Jupyter notebook for scientific computing

I visited EuroSciPy 2017 and re-discovered Ipython in the form of the Jupyter notebook. This is really a good tool for interactive data analysis, and it seems to be the standard, at the moment for this kind of job.

Web sites

http://jupyter.readthedocs.io/en/latest/
http://www.scipy-lectures.org/intro/intro.html

Installation

Most easy is via Anaconda:
https://www.anaconda.com/download/

This installs a bunch of scientific Python libraries together with Jupyter.

Important:
To make the path to Anaconda automatically accessible, notice this question at the end of the installation procedure:

```
Do you wish the installer to prepend the Anaconda2 install location
to PATH in your /home/jcf/.bashrc ? [yes|no]
```

Answer yes here!

Test

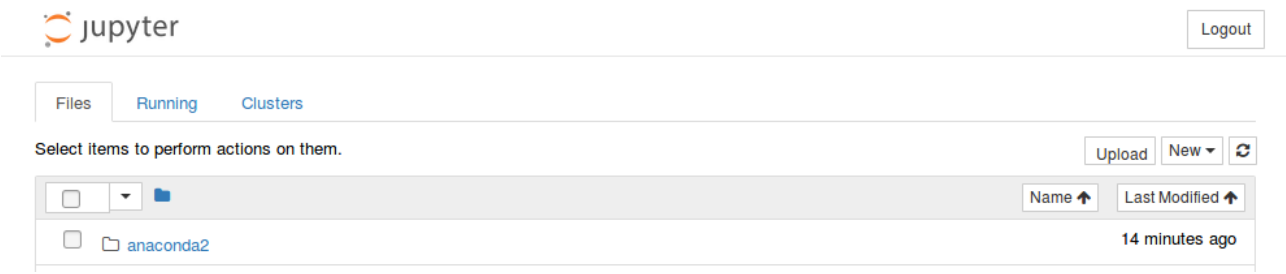
In the home folder, open a terminal and type:

```
jupyter notebook
```

There are messages like this:

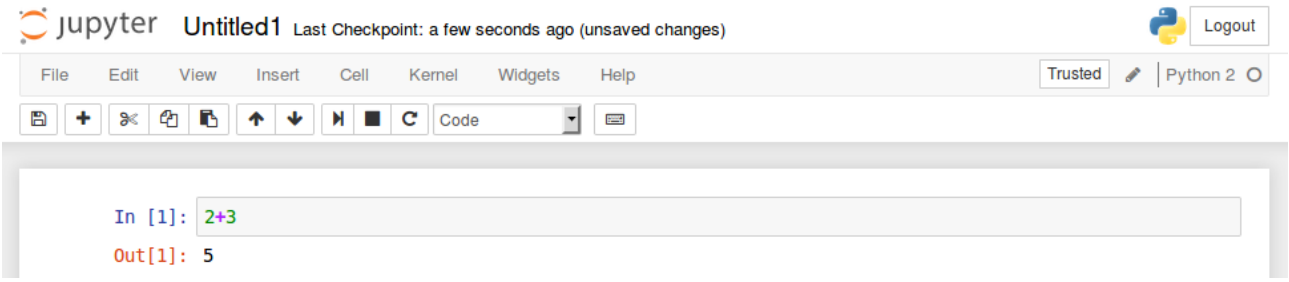
```
[I 09:56:59.424 NotebookApp] Writing notebook server cookie secret to
/run/user/1000/jupyter/notebook_cookie_secret
[I 09:56:59.442 NotebookApp] Serving notebooks from local directory: /home/jcf
[I 09:56:59.442 NotebookApp] 0 active kernels
[I 09:56:59.442 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?
token=b65e3fbf2f7371a6a9e3002895e69ee707662033ffc64fdc
[I 09:56:59.443 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to
skip confirmation).
[C 09:56:59.443 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=b65e3fbf2f7371a6a9e3002895e69ee707662033ffc64fdc
[I 09:56:59.700 NotebookApp] Accepting one-time-token-authenticated connection from 127.0.0.1
[I 09:57:44.328 NotebookApp] Creating new notebook in
[I 09:57:44.336 NotebookApp] Writing notebook-signing key to
/home/jcf/.local/share/jupyter/notebook_secret
[I 09:57:44.864 NotebookApp] Kernel started: a296785b-e78a-476d-a6ec-3979366b2cb1
```

and a browser window pops up that displays the directory listing.
Now you can select an exiting notebook or create a new one by klicking on the **New button** to the right.



Select Python2 (or Python3).

In the now showing input cell, you can enter calculations and **execute the cell with <Ctrl><Enter>**:



Smart help and smart editing

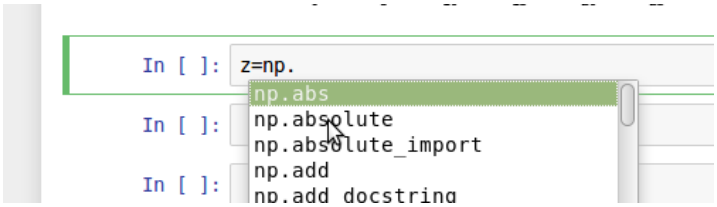
With „?“ a contextual help can be found:

```
In [5]: import numpy as np
        x=np.log?
```

```
Type:            ufunc
String form:     <ufunc 'log'>
File:            ~/anaconda2/lib/python2.7/site-packages/numpy/__init__.py
Docstring:
log(x[, out])

Natural logarithm, element-wise.

The natural logarithm `log` is the inverse of the exponential function,
so that `log(exp(x)) = x`. The natural logarithm is logarithm in base
`e`.
```

….

Tab completion shows available functions:



Magic functions

Magic functions are preceded by a „%“ (this may be omitted if the setting automagic is turned on, as is the case by default) and can do interesting things like:

- %ls
  List a directory

- %timeit <code>
  Get the time that it takes to execute code

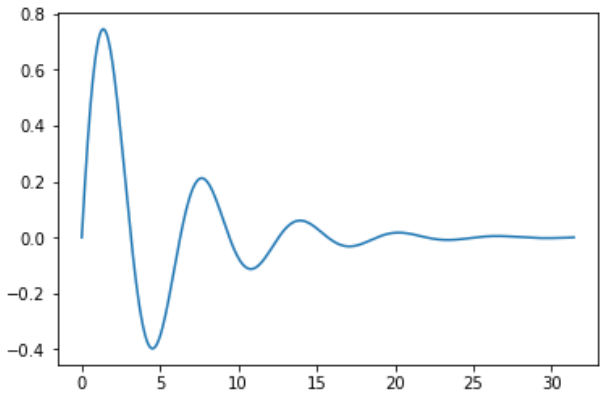
```
In [25]: %timeit z = np.cos(x)

         The slowest run took 4.99 times longer than the fastest. This could mean th
         at an intermediate result is being cached.
         100000 loops, best of 3: 12.2 µs per loop
```

- %magic gives more info on all magic functions

A simple plotting example

```
In [6]: # produce some data:
        import numpy as np
        x= np.linspace(0,10*np.pi, 500)
        y=np.sin(x)*np.exp(-x/5)
```

```
In [2]: # plot the data:
        import matplotlib.pyplot as plt
        %matplotlib inline                # this is to view the diagram!
        plt.plot(x,y)

Out[2]: [<matplotlib.lines.Line2D at 0x7f23cff54b90>]
```



Code and text cells

The default cells are code cells (default language is Python, but it may also be R or others, if they are installed).
To make the notebook a real document, markdown text cells may be inserted.
The cell type can be selected in the toolbar. When a markdown cell is executed, the result is shown.

- Headings consist of 1 to 6 times the „#“ symbol followed by the heading text

- Paragraphs must be separated by empty lines (multiple lines are displayed as one line if not separated by an empty line)
- The LATEX syntax may be used for mathematical expressions

Example:

```
## Heading 2

Paragraphs
must be separated
by empty lines!

Basic formatting:
*italic*
__bold__


* list 1
* list 2
 * list item 2.1
 * list item 2.2

Latex syntax may be used:

y = sin ($\omega$ t)

with $\omega$ = 2 $\pi$ f
```

Heading 2

Paragraphs must be separated by empty lines!

Basic formatting: *italic* **bold**

- list 1
- list 2
  - list item 2.1
  - list item 2.2

Latex syntax may be used:

y = sin (ω t)

with ω = 2 π f

A formula example:

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

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

A collection of examples can be found here:

http://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Typesetting%20Equations.html#The-Lorenz-Equations

Advanced notebooks

Notebooks can contain images and formatted text
https://nbviewer.jupyter.org/github/ipython/ipython/blob/master/examples/IPython%20Kernel/Rich%20Output.ipynb

Images

```
In [5]: from IPython.display import Image
        i = Image(filename='graffitysmall.jpg')
        i
```

Out[5]:

Ther may be an error message (without displaying the image) if it is too big.

HTML

```
In [33]: from IPython.display import HTML
         from IPython.display import display

         s='''
         <html>
         <font color="#E00000">
         <h1>Hello world!</h1>
         </body>
         </html>
         '''
         h = HTML(s)
         display(h)
```

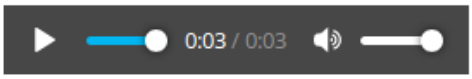
Hello world!

Audio

Even an audio signal can be generated and played in the notebook:

```
In [10]: from IPython.display import Audio
         import numpy as np
         max_time = 3
         f1 = 440.0
         f2 = 437.0
         rate = 10000
         L = 3

         times = np.linspace(0,L,rate*L)
         signal = np.sin(2*np.pi*f1*times) + np.sin(2*np.pi*f2*times)

         Audio(data=signal, rate=rate)
```

Out[10]:

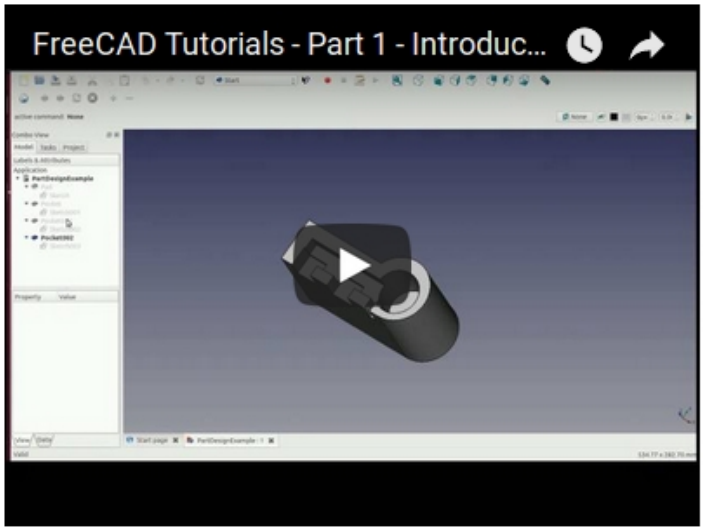


Video

It is possible to display Youtube videos:

```
In [11]: from IPython.display import YouTubeVideo
         YouTubeVideo('QzfT-ZDp0Wo')
```

Out[11]:

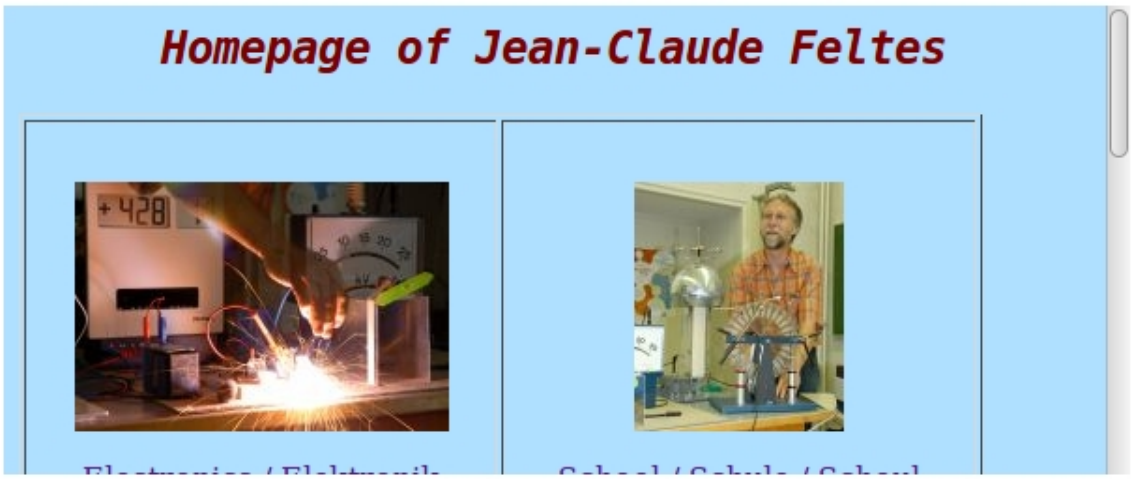


The string 'QzfT-Zdp0Wo' used here is the result of a search on youtube videos.
The original link was 'https://www.youtube.com/watch?v=QzfT-ZDp0Wo'

Embed entire Webpages

```
In [22]: from IPython.display import IFrame
         IFrame('http://staff.ltam.lu/feljc/', width='100%', height=250)
```

Out[22]:



Even the links on the webpage are functional

Export

All export is done in the form of a „Download“. That means that the resulting file(s) are found in the Downloads folder. There are different formats available:

- HTML
  This gives a fine copy of the notebook, including all features. It can be used as is for web publishing.
  However, converting this html to ODT with pandoc results in a corrupt document.

- Markdown
  I tried to convert the resulting md file to ODT text. There was much of the data lost, unfortunately. Code was rendered, but with no syntax coloring. Pictures and images were there, mathematical formulas were lost.

- Latex (Latex must be installed!)

- PDF (for this also, Latex must be installed!)

Copy / Paste to Libre Office

I tried to copy something from the Jupyter notebook and paste it with „Paste Special“ as HTML to a Libre Office documnt. Unfortunately most of the special features were lost.

Conclusions

- Jupyter is an excellent tool for interactive cientific programming.
- The HTML export can be directly used as a document.
- However the export capabilities, especially to PDF and ODT are still poor (september 2017). Let's hope that these will be improved in the future.