



Neural Data Science with **Python**

L1 : Introduction to Python : first steps

Michael Graupner

SPPIN – Saint-Pères Institute for the Neurosciences

Université Paris Cité, CNRS

How do YOU deal with data ?

I give you a 2d array of numbers : measurements and time points ($N=1000$).

time	0.1	0.2	0.3	0.4	0.5	0.6	...	99.8	99.9	100.0
measurment	3.2	4.3	3.8	4.5	3.7	5.1	...	8.3	8.1	9.0

- I would like to know the mean and the standard deviation of the measurements.
- I would like to see the data displayed, i.e., plotted as measurement vs time.

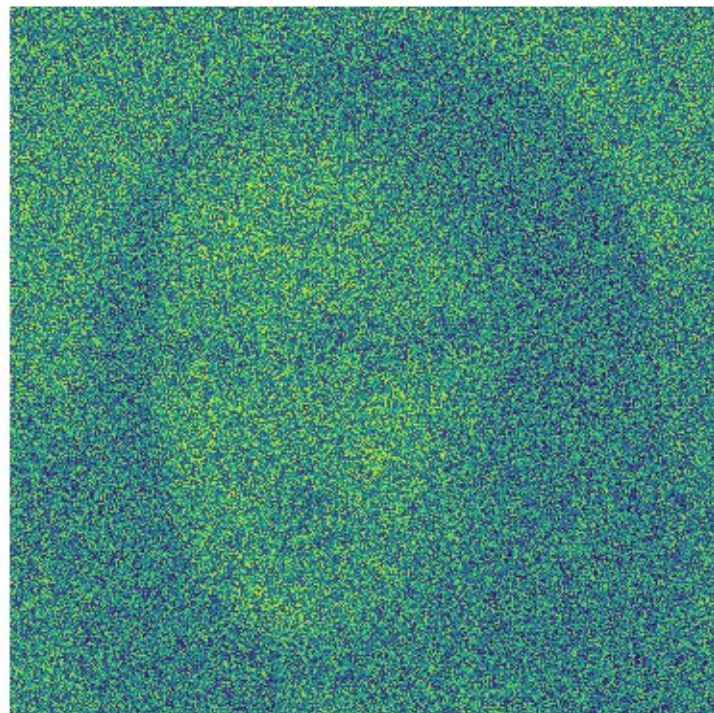
→ **How would you do that today ?**

942	509	688	1093	703	416	664	762	664	822	457	400	614	648	243	218	852	459	524	660	
158	503	226	1016	322	632	168	419	1015	772	893	774	765	237	161	575	351	244	806	584	
692	817	305	168	139	693	507	1121	868	181	943	310	622	345	391	609	1060	1058	1002	401	
1050	826	399	1002	1031	140	310	750	394	924	321	425	524	225	428	633	439	686	587	413	
787	1103	1012	157	839	632	917	256	677	631	419	1014	239	398	951	223	1042	637	577	843	
781	512	994	343	1084	941	343	632	234	518	850	1021	367	310	726	773	672	493	914	506	
516	240	575	790	159	657	983	587	933	170	1010	1106	601	868	872	337	1032	533	452	963	
464	902	456	976	943	279	797	1008	341	679	585	1011	569	584	775	740	505	497	690	944	
179	295	928	549	593	850	375	1068	343	436	535	171	931	557	382	1083	781	536	902	781	
1119	648	590	617	221	622	149	978	487	154	763	1088	942	941	1032	411	642	912	199	958	...
443	150	619	614	747	452	415	502	240	767	160	1066	907	978	1043	382	240	563	939	904	
278	684	1080	582	291	580	1100	701	957	287	650	453	573	730	714	412	311	407	1048	275	
813	520	997	277	308	1069	1082	235	230	1031	972	658	1033	805	869	344	228	586	144	665	
1027	540	1067	293	851	269	274	435	1069	869	941	208	163	1079	887	974	265	733	223	1009	
610	546	628	732	218	184	827	412	942	492	624	348	950	676	471	1114	515	355	821	264	
621	328	734	905	340	875	1056	1102	1038	130	338	918	945	174	931	1005	884	1054	158	767	
511	209	1108	378	1064	478	203	739	868	615	541	641	697	571	920	430	908	299	666	638	
728	151	217	404	481	884	231	705	301	908	295	1091	865	169	806	757	1098	913	1082	942	
221	715	474	329	560	738	1044	672	531	412	224	284	1127	198	602	464	393	385	293	608	
778	1045	1005	689	328	718	395	202	656	1040	489	756	459	582	1010	951	442	731	540	1023	

⋮

942	509	688	1093	703	416	664	762	664	822	457	400	614	648	243	218	852	459	524	660
158	503	226	1016	322	632	168	419	1015	772	893	774	765	237	161	575	351	244	806	584
692	817	305	168	139	693	507	1121	868	181	943	310	622	345	391	609	1060	1058	1002	401
1050	826	399	1002	1031	140	310	750	394	924	321	425								
787	1103	1012	157	839	632	917	256	677	631	419	1014								
781	512	994	343	1084	941	343	632	234	518	850	1021								
516	240	575	790	159	657	983	587	933	170	1010	1106								
464	902	456	976	943	279	797	1008	341	679	585	1011								
179	295	928	549	593	850	375	1068	343	436	535	171								
1119	648	590	617	221	622	149	978	487	154	763	1088								
443	150	619	614	747	452	415	502	240	767	160	1066								
278	684	1080	582	291	580	1100	701	957	287	650	453								
813	520	997	277	308	1069	1082	235	230	1031	972	658								
1027	540	1067	293	851	269	274	435	1069	869	941	208								
610	546	628	732	218	184	827	412	942	492	624	348								
621	328	734	905	340	875	1056	1102	1038	130	338	918								
511	209	1108	378	1064	478	203	739	868	615	541	641								
728	151	217	404	481	884	231	705	301	908	295	1091								
221	715	474	329	560	738	1044	672	531	412	224	284								
778	1045	1005	689	328	718	395	202	656	1040	489	756								

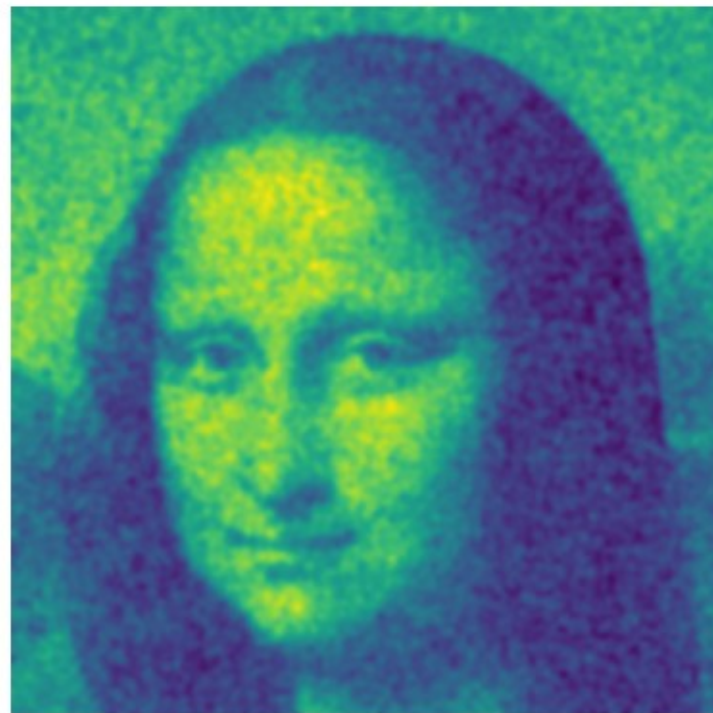
⋮



2D plot

942	509	688	1093	703	416	664	762	664	822	457	400	614	648	243	218	852	459	524	660
158	503	226	1016	322	632	168	419	1015	772	893	774	765	237	161	575	351	244	806	584
692	817	305	168	139	693	507	1121	868	181	943	310	622	345	391	609	1060	1058	1002	401
1050	826	399	1002	1031	140	310	750	394	924	321	425								
787	1103	1012	157	839	632	917	256	677	631	419	1014								
781	512	994	343	1084	941	343	632	234	518	850	1021								
516	240	575	790	159	657	983	587	933	170	1010	1106								
464	902	456	976	943	279	797	1008	341	679	585	1011								
179	295	928	549	593	850	375	1068	343	436	535	171								
1119	648	590	617	221	622	149	978	487	154	763	1088								
443	150	619	614	747	452	415	502	240	767	160	1066								
278	684	1080	582	291	580	1100	701	957	287	650	453								
813	520	997	277	308	1069	1082	235	230	1031	972	658								
1027	540	1067	293	851	269	274	435	1069	869	941	208								
610	546	628	732	218	184	827	412	942	492	624	348								
621	328	734	905	340	875	1056	1102	1038	130	338	918								
511	209	1108	378	1064	478	203	739	868	615	541	641								
728	151	217	404	481	884	231	705	301	908	295	1091								
221	715	474	329	560	738	1044	672	531	412	224	284								
778	1045	1005	689	328	718	395	202	656	1040	489	756								

⋮



smoothing + 2D plot

Python code for the above operations

Python code

```
img = plt.imread('image-noise.tif')
```

→ read image

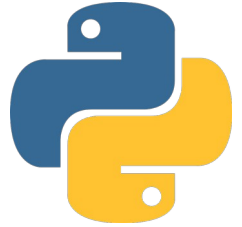
```
imgNew = gaussian_filter(img, sigma=10)
```

→ apply Gaussian filter

```
ax.imshow(imgNew)
```

→ plot/display image

What is Python ?



- modern programming language (since 1991)
- interpreted language (no compilation necessary)
- emphasis is put on the readability of the code
- concepts can be expressed in less lines compared to C/C++ or Java
- extensive libraries available
- build-in visualization

Python - modern programming language

Most popular programming languages in 2021



Rank	Language	Type	Score
1	Python	Web, Desktop, Mobile	100.0
2	Java	Web, Mobile, Desktop	95.4
3	C	Mobile, Desktop, Embedded	94.7
4	C++	Mobile, Desktop, Embedded	92.4
5	JavaScript	Web	88.1
6	C#	Web, Mobile, Desktop, Embedded	82.4
7	R	Desktop	81.7
8	Go	Web, Desktop	77.7
9	HTML	Web	75.4
10	Swift	Mobile, Desktop	70.4
11	Arduino	Embedded	68.4
12	Matlab	Desktop	68.3

[Source : IEEE Spectrum]

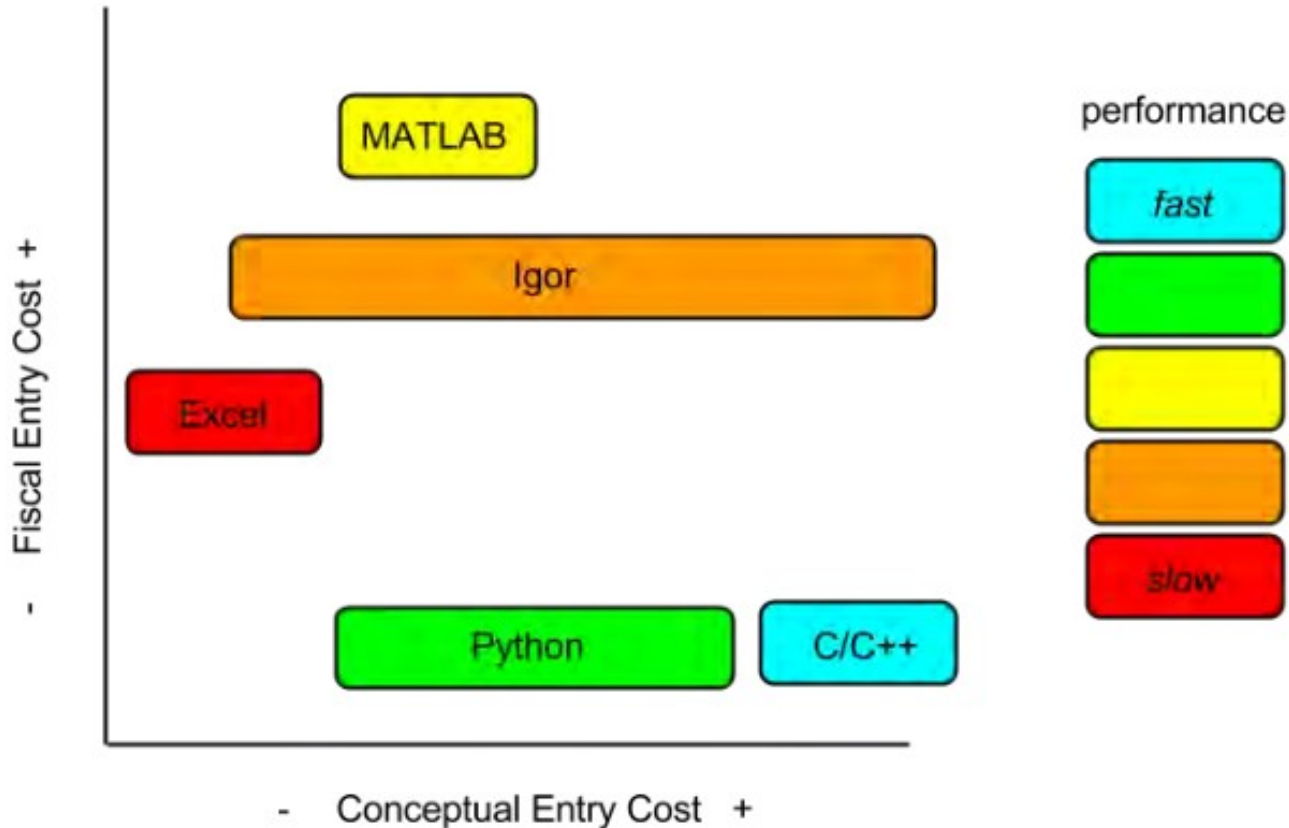
Clear and readable syntax → easy to learn

```
In [1]: 1 # import modules
        2 import numpy as np
        3
        4 # function declaration
        5 def update_values(x):
        6     return x+1
        7
        8 x = 1
        9 if x>0:
       10     print('Hello World!')
       11     x = update_values(x)
       12
       13 print(x)
```

Hello World!

2

Python - free and easy to learn



Extensive standard and third-party libraries

- **wxPython** : graphical toolbox library for GUI development
- **SymPy** : library for symbolic mathematics : can do algebraic evaluations, differentiation, expansions, complex numbers, etc.
- **Pygame** : library for 2D game development
- **Twisted** : major tool for development of network applications
- **OpenCV** : library for extensive computer vision applications

Python modules for Neuroscience applications

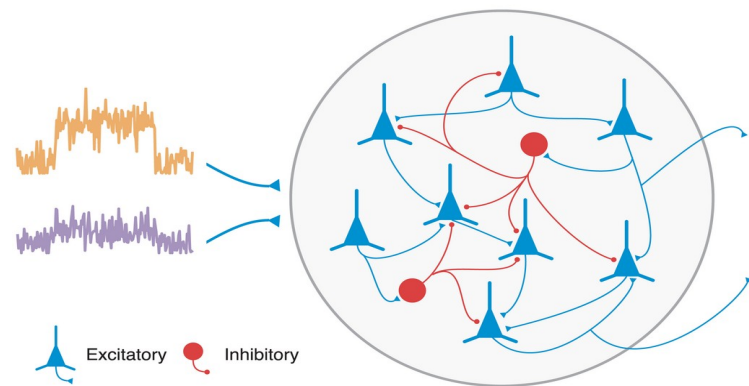
- simulators and simulator interfaces
- data collection and analysis
- sharing, re-use, storage and databasing of data and models
- stimulus generation
- parameter search and optimization
- visualization
- VLSI (very-large-scale integration) hardware interfacing
- machine learning : GPU-based computing

Python in Neuroscience : network simulator

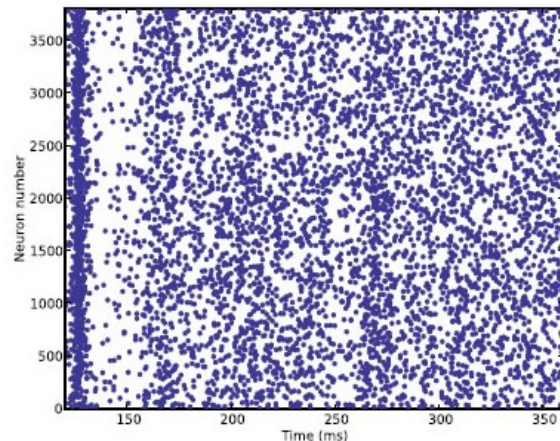
BRIAN

spiking neural network simulator

randomly connected,
recurrent network of
excitatory and inhibitory
neurons



```
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()
```

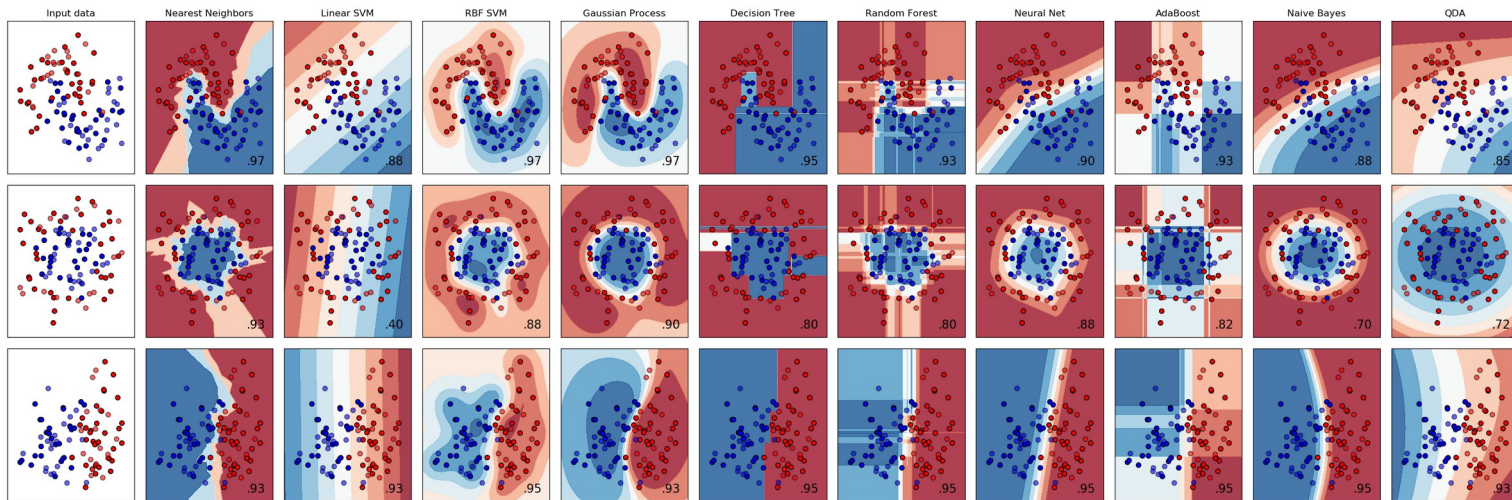


Python in Neuroscience : machine learning



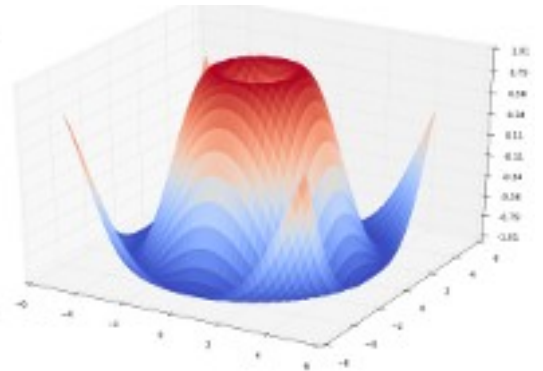
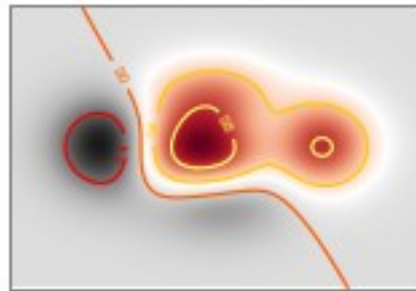
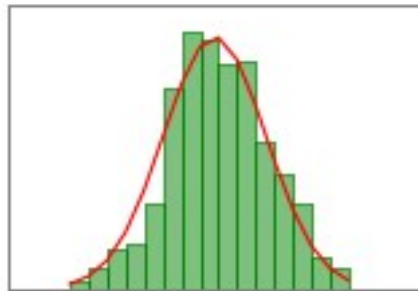
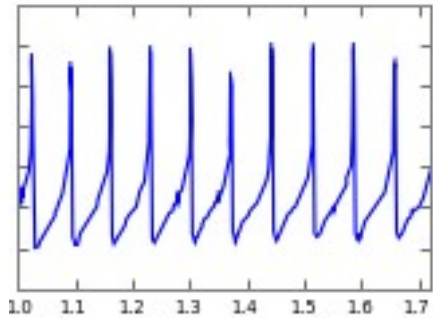
Machine Learning in Python. Simple and efficient tools for data mining and data analysis

e.g. classification
using several
classifiers



Python in Neuroscience : visualization

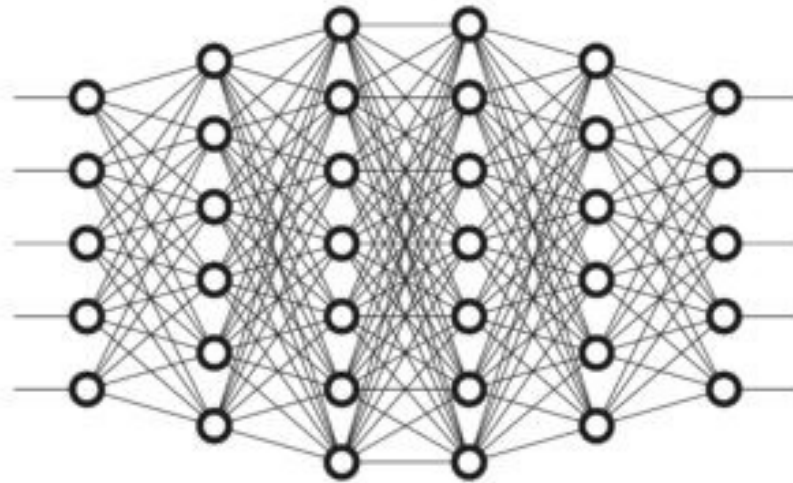
e.g. matplotlib library



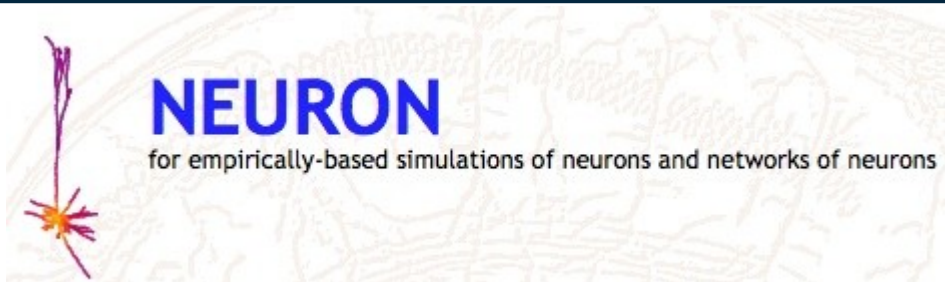
Python in Neuroscience : deep learning/networks



simulate multi-layer networks for deep-learning applications

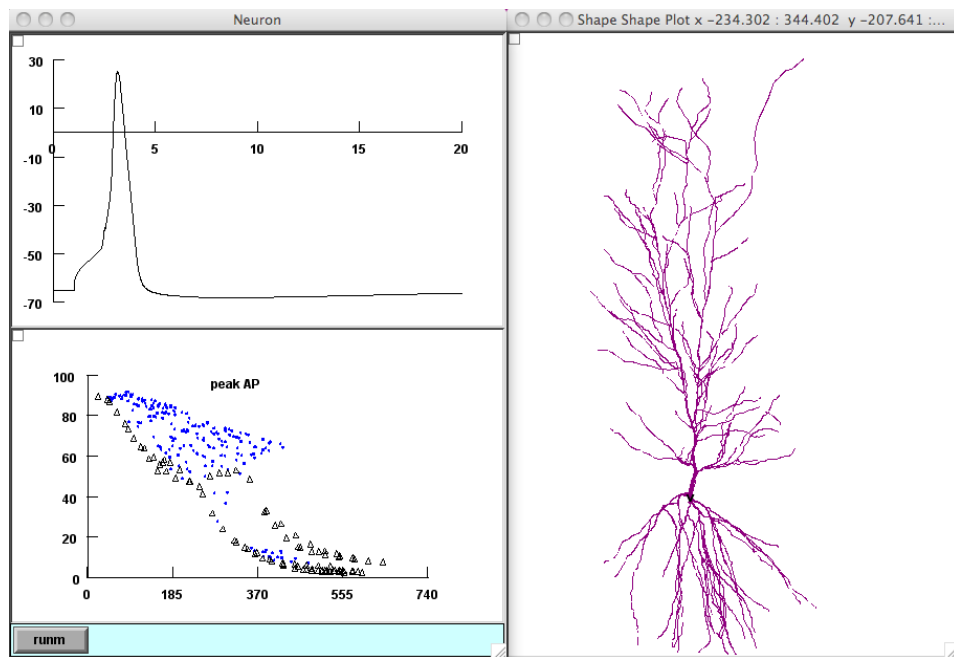


Python in Neuroscience : single neuron simulator



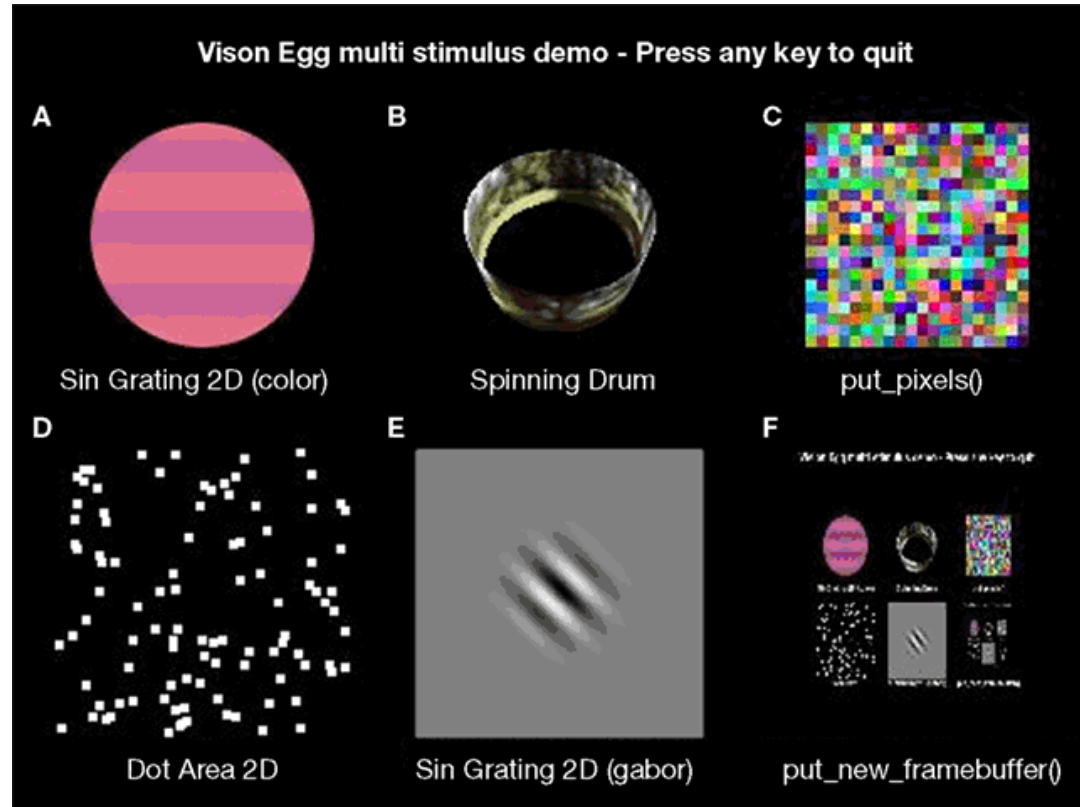
Python interface for NEURON

compartmental model of a
single neurons simulating
the propagation of the
membrane potential



Python in Neuroscience : stimulus generation

e.g. Vision EGG, or PsychoPy



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)

- **Anaconda** from Continuum Analytics : <https://www.continuum.io/downloads>
- Enthought Python : <https://www.enthought.com/>
- Python(x,y) : <http://python-xy.github.io/>

- Mac OS X : Install Fink, then

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

Getting started : interpreter and IDEs

- **ipython**
 - command line interpreter : interactive shell for enhanced introspection, code highlighting and tab completion
- **Jupyter Notebook**
 - command line interpreter in the browser
 - combines code execution, rich text, and visualizations
- **Spyder** (IDE): Scientific PYthon Development EnviRonment
- **PyCharm** (IDE): code development environment

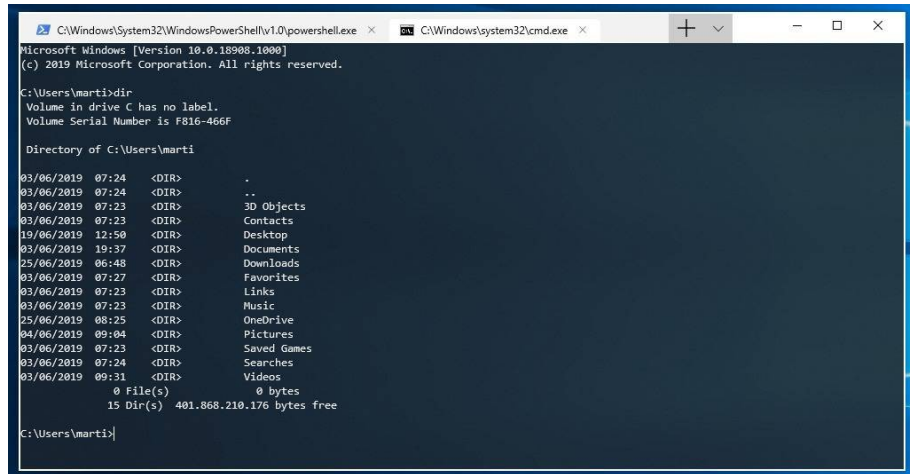
IDE ... Integrated Development Environment

Everything starts with the Terminal

■ What is a Terminal ?

- Terminals, also known as command lines, consoles or shell
- allow to accomplish tasks on a computer without the use of graphical user interfaces (GUIs)
- simple text commands are used to accomplish tasks

Windows terminal (command line, Powershell)

A screenshot of a Windows PowerShell terminal window. The title bar shows 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe'. The window content displays the output of the 'dir' command in the C:\Users\marti directory, showing a list of files and folders with their dates, times, and sizes. The prompt is 'C:\Users\marti>'.

```
Microsoft Windows [Version 10.0.18908.1000]
(c) 2019 Microsoft Corporation. All rights reserved.

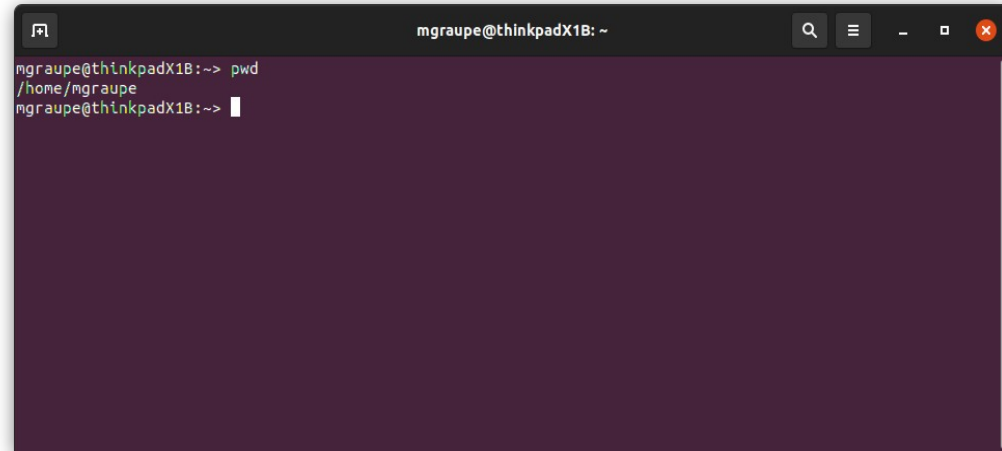
C:\Users\marti>dir
Volume in drive C has no label.
Volume Serial Number is F816-466F

Directory of C:\Users\marti

03/06/2019  07:24    <DIR>        .
03/06/2019  07:24    <DIR>        ..
03/06/2019  07:23    <DIR>        3D Objects
03/06/2019  07:23    <DIR>        Contacts
19/06/2019  12:50    <DIR>        Desktop
03/06/2019  19:37    <DIR>        Documents
25/06/2019  06:48    <DIR>        Downloads
03/06/2019  07:27    <DIR>        Favorites
03/06/2019  07:23    <DIR>        Links
03/06/2019  07:23    <DIR>        Music
25/06/2019  08:25    <DIR>        OneDrive
04/06/2019  09:04    <DIR>        Pictures
03/06/2019  07:23    <DIR>        Saved Games
03/06/2019  07:24    <DIR>        Searches
03/06/2019  09:31    <DIR>        Videos
               0 File(s)            0 bytes
               15 Dir(s)  401.868.210.176 bytes free

C:\Users\marti>
```

Linux terminal

A screenshot of a Linux terminal window. The title bar shows 'mgraupe@thinkpadX1B: ~'. The window content displays the output of the 'pwd' command, showing the current directory as '/home/mgraupe'. The prompt is 'mgraupe@thinkpadX1B:~>'.

```
mgraupe@thinkpadX1B: ~
mgraupe@thinkpadX1B:~> pwd
/home/mgraupe
mgraupe@thinkpadX1B:~>
```

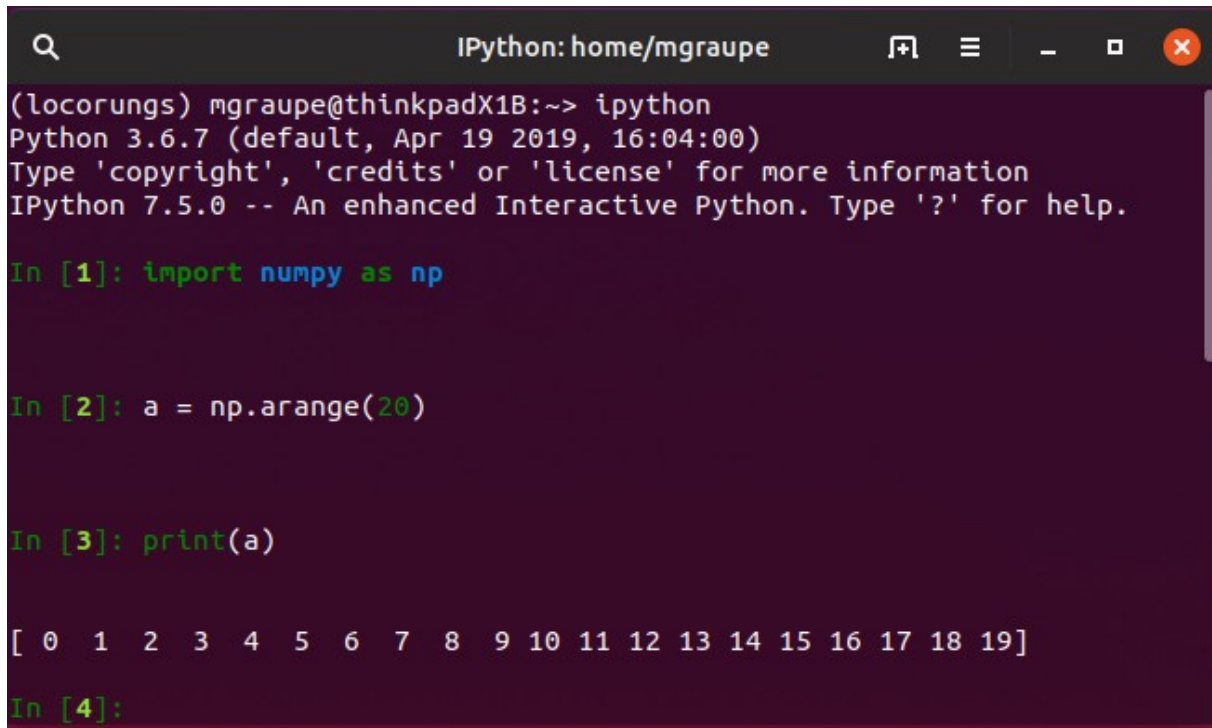
iPython

IP[y]: IPython
Interactive Computing

- Started by typing and executing (by pressing *enter*) **ipython** in the *terminal* application

```
mgraupe@thinkpadx1:~$ ipython
```

- useful for short explorations
- tab completion!

A screenshot of a terminal window titled "IPython: home/mgraupe". The window shows the execution of the 'ipython' command, which starts a Python 3.6.7 environment. The user enters three commands: 'import numpy as np', 'a = np.arange(20)', and 'print(a)'. The output of the third command is a list of integers from 0 to 19. The fourth prompt 'In [4]:' is visible at the bottom.

```
(locorungs) mgraupe@thinkpadX1B:~> ipython
Python 3.6.7 (default, Apr 19 2019, 16:04:00)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.5.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy as np

In [2]: a = np.arange(20)

In [3]: print(a)

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]

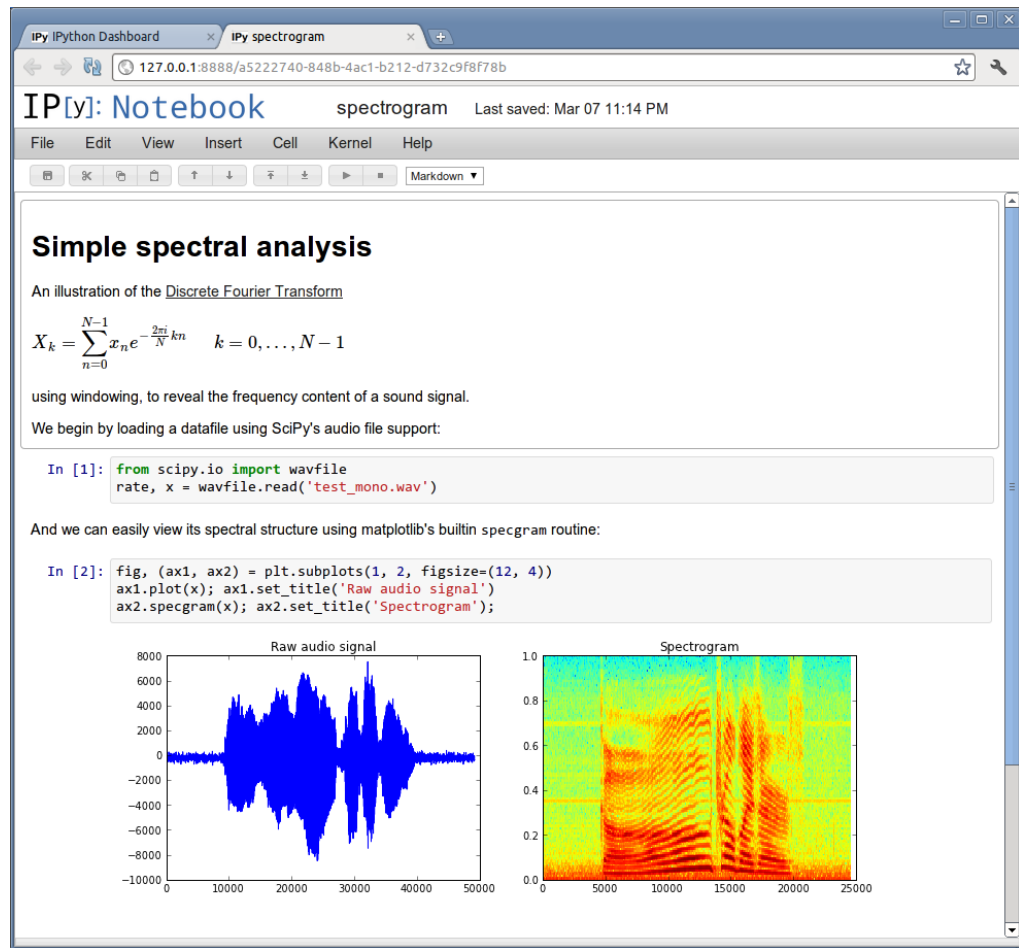
In [4]:
```

Jupyter Notebook

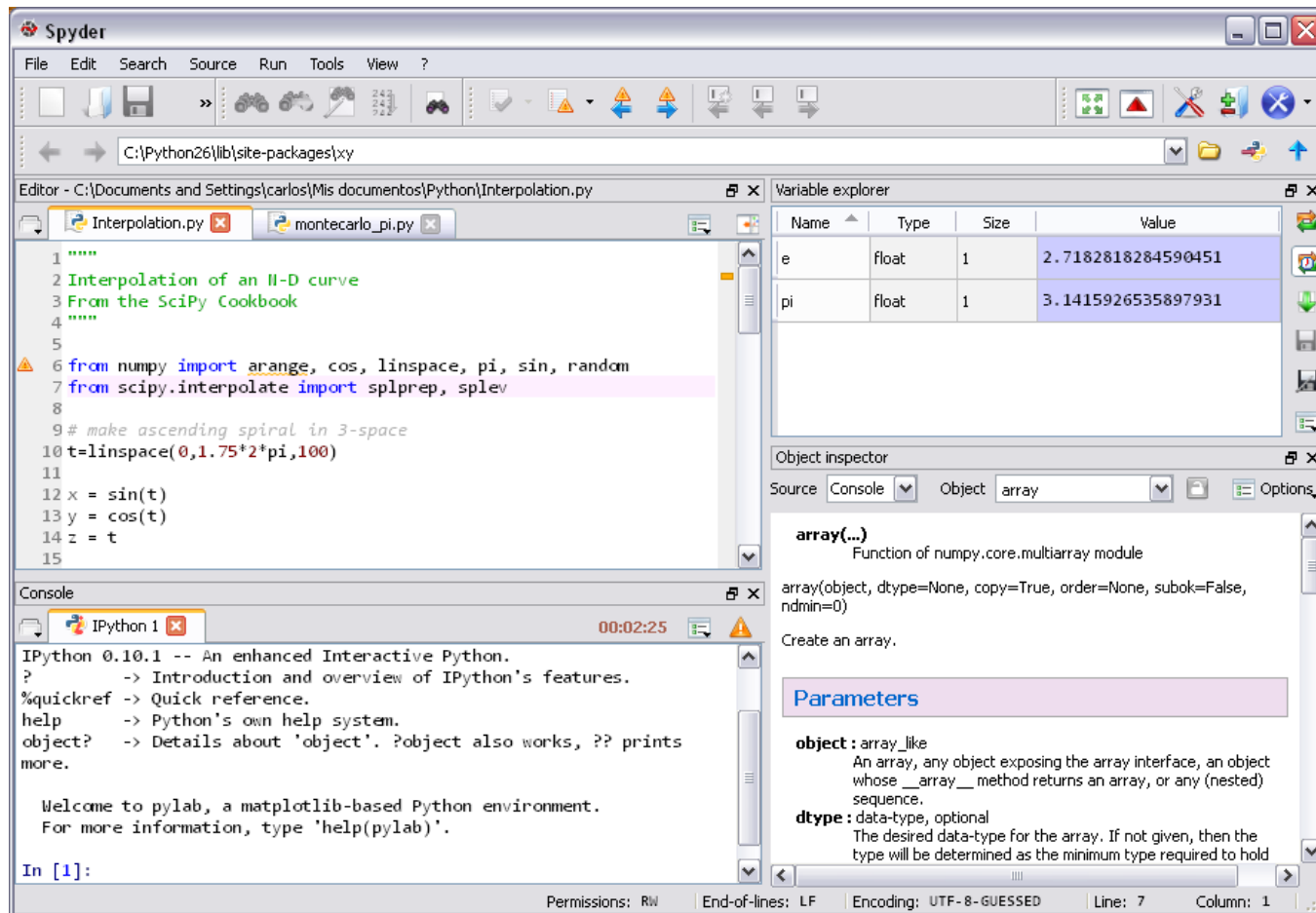
- Started by typing and executing (by pressing *enter*) **jupyter-notebook** in the *terminal* application :

```
mgraupe@thinkpadx1:~$ jupyter-notebook
```

- launched and accessed in a browser (firefox, chrome, safari ...) window



Spyder



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

```
1 """
2 Interpolation of an H-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 console window shows the IPython 0.10.1 prompt and the following text:

```
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 variable explorer window shows the following variables:

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

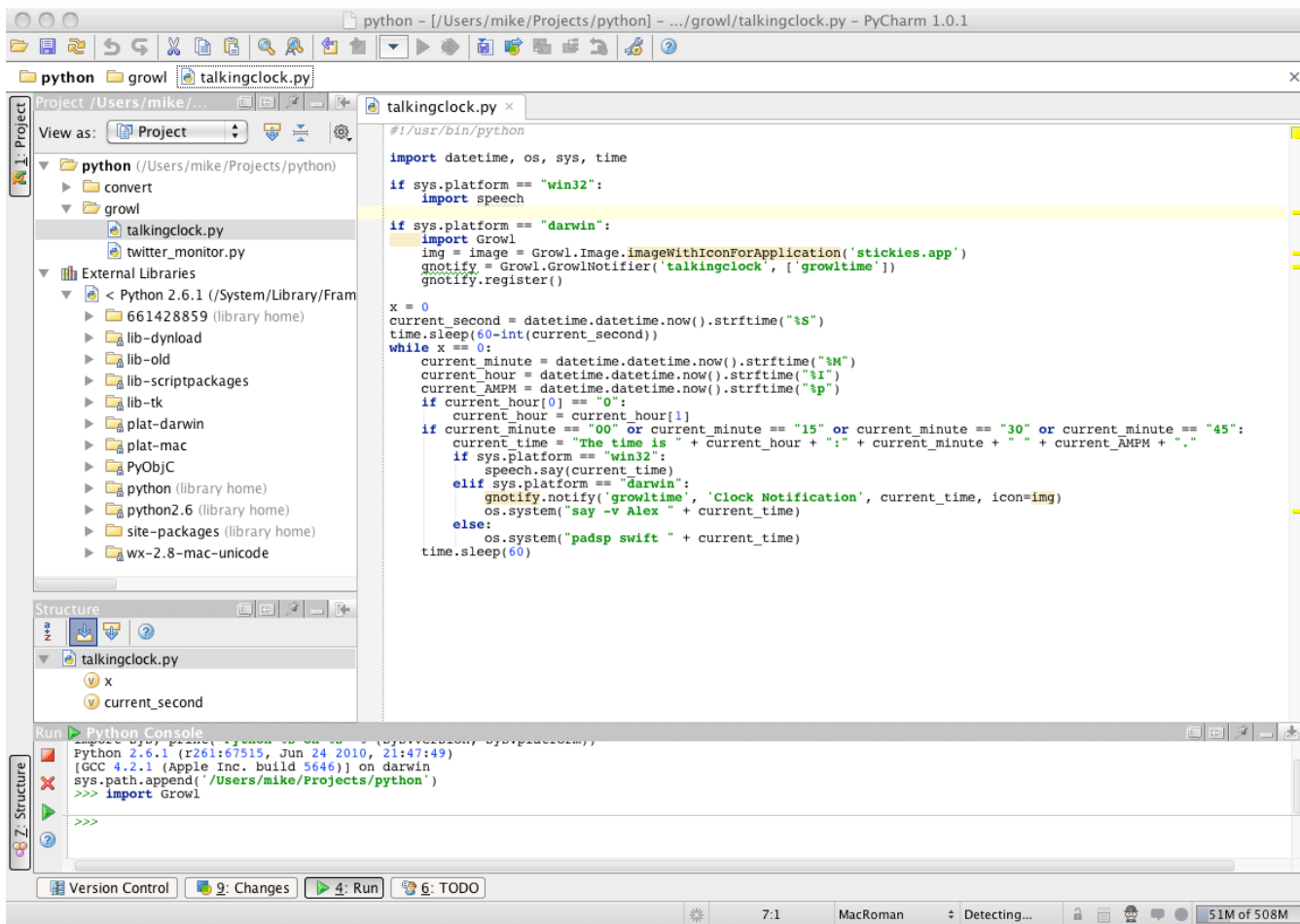
The object inspector window shows the `array(...)` function of the `numpy.core.multiarray` module. The object is `array`.

Parameters:

- 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



PyCharm



Executing Python programs

- Python programs can be run either interactively or as scripts stored in a file
- An interpreter is started by calling **ipython** (or plain **python**, or **jupyter-notebook**)

```
mgraupe@atp:~$ ipython3
Python 3.5.7 (default, Apr  4 2019, 12:02:34)
Type "copyright", "credits" or "license" for more information.

In [1]: print('Hello World!')
Hello World!
In [2]: x = 3
In [3]: print(x+5)
8
In [4]: exit
mgraupe@atp:~$
```

- Scripts are supplied as arguments to the interpreter

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


Online resources : introductions and references

- **W3Schools - Python Tutorials :**

<https://www.w3schools.com/python/default.asp>

- **The Python documentation index :**

<https://docs.python.org/3.6/>

- **Python library reference :**

<https://docs.python.org/3.6/library/>

- **Dive into Python :**

<http://histo.ucsf.edu/BMS270/diveintopython3-r802.pdf>

- **Activestate Python [popular Python recipes] :**

<http://code.activestate.com/recipes/langs/python/>

- **Python tutorial :**

<https://docs.python.org/3.6/tutorial/index.html>

- **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 : general

- a simple Google search :
 - use the keyword “python”
 - specify your operating system (*window, linux, mac*) for package installation, importing queries
 - use the best matching terminology for code questions
 - common sites for useful help : *stackoverflow, askubuntu, github*

Online resources : DataCamp

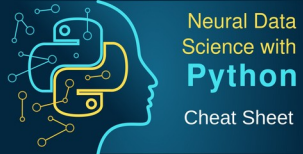
- an interactive to learn, practice and develop further your programming skills
- ideal to to dive into a programming language for starters (videos, extensive help, hints, small assignments with direct feedback)
- learn and extend your programming skills at your own pace
- a commercial website, you have access through this course :
<https://learn.datacamp.com/>



Online resources : Neuroscience

- Front Neuroinform 2015 – *Python in Neuroscience*
<http://journal.frontiersin.org/article/10.3389/fninf.2015.00011/full>
- BCCN cours - *Advanced Scientific Programming in Python* :
<https://python.g-node.org/wiki/schedule>
- Brian simulator :
<http://briansimulator.org/>

General reference for the course : Cheat Sheet



Neural Data
Science with
Python
Cheat Sheet

Find all course information on <https://github.com/mgdraupe/DataSciPy2022>

> Variables and Data Types

Variable Assignment

```
[In[0]]: x=5           #Comments are preceded by '#'
[In[1]]: X
5
```

Calculatoin with Variables

```
[In[2]]: x+2           #Sum of two variables
7
[In[3]]: x-2           #Subtraction of two variables
3
[In[4]]: x*2           #Multiplication of two variables
10
[In[5]]: x**2          #Exponentiation of a variable
25
[In[6]]: x%2           #Remainder of a variable
1
[In[7]]: x/float(2)    #Division of a variable
2.5
```

Types and Type Conversation

```
str('5', '3.7', 'True', 'False') #Variables to strings
int(1)                             #Variables to integers
5, 3, 1, 0
float(1)                           #Variables to floats
5, 3.7, 1.0, 0.0
bool(1)                            #Variables to booleans
True, True, True, False
```

> Libraries



Import Libraries

```
[In[8]]: import numpy           #Imports the numpy library
[In[9]]: import numpy as np     #Imports the numpy library and calls it 'np'
[In[10]]: np.mean(...)         #Using the numpy function 'mean()'
[In[11]]: import matplotlib.pyplot as plt #Imports the matplotlib plotting library and calls 'plt'
```

Selective import

```
[In[12]]: from math import pi   #Imports the specific function 'pi' from the math library
```

> Asking for Help

```
[In[13]]: help(str)
[In[14]]: help(np.mean)
```

> NumPy Arrays

```
[In[15]]: my_list = [2,4,6,8,10]
[In[16]]: my_array = np.array(my_list)
[In[17]]: my_array2 = np.arange(10)
[In[18]]: my_array3 = np.zeros(10)
[In[19]]: my_2darray = np.array([[1,2,3],[4,5,6]])
```

*Creates 1-dimensional array
Creates 1d vector of 10 elements, from 0 to 9
Creates array of length 3 containing zeros
2-dimensional array, 2 rows, 3 columns*

Selecting NumPy Array Elements

Index starts at 0

```
Subset
[In[20]]: my_array[2]      #Select item at index 2
6

Slice
[In[21]]: my_array[1:4]    #Select items at indices 1 to 3
array([4, 6, 8])
[In[22]]: my_array[2:]    #Half items starting from index 2
array([6, 8, 10])
[In[23]]: my_array[-1:]   #Half but the last element
array([2, 4, 6, 8])

Subset 2D NumPy arrays
[In[24]]: my_2darray[1:,1:] #Select all items ('') of row with index 1, my_2darray[rows, columns]
array([[2, 5]])
```

NumPy Array Operations

```
[In[25]]: my_array > 4
array([False, False, True, True, True], dtype=bool)
[In[26]]: my_array + 2
array([6, 8, 10, 12, 14])
[In[27]]: my_array * np.array([1,2,3,4,5])
array([13, 6, 9, 12, 15])
```

NumPy Array Functions

```
[In[28]]: np.shape(my_2darray) #Get the dimensions of the array
(2, 3)
[In[29]]: len(my_array)        #Get the length - number of rows for 2D - of an array
5
[In[30]]: np.mean(my_array)    #Mean of the array
6.8
```

> Lists

See also NumPy Arrays

```
[In[31]]: a = 5
[In[32]]: my_list = ['my', True, 2, 3.14, a]
[In[33]]: my_list2 = [[1, 'a', True], [2, 'b', False]]
```

Selecting List Elements

Index starts at 0

```
Subset and Slice - see NumPy Arrays

Subset Lists of Lists
[In[34]]: my_list2[1][0]
[In[35]]: my_list2[1][1:2]

my_list[0][0] == 'my' #True
```

List Operations

```
[In[36]]: my_list + my_list
['my', True, 2, 3.14, 5, 'my', True, 2, 3.14, 5]
[In[37]]: my_list * 2
['my', True, 2, 3.14, 5, 'my', True, 2, 3.14, 5]
```

List Methods

```
[In[38]]: my_list.index(a) #Get the index of an item
[In[39]]: my_list.count(a) #Count an item
[In[40]]: my_list.append('') #Append an item at a time
[In[41]]: my_list.remove('') #Remove a specific item
[In[42]]: my_list.pop()    #Remove an item at specific index
[In[43]]: my_list.reverse() #Reverse the list
[In[44]]: my_list.insert(0, '') #Insert item at specified index
[In[45]]: my_list.sort()  #Sort the list
```

> Control structures

Compare expressions

```
[In[46]]: x == y           #Returns True if x is equal to y
[In[47]]: x != y           #True if x is NOT equal to y
[In[48]]: x > y             #True if x is larger than y
[In[49]]: x <= y            #True if x is smaller or equal to y
```

Conditional statements

```
[In[50]]: if x>4:           #Condition
    print('x is larger than 4') #Indented line after the colon ':' is executed if condition is True
else:                       #else condition
    print('x is smaller than 4') #If none of the above was True
```

for and while Loops

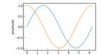
```
[In[51]]: a = np.array([2,6,8,10])
[In[52]]: for i in range(len(a)): #Iterates over sequence; use index of arrays/lists in the loop
    print(i, a[i])              #Indented line
                                #Prints index and array element
0 2
...

[In[53]]: x = 0
[In[54]]: while x < 3:          #While loop is executed as long as condition is True
    print(x)                   #Indented line
    x+=1                       #Multiple indented line are all executed if condition is True
0
...
```

> Plotting

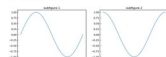
Simple plot

```
[In[55]]: import matplotlib.pyplot as plt #Import matplotlib plotting library
[In[56]]: x = np.linspace(0, 2*np.pi, 1000) #Create uniformly spaced array between 0 and 2*pi
[In[57]]: y1 = np.sin(x); y2 = np.cos(x)   #Calculate sin and cos function of each array element
[In[58]]: plt.plot(x, y1)                 #Plot y1 over x, the sin function
[In[59]]: plt.plot(x, y2)                 #Plot y2 over x, the cos function
[In[60]]: ax1 = fig.add_subplot(2,2,1)    #Add a-axis label
[In[61]]: ax1.set_title('sin')            #Add y-axis label
[In[62]]: plt.show()                     #Show figure
```



Multipanel figure

```
[In[63]]: fig = plt.figure(figsize=(15,5)) #Creates figure of specified size in inches
[In[64]]: ax0 = fig.add_subplot(2,2,1)    #Add0 subplot, format: (rows, columns, number of subplot)
[In[65]]: ax0.set_title('subplot figure 1', size=16) #Add subplot title
[In[66]]: ax0.plot(x, y1)                  #Plot y1 over x, the sin function
[In[67]]: ax1 = fig.add_subplot(2,2,2)    #Add1 subplot, index is here 2 for 2nd subplot!
[In[68]]: ax1.set_title('subplot figure 2', size=16) #Add subplot title
[In[69]]: ax1.plot(x, y2)                  #Plot y2 over x, the cos function
[In[70]]: plt.show()                     #Show figure
```



> Python IDEs (Integrated Development Environment)



ANACONDA
Leading open data science platform powered by Python

SPYDER
Free IDE that is installed with Anaconda

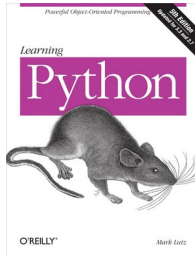
JUPYTER
Create and share documents with live code, runs in browser, start in terminal with \$ jupyter-notebook

IPYTHON
Powerful interactive Python shell start in terminal with \$ ipython

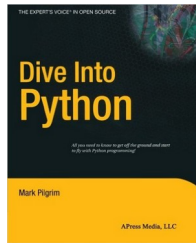
PyCharm
Powerful IDE. Free Community Edition

- Contains basic Python concepts and information
- Reference for often used code in class

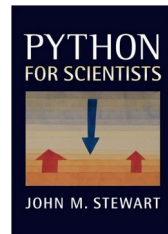
General Python books



- Learning Python, 5th Edition
Mark Lutz
ISBN : 978-1-4493-5573-9

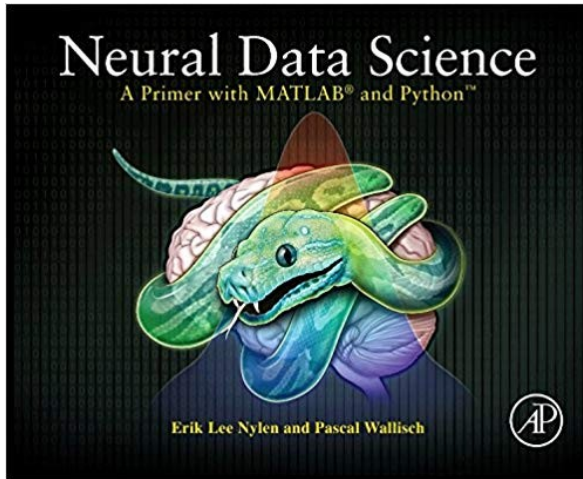


- Dive Into Python (3)
Mark Pilgrim
ISBN: 978-1590593561 (978-1430224150)



- Python for Scientists
John M. Stewart
ISBN: 978-1107686427

Neuroscience specific book




- Neural Data Science
A primer with Matlab and Python
Erik Lee Nylen (Author), Pascal Wallisch (Author)
ISBN-10: 9780128040430

Workflow of the course lecture


- 1) **All course material** (.pdf file of lecture; .ipynb for tutorial; .ipynb for homework assignment) can be accessed on github (code repository site) :

<https://github.com/mgraupe/DataSciPy2021>

- 2) **Visit course website** : Launch the browser ( “Navigateur Web”) and navigate to the link above

- 3) **Download lecture** : Click on lecture link and hit the **Download** button (file will be downloaded automatically to the downloads – Téléchargements – folder); annotate lecture PDF

Workflow of the course tutorial

- 1) **Tutorial material** : can be accessed on github, launch the browser (Navigateur Web) and got to:
<https://github.com/mgraupe/DataSciPy2021>
- 2) **Save tutorial file** : Click on the tutorial link, hit the **Raw** button (the raw file content will be displayed), save (Enregister sous ...) the raw file to your computer
Attention : make sure that the file ending remains **.ipynb**
- 3) **Launch jupyter-notebook** : Start the **terminal** application ( “Emulateur de Terminal”) on your computer, launch the notebook environment by typing and executing **\$ jupyter-notebook** , the jupyter environment starts up in the browser in the directory in which it was started
- 4) **Load the jupyter-notebook file** : In the jupyter environment, you first see the directory and file structure on your computer (relative to the directory in which jupyter was started), **navigate** to the downloaded **.ipynb** file and click on it to launch it
- 5) **Start editing**
- 6) **You can follow the tutorial progress/coding of the lecturer on Deepnote.**
allows you to copy code into your own Jupyter notebook.