3_EditCells

October 17, 2023

0.1 Entering code

Entering code is pretty easy. You just have to click into a cell and type the commands you want to type. If you have multiple lines of code, just press **enter** at the end of the line and start a new one.

- **code blocks** Python identifies blocks of codes belonging together by its identation. This will become important if you write longer code in a cell later. To indent the block, you may use either *whitespaces* or *tabs*.
- **comments** Comments can be added to annotate the code, such that you or someone else can understand the code.
  - Comments in a single line are started with the `#` character at in front of the comment.
  - Comments over multiple lines can be started with `'''`and end with the same `'''`. They are used as `docstrings` to provide a help text.

[2]:
```python
# typical function

def function(x):
    ''' function to calculate a function
    arguments:
        x ... float or integer value
    returns:
        y ... two times the integer value
    '''
    y=2*x # don't forget the identation of the block
    return(y)
```

[3]:
```python
help(function)
```

```
Help on function function in module __main__:

function(x)
    function to calculate a function
    arguments:
        x … float or integer value
    returns:
        y … two times the integer value
```

0.2 Entering Markdown

Text can be added to Jupyter Notebooks using Markdown cells. This is extremely useful providing a complete documentation of your calculations or simulations. In this way, everything really becomes an notebook. You can change the cell type to Markdown by using the "Cell Actions" menu, or with a hardware keyboard shortcut `m`. Markdown is a popular markup language that is a superset of HTML. Its specification can be found here:

https://daringfireball.net/projects/markdown/

Markdown cells can either be **rendered** or **unrendered**.

When they are rendered, you will see a nice formatted representation of the cell's contents.

When they are unrendered, you will see the raw text source of the cell. To render the selected cell, click the button or `shift`+ `enter`. To unrender, select the markdown cell, and press `enter` or just double click.

0.2.1 Markdown basics

Below are some basic markdown examples, in its rendered form. If you which to access how to create specific appearances, double click the individual cells to put the into an unrendered edit mode.

You can make text *italic* or **bold**.

You can build nested itemized or enumerated lists:

- first item
    - first subitem
        - first subsubitem
    - second subitem - first subitem of second subitem - second subitem of second subitem
- second item
    - first subitem
- third item
    - first subitem

Now another list:

1. Here we go
    1. Sublist
        2. Sublist
2. There we go
3. Now this

Here is a blockquote:

> Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Flat is better than nested. Sparse is better than dense. Readability counts. Special cases aren't special enough to break the rules. Although practicality beats purity. Errors should never pass silently. Unless explicitly silenced. In the face of ambiguity, refuse the temptation to guess. There should be one– and preferably only one –obvious way to do it. Although that way may not be obvious at first unless you're Dutch. Now is better than never. Although never

is often better than *right* now. If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea – let's do more of those!

And shorthand for links:

Jupyter's website

0.2.2 Headings

You can add headings by starting a line with one (or multiple) # followed by a space, as in the following example:

```
# Heading 1
# Heading 2
## Heading 2.1
## Heading 2.2
### Heading 2.2.1
```

0.2.3 Embedded code

You can embed code meant for illustration instead of execution in Python:

```
def f(x):
    """a docstring"""
    return x**2
```

or other languages:

```
if (i=0; i<n; i++) {
  printf("hello %d\n", i);
  x += 4;
}
```

0.2.4 LaTeX equations

Courtesy of MathJax, you can include mathematical expressions both inline: $e^{i\pi} + 1 = 0$ and displayed:

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

Inline expressions can be added by surrounding the latex code with $:

```
$e^{i\pi} + 1 = 0$
```

Expressions on their own line are surrounded by $$:

```
$$e^x=\sum_{i=0}^\infty \frac{1}{i!}x^i$$
```

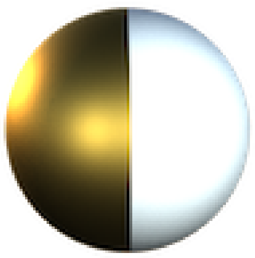
0.2.5 Images

Images may be also directly integrated into a Markdown block.

To include images use

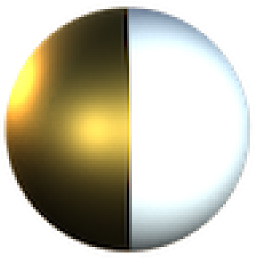
`![alternative text](url)`

for example

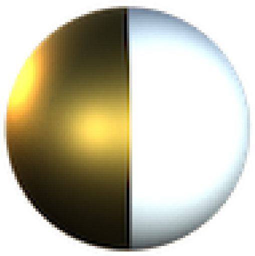


```
[4]: from IPython.display import Image, display;
```

```
[5]: url="https://github.com/fcichos/CompSoft23/raw/master/source/notebooks/Intro/
      ↪img/particle.png"
     display(Image(url))
```



```
[6]: display(Image(filename='img/particle.png'))
```



0.2.6 Videos

To include videos, we use HTML code like

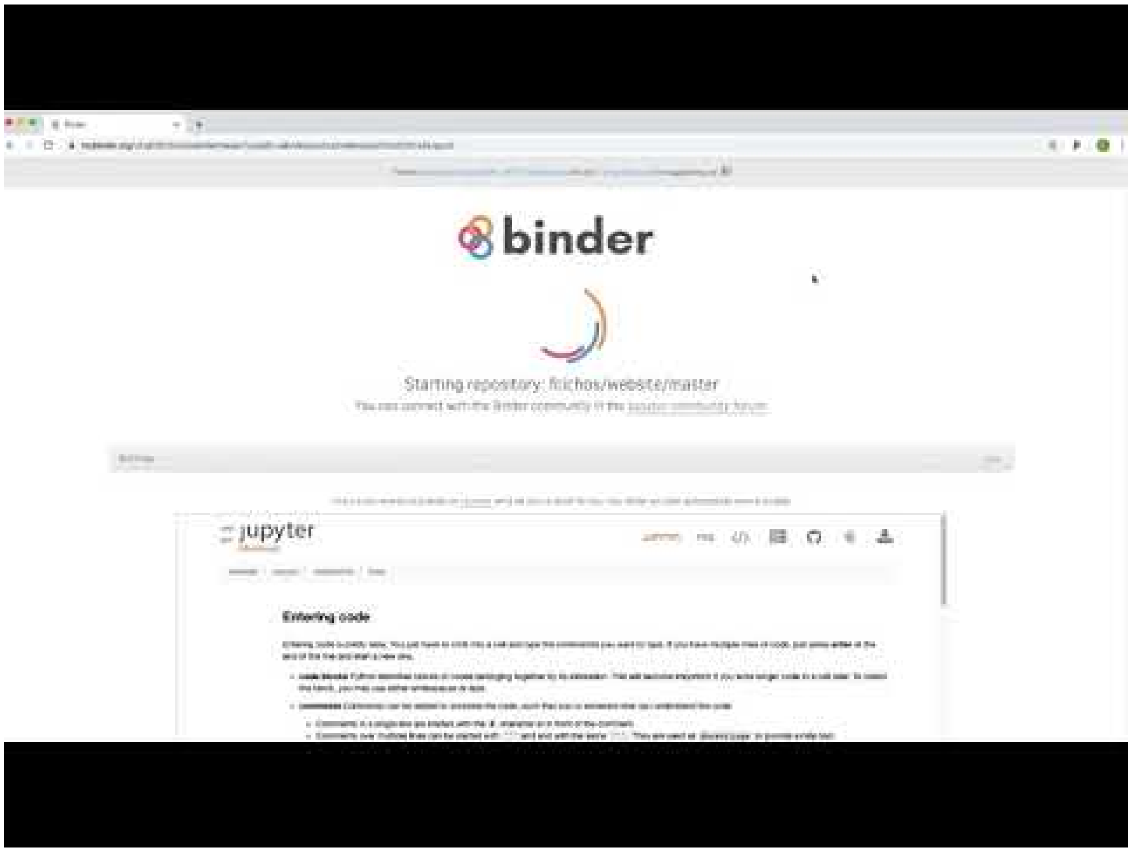
```
<video src="mov/movie.mp4" width="320" height="200" controls preload></video>
```

in the Markdown cell. This works with videos stored locally.

You can embed YouTube Videos as well by using the `IPython` module.

```
[9]: from IPython.display import YouTubeVideo
     YouTubeVideo('QlLx32juGzI',width=600)
```

```
[9]:
```



Starting repository: fiichos/website/master

You can connect with the Binder community in the jupyter community forum.

jupyter

Entering code

[illegible]