Introduction to scientific plotting in Python

CS101 lectures : part 5.2

Professor Lilian Besson

Mahindra École Centrale (School of Computer Science)

April 13th, 2015

Mahindra
École Centrale

Please contact me by email if needed: CS101@crans.org

Slides and examples will be uploaded on **Moodle**.

Overview of the content of this lecture

Again : stay focus, listen and take notes

If you talk or disturb my lectures, I will kick you out *immediately*.
OK ?

Overview of the content of this lecture

1. Presentation
2. The MatPlotLib Python module
   - About MatPlotLib : what, how and why?
   - Different kinds of plotting
3. The bases : how to start using MatPlotLib?
   - How to import MatPlotLib ?
   - Basic concepts
   - Simple examples
   - Customizing a MatPlotLib graphic
4. Conclusion of this first lecture

Again : stay focus, listen and take notes

If you talk or disturb my lectures, I will kick you out *immediately*.
OK ?

What are scientific computation and plotting?

Scientific computation ?

- Compute an expression or a result,
- Approximatively solve a numerical problem,
- Simulate real-world issues with maths and more ...

$\Longrightarrow$ You will study this a lot during the next years.

Scientific plotting ?

- Illustrate a phenomena,
- Visualize data,
- Communicate graphically (like in SE!).

$\Longrightarrow$ You will also practice this during the next years.

What are scientific computation and plotting?

Scientific computation ?

- Compute an expression or a result,
- Approximatively solve a numerical problem,
- Simulate real-world issues with maths and more ...

$\Longrightarrow$ You will study this a lot during the next years.

Scientific plotting ?

- Illustrate a phenomena,
- Visualize data,
- Communicate graphically (like in SE!).

$\Longrightarrow$ You will also practice this during the next years.

What module will we use for plotting data in Python?

We will use the most popular Python module for plotting : MatPlotLib

- MatPlotLib is the single most used Python package for 2D-graphics.
- It provides both a very quick way to visualize data from Python and publication-quality figures in many formats.

The logo of the MatPlotLib project, generated with MatPlotLib itself

References ?

⟶ On-line: `matplotlib.org` (official website).

⟶ How to download? It is included in Anaconda (but can also easily be installed).

⟶ How to learn? Be attentive today and during the next 2 lectures.
And use this tutorial: `github.com/rougier/matplotlib-tutorial`.

What module will we use for plotting data in Python?

We will use the most popular Python module for plotting : MatPlotLib

- MatPlotLib is the single most used Python package for 2D-graphics.
- It provides both a very quick way to visualize data from Python and publication-quality figures in many formats.

The logo of the MatPlotLib project, generated with MatPlotLib itself

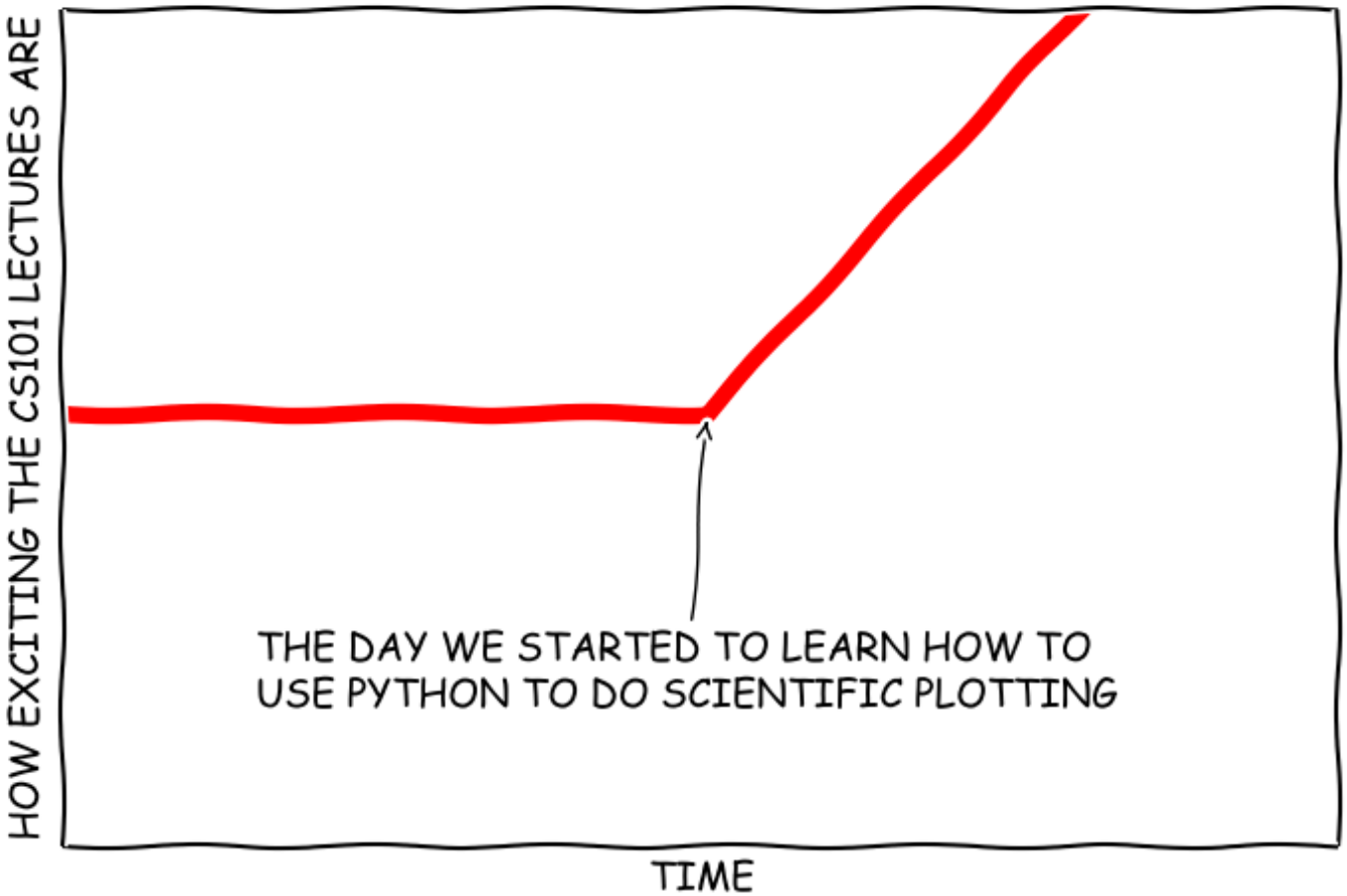
References ?

⟶ On-line: `matplotlib.org` (official website).

⟶ How to download? It is included in Anaconda (but can also easily be installed).

⟶ How to learn? Be attentive today and during the next 2 lectures.
And use this tutorial: `github.com/rougier/matplotlib-tutorial`.

A first demo of what MatPlotLib can do



First demo of MatPlotLib (XKCD style)

A meta example of the MatPlotLib module

What kind of plotting is possible?

Common 2D or 3D scientific plots are supported

We will see basic examples today, and more next week:

- Functions $y = f(x)$,
- Parametric curves: $(x(t), y(t))$,
- Discrete data: (x_i, y_i), for $i \in [1, \ldots, n]$,
- *(Next week)* polar curves, surface $z = f(x, y)$, histogram, pie charts etc.

A MatLab-like interface

MatPlotLib module `pyplot` provides an declarative way of plotting data, really similar to the syntax of MatLab[a]

[a] Note that you will learn and practice with MatLab next year in EE203.

Many examples

Can I see more examples?

Yes, on these page, you will find many examples:

- Some examples from MA101 and MA102:
  → bitbucket.org/lbesson/python-demos,
- Official gallery (with about 450 examples):
  → matplotlib.org/devdocs/examples,
- Another gallery, by N. Rougier:
  → github.com/rougier/gallery.

How to load the MatPlotLib package?

The **preferred** way is the following

```
import numpy as np    # NumPy : Numerical Python
import matplotlib     # Matlab-like Plotting Library
import matplotlib.pyplot as plt
```

- Maths functions and constants are available with the `np.` prefix: `np.cos`, `np.exp`, `np.pi` etc,
- Plotting functions are available with the `plt.` prefix: `plt.plot`, `plt.title`, `plt.savefig`, `plt.legend` etc.

The *lazy* way is the following (using `pylab`)

```
from pylab import *
```

It's a bad habit, but it's quick.

How to load the MatPlotLib package?

The **preferred** way is the following

```
import numpy as np    # NumPy : Numerical Python
import matplotlib     # Matlab-like Plotting Library
import matplotlib.pyplot as plt
```

- Maths functions and constants are available with the `np.` prefix: `np.cos`, `np.exp`, `np.pi` etc,
- Plotting functions are available with the `plt.` prefix: `plt.plot`, `plt.title`, `plt.savefig`, `plt.legend` etc.

The *lazy* way is the following (using `pylab`)

```
from pylab import *
```

It's a bad habit, but it's quick.

Basic concepts of a MatPlotLib plotting script

A MatPlotLib plotting script will be

1. **import the required modules**
   (`numpy as np` and `matplotlib.pyplot as plt`),
2. **load** or **generate some data**,
3. *(optional)* customize the appearance of your figure,
4. **generate a figure** (`plot()`, `bar()`, `pie()` etc),
5. **display it** or **save it** in a file (many formats: PNG, PDF, SVG etc),

The simplest kind of plot : the `plt.plot` function

The basic use of MatPlotLib is to draw some (x, y) 2D points.
If we have some values x_i and y_i $(0 \leqslant i \leqslant n-1)$, stored in two lists (or arrays) `X` and `Y`, then it is easy to draw them with `plt.plot(X, Y)`.

Basic concepts of a MatPlotLib plotting script

A MatPlotLib plotting script will be

1. **import the required modules**
   (`numpy as np` and `matplotlib.pyplot as plt`),
2. **load** or **generate some data**,
3. *(optional)* customize the appearance of your figure,
4. **generate a figure** (`plot()`, `bar()`, `pie()` etc),
5. **display it** or **save it** in a file (many formats: PNG, PDF, SVG etc),

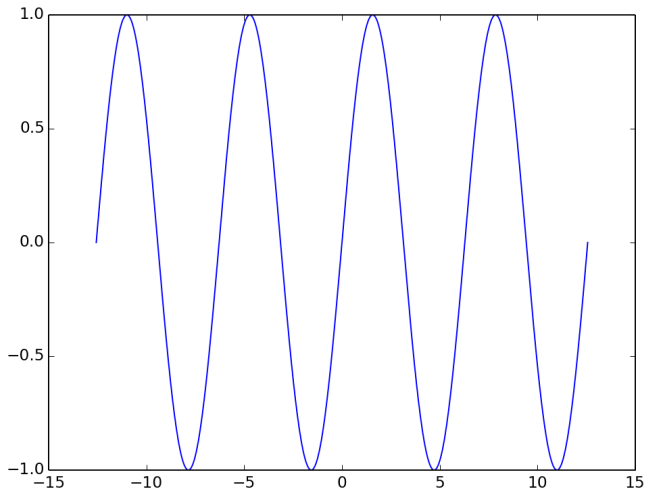
The simplest kind of plot : the `plt.plot` function

The basic use of MatPlotLib is to draw some (x, y) 2D points.
If we have some values x_i and y_i $(0 \leqslant i \leqslant n-1)$, stored in two lists (or arrays) `X` and `Y`, then it is easy to draw them with `plt.plot(X, Y)`.

First example of a MatPlotLib plotting script

Plotting $f : x \mapsto \sin(x)$ in three lines?

```
# 500 points from -4*pi to 4*pi
X = np.linspace(-4*np.pi, 4*np.pi, 500)
Y = np.sin(X)   # computes the values one by one
plt.plot(X, Y)
```



Plot of $y = \sin(x)$ for $-4\pi \leqslant x \leqslant 4\pi$.

Second example of a MatPlotLib plotting script

Plotting more than one function: three possibilities

- several successive calls of `plt.plot`,
- or several pairs of (X, Y) data in *one* use of `plt.plot`,
- more complicated: sub-plots. More details on the online tutorial.

Example with $x \mapsto \sin(x)$ and $x \mapsto \cos(x)$

```
X = np.linspace(-4*np.pi, 4*np.pi, 500)
# 500 points from -4*pi to 4*pi
Y1 = np.sin(X)
Y2 = np.cos(X)
plt.plot(X, Y1, 'r-')  # red continuous line
plt.plot(X, Y2, 'g:')  # green dotted line
# plt.plot(X, Y1, 'r-', X, Y2, 'g:')  # will do the ↩
    same!
```

Second example of a MatPlotLib plotting script

Plotting more than one function: three possibilities

- several successive calls of `plt.plot`,
- or several pairs of (X, Y) data in *one* use of `plt.plot`,
- more complicated: sub-plots. More details on the online tutorial.

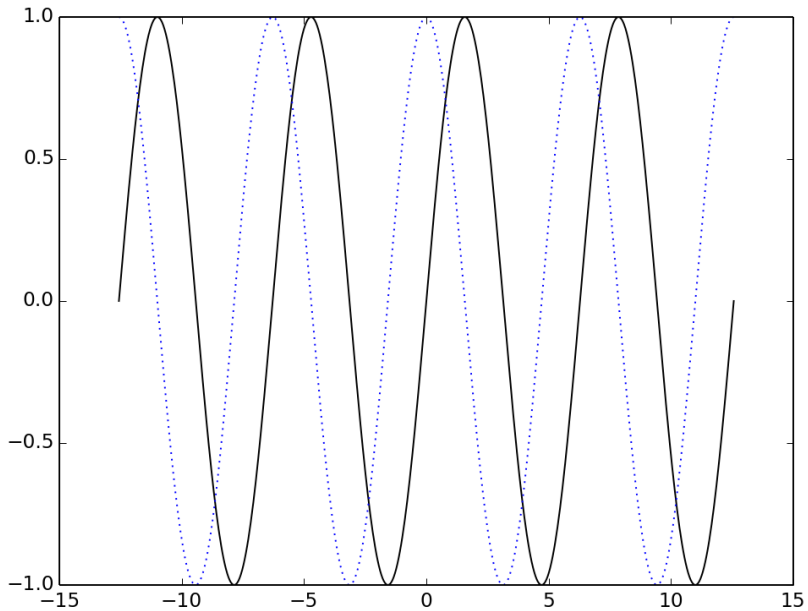
Example with $x \mapsto \sin(x)$ and $x \mapsto \cos(x)$

```
X = np.linspace(-4*np.pi, 4*np.pi, 500)
# 500 points from -4*pi to 4*pi
Y1 = np.sin(X)
Y2 = np.cos(X)
plt.plot(X, Y1, 'r-')   # red continuous line
plt.plot(X, Y2, 'g:')   # green dotted line
# plt.plot(X, Y1, 'r-', X, Y2, 'g:')   # will do the ↩
    same!
```

Second example of a MatPlotLib plotting script

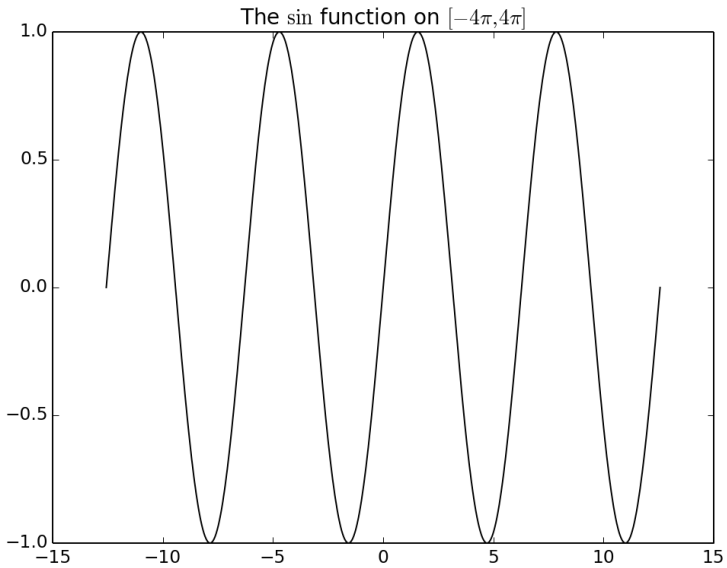
Example

```
X = np.linspace(-4*np.pi, 4*np.pi, 500)
Y1, Y2 = np.sin(X), np.cos(X)
plt.plot(X, Y1, 'r-', X, Y2, 'g:')
```



Plot of $y = \sin(x)$ and $y = \cos(x)$ (for $-4\pi \leqslant x \leqslant 4\pi$).

Adding a title or a legend

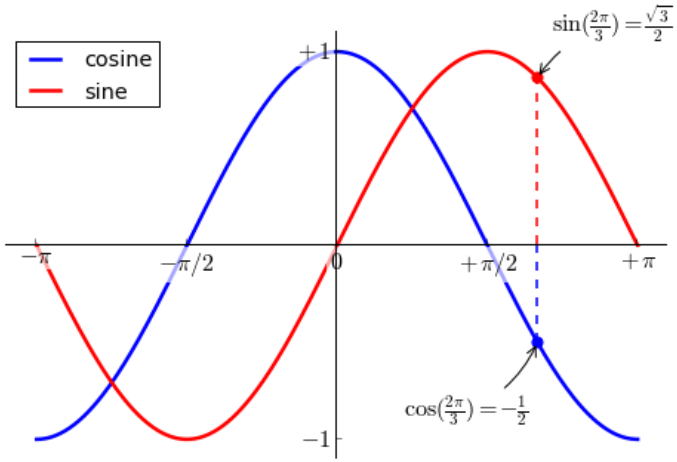


Plot of $y = \sin(x)$ and $y = \cos(x)$ (for $-4\pi \leqslant x \leqslant 4\pi$).

Adding a title or a legend?

- `plt.title()` is called with a single string,
- `plt.legend()` is more complicated,
- and almost anything else is also possible.

Many more options are available



Plot of $y = \sin(x)$ and $y = \cos(x)$ (for $-\pi \leqslant x \leqslant \pi$).

We can also ...

- change the ticks and labels of x and y axes (`ticks`, `x,ylabel`),
- draw any extra information (like the vertical dotted line),
- highlight one specific point,
- write some LaTeX-aware text anywhere in the figure,
- change the transparency of any line, etc.

Quick sum-up about **MatPlotLib**

As an introduction to plotting with MatPlotLib, **we just saw:**

- Where to get some examples on-line,
- How to *properly* import the required modules,
- How to write a simple plotting example (3 lines!),
- More than one plot in an image by overlapping (subplots are harder),
- Add a title or a legend.

More will be studied *next week...* (*new!*)

- Saving plots to an image (`savefig`),
- Polar or logarithmic plots (`polar`, `loglog`),
- Pie charts, histograms etc (`pie`, `bar`),
- 3D plots (contour, surface etc) (`contour`, `contourf`, `plot_surface`),
- Displaying an image (`imread`, `imshow`),
- Bonus: embedding some plots in a web-page (with `mpld3`).

Quick sum-up about **MatPlotLib**

As an introduction to plotting with MatPlotLib, **we just saw:**

- Where to get some examples on-line,
- How to *properly* import the required modules,
- How to write a simple plotting example (3 lines!),
- More than one plot in an image by overlapping (subplots are harder),
- Add a title or a legend.

More will be studied *next week...* (*new!*)

- Saving plots to an image (`savefig`),
- Polar or logarithmic plots (`polar`, `loglog`),
- Pie charts, histograms etc (`pie`, `bar`),
- 3D plots (contour, surface etc) (`contour`, `contourf`, `plot_surface`),
- Displaying an image (`imread`, `imshow`),
- Bonus: embedding some plots in a web-page (with `mpld3`).

Thanks for listening today

Any question?

Reference websites

- The http://matplotlib.org/ documentation for **MatPlotLib**.
- www.MahindraEcoleCentrale.edu.in/portal : **MEC Moodle**.
- And the Python documentation at docs.python.org/2/.

Want to know more?

↪ Practice by yourself on the (450) examples in MatPlotLib doc!

↪ And contact us (e-mail, *flying pigeons*, Moodle etc) if needed.

Next 2 lectures: **more on scientific computations and plotting with Python** (by me again)