

Environnement de travail

Entrée [1]:

```
1 print("Bonjour monde")
2
```

Bonjour monde

Nous considérons, par la suite, que la version de python utilisée est supérieure à 3. Pour connaître la version disponible, taper la commande :

```
1 python --version
```

Pour connaître les versions disponibles sur votre OS vous pouvez vous rendre sur la page de téléchargement de python (<https://www.python.org/downloads/>)

Pour mettre à jour python avec une version plus récente, par exemple la 3.7, taper la commande :

```
1 pip install python3.7
```

Les sections suivantes présentent les environnements qui sont utilisés pour pouvoir facilement manipuler des données, faire des expérimentations, obtenir des visualisation, etc... bref toutes les opérations classiques en sciences des données.

Le premier environnement est **VirtualEnv**, un environnement virtuel qui offre la possibilité de pouvoir tout intégrer dans le cas d'un projet. Cet environnement peut par la suite être partagé et toutes les informations sont disponibles pour que les expérimentations soient reproductibles (version de librairies utilisés, code, etc).

Le second environnement est **Jupyter**, une application web utilisée pour programmer dans de très nombreux langages dont Python. Il permet de réaliser des notebooks, i.e. des programmes contenant à la fois du texte en markdown (un langage de balisage léger) et du code en Python. Ces notebooks sont particulièrement utilisés en science des données pour explorer et analyser des données.

Utilisation de Virtualenv

Python est un langage qui possède différentes versions et de très nombreuses librairies qui évoluent constamment. Il n'est donc pas rare de travailler sur différents projets avec des versions différentes entraînant de possibles incompatibilités. Virtualenv (*Virtual Environment*) vous permet de pouvoir gérer facilement ces différentes librairies.

Installation de virtualenv

L'installation de virtualenv se fait à l'aide de la commande pip install comme l'indique l'exemple :

```
1 pip instal virtualenv
```

ou

```
1 sudo pip install virtualenv
```

Créer et activer un environnement virtuel

La création se fait de la manière suivante :

```
1 virtualenv -p python3 env
```

cette commande permet de créer un environnement appelé *env* dans lequel la version de python par défaut sera python3. Un répertoire appelé *env* est créé dans le répertoire local. Il est possible de l'explorer :

ls env

```
1 bin    include  lib      etc
```

L'activation de l'environnement est réalisé par :

```
1 source env/bin/activate
```

Une fois la commande exécutée l'environnement s'affiche dans le terminal

```
1 (env)
```

La version de python utilisée à présent est celle de votre environnement virtuel. La commande `which` permet de la connaître :

```
1 (env)which python
```

```
/usr/bin/python3
```

Pour connaître les packages installés dans l'environnement

```
1 (env)pip freeze
```

```
matplotlib==2.1.2 mistune==0.8.3 nbconvert==5.3.1 nbformat==4.4.0 nltk==3.2.5  
numpy==1.14.0 pandas==0.22.0 scikit-learn==0.19.1
```

Desactivation et suppression de l'environnement

Pour désactiver l'environnement, il suffit de lancer la commande `deactivate`

```
1 (env)deactivate
```

Attention il faut à chaque fois relancer l'environnement avec `source env/bin/activate`

Pour supprimer l'environnement il suffit d'effacer le répertoire

```
1 rm -rf env
```

Sauvegarde des dépendances

Les dépendances peuvent être sauvegardées et c'est préférable pour pouvoir partager l'environnement. Pour cela il suffit de créer un fichier `requirements.txt` de la manière suivante :

```
1 (env)pip freeze > requirements.txt
```

Installation des dépendances

Si un fichier requirement.txt existe il suffit, pour installer toutes les dépendances du projet de taper

```
1 (env)pip install -r requirement.txt
```

Pour avoir la liste de toutes les installations (outre le pip freeze précédent)

```
1 (env)pip list
```

Utilisation de Jupyter

Jupyter est composé de trois parties :

- La partie notebook (front end) permet d'éditer et d'exécuter des notebooks. Il s'agit d'une application JavaScript utilisable sur un navigateur. Le front end est en charge de gérer en local les notebooks et de pouvoir les transférer au serveur.
- La partie serveur Jupyter est une application simple de serveur.
- La partie protocole Kernel qui permet de décharger la charge du serveur en relayant l'exécution de code à un kernel associé à un langage spécifique (Python, R, ... 40 langages).

Installation de Jupyter

Même si jupyter est installé sur votre OS, il est recommandé de l'installer dans l'environnement virtuel pour éviter des incompatibilités notamment avec les extensions.

```
1 pip install jupyter
```

Il existe de très nombreuses extensions qui sont pratiques à l'usage : possibilité de pouvoir cacher des parties (C.f. les flèches devant les sections pour les faire apparaître à la demande, possibilité de cacher du code pour ne pas surcharger, de ne relancer que certains traitements, de notifier par mail la fin des tâches, etc.) De nombreux exemples sont disponibles : <https://ndres.me/post/best-jupyter-notebook-extensions/>
(<https://ndres.me/post/best-jupyter-notebook-extensions/>)

```
1 pip install jupyter_contrib_nbextensions
```

jupyter_nbextensions_configurator permet de générer une extension du serveur de jupyter-notebook afin de pouvoir facilement activer/désactiver les extensions

```
1 #Installation du configurateur
2 pip install jupyter_nbextensions_configurator
3 jupyter-contrib-nbextension install --sys-prefix
4
5 #Activation du configurateur
6 jupyter-nbextensions_configurator enable --sys-prefix
```

Lancement de Jupyter

Pour lancer jupyter, taper :

```
1 jupyter notebook
2 ou
3 jupyter-notebook
```

Lors de l'exécution une fenêtre s'ouvre sur votre navigateur :




Lors de l'exécution une URL apparaît également. Si vous souhaitez utiliser un autre navigateur il suffit de recopier l'adresse indiquée qui contient le token de sécurité.

un exemple d'URL <http://localhost:8888/?token=ae12bab371826367bf7db657510e7b044ed84914072ed4da>
(<http://localhost:8888/?token=ae12bab371826367bf7db657510e7b044ed84914072ed4da>)

Pour voir les extensions qui peuvent être ajoutées à votre environnement :

```
1 http://localhost:8888/nbextensions
```


Nbextensions configuration ([more information](#))
Logout

Configurable nbextensions
⌵

☒ disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter:

<input type="checkbox"/> (some) LaTeX environments for Jupyter	<input type="checkbox"/> 2to3 Converter	<input checked="" type="checkbox"/> AddBefore	<input type="checkbox"/> Autopep8
<input type="checkbox"/> AutoSaveTime	<input checked="" type="checkbox"/> Autoscroll	<input type="checkbox"/> Cell Filter	<input type="checkbox"/> Code Font Size
<input type="checkbox"/> Code prettify	<input checked="" type="checkbox"/> Codefolding	<input type="checkbox"/> Codefolding in Editor	<input checked="" type="checkbox"/> CodeMirror mode extensions
<input checked="" type="checkbox"/> Collapsible Headings	<input type="checkbox"/> Comment/Uncomment Hotkey	<input checked="" type="checkbox"/> contrib_nbextensions_help_item	<input type="checkbox"/> datestamper
<input type="checkbox"/> Equation Auto Numbering	<input type="checkbox"/> ExecuteTime	<input type="checkbox"/> Execution Dependencies	<input type="checkbox"/> Exercise
<input type="checkbox"/> Exercise2	<input type="checkbox"/> Export Embedded HTML	<input type="checkbox"/> Freeze	<input type="checkbox"/> Gist-it
<input type="checkbox"/> Help panel	<input type="checkbox"/> Hide Header	<input type="checkbox"/> Hide input	<input type="checkbox"/> Hide input all
<input type="checkbox"/> Highlight selected word	<input checked="" type="checkbox"/> highlighter	<input type="checkbox"/> Hinterland	<input type="checkbox"/> Initialization cells
<input type="checkbox"/> isort formatter	<input checked="" type="checkbox"/> jupyter-js-widgets/extension	<input type="checkbox"/> Keyboard shortcut editor	<input checked="" type="checkbox"/> Launch QTConsole
<input type="checkbox"/> Limit Output	<input type="checkbox"/> Live Markdown Preview	<input type="checkbox"/> Load TeX macros	<input type="checkbox"/> Move selected cells
<input type="checkbox"/> Navigation-Hotkeys	<input checked="" type="checkbox"/> Nbextensions dashboard tab	<input checked="" type="checkbox"/> Nbextensions edit menu item	<input type="checkbox"/> nbTranslate
<input type="checkbox"/> Notify	<input type="checkbox"/> Printview	<input type="checkbox"/> Python Markdown	<input type="checkbox"/> Rubberband
<input type="checkbox"/> Ruler	<input checked="" type="checkbox"/> Runtools	<input type="checkbox"/> Scratchpad	<input type="checkbox"/> ScrollDown
<input type="checkbox"/> Select CodeMirror Keymap	<input type="checkbox"/> SKILL Syntax	<input checked="" type="checkbox"/> Skip-Traceback	<input type="checkbox"/> Snippets
<input type="checkbox"/> Snippets Menu	<input type="checkbox"/> spellchecker	<input type="checkbox"/> Split Cells Notebook	<input type="checkbox"/> Table of Contents (2)
<input type="checkbox"/> table_beautifier	<input type="checkbox"/> Toggle all line numbers	<input type="checkbox"/> Tree Filter	<input type="checkbox"/> Variable Inspector
<input checked="" type="checkbox"/> zenmode			

Fermeture de Jupyter

Attention lors de la fermeture du navigateur, le serveur jupyter est toujours activé ou il faut explicitement demander sa fermeture via l'interface. Si jupyter a été lancé avec

```
1 jupyter notebook
```

Il suffit de faire un CTRL-C pour tuer le processus. Par contre s'il a été lancé par :

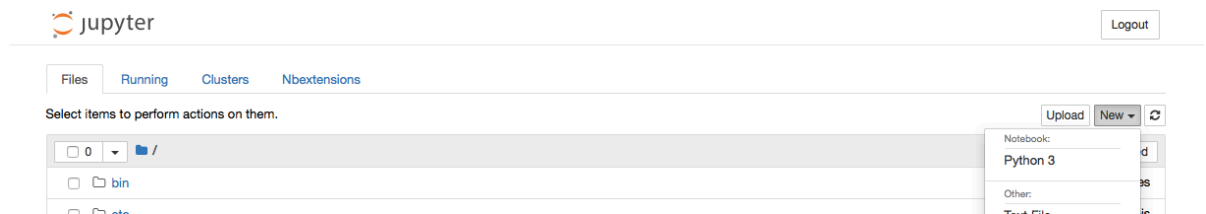
```
1 jupyter notebook &
2 ou
3 jupyter notebook & >> /dev/null
```

Il faut utiliser la commande

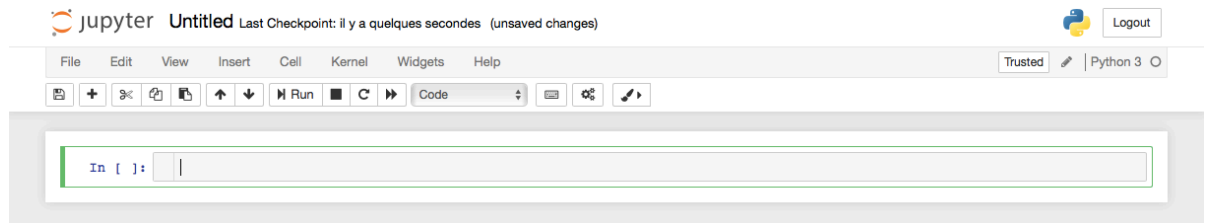
```
1 jupyter notebook stop
2 ou
3 jupyter notebook stop numport (si le processus n'a pas été
   lancé sur le port standard 8888)
```

Principales commandes

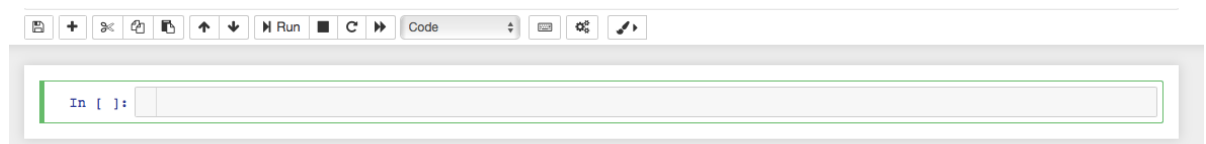
Sur la droite, une liste déroulante apparaît lors de l'appui sur New. Elle permet de créer un nouveau notebook et il est possible de sélectionner Python 3.



Une nouvelle fenêtre apparaît, intitulée Untitled et un fichier Untitled.ipynb (extension d'un notebook - le fichier est écrit en Json et décrit le contenu de la page) est automatiquement créé. Le serveur fait très régulièrement des sauvegardes automatiques.



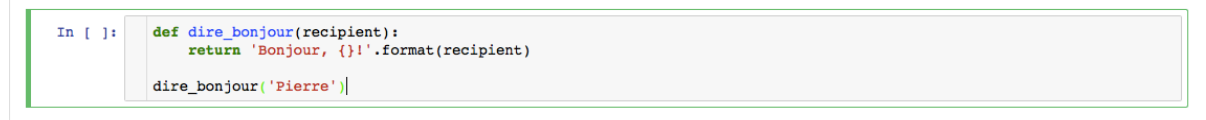
Par défaut la première fenêtre de saisie est en mode "code" :



Il existe quatre types de codage différents pour les cellules :

- Code : écriture de code Python qui peut être exécuté
- Markdown : écriture de texte avec des balises markdown
- Raw NBConvert : écriture de texte simple
- Heading : écriture de titres à l'aide des balises markdown

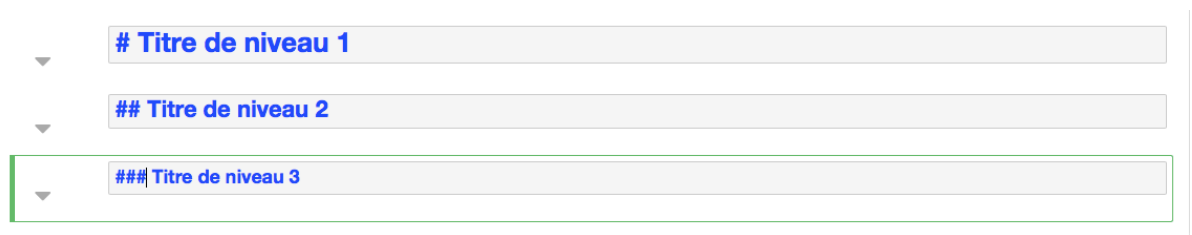
Un exemple de premier programme python.



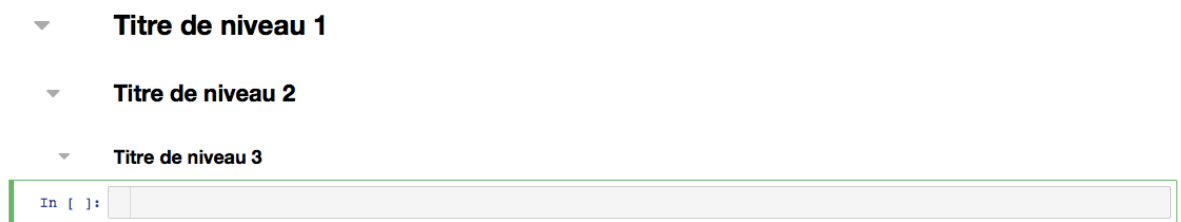
Pour exécuter une cellule il faut taper CTRL-ENTREE. L'exécution produit la figure suivante où nous notons sur la même vue le code et son exécution.



Exemples d'utilisations de type Heading



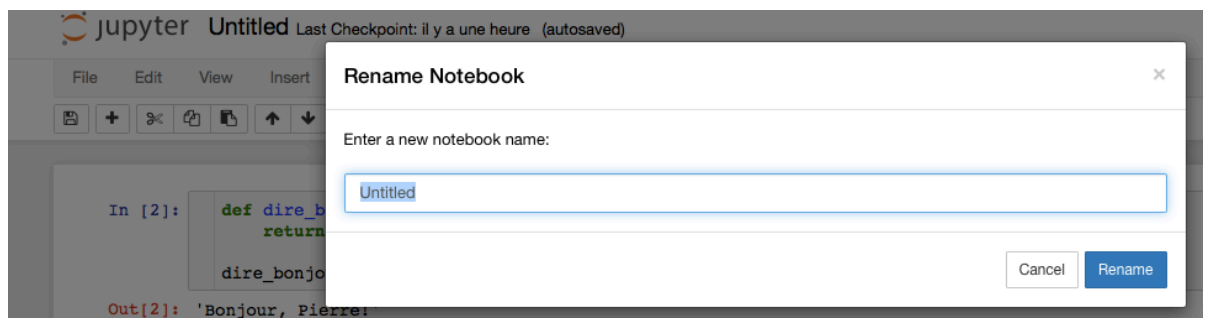
A l'exécution (CTRL-ENTREE) cela donne :



Remarque : les flèches à gauche sont produites par l'extension et permettent de pouvoir cacher les éléments afin de faciliter la lecture.

MAJ-ENTREE permet d'exécuter le code et de créer une nouvelle cellule

Il est possible de renommer le fichier Untitled en cliquant sur le nom :



Le contenu des notebooks peut être sauvegardé en HTML, en PDF. Il est possible de les associer à GitHub. L'adresse du serveur peut être publique. Il existe de très nombreuses ressources disponibles sur le web pour aider à mieux utiliser Jupyter. Il ne faut pas hésiter à les consulter.

Quelques informations sur markdown

Il existe de très nombreuses fonctionnalités utilisables en markdown. Le bloc ci-dessous présente certaines d'entre elles.


```
1 Sauter une ligne : il faut mettre 2 espaces avant de sauter la
  ligne.
2 Ceci est un saut de ligne
3 Ceci est un autre saut de ligne (il y a deux espaces avant le
  retour chariot)
4
5
6
7 Mise en valeur du texte
8 * Texte en gras : __texte__ ou texte
9 * Texte en italique : _texte_ ou texte
10
11
12 Du code Latex peut être introduit. Par exemple, les symboles
  mathématique :  $\sqrt{3x-1}+(1+x)^2$ ,  $\forall x \in \mathbb{R}$ 
13
14
15
16 Ligne horizontale :
17 ***
18
19
20 Indentation de texte : utilisation de > tout le texte est
  indenté jusqu'au prochain retour chariot
21 > ceci est un exemple de texte qui est indenté
22
23 Pour créer un point devant un item : un tiret suivi par un ou
  deux espaces :
24 - item
25
26 Il est possible de créer une hiérarchie. Pour cela il faut
  faire précéder la partie imbriquée par une tabulation.
27 - item
28   - sous item
29
30 Pour créer une énumération faire précéder par 1. devant
  l'item. Il n'est pas nécessaire de mettre autre chose que 1.,
  lors de l'exécution l'énumération est mise à jour
  automatiquement.
31 1. item1
32 1. item2
33 1. item3
34 1. item4
35
36 Il est également possible de créer une hiérarchie
  d'énumérations :
37 1. item1
38   1. item1.1
39   1. item 1.1.1
40
41 Il est possible d'utiliser des boîtes de couleur pour mettre
  en avant des éléments. Utiliser la balise <div> et mettre le
  contenu en HTML.
42 Convention :
```

```

43 - Les boîtes bleus pour des alertes informations
44 - Les boîtes jaunes pour des alertes warning
45 - Les boîtes vertes pour des alertes succès
46 - Les boîtes rouges pour des alertes danger
47
48
49 <div class="alert alert-block alert-info">
50 <b>Exemple :</b> utilisation pour ajouter des informations.
51 </div>
52
53 <div class="alert alert-block alert-warning">
54 <b>Exemple :</b> Utilisation d'une formule qui peut engendrer
    des erreurs.
55 </div>
56
57 <div class="alert alert-block alert-success">
58 <b>Exemple :</b> Lorsque les résultats sont obtenus.
59 </div>
60
61 <div class="alert alert-block alert-danger">
62 <b>Exemple :</b> Prévenir qu'une action peut engendrer la
    perte de données.
63 </div>
64
65 Insertion d'images
66 En local :
67 ![title](img/FirstLine.png "Titre de l'image optionnel")
68
69
70 Possibilité de faire des liens rapides :
71 Une liste de nombreuses fonctionnalités pour markdown est
    disponible ici :
    <https://daringfireball.net/projects/markdown/basics>
72

```

Exécution du bloc précédent :

Sauter une ligne : il faut mettre 2 espaces avant de sauter la ligne.

Ceci est un saut de ligne

Ceci est un autre saut de ligne (il y a deux espaces avant le retour chariot)

Mise en valeur du texte

- Texte en gras : **texte** ou **texte**
- Texte en italique : *texte* ou *texte*

Du code Latex peut être introduit. Par exemple, les symboles mathématique :

$$\sqrt{3x-1} + (1+x)^2, \forall x \in$$

Ligne horizontale :

Indentation de texte : utilisation de > tout le texte est indenté jusqu'au prochain retour chariot

ceci est un exemple de texte qui est indenté

Pour créer un point devant un item : un tiret suivi par un ou deux espaces :

- item

Il est possible de créer une hiérarchie. Pour cela il faut faire précéder la partie imbriquée par une tabulation.

- item
 - sous item

Pour créer une énumération faire précéder par 1. devant l'item. Il n'est pas nécessaire de mettre autre chose que 1., lors de l'exécution l'énumération est mise à jour automatiquement.

1. item1
2. item2
3. item3
4. item4

Il est également possible de créer une hiérarchie d'énumérations :

1. item1
 - A. item1.1
 - a. item 1.1.1

Il est possible d'utiliser des boîtes de couleur pour mettre en avant des éléments. Utiliser la balise

et mettre le contenu en HTML.

Convention :

- Les boîtes bleus pour des alertes informations
- Les boîtes jaunes pour des alertes warning
- Les boîtes vertes pour des alertes succès
- Les boîtes rouges pour des alertes danger

Exemple : utilisation pour ajouter des informations.

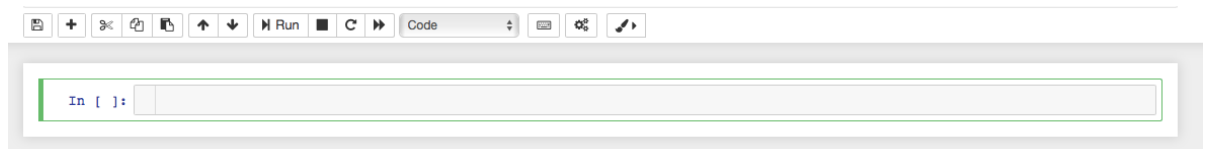
Exemple : Utilisation d'une formule qui peut engendrer des erreurs.

Exemple : Lorsque les résultats sont obtenus.

Exemple : Prévenir qu'une action peut engendrer la perte de données.

Insertion d'images

En local :



Possibilité de faire des liens rapides :

Une liste de nombreuses fonctionnalités pour markdown est disponible ici :

<https://daringfireball.net/projects/markdown/basics>

(<https://daringfireball.net/projects/markdown/basics>)

Générer du latex, du pdf ou des transparents avec Jupyter

Avec Jupyter il est possible de générer du Latex, du pdf, des slides. Pour cela il suffit d'utiliser la commande suivante :

```
1 jupyter nbconvert --to  
  {html|script|markdown|rst|latex|pdf|slides|html|}  
  nomdufichier.ipynb
```

La documentation officielle de nbconvert est disponible :

<https://nbconvert.readthedocs.io/en/latest/index.html>

(<https://nbconvert.readthedocs.io/en/latest/index.html>)

HTML

La valeur par défaut est : html

Par exemple les deux commandes suivantes sont équivalentes :

```
jupyter nbconvert environnement.ipynb
```

```
jupyter nbconvert --to html environnement.ipynb
```

Elles vont créer un fichier environnement.html contenant le contenu du notebook.

SCRIPT

Elle permet de générer un fichier python exécutable directement. Attention les cellules de type "RawNb convert" posent des problèmes. En effet ces dernières indiquent à Jupyter

qu'il ne faut pas les convertir et qu'elles seront sorties telles quelles lors de la conversion.

```
jupyter nbconvert --to script environnement.ipynb
```

génère un fichier python environnement.py.

MARKDOWN

Elle permet de créer un fichier markdown.

```
jupyter nbconvert --to markdown environnement.ipynb
```

génère un fichier markdown environnement.md.

RST

Elle permet de générer un fichier reStructuredText, i.e. un fichier texte qui peut être utilisé par la suite. Remarque : pour la conversion, nbconvert utilise la librairie Pandoc qui permet de convertir des documents sous de multiples formats.

```
jupyter nbconvert --to rst environnement.ipynb
```

génère un fichier python environnement.rst.

LATEX

Elle permet de générer un fichier Latex.

```
jupyter nbconvert --to latex environnement.ipynb
```

génère le fichier environnement.tex.

PDF

Elle permet de générer un fichier pdf.

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

génère d'abord un fichier latex puis conversion en pdf pour obtenir le fichier environnement.pdf

REDIRECTION DE LA SORTIE

Pour toutes les transformations précédentes, il est possible de rediriger la sortie soit vers la sortie standard soit vers un autre fichier.

```
jupyter nbconvert environnement.ipynb --stdout
```

affiche la transformation html sur la sortie standard.

```
jupyter nbconvert environnement.ipynb --output monsite.html
```

génère la transformation dans le fichier monsite.html.

Améliorations

Traitement de plusieurs notebook

Il est possible de transformer plusieurs notebook en même temps. Pour cela créer un fichier mycfg.py et recopier les informations contenant les notebook à transformer :

```
c = get_config()
c.NbConvertApp.notebooks = ["notebook1.ipynb", "notebook2.ipynb",
"notebook2.ipynb"]
```

Utiliser la commande :

```
jupyter nbconvert --config mycfg.py
```

Les templates

Il est aussi possible d'améliorer les documents transformés en utilisant des templates. Jinja2 (<http://jinja.pocoo.org/docs/2.10/> (<http://jinja.pocoo.org/docs/2.10/>)) est un langage de template pour Python pour définir les blocks et la manière de les écrire. Les template ont des sections, définies par des tags qui indique comment afficher le résultat.

Le principe est le suivant :

- Créer un fichier my_template.tplx qui contient les différentes parties que vous souhaitez mettre en forme
- Lancer la commande de transformation en utilisant le template :

```
jupyter nbconvert --to=... --template=./my_template.tpl file.ipynb
```

Transformation Latex et PDF

Ajout de Metadonnées

Pour spécifier un auteur, son affiliation et un titre de document. Dans Jupyter il faut aller dans le menu suivant : Edit/Edit Notebook Metadata et insérer les informations :

```
"latex_metadata": {
  "author": "Nom auteur",
  "affil": "Nom affiliation",
  "title": "Titre du document"
}
```

Attention les noms des attributs doivent correspondre au style Latex utilisé. Par exemple ici affil est le nom utilisé dans le package Latex authblk. author et title correspondent respectivement à l'attribut author et titre qui seront affichés dans la page.

Création d'un template

Par exemple, Vous trouverez sur le site :

<https://gist.githubusercontent.com/goerz/d5019bedacf5956bcf03ca8683dc5217/raw/393f1>
(<https://gist.githubusercontent.com/goerz/d5019bedacf5956bcf03ca8683dc5217/raw/393f1>)

un exemple de template, revtex.tplx, proposé par l'auteur de :

<https://michaelgoerz.net/notes/custom-template-for-converting-jupyter-notebooks-to-latex.html> (<https://michaelgoerz.net/notes/custom-template-for-converting-jupyter-notebooks-to-latex.html>)

Le fichier légèrement modifié est donné ci-dessous. Attention la classe de l'article n'est pas la même et il y a quelques modifications.

Fichier de configuration revtex.tplx

```
1 ((*- extends 'article.tplx' -*))
2
3 % improvement of the initial revtex created bt
4 % See http://blog.juliusschulz.de/blog/ultimate-ipynb-
  notebook#templates
5 % for some useful tips
6
7 %=====
  =====
8 % Document class
9 %=====
  =====
10
11 ((* block docclass *))
12 %\documentclass[10pt,notitlepage,onecolumn,aps,pra]{revtex4-1}
13 \documentclass[10pt,a4,notitlepage,onecolumn]{article}
14 ((* endblock docclass *))
15
16 %=====
  =====
```

```

17 % Packages
18 %=====
19
20 ((* block packages *))
21 \usepackage{authblk}%for affil
22 \usepackage[T1]{fontenc}
23 \usepackage{graphicx}
24 % We will generate all images so they have a width \maxwidth.
  This means
25 % that they will get their normal width if they fit onto the
  page, but
26 % are scaled down if they would overflow the margins.
27 \makeatletter
28 \def\maxwidth{\ifdim\Gin@nat@width>\linewidth\linewidth
29 \else\Gin@nat@width\fi}
30 \makeatother
31 \let\Oldincludegraphics\includegraphics
32 % Set max figure width to be 80% of text width, for now
  hardcoded.
33 \renewcommand{\includegraphics}[1]
  {\Oldincludegraphics[width=.8\maxwidth]{#1}}
34 % Ensure that by default, figures have no caption (until we
  provide a
35 % proper Figure object with a Caption API and a way to capture
  that
36 % in the conversion process - todo).
37 \usepackage{caption}
38 \DeclareCaptionLabelFormat{no label}{}
39 \captionsetup{labelformat=no label}
40
41 \usepackage{adjustbox} % Used to constrain images to a maximum
  size
42 \usepackage{xcolor} % Allow colors to be defined
43 \usepackage{enumerate} % Needed for markdown enumerations to
  work
44 \usepackage{geometry} % Used to adjust the document margins
45 \usepackage{amsmath} % Equations
46 \usepackage{amssymb} % Equations
47 \usepackage{textcomp} % defines textquotesingle
48 % Hack from http://tex.stackexchange.com/a/47451/13684:
49 \AtBeginDocument{%
50   \def\PYZsq{\textquotesingle}% Upright quotes in
  Pygmentized code
51 }
52 \usepackage{upquote} % Upright quotes for verbatim code
53 \usepackage{eurosym} % defines \euro
54 \usepackage[mathletters]{ucs} % Extended unicode (utf-8)
  support
55 \usepackage[utf8x]{inputenc} % Allow utf-8 characters in the
  tex document
56 \usepackage{fancyvrb} % verbatim replacement that allows latex
57 \usepackage{grffile} % extends the file name processing of
  package graphics

```



```

58 % to support a larger range
59 % The hyperref package gives us a pdf with properly built
60 % internal navigation ('pdf bookmarks' for the table of
    contents,
61 % internal cross-reference links, web links for URLs, etc.)
62 \usepackage{hyperref}
63 \usepackage{booktabs} % table support for pandoc > 1.12.2
64 \usepackage[inline]{enumitem} % IRkernel/repr support (it uses
    the enumerate* environment)
65 \usepackage[normalem]{ulem} % ulem is needed to support
    strikethroughs (\sout)
66 % normalem makes italics be
    italics, not underlines
67 \usepackage{braket}
68 ((* endblock packages *))
69
70 %=====
    =====
71 % Title Page
72 %=====
    =====
73
74 ((* block title -*))
75 ((*- endblock title *))
76 ((* block author -*))
77 ((* endblock author *))
78
79 ((* block maketitle *))
80
81 ((*- if nb.metadata.get("latex_metadata", {}).get("title",
    ""): -*))
82 \title{((( nb.metadata["latex_metadata"]["title"] ))))}
83 ((*- else -*))
84 \title{((( resources.metadata.name | ascii_only | escape_latex
    )))}}
85 ((*- endif *))
86
87 ((*- if nb.metadata.get("latex_metadata", {}).get("author",
    ""): -*))
88 \author{((( nb.metadata["latex_metadata"]["author"] ))))}
89 ((*- else -*))
90 \author{John Doe}
91 ((*- endif *))
92
93 ((*- if nb.metadata.get("latex_metadata",
    {}).get("affiliation", ""): -*))
94 \affil{((( nb.metadata["latex_metadata"]["affiliation"] ))))}
95 ((*- endif *))
96
97 \date{\today}
98 \maketitle
99
100 ((* endblock maketitle *))
101

```

```

102
103 %=====
=====
104 % Input
105 %=====
=====
106
107 % Input cells can be hidden using the "Hide input" and "Hide
input all"
108 % nbextensions (which set the hide_input metadata flags)
109
110 ((* block input scoped *))
111 ((*- if cell.metadata.hide_input or nb.metadata.hide_input: -
*))
112 ((*- else -*))
113 ((( custom_add_prompt(cell.source | wrap_text(88) |
highlight_code(strip_verbatim=True), cell, 'In ', 'incolor')
)))
114 ((*- endif *))
115 ((* endblock input *))
116
117
118 %=====
=====
119 % Output
120 %=====
=====
121
122 ((* block output_group -*))
123 ((*- if cell.metadata.hide_output: -*))
124 ((*- else -*))
125 ((( super() )))
126 ((*- endif -*))
127 ((* endblock output_group *))
128
129 % Display stream output with coloring
130 ((* block stream *))
131 \begin{Verbatim}[commandchars=\\\{\}]
132 ((( output.text | wrap_text(86) | escape_latex | ansi2latex
)))
133 \end{Verbatim}
134 ((* endblock stream *))
135
136 %=====
=====
137 % Define macro custom_add_prompt() (derived from add_prompt()
macro in style_ipython.tplx)
138 %=====
=====
139
140 ((* macro custom_add_prompt(text, cell, prompt, prompt_color)
-*))
141 ((*- if cell.execution_count is defined -*))
142 ((*- set execution_count = "" ~ (cell.execution_count |

```

```

143     replace(None, " ") -*)
144     ((*- else -*))
145     ((*- set execution_count = " " -*))
146     ((*- set indention = " " * (execution_count | length + 7)
    -*))
147 \begin{Verbatim}[commandchars=\\\{\}]
148 ((( text | add_prompts(first='\color{' ~ prompt_color ~ '}' ~
    prompt ~ '[\color{' ~ prompt_color ~ '}' ~ execution_count ~
    '}]:' , cont=indention) )))
149 \end{Verbatim}
150 ((*- endmacro *))
151
152 %=====
    =====
153 % Bibliography
154 %=====
    =====
155
156 % Insert citations in markdown as e.g.
157 %     <cite data-cite="DevoretS2013">[DevoretS2013]</cite>
158 % requires file references.bib in current directory (or the
    file set as "bib" in the latex_metadata)
159
160 ((* block bibliography *))
161 \bibliography{((( nb.metadata.get("latex_metadata",
    {})).get("bib", "references") ))})
162 ((* endblock bibliography *))
163
164

```

Utilisation du template

```

1 jupyter nbconvert --to=latex --template=./revtex.tplx
  file.ipynb

```

ou bien pour obtenir le pdf :

```

1 jupyter nbconvert --to=pdf --
  PdfExporter.template_file=./revtex.tplx file.ipynb

```

Transformation HTML

Creation d'un template

Le fichier de template, my_html_template.tpl modifie fait un encadrement épais en bleu de toutes les cellules.

```
1 {% extends 'full.tpl'%}
2
3 {% block any_cell %}
4     <div style="border-style: solid;border-width: 5px; border-
5     color: blue">
6         {{ super() }}
7     </div>
8 {% endblock any_cell %}
9
```

Utilisation du template

```
1 jupyter nbconvert --to=html --template=./my_html_template.tpl
  file.ipynb
```

Creation de transparents

Il est possible de faire des transparents à partir de Jupyter. L'approche s'appuie sur reveal.js (<https://revealjs.com/#/> (<https://revealjs.com/#/>)) qui doit être disponible sur votre OS.

Un fichier .html est automatiquement créé et un appel à reveal.js est fait pour naviguer en utilisant les flèches.

Choix de la présentation

Se rendre sur le menu View/Cell Toolbar et sélectionner Slideshow. Cela a pour effet d'afficher à droite un menu contextuel pour sélectionner le type de transparent que l'on souhaite faire.

Il existe 4 types de présentations possibles pour les cellules :

Slide : Chaque cellule sera présentée sur un transparent différent. Il faut utiliser les flèches de droite et gauche pour naviguer entre les différents transparents.

Sub-slide : est une continuité du transparent précédent. Il faut utiliser les flèches haut-bas pour naviguer.

Fragment : chaque cellule apparait progressivement sur le même transparent.

Skip : permet de ne pas afficher la cellule.

Notes : permet d'afficher des notes uniquement pour le présentateur.

Pour générer la présentation, il faut utiliser la commande suivante :

```
1 jupyter nbconvert monnotebook.ipynb --to slides --post  
  serve
```

--post serve permet de lancer automatiquement un serveur et exécute l'application revealjs avec les données transformées.

Il est possible de créer une page web qui soit lancée dès l'accès à un serveur. Pour cela, il faut :

- Renommer le fichier notebook.ipynb en index.ipynb pour générer un fichier index.html.
- Cloner reveal.js dans le répertoire où se situe le notebook :
git submodule add <https://github.com/hakimel/reveal.js.git> reveal.js
(<https://github.com/hakimel/reveal.js.git>) reveal.js
- Faire la conversion en spécifiant d'utiliser reveal.js:
jupyter nbconvert --to slides index.ipynb --reveal-prefix=reveal.js

Il est possible d'utiliser les options de nbconvert (voir https://nbconvert.readthedocs.io/en/latest/config_options.html (https://nbconvert.readthedocs.io/en/latest/config_options.html))

```
1 jupyter nbconvert mynotebook.ipynb --to slides --post serve  
2   --SlidesExporter.reveal_theme=sky  
3   --SlidesExporter.reveal_scroll=True  
4   --SlidesExporter.reveal_transition=none
```

va générer un fichier mynotebook.html pour lequel le theme est en sky (les différents theme sont disponibles : <https://github.com/hakimel/reveal.js/tree/master/css/theme> (<https://github.com/hakimel/reveal.js/tree/master/css/theme>)), autorisant le scroll lors de grande page, sans transition particulière et ouvre une fenêtre sur le navigateur pour lancer le fichier mynotebook.html.