pynotebook

Present a Jupyter notebook, with tcolorbox, and listings or piton/pyluatex.

Version 0.1.0 - 15/02/2024

Cédric Pierquet
c pierquet - at - outlook . fr
https://github.com/cpierquet/pynotebook

Contents

1 Samples, with listings

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 ;-)
```

In [1]:

```python
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)]
```

Out [1]:

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

2 History

v0.1.0 : Initial version

3 The package pynotebook

3.1 Ideas

The idea is to provides environments to reproduce a Jupyter notebook :

- with *blocks* for RAW or Markdown ;
- with `listings` and no limitation with compiler, but without code execution ;
- with `piton` and `pyluatex` with LuaLaTeX and –shell-escape.

The documentation use pdfLaTeX, but examples with LuaLaTeX are given in an other doc.

3.2 Loading

The package loads within the preamble, with `\usepackage{pynotebook}`.
The loaded packages are `tcolorbox` (with `skins,breakable,listings`), `calc`, `xstring` and `iftex`.
If LuaLaTeX is detected, `piton` is loaded (but there's an option to avoid the loading), whereas `pyluatex` needs to be manually loaded, due to the declaration of the executable.

```
%with pdflatex
\usepackage{pynotebook}
```

```
%with LuaLaTeX and piton
\usepackage{pynotebook}
\usepackage[options]{pyluatex}
```

```
%with LuaLaTeX but without piton capability
\usepackage[nopiton]{pynotebook}
```

3.3 Global usage

In order to respect the left-alignment, the *titles* `In [ ]` and `Out[ ]` can add a blank character, to avoid offset due to counter with two digits !

4 Common text blocks

4.1 Intro

The different text blocks are given with their own output.
The package provides environments :

- for a RAW block, with `teletype` font ; for a Mardown block, with all LaTeX support ;
- a version with piton is given, in order to align perfectly the blocks !

```
\begin{NotebookRaw}[options tcbox]{<width>}
<code>
\end{NotebookRaw}
```

```
\begin{NotebookMarkdown}[options tcbox]{<width>}
<code>
\end{NotebookMarkdown}
```

```
\begin{NotebookPitonRaw}[options tcbox]{<width>}
<code>
\end{NotebookPitonRaw}
```

```
\begin{NotebookPitonMarkdown}[options tcbox]{<width>}
<code>
\end{NotebookPitonMarkdown}
```

4.2 Examples

```
\begin{NotebookMarkdown}{\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{NotebookMarkdown}

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

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

This is a test for a Markdown block.
It's possible to use LaTeX formulas, like

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

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

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

5 The code blocks, with listings

5.1 Intro

With `listings`, the different blocks are given with their own output (no code execution).
The package provides environments :

- with `In [...]` ;
- with `Out[...]` ;
- without *header*, eg for a *console execution*.

```
\begin{NotebookIn}(*)[options tcbox]{<width>}
<code>
\end{NotebookIn}
```

```
\begin{NotebookOut}(*)[options tcbox]{<width>}
<code>
\end{NotebookOut}
```

```
\begin{NotebookConsole}[options tcbox]{<width>}
<code>
\end{NotebookConsole}
```

The starred versions removes the counter, and don't display it.
The blocks with *header* (`In`/`Out`) are automatically numbered, and the global style is fixed.

5.2 Examples

```
\begin{NotebookIn}{\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{NotebookIn}

\begin{NotebookOut}{\linewidth}
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
\end{NotebookOut}

\begin{NotebookConsole}{\linewidth}
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
\end{NotebookConsole}
```

In [2]:

```
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)]
```

Out [2]:

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

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

```
\begin{NotebookIn}*[flush right]{13cm}
def fibonacci_of(n) :
  if n in {0,1} :
    return n
  return fibonacci_of(n-1) + fibonacci_of(n-2)

\end{NotebookIn}

\begin{NotebookOut}*[flush right]{13cm}
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
\end{NotebookOut}

\begin{NotebookConsole}[flush right]{13cm}
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
\end{NotebookConsole}
```

```
In [  ]: 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)]
```

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

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

6 The code blocks, with piton and pyluatex

6.1 Intro

With `piton` and `pyluatex`, the different blocks are given with the code to be displayed (`In/Out`) or with the code to be executed (Out or Console).
The package provides environments :

- with `In [...]` ;
- with `Out[...]` ;
- without *header*, eg for a *console execution*.

```
\begin{NotebookPitonIn}(*)[options tcbox]{<width>}
<code>
\end{NotebookPitonIn}
```

```
\begin{NotebookPitonOut}(*)[options tcbox]{<width>}
<code>
\end{NotebookPitonOut}
```

```
\begin{NotebookPitonConsole}[options tcbox]{<width>}
<code>
\end{NotebookPitonConsole}
```

The starred versions removes the counter, and don't display it.
The blocks with *header* (`In/Out`) are automatically numbered, and the global style is fixed.

6.2 Examples

Due to the necessary usage of LuaLaTeX and –shell-escape, examples are given in a separate file.

7 Some customization

7.1 Ideas

The package provides to macro, in order to :

- configure the *words* `In/Out` in french ;
- configure the spacing before and after the boxes (`0.33\baselineskip` by default).

```
\SetJupyterLng{fr}           %set french words

\SetJupyterParSkip{<length>} %modify space before/after (or default)

\setcounter{JupyterIn}{<nb>} %modify the counter
```

7.2 Examples

```
\SetJupyterLng{fr}
\SetJupyterParSkip{\baselineskip}
\setcounter{JupyterIn}{14}

\begin{NotebookIn}{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{NotebookIn}

\begin{NotebookOut}{0.75\linewidth}
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
\end{NotebookOut}
```

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