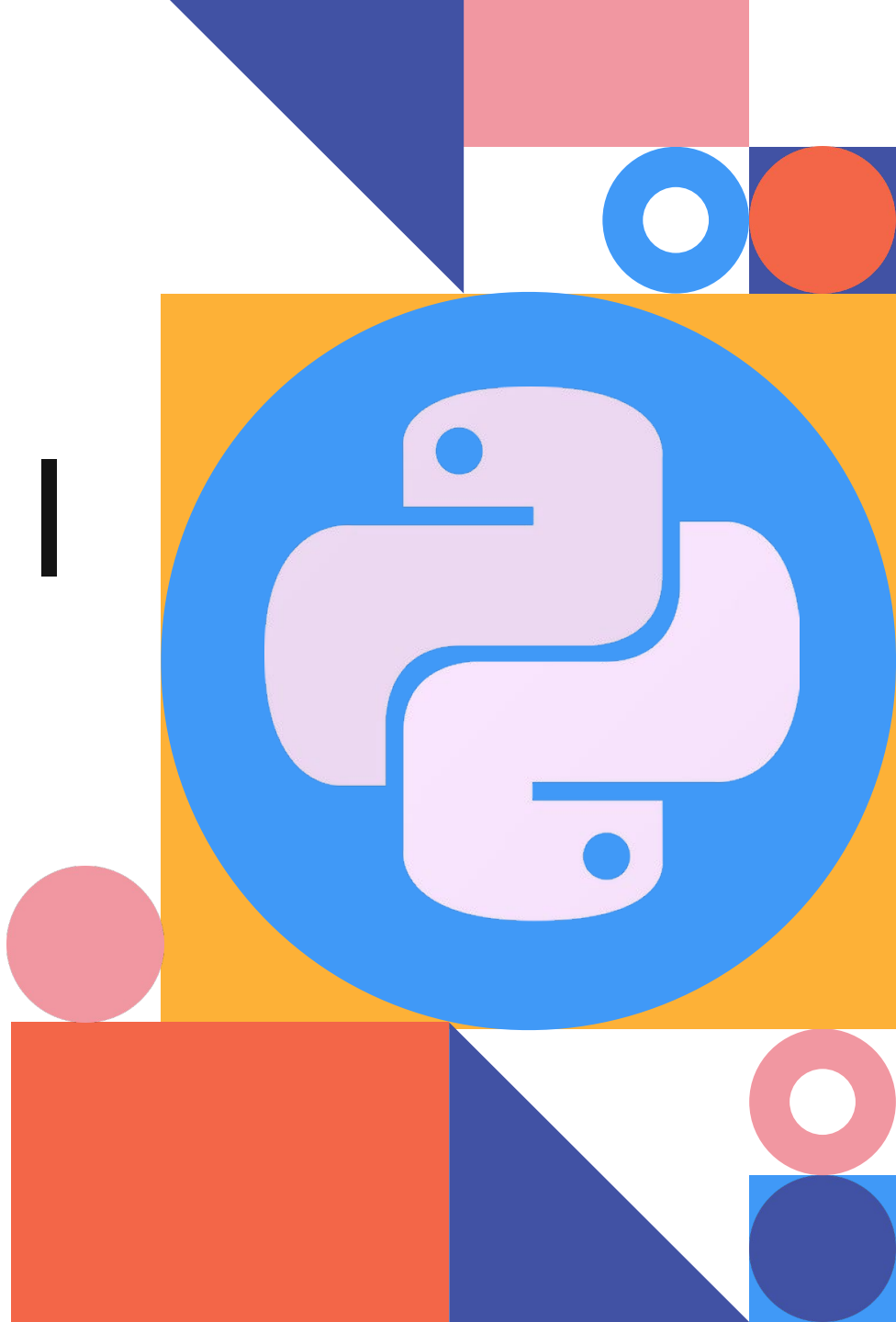
Data Visualization I



Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

Contents

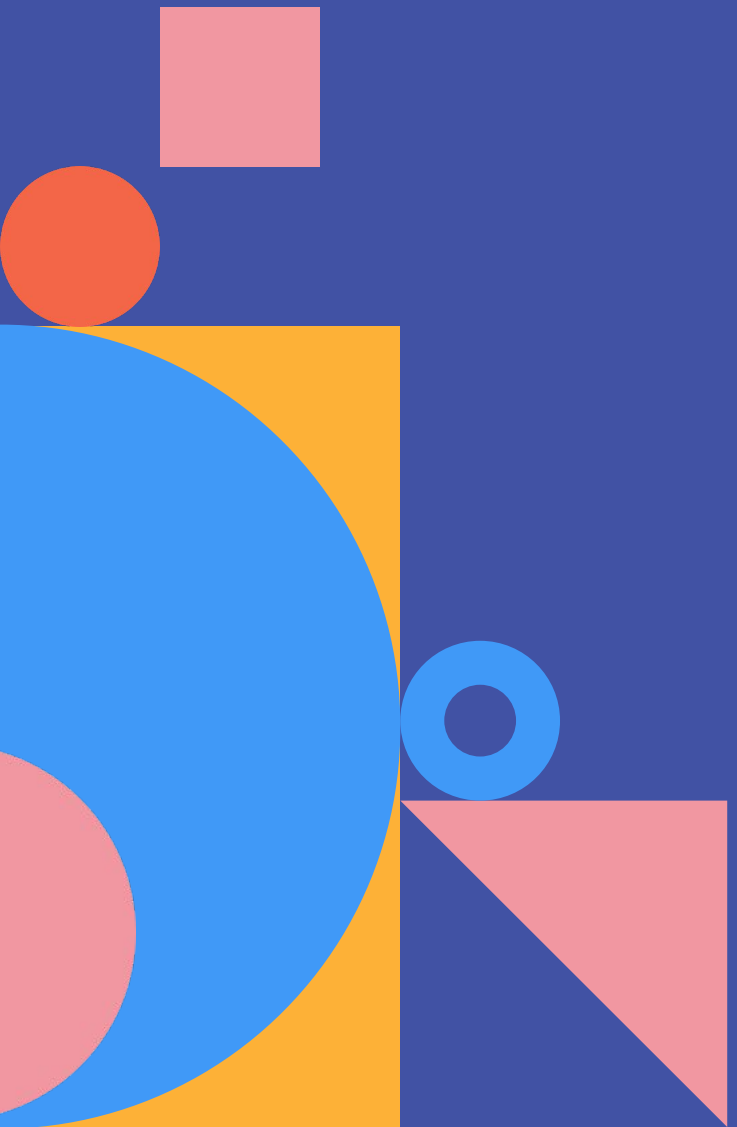
01 Visualization

Matplotlib

02 Basic plotting

Plot

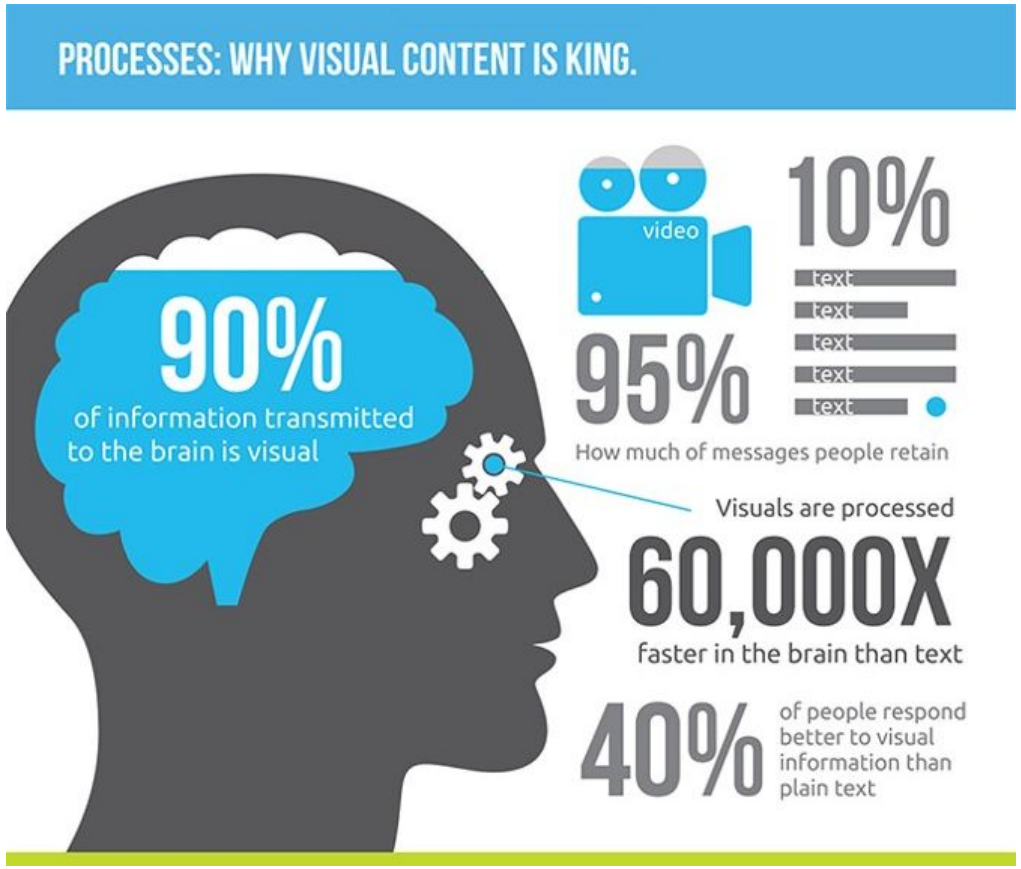
03 Subplot



01 Visualization

Why Visuals?

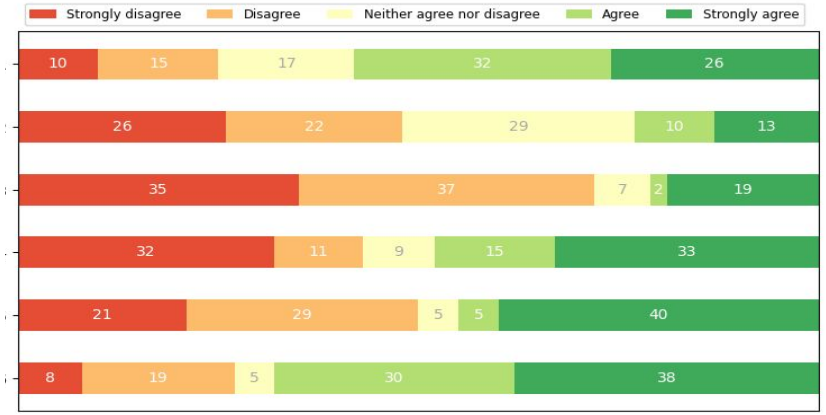
- For exploratory data analysis
- Communicate data clearly
- Share unbiased presentation of data
- Support recommendations

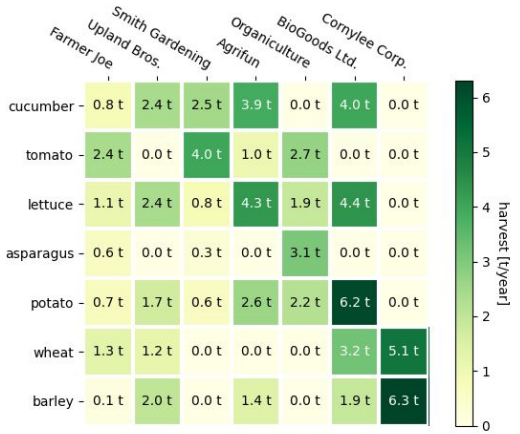


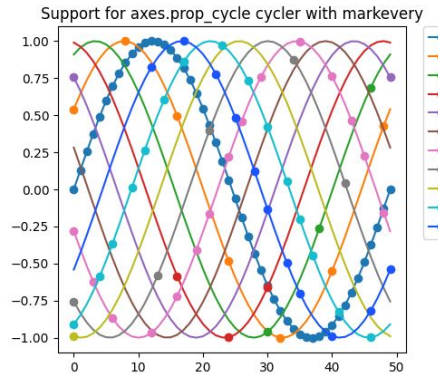
matplotlib

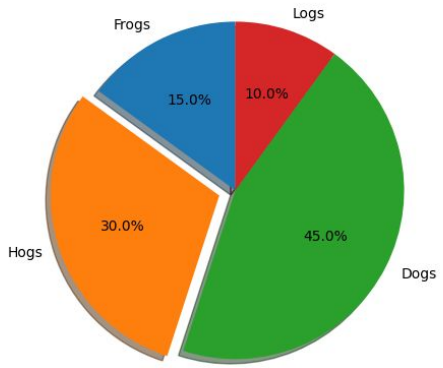
A comprehensive library for creating static, animated, and interactive visualizations in Python

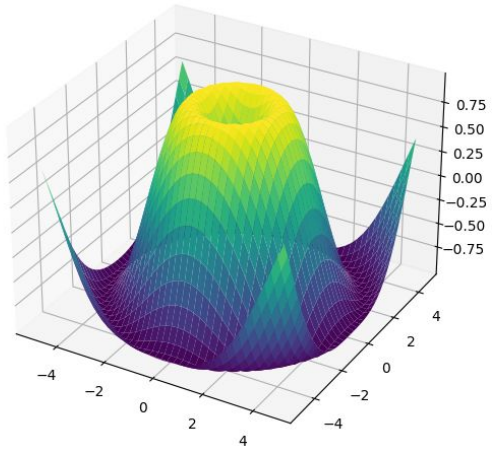
```
$ pip install matplotlib
```







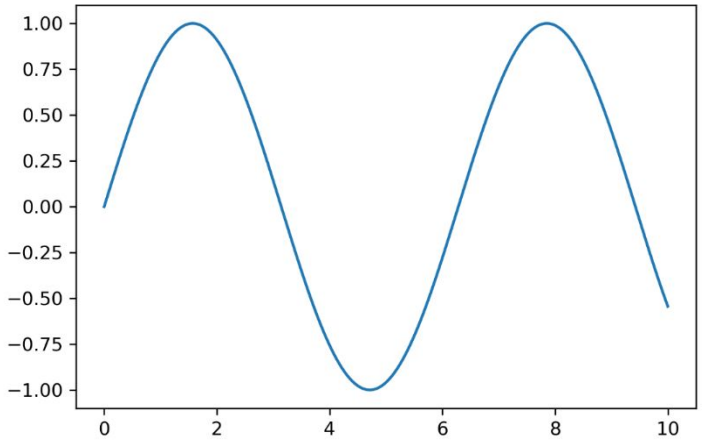




02 Basic Plotting

Plot from “Data”

```
import matplotlib.pyplot as plt
import numpy as np
plt.plot(y=np.sin(x))
```



Matplotlib does not understand functions!

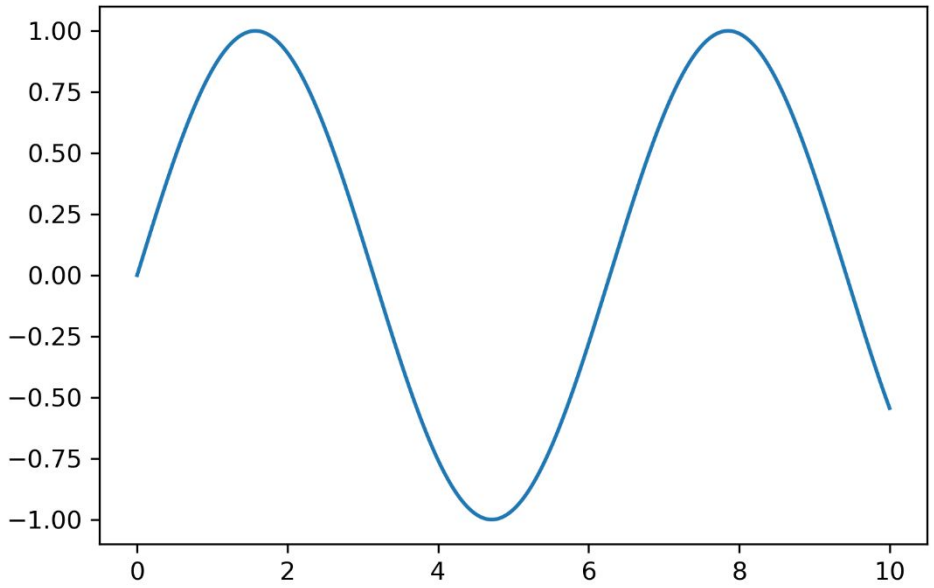
Our Strategy

1. Generate the numeric values of a function over a specific range
2. Display the data “points” in a graphical way

Simple Line Plot

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 1000)
plt.plot(x, np.sin(x))
```

```
plt.show()
```

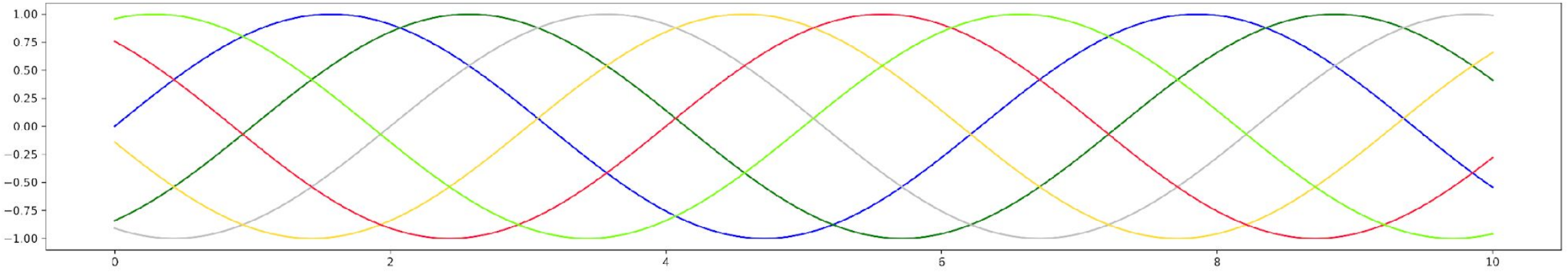


Try

```
x = np.linspace(0, 10, 10)
plt.plot(x, np.sin(x))
```

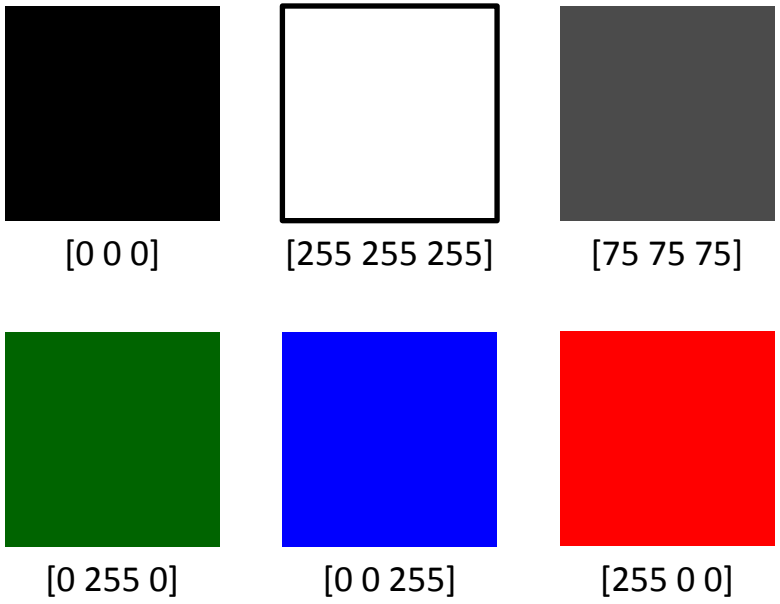
Line Color

```
plt.plot(x, np.sin(x - 0), color='blue')        # specify color by name
plt.plot(x, np.sin(x - 1), color='g')           # short color code (rgbcmyk)
plt.plot(x, np.sin(x - 2), color='0.75')        # Grayscale between 0 and 1
plt.plot(x, np.sin(x - 3), color='#FFDD44')     # Hex code (RRGGBB from 00 to FF)
plt.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3)) # RGB tuple, values 0 to 1
plt.plot(x, np.sin(x - 5), color='chartreuse')  # all HTML color names supported
```



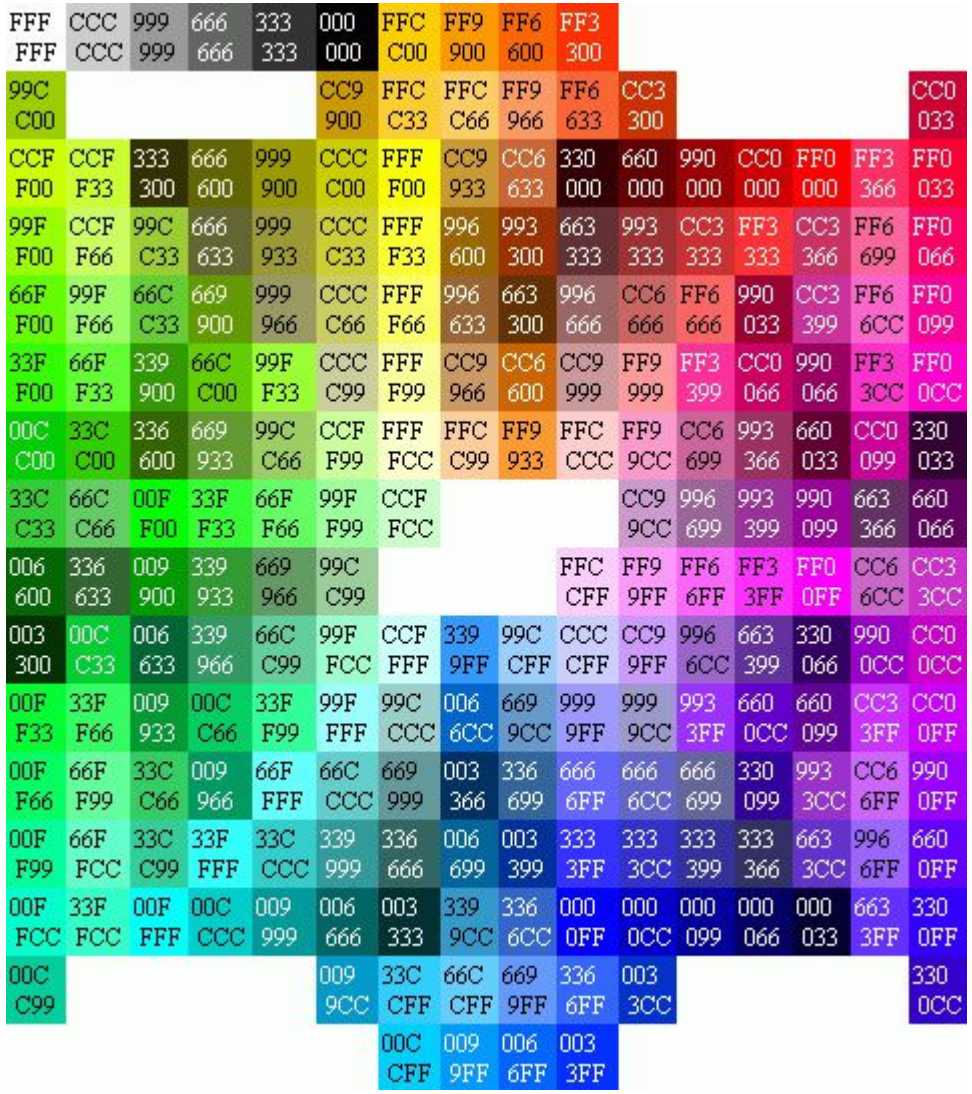
Colo Space

8-bit equivalence:



[0 0 0] [255 255 255] [75 75 75]

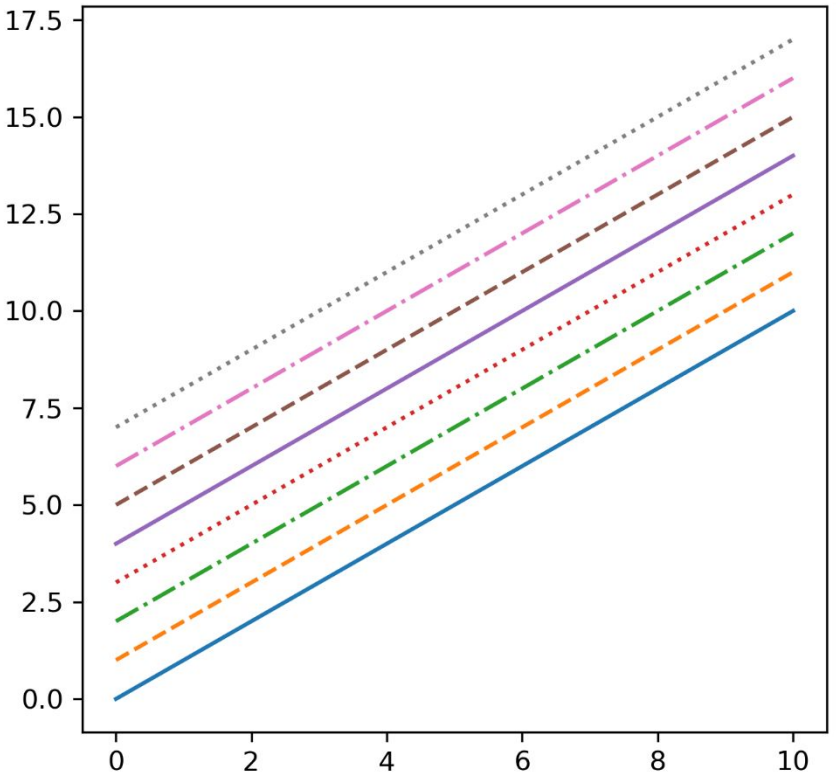
[0 255 0] [0 0 255] [255 0 0]



Line Style

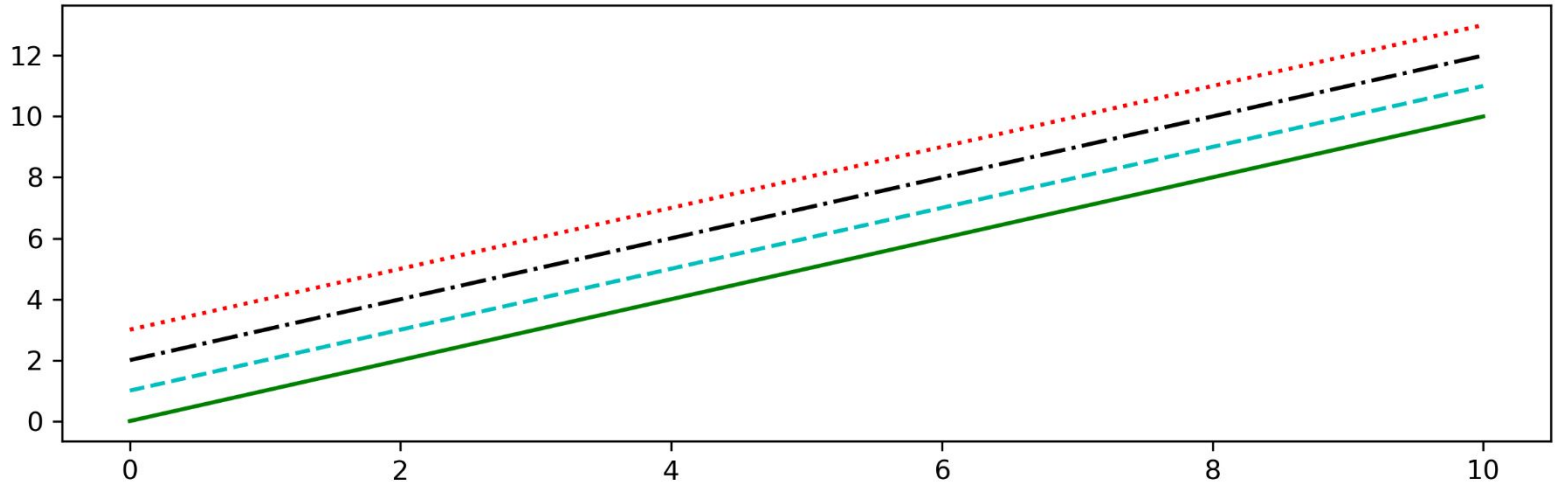
```
plt.plot(x, x + 0, linestyle='solid')
plt.plot(x, x + 1, linestyle='dashed')
plt.plot(x, x + 2, linestyle='dashdot')
plt.plot(x, x + 3, linestyle='dotted');

# For short, you can use the following codes:
plt.plot(x, x + 4, linestyle='-')  # solid
plt.plot(x, x + 5, linestyle='--') # dashed
plt.plot(x, x + 6, linestyle='-.') # dashdot
plt.plot(x, x + 7, linestyle=':');  # dotted
```



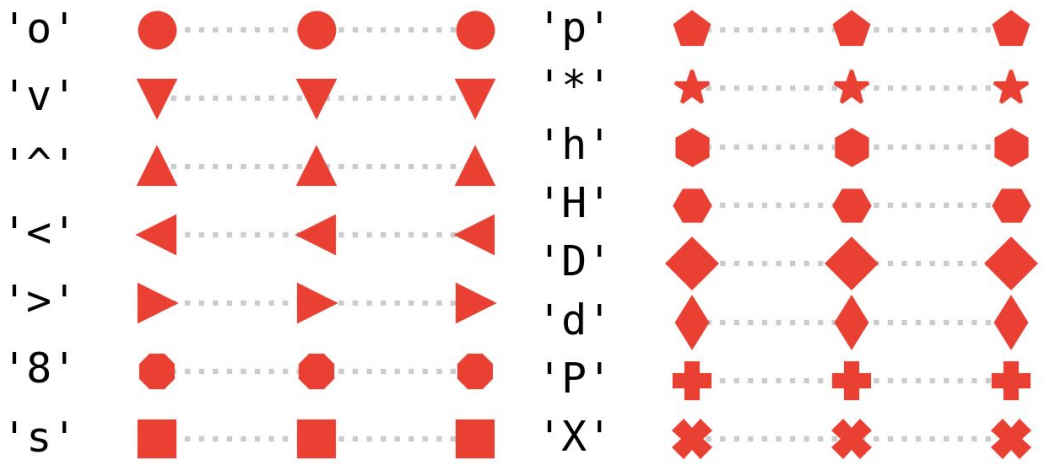
Line Color and Style Together

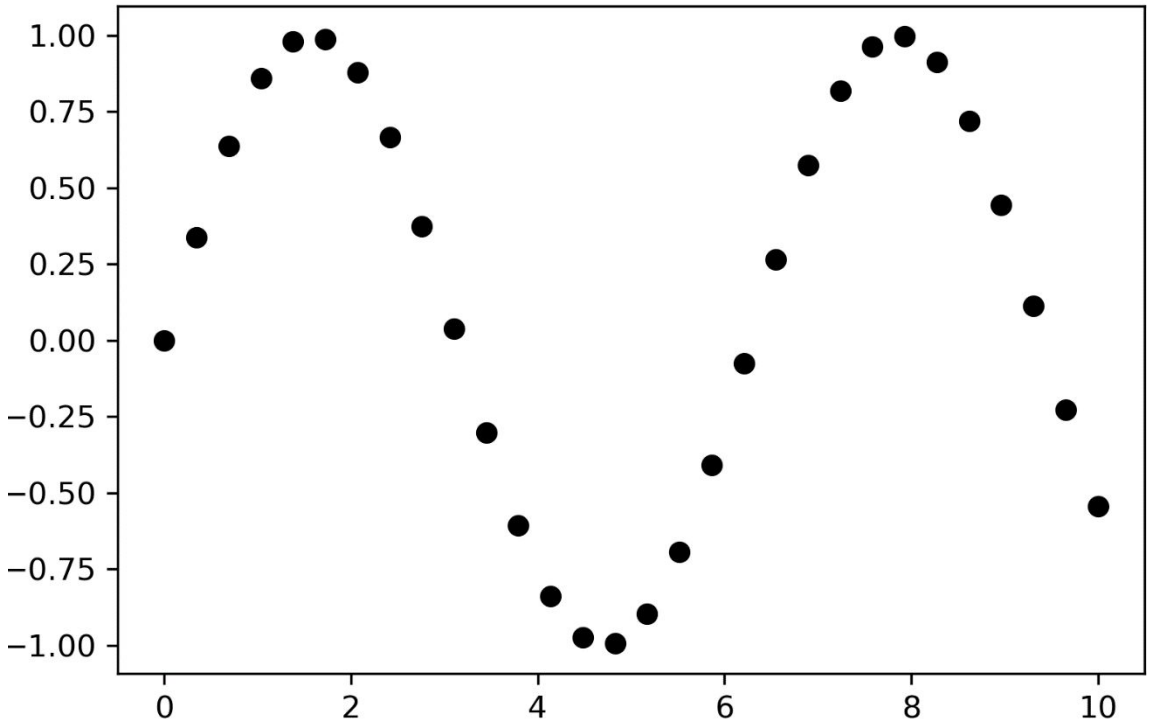
```
plt.plot(x, x + 0, '-g')  # solid green
plt.plot(x, x + 1, '--c') # dashed cyan
plt.plot(x, x + 2, '-.k') # dashdot black
plt.plot(x, x + 3, ':r');  # dotted red
```



Marker

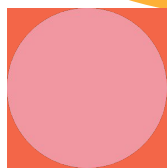
```
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.plot(x, y, 'o', color='black')
```





Quiz

What will `plt.plot(x, y, '-ok')` shows?



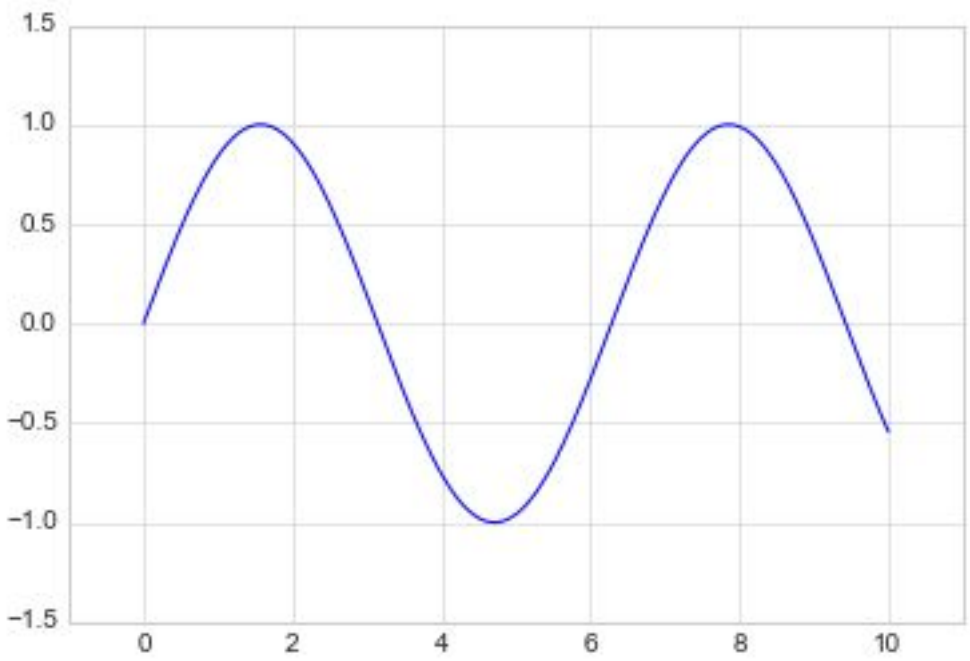
Axes Limits

```
x = np.linspace(0, 10, 1000)
plt.plot(x, np.sin(x))

plt.xlim(-1, 11)
plt.ylim(-1.5, 1.5);
```

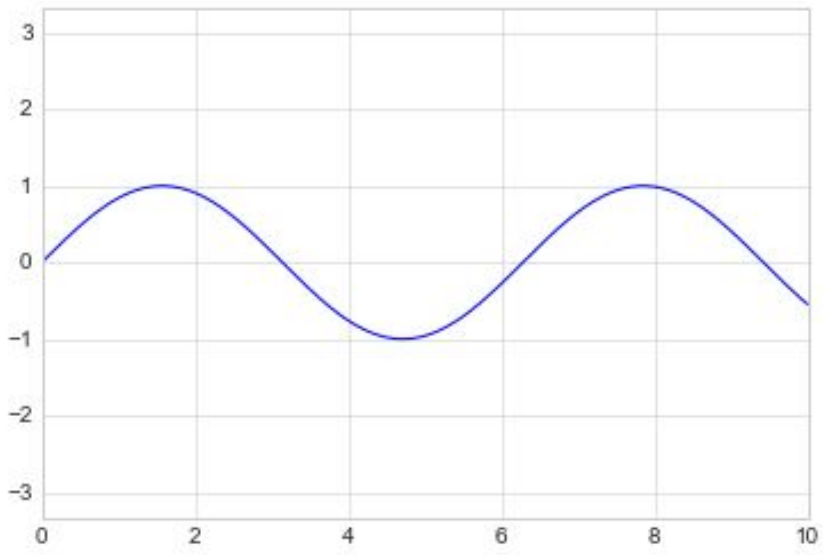
[xmin, xmax, ymin, ymax]

```
plt.axis([-1, 11, -1.5, 1.5])
```

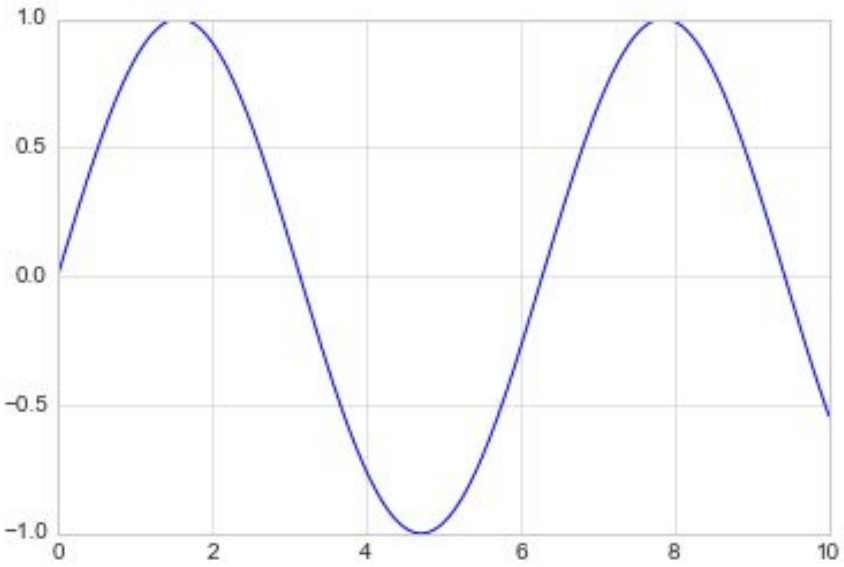


Axis Equal and Tight

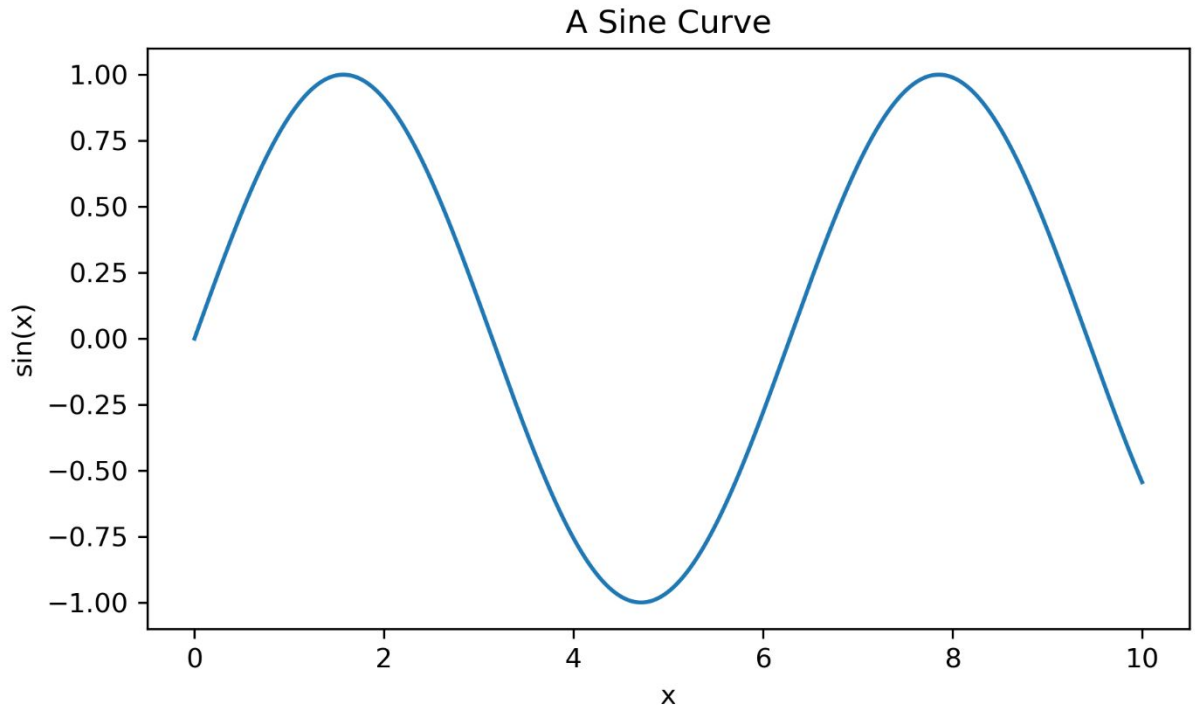
```
plt.plot(x, np.sin(x))
plt.axis('equal');
```




```
plt.plot(x, np.sin(x))
plt.axis('tight');
```



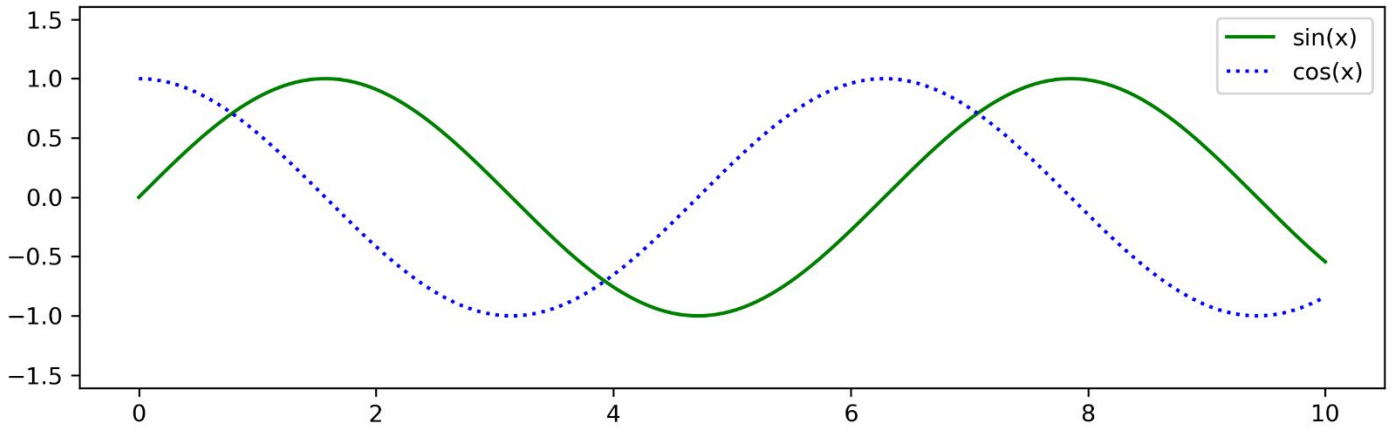
Title and Axis Labels




```
plt.plot(x, np.sin(x))
plt.title("A Sine Curve")
plt.xlabel("x")
plt.ylabel("sin(x)")
```

Legend

```
plt.plot(x, np.sin(x), '-g', label='sin(x)')
plt.plot(x, np.cos(x), ':b', label='cos(x)')
plt.legend()
```

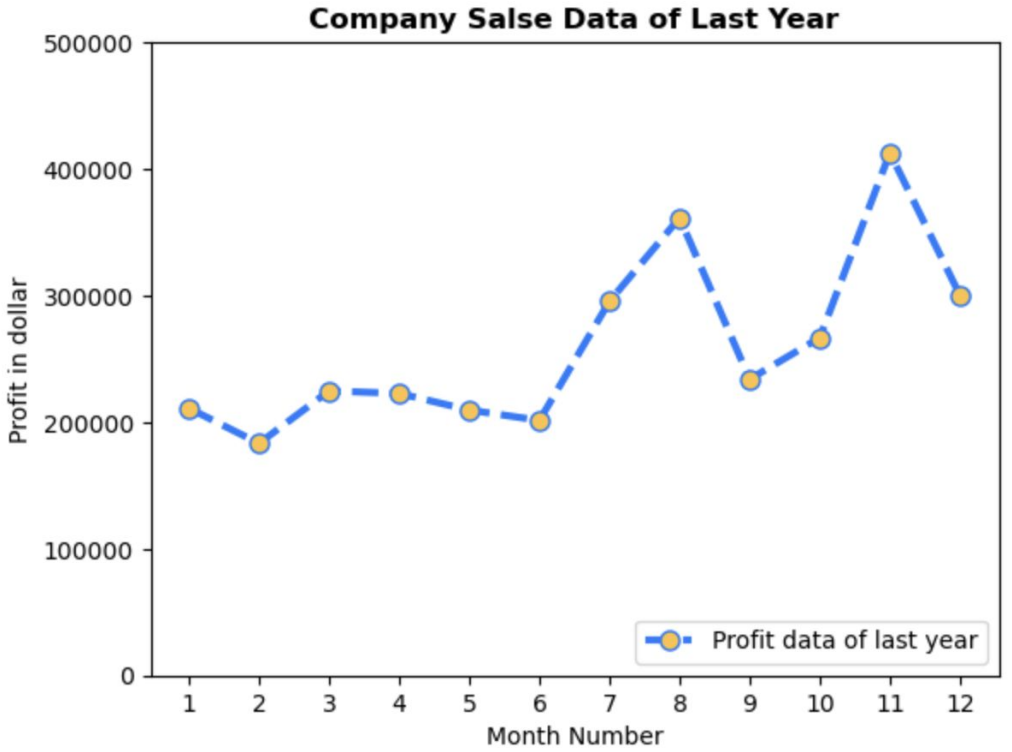


Exercise – The Sales Dataset (15 mins)

Get total profit of all months in sales_data.csv and show a line plot. Try using the same style properties as the following figure.

Several properties you may want to know:

- color
- linestyle
- marker
- markeredgecolor
- markerfacecolor
- **Fontdict (or weight**
- **loc**

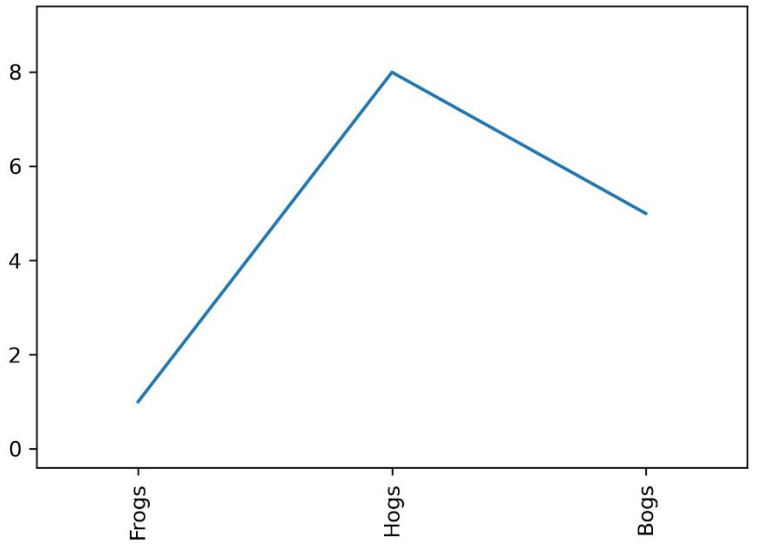


Ticks

```
x = [1, 2, 3]
y = [1, 8, 5]

labels = ['Frogs', 'Hogs', 'Bogs']

plt.plot(x, y)
plt.xticks(x, labels, rotation='vertical')
```

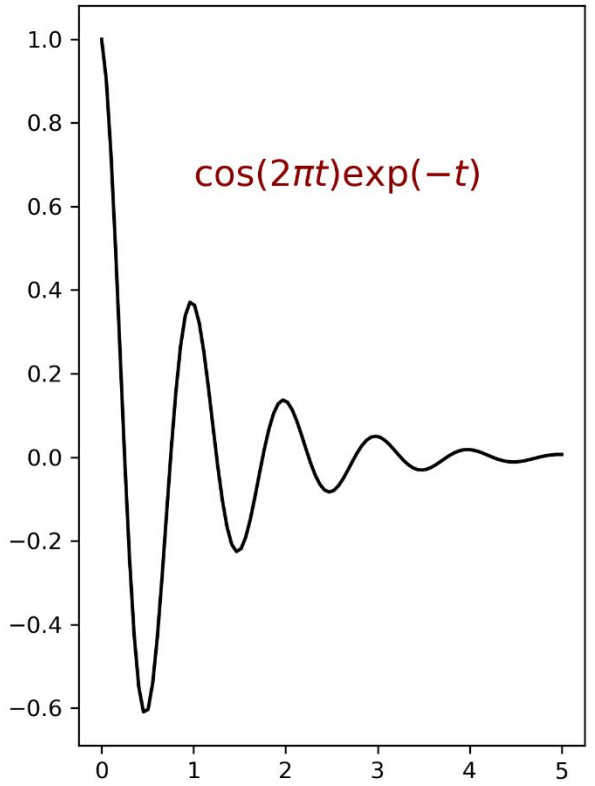


Text and Annotation

```
font = {'family': 'serif',
        'color': 'darkred',
        'weight': 'normal',
        'size': 16,
        }

x = np.linspace(0.0, 5.0, 100)
y = np.cos(2*np.pi*x) * np.exp(-x)

plt.plot(x, y, 'k')
plt.text(2, 0.65, r'$\cos(2 \pi t) \exp(-t
)$', fontdict=font)
```



plt.annotate()

LaTex

Y.F. Kuo & K.T. Yeh

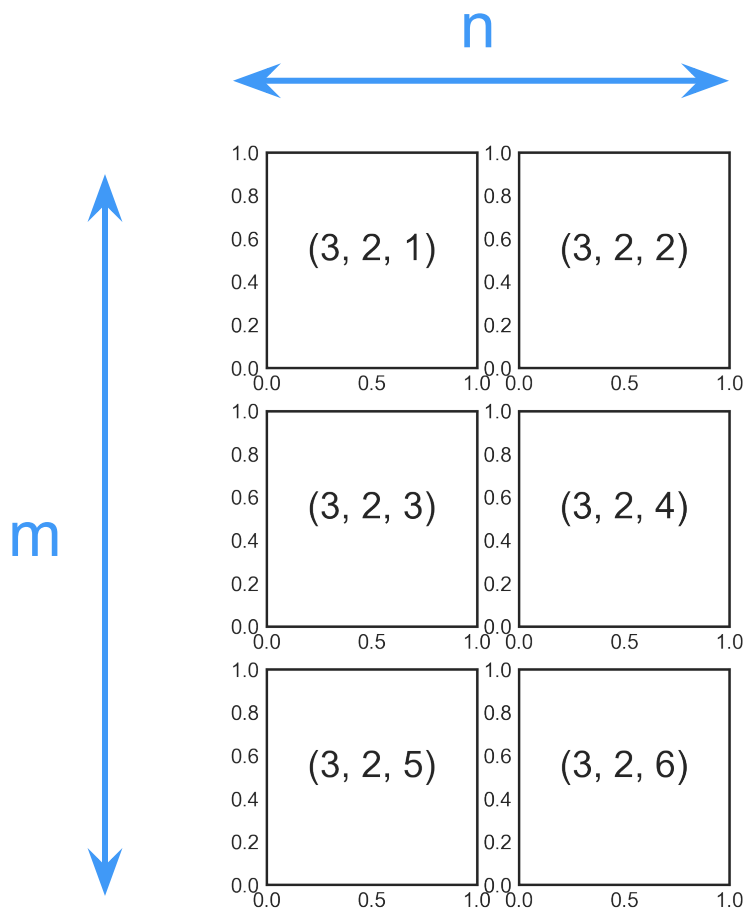
03 Subplot

Subplots

Several small plots “in a figure”

```
plt.subplot(m, n, i)
```

```
for i in range(1, 7):
    plt.subplot(3, 2, i)
    plt.text(0.5, 0.5, str((3, 2, i)),
             fontsize=18, ha='center')
```

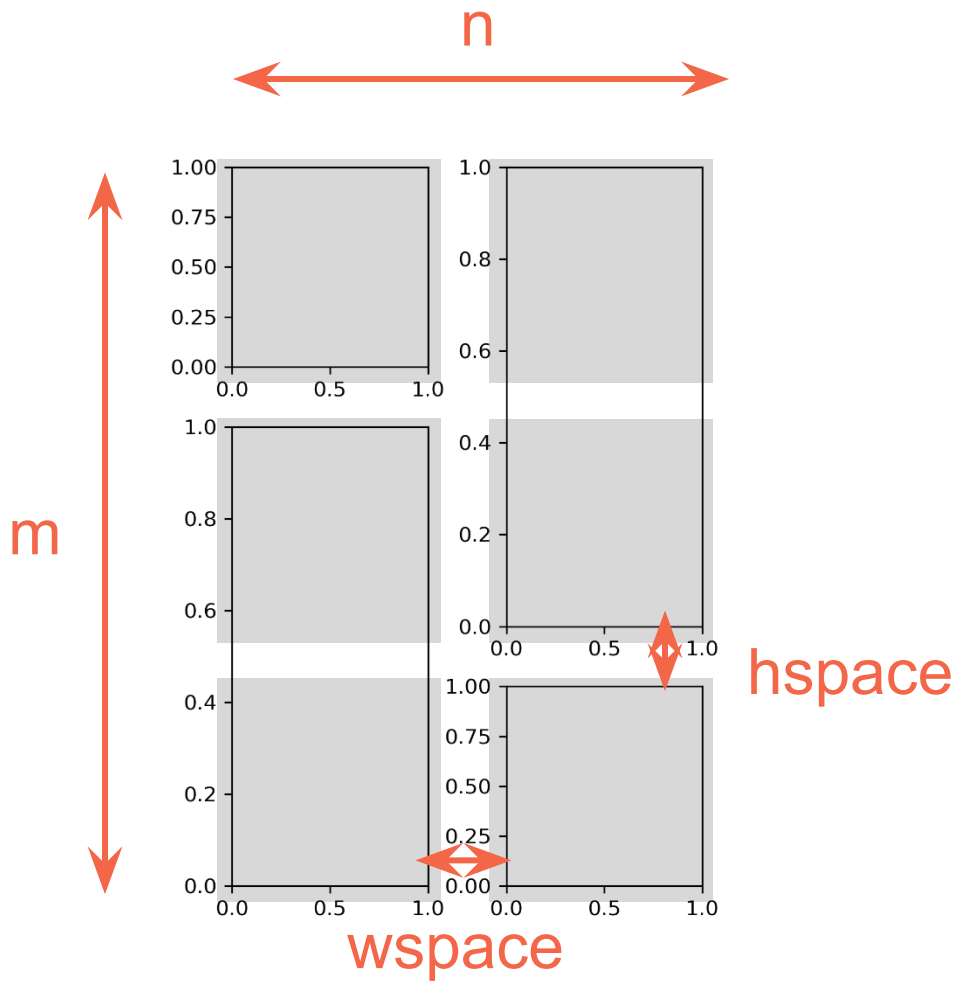


Complicated Arrangements

```
grid = plt.GridSpec(3, 2, wspace=0.4,
                    hspace=0.3)
```

The `plt.GridSpec()` object does not create a plot by itself.

```
plt.subplot(grid[0, 0]); ...
plt.subplot(grid[1:, 0]); ...
plt.subplot(grid[:2, 1]); ...
plt.subplot(grid[2, 1]); ...
```



Further Resources

Matplotlib's online documentation can be a helpful reference.

Also check out the Matplotlib's gallery, you can visually inspect and learn about a wide range of different plotting styles and visualization techniques.