Jupyter et ses notebooks

Les notebooks Jupyter sont des cahiers électroniques qui, dans le même document, peuvent rassembler du texte, des images, des formules mathématiques et du code informatique exécutable. Ils sont manipulables interactivement dans un navigateur web.

Initialement développés pour les langages de programmation Julia, Python et R (d'où le nom Jupyter), les notebooks Jupyter supportent près de 40 langages différents.

La cellule est l'élément de base d'un notebook Jupyter. Elle peut contenir du texte formaté au format Markdown ou du code informatique qui pourra être exécuté.

Voici un exemple de notebook Jupyter (figure .1) :

Ce notebook est constitué de cinq cellules : deux avec du texte en Markdown (la première et la dernière) et trois avec du code Python (notées avec `In [ ]`).

.1 Installation

Avec la distribution Miniconda, les notebooks Jupyter s'installent avec la commande :

```
$ conda install -y jupyterlab
```

Pour être exact, la commande précédente installe un peu plus que les notebooks Jupyter mais nous verrons cela par la suite.

.2 Lancement de Jupyter et création d'un notebook

Pour lancer les notebooks Jupyter, utilisez la commande suivante depuis un *shell* :

```
$ jupyter-notebook
```

Une nouvelle page devrait s'ouvrir dans votre navigateur web et ressembler à la figure .2.

Cette interface liste les notebooks Jupyter existants (pour le moment aucun).

Pour créer un notebook, cliquez sur le bouton à droite *New* puis sélectionnez *Python 3*. Vous noterez au passage qu'il est également possible de créer un fichier texte, un répertoire ou bien encore de lancer un *shell* via un *Terminal* (voir figure .3).

Le notebook fraîchement créé ne contient qu'une cellule vide.

La première chose à faire est de donner un nom à votre notebook en cliquant sur *Untitled*, à droite du logo de Jupyter. Si le nom de votre notebook est *test* alors le fichier *test.ipynb* sera créé dans le répertoire depuis lequel vous avez lancé Jupyter.

Remarque

L'extension *.ipynb* est l'extension de fichier des notebooks Jupyter.

Vous pouvez entrer des instructions Python dans la première cellule. Par exemple :

```
a = 2
b = 3
print(a+b)
```

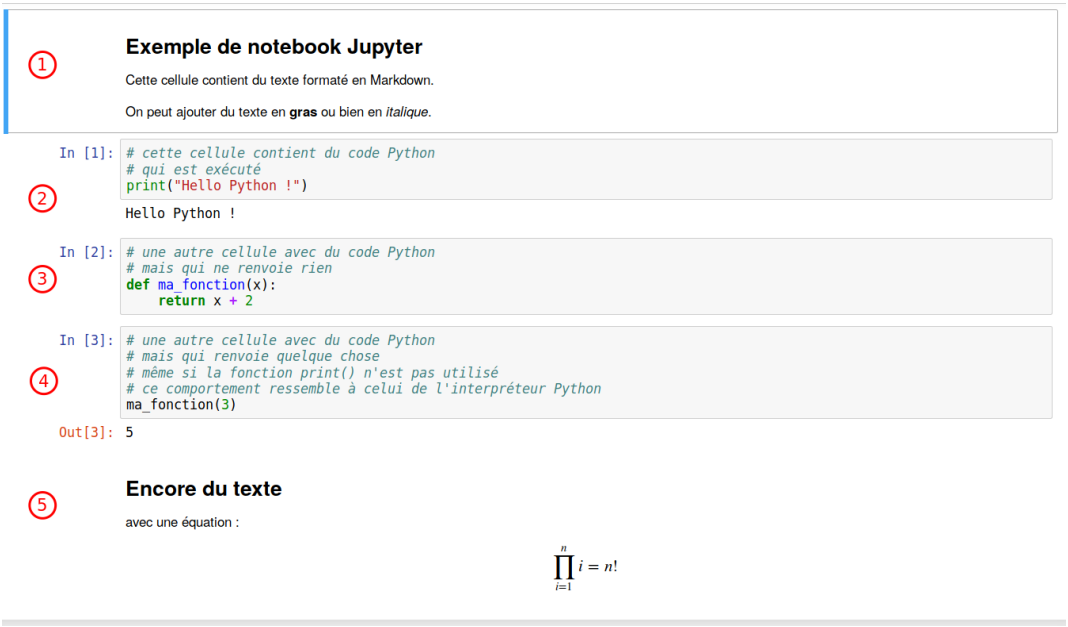

FIGURE 1 – Exemple de notebook Jupyter. Les chiffres entourés désignent les différentes cellules.

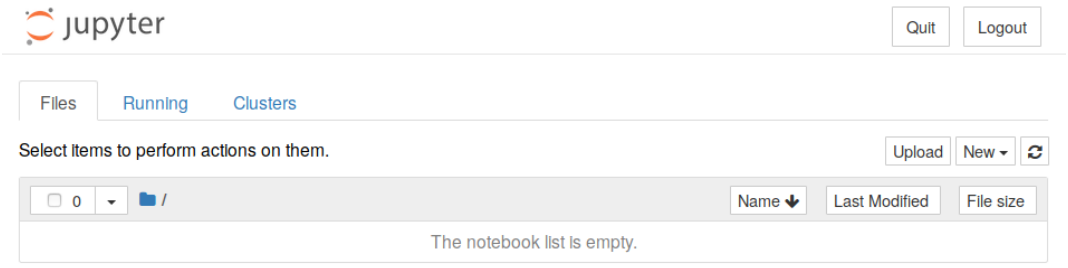

FIGURE 2 – Interface de Jupyter.

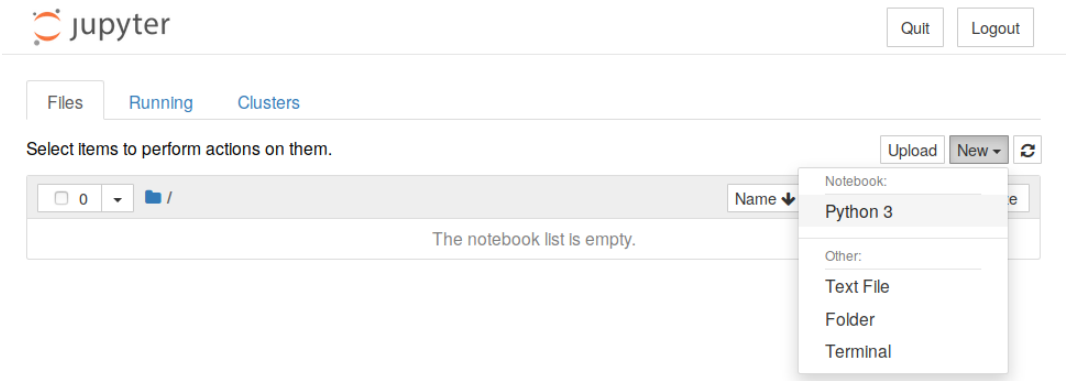

FIGURE 3 – Création d'un nouveau notebook.

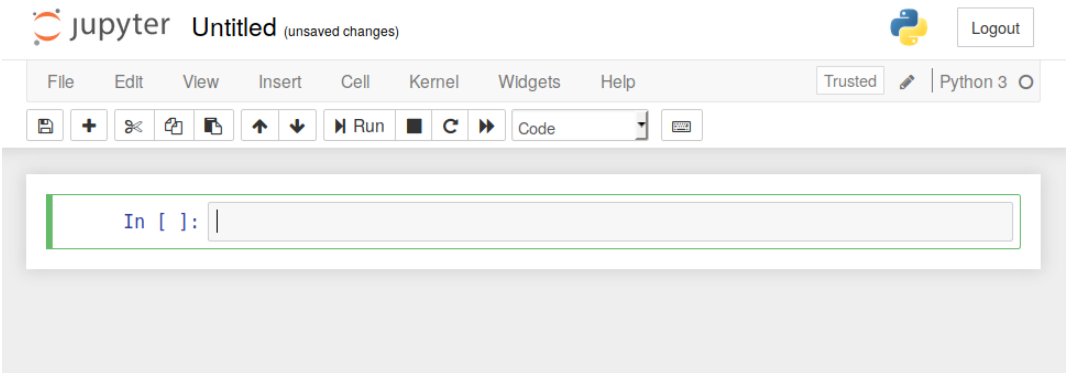

FIGURE 4 – Nouveau notebook.

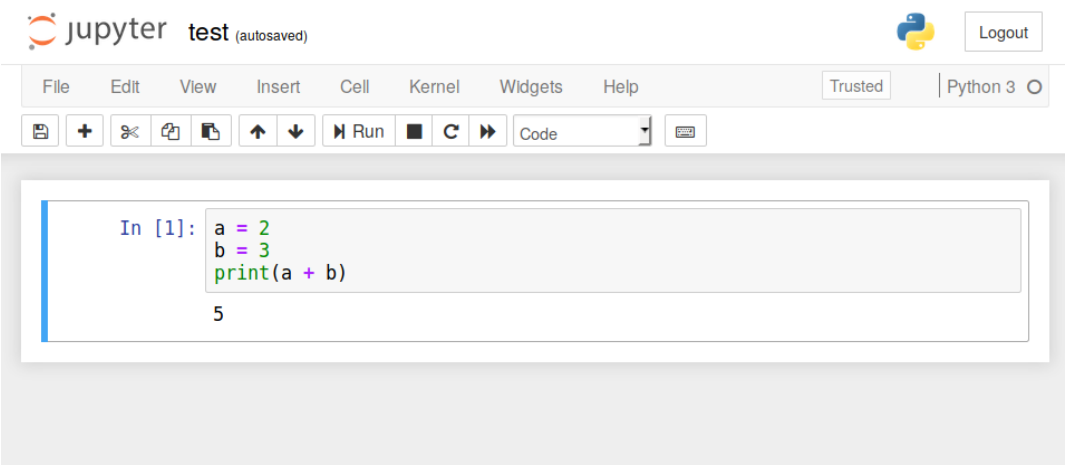

FIGURE 5 – Exécution d'une première cellule.

Pour exécuter le contenu de cette cellule, vous avez plusieurs possibilités :
— Cliquer sur le menu *Cell*, puis *Run Cells*.
— Cliquer sur le bouton *Run* (sous la barre de menu).
— Presser simultanément les touches *Ctrl* + *Entrée*.

Dans tous les cas, vous devriez obtenir quelque chose qui ressemble à l'image 5. La notation `In [1]` à gauche de la cellule indique qu'il s'agit de la première cellule exécutée.

Pour créer une nouvelle cellule, vous avez, ici encore, plusieurs possibilités :
— Cliquer sur l'icône + sous la barre de menu.
— Cliquer sur le menu *Insert*, puis *Insert Cell Below*.

Une nouvelle cellule vide devrait apparaître.

Vous pouvez également créer une nouvelle cellule en positionnant votre curseur dans la première cellule, puis en pressant simultanément les touches *Alt* + *Entrée*. Si vous utilisez cette combinaison de touches, vous remarquerez que le numéro à gauche de la première cellule est passée de `In [1]` à `In [2]` car vous avez exécutez la première cellule *puis* créez une nouvelle cellule.

Vous pouvez ainsi créer plusieurs cellules les unes à la suite des autres. Un objet créé dans une cellule antérieure sera disponible dans les cellules suivantes. Par exemple, dans la figure .6, nous avons quatre cellules. Vous remarquerez que pour les cellules 3 et 4, le résultat renvoyé par le code Python est précédé par `Out [3]` et `Out [4]`.

Dans un notebook Jupyter, il est parfaitement possible de réexécuter une cellule précédente. Par exemple la première cellule, qui porte désormais à sa gauche la numérotation `In [5]` (voir figure .7).

Attention

La possibilité d'exécuter les cellules d'un notebook Jupyter dans un ordre arbitraire peut prêter à confusion, notamment si vous modifiez la même variable d'une cellule à l'autre.

Nous vous recommandons de régulièrement relancer complètement l'exécution de toutes les cellules de votre notebook, de la première à la dernière, en cliquant sur le menu *Kernel* puis *Restart & Run All* et enfin de valider le message *Restart and Run All Cells*.

3 Le format Markdown

Dans le tout premier exemple (figure .1), nous avons vu qu'il était possible de mettre du texte au format Markdown dans une cellule.

Il faut cependant indiquer à Jupyter que cette cellule est au format Markdown en cliquant sur *Code* sous la barre de menu puis en choisissant *Markdown*.

Le format Markdown permet de rédiger du texte formaté (gras, italique, liens, titres, images, formules mathématiques...) avec quelques balises très simples. Voici un exemple dans une notebook Jupyter (figure .8) et le rendu lorsque la cellule est

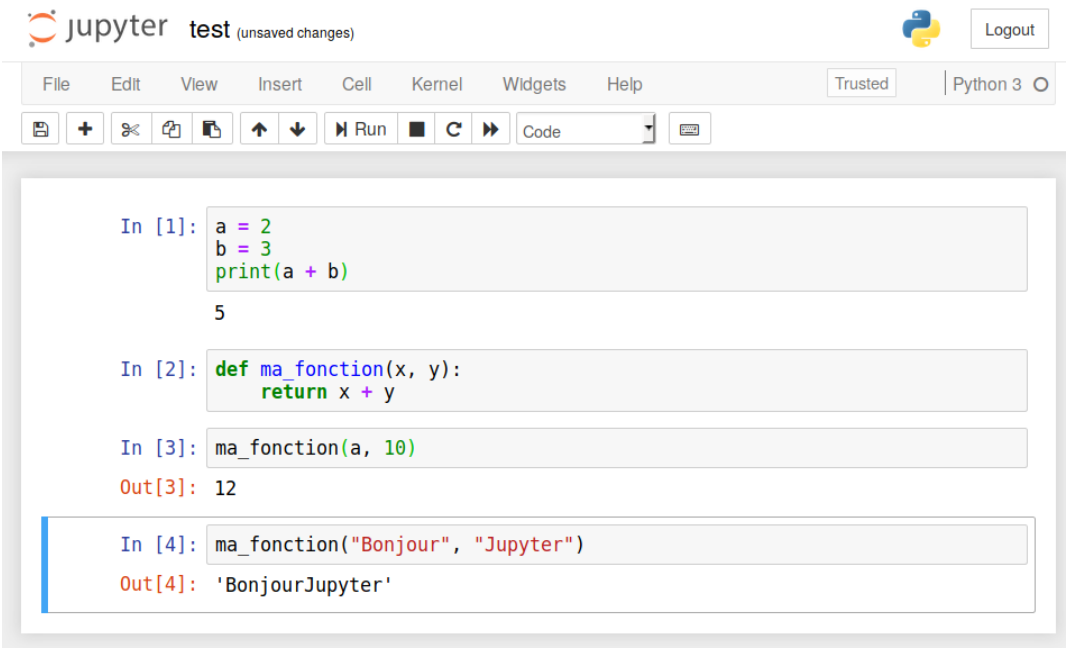

FIGURE .6 – Notebook avec plusieurs cellules de code Python.

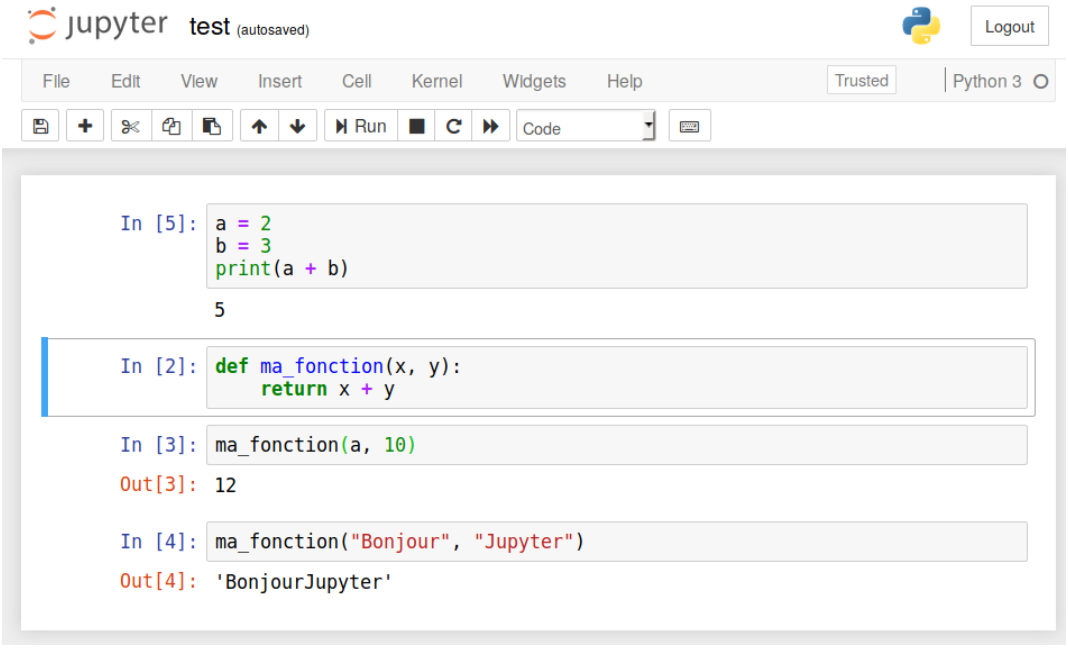

FIGURE .7 – Notebook avec une cellule ré-exécutée.

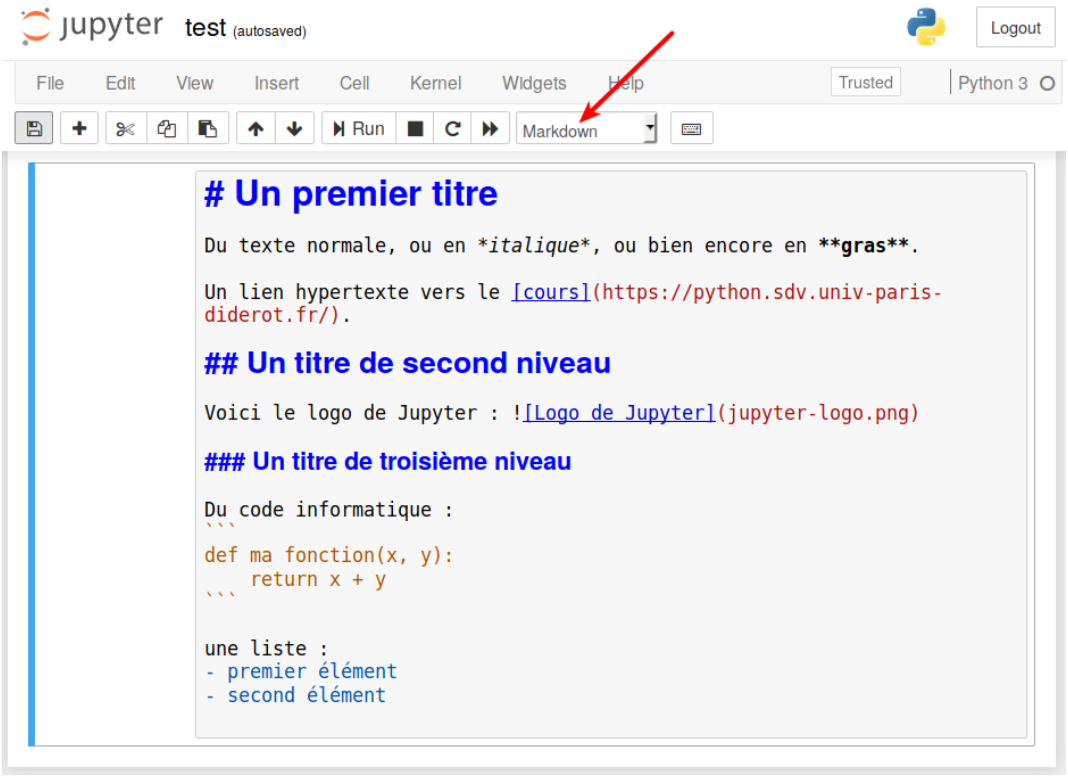

FIGURE .8 – Notebook avec une cellule au format Markdown.

exécutée (figure .9).

Notez qu'une cellule Markdown n'a pas le marqueur `In [ ]` à sa gauche.

Le format Markdown permet de rapidement et très simplement rédiger du texte structuré. Ce cours est par exemple complètement rédigé en Markdown ;-)

Nous vous conseillons d'explorer les possibilités du Markdown en consultant la page Wikipédia [1] ou directement la page de référence [2].

.4 Des graphiques dans les notebooks

Un autre intérêt des notebooks Jupyter est de pouvoir y incorporer des graphiques réalisés avec la bibliothèque *matplotlib*. Voici un exemple en reprenant un graphique présenté dans le 17 *Quelques modules d'intérêt en bioinformatique* (figure .10).

La différence notable est l'utilisation de la commande :

```
%matplotlib inline
```

qui n'est à lancer qu'une seule fois (en général dans la première cellule du notebook) et qui permet l'incorporation de figures dans un notebook Jupyter.

Remarque

Pour quitter l'interface des notebooks Jupyter, il faut, dans le premier onglet qui est apparu, cliquer sur le bouton *Quit* (figure .2).

Une méthode plus radicale est de revenir sur le *shell* depuis lequel les notebooks Jupyter ont été lancés puis de presser deux fois la combinaison de touches *Ctrl* + *C*.

1. https://fr.wikipedia.org/wiki/Markdown
2. https://daringfireball.net/projects/markdown/syntax

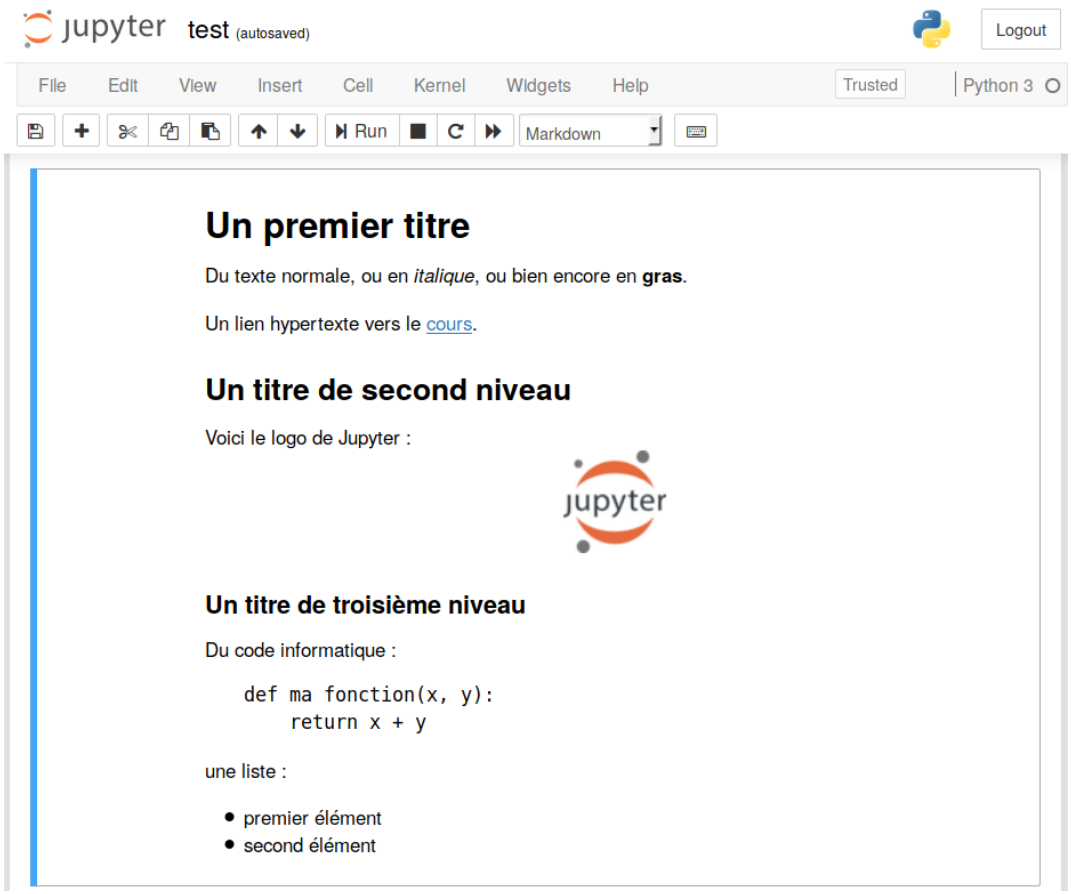

FIGURE .9 – Notebook avec une cellule au format Markdown (après exécution).

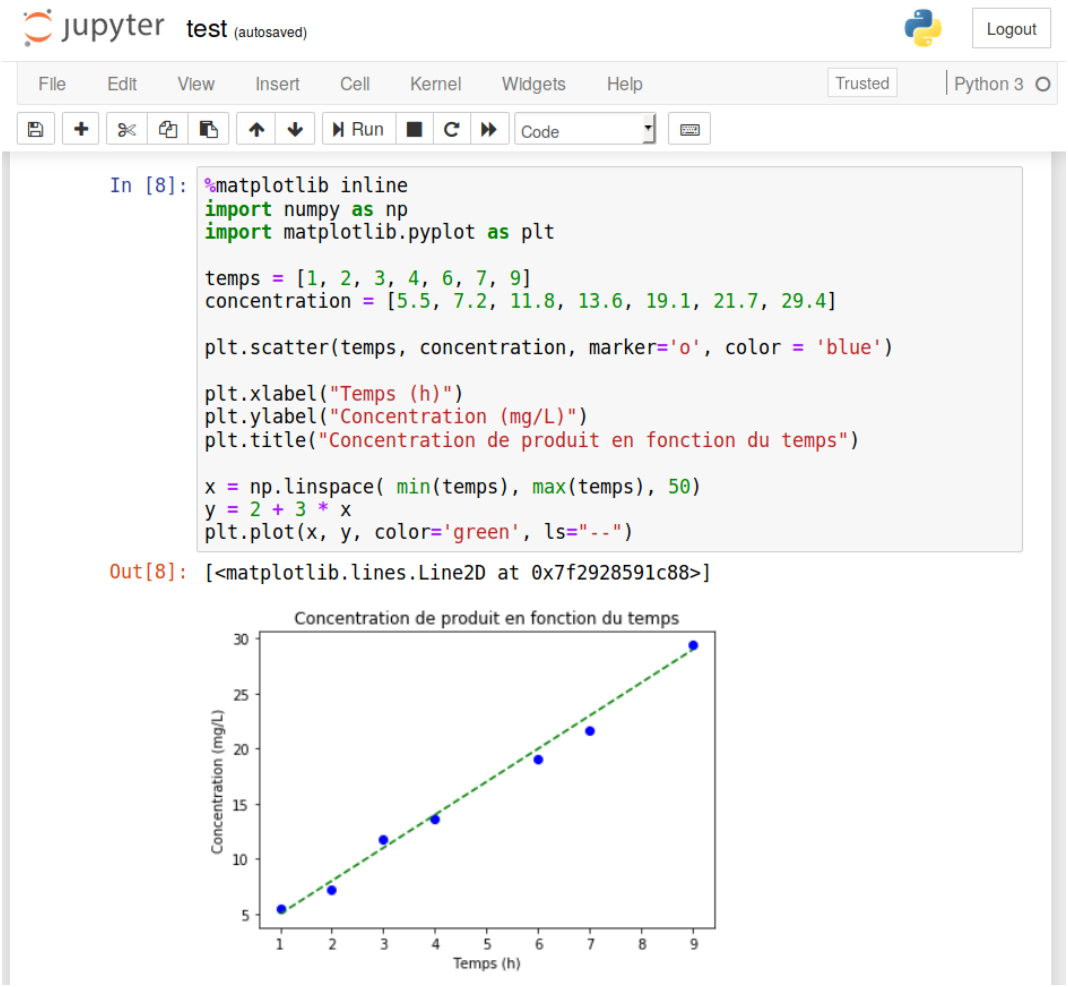

FIGURE .10 – Incorporation d'un graphique dans un notebook Jupyter.

```
In [9]: %whos
Variable        Type        Data/Info
-------------------------------------
a               int         2
b               int         3
concentration   list        n=7
ma_fonction     function    <function ma_fonction at 0x7f2950b46048>
np              module      <module 'numpy' from '/ho<...>kages/numpy/_
_init__.py'>
plt             module      <module 'matplotlib.pyplo<...>es/matplotlib
/pyplot.py'>
temps           list        n=7
x               ndarray     50: 50 elems, type `float64`, 400 bytes
y               ndarray     50: 50 elems, type `float64`, 400 bytes
```

FIGURE .11 – *Magic command* `%whos`.

.5 Les *magic commands*

La commande précédente (`%matplotlib inline`) est une *magic command*. Il en existe beaucoup, en voici deux :

— `%whos` liste tous les objets (variables, fonctions, modules...) utilisés dans le notebook (voir figure .11).

— `%history` liste toutes les commandes Python lancées dans un notebook (voir figure .12).

Enfin, avec les environnements Linux ou Mac OS X, il est possible de lancer une commande Unix depuis un notebook Jupyter. Il faut pour cela précéder la commande du symbole « ! ». La figure .13 illustre cette possibilité avec la commande `ls` qui affiche le contenu d'un répertoire.

.6 JupyterLab

En 2018, le consortium Jupyter a lancé *JupyterLab* qui est un environnement complet d'analyse. Pour obtenir cette interface, lancez la commande suivante depuis un *shell* :

```
$ jupyter lab
```

Une nouvelle page devrait s'ouvrir dans votre navigateur web et vous devriez obtenir une interface similaire à la figure .14.

L'interface proposée par JupyterLab est très riche. On peut y organiser un notebook Jupyter « classique » avec une figure en encart, un *shell* (voir figure .15)... Les possibilités sont infinies !

Pour aller plus loin

Les *notebooks* Jupyter sont particulièrement adaptés à l'analyse de données en combinaison avec les modules *matplotlib* et *pandas*.

```
In [7]: %history

        a = 2
        b = 3
        print(a + b)
        def ma_fonction(x, y):
            return x + y
        ma_fonction(a, 10)
        ma_fonction("Bonjour", "Jupyter")
        %matplotlib inline
        import numpy as np
        import matplotlib.pyplot as plt

        temps = [1, 2, 3, 4, 6, 7, 9]
        concentration = [5.5, 7.2, 11.8, 13.6, 19.1, 21.7, 29.4]

        plt.scatter(temps, concentration, marker='o', color = 'blue')

        plt.xlabel("Temps (h)")
        plt.ylabel("Concentration (mg/L)")
        plt.title("Concentration de produit en fonction du temps")

        x = np.linspace( min(temps), max(temps), 50)
        y = 2 + 3 * x
        plt.plot(x, y, color='green', ls="--")
        %whos
        %history
```

FIGURE .12 – *Magic command* %history.

```
In [12]: !ls
         jupyter-exemple.ipynb  jupyter-logo.png  test.ipynb
```

FIGURE .13 – Lancement d'une commande Unix.

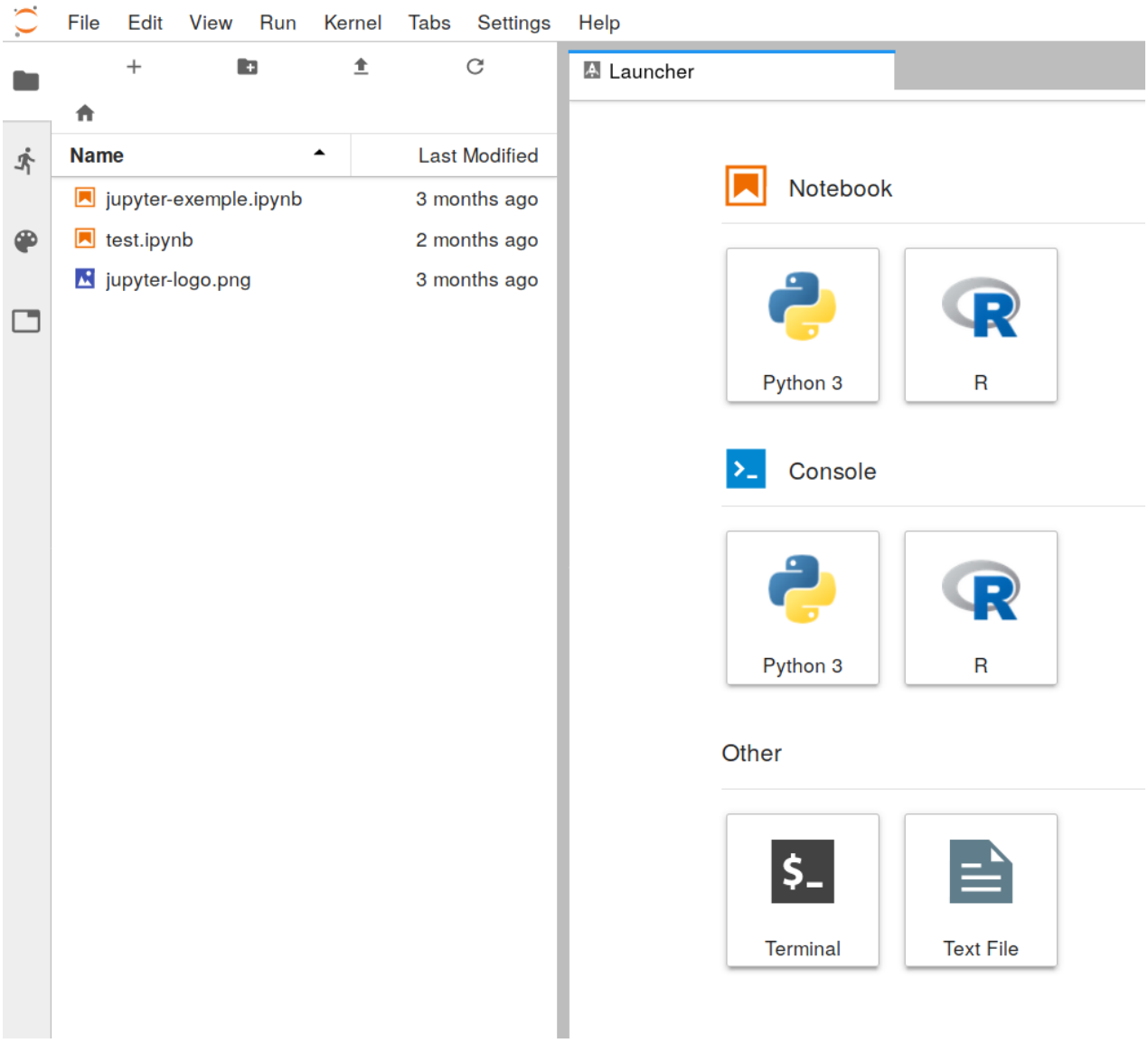

FIGURE .14 – Interface de JupyterLab.

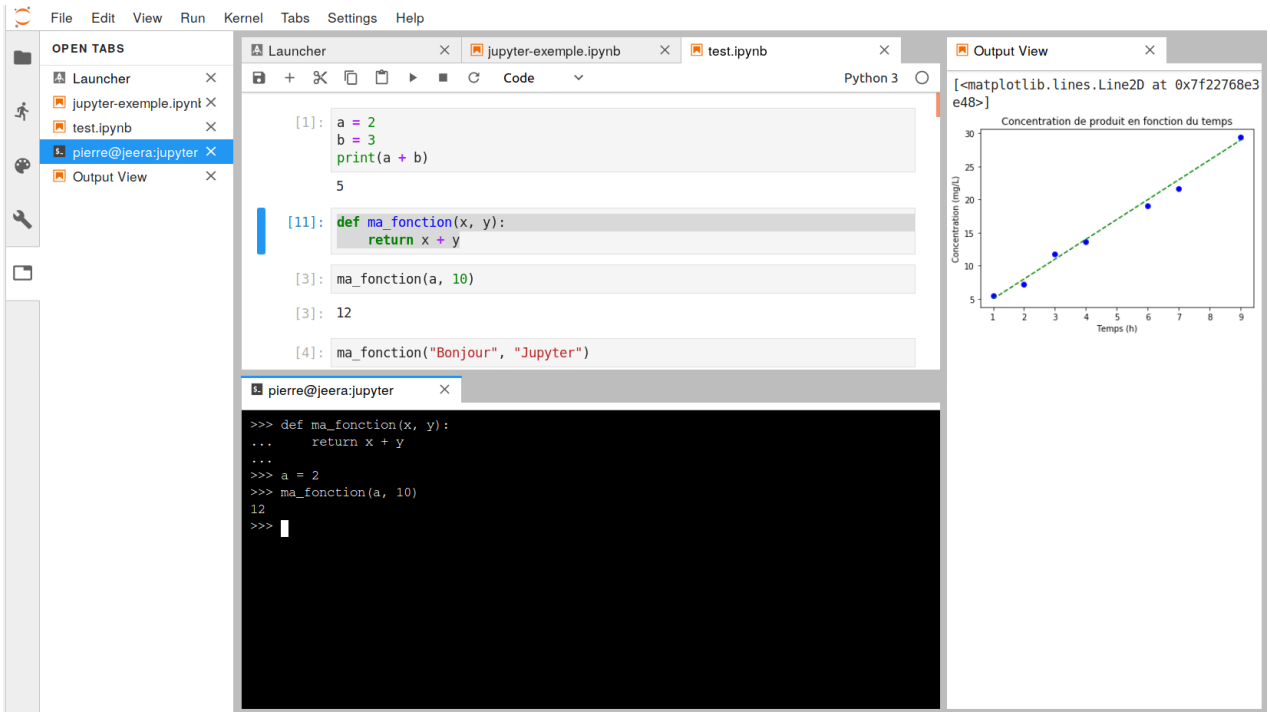

FIGURE .15 – JupyterLab comme environnement d'analyse.