Day 5: Plot and visualize

Just like numerical functions are contained in NumPy, plotting functions are contained in another module, caled Matplotlib.

Importing Matplotlib

Like NumPy, there are two ways to import and use Matplotlib:

- directly: the functions will reside in matplotlib.pyplot/plt

```
import matplotlib.pyplot as plt
plt.plot(...)
```

- via pylab: the functions are global (faster, more dangerous)

```
from pylab import *
plot(...)
```

Note: I will use direct import in the following examples.

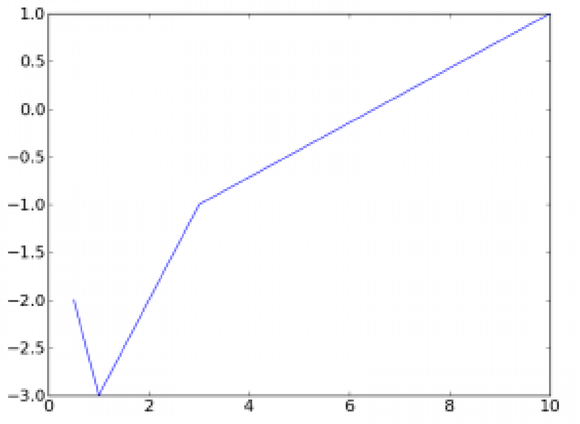
Basic plotting

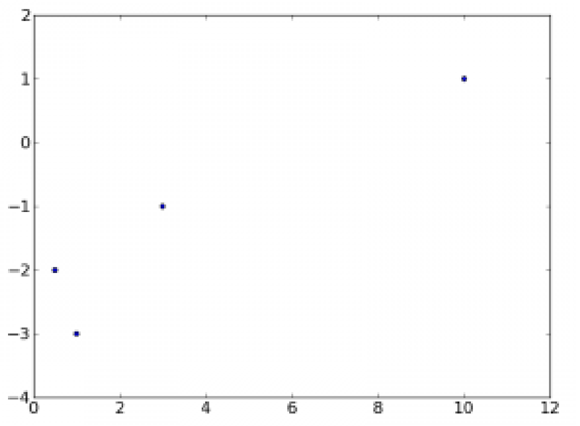
A line has x and y coordinates. How do you plot it?

```
x = [0.5, 1, 3, 10]
y = [-2, -3, -1, 1]
plt.plot(x, y)
```

How to plot single points instead of a line?

```
plt.scatter(x, y)
```





Wait a sec! No figure (or the likes)!

If you have problems such as the figure does not show up, update, etc., you have to activate the interactive mode at the beginning of your script:

```
plt.ion()
```

and/or to call the figure explicitely:

```
# To show the figure
plt.show()

# To update an existing figure
plt.draw()
```

Save your plot

After a plot is done, you can save the figure with:

```
plt.savefig('my_figure.png')
```

Note: you can save in different formats by just changing the format to e.g. '.pdf' or '.svg'.

What is x?

Usually, you want to add some more information to the graph than a simple curve. To add axes labels:

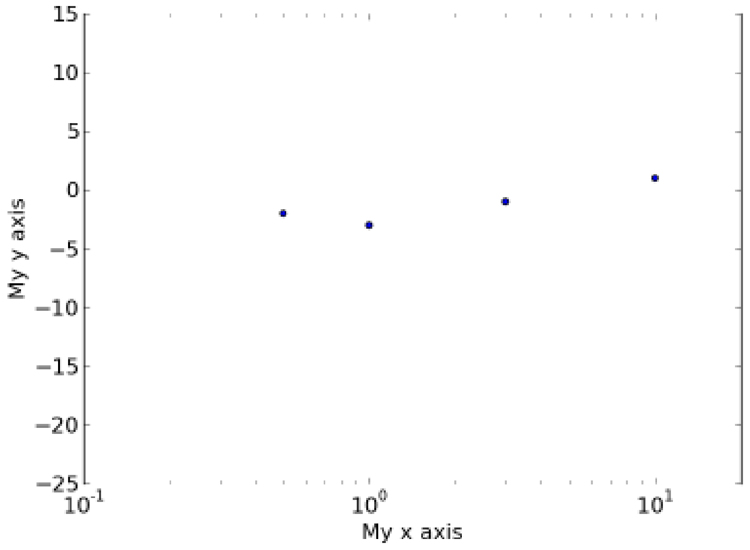
```
plt.xlabel('My x axis')
plt.ylabel('My y axis')
```

To limit the range of the plot:

```
plt.xlim(-10, 20)
plt.ylim(-25, 15)
```

You can switch between linear and logarithmic scales:

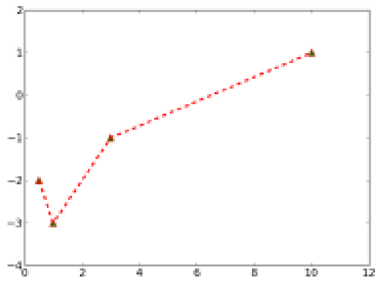
```
# Use log scale
plt.xscale('log')

# Go back to linear scale
plt.xscale('linear')
```



Make it pretty!

You can of course change the style of your plots. You can change single lines/scatters via:

```
plt.plot(x, y, color='red', linewidth=3, linestyle='--')
plt.scatter(x, y, marker='^', s=100, edgecolor='red', facecolor='green')
```



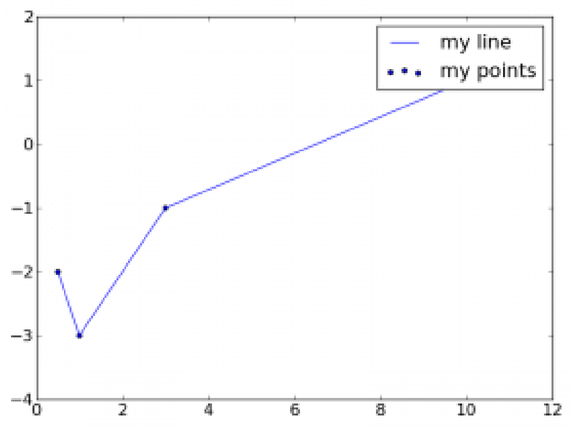
Legends

If you want to have a legend, you have to give **labels** to your curves:

```
plt.plot(x, y, label='my line')
plt.scatter(x, y, label='my points')
```

and finally show the legend:

```
plt.legend()
```



Histograms

If you have a table of data and want to plot the histogram:

```
plt.hist(data)
```