

Jupyter notebook custom conversion

Romain Madar

August 2018

Contents

1	nbconvert latex test	2
2	Printing using python	2
3	Pyout (and Text Wrapping)	3
4	Image and plots	5
4.1	As plain text using markdown	5
4.2	Plots produced by the code	6
5	Operator Highlighting Check	9
6	Tables	10
6.1	Markdown as plain text	10
6.2	Pandas as default and Markdown	10
7	Sympy output	13
8	Line Length	13

1 nbconvert latex test

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc luctus bibendum felis dictum sodales. Ut suscipit, orci ut interdum imperdiet, purus ligula mollis *justo*, non malesuada nisl augue eget lorem. Donec bibendum, erat sit amet porttitor aliquam, urna lorem ornare libero, in vehicula diam diam ut ante. Nam non urna rhoncus, accumsan elit sit amet, mollis tellus. Vestibulum nec tellus metus. Vestibulum tempor, ligula et vehicula rhoncus, sapien turpis faucibus lorem, id dapibus turpis mauris ac orci. Sed volutpat vestibulum venenatis.

This is a test list:

1. item 1
 - subitem 1
 - subitem 2
2. item 2
3. item 3

2 Printing using python

```
next_paragraph = """
Aenean vitae diam consectetur, tempus arcu quis, ultricies urna. Vivamus
↪ venenatis sem
quis orci condimentum, sed feugiat dui porta.
"""

def identity_dec(ob):
    return ob

@identity_dec
def nifty_print(text):
    """Used to test syntax highlighting"""

    print(text * 2)

nifty_print(next_paragraph)
```

Aenean vitae diam consectetur, tempus arcu quis, ultricies urna. Vivamus
venenatis sem
quis orci condimentum, sed feugiat dui porta.

Aenean vitae diam consectetur, tempus arcu quis, ultricies urna. Vivamus
venenatis sem
quis orci condimentum, sed feugiat dui porta.

3 Pyout (and Text Wrapping)

```
Text = """
Aliquam blandit aliquet enim, eget scelerisque eros adipiscing quis. Nunc sed
↳ metus
ut lorem condimentum condimentum nec id enim. Sed malesuada cursus hendrerit.
↳ Praesent
et commodo justo. Interdum et malesuada fames ac ante ipsum primis in
↳ faucibus.
Curabitur et magna ante. Proin luctus tellus sit amet egestas laoreet. Sed
↳ dapibus
neque ac nulla mollis cursus. Fusce mollis egestas libero mattis facilisis.
"""
Text #Use print(Text) instead to get text wrapping in pdf
```

```
'\nAliquam blandit aliquet enim, eget scelerisque eros adipiscing quis. Nunc
sed metus \nut lorem condimentum condimentum nec id enim. Sed malesuada
cursus hendrerit. Praesent \net commodo justo. Interdum et malesuada fames ac
ante ipsum primis in faucibus. \nCurabitur et magna ante. Proin luctus tellus
sit amet egestas laoreet. Sed dapibus \nneque ac nulla mollis cursus. Fusce
mollis egestas libero mattis facilisis.\n'
```

```
print(Text)
```

Aliquam blandit aliquet enim, eget scelerisque eros adipiscing quis. Nunc sed
metus
ut lorem condimentum condimentum nec id enim. Sed malesuada cursus hendrerit.
Praesent
et commodo justo. Interdum et malesuada fames ac ante ipsum primis in
faucibus.
Curabitur et magna ante. Proin luctus tellus sit amet egestas laoreet. Sed
dapibus
neque ac nulla mollis cursus. Fusce mollis egestas libero mattis facilisis.

```
import numpy as np
```

```
a = np.random.rand(10,10)
```

```
print(a)
```

```
a
```

```
[[0.90466956 0.51427359 0.59960171 0.11940667 0.14231845 0.08103596
 0.56757406 0.05514305 0.97864219 0.09874248]
 [0.42981847 0.90735708 0.56881384 0.05849852 0.57152156 0.86621931
 0.32921324 0.30842128 0.56063931 0.05262495]
 [0.88870019 0.72530912 0.4876109 0.40415864 0.78580043 0.97324715
 0.79851827 0.90186061 0.52643866 0.87622605]
 [0.10029028 0.32284391 0.98056156 0.70430387 0.59692472 0.73414243
 0.38480575 0.53290454 0.18618216 0.04759213]
 [0.13808277 0.16401501 0.56757759 0.25866075 0.50247413 0.60030059
 0.50093564 0.27557969 0.28007316 0.03740728]
 [0.31763597 0.09440636 0.60255811 0.60010303 0.82887226 0.19723087
 0.21013864 0.25927747 0.9752489 0.9413291 ]
 [0.91718241 0.1882523 0.63560747 0.13691833 0.23424879 0.3058812
 0.63607707 0.25138014 0.11743956 0.08138231]
 [0.10104675 0.23881601 0.52650728 0.85805806 0.69742441 0.77739132
 0.97433189 0.39140405 0.34178549 0.966936 ]
 [0.4001062 0.86674045 0.80017885 0.88494963 0.65247756 0.20569015
 0.69799131 0.37480885 0.10765384 0.06541536]
 [0.34959811 0.22965998 0.64237429 0.15935986 0.10170083 0.10723806
 0.7477006 0.23640309 0.09760103 0.09873411]]
```

```
array([[0.90466956, 0.51427359, 0.59960171, 0.11940667, 0.14231845,
        0.08103596, 0.56757406, 0.05514305, 0.97864219, 0.09874248],
       [0.42981847, 0.90735708, 0.56881384, 0.05849852, 0.57152156,
        0.86621931, 0.32921324, 0.30842128, 0.56063931, 0.05262495],
       [0.88870019, 0.72530912, 0.4876109 , 0.40415864, 0.78580043,
        0.97324715, 0.79851827, 0.90186061, 0.52643866, 0.87622605],
       [0.10029028, 0.32284391, 0.98056156, 0.70430387, 0.59692472,
        0.73414243, 0.38480575, 0.53290454, 0.18618216, 0.04759213],
       [0.13808277, 0.16401501, 0.56757759, 0.25866075, 0.50247413,
        0.60030059, 0.50093564, 0.27557969, 0.28007316, 0.03740728],
       [0.31763597, 0.09440636, 0.60255811, 0.60010303, 0.82887226,
        0.19723087, 0.21013864, 0.25927747, 0.9752489 , 0.9413291 ],
       [0.91718241, 0.1882523 , 0.63560747, 0.13691833, 0.23424879,
```

```

0.3058812 , 0.63607707, 0.25138014, 0.11743956, 0.08138231],
[0.10104675, 0.23881601, 0.52650728, 0.85805806, 0.69742441,
0.77739132, 0.97433189, 0.39140405, 0.34178549, 0.966936 ],
[0.4001062 , 0.86674045, 0.80017885, 0.88494963, 0.65247756,
0.20569015, 0.69799131, 0.37480885, 0.10765384, 0.06541536],
[0.34959811, 0.22965998, 0.64237429, 0.15935986, 0.10170083,
0.10723806, 0.7477006 , 0.23640309, 0.09760103, 0.09873411]])

```

4 Image and plots

4.1 As plain text using markdown

Once exported as markdown and converted to latex/pdf with pandoc, the `{width=60%}` will fix the width of the picture and the `My legend` will appear as caption:

```

! [My legend] (figures/magnetostatics_field.png){width=50% #figlabel}

```

gives the result shown in [this figure](#).

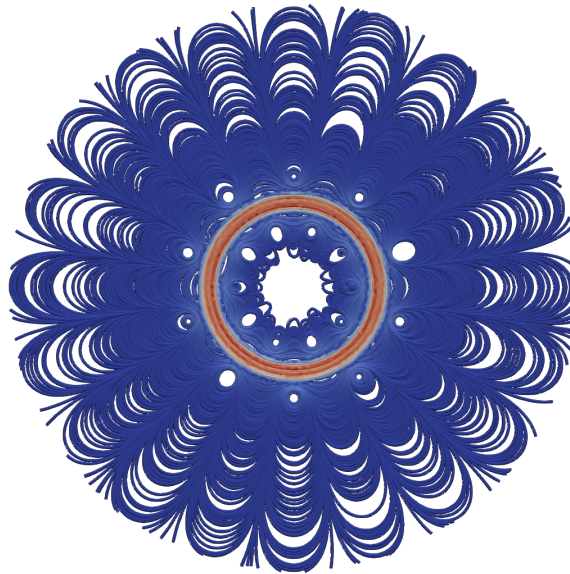


Figure 1: My legend

```

from IPython.core.display import Image
Image(data="http://ipython.org/_static/IPy_header.png")

```

IP[y]: IPython Interactive Computing

Figure 2: png

4.2 Plots produced by the code

```
import numpy as np
x = np.linspace(-10,10,300)
y = np.sin(x)
plt.figure(figsize=(4,3),dpi=100)
p=plt.plot(x,y)
```

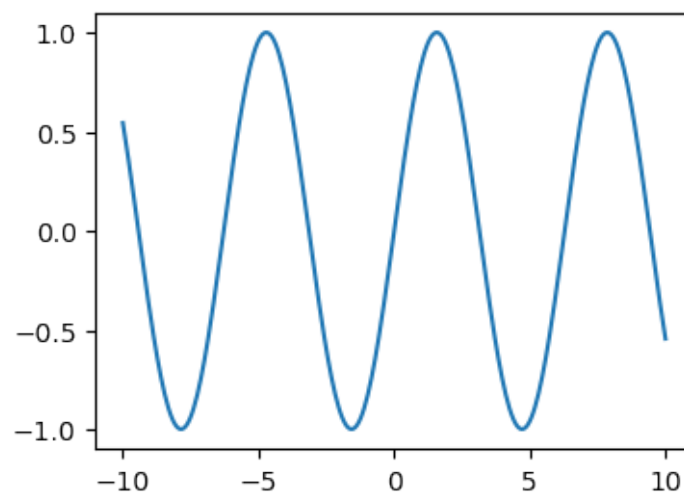


Figure 3: png

```
plt.figure(figsize=(8,3))
plt.plot(x,y)
plt.xlabel('$X_{e}$ [SI]')
plt.ylabel('$Y_{UPDATE12}$ [SI]')
plt.tight_layout()
jpu.plt2md('myplot','This is a test of how to get proper'+\
          'plot from Jupyter notebook in MD, '+\
          'to be processed using PANDOC','100%')
```

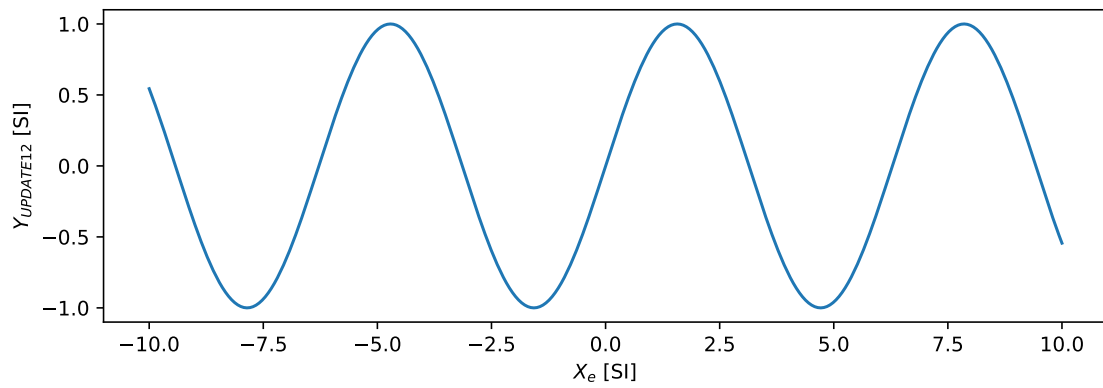


Figure 4: This is a test of how to get properplot from Jupyter notebook in MD,to be processed using PANDOC

We can then refer to a given figure using cross-references [like this](#), obtained with:

```
[like this] (#myplot)
```

```
plt.figure(figsize=(3,3))
plt.plot(np.sqrt(np.abs(x)),y**2)
plt.xlabel('$X_{e}$ [SI]')
plt.ylabel('$Y_{OTHER}$ [SI]')
plt.tight_layout()
jpu.plt2md('myplot2','This is another test of how to get proper'+\
          'plot from Jupyter notebook in MD,'+\
          'to be processed using PANDOC','50%')
```

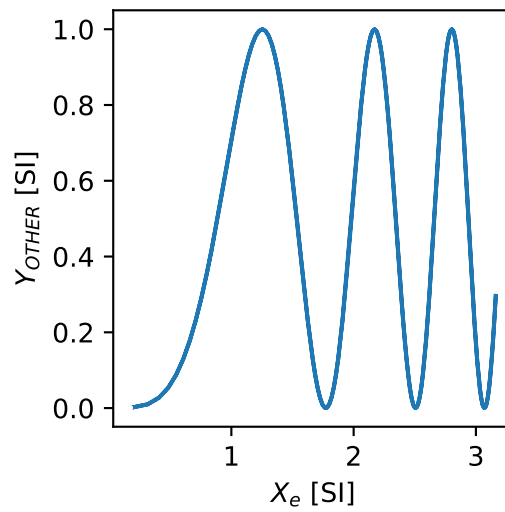


Figure 5: This is another test of how to get properplot from Jupyter notebook in MD,to be processed using PANDOC

```
plt.figure(figsize=(6,4))
plt.plot(x**3,y)
plt.plot(x**3,y+1)
plt.xlabel('test')
plt.ylabel('test2')
plt.tight_layout()
jpu.plt2md('myplot3','Adding more plot without re-creating a figure: curves
↳ cumulates','50%')
```

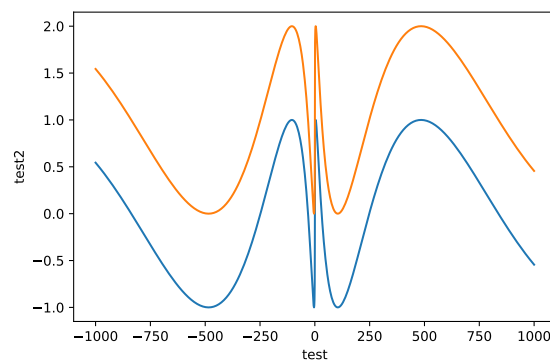


Figure 6: Adding more plot without re-creating a figure: curves cumulates

```
plt.figure(figsize=(6,3))
plt.plot(x**2,y)
```



```
plt.xlabel('$X_{\omega}$ [Hz]')
plt.ylabel('$Y_{\varphi}$ [$\hbar$]')
plt.title('My beautiul plot')
plt.tight_layout()
jpu.plt2md('myplot4', 'More plot with re-creating a figure: only last
↳ curve', '100%')
```

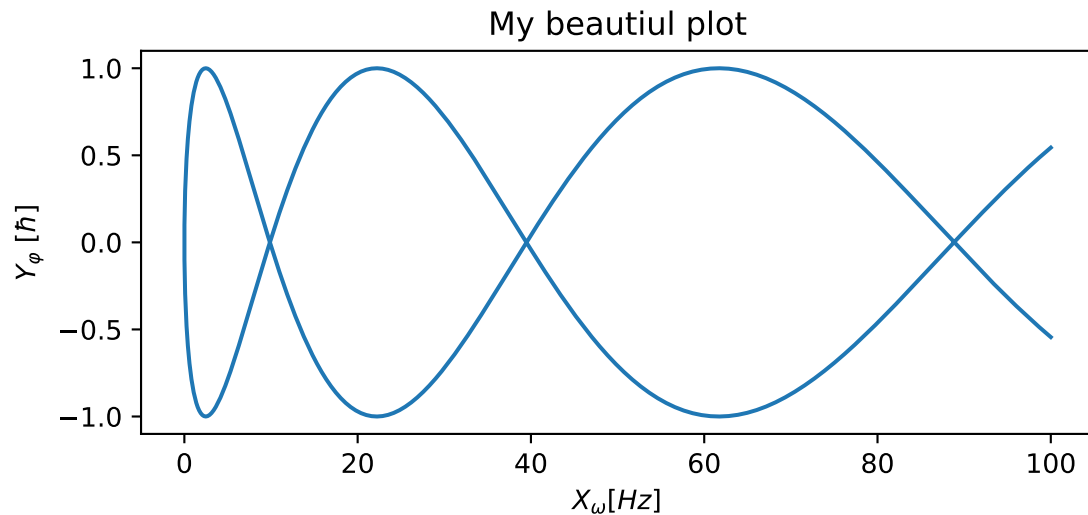


Figure 7: More plot with re-creating a figure: only last curve

5 Operator Highlighting Check

```
#This is a comment with an operation x @ y in it.
test = 5**9 + 2 - x @ y / (7 % 2) + True * 7
print(test)

a = set([1,2,3,4,5,6,7,8,9,0])
b = set([2,4,6,8,0])
a & b
```

1952904.9236703357

{0, 2, 4, 6, 8}

6 Tables

6.1 Markdown as plain text

First a *markdown* [table](#):

Table 1: my caption

Column 1	Column 2
1	3
a	b
4	&

6.2 Pandas as default and Markdown

```
import pandas as pd
df=pd.DataFrame(np.random.randn(10,3))
```

Default printing is HTML, so it looks good on the web but it is not well rendered in pdf via ipynb->MB->pdf (using nbconvert and pandoc). A special function `df2md(df)` is included in the `jupy_pandoc_utils` package to write out a markdown format. This allows to set caption and even to refer to the table in the main document [like this](#) (using vanilla pandoc) or cited like table Tbl. 2 (using pandoc-crossref filter, but hyperlink doesn't seem to work in HTML though)

```
# Good in HTML, but not pure markdown
# Impossible to put a caption
df.describe()
```

```
0
1
2
count
10.000000
```

10.000000

10.000000

mean

0.297303

-0.770918

0.039299

std

1.276659

0.570637

0.830404

min

-1.346284

-1.500445

-1.798771

25%

-0.843660

-1.223944

-0.091879

50%

0.061822

-0.841636

0.215294

75%

1.276288

-0.489515

0.607642

max

2.279334

0.159813

0.929222

```
# Good in pure MD and possible to put a caption and a label
jpu.df2md(df.describe(), 'Caption table', '#tbl:label2')
```

Table 2: Caption table

labels	0	1	2
count	10	10	10
mean	0.297303	-0.770918	0.0392989
std	1.27666	0.570637	0.830404
min	-1.34628	-1.50044	-1.79877
25%	-0.84366	-1.22394	-0.0918792
50%	0.0618221	-0.841636	0.215294
75%	1.27629	-0.489515	0.607642
max	2.27933	0.159813	0.929222

One might want to display table without showing the code which leads to it, especially in a proper documentation. The following block will display the table in the final document but not the line

```
jpu.df2md(df.describe(), 'Caption table with hidden source code this time.
→  {#tbl:label3}')
```

which produces it, as shown in Tbl. 3

Table 3: Caption table with hidden source code this time

labels	0	1	2
count	10	10	10
mean	0.297303	-0.770918	0.0392989
std	1.27666	0.570637	0.830404
min	-1.34628	-1.50044	-1.79877
25%	-0.84366	-1.22394	-0.0918792
50%	0.0618221	-0.841636	0.215294
75%	1.27629	-0.489515	0.607642
max	2.27933	0.159813	0.929222

7 Sympy output

```
import sympy
from sympy.abc import x, n, m
sympy.init_printing()
theta = sympy.Symbol('theta')
phi = sympy.Symbol('phi')

sympy.simplify(sympy.Ynm(n,m,theta,phi).expand(func=True))
```

$$\frac{\sqrt{\frac{(2n+1)\Gamma(-m+n+1)}{\Gamma(m+n+1)}} e^{im\phi} P_n^{(m)}(\cos(\theta))}{2\sqrt{\pi}}$$

x + y as plain text.

$$\frac{P_n^{(m)}(\cos(\theta))}{2\sqrt{\pi}} \sqrt{\frac{(-m+n)!}{(m+n)!}} (2n+1) e^{im\phi}$$

8 Line Length

```
1 3 5 7 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75
↪ 78 81 84 87 90 93 96 99 103
```