

pynotebook

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

Version 0.1.3 - 30/03/2024

Cédric Pierquet

c pierquet - at - outlook . fr

<https://github.com/cpierquet/pynotebook>

Contents

1	Samples, with listings	2
2	History	3
3	The package pynotebook	4
3.1	Ideas	4
3.2	Loading	4
3.3	Global usage	4
4	Common text blocks	5
4.1	Intro	5
4.2	Examples	5
5	The code blocks, with listings	6
5.1	Intro	6
5.2	Examples	6
6	The code blocks, with piton and pyluatex	8
6.1	Intro	8
6.2	Examples	8
7	Some customization	9
7.1	Ideas	9
7.2	Examples	9

Thanks to F. Pantigny for his package piton and his help. <https://ctan.org/pkg/piton>
And thanks to Tobias Enderle for his package pyluatex. <https://ctan.org/pkg/pyluatex>

1 Samples, with listings

This is a test for a Markdown block.

It's possible to use L^AT_EX formulas, like

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

```
1 This is a sample block, with RAW output.
2
3 Just to use all capacities of Jupyter notebook ;-)
```

```
In [1]: 1 def fibonacci_aux(n,a,b):
2         if n == 0 :
3             return a
4         elif n == 1 :
5             return b
6         else:
7             return fibonacci_aux(n-1,b,a+b)
8
9 def fibonacci_of(n):
10         return fibonacci_aux(n,0,1)
11
12 print([fibonacci_of(n) for n in range(10)])
```

```
Out [1]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

```
1 Now were going to work with lists.
2
3 Just a little example with a prime numbers, in french.
```

```
In [2]: 1 def proc_exec() :
2         choix = "o"
3         while choix == "o" :
4             n = -1
5             while n <= 1 :
6                 n = int(input("Saisir un entier n, supérieur à 2 : "))
7                 if estpremier(n) == True :
8                     print(f"{n} est premier.")
9                 else :
10                    print(f"{n} n'est pas premier.")
11                    listeres = listenombrepremiers(n)
12                    print(f"La liste des entiers premiers <= à {n} est {listeres}.")
13                    print(f"Il y a donc {len(listeres)} entiers premiers <= à {n}.")
14                    choix = input("Recommencer [o/n] ? ")
```

```
In [3]: 1 proc_exec()
```

```
Saisir un entier n, supérieur à 2 : 14
14 n'est pas premier.
La liste des entiers premiers <= à 14 est [2, 3, 5, 7, 11, 13].
Il y a donc 6 entiers premiers <= à 14.
Recommencer [o/n] ? o
Saisir un entier n, supérieur à 2 : 1
Saisir un entier n, supérieur à 2 : -3
Saisir un entier n, supérieur à 2 : 25
25 n'est pas premier.
La liste des entiers premiers <= à 25 est [2, 3, 5, 7, 11, 13, 17, 19, 23].
Il y a donc 9 entiers premiers <= à 25.
Recommencer [o/n] ? n
```

2 History

- v0.1.3 : Modification with `gobble` (for `piton`)
- v0.1.2 : Enhancements with `gobble` (for `piton`)
- v0.1.1 : New block `ln/Out` with `piton/pyluatex` (tks to F. Pantigny)
- v0.1.0 : Initial version

3 The package pynotebook

3.1 Ideas

The idea is to provide 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 Lua \LaTeX and `-shell-escape`.

The documentation use pdf \LaTeX , but examples with Lua \LaTeX 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 Lua \LaTeX 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 Markdown block, with all L^AT_EX support ;
- a version with `piton` is given, in order to align perfectly the blocks !

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

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

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

```
\begin{NotebookPitonMarkdown}[options tcheckbox]{<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 %
\[
  \left\lbracket\begin{array}{l}
    F_0 = 0 \\
    F_1 = 1 \\
    F_{n+2} = F_{n+1} + F_n
  \end{array}\right.
\]
\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 L^AT_EX formulas, like

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

```
1 This is a sample block, with RAW output.
2
3 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 tcbbox]{<width>}
<code>
\end{NotebookIn}
```

```
\begin{NotebookOut}[*] [options tcbbox]{<width>}
<code>
\end{NotebookOut}
```

```
\begin{NotebookConsole} [options tcbbox]{<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_aux(n,a,b):
    if n == 0 :
        return a
    elif n == 1 :
        return b
    else:
        return fibonacci_aux(n-1,b,a+b)

def fibonacci_of(n):
    return fibonacci_aux(n,0,1)

[ fibonacci_of(n) for n in range(10) ]
\end{NotebookIn}

\begin{NotebookOut}{\linewidth}
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
\end{NotebookOut}

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

```
In [4]: 1 def fibonacci_aux(n,a,b):
2         if n == 0 :
3             return a
4         elif n == 1 :
5             return b
6         else:
7             return fibonacci_aux(n-1,b,a+b)
8
9 def fibonacci_of(n):
10         return fibonacci_aux(n,0,1)
11
12 [fibonacci_of(n) for n in range(10)]
```

```
Out [4]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

```
\begin{NotebookIn}*[flush right]{13cm}
def fibonacci_aux(n,a,b):
    if n == 0 :
        return a
    elif n == 1 :
        return b
    else:
        return fibonacci_aux(n-1,b,a+b)

def fibonacci_of(n):
    return fibonacci_aux(n,0,1)

[fibonacci_of(n) for n in range(10)]
\end{NotebookIn}
```

```
\begin{NotebookOut}*[flush right]{13cm}
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
\end{NotebookOut}
```

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

```
In [ ]: 1 def fibonacci_aux(n,a,b):
2         if n == 0 :
3             return a
4         elif n == 1 :
5             return b
6         else:
7             return fibonacci_aux(n-1,b,a+b)
8
9 def fibonacci_of(n):
10         return fibonacci_aux(n,0,1)
11
12 [fibonacci_of(n) for n in range(10)]
```

```
Out [ ]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

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

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

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

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

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

```
\begin{NotebookPitonInOut}(*)[options tcbbox]{width}<gobble options>
<code>
\end{NotebookPitonInOut}
```

```
\begin{NotebookPitonConsole}[options tcbbox]{width}<gobble options>
<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.

`gobble options` are given within `piton` syntax :

- `auto-gobble` ;
- `env-gobble` ;
- `gobble=xx` ;
- `tabs-auto-gobble` ;
- a mix of them.

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 macros, 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_aux(n,a,b):
    if n == 0 :
        return a
    elif n == 1 :
        return b
    else:
        return fibonacci_aux(n-1,b,a+b)

def fibonacci_of(n):
    return fibonacci_aux(n,0,1)

[ 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]
\end{NotebookOut}
```

```
Entrée[15]: 1 def fibonacci_aux(n,a,b):
2             if n == 0 :
3                 return a
4             elif n == 1 :
5                 return b
6             else:
7                 return fibonacci_aux(n-1,b,a+b)
8
9 def fibonacci_of(n):
10            return fibonacci_aux(n,0,1)
11
12 [ fibonacci_of(n) for n in range(10) ]
```

```
Sortie[15]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```