



tutorial : modeling synaptic plasticity

“Computational Neuroscience by the Mediterranean”

Winter School, Jan 20th, 2016

Michael Graupner

Université Paris Descartes – CNRS UMR 8118, Paris, France

michael.graupner@parisdescartes.fr

slides & scripts: http://www.biomedicale.univ-paris5.fr/~mgraupe/beirut_python

Outline

1. General introduction

why python, installation, running python

2. First steps

syntax, modules and functions

(interactive IPython Notebook)

3. Brian introduction

simulating a spiking recurrent network

(interactive IPython Notebook)

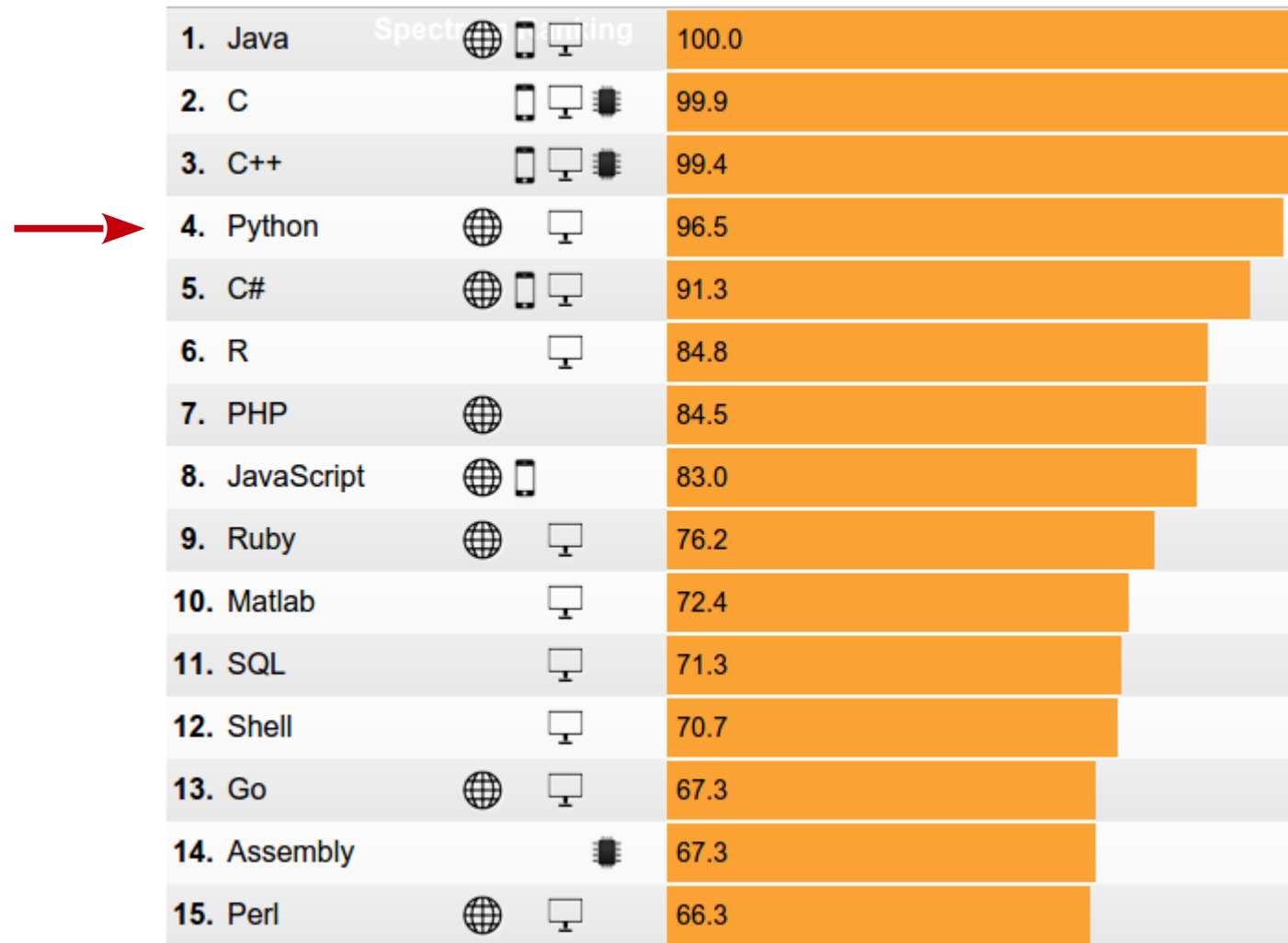
4. Modeling calcium-based synaptic plasticity

building a model simulation in python - exercises

(interactive IPython Notebook)

Python – modern object-oriented programming language

The Top Programming Languages 2015



[Source : IEEE Spectrum]

Python – very clear, readable syntax → easy to learn

```
In [1]: # import modules
        from pylab import *

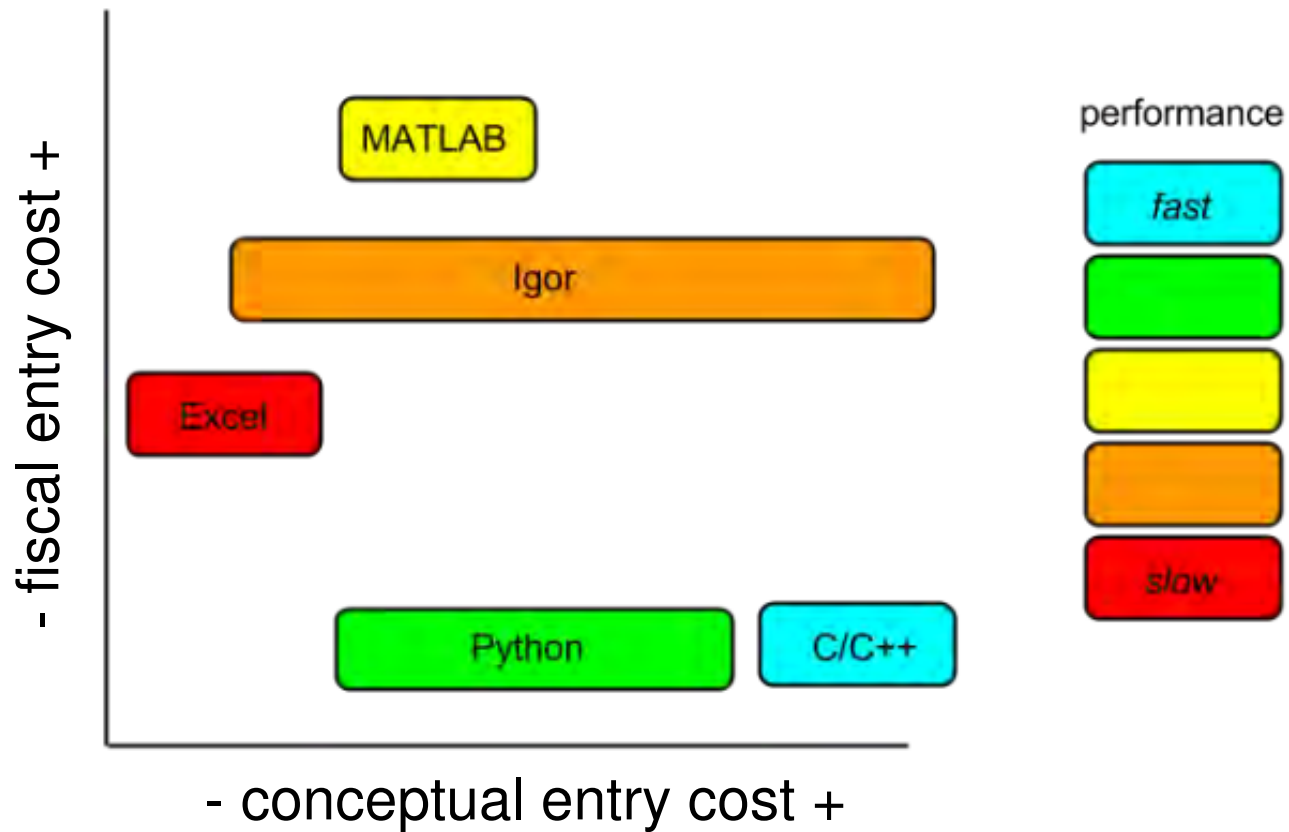
        # function declaration
        def update_values(x):
            return x+1

        x = 1
        if x>0:
            print 'Hello World!'
            x = update_values(x)

        print x
```

```
Hello World!
2
```

Python – extensive standard libraries, “Batteries included”



Python – third party extensions for virtually every task:

e.g. Python bindings for GUI toolkit



Python modules for Neuroscience

- simulators and simulator interfaces
- data collection and analysis
- sharing, re-use, storage and databasing of models and data
- stimulus generation
- parameter search and optimization
- visualization
- VLSI (very-large-scale integration) hardware interfacing

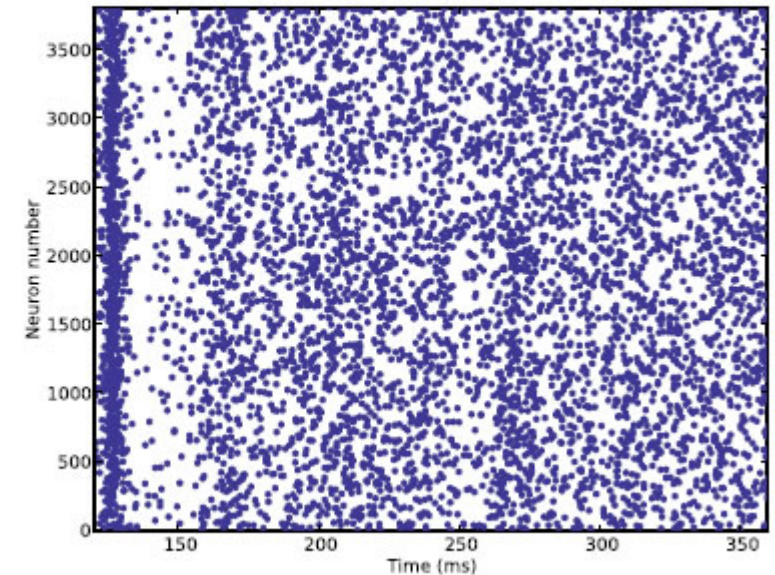
Py in Neuroscience : simulators and simulator interfaces

e.g. Brian – the spiking neural network simulator

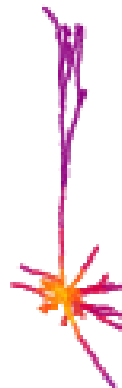
BRIAN

recurrent, randomly connected network

```
1 from brian import *
2 eqs = '''
3 dv/dt = (ge+gi-(v+49*mV))/(20*ms) : volt
4 dge/dt = -ge/(5*ms) : volt
5 dgi/dt = -gi/(10*ms) : volt
6 '''
7 P = NeuronGroup(4000, eqs, threshold=-50*mV, reset=-60*mV)
8 P.v = -60*mV+10*mV*rand(len(P))
9 Pe = P.subgroup(3200)
10 Pi = P.subgroup(800)
11 Ce = Connection(Pe, P, 'ge', weight=1.62*mV, sparseness=0.02)
12 Ci = Connection(Pi, P, 'gi', weight=-9*mV, sparseness=0.02)
13 M = SpikeMonitor(P)
14 run(1*second)
15 raster_plot(M)
16 show()
```

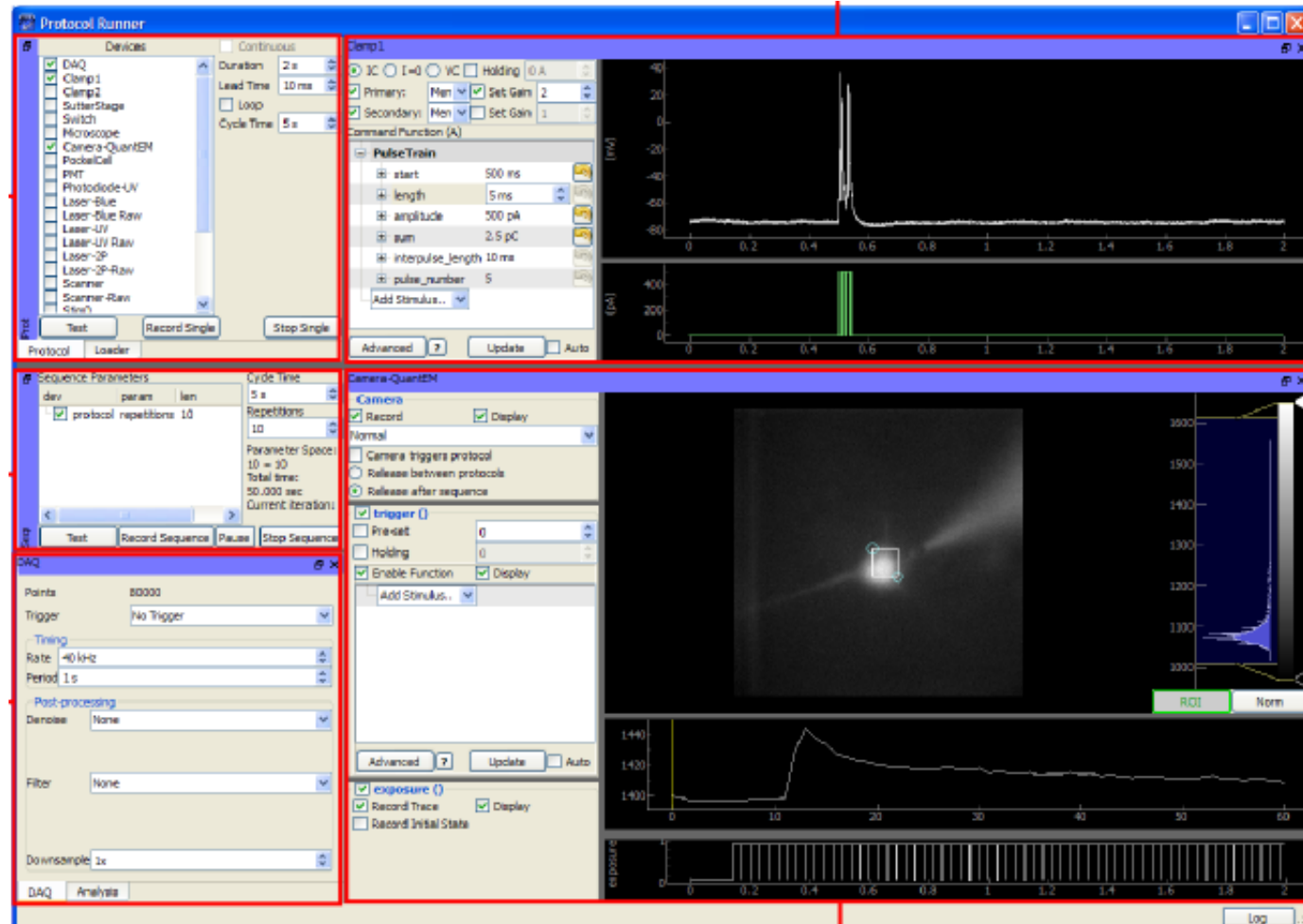


e.g. Python interface for NEURON



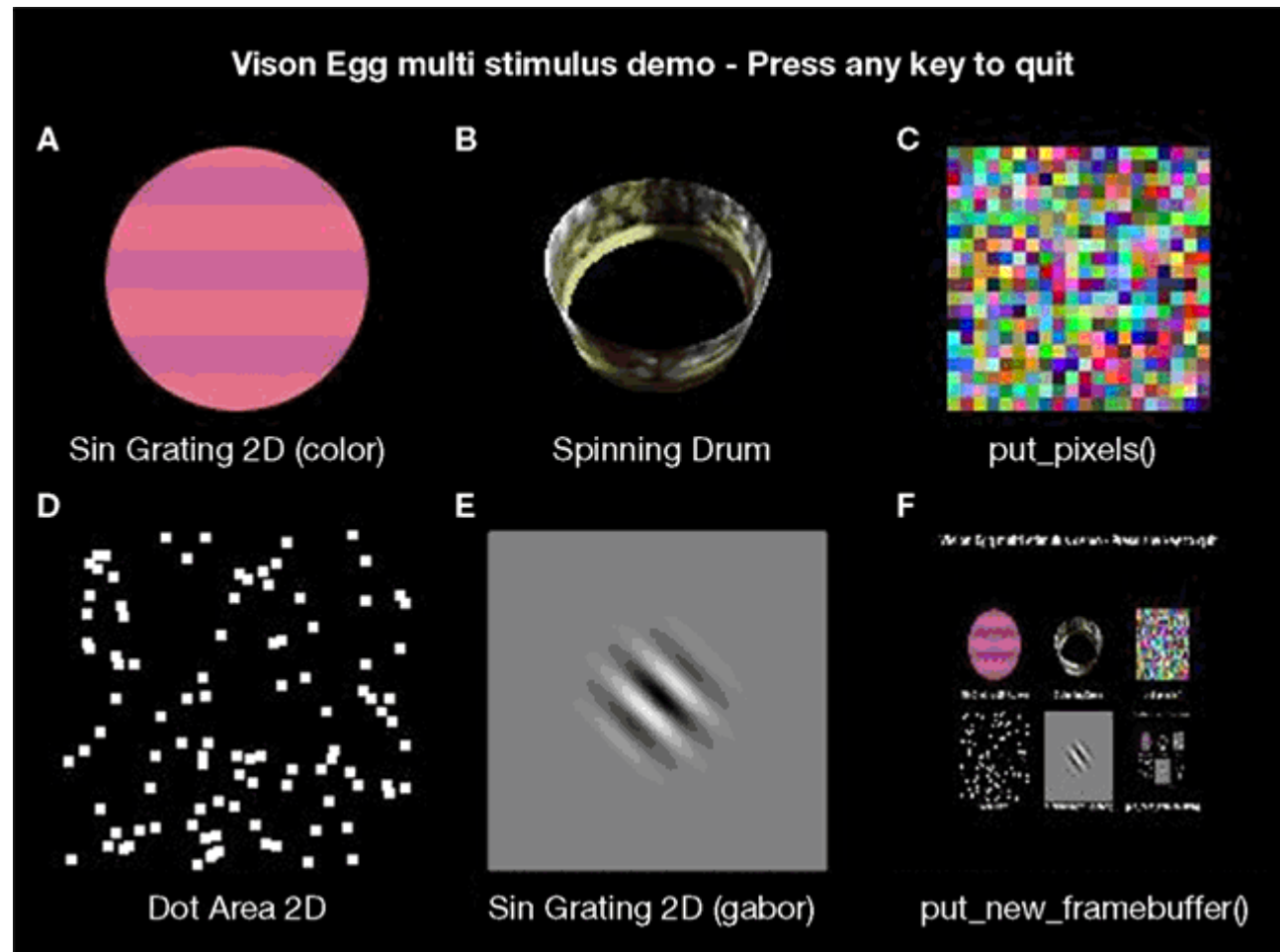
Py in Neuroscience : data collection and analysis

e.g. ACQ4



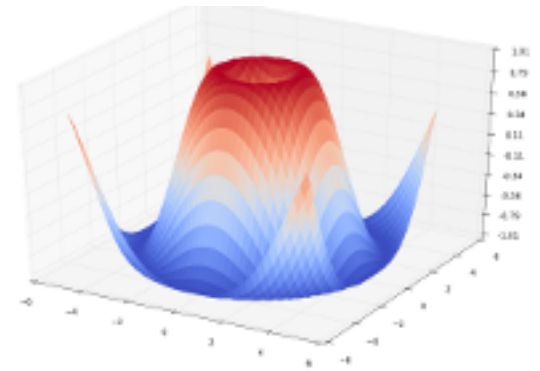
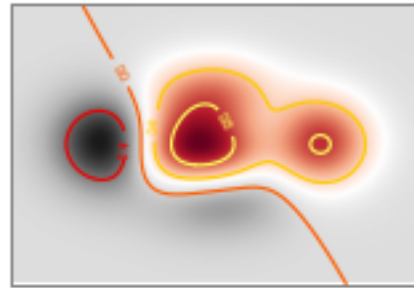
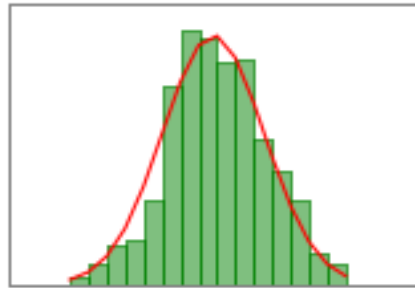
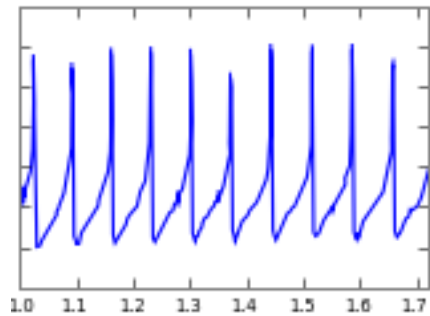
Py in Neuroscience : stimulus generation

e.g. Vision EGG, or PsychoPy



Py in Neuroscience : visualization

e.g. matplotlib library



Getting started : Python installation

- Debian + Ubuntu Linux

```
apt-get install python-numpy python-scipy python-matplotlib \  
ipython
```

- Windows, Mac OS X (distributions for package handling)

- Enthought Python : <https://www.enthought.com/>

- Anaconda from Continuum Analytics : <https://www.continuum.io/downloads>

- Python(x,y) <http://python-xy.github.io/>

- Alternative for Mac OS X : Install Fink, then

```
fink install scipy-core-py25 scipy-py25 matplotlib-py25 ipython-py25
```

Getting started : interpreters and IDEs

- **ipython**

- An interactive shell for with enhanced introspection, code highlighting and tab completion

- **Jython**

- Another Python interpreter, written in Java instead of C

- **IronPython**

- a python implementation for the .NET framework
- integrates nicely with other .NET languages

- **Spyder** : Scientific PYthon Development EnviRonment

- **IPython Notebook**

- interactive shell in the browser
- combines code execution, rich text, mathematics, plots and rich media

Spyder screenshot



The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script named `Interpolation.py` with the following code:

```
1 """
2 Interpolation of an II-D curve
3 From the SciPy Cookbook
4 """
5
6 from numpy import arange, cos, linspace, pi, sin, random
7 from scipy.interpolate import splprep, splev
8
9 # make ascending spiral in 3-space
10 t=linspace(0,1.75*2*pi,100)
11
12 x = sin(t)
13 y = cos(t)
14 z = t
15
```

The Variable explorer panel on the right shows the following variables:

| Name | Type | Size | Value |
|------|-------|------|--------------------|
| e | float | 1 | 2.7182818284590451 |
| pi | float | 1 | 3.1415926535897931 |

The Object inspector panel shows the `array(...)` function from the `numpy.core.multiarray` module. The source is set to `Console` and the object is `array`. The description reads: "Function of numpy.core.multiarray module. array(object, dtype=None, copy=True, order=None, subok=False, ndmin=0). Create an array." The Parameters section lists:

- object**: array_like. An array, any object exposing the array interface, an object whose `__array__` method returns an array, or any (nested) sequence.
- dtype**: data-type, optional. The desired data-type for the array. If not given, then the type will be determined as the minimum type required to hold

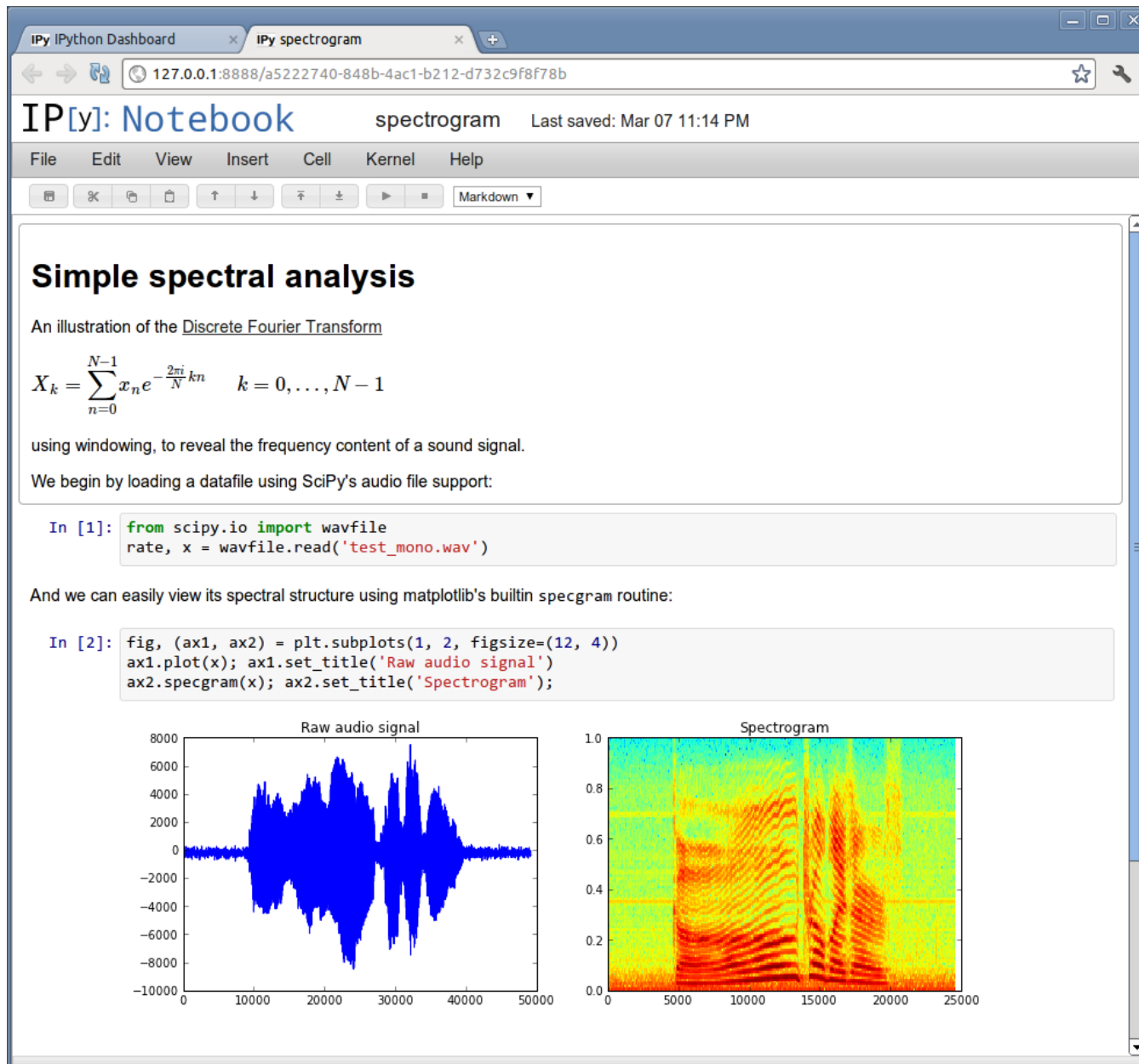
The Console panel shows the IPython 0.10.1 prompt and the following output:

```
IPython 0.10.1 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object'. ?object also works, ?? prints more.

Welcome to pylab, a matplotlib-based Python environment.
For more information, type 'help(pylab)'.

In [1]:
```

The status bar at the bottom indicates: Permissions: RW | End-of-lines: LF | Encoding: UTF-8-GUESSED | Line: 7 | Column: 1



The screenshot shows an IPython Notebook window titled "IPy spectrogram". The browser address bar shows the URL "127.0.0.1:8888/a5222740-848b-4ac1-b212-d732c9f8f78b". The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for undo, redo, copy, paste, and execution. The main content area displays the following text and code:

Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#)

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

using windowing, to reveal the frequency content of a sound signal.

We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

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

```
In [2]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```

The output shows two plots side-by-side. The left plot, titled "Raw audio signal", displays a blue waveform with an amplitude range from -10000 to 8000 and a time axis from 0 to 50000. The right plot, titled "Spectrogram", is a 2D heatmap showing frequency content over time, with a vertical axis from 0.0 to 1.0 and a horizontal axis from 0 to 25000.

Executing Python programs

- Python programs can be run either interactively or as scripts stored in a file
- The interpreter is started by calling **python**

```
mgraupe@thinkpadx1:~> python
Python 2.7.10 (default, Oct 14 2015, 16:09:02)
Type "help", "copyright", "credits" or "license"
for more information.
>>> print 'Hello world!'
Hello world!
>>> x = 3
>>> print x+5
8
```

- Scripts are supplied as arguments to the interpreter

```
mgraupe@thinkpadx1:~> python hello_world.py
Hello world!
```

- **python -i** gives an interactive prompt after the script

Python scripts

- The default extension for Python files is **.py**
- Scripts start with the interpreter they should use

```
#!/usr/bin/env python  
print 'Hello world!'
```

- Optionally, you can specify the file encoding in line 2

```
#!/usr/bin/env python  
# * coding: utf8 *  
print 'Total: 42 €'
```

- Scripts have to be executable : **chmod u+x <file>**
- Execute scripts as standalone programs

```
mgraupe@thinkpadx1:~> ./hello_world.py  
Hello world!
```

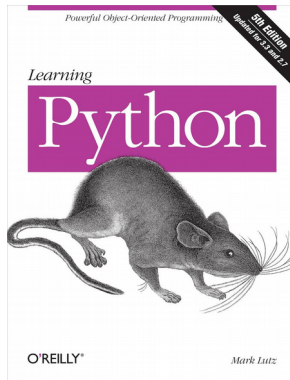
Online Resources - General

- The Python documentation index :
<https://docs.python.org/2.7/>
- Python library reference :
<https://docs.python.org/2.7/library/>
- Dive into Python
<http://www.diveintopython.net/>
- Activestate Python cookbook :
<http://aspn.activestate.com/ASPN/Cookbook/Python>
- The Python Tutorial :
<https://docs.python.org/2/tutorial/index.html>
- Tentative Numpy Tutorial :
<http://www.time.mk/trajkovski/teaching/imi/2010-fall/NumPy/Tentative%20NumPy%20Tutorial%20-.html>
- Scipy Reference :
<http://docs.scipy.org/doc/scipy/reference/genindex.html>

Online Resources - Neuroscience

- Front Neuroinform 2015 – *Python in Neuroscience* :
<http://journal.frontiersin.org/article/10.3389/fninf.2015.00011/full>
- BCCN/FACETS Student Workshop - *Using Python for Computational*
<http://neuralensemble.org/cookbook/wiki/FacetsPythonCourse2008>
- BCCN course - *Advanced Scientific Programming in Python* :
<https://python.g-node.org/wiki/schedule>
- Brian simulator :
<http://briansimulator.org/>

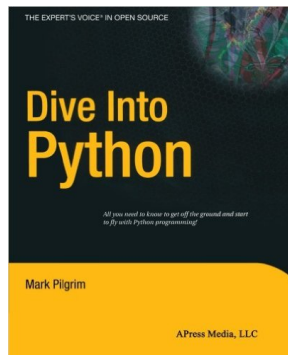
Books



- Learning Python, 5th Edition

Mark Lutz

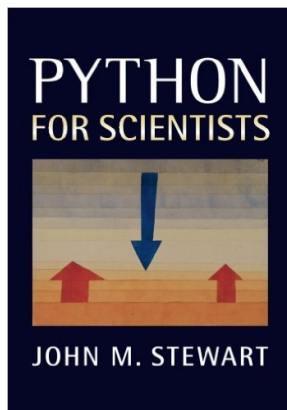
ISBN : 978-1-4493-5573-9



- Dive Into Python (3)

Mark Pilgrim

ISBN: 978-1590593561 (978-1430224150)



- Python for Scientists

John M. Stewart

ISBN: 978-1107686427