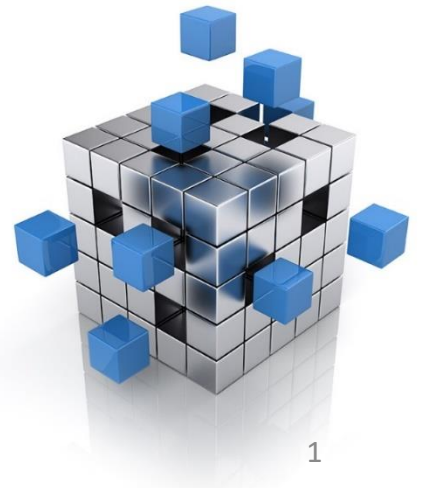


# Premier pas

## Introduction au traitement des données



# Sommaire

- Vue d'ensemble des logiciel d'analyse de données
- Pourquoi Python ?
- Python comment ?
- Un serpent peut en cacher un autre
- Introduction à Jupyter Notebook
- Concrètement
- Les conditions
- Les listes
- Les boucles

# Vue d'ensemble des logiciels

- Les logiciel Bureautique :
  - Excel
- Les logiciels généralistes à interface graphique :
  - SAS
  - SPSS
  - Tanagra
  - Statistica
  - RapidMiner

# Vue d'ensemble des logiciels

- Les logiciels généraliste en ligne de commande :
  - R
  - Python
- Les logiciels spécialisés :
  - Requête SQL : Oracle SQL Developer, PL/SQL Developer
  - Requête Hadoop : Hive, Pig
  - Visualisation : D3JS

# Pourquoi Python ?

## Ses caractéristiques



- Langage de programmation généraliste avec un écosystème scientifique.
- Open source et donc en constante évolution.
- Forte communauté et donc facilité à trouver de l'aide sur les forums.
- Cadre unifié pour mettre en œuvre tout le processus de l'analyse données.
- Langage interprété et non compilable -> On peut traiter le programme ligne par ligne.

# Python comment ?

- **Python en mode interactif**

Peut s'utiliser comme une calculatrice (très améliorée).

On lance python dans un shell.

Le prompteur `>>>` python apparaît.

Puis on "discute" avec python en tapant des commandes ou instructions.

# Python comment ?

- **Les scripts**

Un programme est une séquence d'instructions.

Dans le cas d'un programme en langage Python, on parle souvent de **script Python**.

Un script se présente sous la forme d'un fichier texte avec l'extension **.py**

# Les bibliothèques sous python

- Une des grandes forces du langage Python réside dans le nombre important de bibliothèques logicielles externes disponibles.
- Une bibliothèque est un ensemble de fonctions. Celles-ci sont regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.
- Une bibliothèques doit être importé pour être utilisée



# Les bibliothèques sous python

- NumPy

- Module pour la manipulation de matrices, tableaux multidimensionnels et fonctions mathématiques opérant sur ces tableaux.

- SciPy

- Module pour l'optimisation, l'algèbre linéaire, les statistiques, traitement d'image, ...

- Pandas

- Module permettant la manipulation et l'analyse de données, notamment des données numériques et des série temporelle -> DataFrame, Panels, ...

# Les bibliothèques sous python

- Matplotlib

- Permet de tracer et visualiser des données sous forme de graph

- Scikit-learn

- Destiné à l'apprentissage automatique -> Forêts aléatoires, régression logistiques, classement, ...

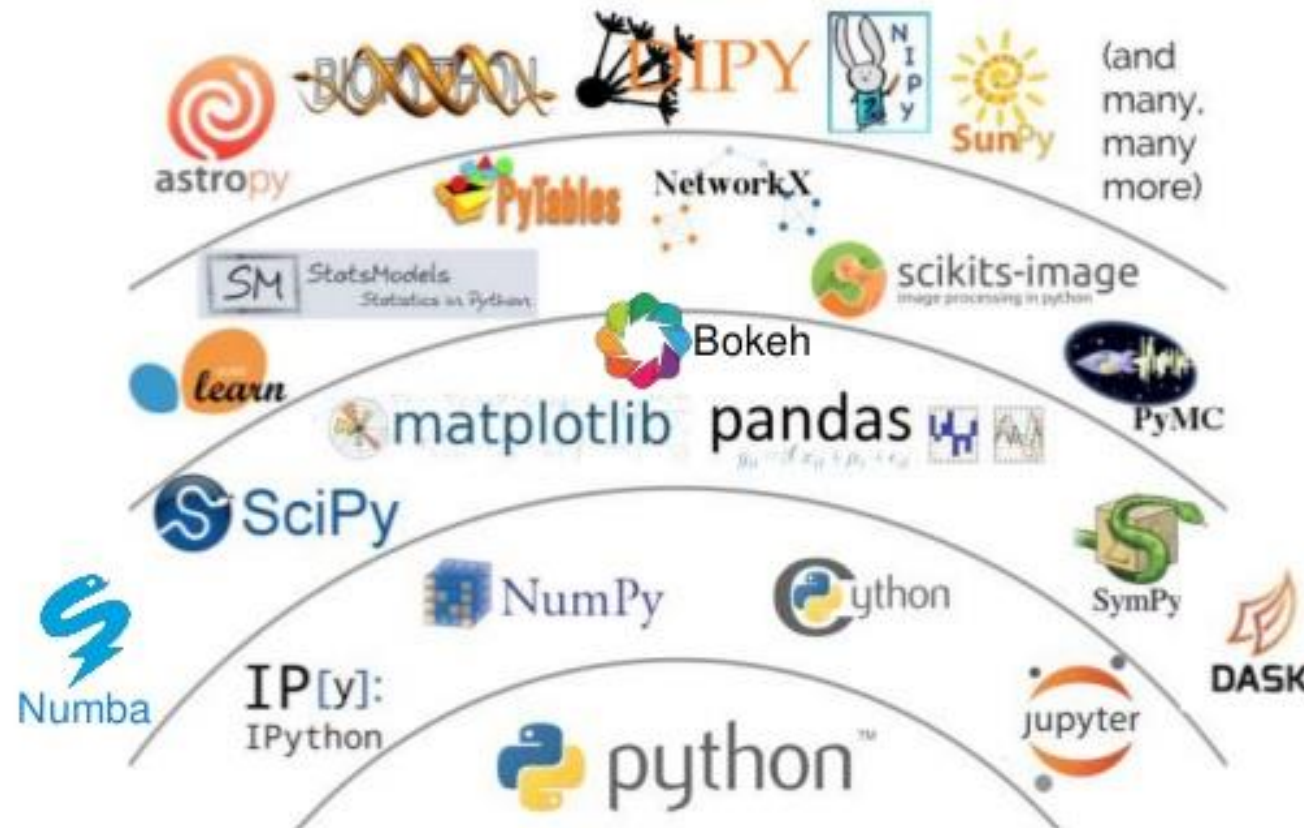
- Tensorflow

- Apprentissage automatique pour intelligence Artificielle

# Un serpent peut en cacher un autre



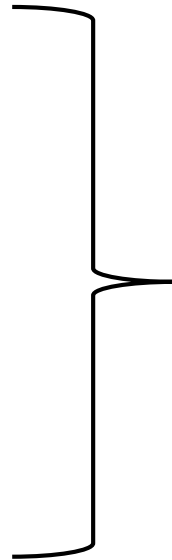
## ANACONDA®



# Pourquoi python ?

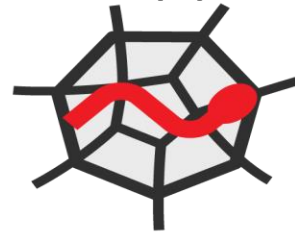
## Les librairie sous python

- SciPy
- NumPy
- Pandas
- Scikit-learn
- Tensorflow



# Pourquoi Python ?

- Interface de développement :



# SPYDER

The Scientific Python Development Environment

The screenshot displays the Spyder Python IDE interface. The main window is divided into several panes:

- Project Explorer:** Shows the file structure of the current project, including folders like 'Data', 'Scripts', and 'Tests'.
- Code Editor:** Contains a Python script with the following code:

```
6
7 import pylab
8 from numpy import cos, linspace, pi, sin, random
9 from scipy.interpolate import splprep, splev
10
11 #%% Generate data for analysis
12
13 # Make ascending spiral in 3-space
14 t = linspace(0, 1.75 * 2 * pi, 100)
15
16 x = sin(t)
17 y = cos(t)
18 z = t
19
20 # Add noise
21 x += random.normal(scale=0.1, size=x.shape)
22 y += random.normal(scale=0.1, size=y.shape)
23 z += random.normal(scale=0.1, size=z.shape)
24
25
26 #%% Perform calculations
27
28 # Spline parameters
29 smoothness = 3.0 # Smoothness parameter
30 k_param = 2 # Spline order
31 nests = -1 # Estimate of number of knots needed (-1 = maximal)
32
33 # Find the knot points
34 knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)
35
36 # Evaluate spline, including interpolated points
37 xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)
38
39
40 #%% Plot results
41
42 # TODO: Rewrite to avoid code smell
43 pylab.subplot(2, 2, 1)
44 data = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
45 fit = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')
46 pylab.legend()
47 pylab.xlabel('x')
48 pylab.ylabel('y')
```
- Variable Explorer:** Displays a table of variables and their values. The table has columns for Name, Type, Size, and Value.
- Python Console:** Shows the execution of the code, including the generation of a 3D plot and a 2D polar plot.

Name	Type	Size	Value
bars	container.BarContainer	20	BarContainer object of matplotlib.cont...
df	DataFrame	(3, 2)	Column names: bools, ints
filename	str	1	C:\ProgramData\Anaconda3\lib\site-pack...
list_test	list	2	[Dataframe, Numpy array]
nrows	int	1	344
r	float64	1	7.611082589334796
radii	float64	(20,)	Min: 0.4983036630535087 Max: 9.856848974942551
region	tuple	2	(slice, slice)
rgb	float64	(45, 45, 4)	Min: 0.0 Max: 1.0
series	Series	(1,)	Series object of pandas.core.series mo...
test_none	NoneType	1	NoneType object of builtin module

The Python console shows the execution of the code, including the generation of a 3D plot and a 2D polar plot. The 3D plot shows a surface with a color gradient from blue to red. The 2D polar plot shows a series of colored segments arranged in a circular pattern.

# Introduction Jupyter Notebook



Application web qui permet aux utilisateurs d'interagir avec le code et de transformer le navigateur web en terminal interactif et en logiciel de traitement de texte en même temps.

- Programmer dans le navigateur
- Le code, les instructions et la sortie son afficher en ligne
- Utile pour écrire un code qui raconte une histoire
- Utile pour réaliser des compte rendue
- Utiliser par les scientifiques et les chercheurs

# Introduction Jupyter Notebook



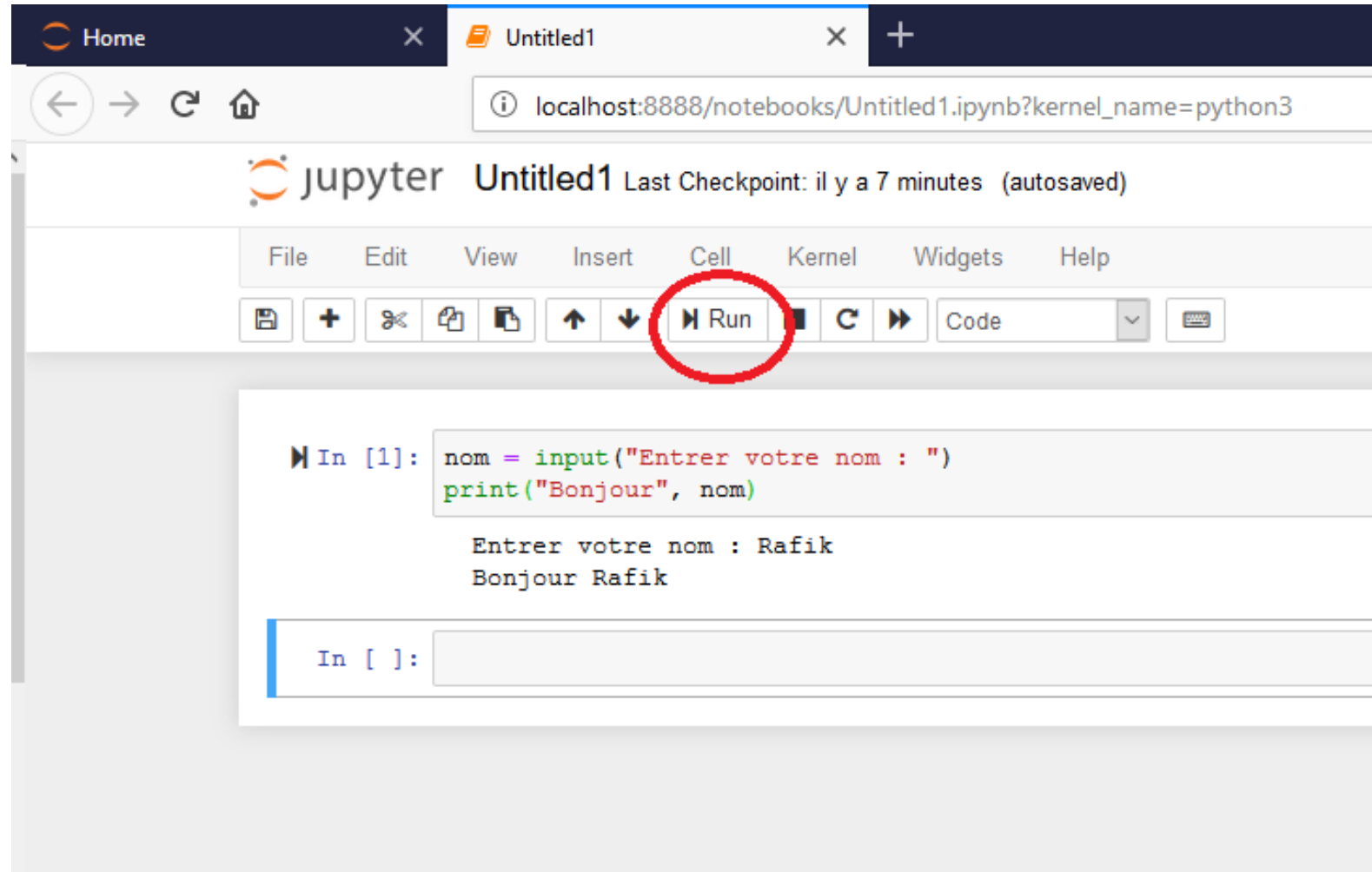
# Introduction Jupyter Notebook



A screenshot of the Jupyter Notebook web interface. The browser address bar shows 'localhost:8888/tree'. The Jupyter logo is in the top left, and 'Quit' and 'Logout' buttons are in the top right. Below the logo are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, showing a file browser. At the top of the file browser, there are buttons for 'Upload', 'New', and a refresh icon. The 'New' button is circled in red. Below these buttons is a table of files and folders. The table has columns for 'Name', 'Last Modified', and 'File size'. The files listed include '3D Objects', 'Contacts', 'Desktop', 'Documents', 'Downloads', 'Favorites', 'Links', 'Music', 'OneDrive', 'Pictures', 'Roaming', 'Saved Games', 'Searches', 'Videos', and 'Untitled.ipynb'. The 'Untitled.ipynb' file is highlighted in blue. The 'Last Modified' column shows times like 'il y a 2 ans' and 'il y a 2 heures'. The 'File size' column shows '555 B' for the 'Untitled.ipynb' file.



# Introduction Jupyter Notebook



Raccourci clavier pour  
Run

Shift



Entrée

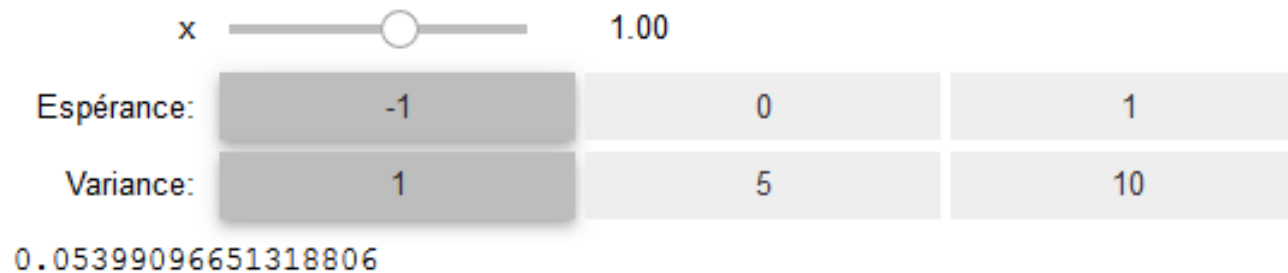
# Introduction Jupyter Notebook

## Générer un curseur

► In [36]:

```
from ipywidgets import ToggleButtons, interact
import scipy.stats

res=interact(scipy.stats.norm.pdf, x=FloatSlider(min=-8, max=8, step=1, value=1),
             loc=ToggleButtons(description='Espérance:', options=[-1, 0, 1]),
             scale=ToggleButtons(description='Variance:', options=[1, 5, 10]));
```



# Introduction Jupyter Notebook

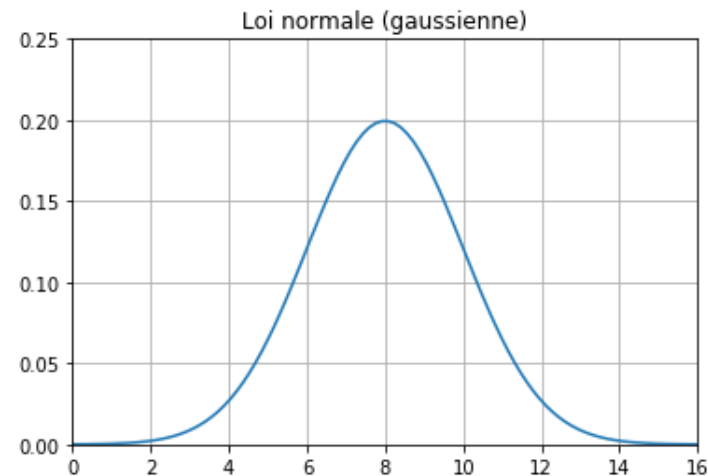
## Tracer un graph

```
In [28]: import matplotlib.pyplot as plt
import scipy.stats
import numpy as np

x = np.linspace(0, 16.0, 100)

plt.plot(x, scipy.stats.norm.pdf(x, 8, 2))

plt.grid()
plt.xlim(0, 16)
plt.ylim(0, 0.25)
plt.title('Loi normale (gaussienne)')
plt.savefig("normal_distribution.png")
plt.show()
```



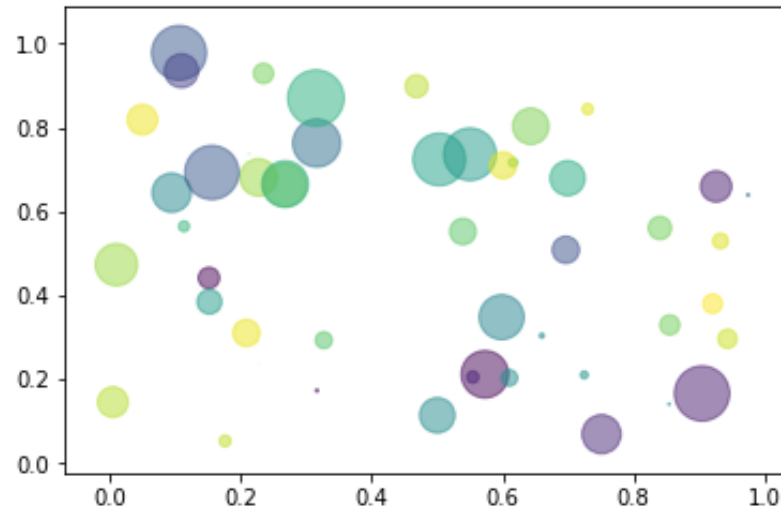
# Introduction Jupyter Notebook

## Tracer un graph

```
In [50]: import numpy as np
import matplotlib.pyplot as plt

N=50
x=np.random.rand(N)
y=np.random.rand(N)
colors=np.random.rand(N)
area=np.pi * (15 * np.random.rand(N))**2

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



# Introduction Jupyter Notebook

## Créer un DataFrame

```
In [55]: import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(10, 5))
df.head()
```

Out[55]:

	0	1	2	3	4
0	0.101293	-1.158496	0.339858	-0.617471	-1.256263
1	-0.031837	0.210922	0.788922	1.415832	0.170070
2	1.025123	-1.475479	0.882929	0.030915	0.401047
3	0.311811	-0.376678	0.117401	-0.275404	0.660441
4	0.838328	-1.531435	-0.314659	0.910114	-1.912346

# Introduction Jupyter Notebook

## Les fonctions Magic

2 types de fonctions Magic :

- Les fonctions Magic, préfixées par un seul symbole %, agissent uniquement sur la ligne sur la quel elles se trouvent
- Les fonctions Magic, préfixées de 2 symboles %% agissent sur l'ensemble de la cellule.

# Introduction Jupyter Notebook

## Les fonctions Magic

```
In [41]: %pwd
```

```
Out[41]: 'C:\\Users\\rafik'
```

```
In [42]: %timeit range(1000)
```

```
233 ns ± 7.92 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

```
In [43]: %%timeit  
range(10)  
range(100)  
range(1000)
```

```
587 ns ± 16.9 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

# Introduction Jupyter Notebook

## Les fonctions Magic

In [24]: `%lsmagic`

Out[24]: Available line magics:

```
%alias %alias_magic %autocall %automagic %autosave %bookmark %cd %clear %cls %colors %config %connect_info %copy %ddir %debug %dhist %dirs %doctest_mode %echo %ed %edit %env %gui %hist %history %killbgscripts %ldir %less %load %load_ext %loadpy %logout %logon %logstart %logstate %logstop %ls %lsmagic %macro %magic %matplotlib %mkdir %more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %ren %rep %rerun %reset %reset_selective %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode
```

Available cell magics:

```
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javascript %%js %%latex %%markdown %%perl %%p  
run %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile
```

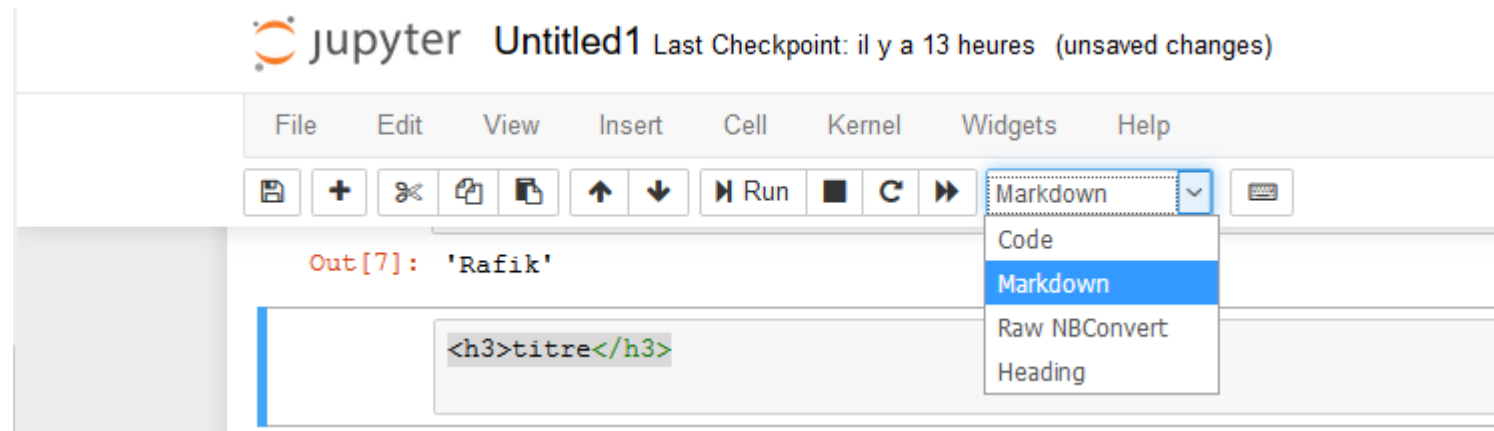
Automagic is ON, % prefix IS NOT needed for line magics.



# Introduction Jupyter Notebook

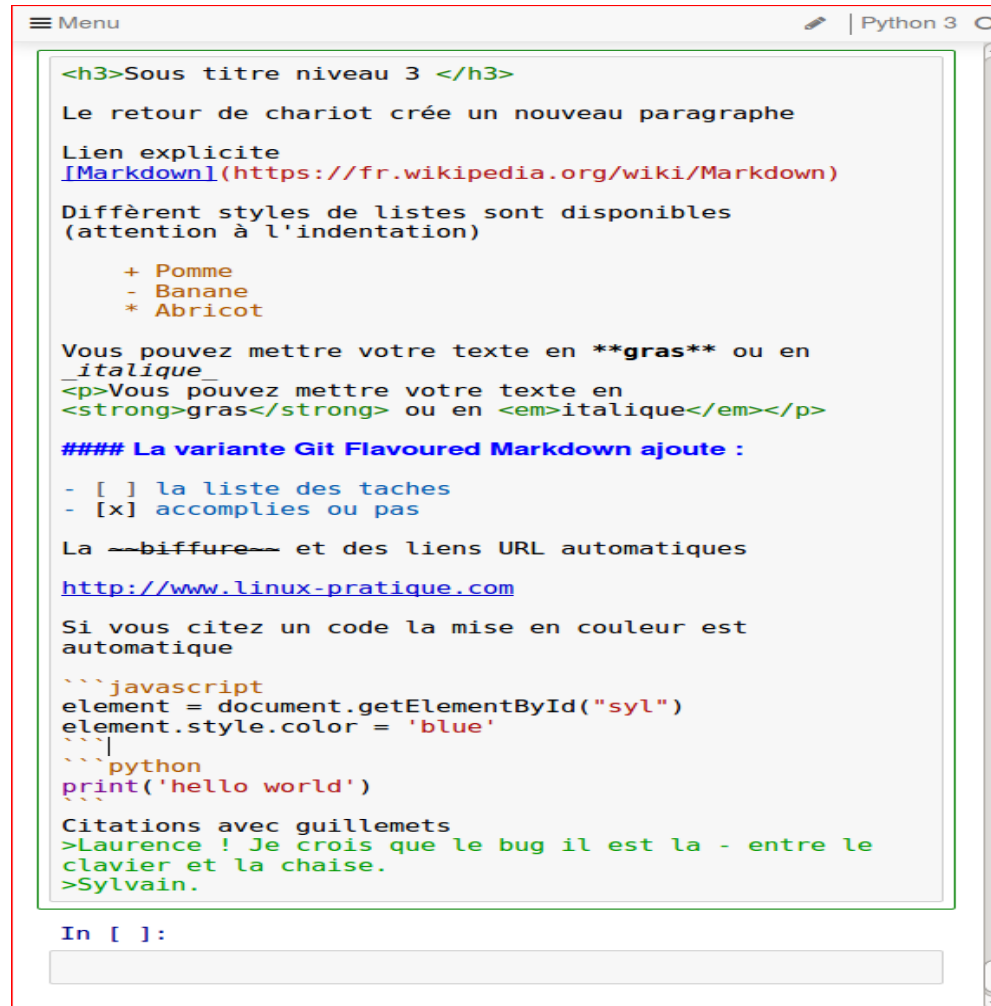
## Programmation et rédaction

- Possibilité de structurer le texte en utilisant des balise HTML/CSS, Markdown, ou LaTeX



# Introduction Jupyter Notebook

## Programmation et rédaction



The screenshot shows a Jupyter Notebook interface with a 'Menu' button and 'Python 3' selected. The main area displays raw Markdown code. The code includes a level 3 heading, a paragraph about line breaks, a link to the Wikipedia page on Markdown, a list of fruits with different list styles, instructions on bold and italic text, a section on Git Flavoured Markdown with a task list, strikethrough text, a URL, a code block with JavaScript and Python, and a citation example.

```
<h3>Sous titre niveau 3 </h3>

Le retour de chariot crée un nouveau paragraphe

Lien explicite
[Markdown](https://fr.wikipedia.org/wiki/Markdown)

Diffèrent styles de listes sont disponibles
(attention à l'indentation)

+ Pomme
- Banane
* Abricot

Vous pouvez mettre votre texte en gras ou en
italique
<p>Vous pouvez mettre votre texte en
<strong>gras</strong> ou en <em>italique</em></p>

#### La variante Git Flavoured Markdown ajoute :

- [ ] la liste des taches
- [x] accomplies ou pas

La biffure et des liens URL automatiques

http://www.linux-pratique.com

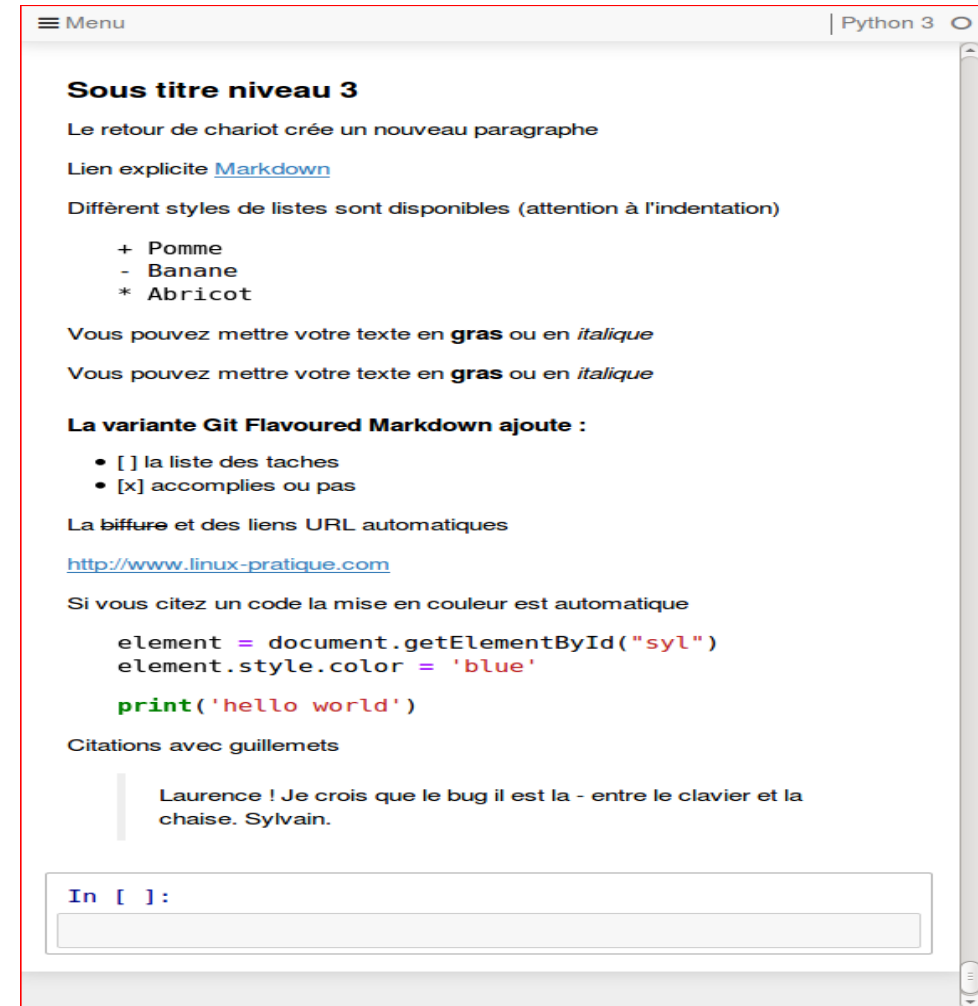
Si vous citez un code la mise en couleur est
automatique

```javascript
element = document.getElementById("syl")
element.style.color = 'blue'
```

```python
print('hello world')
```

Citations avec guillemets
>Laurence ! Je crois que le bug il est la - entre le
clavier et la chaise.
>Sylvain.

In [ ]:
```



The screenshot shows the same Jupyter Notebook interface but in rendered view. The Markdown code from the previous image is now formatted into a readable document. It features a level 3 heading, a paragraph, a link, a list, bold and italic text, a task list, strikethrough text, a URL, a code block with syntax highlighting, and a citation.

**Sous titre niveau 3**

Le retour de chariot crée un nouveau paragraphe

Lien explicite [Markdown](https://fr.wikipedia.org/wiki/Markdown)

Diffèrent styles de listes sont disponibles (attention à l'indentation)

- + Pomme
- Banane
- \* Abricot

Vous pouvez mettre votre texte en **gras** ou en *italique*

Vous pouvez mettre votre texte en **gras** ou en *italique*

**La variante Git Flavoured Markdown ajoute :**

- [ ] la liste des taches
- [x] accomplies ou pas

La ~~biffure~~ et des liens URL automatiques

<http://www.linux-pratique.com>

Si vous citez un code la mise en couleur est automatique

```
element = document.getElementById("syl")
element.style.color = 'blue'

print('hello world')
```

Citations avec guillemets

Laurence ! Je crois que le bug il est la - entre le clavier et la chaise. Sylvain.

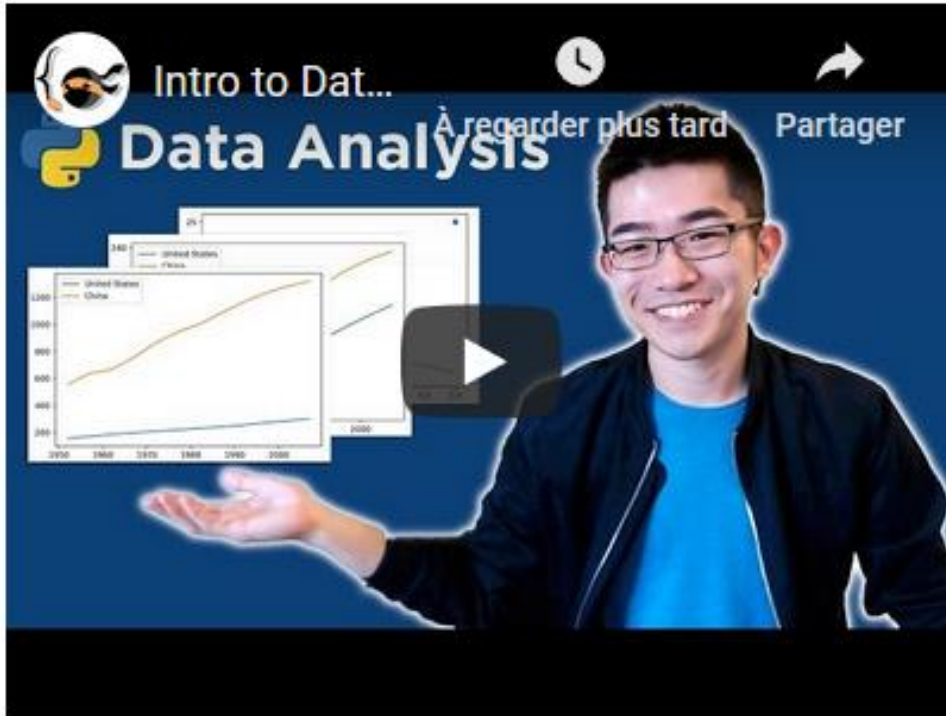
In [ ]:

# Introduction Jupyter Notebook

## Programmation et rédaction

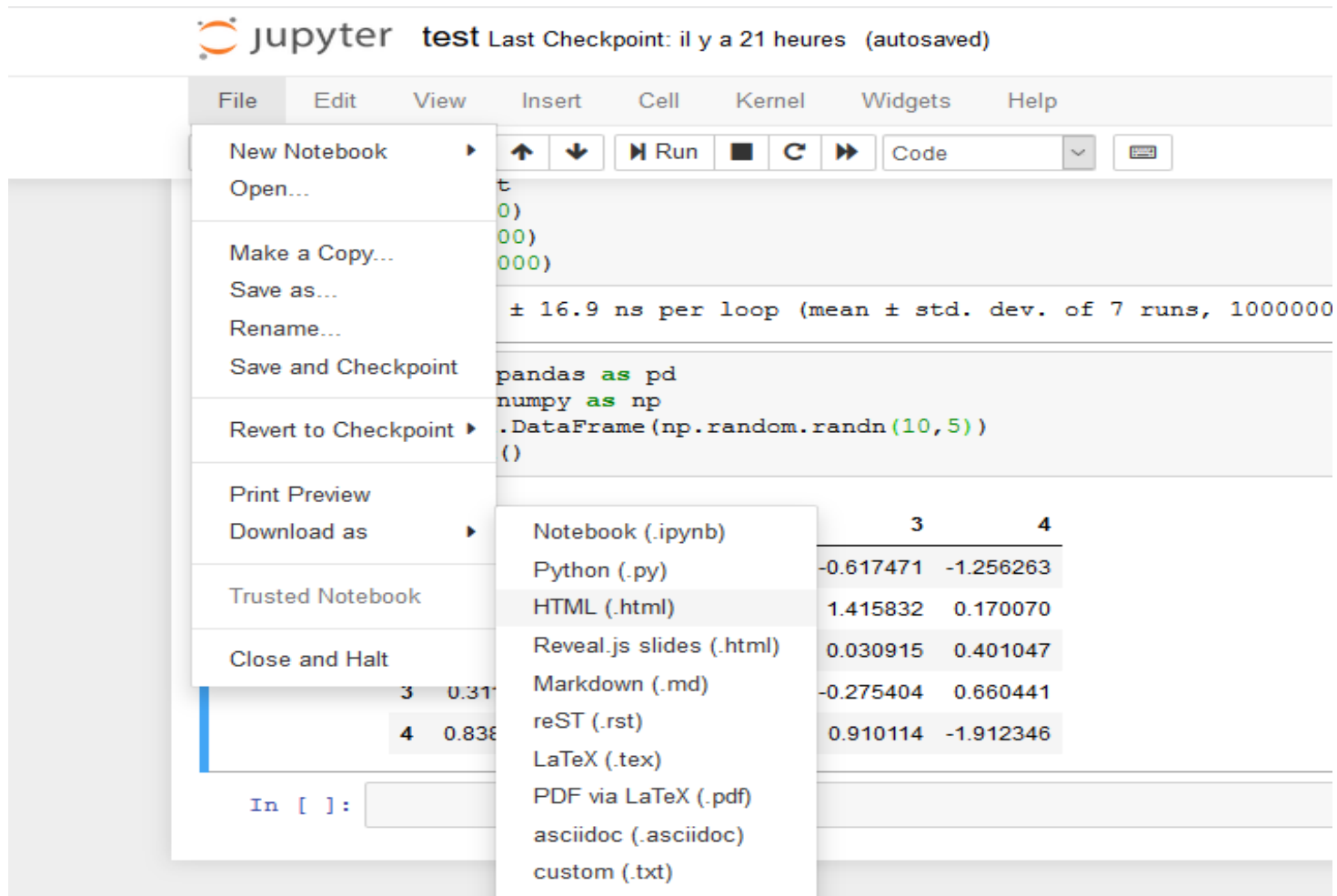
```
► In [26]: from IPython.display import YouTubeVideo  
YouTubeVideo('a9UrKTVeeZA')
```

Out[26]:



# Introduction Jupyter Notebook

## Programmation et rédaction -> Exporter son document



# En pratique ça marche comment

- **Variable:** un emplacement mémoire que l'on peut référencer par un nom. Sert à stocker une information.
- Autrement dit : une boîte dans laquelle on peut mettre une valeur. ensuite, on peut réutiliser cette valeur (sans la connaître) en écrivant le nom de la variable.

$x = 3$

*nom = 'Rafik'*

# En pratique ça marche comment

Python distingue différentes sortes de variables suivant le type de valeur qu'elles mémorisent :

- un nombre entier
- une chaîne de caractères
- un nombre à virgule
- des valeurs complexes de type liste

# On passe à la pratique

## Télécharger et installer Anaconda

# On passe à la pratique

En utilisant la bibliothèque math :

1. Trouver la solution de  $3x^2 - 7x = 23$
2. Il y a une fonction qui renvoie le plus grand diviseur commun à a et b.  
Trouver la forme irréductible de la somme :

$$\frac{217}{440} + \frac{101}{256} + \frac{86}{71}$$

3. Écrire un programme qui, à partir de la saisie d'un rayon et d'une hauteur, calcule le volume d'un cône droit. (Utiliser l'instruction input pour saisir les valeurs et print pour afficher le resultat).



# Les conditions

**if test** : effectue les instructions qui suivent si le test est respecté.

On remarque que les instructions qu'on veut exécuter dans la structure conditionnelle ne sont pas alignées avec le if mais sont en retraits : on dit que ces instructions sont indentées.

- Structure conditionnelle sous Python

*if test :*

*instructions1*

*else :*

*Instructions2*

Cela effectue les instructions1 lorsque le test est vérifié, sinon effectue les instructions2.

- La ligne qui précède l'indentation se finit toujours par deux points. Appuyer sur la touche Entrée après avoir tapé « : » effectue automatiquement l'indentation.

# Les conditions

*if test1 :*  
    *instructions1*

*elif test2 :*  
    *Instructions2*

Cela effectue les instructions1 indentées lorsque le test1 est vérifié, sinon effectue le test2 et, si celui-ci est vérifié, effectue les instructions2 indentées.

Remarques :

- On peut enchaîner autant de "elif" que nécessaire.
- On peut terminer une série de "elif" par un "else" afin d'être sûr de traiter tous les cas.

# On passe à la pratique

1. Ecrire un programme qui à partir d'une saisie d'un nombre vous dit s'il est pair ou impair.
2. Etant donné les longueurs des côtés d'un triangle (hypoténuse, côté adjacent) écrire un programme qui vérifie si un triangle est rectangle.

# En pratique ça marche comment

- **programmation orientée objet** : un style de programmation dans lequel les données et les opérations qui les manipulent sont organisées en **classes et méthodes**.
- **Liste** : Variable contenant une liste **ordonnée** d'éléments

*maListe = [ 4.7, 53, 2019, "formation", "Montpellier" ]*

# En pratique ça marche comment

- *maListe.append( x )* ajoute l'élément *x* à la fin de *maListe*
- *maListe[i]* donne le *i*ème élément de *maListe*
- *maListe.insert( i, x )* insère *x* à la position *i* dans *maListe*
- *maListe.remove( x )* supprime le premier élément égal à *x* s'il y est
- *maListe.count( x )* compte le nb d'occurrences de *x* dans *maListe*
- *maListe.sort()* tri *maListe*
- *maListe.reverse()* renverse l'ordre de *maListe*

# Range()

L'instruction range(début,fin,pas) génère une liste d'entiers (les paramètres début et pas sont optionnels)

- Dans l'intervalle [0 ; fin[ si un seul paramètre est renseigné.  
L=range(5) va créer la liste [0, 1, 2, 3, 4] de 5 termes, le premier terme sera L[0]=0, le dernier L[4]=5.
- Dans l'intervalle [début ; fin[ si 2 paramètres sont renseignés
- L=range(1,5) va créer la liste [1, 2, 3, 4], le premier terme sera L[0]=1 et le dernier L[3]=5.
- Dans l'intervalle [début ; fin[ mais de pas en pas, si les 3 paramètres sont renseignés.

# La boucle for

```
for i in range(4):  
    print("i a pour valeur", i)
```

```
c = [41, 32, 56, 34, 7]
```

```
for i in range(len(c)):  
    print("i vaut", i, "et c[i] vaut", c[i])
```

Pour tout  $i$  dans  $[0, 1, 2, 3]$   
afficher «  $i$  a pour valeur » la valeur  $i$

Pour tout  $i$  dans  $[0, 1, 2, 3, 5]$   
afficher «  $i$  vaut » la valeur  $i$  « et  $c[i]$   
vaut » la  $i$ ème valeur de la liste

# La boucle while

```
x = 1
```

```
while x < 10:  
    print("x a pour valeur", x)  
    x = x * 2
```

```
print("Fin")
```

Tant que x est inférieur à 10  
 afficher « x a pour valeur » la valeur x  
 x prend la valeur  $x^2$

Afficher « Fin »



# Pratique

- Créer un programme qui donne le prix TTC après avoir saisi le prix HT. Ce programme doit se répéter pour pouvoir entrer plusieurs prix à la suite et ne s'arrêter que si l'utilisateur rentre 0.
- Ecrire un programme qui vérifie si la liste [1, 2, 3, 4, 3, 2, 1] est un palindrome.