SCATTER PLOTS

Another commonly used plot type is the simple scatter plot, a close cousin of the line plot. Instead of points being joined by line segments, here the points are represented individually with a dot, circle, or other shape.

Syntax

plt.plot(x, y, 'type of symbol ', color);

Example

plt.plot(x, y, 'o', color='black');

- The third argument in the function call is a character that represents the type of symbol used for the plotting. Just as you can specify options such as '-' and '--' to control the line style, the marker style has its own set of short string codes.

Example

- Various symbols used to specify ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']
- Short hand assignment of line, symbol and color also allowed.

  plt.plot(x, y, '-ok');
- Additional arguments in plt.plot()

We can specify some other parameters related with scatter plot which makes it more attractive. They are color, marker size, linewidth, marker face color, marker edge color, marker edge width, etc

Example

```
plt.plot(x, y, '-p', color='gray',
markersize=15, linewidth=4,
markerfacecolor='white',
markeredgecolor='gray',
markeredgewidth=2)
plt.ylim(-1.2, 1.2);
```

Scatter Plots with plt.scatter

- A second, more powerful method of creating scatter plots is the plt.scatter function, which can be used very similarly to the plt.plot function
  plt.scatter(x, y, marker='o');
- The primary difference of plt.scatter from plt.plot is that it can be used to create scatter plots where the properties of each individual point (size, face color, edge color, etc.) can be individually controlled or mapped to data.
- Notice that the color argument is automatically mapped to a color scale (shown here by the colorbar() command), and the size argument is given in pixels.
- Cmap – color map used in scatter plot gives different color combinations.

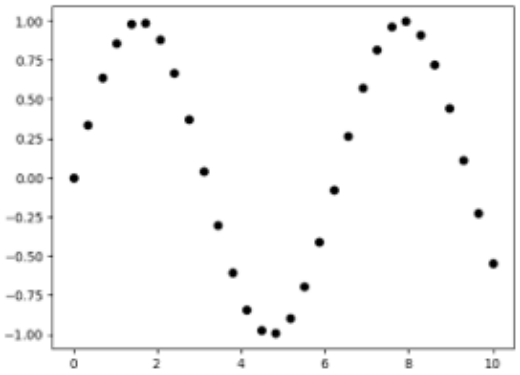
```
Perceptually Uniform Sequential
['viridis', 'plasma', 'inferno', 'magma']
Sequential
['Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Reds', 'YlOrBr', 'YlOrRd',
'OrRd', 'PuRd', 'RdPu', 'BuPu', 'GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn',
'YlGn']
Sequential (2)
['binary', 'gist_yarg', 'gist_gray', 'gray', 'bone', 'pink', 'spring', 'summer',
'autumn', 'winter', 'cool', 'Wistia', 'hot', 'afmhot', 'gist_heat', 'copper']
```

```
Diverging
['PiYG', 'PRGn', 'BrBG', 'PuOr', 'RdGy', 'RdBu', 'RdYlBu', 'RdYlGn', 'Spectral',
'coolwarm', 'bwr', 'seismic']
Qualitative
['Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'Set1', 'Set2', 'Set3',
'tab10', 'tab20', 'tab20b', 'tab20c']
Miscellaneous
['flag', 'prism', 'ocean', 'gist_earth', 'terrain', 'gist_stern', 'gnuplot',
'gnuplot2', 'CMRmap', 'cubehelix', 'brg', 'hsv', 'gist_rainbow', 'rainbow',
'jet', 'nipy_spectral', 'gist_ncar']
```

Example programs.

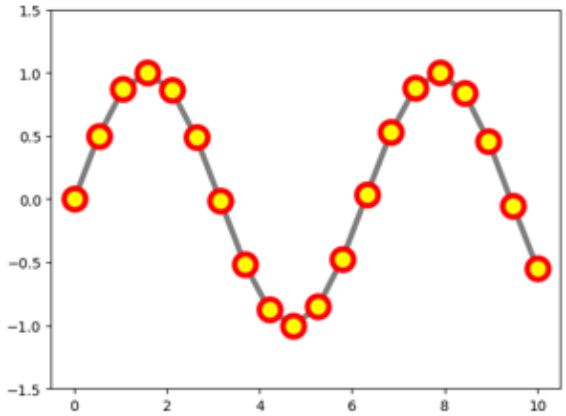
Simple scatter plot.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.plot(x, y, 'o', color='black');
```



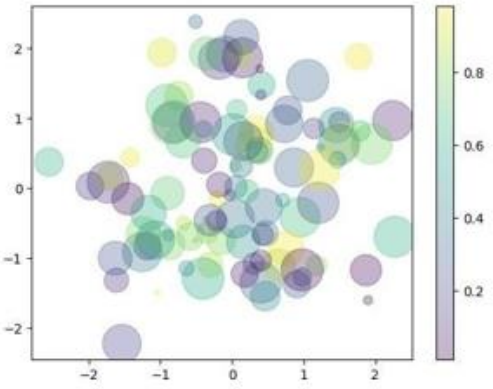
Scatter plot with edge color, face color, size, and width of marker. (Scatter plot with line)

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 20)
y = np.sin(x)
plt.plot(x, y, '-o', color='gray',
markersize=15, linewidth=4,
markerfacecolor='yellow',
markeredgecolor='red',
markeredgewidth=4)
plt.ylim(-1.5, 1.5);
```



Scatter plot with random colors, size and transparency

```
import numpy as np
import matplotlib.pyplot as plt
rng = np.random.RandomState(0)
x = rng.randn(100)
y = rng.randn(100)
colors = rng.rand(100)
sizes = 1000 * rng.rand(100)
plt.scatter(x, y, c=colors, s=sizes, alpha=0.3, map='viridis')
plt.colorbar()
```



VISUALIZING ERRORS

For any scientific measurement, accurate accounting for errors is nearly as important, if not more important, than accurate reporting of the number itself. For example, imagine that I am using some astrophysical observations to estimate the Hubble Constant, the local measurement of the expansion rate of the Universe.

In visualization of data and results, showing these errors effectively can make a plot convey much more complete information.

Types of errors

- Basic Errorbars
- Continuous Errors

Basic Errorbars

A basic errorbar can be created with a single Matplotlib function call.

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
x = np.linspace(0, 10, 50)
dy = 0.8
y = np.sin(x) + dy * np.random.randn(50)
plt.errorbar(x, y, yerr=dy, fmt='.k');
```