pynotebook (0.1.0), with piton and pyluatex

1 Preamble

```
\documentclass{article}
\usepackage{pynotebook}
\usepackage[executable=python]{pyluatex}  % with a specific compilation !!
```

2 Examples of text blocks

```
\begin{NotebookPitonMarkdown}{\linewidth}
{\Large\bfseries This is a test for a \textsf{Markdown} block.}

It's possible to use \LaTeX{} formulas, like %
\[
  \left\lbrace\begin{array}{l}
    F_0 = 0\\
    F_1 = 1 \\
    F_{n+2} = F_{n+1} + F_n
  \end{array}\right.
\]
\end{NotebookPitonMarkdown}

\begin{NotebookPitonRaw}{\linewidth}
This is a sample block, with RAW output.

Just to use all capacities of Jupyter notebook ;-)
\end{NotebookPitonRaw}
```

This is a test for a Markdown block.

It's possible to use LaTeX formulas, like

$$\left\{\begin{array}{l} F_0 = 0 \\ F_1 = 1 \\ F_{n+2} = F_{n+1} + F_n \end{array}\right.$$

```
This is a sample block, with RAW output.

Just to use all capacities of Jupyter notebook ;-)
```

3 Examples of code blocks (with execution of code !)

```
\begin{NotebookPitonIn}{0.75\linewidth}
def fibonacci_of(n) :
  if n in {0,1} :
    return n
  return fibonacci_of(n-1) + fibonacci_of(n-2)

[fibonacci_of(n) for n in range(15)]
\end{NotebookPitonIn}
```

In [1]:

```
def fibonacci_of(n) :
    if n in {0,1} :
        return n
    return fibonacci_of(n-1) + fibonacci_of(n-2)

[fibonacci_of(n) for n in range(15)]
```

```
\begin{NotebookPitonOut}{0.75\linewidth}
def fibonacci_of(n) :
  if n in {0,1} :
    return n
  return fibonacci_of(n-1) + fibonacci_of(n-2)

print([fibonacci_of(n) for n in range(15)])
\end{NotebookPitonOut}
```

```
Out [1]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
```

```
\SetJupyterLng{fr}
\SetJupyterParSkip{\baselineskip}
\setcounter{JupyterIn}{11}
```

```
\begin{NotebookPitonIn}[center]{0.75\linewidth}
def fibonacci_of(n) :
  if n in {0,1} :
    return n
  return fibonacci_of(n-1) + fibonacci_of(n-2)

[fibonacci_of(n) for n in range(15)]
\end{NotebookPitonIn}
```

Entrée[15]:

```
1 def fibonacci_of(n) :
2     if n in {0,1} :
3         return n
4     return fibonacci_of(n-1) + fibonacci_of(n-2)
5
6 [fibonacci_of(n) for n in range(15)]
```

```
\begin{NotebookPitonOut}[center]{0.75\linewidth}
def fibonacci_of(n) :
  if n in {0,1} :
    return n
  return fibonacci_of(n-1) + fibonacci_of(n-2)

print([fibonacci_of(n) for n in range(15)])
\end{NotebookPitonOut}
```

Sortie[15]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]

```
\begin{NotebookPitonConsole}[center]{0.75\linewidth}
def fibonacci_of(n) :
  if n in {0,1} :
    return n
  return fibonacci_of(n-1) + fibonacci_of(n-2)

print([fibonacci_of(n) for n in range(15)])
\end{NotebookPitonConsole}
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
```

4 Global example

This is a test for a **Markdown** block.

It's possible to use LaTeX formulas, like

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_{n+2} = F_{n+1} + F_n \end{cases}$$

```
This is a sample block, with RAW output.

Just to use all capacities of Jupyter notebook ;-)
```

Entrée [1]:

```
def fibonacci_of(n) :
    if n in {0,1} :
        return n
    return fibonacci_of(n-1) + fibonacci_of(n-2)

[fibonacci_of(n) for n in range(15)]
```

Sortie [1]:

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
```

```
Let's compute Fibonacci terms from 10th to 20th :-)
```

Entrée [2]:

```
[fibonacci_of(n) for n in range(10,21)]
```

```
[55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765]
```