Introduction to Scientific Python

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

Software Carpentry Workshop

2023-01-10-cfm

CFM, San Sebastian, 27th January 2023

Why Python?

python
powered

```
print("Hello, world!")
```

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:

- A Python program can have 5-10 times less lines than its C or FORTRAN counterpart
- Thus, program development using Python is 5-10 times faster than using C/C++...

and...

The number of bugs in a program scales linearly with the number of lines of the program.

Why Python?

We should forget about small efficiencies, say about 97% of the time: ***premature optimization is the root of all evil*** *(in programming).*

– Computer Programming as an Art (1974), Donald Knuth

In software engineering, it is a good approximation that 90% of the execution time of a computer program is spent executing 10% of the code

```
print("Hello, world!")
```

Why Python?

We should forget about small efficiencies, say about 97% of the time: ***premature optimization is the root of all evil*** *(in programming).*

– Computer Programming as an Art (1974), Donald Knuth

In software engineering, it is a good approximation that 90% of the execution time of a computer program is spent executing 10% of the code

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...

Why Python? ...and PyPi

Find, install and publish Python packages with the Python Package Index

Search projects

Or browse projects

431,155 projects | 4,119,890 releases | 7,451,535 files | 663,148 users

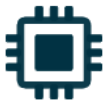
The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages.

Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

Python Scientific Computing Environment

Quantum Computing



QuTiP
PyQuil
Qiskit
PennyLane

Statistical Computing

Pandas
statsmodels
Xarray
Seaborn

Signal Processing

SciPy
PyWavelets
python-control

Image Processing

Scikit-image
OpenCV
Mahotas

Graphs and Networks

NetworkX
graph-tool
igraph
PyGSP

Astronomy Processes

AstroPy
SunPy
SpacePy

Cognitive Psychology

PsychoPy

Bioinformatics

BioPython
Scikit-Bio
PyEnsembl
ETE

Bayesian Inference

PyStan
PyMC3
ArviZ
emcee

Mathematical Analysis

SciPy
SymPy
cvxpy
FEniCS

Chemistry

Cantera
MDAnalysis
RDKit

Geoscience

Pangeo
Simpeg
ObsPy
Fatiando a Terra

Geographic Processing

Shapely
GeoPandas
Folium

Architecture & Engineering

COMPAS
City Energy Analyst
Sverchok

Python Scientific Computing Environment

The SciPy stack consists of Python along with the most commonly used scientific, mathematical, and ML libraries.

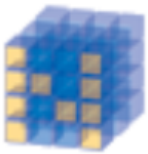
These include **NumPy**, **Matplotlib**, the **SciPy** library itself, and **IPython / Jupyter**

Python Scientific Computing Environment

The SciPy stack consists of Python along with the most commonly used scientific, mathematical, and ML libraries.

These include **NumPy**, **Matplotlib**, the **SciPy** library itself, and **IPython / Jupyter**

NumPy



NumPy

Base
N-dimensional
array package

SciPy library
Fundamental
library for scientific
computing

Matplotlib
Comprehensive 2D
Plotting

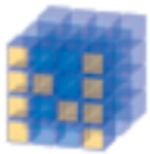
IP[y]:
IPython

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

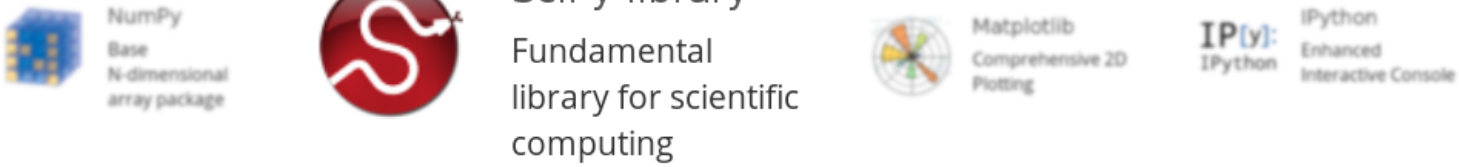
Matplotlib
Comprehensive 2D Plotting

IP[y]: IPython

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(x)

# 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

SciPy library

Fundamental library for scientific computing

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

SciPy library

Fundamental library for scientific computing


```
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 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.

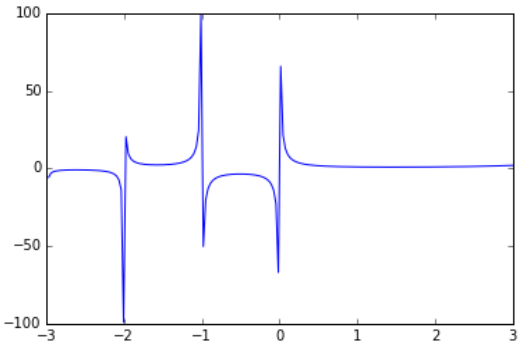
For simple plotting the `pyplot` interface provides a MATLAB-like interface, particularly when combined with the `IPython / Jupyter` environment.

SciPy

Matplotlib

Comprehensive 2D Plotting


```
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")
```

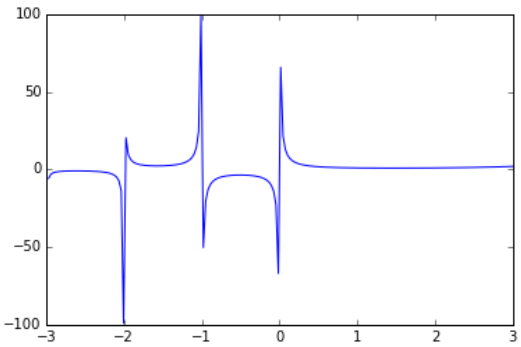


SciPy

Matplotlib

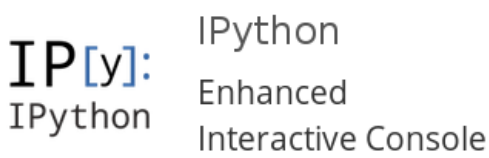
Comprehensive 2D Plotting


```
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 browse matplotlib gallery for examples!

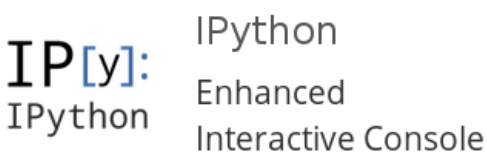
IPython / Jupyter



The **IPython** / **Jupyter Notebook** / **JupyterLab** is an open-source web-based interactive computing system that enables users to create and share documents that contain live code, LaTeXequations, 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



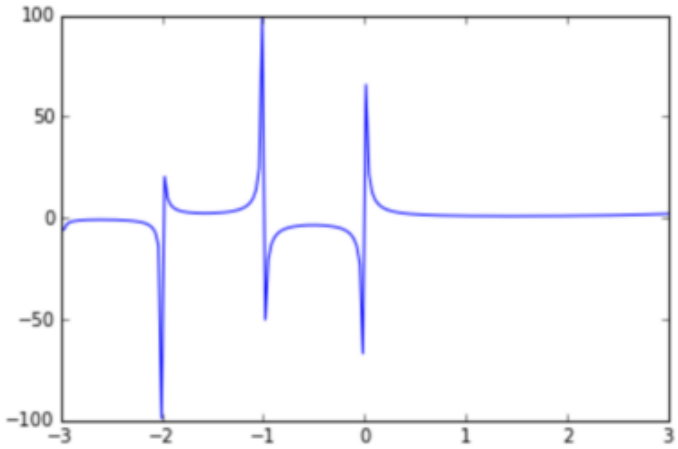
Ipython / Jupyter notebooks allows us to:

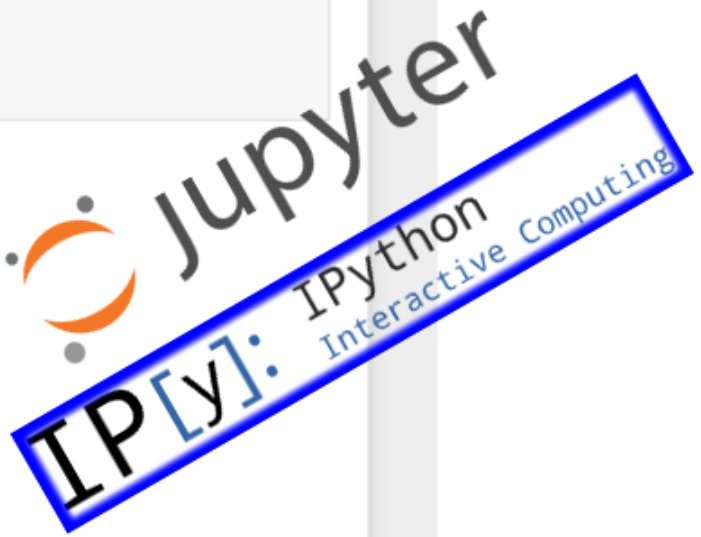
- **Edit code in the browser**, with syntax highlighting, indentation, and tab completion/introspection.
- **Run code in the browser**, with results attached to the code generating them.
- See the results with **rich media representation** as HTML, LaTeX, PDF, PNG, etc.
- Embed **interactive user interface controls** and visualization.
- Author **narrative text** using Markdown markup language.
- Build **hierarchical documents** with headings, sections, etc.
- Use LaTeX syntax in Markdown, rendered in the browser.

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 [ ]:

IPython / Jupyter

jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Markdown CellToolbar

Simple spectral analysis

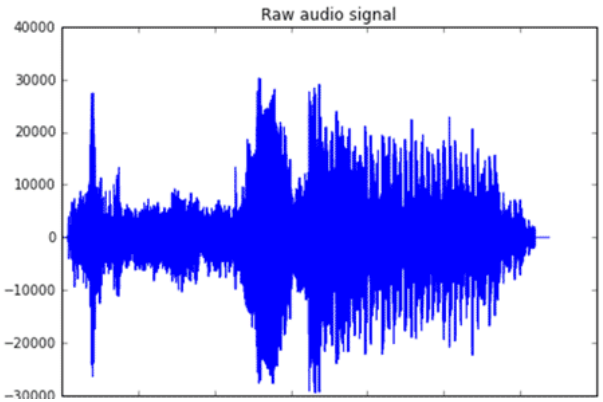
An illustration of the Discrete Fourier Transform

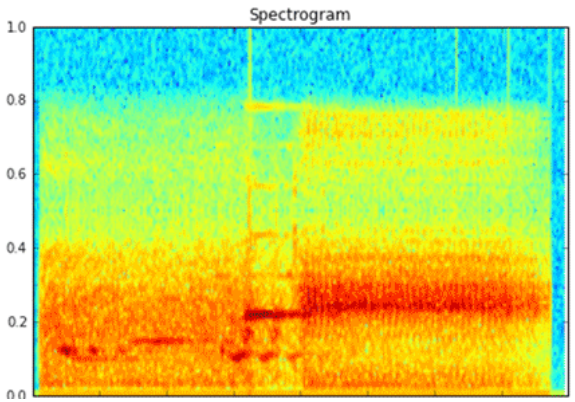
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N} kn} \quad k = 0, \ldots, N-1$$

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

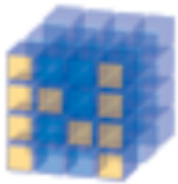
And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

In [5]:
```
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 play with it!