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 Highlighing Check **12**

6 Tables **13**

6.1 Markdown as plain text 13

6.2 Pandas as default and Markdown 13

7 Sympy output **16**

8 Cell tags **16**

8.1 No tag at all 16

8.2 Tag `hide` 17

8.3 Tag `hide_input` 17

8.4 Tag `hide_ouput` 17

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
```

```
[[0.62700725 0.26081733 0.95340885 0.21671471 0.75910717 0.40181619
  0.45623474 0.3285357  0.85356038 0.24432894]
 [0.81476144 0.61562474 0.4591993  0.3250914  0.61241612 0.40349597
  0.07681138 0.0754312  0.79515141 0.55149073]
 [0.87729401 0.31116951 0.41637531 0.21537057 0.49408607 0.71757091
  0.95260321 0.04917473 0.78805643 0.47774209]
 [0.50899341 0.31887905 0.2065325  0.74680579 0.92657773 0.69588066
  0.07029994 0.85724744 0.98815397 0.04235898]
 [0.53924064 0.681954   0.98901054 0.2708235  0.49798494 0.7033097
  0.12139082 0.9582635  0.78622896 0.27142782]
 [0.42561997 0.72058827 0.71853415 0.39014238 0.77526377 0.27103878
  0.88255786 0.87605381 0.30814998 0.93653828]
 [0.99548486 0.48758976 0.29550014 0.17276068 0.77922546 0.98184958
  0.90418671 0.1117684  0.4899384  0.1587398 ]
 [0.42252942 0.88736155 0.77008558 0.4014361  0.15536484 0.43598829
  0.61792713 0.04761771 0.25242741 0.786188  ]
 [0.76391558 0.00739227 0.35831757 0.1117959  0.04947653 0.42891628
  0.71117318 0.569816   0.19804252 0.29726679]
 [0.37986381 0.94883542 0.31790772 0.31700223 0.41713151 0.79208272
  0.17414851 0.23828334 0.21169432 0.77084355]]
```

```
array([[0.62700725, 0.26081733, 0.95340885, 0.21671471, 0.75910717,
        0.40181619, 0.45623474, 0.3285357 , 0.85356038, 0.24432894],
       [0.81476144, 0.61562474, 0.4591993 , 0.3250914 , 0.61241612,
        0.40349597, 0.07681138, 0.0754312 , 0.79515141, 0.55149073],
       [0.87729401, 0.31116951, 0.41637531, 0.21537057, 0.49408607,
        0.71757091, 0.95260321, 0.04917473, 0.78805643, 0.47774209],
       [0.50899341, 0.31887905, 0.2065325 , 0.74680579, 0.92657773,
        0.69588066, 0.07029994, 0.85724744, 0.98815397, 0.04235898],
       [0.53924064, 0.681954  , 0.98901054, 0.2708235 , 0.49798494,
        0.7033097 , 0.12139082, 0.9582635 , 0.78622896, 0.27142782],
       [0.42561997, 0.72058827, 0.71853415, 0.39014238, 0.77526377,
        0.27103878, 0.88255786, 0.87605381, 0.30814998, 0.93653828],
       [0.99548486, 0.48758976, 0.29550014, 0.17276068, 0.77922546,
```

```
 0.98184958, 0.90418671, 0.1117684 , 0.4899384 , 0.1587398 ],
[0.42252942, 0.88736155, 0.77008558, 0.4014361 , 0.15536484,
 0.43598829, 0.61792713, 0.04761771, 0.25242741, 0.786188  ],
[0.76391558, 0.00739227, 0.35831757, 0.1117959 , 0.04947653,
 0.42891628, 0.71117318, 0.569816  , 0.19804252, 0.29726679],
[0.37986381, 0.94883542, 0.31790772, 0.31700223, 0.41713151,
 0.79208272, 0.17414851, 0.23828334, 0.21169432, 0.77084355]])
```

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 showns in this figure.

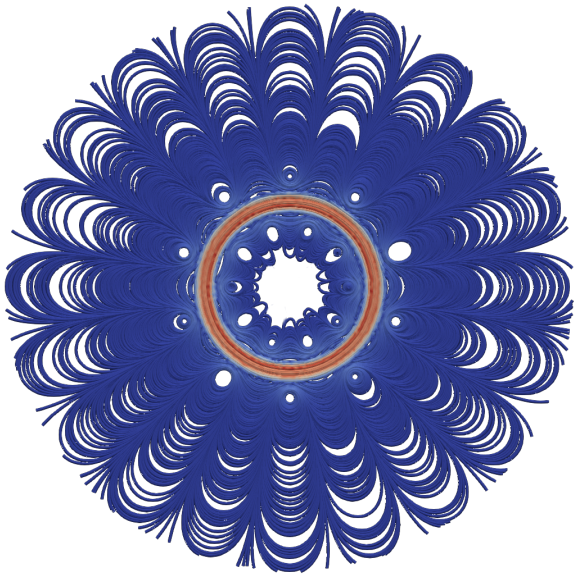

Figure 1: My legend

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

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)
```

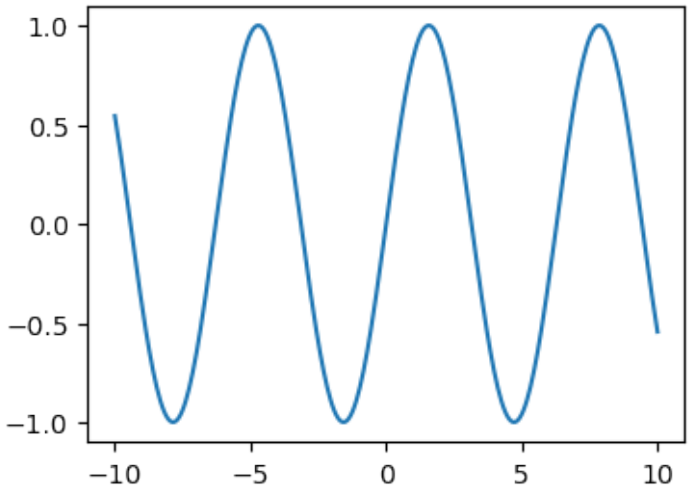

Figure 3: png

```
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%','myplot_LABEL')
```

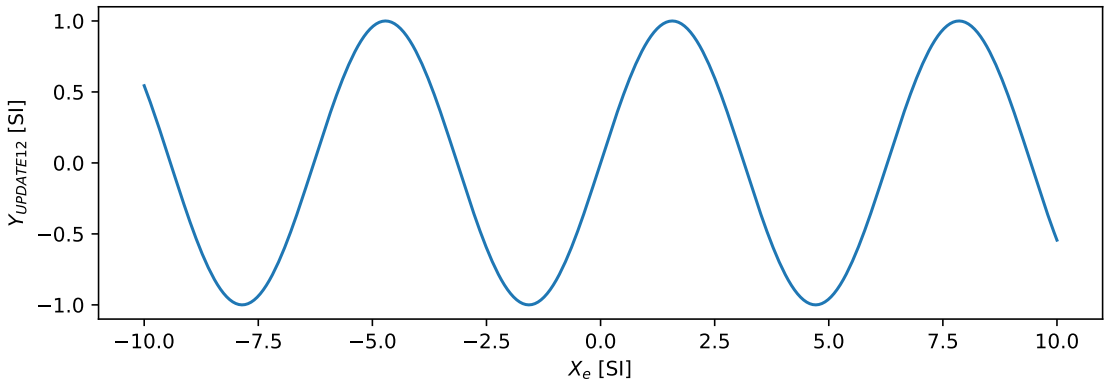

Figure 4: This is a test of how to get properplot from Jupyter notebook in MD,to be processed using PANDOC

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%','myplot2_LABEL')
```

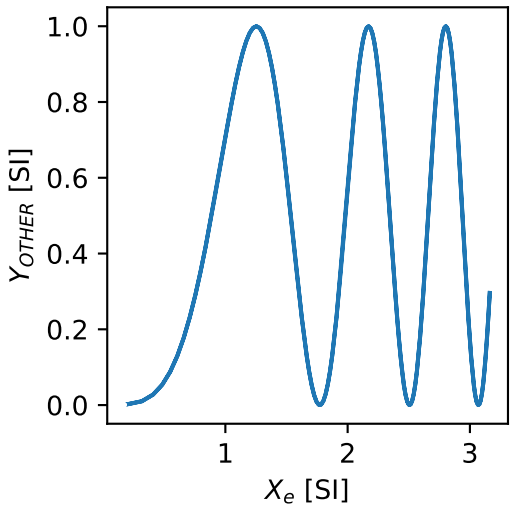

Figure 5: This is another test of how to get properplot from Jupyter notebook in MD,to be processed using PANDOC

```
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')
```

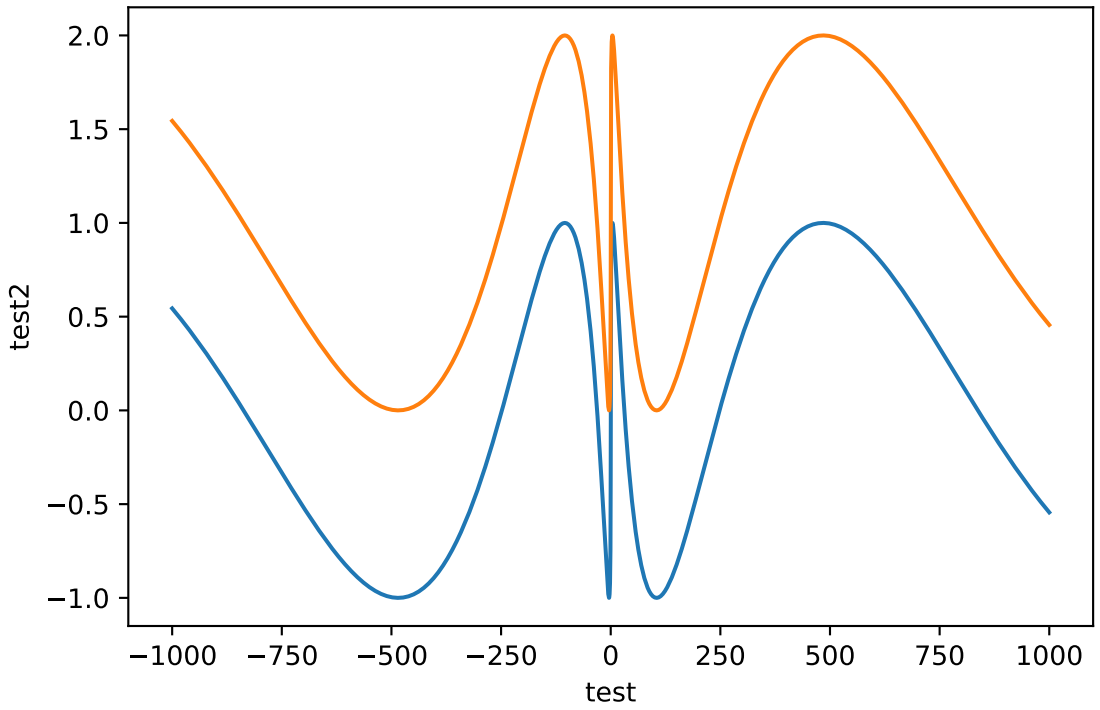

Figure 6: Adding more plot without re-creating a figure: curves cumulates

```
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%')
```

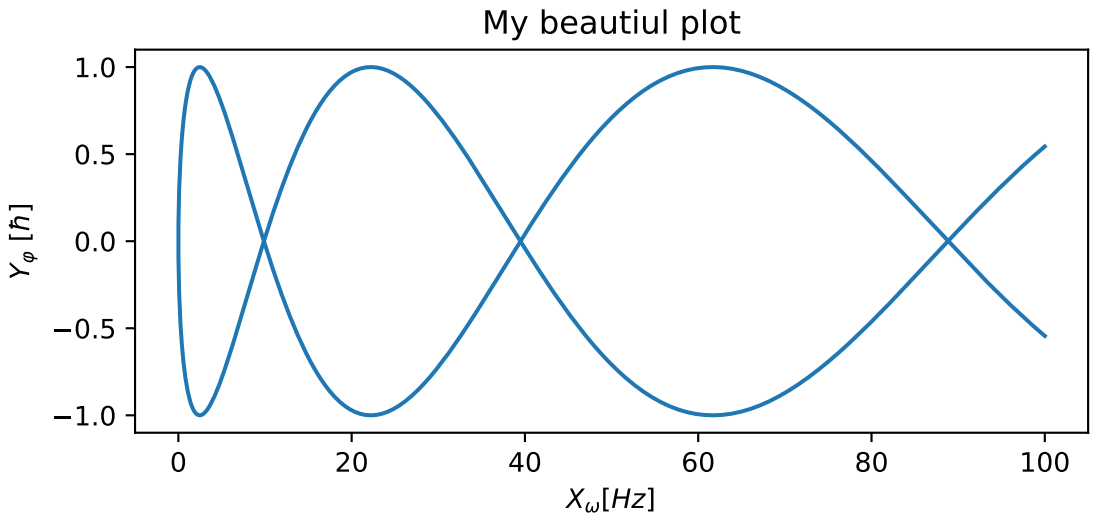

Figure 7: More plot with re-creating a figure: only last curve

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

# Fixing random state for reproducibility
np.random.seed(19680801)


# the random data
x = np.random.randn(1000)
y = np.random.randn(1000)


fig, axScatter = plt.subplots(figsize=(5.5, 5.5))

# the scatter plot:
axScatter.scatter(x, y)
axScatter.set_aspect(1.)

# create new axes on the right and on the top of the current axes
# The first argument of the new_vertical(new_horizontal) method is
# the height (width) of the axes to be created in inches.
divider = make_axes_locatable(axScatter)
axHistx = divider.append_axes("top", 1.2, pad=0.1, sharex=axScatter)
axHisty = divider.append_axes("right", 1.2, pad=0.1, sharey=axScatter)
```

```
# make some labels invisible
axHistx.xaxis.set_tick_params(labelbottom=False)
axHisty.yaxis.set_tick_params(labelleft=False)

# now determine nice limits by hand:
binwidth = 0.25
xymax = max(np.max(np.abs(x)), np.max(np.abs(y)))
lim = (int(xymax/binwidth) + 1)*binwidth

bins = np.arange(-lim, lim + binwidth, binwidth)
axHistx.hist(x, bins=bins)
axHisty.hist(y, bins=bins, orientation='horizontal')

# the xaxis of axHistx and yaxis of axHisty are shared with axScatter,
# thus there is no need to manually adjust the xlim and ylim of these
# axis.

axHistx.set_yticks([0, 50, 100])
axHisty.set_xticks([0, 50, 100])

jpu.plt2md('scatter','Test of a scatter plot taken'+
           'from [matplotlib
            ↪  webpage](https://matplotlib.org/gallery/axes_grid1/scatter_hist_locatable_axe
           '#sphx-glr-gallery-axes-grid1-scatter-hist-locatable-axes-py) '+
           '(and yes, markdown links work in the caption too)','100%')
```

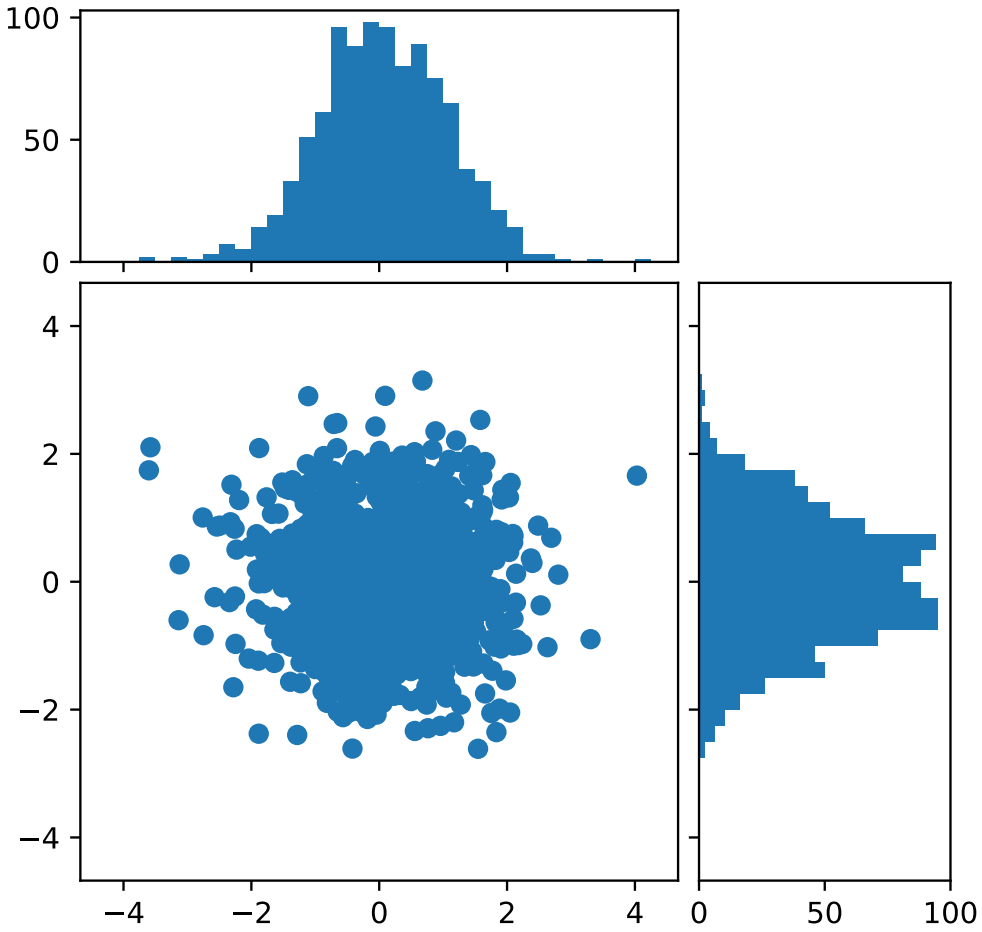

Figure 8: Test of a scatter plot takenfrom matplotlib webpage (and yes, markdown links work in the caption too)

5 Operator Highlighing 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 Table 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

-0.124437

-0.193223

0.086777

std

0.865165

0.856129

0.811450

min

-1.667288

-1.287436

-1.705184

25%

-0.570475

-0.762087

-0.242516

50%

-0.040271

-0.230659

0.453195

75%

0.438126

0.007493

0.646274

max

1.100231

1.791935

0.828771

```
# 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	-0.124437	-0.193223	0.0867771
std	0.865165	0.856129	0.81145
min	-1.66729	-1.28744	-1.70518
25%	-0.570475	-0.762087	-0.242516
50%	-0.0402713	-0.230659	0.453195
75%	0.438126	0.00749318	0.646274
max	1.10023	1.79194	0.828771

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 Table 3

Table 3: Caption table with hidden source code this time

labels	0	1	2
count	10	10	10
mean	-0.124437	-0.193223	0.0867771
std	0.865165	0.856129	0.81145
min	-1.66729	-1.28744	-1.70518
25%	-0.570475	-0.762087	-0.242516
50%	-0.0402713	-0.230659	0.453195

labels	0	1	2
75%	0.438126	0.00749318	0.646274
max	1.10023	1.79194	0.828771

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 Cell tags

The next three cells have different tags, used in `nbconverter_md_pandoc.tpl`

8.1 No tag at all

```
l='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'
print(l.split(' '))
l
```

['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']

'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'

8.2 Tag `hide`

8.3 Tag `hide_input`

['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']

'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'

8.4 Tag `hide_ouput`

```
l='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'
print(l.split(' '))
l
```