

Jupyter Notebook : introduction et usages

Anne Cadiou

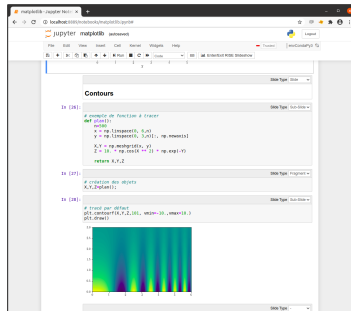
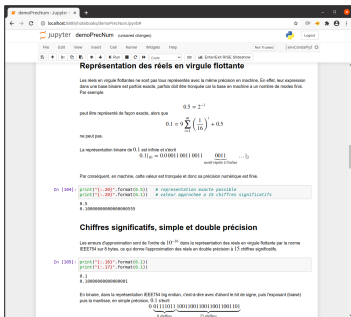
Laboratoire de Mécanique des Fluides et d'Acoustique

Ateliers et Séminaires Pour l'Informatique et le Calcul Scientifique
PMCS2I - LMFA
Vendredi 6 novembre 2020



Introduction

Un Jupyter Notebook est une **application web open source** qui permet de créer et partager des **cahiers de laboratoire**, documents regroupant dans une même page du texte, du code, des équations, des graphiques, etc.



L'objectif est de **faciliter la prise de notes** sur un projet de **permettre de refaire les calculs et analyses de données associées**. Ces notebooks sont plus particulièrement adaptés à la réalisation de tutoriaux interactifs.

Dans le cadre de développement de codes, ils peuvent aider à la documentation, la réalisation de tests. Ils sont plus particulièrement intéressants à utiliser comme feuille de synthèse, les lignes de codes étant écrites par ailleurs avec un éditeur, dans un environnement de développement. Pour ce faire, l'usage de Jupyter Lab peut être intéressant.



<https://jupyter.org>

Caractéristiques

Les notebooks sont des **documents JSON**, contenant une liste ordonnée de cellules (entrée/sortie) qui peuvent contenir des lignes de code, du texte (au format Markdown), des mathématiques (syntaxe proche de \LaTeX), des figures. Ils peuvent intégrer des fichiers mp4, etc.

Cette interface ressemble à celles de Mathematica, Maple ou SaGeMAth.

L'extension des fichiers est `.ipynb`

Un Jupyter Notebook peut être transformé en diapositive ou converti dans d'autres formats ouverts (HTML, présentations avec `reveal.js`, \LaTeX , PDF, ReStructuredTex, Markdown, Python) en exploitant `nbconvert`. Ces conversions peuvent être obtenues via la page web (Download As).

Kernel

Les `kernel` de Jupyter sont des environnements d'exécution des Notebooks. Pour un notebook donné, il est unique. À l'origine, il est basé sur le noyau IPython.

En général, un noyau permet l'exécution d'un langage donné.

Les noyaux ne sont pas attachés aux clients Notebooks, et peuvent être connectés aux clients en local ou par le réseau. Plusieurs clients peuvent s'y rattacher et donc partager le même environnement.

Historique

Les Jupyter Notebooks font partie du Project Jupyter (organisation pour le développement de logiciels opensource).

Nommés en hommage aux carnets de notes de Galilée, consignant la découverte des lunes de Jupiter. Rappelle aussi les langages de base supportés par Jupyter (Julia, Python, R).

Un environnement d'exécution s'appelle un noyau (kernel).

- 2014 : Fernando Pérez crée le Project Jupyter, à partir d'IPython ;
- 2015 : GitHub s'accorde avec Project Jupyter pour que le format des notebooks soit lu sur sa plateforme ;
- 2018 : Première version stable de JupyterLab (interface utilisateur regroupant les notebooks, terminal, éditeur de texte, navigateur, etc.) ;
- 2019 : Première version stable de JupyterHub (1.0), un serveur multi-utilisateurs pour les notebooks.

Actuellement, les notebooks s'interfacent avec différents environnements d'exécution dans une douzaine de langages.

Évolutions

Collaborer ou former via un notebook se popularise.

Créer son notebook et le partager peut se faire sur la plateforme Binder, <http://mybinder.org>.

Les notebooks permettent de travailler avec un noyau C, C++ ou Fortran.

En 2019, Google Drive exploite Colab (un environnement de type Jupyter Notebook) pour accéder à ses grilles.

En 2020, VS Code permet de travailler à la façon des Jupyter Notebooks.

Installation

En local sur sa machine, il est conseillé d'installer jupyter-notebook dans un environnement virtuel (virtualenv ou conda). Exemple d'installation avec conda :

Création de l'environnement

```
conda create --name envCondaPy3 python=3.8
```

Activation de l'environnement

```
conda activate envCondaPy3
```

Installation des packages pour les Notebook

```
conda install jupyter-notebook
conda install anaconda-navigator
```

Ajouter le noyau dans le jupyter-notebook

```
conda install ipykernel
python -m ipykernel install --user --name=envCondaPy3
```


Démarrage du serveur

Démarrage du serveur de Jupyter Notebook

```
jupyter-notebook
```

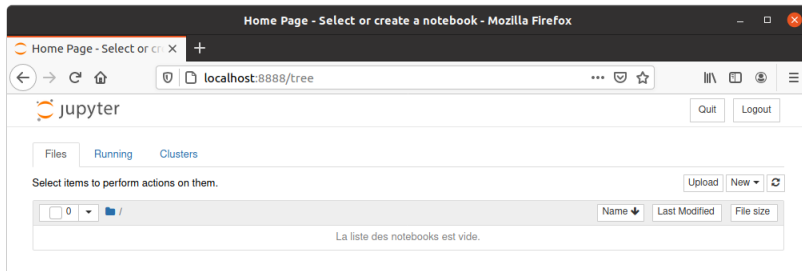
ou

```
jupyter notebook
```

Cela ouvre en local dans le navigateur par défaut une page du type :

```
http://localhost:8888/tree
```

qui ressemble à :



Le serveur tourne, mais aucun notebook n'est encore lancé.

Quelques paramètres utiles

Le numéro du port (8888 par défaut) s'inscrémente automatiquement suivant le nombre de serveur démarrés. Il peut être spécifié au lancement du serveur :

```
jupyter-notebook --port=9999
```

Le serveur peut aussi être démarré sans lancer de navigateur :

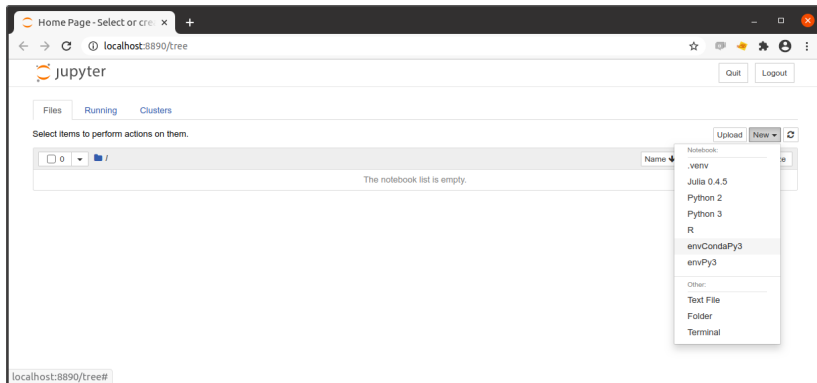
```
jupyter notebook --no-browser
```

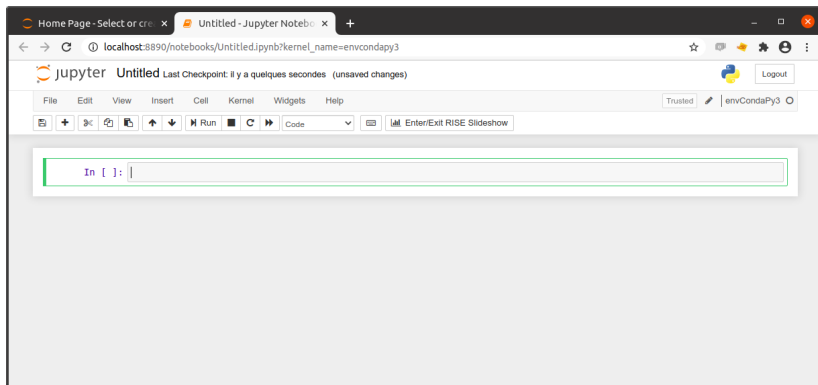
Cela peut être utile en cas de lancement du serveur sur une machine distante, à partager éventuellement avec plusieurs utilisateurs, ou sur une machine sans serveur X11. Procédure :

- Démarrer le serveur sur la machine distante ;
- Créer un tunnel ssh entre la machine distante et la machine locale ;
- Ouvrir un navigateur sur la machine locale en utilisant le tunnel ssh.

Création d'un notebook

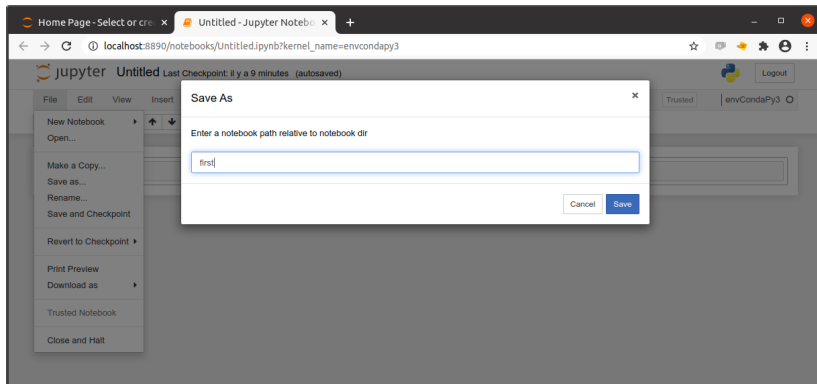
Cliquer sur le bouton **New** et sélectionner le noyau que vous voulez utiliser. Il doit avoir été installé sur la machine où est démarré le serveur et intégré au serveur jupyter qui a été lancé pour être dans les choix proposés.



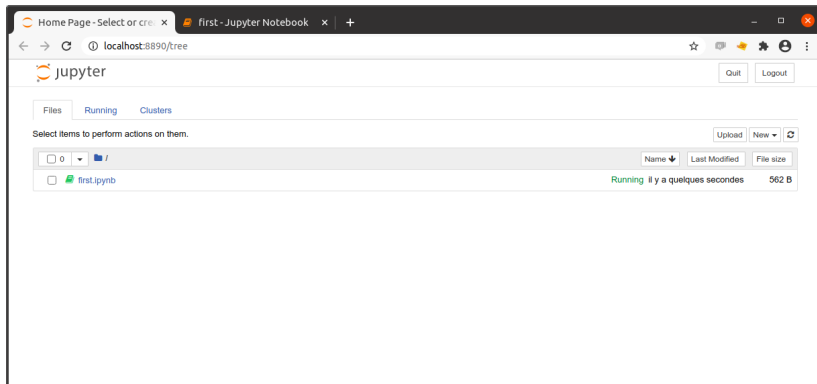


Le notebook est par défaut nommé Untitled.

Pour lui donner un nom, le sauver sous le nom voulu.

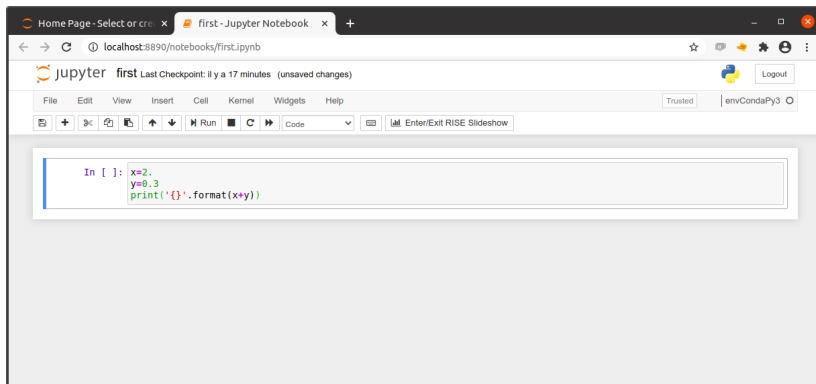


Il apparaît alors dans l'aborescence, sous ce nom :



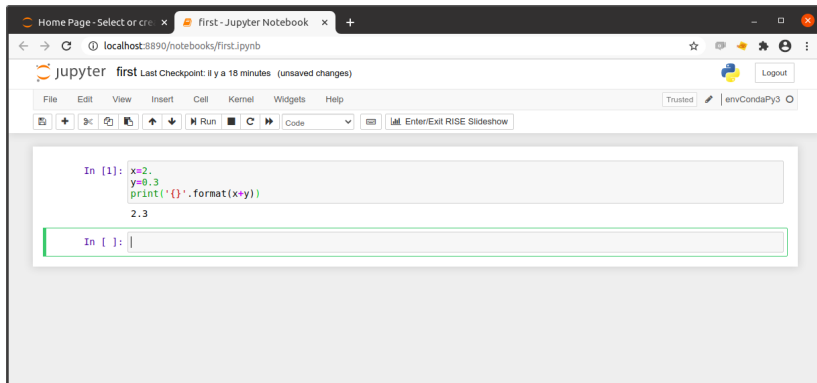
Saisir du code

Ici l'environnement envCondaPy3 est utilisé, qui est basé sur Python 3. Du code en ce langage peut donc être rentré dans les cellules du notebook.



Exécuter du code

Pour exécuter ce code, appuyer sur **SHIFT+ENTER**.



Les cellules sont numérotées suivant leur ordre d'exécution. Elles sont exécutées séquentiellement dans la page. Les variables définies dans une cellule sont partagées, de même que les imports. Lorsqu'on revient en arrière dans la page et ré-exécute la cellule, les variables associées sont modifiées.

Menus

Les notebook ont des menus, proposant des opérations couramment rencontrées dans des applications et d'autres plus spécifiques.

File Manipule les fichiers. Permet de sauvegarder et faire des reprises dans l'état d'exécution du notebook (Save and Checkpoint). Permet de convertir en un autre format (Download as);

Edit Manipule les cellules (couper, copier, réordonner, etc.);

View Déplie ou cache les menus. Permet de numéroter les lignes dans les cellules. Ajoute le rôle de chaque cellule dans le mode Slide Show (diaporama avec `node.js`);

Insert Ajoute une cellule avant ou après la cellule courante;

Cell Exécute une cellule ou un ensemble. Remet à zéro toutes les sorties;

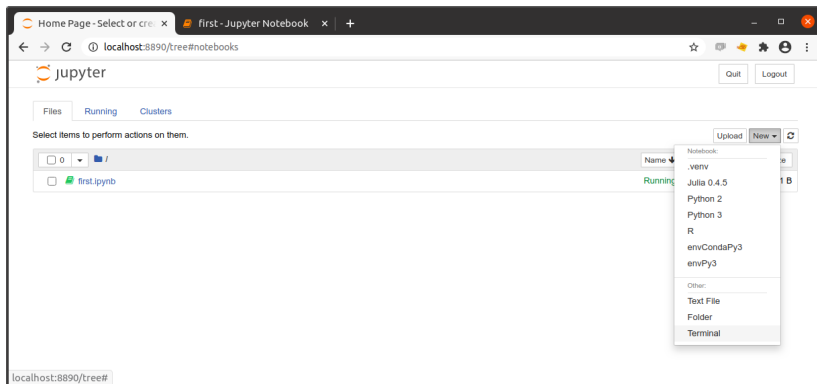
Kernel Permet de stopper ou redémarrer le noyau, ou d'en changer;

Widgets Ajoute du JavaScript;

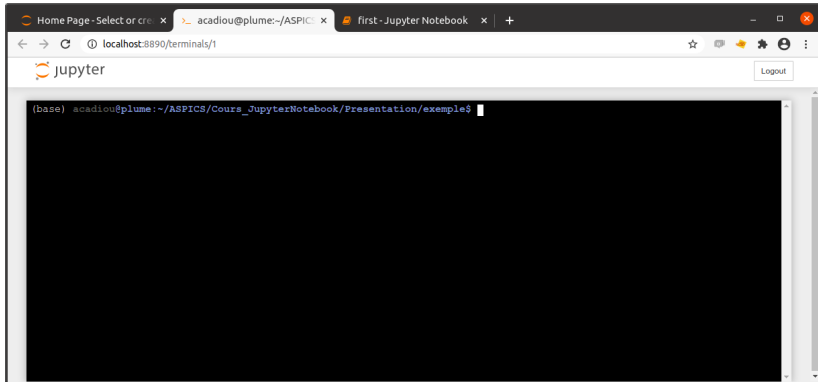
Help Aide et lien vers la documentation, les raccourcis clavier, la syntaxe Markdown, etc.

Démarrer un terminal

Pour utiliser des commandes bash, ajouter un kernel, trouver des fichiers, etc., le notebook permet de démarrer un terminal dans le navigateur.

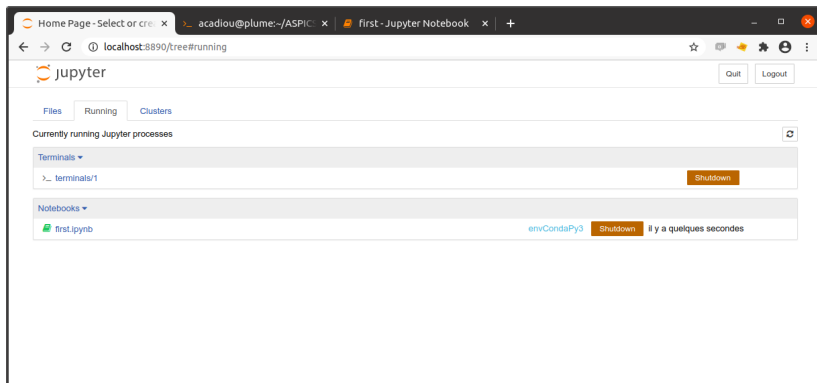


Démarrer un terminal



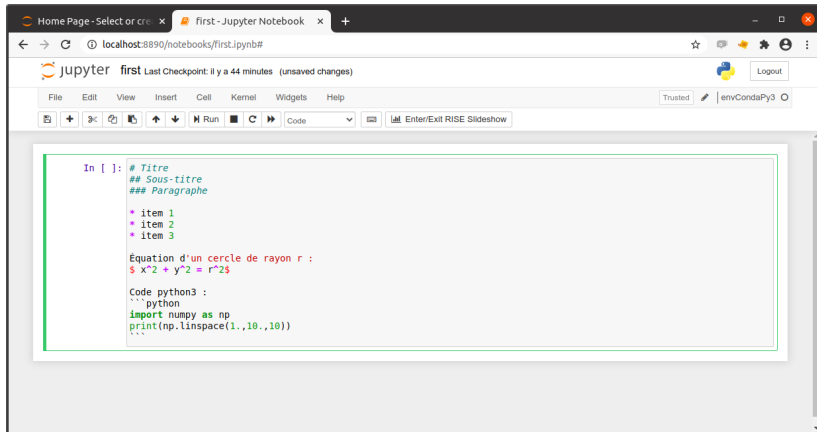
État du serveur

Dans le menu, les notifications permettent d'avoir une indication de l'état du serveur. Par notebook, cela peut aussi être consulté dans le navigateur, dans le menu Running.

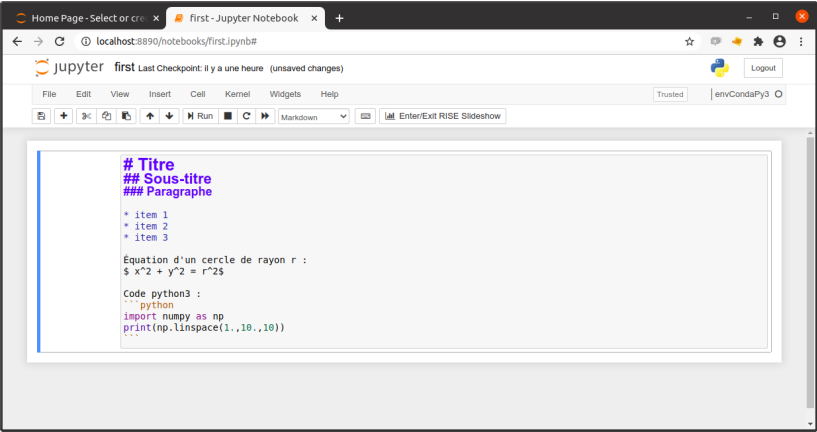


Mise en forme des cellules

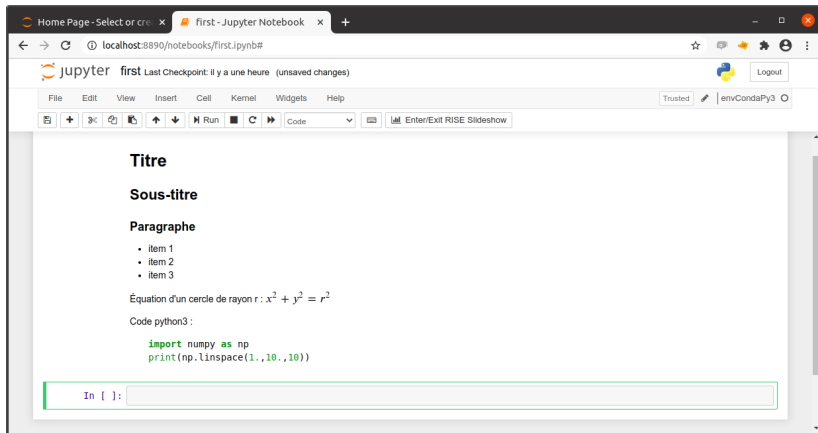
Les cellules peuvent utiliser du code ou du texte (format Markdown). Il est donc possible de mettre des sections, sous-sections, etc. d'insérer du code (qui ne doit pas être exécuté) ou des expressions mathématiques (rendu depuis $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$).



Sélectionner le mode Markdown :



Une fois exécuté avec SHIFT+ENTER :



Exporter

Dans le menu, la section File -> Download as permet de convertir le notebookd en d'autres format. La conversion peut aussi se faire en ligne de commande avec nbconvert.

- HTML
- \LaTeX
- PDF
- RevealJS
- Markdown
- ReStructured Text
- Executable script

Il faut noter que nbconvert dépend de pandoc et \LaTeX . Ces outils doivent être présents sur la machine pour pouvoir être utilisés.

Utilisation en ligne de commande

```
jupyter-nbconvert <input notebook> --to <outputformat>
```

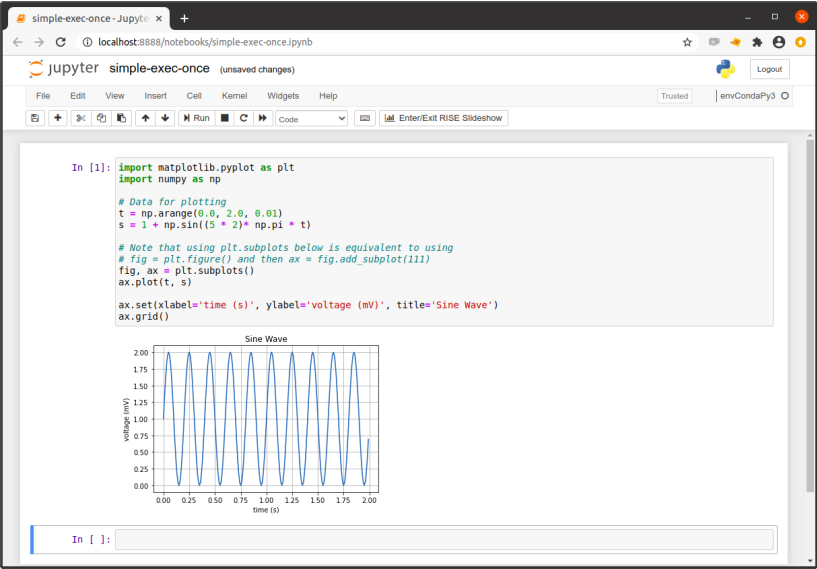
Exemple de conversion en PDF :

```
jupyter-nbconvert first.ipynb --to pdf
```

Exemple de conversion en Python :

```
jupyter-nbconvert first.ipynb --to python
```

Exemple d'export d'un Notebook



simple-exec-once

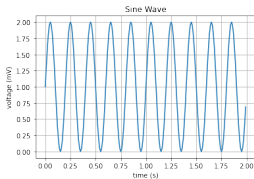
November, 6th

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

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin((5 * 2) * np.pi * t)

# Note that using plt.subplots below is equivalent to using
# fig = plt.figure() and then ax = fig.add_subplot(111)
fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)', title='Sine Wave')
ax.grid()
```



1

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Data for plotting
```

```
t = np.arange(0.0, 2.0, 0.01)
```

```
s = 1 + np.sin((5 * 2) * np.pi * t)
```

```
# Note that using plt.subplots below is
# equivalent to using
```

```
# fig = plt.figure() and then ax = fig.
# add_subplot(111)
```

```
fig, ax = plt.subplots()
```

```
ax.plot(t, s)
```

```
ax.set(xlabel='time (s)', ylabel='
      voltage (mV)', title='Sine Wave')
```

```
ax.grid()
```

```
# In[ ]:
```

Interaction avec du code python

Exemple sur le notebook `exemple01-code-python.ipynb`



exemple01-code-python - x +

localhost:8890/notebooks/exemples/exemple01-code-python.ipynb

jupyter exemple01-code-python Last Checkpoint: lundi dernier à 21:40 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted envCondaPy3

Run Code Enter/Exit RISE Slideshow

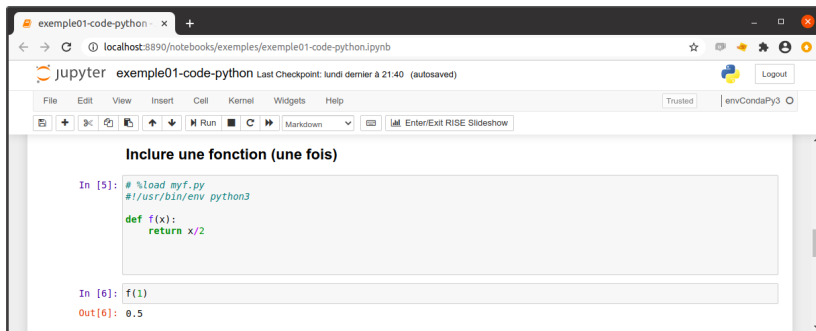
Définir et utiliser une fonction

Conversion d'un entier décimal en binaire

```
In [3]: def convInt2Binary(x):
        answer = []
        i = 0
        while(x>0):
            reste = x % 2
            x = x // 2
            answer.append(reste)
        answer.reverse()
        for i in range(len(answer)):
            print(answer[i], end="")
        return
```

```
In [4]: x=int(input("Saisir un entier à convertir: "))
        convInt2Binary(x)

        Saisir un entier à convertir: 5
        101
```



exemple01-code-python - x +

localhost:8890/notebooks/exemples/exemple01-code-python.ipynb

jupyter exemple01-code-python Last Checkpoint: lundi dernier à 21:40 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted envCondaPy3

Run Enter/Exit RISE Slideshow

Inclure une fonction (recharge la fonction dans son état avant exécution de la cellule qui relit le fichier)

```
In [7]: %run myf.py
```

```
In [8]: f(1)
```

```
Out[8]: 0.5
```

Inclure un module (en s'adaptant à son état sans devoir réexécuter la cellule qui importe le fichier)

à placer en principe en tout-début du notebook :

```
In [9]: %load_ext autoreload
        %autoreload 2
```

```
In [10]: from myf import f
```

```
In [11]: f(1)
```

```
Out[11]: 0.5
```

Remarques

Noter que l'insertion d'une fonction externe se fait :
une fois pour toute, indépendamment du fichier avec la commande `load` :

```
load f.py
```

en s'adaptant à l'état de la fonction au moment de l'exécution de la cellule qui l'intègre au notebook, par la commande `run` :

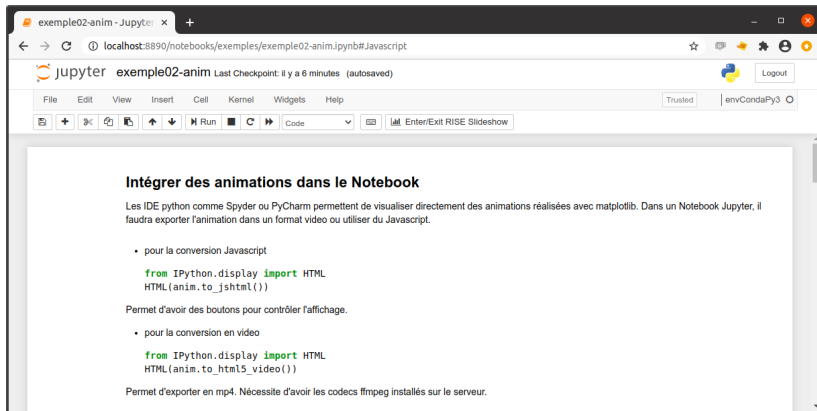
```
run f.py
```

en s'adaptant à l'état de la fonction (ou du module) au moment de l'exécution de la fonction (ou des méthodes) dans leurs cellules avec l'extension `autoreload`

```
%load_ext autoreload
%autoreload 2
from myf import f
f(1)
```


Exemple d'animation

Exemple sur le notebook `exemple02-anim.ipynb`



```
exemple02-anim - Jupyter x +
localhost:8890/notebooks/examples/exemple02-anim.ipynb#Javascript
jupyter exemple02-anim Last Checkpoint: il y a une minute (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted envCondaPy3
In [2]: import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation
from IPython.display import HTML

x = np.linspace(-4, 4, 100)

# function and derivative
def f(x):
    return x*x

def df(x):
    return 2.*x

y = f(x)
dy = df(x)

# figure
fig, ax = plt.subplots()
plt.close()
ax.set_xlim((-4, 4))
ax.set_ylim((-2, 20))
line1, = ax.plot([], [], lw=2)
line2, = ax.plot([], [], lw=2)

# initialization
def init():
    line1.set_data(x, y)
    return (line1,)

# animation function: this is called sequentially
def animate(i):
    at_x = x[i]

    # gradient line: m*x + b
    m = df(at_x)
    b = f(at_x) - df(at_x)*at_x
    gradient_line = m*x + b

    line2.set_data(x, gradient_line)
    return (line2,)
```

exemple02-anim - Jupyter x +

localhost:8890/notebooks/examples/exemple02-anim.ipynb#Javascript

jupyter exemple02-anim Last Checkpoint: il y a 2 minutes (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted envCondaPy3

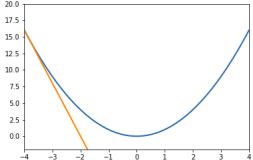
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=100, interval=100, blit=True)

Javascript

```
In [3]: def display_animation(anim):
plt.close(anim._fig)
return HTML(anim.to_jshtml())
```

```
In [4]: display_animation(anim)
```

Out[4]:



Controls: - < > >> >>> >>>> +

☐ Once ☒ Loop ☐ Reflect

example02-anim - Jupyter x +

localhost:8890/notebooks/examples/example02-anim.ipynb#JavaScript

jupyter example02-anim Last Checkpoint: 11 y a 3 minutes (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted envCondaPy3

Code Enter/Exit RISE Slideshow

video

```
In [11]: from tempfile import NamedTemporaryFile
import base64

VIDEO_TAG = """<video controls>
<source src="data:video/x-m4v;base64,{0}" type="video/mp4">
Your browser does not support the video tag.
</video>"""

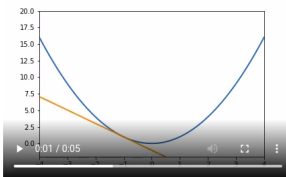
def anim_to_html(anim):
    if not hasattr(anim, 'encoded_video'):
        with NamedTemporaryFile(suffix='.mp4') as f:
            anim.save(f.name, fps=20, extra_args=['-vcodec', 'libx264'])
            video = open(f.name, "rb").read()
            anim_encoded_video = base64.b64encode(video)

    return VIDEO_TAG.format(anim_encoded_video.decode('ascii'))

In [12]: def display_animation(anim):
plt.close(anim._fig)
return HTML(anim_to_html(anim))

In [13]: display_animation(anim)

Out[13]:
```



Exploration interactive de paramètres

En utilisant la fonction interactive des `ipywidgets`, on peut faire évoluer des paramètres avec des curseurs graphiques (et récupérer les valeurs explorées).
Exemple sur le notebook `exemple03-widgets.ipynb`

exemple03-widgets - Jupyter Notebook - Mozilla Firefox

exemple03-widgets - Jup X +

localhost:8888/notebooks/exemple03-widgets.ipynb

jupyter exemple03-widgets Dernière Sauvegarde : 18/08/2020 (auto-sauvegardé) Logout

File Edit View Insert Cell Kernel Widgets Help

Flable envCondaPy3

Exécuter

Markdown

Explorer le système dynamique de Lorenz

Le système de Lorenz L63 est un modèle simplifié de la convection thermique dans l'atmosphère. Il s'écrit :

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

où σ est le nombre de Prandtl, ρ est une normalisation du nombre de Rayleigh et β un nombre d'onde adimensionnel. Le système du troisième ordre est un problème aux conditions initiales. Lorsque $\sigma = 10$, $\rho = 8/3$ et $\beta = 28$, il a un comportement déterministe chaotique.

exemple03-widgets - Jupyter Notebook - Mozilla Firefox

exemple03-widgets - Jupyter X +

localhost:8888/notebooks/exemple03-widgets.ipynb

jupyter exemple03-widgets Dernière Sauvegarde : 18/08/2020 (modifié)

File Edit View Insert Cell Kernel Widgets Help

Flairé envCondaPy3

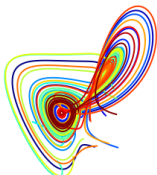
Entrée [1]: `%matplotlib inline`

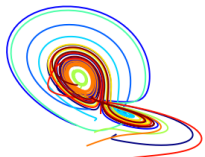
Entrée [2]: `%load_ext autoreload`
`%autoreload 2`

Entrée [3]: `from Lorenz import solve_lorenz`

Résolution pour un jeu de paramètres

Entrée [4]: `t, dx = solve_lorenz(angle=68, N=20, sigma=10.0, beta=8./3, rho=28.0)`





exemple03-widgets - Jupyter Notebook - Mozilla Firefox

exemple03-widgets - Jupyter X +

localhost:8888/notebooks/exemple03-widgets.ipynb

jupyter exemple03-widgets Dernière Sauvegarde : 18/08/2020 (auto-sauvegardé)

File Edit View Insert Cell Kernel Widgets Help

Exécuter Markdown Enter/Exit RISE Slideshow

Les arguments peuvent être récupérés par la méthode kwargs.

```
Entrée [16]: t, dx = w.result
```

```
Entrée [17]: w.kwargs
```

```
Out[17]: {'N': 29,
          'angle': 152.1,
          'max_time': 2.0,
          'sigma': 28.5,
          'beta': -1.06667,
          'rho': 34.9}
```

```
Entrée [ ]:
```


Versionner

Un Jupyter Notebook regroupe les données, codes et résultats. Lorsque des images sont manipulées ou produites, leur versionnement peut être perçu comme un changement important par git.

Dans le cas où seuls les codes sont à versionner, ce problème peut être contourné en :

- nettoyant les sorties avant tout commit ;

avantage : simplicité

inconvénient : opération manuelle, ne permet pas de partager un état de reprise du notebook

- convertir en python pour le commit ;

```
jupyter nbconvert --to="python"
```

avantage : code facile à versionner et partager ;

inconvénient : pas intégré au notebook, donc nécessite de mettre en oeuvre la procédure (par exemple par intégration continue)

Exemple

simple - Jupyter Notebook - Mozilla Firefox

simple - Jupyter Notebook x +

localhost:8888/notebooks/simple.ipynb

jupyter simple Dernière Sauvegarde : 04/08/2020 (auto-sauvegardé)

Logout

File Edit View Insert Cell Kernel Widgets Help

Fiabilité envCondaPy3

Exécuter Code Enter/Exit RISE Slideshow

Entrée []:

```
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin((5 * 2) * np.pi * t)

# Note that using plt.subplots below is equivalent to using
# fig = plt.figure() and then ax = fig.add_subplot(111)
fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)', title='Sine Wave')
ax.grid()
```

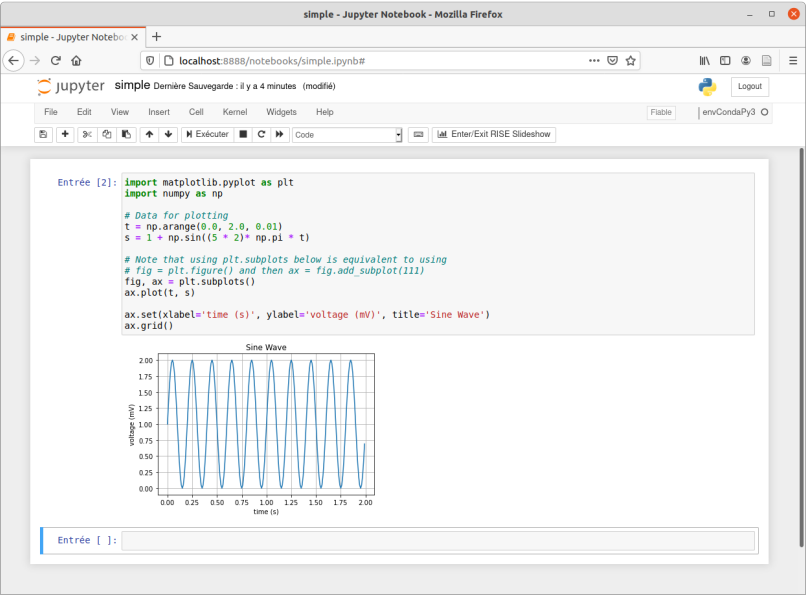
JSON

Un notebook est un fichier JSON...

```

1  {
2  "cells": [
3  {
4    "cell_type": "code",
5    "execution_count": null,
6    "metadata": {},
7    "outputs": [],
8    "source": [
9      "import matplotlib.pyplot as plt\n",
10     "import numpy as np\n",
11     "\n",
12     "# Data for plotting\n",
13     "t = np.arange(0.0, 2.0, 0.01)\n",
14     "s = 1 + np.sin((5 * 2) * np.pi * t)\n",
15     "\n",
16     "# Note that using plt.subplots below is equivalent to using\n",
17     "# fig = plt.figure() and then ax = fig.add_subplot(111)\n",
18     "fig, ax = plt.subplots()\n",
19     "ax.plot(t, s)\n",
20     "\n",
21     "ax.set(xlabel='time (s)', ylabel='voltage (mV)', title='Sine Wave')\n",
22     "ax.grid()
23   ]
24 },
25 {
26   "cell_type": "code",
27   "execution_count": null,
28   "metadata": {},
29   "outputs": [],
30   "source": [
31   ]
32 },
33   "metadata": {
34     "kernelspec": {
35       "display_name": "envCondaPy3",
36       "language": "python",
37       "name": "envcondapy3"
38     },
39     "language_info": {

```



... dans lequel les résultats sont encapsulés.

Open

simple.ipynb

Save

~/ASPICS/Cours_JupyterNotebook/Presentation/exemple

```

1 {
2   "cells": [
3     {
4       "cell_type": "code",
5       "execution_count": 1,
6       "metadata": {},
7       "outputs": [
8         {
9           "data": {
10            "image/png":
11              "iVBORw0KGgoAAAANSUhEUgAAAYAAAGAAEwAAABAAAAHSCVQICAgIahkIAAAAAwSfL2AAAEgAACiB0t1/+
12              AACADh0R7h0u29mdHdhcUABNF6cgvdxoPy1B2ZJ2aw9uMy4ylJisIgh0dHAgly9TYXrw690b0LlM9yzy+WHYJAAAGAELEQ
13              PB80C95d233j3PccUoR1UKECBE1LEvSpQCIUKECBFWjyIBESFChagRO1ISEBEIRigooSM1AREH00oITeoiEHARIKSIEKjIq
14              JSLbVnocESKsFKQCI5JHClJyn418JCISETkvI1+KyA8AKKK+Rcn1W0s4t4r1/9L2/40iorr8tt40/-
15              QgRdB8E1AgTGVjIBPCwL8H1gAbgX8GLC2TfG54s03/DwB70/-
16              22gVJqvxJYTKTjUHAARPGo4eMasqlVkaXqSqm1UmqbUodABH5WRF5wTvZ1e1/-
17              HAAITMgFEfma1Ejb8f9JRPa4x7aKyMe6H00uFDEPH67H/h3wJYlvz3n3vd3R0S0iMyJyBis1cr7/+
18              +U1UHSRNMW1ue1EzopIMuCYIKtoihARPGo4x2qL1L/j4h8XkQu8XHNjwI/ANwC/CTWwA18JC/-
19              wT4G8CLPPAT7rc4zUg7d4DHGvh5WB/kt+ec//90nArjPzTeC/lc1UuoE8DLwN9vu/-
20              dPAnymLqgHFFCQX0QCISjHBKqpoEa+QAF/CJwRke+JyHSPy/-
21              6VUMPwXUQZ2I6zCPARPPAV1VJ71F114P8Ebu2ksSulysCwA0u9j+l1lQIs4B7v90APoue/-
22              8dKqXNKqZp56rxdhSrn3MT9E3gYwVmvuj+FmhMESL4Q5QgInyK4C6eP6uUugK4Cbqcx93TD0eqQ1w5v77Y80v1Mi s1MwC5wHB8
23              ACITHaC3oLw30uZecumN/xjHnXWJUmokYoES9C1LZoFt7VgFBr6lWiWzG44pQoSe1AREH18MROR6Vw/vw3/-
24              jmh3Z5sat3sU+DUrUdG6916S1/ESP818CpoC/jSsgLFIxQdPub178YRyoub8nROTXYkYl9/-
25              0XD845YhvtbEdEwR1VREJCAif3Qw09wFvCoicZiCYRfwK0FvpJT6LvBbwLdFZM69z+d7nF8A3sCJJ+Xq0/-
26              8C8BktAbEv-D500P9wUMKxDNrxPeA64J3RsqfumuCJE6AeJGgZFiBAhQoR0iCyCIEBIRIjQEZGA1BAhQ00IHREJjAgRIKSi0BGRgI
27              PnjJvuULYrYnn279ru2EY0rOFBz2KJxBUM0ruD0GRuW03VZUyMXU40IESJ6IEThQESTECFChT6IBESECEBIR01ISEBIEBAHq0S0IA
28              Gc3HfCYeEPNgB3AuvUgeVUHQg2+65A0exUuebrx63hh8idu3j86y73x94HQbQcwfwnEuV1J4LSpnCvwXK6Z/-
29              1dswNlbMxw3v943b1SLT799/Q1NFzhjrx86z4HZwc+kekPxZVePUKjUBK57/+k8z+w9fX0C6AP/-
30              fz60cmC2uCG0d0G0ntYchBdyG05e3HRTZX0v+mPtbp9/-
31              v2n0L845C06anp8lms1pj2ofzhA9981lCV91blyEb0+9w+PdiN57/2fIFctcF112wnJjTwuv01fnt10u8snHwX7g21fGc6bu8rLLT
32              sgiHf413ArTimgoxa9k16R1DTZPLR+XtBx1usbvFm1559fPaqZMdZbM2x395Rys+50v/+B0CYtG5fy7rj0LNBKvFPb1/YAK/-
33              96r1040w9wP07771cgYB8Cvpl6vGDLne4RPX4ifmPSj0GPAawZcsWlclktH4ZzWbpd02jJ70Mn0e4X0w0Z27Rurc09p/0c/-
34              KJZ2whkS23ctUvYMDv27u4FD0F8Y3p05w/N5Xd5XGJ5gdX7hmsdR00G5zfwPdl6pdr+dsX37T5Mo3uW983D3vcz1D5/-
35              UHcb179993mgynker6cudE43w44+gsF7a+CAhX3HAH1142PjDaf/-
36              nfdGLH0FAEIS05p+H5NubYXknK3iefKagduknYXyeeB76I7rP25/ENjD7tkY993/-
37              A1m4e0e06dmM1YLCJ14wuPFLFL6FDXKMBT2/+VAE70+H2g0L904BUP250ICU/-
38              YPUMT3hg77a7TdTG0rd76G260ptu2eIRETdp+Y4j5wbLcXtx/LovKnURMms8/K0x1n3mhUuff/-
39              WchVrH7Y0K7jS+R1ANbds+gBTJ+ua+VugeIxcF89u0Lwqzd4es8PejHh9UPNoZCvd04229rZnqgdV1seX8wsVdhY+HFDaurCZxSTA
40              V5TtYvg7JhNdDeQ00qdBfAhR0RQ0KbXzRPXegeGrPKR0KvYtgZmLVV47dH5gtLftnuGkYb55JoY29463ETedY0eK5P/-/
41              gNOAB+d7G0Fust08xknk7u03ddYv72ze/0BhIVlZjxf3n+0m7rm08NrjowrHNKZP/+sR0JER8e7x3MB09899w93Xr0GaydHAn/-
42              m10+eSUKjYp0C381D9w7TA096bfc102l+Ru3bys7wz12mD12fzZXYcv5DPfvoqUoLY87vudt18104F/-
43              g7WGRF52/3ZiLyJRH5knv048BBYD/-
44              wh0F18K1YcV4PTn3QPB6VXubotj7yhtu2fy0DXH339oM+kBftStuS17j+X47I3ruX06wEzCw/-/
45              PT8Q2p5W+XP3XcPhg8GtndUG4gn9p2ioesv48dvvpXkVUF235MB0M7u0E2L3uBhB76ch66/-
46              jKT2nB5YUsk292b4+POEY3P3fnnQHqNkUP53n4JkFvj9Gdw+Heedy7mBBU+37T5FOHjYsNBnJwaZuAnlUrZPdd5q/-
47              dsM0h79pa/LyJZC0nB5I7af04V58G03b+T+a9exdcDwo15sZjG9oJQSpdTNSl6b3T+PK6UeVUo96p6j1fK/-

```

```
jupyter nbconvert --to python simple.ipynb
```

```

1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[2]:
5
6
7 import matplotlib.pyplot as plt
8 import numpy as np
9
10 # Data for plotting
11 t = np.arange(0.0, 2.0, 0.01)
12 s = 1 + np.sin((5 * 2) * np.pi * t)
13
14 # Note that using plt.subplots below is equivalent to using
15 # fig = plt.figure() and then ax = fig.add_subplot(111)
16 fig, ax = plt.subplots()
17 ax.plot(t, s)
18
19 ax.set(xlabel='time (s)', ylabel='voltage (mV)', title='Sine Wave')
20 ax.grid()
21
22
23 # In[ ]:
24
25
26
27

```

Outils utiles

<https://nbdime.readthedocs.io/en/latest/> est un outil pour aider à l'analyse des différences et au versionnement.

`nbdiff-web simple-1.ipynb simple-2.ipynb`

nbdime - diff and merge your Jupyter notebooks - Mozilla Firefox

nbdime - diff and merge: X +

127.0.0.1:40551/diff?base=simple-1.ipynb&remote=simple-2.ipynb

Notebook Diff

Enter notebook filenames or URLs in the form below to get started.
Please input filenames/URLs of notebooks to diff:

Base: Remote: Diff files

☒ Hide unchanged cells ☐ Trust outputs

Base Remote

In [2]:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Data for plotting
5 t = np.arange(0.0, 2.0, 0.01)
6 s = 1 + np.sin((5 * 2) * np.pi * t)
7
8 # Note that using plt.subplots below is equivalent to using
9 (...)
```

In [2]:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Data for plotting
5 t = np.arange(0.0, 5.0, 0.01)
6 s = 1 + np.sin((15 * 2) * np.pi * t)
7
8 # Note that using plt.subplots below is equivalent to using
9 (...)
```

Outputs changed

Output deleted

Output added

1 unchanged cell(s) hidden

Serveur distant

En général, les machines distantes n'ont pas de serveur graphique. Pour travailler sur ces machines, il faut donc démarrer le serveur Jupyter sans navigateur.

Par exemple, une fois l'environnement du noyau installé sur la machine distante (ici créé avec conda dans l'exemple), on voudra lancer, depuis la machine locale, un script comme :

```
ssh -f newton.ecl "conda init; export PATH="/home/lmfa/acadiou/miniconda3/bin:
$PATH"; cd ASPICS/exemples; conda activate envCondaPy3; jupyter notebook --no
-browser --port=8889"
```

où `newton.ecl` est l'alias `ssh` de la connexion à la machine distante.

Ce script démarre le serveur distant. Pour travailler Il suffit alors de définir un tunnel `ssh` :

```
ssh -N -f -L localhost:8889:localhost:8889 newton.ecl
```

```
[I 22:29:57.038 NotebookApp] Serving notebooks from local directory: /home/lmfa/
acadiou/ASPICS/exemples
[I 22:29:57.038 NotebookApp] The Jupyter Notebook is running at:
[I 22:29:57.038 NotebookApp] http://localhost:9999/?token=7
a23f3b493a6a80475038b1d4dc939eb0099d162ff4e0799
```




Pour stopper le serveur :

```
ssh -f newton.ecl "conda init; export PATH="/home/lmfa/acadiou/miniconda3/bin:
$PATH"; cd ASPICS/exemples; conda activate envCondaPy3; jupyter notebook 9999
stop"
```

L'exemple est donné ici avec un environnement conda, mais un environnement virtuel avec virtualenv conviendra tout aussi bien.

JupyterLab

JupyterLab est un environnement de développement (IDE) qui s'ouvre dans le navigateur.

Installation avec conda

```
conda install -c conda-forge jupyterlab
```

ou pip

```
pip install jupyterlab
```

JupyterLab se lance ensuite par :

```
jupyter lab
```


JupyterLab - Mozilla Firefox

localhost:8888/lab

File Edit View Run Kernel Tabs Settings Help

Name	Last Modified
scripts	2 months ago
exemple01-code...	2 months ago
exemple02-anim.j...	2 months ago
exemple03-widget...	a month ago
exemple04-rise.ip...	14 days ago
fig.png	2 minutes ago
figTex.png	2 minutes ago
figure.png	2 minutes ago
Lorenz.py	2 months ago
matplotlib_cool.png	2 minutes ago
matplotlib_coolwa...	2 minutes ago
matplotlib_GinBu.p...	2 minutes ago
matplotlib_gray.png	2 minutes ago
matplotlib_jet.png	2 minutes ago
matplotlib_parula....	2 minutes ago
matplotlib_rainbo...	2 minutes ago
matplotlib_winter....	2 minutes ago
matplotlib.ipynb	2 months ago
myI.py	2 months ago
package-list.txt	2 months ago
requirements.yml	a month ago
run-env.sh	2 months ago

Console 1 matplotlib.ipynb

Python 3.8.3 (default, Jul 2 2020, 16:21:59)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.

```
[1]: import matplotlib
```

[]:

envCondaPy3 | idle Ln 1, Col 1 Console 1

JupyterHub

JupyterHub est un serveur multi-utilisateurs, permettant de connecter plusieurs instances de notebooks (fonctionne avec des proxy). Le Hub peut gérer des authentifications.

Exemple pour les personnels avec un compte `univ-lyon1.fr`



Jupyter - Mozilla Firefox

Jupyter

https://jupyter.mecanique.univ-lyon1.fr/pages/jupyter-v1.html

Caractéristiques :

- Système [Debian 10 aka Buster](#)
- notebook : 6.0.3
- jupyterhub : 1.1.0
- Utilisateurs ≤ 240 (6×40)

Limitations des ressources

Chaque utilisateur est limité à 1 Go de mémoire et un CPU complet (Xeon@2095MHz).
Votre noyau plantera et sera redémarré en cas de dépassement mémoire.

Vous pouvez utiliser le module `psutil` pour connaître l'utilisation mémoire du noyau courant :

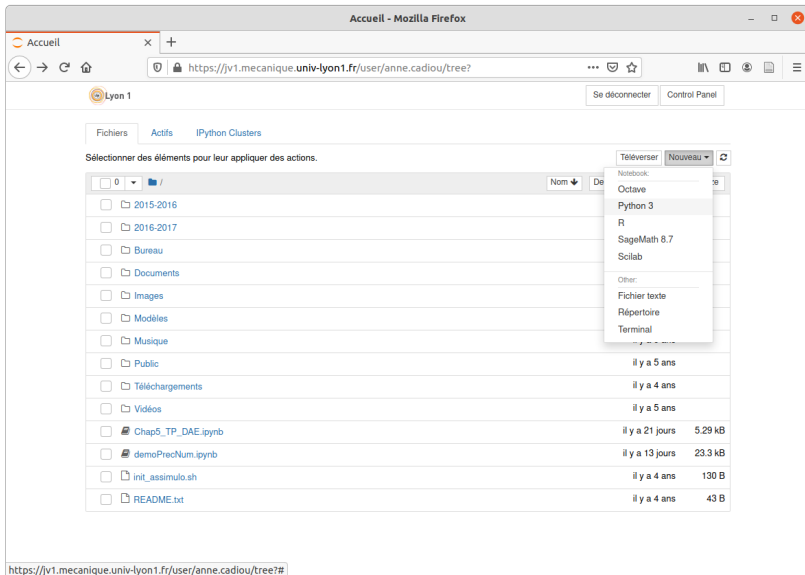
```
import psutil
print('{:.0f} Mo used'.format(psutil.Process().memory_full_info().uss/1024/1024))
print('{:.0f} Mo swapped'.format(psutil.Process().memory_full_info().swap/1024/1024))

# Using standard library only you can read /proc/self/status
with open('/proc/self/status') as fd:
    for line in fd:
        if line.startswith('VmRSS'):
            print(line.split(':')[1].strip())
```

De plus le serveur jupyter occupe environ 70 Mo.

Noyaux:

- Python 3.7 [Docs](#)
 - jupyterlab : 2.0.0
 - scipy : 1.2.1
 - matplotlib : 3.0.3
 - pandas : 0.24.2
 - sympy : 1.4
- Sage 8.7
 - python 3.6



matplotlib - Jupyter Notebook - Mozilla Firefox

Accueil x matplotlib - Jupyter Note x +

<https://jv1.mecanique.univ-lyon1.fr/user/anne.cadiou/notebooks/matplotlib.ipynb>

Lyon 1 matplotlib Dernière Sauvegarde : il y a une minute (modifié)
 Se déconnecter
Control Panel

Fichier Édition Affichage Insérer Cellule Noyau Widgets Aide
 Flable
Python 3 ○

Exécuter
Code
Memory: 164 / 1536 MB

Modules

Entrée [1]:

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Tracé de courbes

Entrée [2]:

```
# exemple de fonctions à tracer
L=15.
N=256
X = np.linspace(-np.pi*L, np.pi*L, N, endpoint=True)

def f(X):
    freq=5.
    return np.cos(X)*np.sin(X/freq)
def g(X):
    return np.sin(X)
```

Sortie dans une fenêtre graphique

Entrée [3]:

```
%matplotlib inline
# Dans le notebook, inline peut être remplacé
# par notebook pour avoir des graphes interactifs

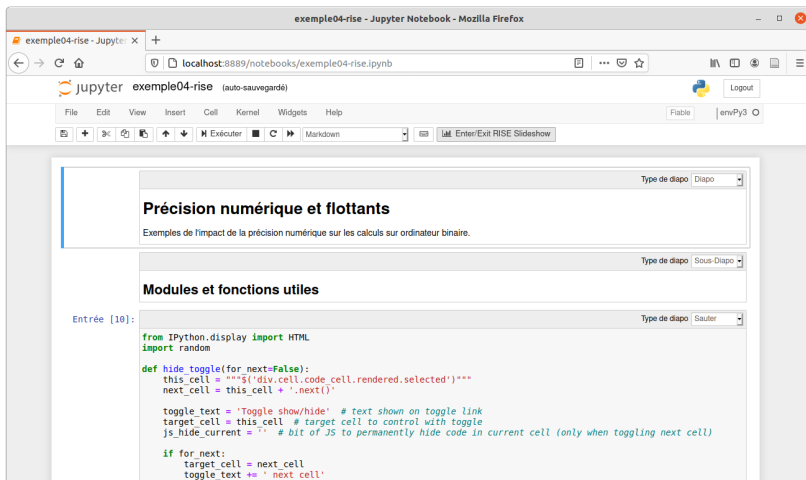
# Dans ipython, le mode interactif
# peut être activé par %matplotlib

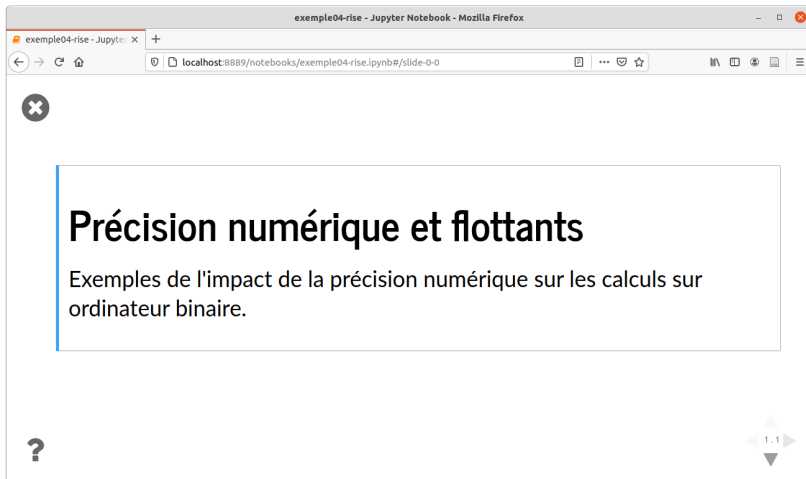
# Dans un script, l'accès interactif
```


Diaporama

Les notebook peuvent être transformés en mode de diaporama.
Pour cela, il suffit d'installer l'extension RISE

```
jupyter-nbextension install rise --py --sys-prefix
jupyter-nbextension enable rise --py --sys-prefix
```





Le style des pages peut être modifié via un `.css`

nbgrader

`https://nbgrader.readthedocs.io/en/stable/`

permet de préparer des feuilles d'exercices pour chaque élève, en intégrant l'évaluation par comparaison aux réponses attendues, d'intégrer des macros personnalisées en fonction de chaque élève, etc.

Références

- <https://jupyter.org>
- Noyau C <https://github.com/brendan-rius/jupyter-c-kernel>
- Noyau Fortran
<https://github.com/ZedThree/jupyter-fortran-kernel>
- VS Code <https://pbpython.com/notebook-alternative.html>
- Boutons interactifs
<https://ipywidgets.readthedocs.io/en/stable/examples>
- Versionner
<https://nextjournal.com/schmudde/how-to-version-control-jupyter>
- <https://nbdime.readthedocs.io/en/latest/>
- <https://pypi.org/project/slurm-jupyter/>