Introduction to Scientific Python

Iñigo Aldazabal Mensa – Centro de Física de Materiales
(CSIC-UPV/EHU)

CFM Software Carpentry Tracks

San Sebastián, 4th May 2017

Why Python?

Why Python?

Python is inherently slow compared to C/C++ or FORTRAN, so why Python for Scientific Computing?

Python is slow, but...

> *Syntactically, Python code looks like executable pseudo code. Program development using Python is 5-10 times faster than using C/C++...*
>
> *The best approach is often to write only the performance-critical parts of the application in C++ or Java, and use Python for all higher-level control and customization.*
>
> *– Guido van Rossum*

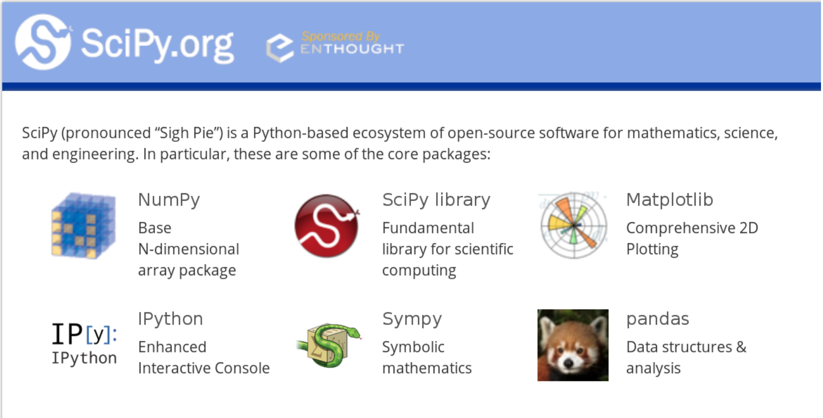
Why Python?

We have (for free)

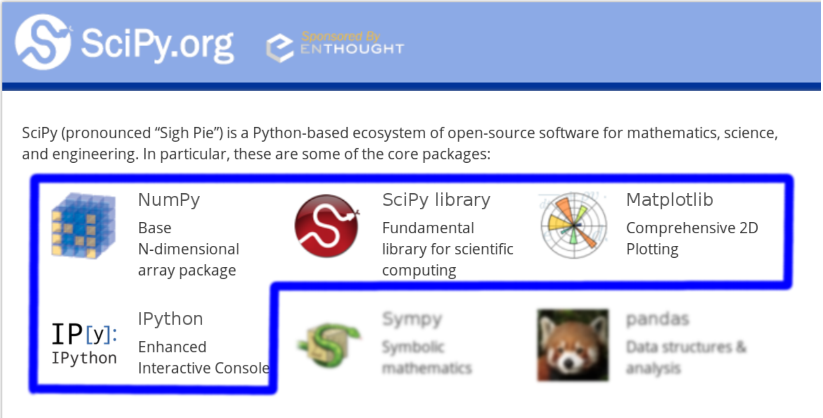
- A general purpose language with a huge spectrum of freely available libraries for almost anything you can think of.
- A very easy to learn (and read) language that smoothly interfaces with C/C++ and FORTRAN (eg. calculation kernels).
- Lots of wrappers for well stablished, fast and long time tested numerical packages.
- Lots of high level utility libraries for scientific computing: plotting, data analisys, parallelization, ...

Why Python? Batteries Included

Python Scientific Computing Environment



Python Scientific Computing Environment



NumPy

NumPy
Base
N-dimensional
array package

SciPy library
Fundamental library for scientific computing

Matplotlib
Comprehensive 2D Plotting

IPython
Enhanced Interactive Console

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

NumPy

NumPy
Base
N-dimensional
array package

SciPy library
Fundamental
library for scientific
computing

Matplotlib
Comprehensive 2D
Plotting

IPython
Enhanced
Interactive Console

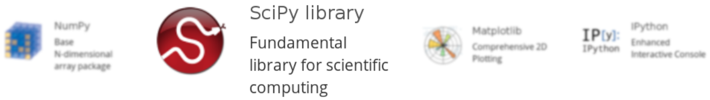
```
import numpy as np

# Create a numpy array, x
x = np.array( [1.1, 1.3, 1.5] )
y = np.sin(a)

# create a random two dimensional numpy array, A
A = np.random.rand(3,3)

A.transpose()
A.trace()
```

SciPy



SciPy is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python.

Much of SciPy is a thin layer of code on top of the C and FORTRAN scientific routines that are freely available at http://www.netlib.org/.

Provides the user with high-level commands and classes for manipulating and visualizing data.

With SciPy an interactive Python session becomes a data-processing and system-prototyping environment rivaling sytems such as MATLAB, IDL, Octave, R-Lab, and SciLab.

SciPy

NumPy
Base
N-dimensional
array package

SciPy library
Fundamental
library for scientific
computing

Matplotlib
Comprehensive 2D
Plotting

IPython
Enhanced
Interactive Console

SciPy subpackages (some of them)

- constants: Physical and mathematical constants
- fftpack: Fast Fourier Transform routines
- integrate: Integration and ordinary differential equation solvers
- interpolate: Interpolation and smoothing splines
- linalg: Linear algebra
- optimize: Optimization and root-finding
- signal: Signal processing
- special: Special functions
- ...

SciPy

NumPy
Base
N-dimensional
array package

SciPy library
Fundamental
library for scientific
computing

Matplotlib
Comprehensive 2D
Plotting

IPython
Enhanced
Interactive Console

```
import numpy as np
from scipy.special import gamma

x = np.array( [1.1, 2., 3.] )
y = gamma(x)

print (y)
```

```
[ 0.95135077,  1. ,  2. ]
```

matplotlib

Matplotlib
Comprehensive 2D Plotting

matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

matplotlib can be used in Python scripts, the IPython shell, Jupyter Notebooks (ala MATLAB® or Mathematica®), and several graphical user interface toolkits.

For simple plotting the `pyplot` interface provides a MATLAB-like interface, particularly when combined with the `IPython / Jupyter` environment.

SciPy

NumPy Base N-dimensional array package

SciPy library Fundamental library for scientific computing

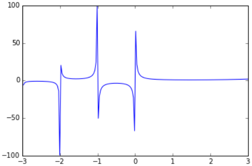
Matplotlib Comprehensive 2D Plotting

IPython Enhanced Interactive Console

```
import numpy as np
from scipy.special import gamma
import matplotlib.pyplot as plt

x = np.linspace( start=-3., stop
     =3., num=200 )
y = gamma(x)

plt.plot(x,y)
plt.savefig("gamma.png")
```



SciPy

NumPy Base N-dimensional array package

SciPy library Fundamental library for scientific computing

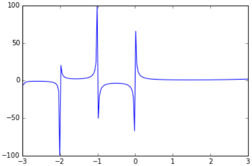
Matplotlib Comprehensive 2D Plotting

IPython Enhanced Interactive Console

```
import numpy as np
from scipy.special import gamma
import matplotlib.pyplot as plt

x = np.linspace( start=-3., stop
    =3., num=200 )
y = gamma(x)

plt.plot(x,y)
plt.savefig("gamma.png")
```



Just google matplotlib images for examples!

IPython / Jupyter

IPython
Enhanced
Interactive Console

The **IPython / Jupyter Notebook** is an open-source web-based interactive computing system that enables users to create and share documents that contain live code, $\LaTeX$equations, visualizations and explanatory text.

These documents contain a full record of a computation and its results and can be shared on email, Dropbox, version control systems (like git/GitHub) or with the Jupyter online notebook viewer nbviewer.jupyter.org.

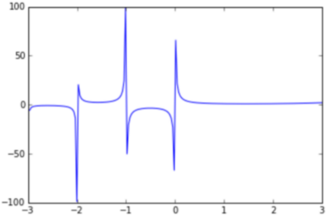
IPython / Jupyter

```
In [45]: %matplotlib inline
         import numpy as np
         from scipy.special import gamma
         import matplotlib.pyplot as plt

         x = np.linspace( start=-3., stop=3., num=200 )
         y = gamma(x)

         plt.plot(x,y)

Out[45]: [<matplotlib.lines.Line2D at 0x7fc4ac4660b8>]
```




```
In [ ]: |
```

Jupyter

jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Markdown CellToolbar

Simple spectral analysis

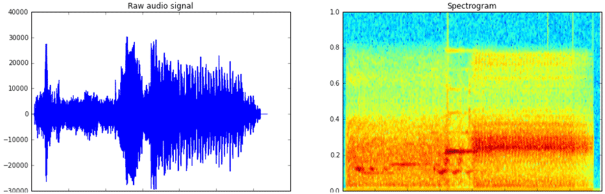
An illustration of the Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N}kn} \quad k = 0, \ldots, N-1$$

In [2]:
```python
from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

In [5]:
```python
fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```



Python Scientific Computing Environment

Let's try it!