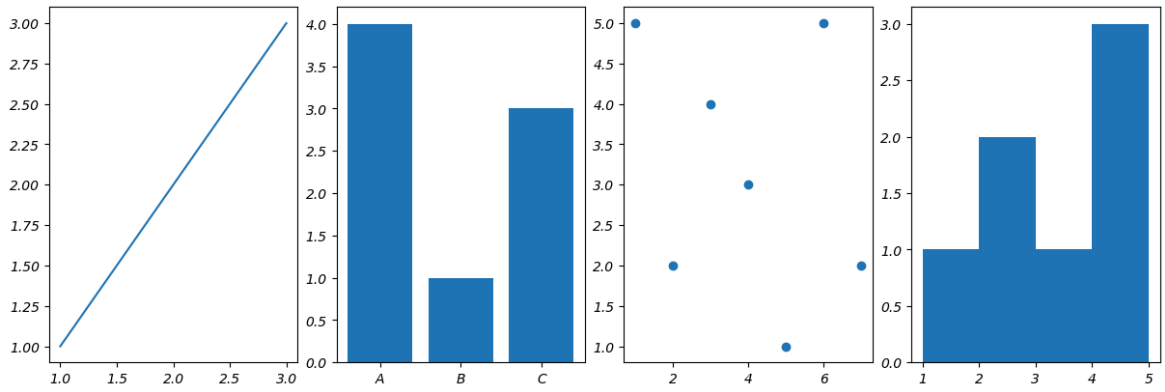
In [13]:
```
# Add markers to third Axes object
axs[2].scatter([1, 2, 3, 4, 5, 6, 7], [5, 2, 4, 3, 1, 5, 2])

# Display the figure
fig
```

Out[13]:



In [14]:
```
# Add histogram to fourth Axes object
axs[3].hist([5, 2, 4, 3, 1, 5, 2], bins=4)

# Display the figure
fig
```

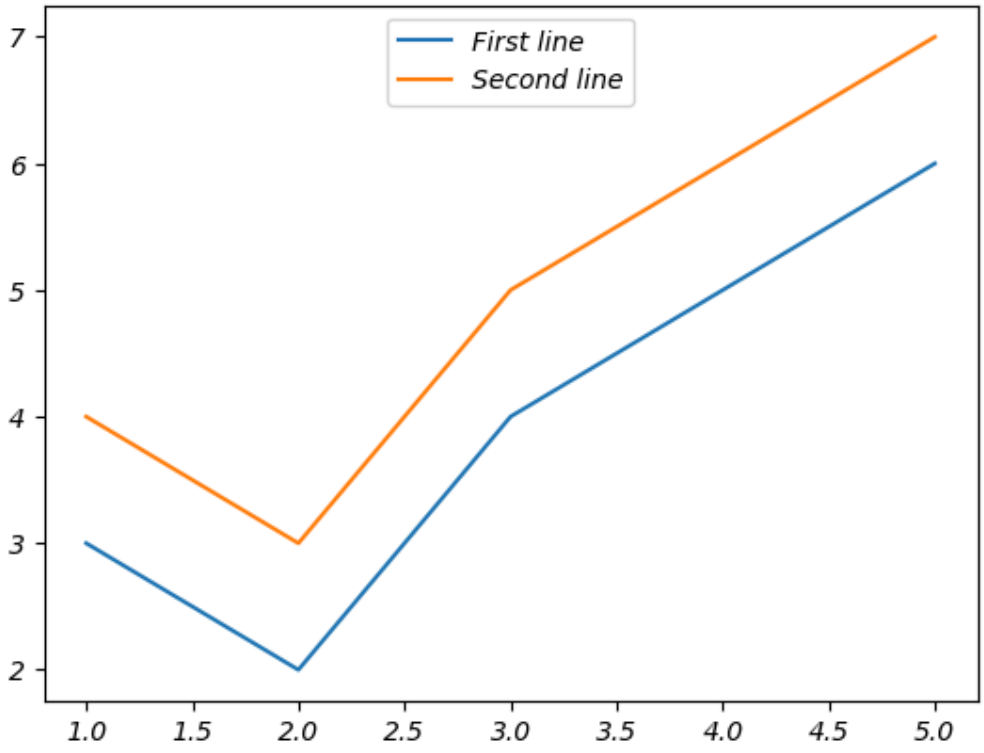
Out[14]:



How to add a legend

In [15]:

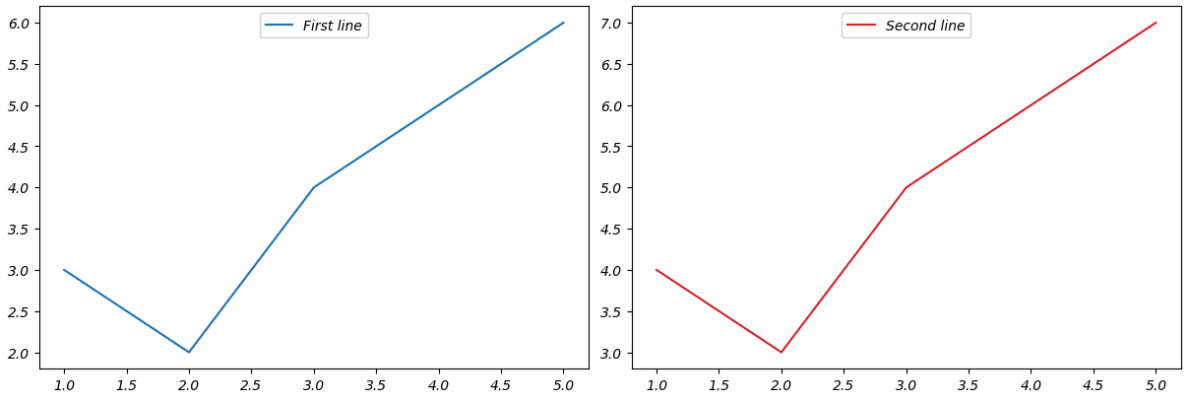
```
# Create a Figure and Axes objects
fig, ax = plt.subplots()

# Add first line
ax.plot([1, 2, 3, 4, 5], [3, 2, 4, 5, 6], label='First line')

# Add second line
ax.plot([1, 2, 3, 4, 5], [4, 3, 5, 6, 7], label='Second line')

# Add a legend
ax.legend(loc='upper center');
```



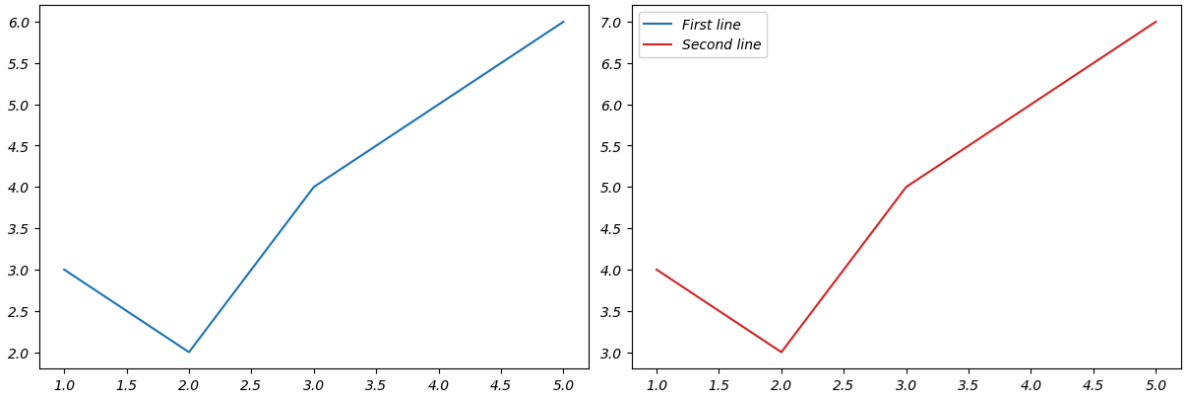
In [16]:

```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=1, ncols=2, layout='constrained', figsize=(12, 4))

# Add first line to first subplot
axs[0].plot([1, 2, 3, 4, 5], [3, 2, 4, 5, 6], color='tab:blue', label='First line')

# Add a legend
axs[0].legend(loc='upper center');

# Add second line to second subplot
```

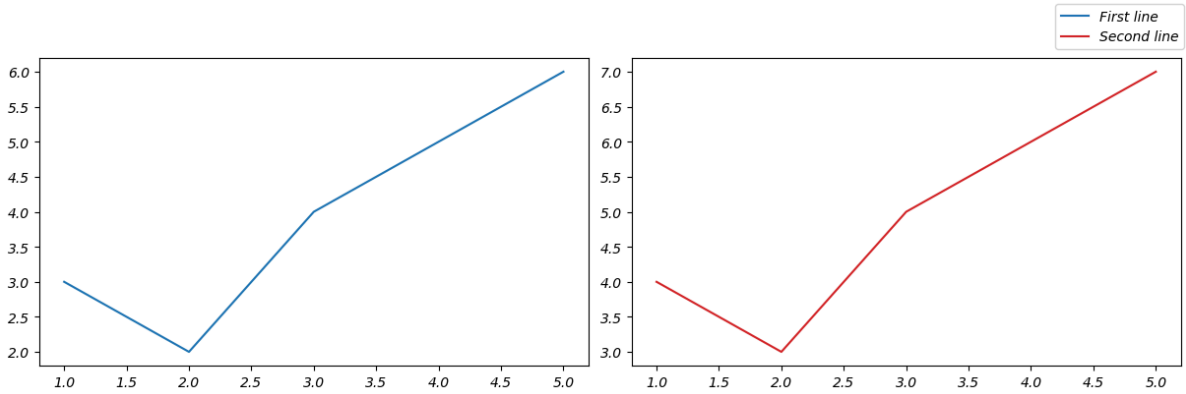
```
axs[1].plot([1, 2, 3, 4, 5], [4, 3, 5, 6, 7], color='tab:red', label='Second line')

# Add a legend
axs[1].legend(loc='upper center');
```



In [17]:
```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=1, ncols=2, layout='constrained', figsize=(12, 4))

# Add first line to first subplot
line1,  = axs[0].plot([1, 2, 3, 4, 5], [3, 2, 4, 5, 6], color='tab:blue')

# Add second line to second subplot
line2,  = axs[1].plot([1, 2, 3, 4, 5], [4, 3, 5, 6, 7], color='tab:red')

# Add a legend to second subplot
axs[1].legend([line1, line2], ['First line', 'Second line']);
```

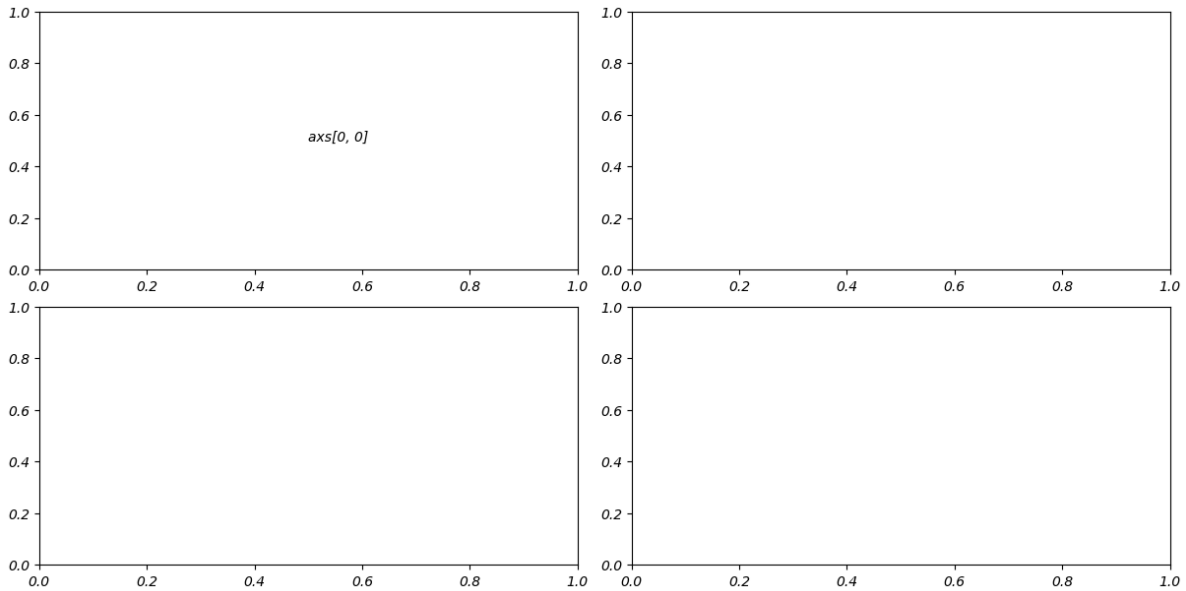


In [18]:
```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=1, ncols=2, layout='constrained', figsize=(12, 4))

# Add first line to first subplot
axs[0].plot([1, 2, 3, 4, 5], [3, 2, 4, 5, 6], color='tab:blue', label='First line')

# Add second line to second subplot
axs[1].plot([1, 2, 3, 4, 5], [4, 3, 5, 6, 7], color='tab:red', label='Second line')

# Add a legend above the second subplot
fig.legend(loc='outside upper right');
```



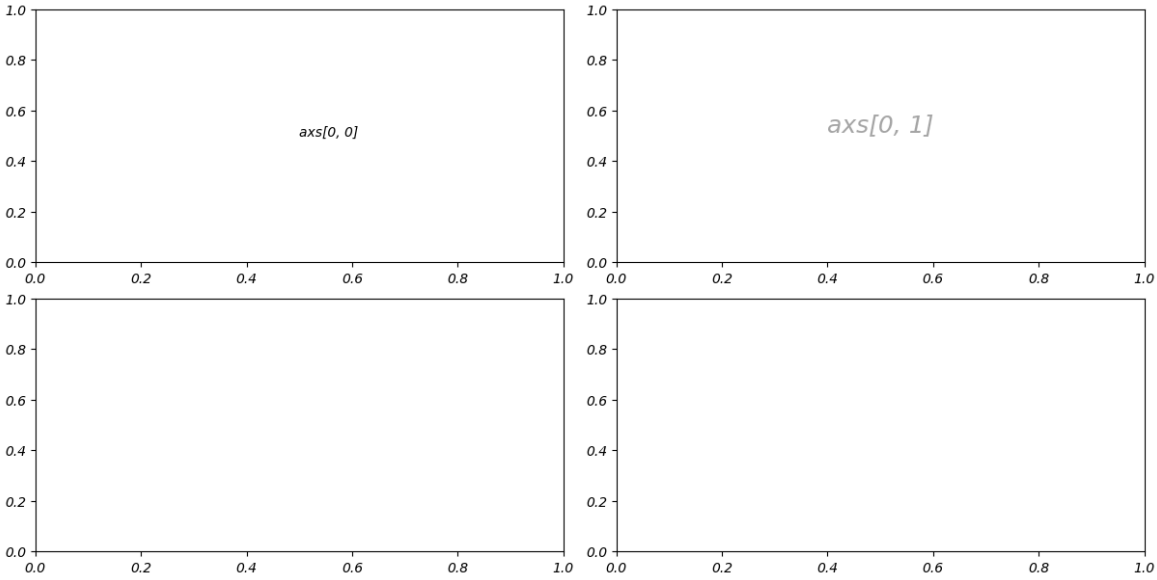
How to add text

In [19]:
```
# Create Figure and Axes objects
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(12, 6), layout='constrained')

# Annotate top left subplot
axs[0, 0].annotate(text='axs[0, 0]', xy=(0.5, 0.5));
```



In [20]:
```
# Annotate top right subplot
axs[0, 1].annotate(text='axs[0, 1]',
                   xy=(0.5, 0.5),
                   horizontalalignment='center',
                   verticalalignment='bottom',
                   fontsize=18,
                   color='darkgray')

# Display the figure
fig
```

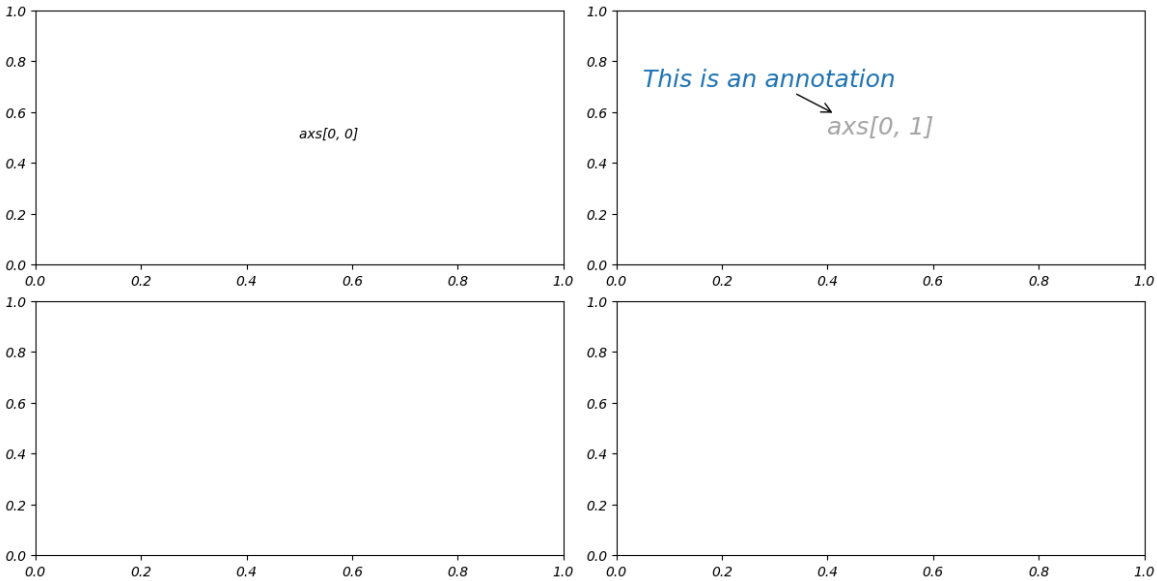
Out[20]:



In [21]:

```
# Annotate top right subplot again
axs[0, 1].annotate(text='This is an annotation',
                   xy=(0.5, 0.5),
                   xytext=(0.05, 0.7),
                   arrowprops={'arrowstyle': '->', 'shrinkA': 0, 'shrinkB': 40},
                   fontsize=18,
                   color='tab:blue');

# Display the figure
fig
```

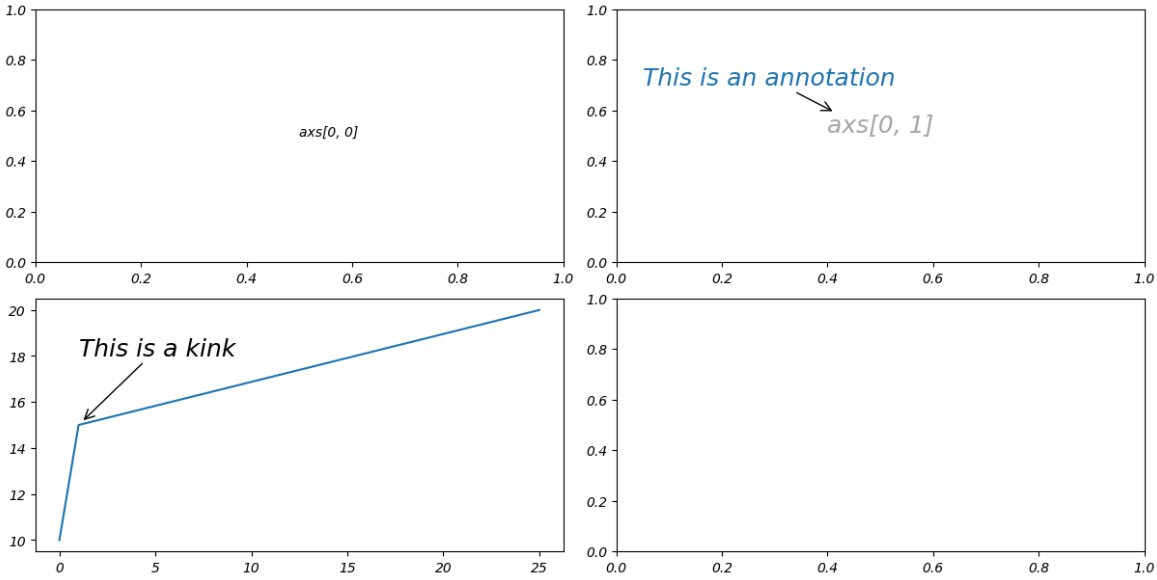
Out[21]:



In [22]:

```
# Add line to bottom left subplot
axs[1, 0].plot([0, 1, 25], [10, 15, 20])

# Annotate subplot
axs[1, 0].annotate(text='This is a kink',
                   xy=(1, 15),
                   xytext=(1, 18),
                   arrowprops={'arrowstyle': '->', 'shrinkA': 0, 'shrinkB': 5},
                   fontsize=18);

# Display the figure
fig
```

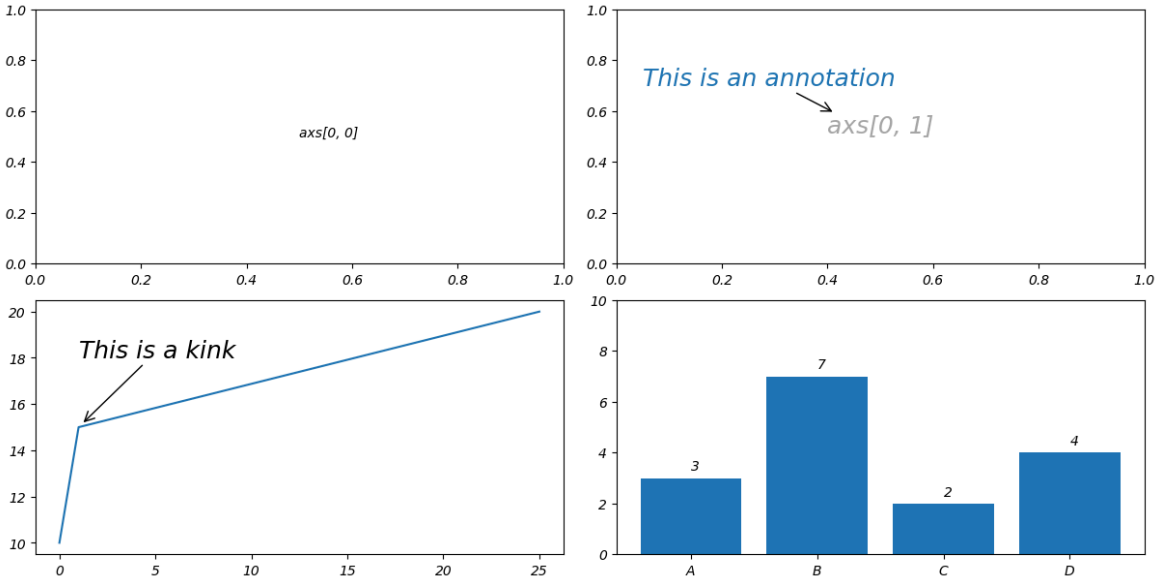
Out[22]:



In [23]:

```
# Add bars to bottom right subplot
axs[1, 1].bar(['A', 'B', 'C', 'D'], [3, 7, 2, 4])

# Annotate the four bars
for i in range(4):

    # Get the height of the bar
    height_of_bar = axs[1, 1].patches[i].get_height()

    # Annotate the bar
    axs[1, 1].annotate(text=height_of_bar, xy=(i, height_of_bar + 0.3))

# Adjust ylim
axs[1, 1].set_ylim(0, 10)

# Display the figure
fig
```

Out[23]:

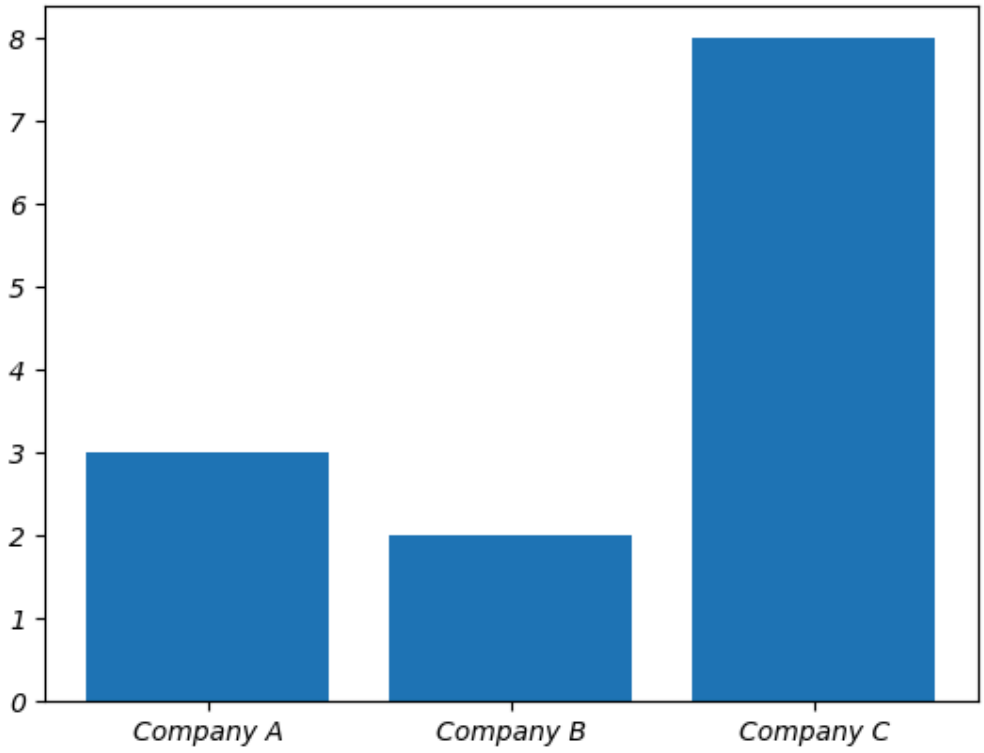


3) How to customize Artists in your plot

In [24]:

```
# Create Figure and Axes objects
fig, ax = plt.subplots()
```

```
# Add bars
ax.bar(['Company A', 'Company B', 'Company C'], [3, 2, 8]);
```



Each Artist has some properties. You can get the properties of an artist using the properties() function.

In [25]:
```
# Get bars
for bar in ax.patches:
    print(bar)
```

```
Rectangle(xy=(-0.4, 0), width=0.8, height=3, angle=0)
Rectangle(xy=(0.6, 0), width=0.8, height=2, angle=0)
Rectangle(xy=(1.6, 0), width=0.8, height=8, angle=0)
```

In [26]:
```
# Get properties of first bar
bars = ax.patches
bars[0].properties()
```

Out[26]:

```
{'agg_filter': None,
 'alpha': None,
 'angle': 0.0,
 'animated': False,
 'antialiased': True,
 'bbox': Bbox([[-0.4, 0.0], [0.4, 3.0]]),
 'capstyle': 'butt',
 'center': array([0. , 1.5]),
 'children': [],
 'clip_box': <matplotlib.transforms.TransformedBbox at 0x1c6203505d0>,
 'clip_on': True,
 'clip_path': None,
 'corners': array([[-0.4,  0. ],
       [ 0.4,  0. ],
       [ 0.4,  3. ],
       [-0.4,  3. ]]),
 'data_transform': <matplotlib.transforms.CompositeGenericTransform at 0x1c62031a550
>,
 'edgecolor': (0.0, 0.0, 0.0, 0.0),
 'extents': Bbox([[102.54545454545455, 52.8], [231.37662337662334, 184.7999999999999
5]]),
 'facecolor': (0.12156862745098039,
  0.4666666666666667,
  0.7058823529411765,
  1.0),
 'figure': <Figure size 640x480 with 1 Axes>,
 'fill': True,
 'gid': None,
 'hatch': None,
 'hatch_linewidth': 1.0,
 'height': np.int64(3),
 'in_layout': True,
 'joinstyle': 'miter',
 'label': '_nolegend_',
 'linestyle': 'solid',
 'linewidth': 1.0,
 'mouseover': False,
 'patch_transform': <matplotlib.transforms.CompositeGenericTransform at 0x1c6216a71d0
>,
 'path': Path(array([[0., 0.],
       [1., 0.],
       [1., 1.],
       [0., 1.],
       [0., 0.]]), array([ 1,  2,  2,  2, 79], dtype=uint8)),
 'path_effects': [],
 'picker': None,
 'rasterized': False,
 'sketch_params': None,
 'snap': None,
 'tightbbox': Bbox([[102.54545454545455, 52.8], [231.37662337662334, 184.799999999999
95]]),
 'transform': <matplotlib.transforms.CompositeGenericTransform at 0x1c6216a4d50>,
 'transformed_clip_path_and_affine': (None, None),
 'url': None,
 'verts': array([[102.54545455,  52.8       ],
       [231.37662338,  52.8       ],
       [231.37662338, 184.8       ],
       [102.54545455, 184.8       ],
       [102.54545455,  52.8       ]]),
 'visible': True,
 'width': np.float64(0.8),
 'window_extent': Bbox([[102.54545454545455, 52.8], [231.37662337662334, 184.79999999
999995]]),
```

```
 'x': np.float64(-0.4),
 'xy': (np.float64(-0.4), np.int64(0)),
 'y': np.int64(0),
 'zorder': 1}
```

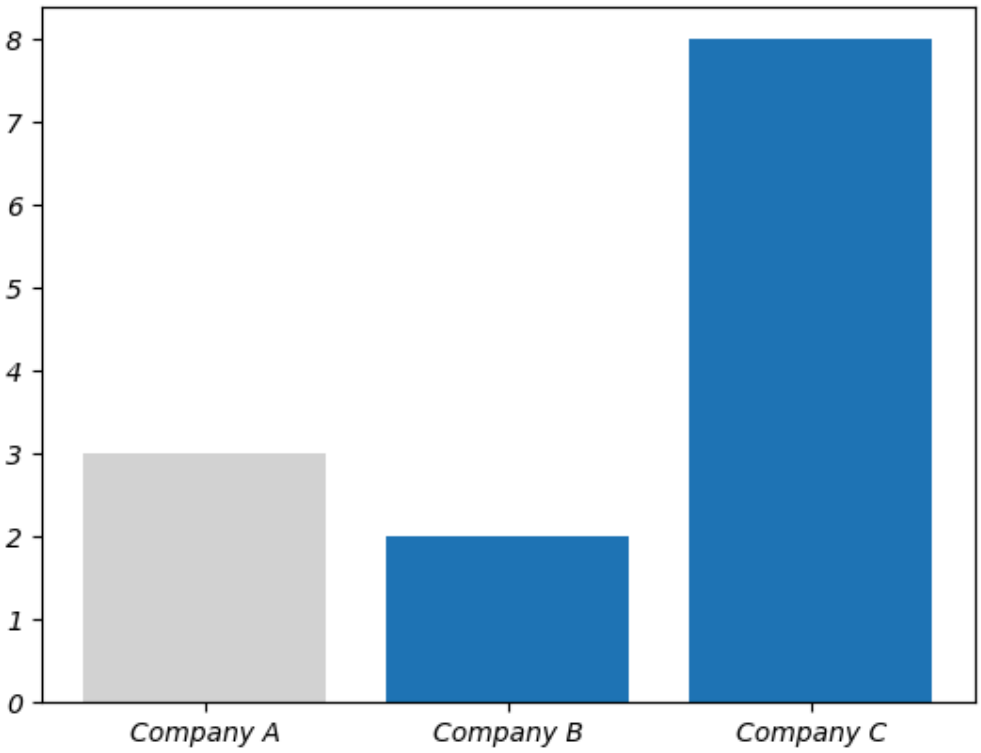
Each of the properties is accessed with a setter (**set_property()**) or getter (**get_property()**) function

In [27]:
```
# Get the facecolor of the first bar
bars[0].get_facecolor()
```

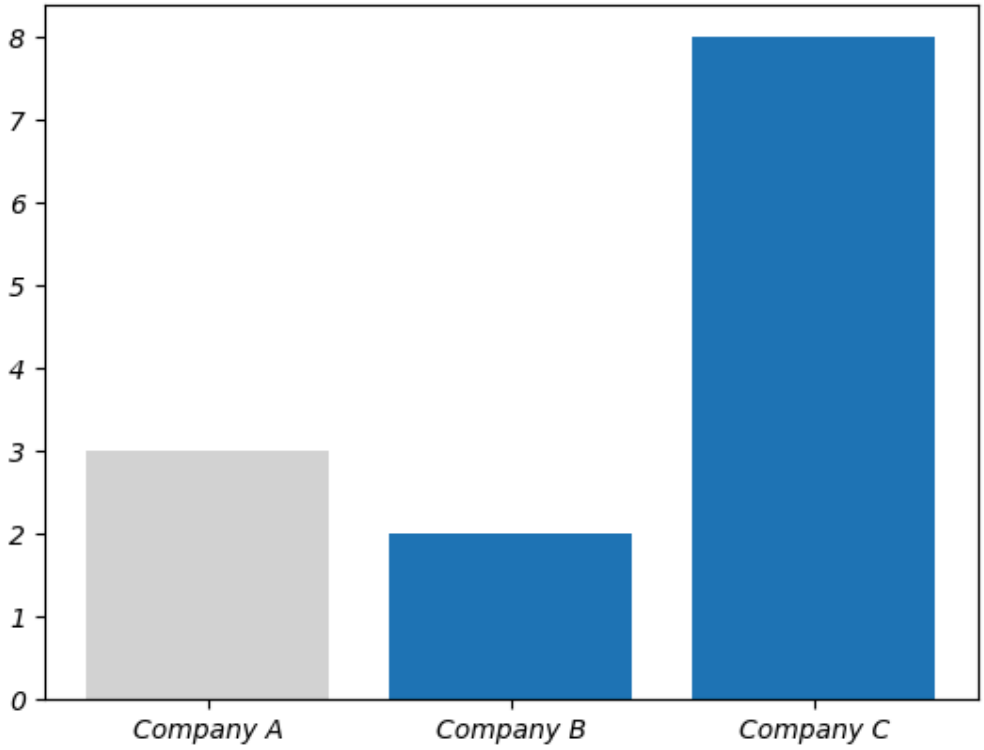
Out[27]:
```
(0.12156862745098039, 0.4666666666666667, 0.7058823529411765, 1.0)
```

In [28]:
```
# Set the facecolor of the first bar
bars[0].set_facecolor('lightgray')

# Display the figure
fig
```

Out[28]:

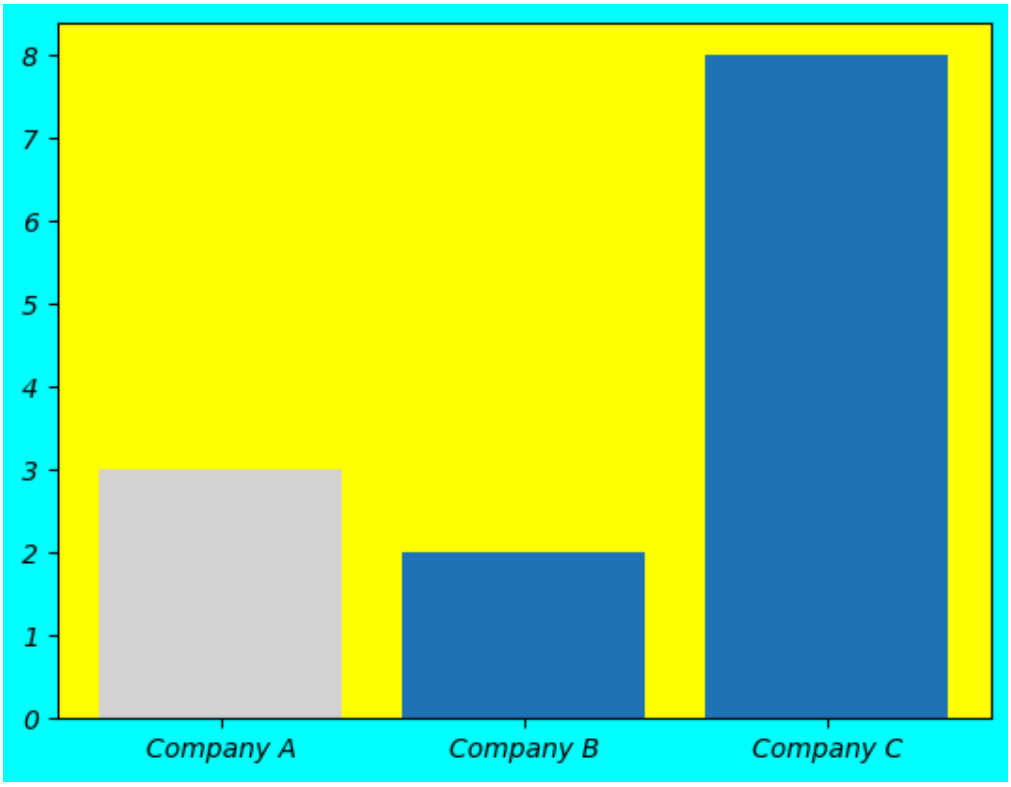


In [29]:
```
# Create Figure and Axes objects
fig, ax = plt.subplots()

# Add bars
ax.bar(['Company A', 'Company B', 'Company C'], [3, 2, 8], color=['lightgray', 'tab:bl
```



In [30]:
```
# Set properties
fig.set_facecolor('cyan')
ax.set_facecolor('yellow')

# Display the figure
fig
```

Out[30]:



In [31]:
```
# Set multiple properties in one line
fig.set(facecolor='red', linewidth=10, edgecolor='magenta')

# Display the figure
fig
```

Out[31]: