Matplotlib

Object Oriented Programming and Scripting in Python

Basic Info

- Extension that permits to handle 2D charts.
- Uses numpy for handling high dimensinal data
- Website: http://www.matplotlib.org
- Plotting functions incuded in the module **`pyplot`**

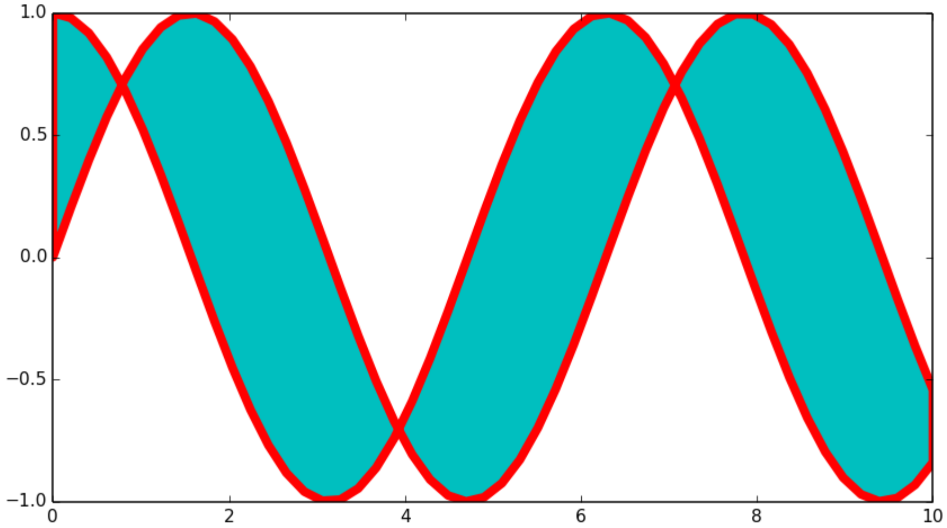
Import :

```
>>> from matplotlib import pyplot
>>> import matplotlib.pyplot as pl
```

Plotting Multiplo

- Esempio:

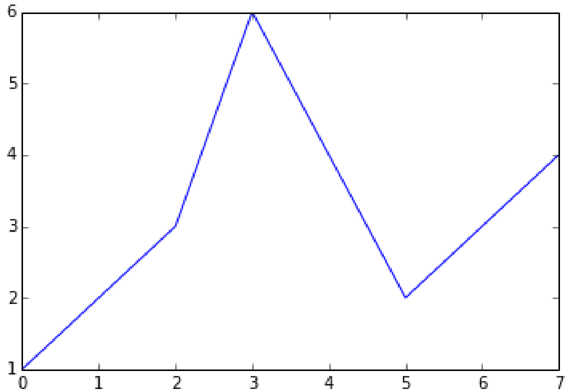
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> z = np.cos(x)
>>> pl.fill_between(x,y,z,color='r',
facecolor='c', linewidth=5)
>>> pl.show()
```



Plotting: Basic Functions

- `plot(*args, **kwargs)`
- For showing a plot: `show()`
  - #Example

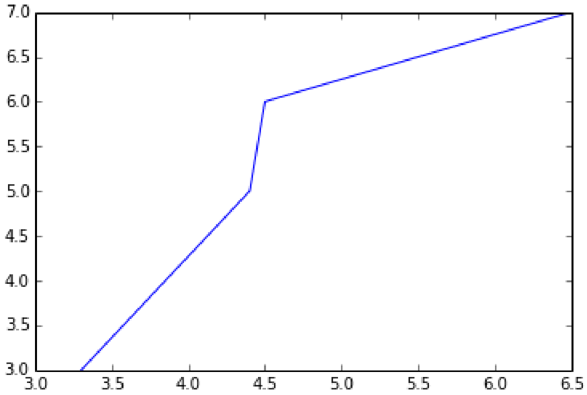
  -
    ```
    >>> import matplotlib.pyplot as pl
    >>> pl.plot([1,2,3,6,4,2,3,4])
    >>> pl.show()
    ```



Plotting: Basic Functions

- For 2D charts, the main parameters are the domain (x axis) and the codomain (y axis): **`plot(x,y, ...)`**
- #Examples

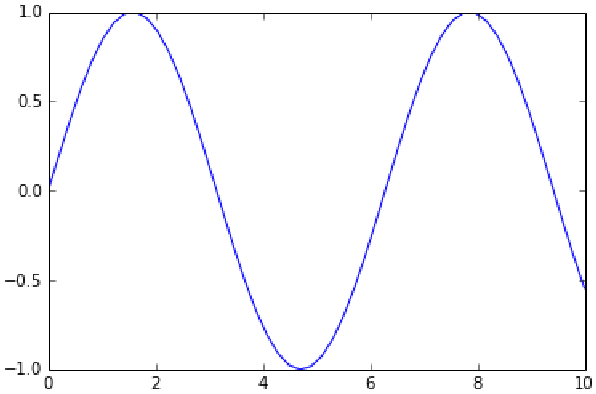
```
>>> x = [3.3,4.4,4.5,6.5]
>>> y = [3.,5.,6.,7.]
>>> pl.plot(x,y)
>>> pl.show()
```



Plotting: Basic Functions

- Functions plotting : use of numpy
- #Example :

```
>>> import numpy as np
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y)
>>> pl.show()
```



Plotting: Line Style

- Optional parameter `linestyle` (or `ls`) - default : lsolid line:
- `plot(… , linestyle='…', ...)`
- Possible values:

  `'-'` or `'solid'` → Solid line

  `'--'` or `'dashed'` → Dashed line

  `'-.'` or `'dash_dot'` → Alternates dots and dashes

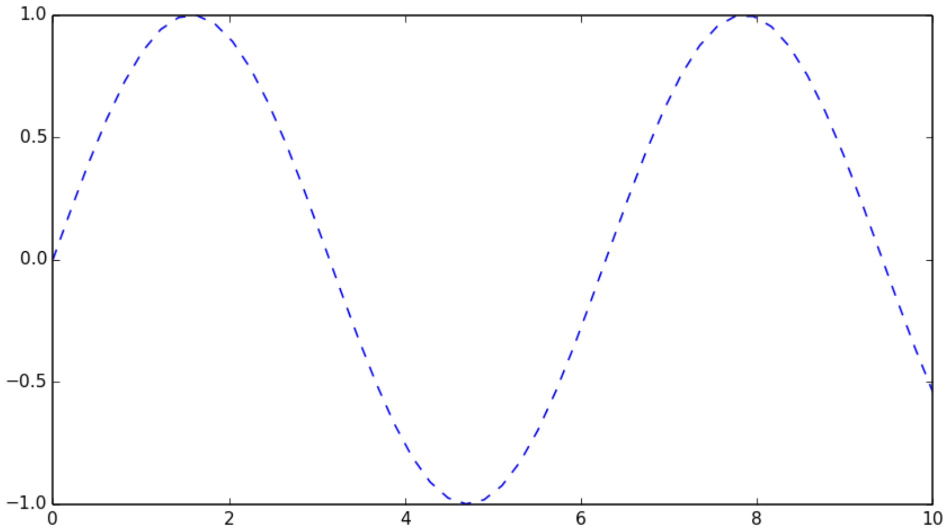
  `':'` or `'dotted'` → Dotted line

  `''` or `' '` or `'None'` → No line

Plotting: Line Style

- Example: dotted line

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y, linestyle='--')
>>> pl.show()
```



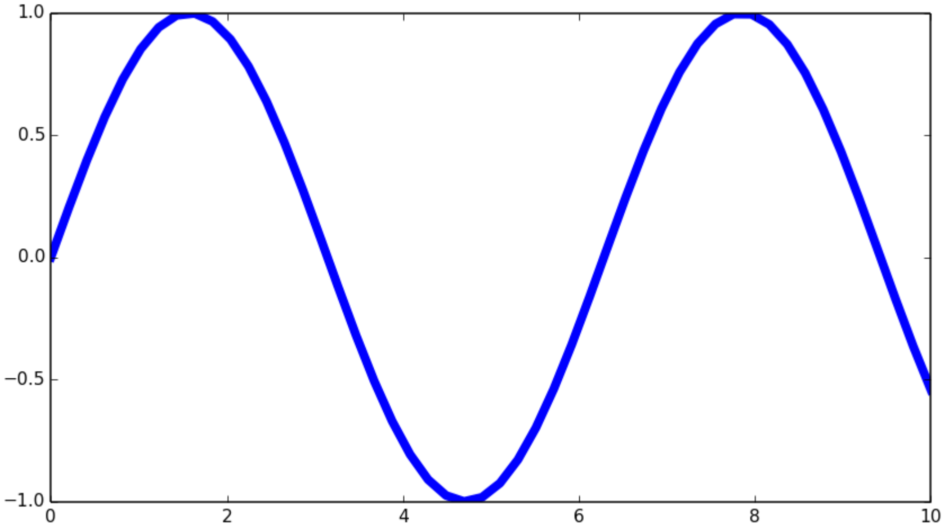
Plotting: Line Width

- Optional parameter `linewidth`:

  `plot(… , **linewidt=…**, …)`

- Example:

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y, linewidth = 5)
>>> pl.show()
```



Plotting: Marker

- marker: style of each point (default: no marker):

  plot(… , **marker=**'…', …)

- Possible values:

  '.' → Dot   '+' → Cross

  ',' → Pixel   'x' → X

  'o' → Round   'D' → Rombo

  's' → Square   '*' → Star

  'v' '^' '<' '>' → Triangles (orientated depending of the used char)

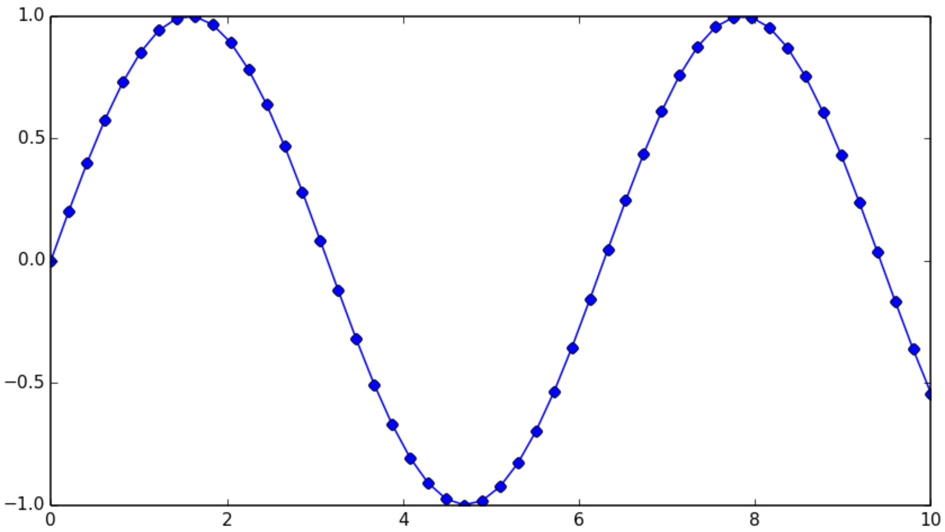
  '' or ' ' or 'None' → No marker

  ...

Plotting: Marker

- Example: rounded marker

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y, marker='o')
>>> pl.show()
```



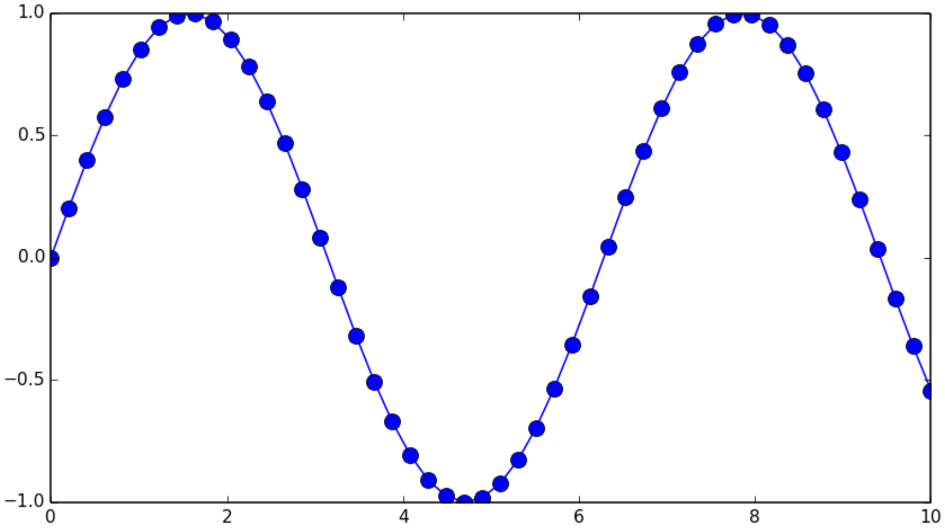
Plotting: Marker Width

- `markersize` (or `ms`):

  `plot(… , markersize=…, …)`

- Example:

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y,marker='o',markersize=9)
>>> pl.show()
```



Plotting: Line Colors

- `color` (or `c`):
- `plot(… , `**`color='…'`**`, ...)`
- Possible values:

  `'b'` or `'blue'`

  `'g'` or `'green'`

  `'r'` or `'red'`

  `'m'` or `'magenta'`

  `'k'` or `'black'`

  `'y'` or `'yellow'`

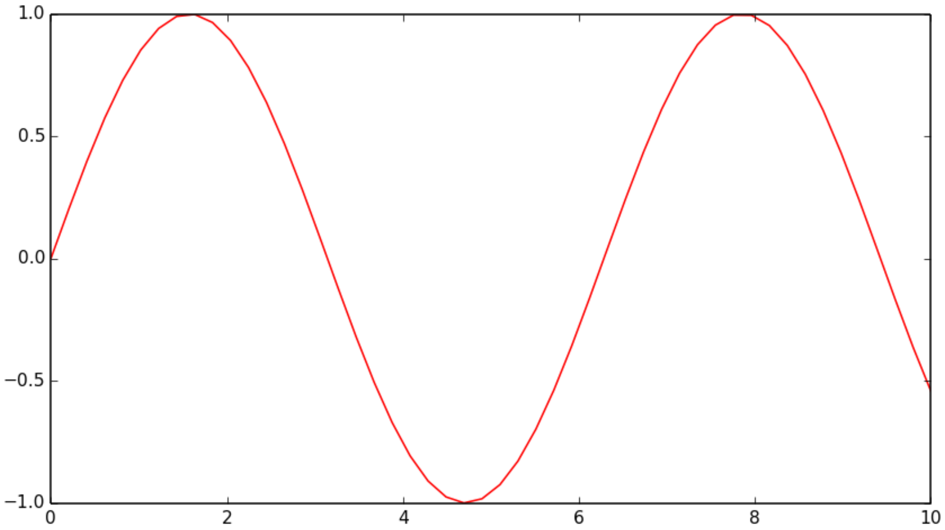
  `'c'` or `'cyan'`

  others → hexadecimal code (ex. `'FF5C4A')`

Plotting: Line Colors

- Examples:

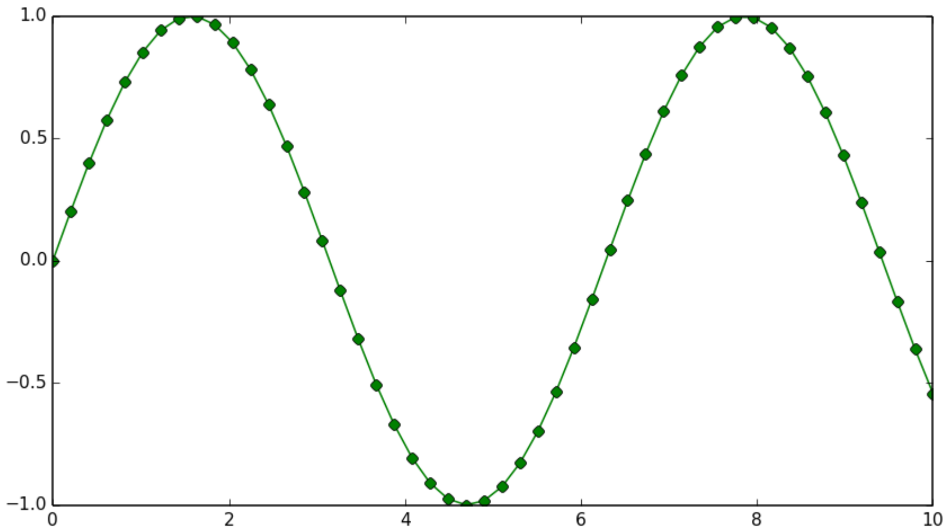
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y,color='red')
>>> pl.show()
```



Plotting: Compact expression

- String containing settings on lie width, color, and marker

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y,'go-')
>>> pl.show()
```



Labels and Legend

- Label :

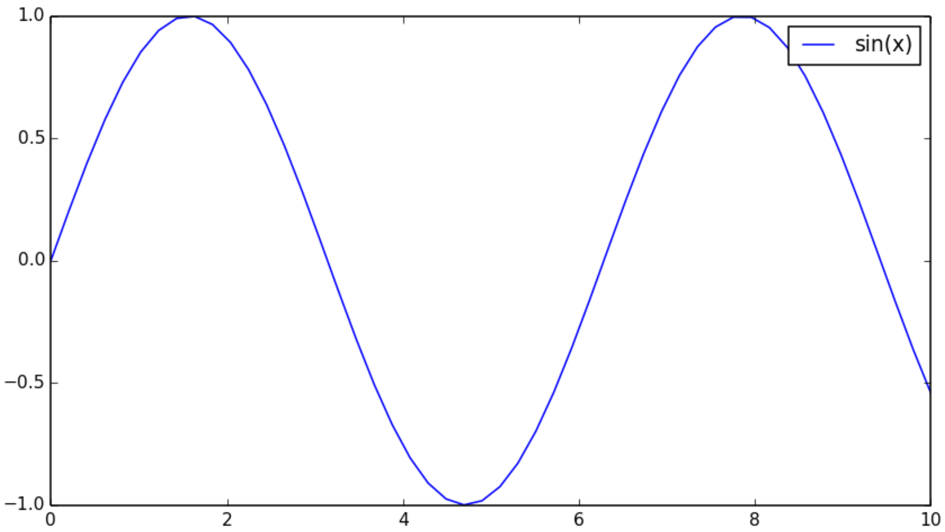
  plot(… , **label='labelstring'**, …)
- legend: shows the legend of the chart:

  pl.legend(*args)

Labels and Legend

- Example :

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y,label='sin(x)')
>>> pl.legend()
>>> pl.show()
```



Labels and Legend

- Legend location: loc

```
pl.legend(loc = '…')
```

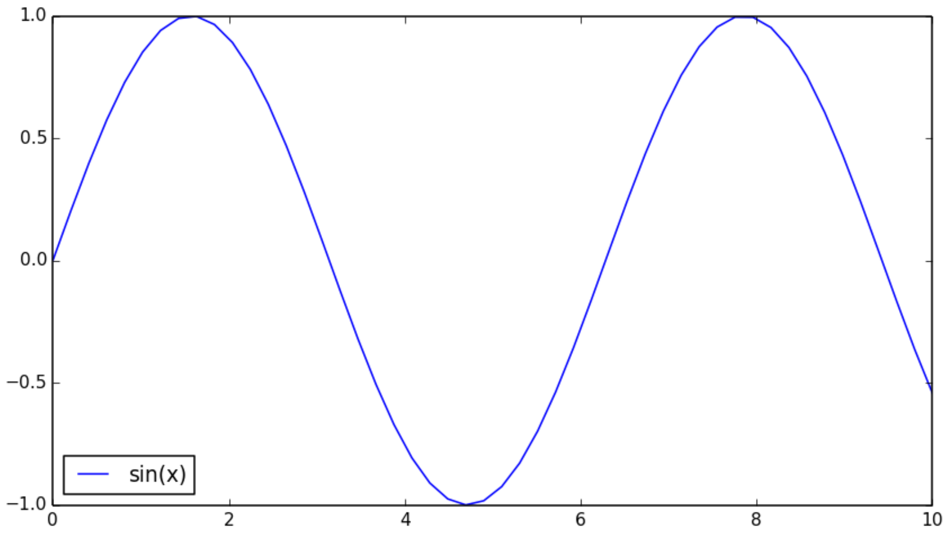
- Possible values:

```
'best'
'upper'
'right'
'left'
 ...
```

Labels and Legend

- Example

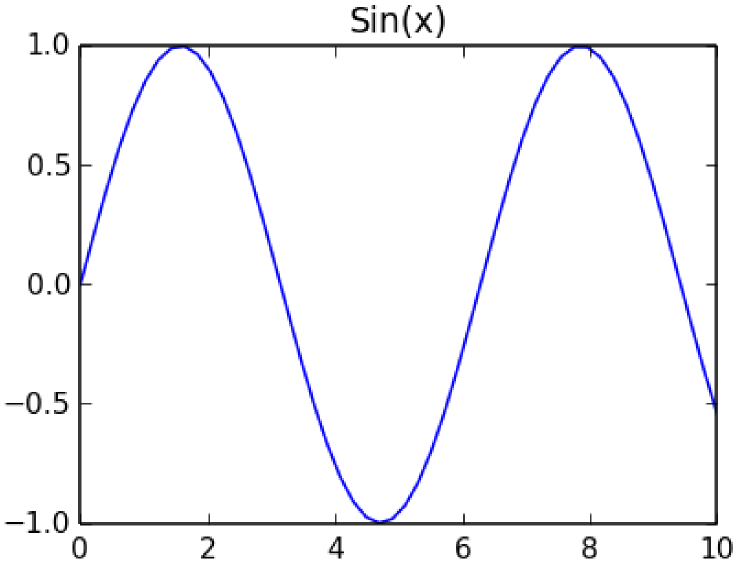
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y,label='sin(x)')
>>> pl.legend(loc='best')
>>> pl.show()
```



Title

- `title`: takes a string as parameters, the string being the title of the chart

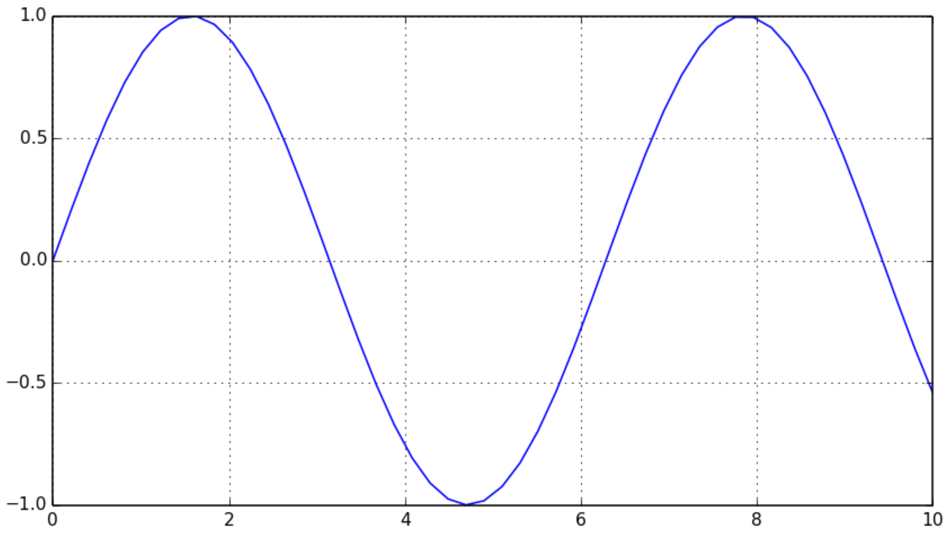
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y)
>>> pl.title('Sin(x)')
>>> pl.show()
```



Grid

- **grid(True/False)**

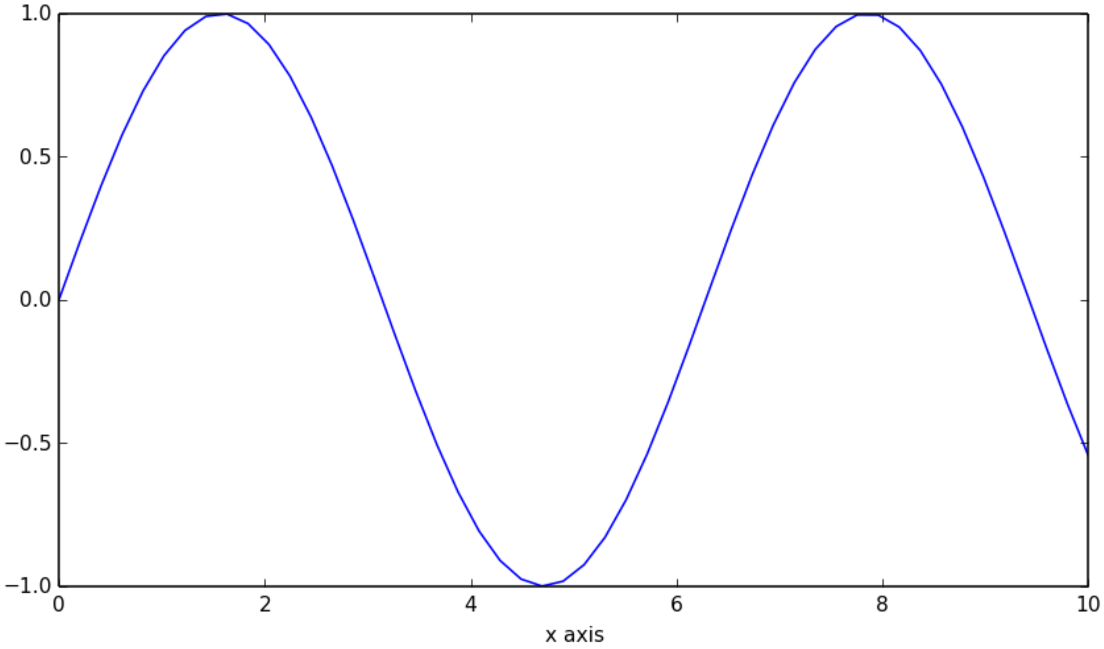
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.plot(x,y)
>>> pl.grid(True)
>>> pl.show()
```



Axes

- Axes labels : `xlabel('XAxisName'); ylabel('YAxisName')`

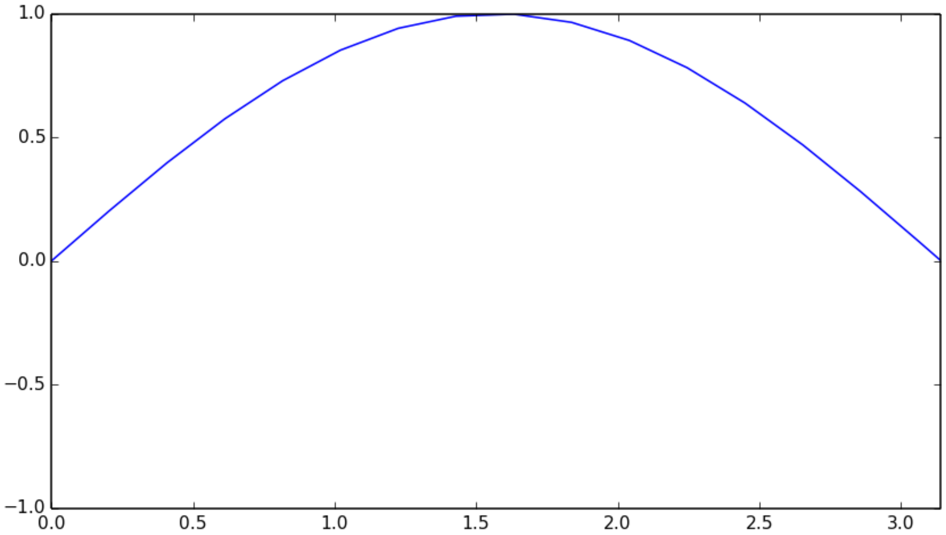
```
>>> pl.plot(x,y)
>>> pl.xlabel('x axis')
>>> pl.ylabel('y axis')
>>> pl.show()
```



Axis

- Axis limits: `xlim(limInf, limSup); ylim(limInf, limSup)`

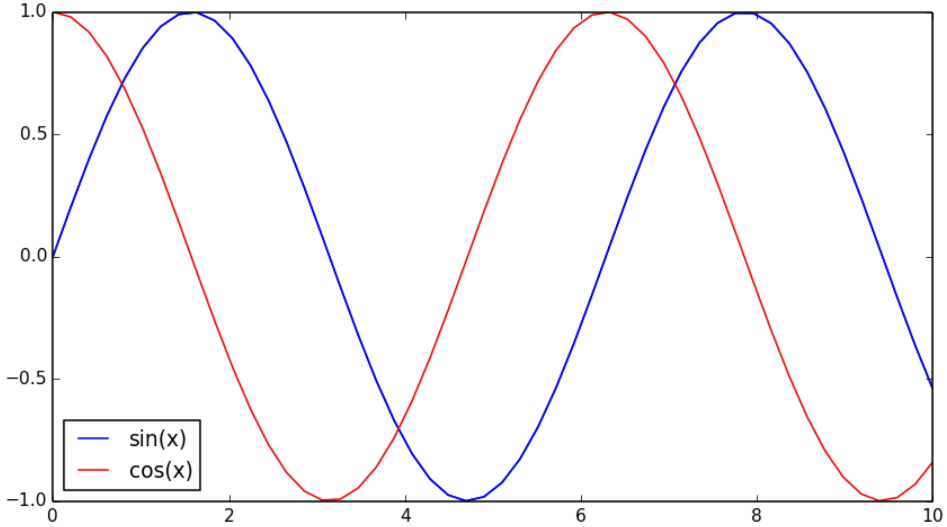
```
>>> pl.plot(x,y)
>>> pl.xlim(0, pi)
>>> pl.ylim(-1, 1)
>>> pl.show()
```



Multiple Plotting

- More charts in the same window:

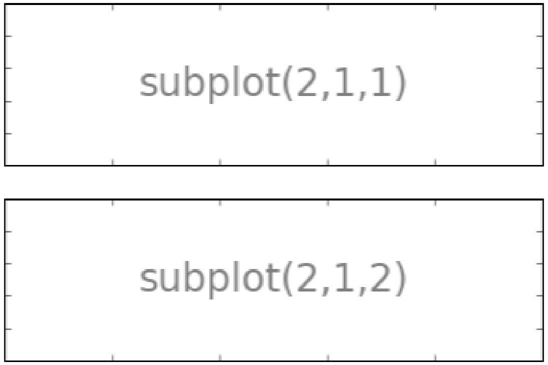
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> z = np.cos(x)
>>> pl.plot(x,y,label='sin(x)',color='b')
>>> pl.plot(x,z,label='cos(x)',color='r')
>>> pl.legend(loc='best')
>>> pl.show()
```



Multiple Plotting

- Subplot: divides the window in subcharts.

  `subplot(r, c, n)` → *r* number of rows, *c* number of columns, *n* indicates in which subchart the plotting should be made, for example:

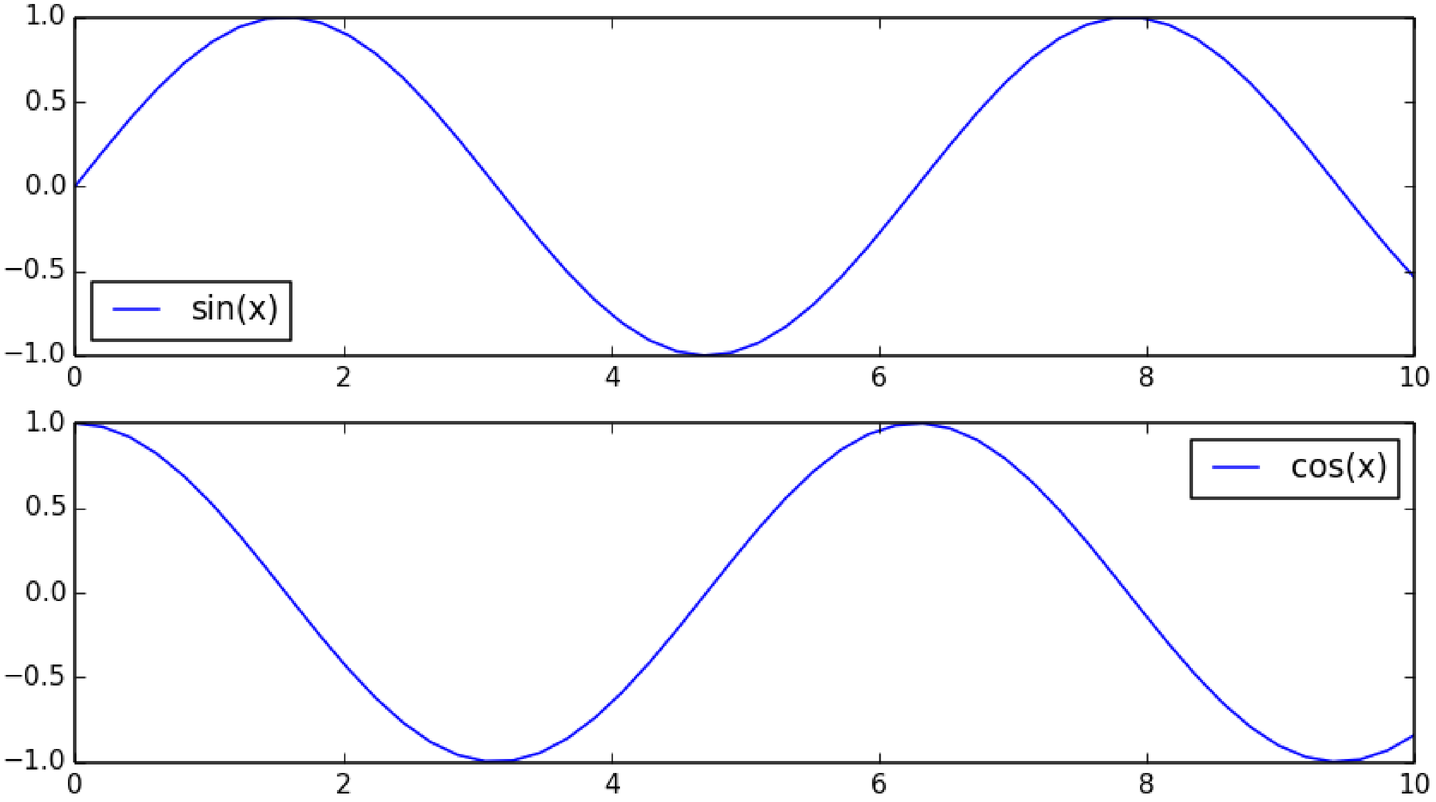


Multiple Plotting

- Example:

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> z = np.cos(x)
>>> pl.subplot(2, 1, 1)
>>> pl.plot(x,y,label='sin(x)')
>>> pl.legend(loc='best')
>>> pl.subplot(2, 1, 2)
>>> pl.plot(x,z,label='cos(x)')
>>> pl.legend(loc='best')
>>> pl.show()
```

Multiple Plotting



Area

- Fill an area ander a curve:

```
fill_between(x,y,z=0, **kwargs)
```

- If z is not given, the area will be enclosed by the curve and the x axes.

```
>>> pl.fill_between(x,y)
```

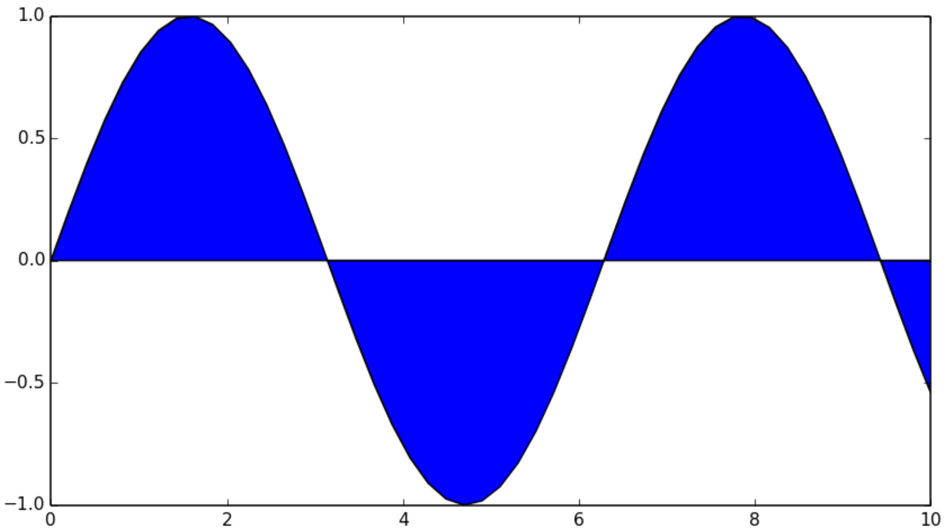
- If z is given, the area is enclosed by the curve y and z.

```
>>> pl.fill_between(x,y, z)
```

Area

- Example:

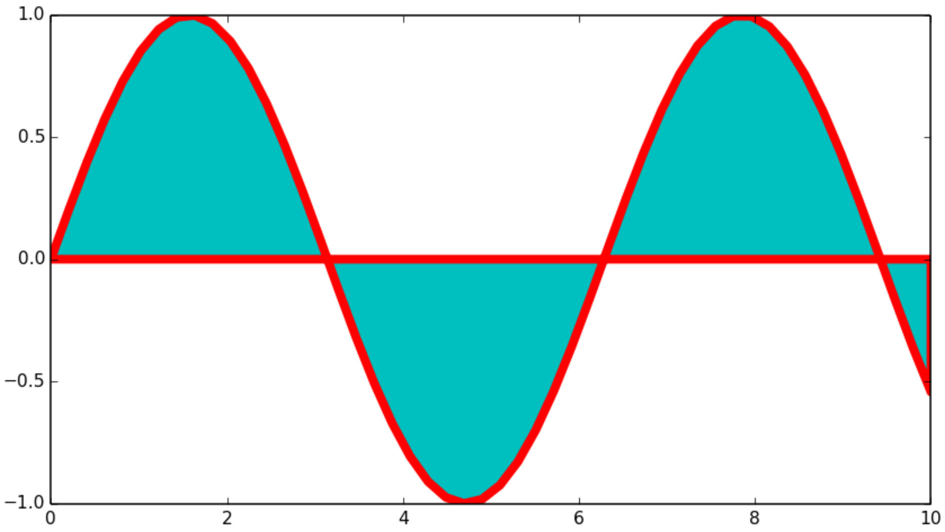
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.fill_between(x,y,label='sin(x))
>>> pl.show()
```



Area

- Parameters: facecolor (area color) and color (line color):

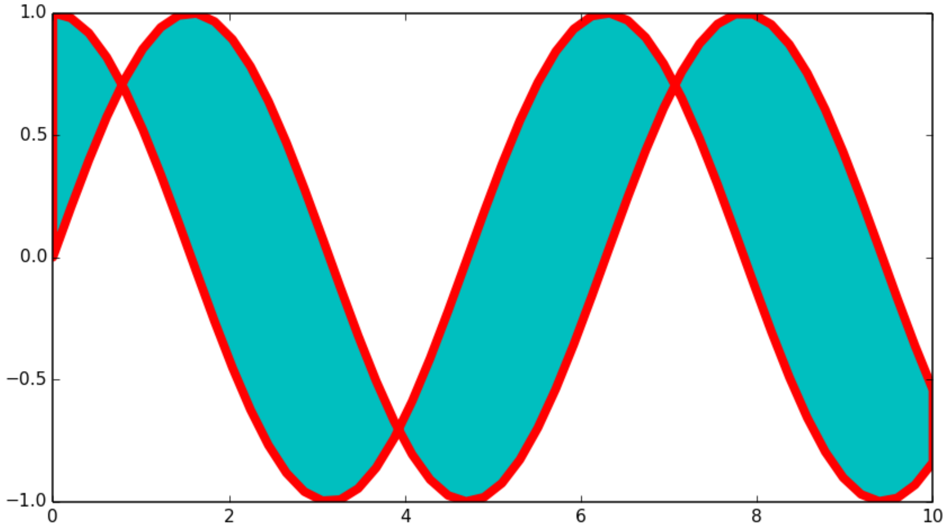
```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> pl.fill_between(x,y,color='r',
facecolor='c', linewidth=5)
>>> pl.show()
```



Area

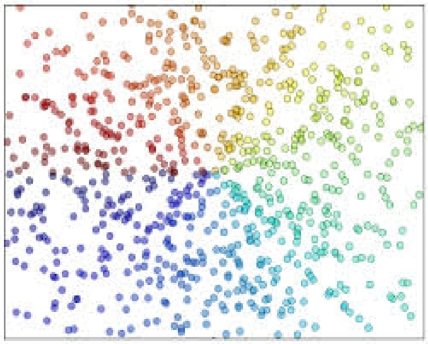
- Example:

```
>>> x = np.linspace(0.,10.,50)
>>> y = np.sin(x)
>>> z = np.cos(x)
>>> pl.fill_between(x,y,z,color='r',
facecolor='c', linewidth=5)
>>> pl.show()
```

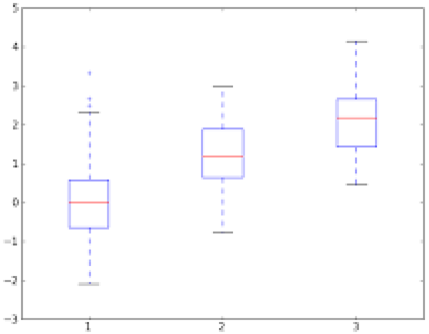


Other Plotting Functions

- `scatter(x,y,*args, **kwargs)`: makes a scatter plot of x versus y.

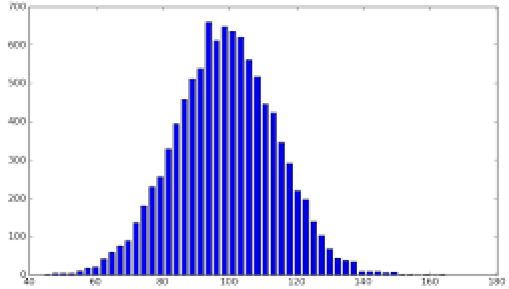


- `boxplot(x, *args, **kwargs):` makes a box and whisker plot for each vector of x.

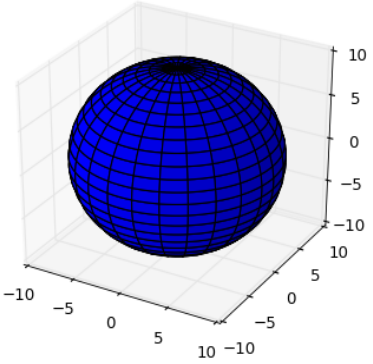


Other Plotting Functions

- hist(x,*args, **kwargs): makes a histogram plot of x.



- `plot_surface(x,y,z,*args, **kwargs)`: 3D Plotting with highlighted surface.



End !