Programming in Python[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Academic year 2025/26, I semester

[1] © 2025 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. http://creativecommons.org/licenses/by-sa/4.0/deed.it

Lecture XV: Matplotlib

Matplotlib

When you have arrays with many data it is useful to have a way to display them graphically.

- The most popular is `matplotlib` https://matplotlib.org/
- Many other graphical frameworks (e.g., `seaborn`, `plotnine`) based on it
- Many, many possibilities to tune your graphics! It's hard to master every detail.
- Be careful: it can be used with two different styles.
  1. The (preferred) object-oriented way: clean and rational, but a bit more verbose
  2. The procedural way: mostly useful only for "throw-away" scripts, but for this reason more common in the examples you can find online

Graphical output is an operating system service

- Output is a service provided by the operating system: *textual* output is very standardized even across different platform, graphics is not so stable
- When you deal with graphical programs: expect installation headaches, portability glitches, etc.

The OO style

- You need always to objects: a `Figure` and a `Axes`
- plotting happens on axes, framed in a figure
- very flexible: you can add plots on the same axis, or you can have many axes collected in a single figure

Basic example


```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-2*np.pi, 2*np.pi, 100)

fig, ax = plt.subplots()

ax.plot(x, np.sin(x))

fig.show()
```

Many different types of charts

If `ax` is a `Axes`

- Scatter-plots `ax.scatter`
- Bar-plots `ax.bar`
- Histograms `ax.hist`
- 2D `ax.imshow`

Tweaks

- add labels, legends, titles
- add a grid
- combine multiple plots on the same axis
- combine multiple axes on the same figure

Save your pictures!

A `Figure` can be saved in a file with `savefig`. You should keep in mind the difference between:

- bitmap formats (`png` `jpg` ...): the file is matrix of pixels
- vector formats (`svg` `pdf` ...): the file is a set of instructions to reproduce the picture, less portable but it can be magnified

Using the notebook in a virtual environment

Since we are now interested in graphics, Jupyter notebooks can be very convenient to see pictures together with the code.

1. We set up a virtual environment as usual
2. With `pip install notebook` we have the Jupyter notebook machinery available
3. I normally want to have also a clean `.py` file, since `.ipynb` do not play well with configuration management (git) and other command line tools like the type checker or doctest: thus I suggest to install `jupytext`; it needs a `jupytext.toml` text file telling `.ipynb` and `.py` files are paired, *i.e.*, they are kept synchronized.

   ```
   # Always pair ipynb notebooks to py files
   formats = "ipynb,py:percent"
   ```

4. launch the notebook with `jupyter notebook`