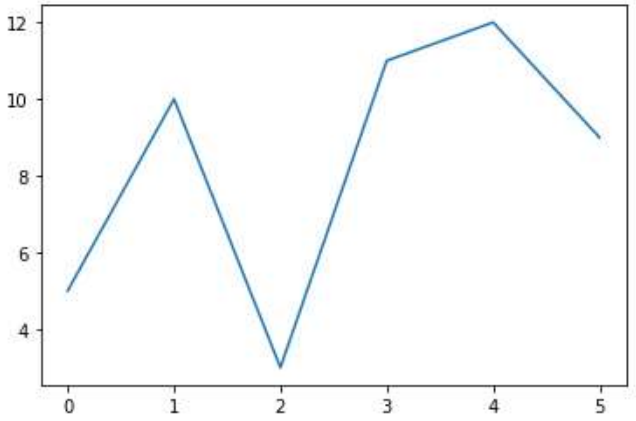
Python library - matplotlib (*)

Matplotlib is a graph plotting library for creating static, animated, and interactive visualizations in Python. Matplotlib.pyplot makes matplotlib work like MATLAB.

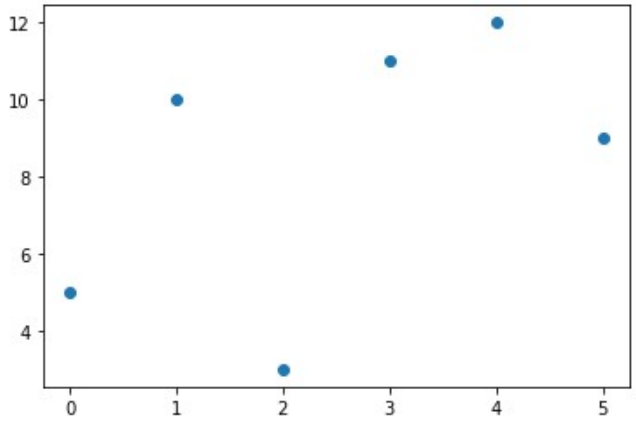
```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
```

```
In [2]: xpoints = np.array([0, 1, 2, 3, 4, 5])
        ypoints = np.array([5, 10, 3, 11, 12, 9])
```

```
In [3]: # Draw a line connecting xpoints,ypoints
        plt.plot(xpoints, ypoints);
```




```
In [4]: # Draw markers only
        plt.plot(xpoints, ypoints, 'o');
```



(*) https://matplotlib.org/stable/tutorials/index.html
https://www.w3schools.com/python/matplotlib_pyplot.asp
https://www.youtube.com/playlist?list=PL-osiE80TeTvipOqomVEeZ1HRrcEvtZB_

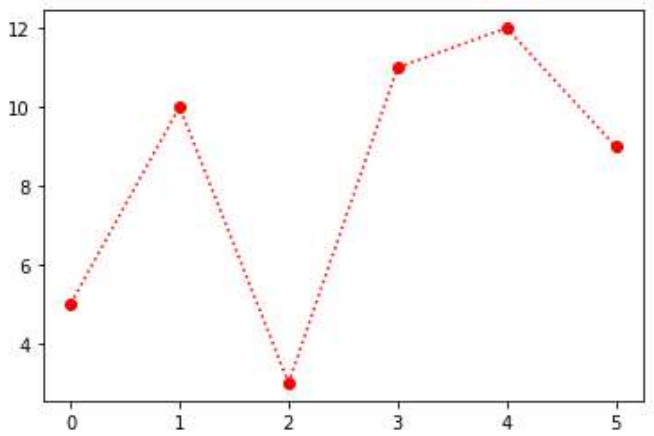
Python library - matplotlib - plot()

The **plot()** method is used to display points on a graph. By default, the points are connected by a line from point to point.

plot(y-array)	the default x-array has the same length as y-array [0, 1, …]
plot(x-array, y-array)	x-axis array of points and y-axis array of points
plot(x-array, y-array, fmt)	fmt = '[marker][line][color]' a format string setting marker, line and color

marker: '.' point; 'o' circle; 'x' cross
line: '-' solid; '--' dashed; '-.' dash-dot; ':' dotted
color: 'b' blue; 'g' green; 'r' red; 'c' cyan; 'k' black; 'w' white

```
In [5]: # Draw with a marker'o', a dotted line ':' using a red 'r' color
        plt.plot(xpoints, ypoints, 'o:r');
```



[ ] meaning the parameter is optional unless used with lists

Python library - matplotlib - plot()

The generic signatures of the **plot()** method are the following:

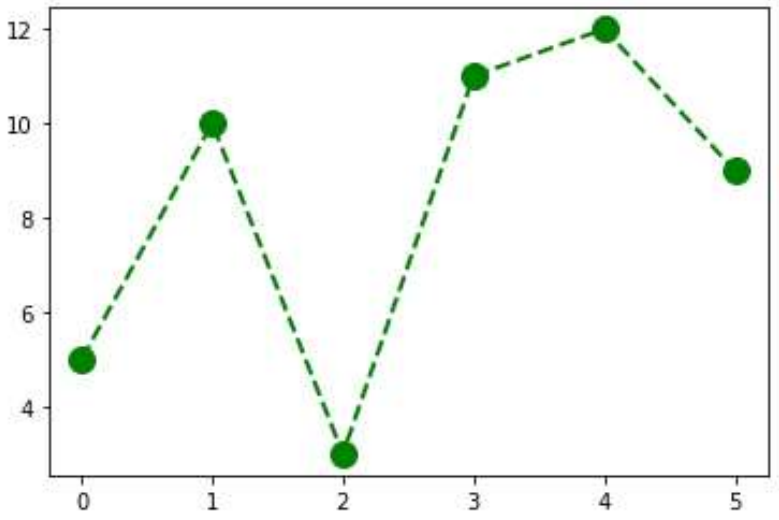
plot([x], y, [fmt], *, data=None, **kwargs)

plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

The kwargs arguments specify optional properties such as:

color or **c**	line color ('red' or '#FF0000')
label	line label (str)
linestyle	line style ('-', '--', '-.', ':')
linewidth	line width (float)
marker	point marker ('.', 'o', 'x')
markersize	marker size (float)

```
In [6]: # Draw a green dashed line with circle marks
        plt.plot(xpoints, ypoints, color='green', marker='o',
                 linestyle='dashed', linewidth=2, markersize=12);
```

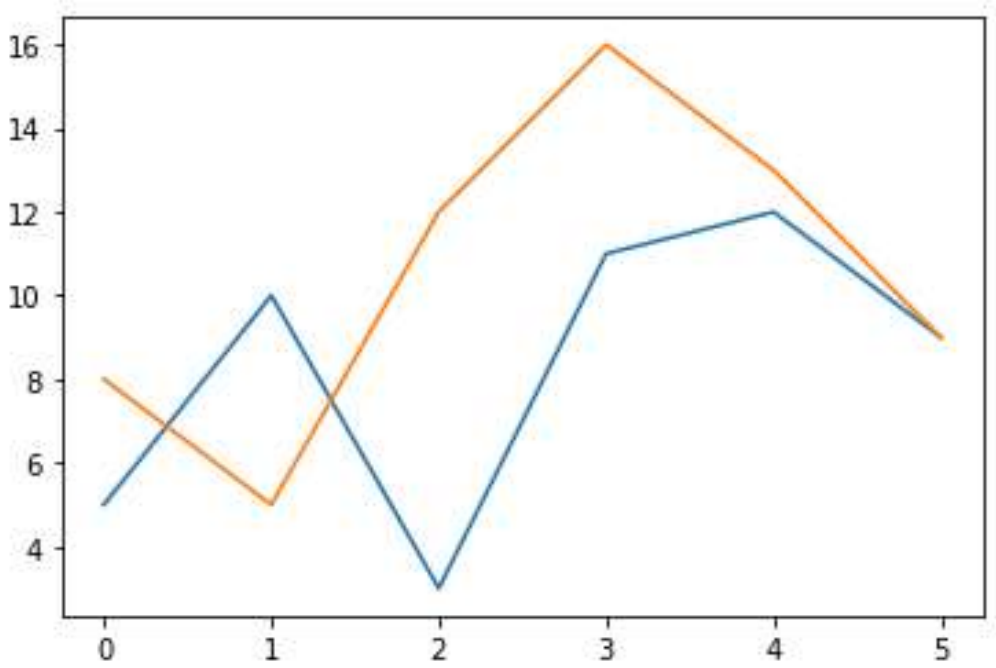


Python library - matplotlib - plot()

Draw multiple lines

```
In [7]: xpoints = np.array([0, 1, 2, 3, 4, 5])
        ypoints1 = np.array([5, 10, 3, 11, 12, 9])
        ypoints2 = np.array([8, 5, 12, 16, 13, 9])
```

```
In [8]: # Draw two lines connecting ypoints1
        # and connecting ypoints2
        plt.plot(xpoints, ypoints1)
        plt.plot(xpoints, ypoints2)
        # alternative way to plot the two lines
        plt.plot(xpoints, ypoints1, ypoints2);
```



Python library - matplotlib - plot()

Labels, title, grid and legend

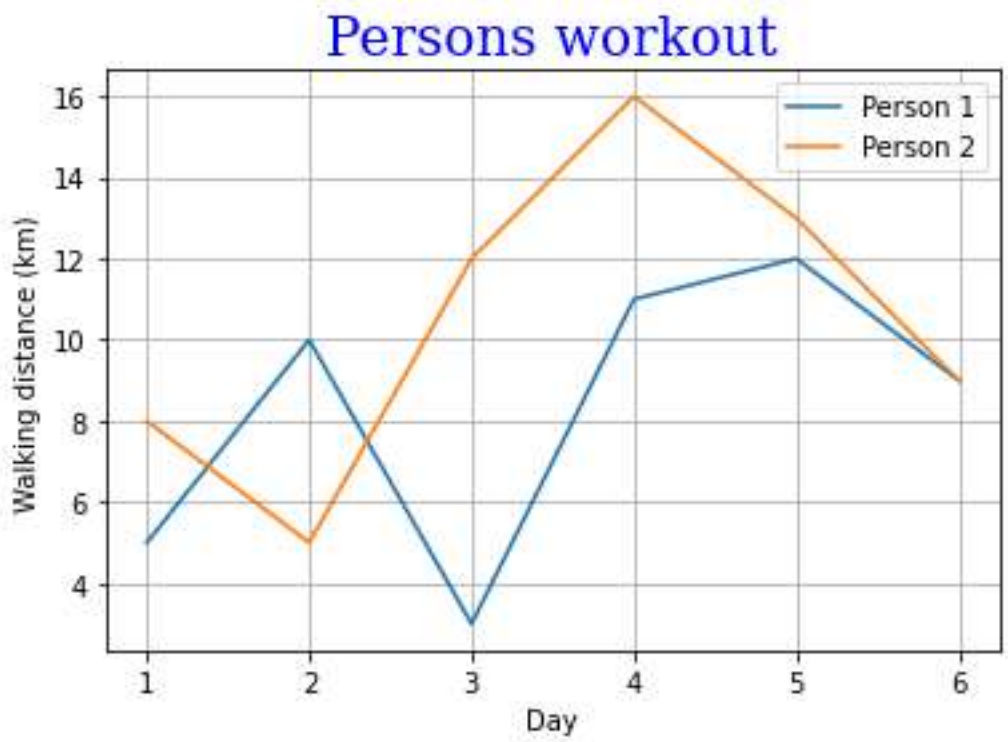
```
In [9]: plt.plot(xpoints, ypoints1, label = 'Person 1')
        plt.plot(xpoints, ypoints2, label = 'Person 2')

        # x axis label
        plt.xlabel('Day')
        # y axis label
        plt.ylabel('Walking distance (km)')

        # title with font characteristics
        font1 = {'family':'serif','color':'blue','size':20}
        plt.title('Persons workout', fontdict = font1)

        # draw the grid
        plt.grid()

        # draw the legend
        plt.legend();
```



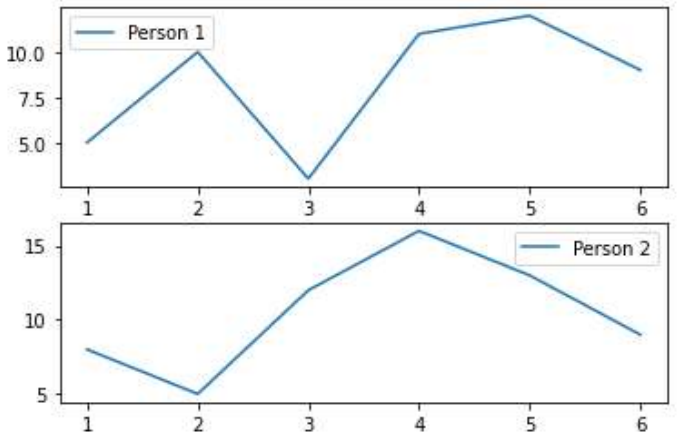
Python library - matplotlib - subplot()

subplot(rows, columns, index)

The **subplot()** method is used to display multiple plots in one figure. The figure layout is organized in rows and columns. The index defines the plot position.

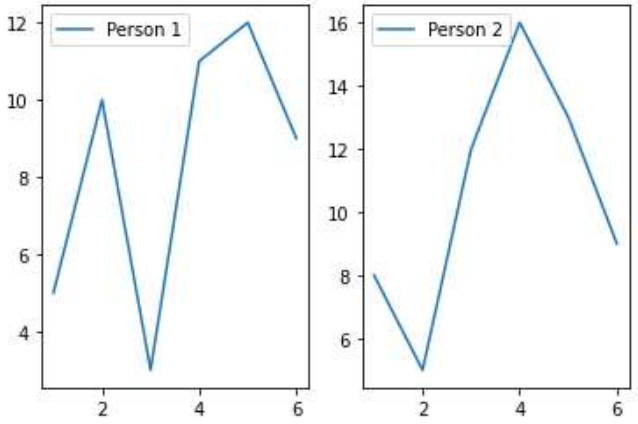
```
In [10]: # subplot for person 1
         plt.subplot(2, 1, 1)
         plt.plot(xpoints, ypoints1, label = 'Person 1')
         plt.legend()

         # subplot for person 2
         plt.subplot(2, 1, 2)
         plt.plot(xpoints, ypoints2, label = 'Person 2')
         plt.legend();
```




```
In [11]: # subplot for person 1
         plt.subplot(1, 2, 1)
         plt.plot(xpoints, ypoints1, label = 'Person 1')
         plt.legend()

         # subplot for person 2
         plt.subplot(1, 2, 2)
         plt.plot(xpoints, ypoints2, label = 'Person 2')
         plt.legend();
```

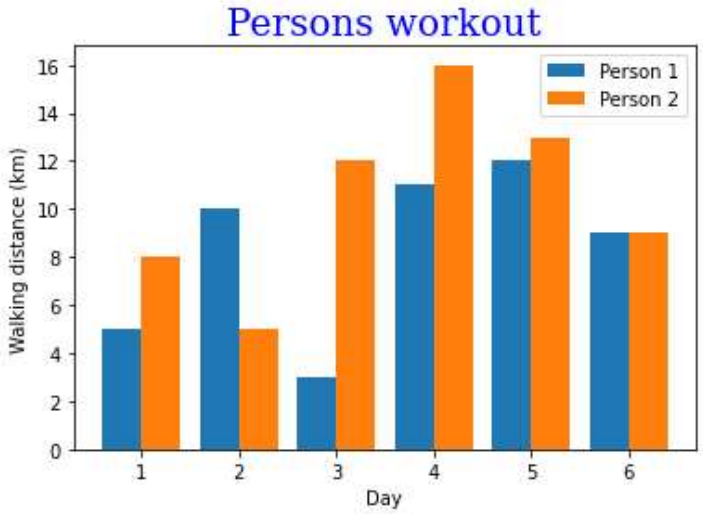


Python library - matplotlib - Bar charts

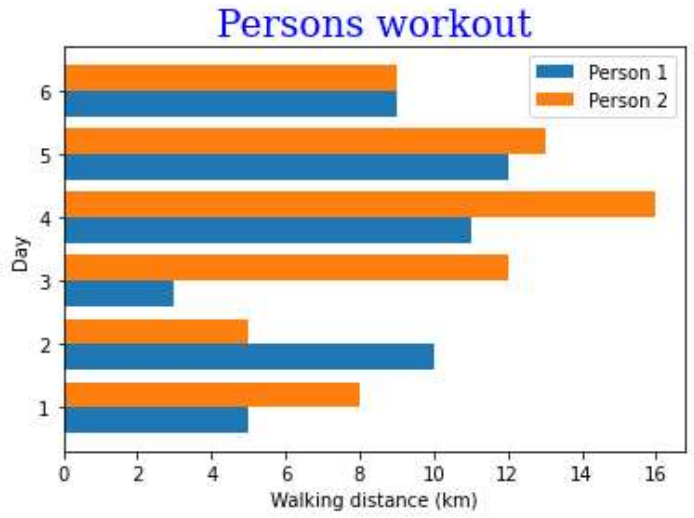
Bar chart and horizontal bars chart

```
In [12]: xpoints = np.array([1, 2, 3, 4, 5, 6])
         ypoints1 = np.array([5, 10, 3, 11, 12, 9])
         ypoints2 = np.array([8, 5, 12, 16, 13, 9])
```

```
In [13]: x_index = np.arange(len(xpoints))
         plt.bar(x_index - 0.2, ypoints1, width = 0.4, label = 'Person 1')
         plt.bar(x_index + 0.2, ypoints2, width = 0.4, label = 'Person 2')
         plt.xticks(ticks=x_index, labels = xpoints)
         plt.xlabel('Day')
         plt.ylabel('Walking distance (km)')
         font1 = {'family':'serif','color':'blue','size':20}
         plt.title('Persons workout', fontdict = font1)
         plt.legend();
```

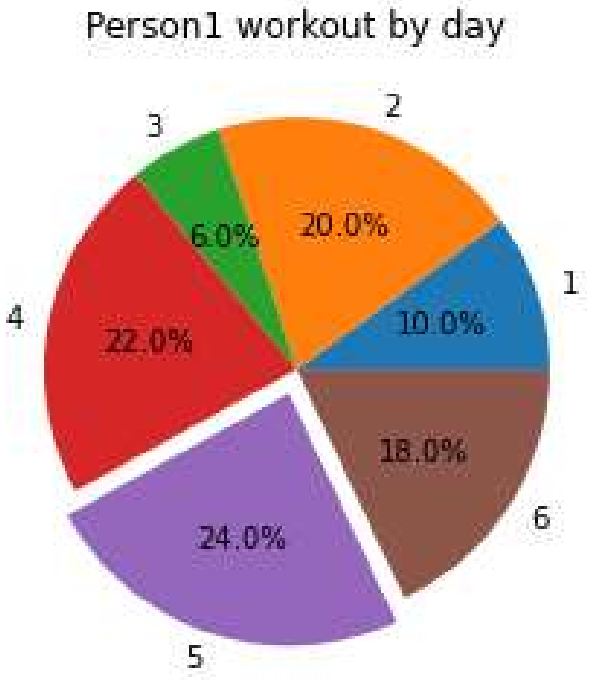



```
In [14]: x_index = np.arange(len(xpoints))
         plt.barh(x_index - 0.2, ypoints1, height = 0.4, label = 'Person 1')
         plt.barh(x_index + 0.2, ypoints2, height = 0.4, label = 'Person 2')
         plt.yticks(ticks=x_index, labels = xpoints)
         plt.ylabel('Day')
         plt.xlabel('Walking distance (km)')
         font1 = {'family':'serif','color':'blue','size':20}
         plt.title('Persons workout', fontdict = font1)
         plt.legend();
```



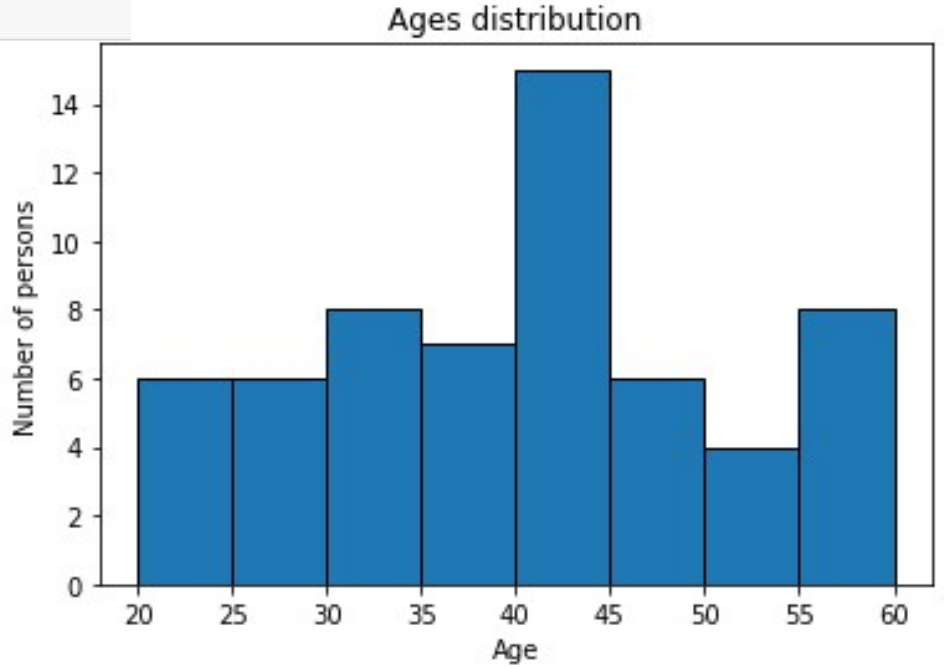
Python library - matplotlib - Pie charts

```
In [15]: xpoints = np.array([1, 2, 3, 4, 5, 6])
         ypoints = np.array([5, 10, 3, 11, 12, 9])
         explode = [0, 0, 0, 0, 0.1, 0]
         plt.pie(ypoints, labels = xpoints, explode = explode, autopct = '%1.1f%%')
         plt.title('Person1 workout by day');
```



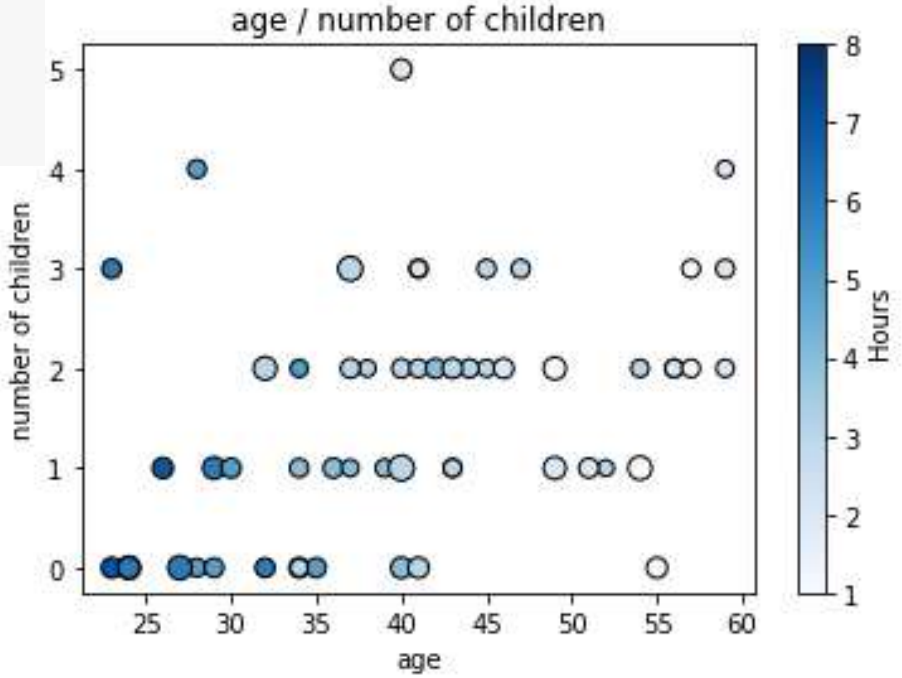
Python library - matplotlib - histograms

```
In [16]: import pandas as pd
data = pd.read_csv('gym.csv', delimiter = ';')
bins = [20, 25, 30, 35, 40, 45, 50, 55, 60]
idades = data['age']
plt.hist(idades, bins=bins, edgecolor = 'black')
plt.xlabel('Age')
plt.ylabel('Number of persons')
plt.title('Ages distribution')
```



Python library - matplotlib – scatter plots

```
In [17]: import pandas as pd
         data = pd.read_csv('gym.csv', delimiter = ';')
         ages = data['age']
         children = data['children']
         colors = data['hours']
         sizes = data['weight']
         plt.scatter(ages, children, s=sizes,
                     c=colors, cmap = 'Blues', edgecolor = 'black')
         cbar = plt.colorbar()
         cbar.set_label('Hours')
         plt.xlabel('age')
         plt.ylabel('number of children')
         plt.title('age / number of children');
```



Python library – matplotlib - animation (*)

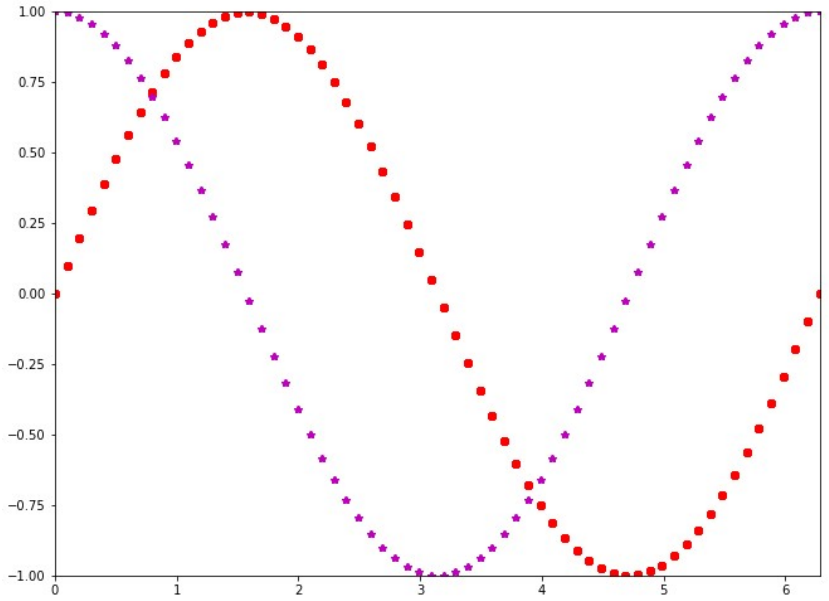
```
In [18]: import numpy as np
         import matplotlib.pyplot as plt
         from matplotlib.animation import FuncAnimation, PillowWriter
         %matplotlib notebook

         fig, ax = plt.subplots()
         x, ysin, ycos = [], [], []
         ln1, = plt.plot([], [], 'ro')
         ln2, = plt.plot([], [], 'm*')

         def init():
             ax.set_xlim(0, 2*np.pi)
             ax.set_ylim(-1, 1)

         def update(i):
             x.append(i)
             ysin.append(np.sin(i))
             ycos.append(np.cos(i))
             ln1.set_data(x, ysin)
             ln2.set_data(x, ycos)

         ani = FuncAnimation(fig, update, np.linspace(0, 2*np.pi, 64), init_func=init)
         # writer = PillowWriter(fps=25)
         # ani.save("demo_sine.gif", writer=writer)
         plt.show()
```



(*) https://matplotlib.org/stable/api/animation_api.html
https://www.c-sharpcorner.com/article/create-animated-gif-using-python-matplotlib/