Computational Radiometry Work Package

Document Number	Equipment or Sub-System

Subject

01-IPythonHintsAndTips

Distribution

Conclusions/Decisions/Amendments

Author CJ Willers	Signature

Date	Date	Date August 23, 2021
Previous Package No.	Superseding Package No.	Current Package No.

This notebook forms part of a series on computational optical radiometry[1]. The notebooks can be downloaded from Github[2]. These notebooks are constantly revised and updated, please revisit from time to time.

This notebook was written several Ipython/Jupyter generations ago, some information may be no longer applicable.

1 1 Jupyter / IPython notebook hints and tips

The date of this document and module versions used in this document are given at the end of the file.

Feedback is appreciated: neliswillers at gmail dot com.

2 Jupyter and IPython

From `http://ipython.org/`:

"IPython is a growing project, with increasingly language-agnostic components. IPython 3.x will be the last monolithic release of IPython, containing the notebook server, qtconsole, etc. The language-agnostic parts of the project: the notebook format, message protocol, qtconsole, notebook web application, etc. will move to new projects under the name Jupyter[3]. IPython itself will return to being focused on interactive Python, part of which will be providing a Python kernel for Jupyter. IPython 3.0 contains some indications of the project transition, including the logo in the notebook web UI being that of Jupyter."

In this document, all references to IPython refer to the IPython kernel, running on top of Jupyter.

IPython versions 2.x use the nbformat 3 file format.

IPython versions 3.x use the nbformat 4 file format.

To convert from nb4 to nb3 format (from with Jupyter IPython 3 installed) type:

`ipython nbconvert --to notebook --nbformat 3 mynotebook.ipynb`

`http://ipython.org/ipython-doc/3/notebook/nbformat.html#nbformat`[4]

`http://ipython.org/ipython-doc/3/whatsnew/version3.html`[5]

2.1 Overview

This notebook provides a brief summary of how to start up and use the IPython notebook. Introductions are given to magic commands, help functionality, using IPython for scientific work, cell memory, markdown, citations, embedding media files, and a few lesser used functions.

```
from IPython.display import display
from IPython.display import Image
from IPython.display import HTML
```

Lorena Barba's tutorial[6], see this notebook[7] for more detail on the Jupyter notebook.

2.2 Why use the IPython notebook?

The IPython notebook is an effective means to capture technical story lines or flow-of-thought;

The IPython notebook can never replace formal documentation, at least in its present form. The notebook should also not be used as a primary software development environment. The notebook lives alongside other forms of documentation and coding. Having said that, there is a definite place for the IPython notebook in several environments such as engineering research and development, teaching, and scientific research and experimentation (which it was initially developed for).

The IPython notebook is a very effective means to capture your work as you progress through an investigation comprising research, coding, and record keeping. It is also an excellent way to develop slides or teaching material where one wants to combine code, text and results in a story-line. It is used widely in Python conferences and lectures. The notebook is also a convenient means to do homework assignments.

`http://nbviewer.ipython.org/urls/raw.github.com/ellisonbg/talk-strata2013/master/StrataIPythonSlides.ipynb`[8] - why use notebooks?

`http://nbviewer.ipython.org/`[9] - gallery of notebooks

`https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks` - a gallery of notebooks

2.3 Getting started in IPython

During early 2015 the IPython 2.x tool was upgraded to IPython 3.0. When installing, use the latest version you can find.

If you installed the Anaconda Python distribution (`http://docs.continuum.io/anaconda/install.html`[11]), it already has IPython, no need to download it.

If you are not using Anaconda, follow the steps outlined on these sites:

`http://ipython.org/` - start here, download the latest version from here.

`http://ipython.org/ipython-doc/dev/interactive/notebook.html`[12] - short and concise, up and running quickly.

`http://nbviewer.ipython.org/urls/raw.github.com/Unidata/tds-python-workshop/master/ipython-notebook.ipynb`[13] - simple intro

`http://blog.safaribooksonline.com/2013/12/12/start-ipython-notebook/`[14] linux install and general use instructions

`http://ipython.org/ipython-doc/stable/notebook/index.html`[15] The IPython notebook.

`http://www.astro.washington.edu/users/vanderplas/Astr599/notebooks/03\_IPython\_intro`[16]

For a good overview of the history and key user guide tips, see 1. `https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook#gs.KMCTS4w`[17]. 2. `https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/`[18]

2.4 Diffing and Merging Notebooks

https://nbdime.readthedocs.io/en/latest/

nbdime provides ’content-aware’ diffing and merging of Jupyter notebooks. It understands the structure of notebook documents. Therefore, it can make intelligent decisions when diffing and merging notebooks.

2.5 Working in a Command Window

IPython must be started from a command window (in Windows) or a terminal console in Linux. If you use Linux, you already know how to do this. If you are using Windows, and you don't know learn here:

`http://www.cs.princeton.edu/courses/archive/spr05/cos126/cmd-prompt.html`[19]

`http://www.bleepingcomputer.com/tutorials/windows-command-prompt-introduction/`[20]

IPython 2.x allows you to open notebooks only from the present directory (getcwd) and in nested subdirectories. So, open the command window somewhere where your notebooks can be reached from.

This link will help you to create a context menu in Explorer to start a command window in a given directory:

`http://stackoverflow.com/questions/1077814/assigning-a-shortcut-to-open-cmd-here`[2

`https://stackoverflow.com/questions/60904/how-can-i-open-a-cmd-window-in-a-specifi`

Scroll to the bottom of the pages. Alternatively just download this file and double click on it:

`https://raw.githubusercontent.com/NelisW/ComputationalRadiometry/master/cmd-here/cmd-window-here.reg`[23]

2.6 Starting IPython

2.6.1 Command line

IPython files are json[24]-format files, with the file extension `.ipynb`.

After you installed IPython you must start the server in a command window. The current version of IPython expects the notebook files to be in the same directory where it was started up (or below). So if you want to work in `c:\myfiles` then change to that directory and start IPython in the required directory.

Open a command window (DOS box), and create a directory where you want to work with the notebooks. Type the following commands in the command window (pressing Enter after each line):

`cd \`

`mkdir myfiles` [only do this the first time]

`cd c:\myfiles`

The command window should now display `c:\myfiles>`. Then type the following command (followed by Enter):

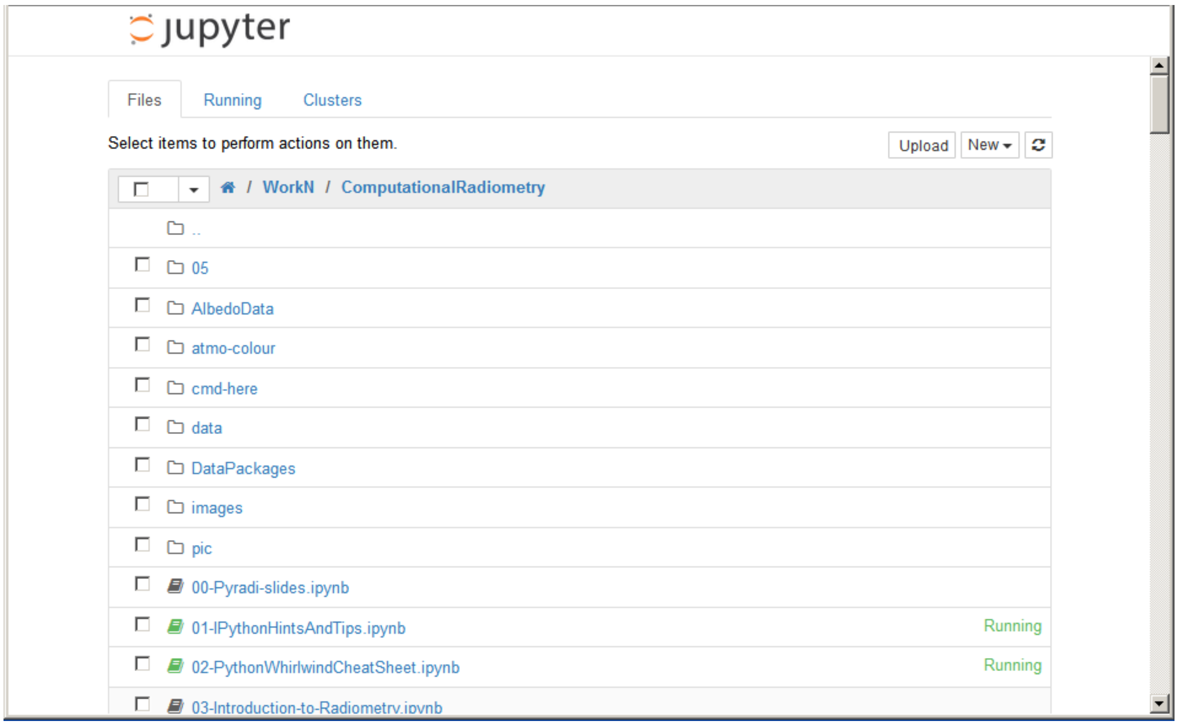
```
ipython notebook
```

2.6.2 Serving IPython pages

After a while IPython opens your web browser and display the IPython portal page.

Side note on browsers: Microsoft Internet Explorer does not run IPython very nicely. Consider using Firefox or Chrome.

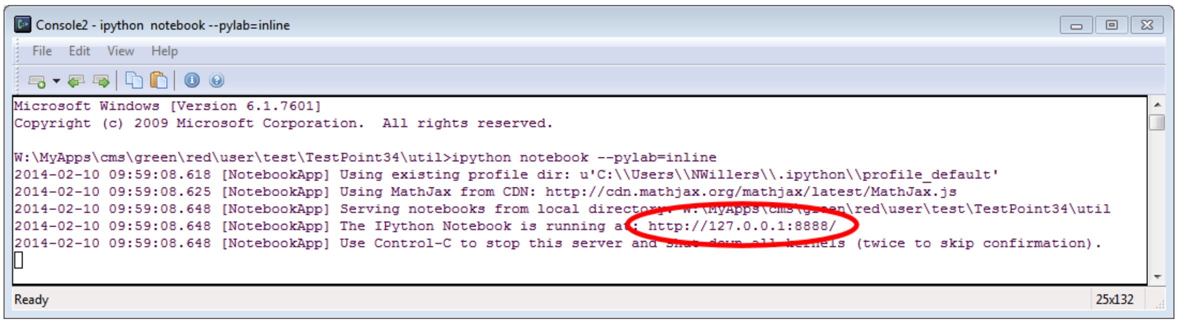
```
display(Image(filename='images/portalpage.png'))
```


Select items to perform actions on them.

```
WorkN / ComputationalRadiometry
..
05
AlbedoData
atmo-colour
cmd-here
data
DataPackages
images
pic
00-Pyradi-slides.ipynb
01-IPythonHintsAndTips.ipynb                                   Running
02-PythonWhirlwindCheatSheet.ipynb                             Running
03-Introduction-to-Radiometry.ipynb
```

IPython works by starting a web server on your PC and then serving pages from this server. This web server has an IP address of `127.0.0.1` (localhost), on port 8888, i.e., browser address `127.0.0.1:8888` or `localhost:8888`. You can at any time open a new browser tab and enter this address and the server portal page should appear. This is sometimes convenient when you close all IPython tabs in the browser, but the server is still running.

```
display(Image(filename='images/ipaddress.png'))
```


Console2 - ipython notebook --pylab=inline

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

W:\MyApps\cms\green\red\user\test\TestPoint34\util>ipython notebook --pylab=inline
2014-02-10 09:59:08.618 [NotebookApp] Using existing profile dir: u'C:\\Users\\NWillers\\.ipython\\profile_default'
2014-02-10 09:59:08.625 [NotebookApp] Using MathJax from CDN: http://cdn.mathjax.org/mathjax/latest/MathJax.js
2014-02-10 09:59:08.648 [NotebookApp] Serving notebooks from local directory: W:\MyApps\cms\green\red\user\test\TestPoint34\util
2014-02-10 09:59:08.648 [NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888/
2014-02-10 09:59:08.648 [NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

Ready   25x132

You can see all the IPython options by typing `ipython --help`.

2.7 Starting more than one IPython notebook server

From the manual: "You can start more than one notebook server at the same time, if you want to work on notebooks in different directories. By default the first notebook server starts on port 8888, and later notebook servers search for ports near that one. You can also manually specify the port with the –port option."

`http://ipython.org/ipython-doc/dev/interactive/notebook.html#starting-the-notebook`

2.8 IPython notebook security

The IPython notebook server acts as a generic file server for files inside the same tree as your notebooks. Access is not granted *outside* the notebook folder so you have strict control over what files are visible. It is recommended that you do not run the notebook server with a notebook directory at a high level in your file system (e.g. your home directory).

When you run the notebook in a password-protected manner, local file access is restricted to authenticated users unless read-only views are active.

More information is required here

2.9 Live Connection to the Internet

IPython uses MathJax to render LaTeX, as in $y = ax^2 + bx + c$. The default mode is to access MathJax online, live, as you are working. This means that you must be working on a PC connected to the Internet in order to render LaTeX. This is only required when LaTeX rendering is requested, otherwise you don't need Internet access. Once the LaTeX is rendered it is embedded as an image in the document and you do not need to be connected to MathJax to view the previously rendered image.

One catch is that if you live behind a server that requires you to authenticate before logging in to the Internet, this means that you must authenticate your Internet access prior to starting the ipython server.

2.10 IPython and your Firewall

If IPython does not work, it may be because your antivirus software or firewall prevents it from working. The firewall might not be set up to grant execution rights to the default IPython notebook server operating 127.0.0.1:8888. In this case the cells will simply not execute, with no warning. You don't receive any output in from the cells.

2.10.1 Using localhost to bypass the firewall

This approach does not require any changes to your firewall. Some firewalls are set up to grant `localhost` execution rights. In this case the server can be started with the command

```
ipython notebook --ip=localhost
```

Once started, the pages are served from

```
http://localhost:8888/
```

and not from `http://127.0.0.1:8888/`.

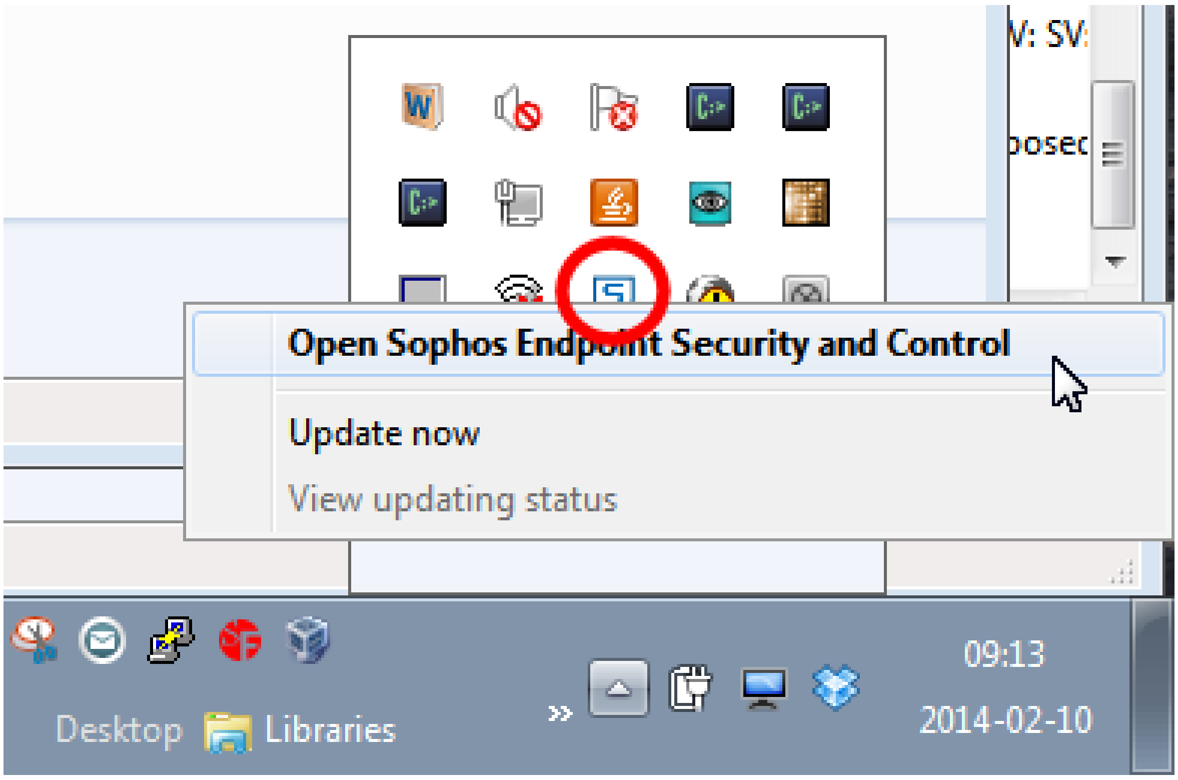
2.10.2 Setting up the Sophos firewall

Here is a description of how to fix this for Sophos - it may be necessary to do this after each reboot. Other firewalls will have similar functionality.

Sophos blocks the browser to not allows execution of code as is required by ipython. To enable the browser ipython functionality change the authorisation on your local loopback IP address 127.0.0.1.

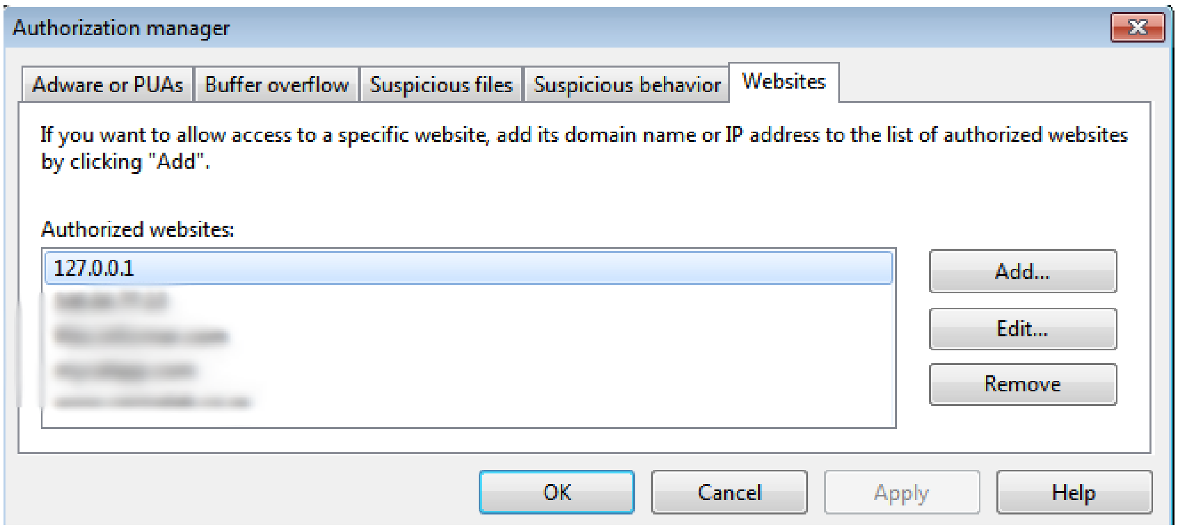
For this to work, you must have Administrator rights, or at least belong to the Admin group (call your ICT sysadmin if you don't have these rights. Start by opening the Sophos application from the system tray by right-clicking on the S-shield icon and selecting the

```
display(Image(filename='images/sophos01.png'))
```



Once in the Sophos control panel, select the following menu option `<Configure><Anti-virus><Authorizat`
The following window should appear:

```
display(Image(filename='images/sophos02.png'))
```



If you want to allow access to a specific website, add its domain name or IP address to the list of authorized websites by clicking "Add".

Authorized websites:

127.0.0.1

If the 127.0.0.1 loopback IP address does not appear in the list, add it as shown above.

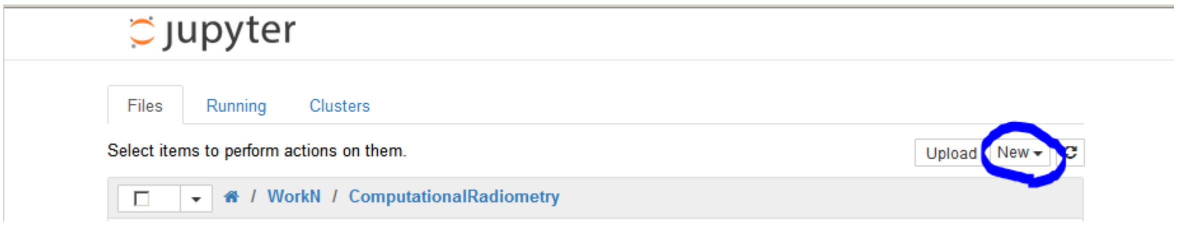
Now close any ipython notebooks that you have, also the page at 127.0.0.1:8888. Close the command window and re-open a new one. Open a new tab and enter the local address 127.0.0.1:8888. This should display the notebooks in the current directory and once opened, they should work.

2.11 Starting a new notebook

To create a new page click on the `New Notebook` button on the IPython notebook portal.

You can also make a copy of an existing notebook by selecting the appropriate menu option under the `File` menu.

```
display(Image(filename='images/ipython-newnotebook.png'))
```



Select items to perform actions on them.

2.12 Notebook cells

The notebook comprises a 'behind-the-scene' kernel that does all the processing and a 'front-man' rendering in the browser window. It is important to understand that the web page browser does very little work, it only renders the information under instruction from the kernel. The kernel contains all the code, text and data.

The notebook consists of a number of cells which can contain code, text or other information. Each cell is 'executed' or 'run' by clicking on the cell and pressing the Shift-Enter key combination. Cells can also be run in sequence from the Cell menu entry. Running a cell does at least two things: (1) the cell changes/updates its data in the kernel and (2) the updated data is rendered in the browser window.

Writing the notebook becomes the process of creating cells and adding text or code to the cell.

Once created the cells must be executed. The cell execution must be in the order entered in the notebook (from start to end). You can do this manually or use the menu entry to run all cells consecutively.

It is possible and often done that cells are moved up or down, changing their location in the sequence. When the cell sequence is changed you must keep track by executing the cells in their new locations. If the execution in strict sequence is not followed it can lead to all sorts of difficulties, see Notebooks Remember[26]

2.13 Saving and closing the Notebook

The notebook is saved in its json format by saving from the IPython menu. Click on the save button (leftmost button with the mouse-over message 'Save and Checkpoint') or select the `File Save and Checkpoi` menu option.

The IPython notebook file is saved and closed by selecting the `File Close and Halt` menu option.

Never use a Python `exit()` function in your notebook, because the notebook will interpret this as an exit to its own process, closing down the server.

2.14 Converting the notebook to other formats

2.14.1 IPython versions and notebook versions

IPython version 2.x writes files in notebook format nbformat 3.

IPython version 3.x writes files in notebook format nbformat 4.

IPython 3 can read nbformat 3 files, converting on opening. When saved the file will be in nbformat 4.

To convert a file from nbformat 4 to nbformat 3, use the following command line command:

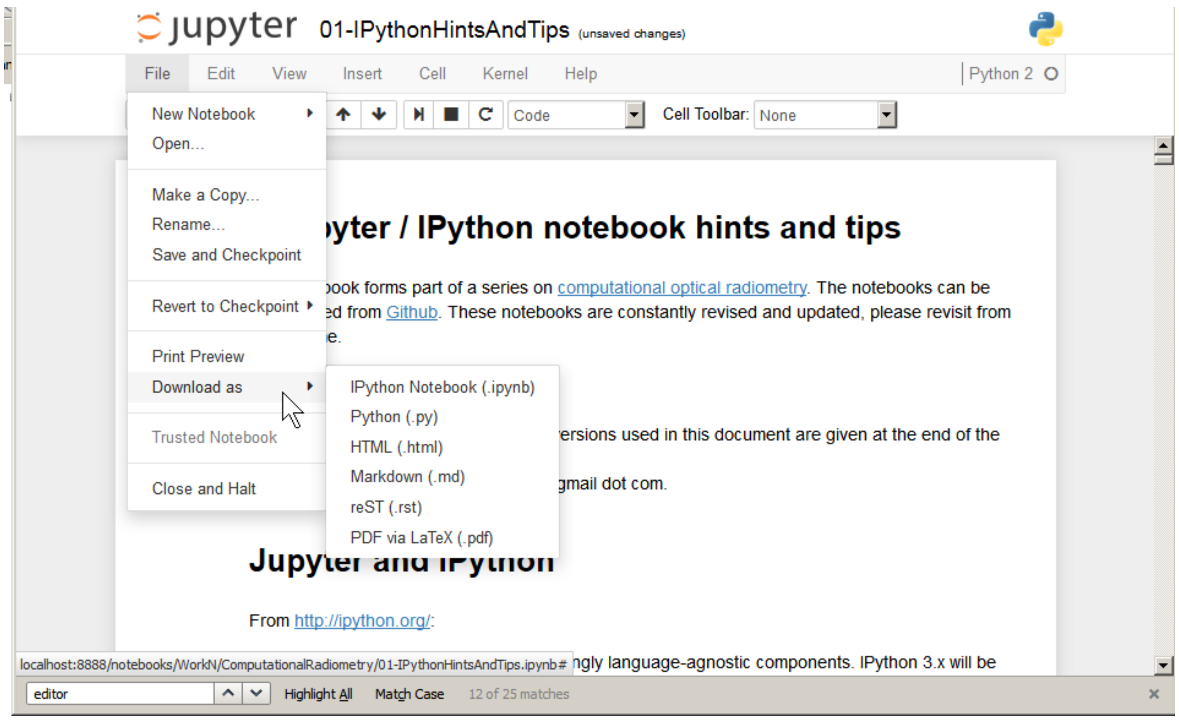
```
ipython nbconvert --to notebook --nbformat 3 MyNotebook.ipynb
```

2.14.2 IPython built-in conversion

Starting from IPython 2, the notebook can be converted to a Python file (code with embedded comments), an HTML file or a ReST file. On the *File* menu, use *Download as*.

The notebook can also be saved to an HTML file by saving from your browser's menu. This saved HTML file is, however, not fully self-contained (images and JavaScript files are in a directory). So this is *not* the ideal way to save the file.

```
HTML('<img src="images/convertfile.png" width=600 height=300/>')
```


jupyter 01-IPythonHintsAndTips (unsaved changes)
File Edit View Insert Cell Kernel Help
Python 2
Code
Cell Toolbar: None
New Notebook
Open...
Make a Copy...
Rename...
Save and Checkpoint
Revert to Checkpoint
Print Preview
Download as
Trusted Notebook
Close and Halt
IPython Notebook (.ipynb)
Python (.py)
HTML (.html)
Markdown (.md)
reST (.rst)
PDF via LaTeX (.pdf)
yter / IPython notebook hints and tips
ook forms part of a series on computational optical radiometry. The notebooks can be
ed from Github. These notebooks are constantly revised and updated, please revisit from
e.
ersions used in this document are given at the end of the
gmail dot com.
Jupyter and IPython
From http://ipython.org/:
localhost:8888/notebooks/WorkN/ComputationalRadiometry/01-IPythonHintsAndTips.ipynb#
ngly language-agnostic components. IPython 3.x will be
editor
Highlight All
Match Case
12 of 25 matches

2.14.3 Command-line conversion

On the command line IPython has a `convert` subcommand to convert from the IPython format to other formats. In the command window, where the file is located type:

```
ipython nbconvert filename.ipynb
```

This will create a fully self-contained HTML file that you can print or mail as a single file. The only problem is that HTML does not print very well, especially with large figures.

The notebook can be converted to LaTeX and then compiled to a PDF format with the following command:

```
ipython nbconvert --to latex --post filename.ipynb
```

The LaTeX document style can be specified as follows:

```
ipython nbconvert --to latex filename.ipynb --template=article
```

once the tex document is ready run pdflatex:

```
pdflatex filename.tex
```

`http://ipython.org/ipython-doc/rel-1.0.0/interactive/nbconvert.html`[27]

`http://blog.fperez.org/2012/09/blogging-with-ipython-notebook.html`[28]

`http://nbviewer.ipython.org/url/www.damian.oquanta.info/posts/blogging-with-nikola` `ipynb`[29] - blogging with Nikola

`http://www.slideviper.oquanta.info/tutorial/slideshow\_tutorial\_slides.html?` `transition=none#/`[30] - slides

`http://www.damian.oquanta.info/`[31] - a couple of links

`http://nbviewer.ipython.org/github/Carreau/posts/blob/master/06-NBconvert-Doc-Draf` `ipynb`[32] - using the nbconvert API

Tips to use IPython for publication ready work: http://blog.juliusschulz.de/blog/ultimate-ipython-notebook

2.14.4 LaTeX docs with template control

The standard LaTeX converter does not provide good control over the style of the resultant document. If you require better control of the document style look at my ipnb2tex script[33]. The script support floating figures and tables and citations. See this PDF[34] for an example of the output from this converter. Using this converter you can fully control the LaTeX template to achieve the document format you require.

Making publication ready Python Notebooks[35] provides very useful information on preparing notebooks such that it can be used for final publications.

2.14.5 High-quality graphics in LaTeX

By default the Matplotlib backend for IPython generates png files. The quality of these files are not all that good for publication. LaTeX traditionally uses Encapsulated PostScript (eps) files for publications, or PDF files in PDFLaTeX. It is possible to instruct the backend to render in Scalable Vector Graphics (svg) format by using: `\%config InlineBackend.figure\_format = 'svg'`

The svg file is, however, not rendereable in LaTeX and must be converted to PDF for use in PDFLaTeX. In order to do the conversion you must have Inkscape[36] on your PC (and on Windows the path to the inkscape executable must be on your PATH). The nbconvert process will convert the svg files created by Matplotlib/IPython to PDF files using Inkscape on your PC.

`http://ipython.org/ipython-doc/rel-1.0.0/interactive/nbconvert.html`[37]

`https://stackoverflow.com/questions/19659864/ipython-nbconvert-and-latex-use-eps-i`

`https://stackoverflow.com/questions/19600234/nbconvert-pdf-latex-page-formatting-i`

`https://stackoverflow.com/questions/19524554/suppress-code-in-nbconvert-ipython`[40]

2.15 Notebook viewer on the internet

There is a website nbviewer.ipython.org/[41] that will convert a notebook from ipynb format to html and display it in your browser. Just browse over to `http://nbviewer.ipython.org/`[42] and enter the URL of the notebook you want to view.

A particularly useful feature of `http://nbviewer.ipython.org/`[43] is that you can add the GitHub username of someone and then can view all the notebooks by that user.

The `nbviewer` site keeps a cache of the most recently calculated version of your notebook. To force an updated calculation add this to the end of the URL: `?flush\_cache=true`.

Alternatively, you can build a composite URL such as follows to view a notebook.

2.16 Keyboard Shortcuts

Jupyter stores a list of keyboard shortcuts under the menu at the top: `Help > Keyboard Shortcuts`. Another way to access keyboard shortcuts, and a handy way to learn them is to use the command palette: `Ctrl` Shift + C+ or `Ctrl` Shift + P+ , but note that this key press combination may be hardwired in the browser..

http://ipython.readthedocs.io/en/stable/config/shortcuts/index.html

Josh Devlin[44] provides the following summary of useful keyboard shortcuts (but study the full set as described above):

- `Esc` will take you into command mode where you can navigate around your notebook with arrow keys.
- While in command mode:
- `A` to insert a new cell above the current cell,
- `B` to insert a new cell below.
- `M` to change the current cell to Markdown,
- `Y` to change it back to code
- `D` D+ (press the key twice) to delete the current cell

- `Enter` will take you from command mode back into edit mode for the given cell.
- `Shift` Tab+ will show you the Docstring (documentation) for the the object you have just typed in a code cell - you can keep pressing this short cut to cycle through a few modes of documentation. This operation requires `install pyreadline`.
- `Ctrl` Shift + -+ will split the current cell into two from where your cursor is.
- `Esc` F+ Find and replace on your code but not the outputs.
- `Esc` O+ Toggle cell output.
- Select Multiple Cells:
- `Shift` J or Shift + Down+ selects the next sell in a downwards direction.
- `Shift` K or Shift + Up+select sells in an upwards direction.
- Once cells are selected, you can then delete / copy / cut / paste / run them as a batch. This is helpful when you need to move parts of a notebook.
- You can also use `Shift` M+ to merge multiple cells.

2.16.1 Pretty Print all cell outputs

Normally only the last output in the cell will be printed. For everything else, you have to manually add print(), which is fine but not super convenient. You can change that by adding this at the top of the notebook:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast\_node\_interactivity = "all"
```

Any time you want to go back to the original setting, just run

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast\_node\_interactivity = "last\_expr"
```

Just be aware that you have to run the setting change in a separate cell for it to take effect for the next cell run.

2.17 Magics and System Commands

IPython has a set of predefined ’magic functions’ that you can call with a command line style syntax. There are two kinds of magics, line-oriented and cell-oriented. Line magics are prefixed with the % character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes. Cell magics are prefixed with a double %%, and they are functions that get as an argument not only the rest of the line, but also the lines below it in a separate argument.

http://nbviewer.ipython.org/github/ipython/ipython/blob/master/examples/notebooks/
Cell\%20Magics.ipynb[45]

http://ipython.org/ipython-doc/dev/interactive/tutorial.html#magic-functions[46]

http://damontallen.github.io/IPython-quick-ref-sheets/[47]

System commands (as you would normally type in a command window) can be executed by pre-pending with an exclamation mark.

```
import os
# test to see if this is Linux or Windows
if os.path == '/':
    !ls *.ipynb
else:
    !dir *.ipynb
```

```
 Volume in drive K has no label.
 Volume Serial Number is D861-697A

 Directory of K:\WorkN\ComputationalRadiometry

2021/08/23  18:46            56y931 00-Installing-Python-pyradi.ipynb
2021/08/23  20:21           709y792 01-IPythonHintsAndTips.ipynb
2021/08/23  20:17           865y609 02-PythonWhirlwindCheatSheet.ipynb
2021/08/23  16:06           699y014 03-Introduction-to-Radiometry.ipynb
2021/08/23  16:13         1y736y911 04-↵
   IntroductionToComputationalRadiometryWithPyradi.ipynb
2019/10/08  06:23         1y615y272 05a-PlottingWithPyradi-↵
   GeneralAndCartesian.ipynb
2021/03/15  21:17         4y022y544 05b-PlottingWithPyradi-Polar-and-3D↵
   .ipynb
2021/06/01  17:45            91y205 06-Diverse-pyradi-utilities.ipynb
2021/06/04  12:08         1y943y496 07-Optical-Sources.ipynb
2021/07/18  17:46           801y635 08-ModtranFileProcessing.ipynb
2021/05/10  10:45           297y868 09a-DetectorModelling.ipynb
2021/06/02  14:03           197y015 09b-StaringArrayDetectors.ipynb
2021/06/02  14:03         3y502y386 09c-StaringArrayDetectors-Visual-↵
   low-light.ipynb
2021/06/02  14:26         3y689y616 09d-StaringArrayDetectors-Infrared-↵
   sensor.ipynb
2018/04/04  10:15         1y726y703 10-ImageUtilities.ipynb
2018/05/28  18:54           860y315 11-InfraredMeasurementAndAnalysis.↵
   ipynb
2020/11/11  16:23           282y034 12a-FlameSensorAnalysis.ipynb
2016/12/13  13:55         4y400y310 12b-AlbedoDerivation.ipynb
2016/12/13  13:55         9y543y136 12c-AtmosphericEffectColourCoords.↵
   ipynb
2016/12/13  13:55           341y897 12d-SpectralTemperatureEstimation.↵
   ipynb
2016/12/13  13:55         1y147y431 12e-Cloth-Targets.ipynb
2017/02/16  09:28         2y600y608 12f-FOV-optimisation.ipynb
2018/03/24  23:02           553y401 12g-Plume-texture-Copy1.ipynb
2018/04/04  10:17           223y302 12g-Plume-texture.ipynb
2020/11/01  10:58         2y033y002 12h-MWIR-Well-fill.ipynb
2017/04/11  13:56             1y124 12i-Pixel-crosstalk-MTF-impact-MWIR↵
   -performance.ipynb
2020/11/02  15:46           413y143 12j-laser-systems-performance.ipynb
2016/12/13  13:55             2y085 99-Pyradi-slides.ipynb
2017/01/13  12:27            30y544 PlayStats.ipynb
              29 File(s)     44y388y329 bytes
```

```
               0 Dir(s)  1y310y544y900y096 bytes free
```

There are 'magics' to support a number of other languages in the IPython notebook. From the IPython notebook website[48]:

"We ship the official IPython kernel, but kernels for other languages such as Julia and Haskell are actively developed and used. Additionally, the IPython kernel supports multi-language integration, letting you for example mix Python code with Cython, R, Octave, and scripting in Bash, Perl or Ruby."

`http://andrew.gibiansky.com/blog/ipython/ipython-kernels/`[49]

`http://ipython.org/ipython-doc/stable/development/messaging.html`[50]

2.18 Magics for general use

Thanks to Josh Devlin[51] for some of the examples shown here.

List all the magics currently available

```
%lsmagic
```

```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %↵
   bookmark  %cd  %clear  %cls  %colors  %conda  %config  %connect_info↵
     %copy  %ddir  %debug  %dhist  %dirs  %doctest_mode  %echo  %ed  %↵
   edit  %env  %gui  %hist  %history  %killbgscripts  %ldir  %less  %↵
   load  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %↵
   logstop  %ls  %lsmagic  %macro  %magic  %matplotlib  %mkdir  %more ↵
   %notebook  %page  %pastebin  %pdb  %pdef  %pdoc  %pfile  %pinfo  %↵
   pinfo2  %pip  %popd  %pprint  %precision  %prun  %psearch  %psource ↵
    %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %↵
   rehashx  %reload_ext  %ren  %rep  %rerun  %reset  %reset_selective ↵
   %rmdir  %run  %save  %sc  %set_env  %store  %sx  %system  %tb  %time↵
     %timeit  %unalias  %unload_ext  %who  %who_ls  %whos  %xdel  %↵
   xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%cmd  %%debug  %%file  %%html ↵
   %%javascript  %%js  %%latex  %%markdown  %%perl  %%prun  %%pypy  %%↵
   python  %%python2  %%python3  %%ruby  %%script  %%sh  %%svg  %%sx ↵
   %%system  %%time  %%timeit  %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

To learn more about a magic execute ? followed by the magic as in

`?\%pastebin`

it will open a new panel where the docstring is displayed.

```
?%timeit
```

BTW, you can also display the docstring of any python function by

```
import numpy as np
?np.asarray
```

Remove variables from the IPython notebook by using the \%reset and \%reset_selective varname magics. \%reset removes all variables (i.e., cleans out the whole lot), whereas with \%reset_selective va you can specificy which variables must be cleared. Note[52] that varname is a regular expression, but such that if a single letter is given all variables starting with that letter will be erased. So to ensure that only a single variable is removed, anchor both ends as shown below with ^ (beginning of line) and $ (end of line).

```
#remove only variable b
a=1; b=2; c=3; b1m=4; b2m=5; b3m=6; b4m=7; b2s=8
%reset_selective -f ^b$
%who_ls
```

```
['HTML',
 'Image',
 'a',
 'b1m',
 'b2m',
 'b2s',
 'b3m',
 'b4m',
 'c',
 'display',
 'fi',
 'one',
 'os',
 'three',
 'two']
```

```
#remove all variables starting with the letter b
a=1; b=2; c=3; b1m=4; b2m=5; b3m=6; b4m=7; b2s=8
%reset_selective -f b
%who_ls
```

```
['HTML', 'Image', 'a', 'c', 'display', 'fi', 'one', 'os', 'three', 'two↵
   ']
```

The \%who command without any arguments will list all variables that exist in the global scope. Passing a parameter like str will list only variables of that type.

```
one = "for the money"
two = "for the show"
three = "to get ready now go cat go"
%who str
```

```
one   three    two
```

Use the \%\%writefile magic to write contents to a file. The file can be read in normal Python or it can be read and popped up in a window by \%pycat.

```
%%writefile test.txt
This is a test file!
It can contain anything I want...


more...
```

```
Overwriting test.txt
```

```
#open the file and read its contents
with open('test.txt', 'r') as fi:
    print('{}'.format(' '.join(fi.readlines())))
```

```
This is a test file!
 It can contain anything I want...


 more...
```

```
%pycat test.txt
```

The %edit magic is supposed to start up an editor from within IPython - I never got that to work under windows.

`https://ipython.org/ipython-doc/dev/interactive/magics.html#magic-edit`[53]

`http://ipython.org/ipython-doc/1/config/editors.html`[54]

`http://stackoverflow.com/questions/15681153/external-editor-for-ipython-notebook`[55

`http://stackoverflow.com/questions/3438531/ipython-workflow-edit-run`[56]

This works ok if you have a fast editor:

`http://stackoverflow.com/questions/28309430/edit-ipython-cell-in-an-external-edito`

Just replace gvim with your editor's exe and make sure it is on the path.

You can manage environment variables of your notebook without restarting the jupyter server process, `\%env` is the most convenient way. Running `\%env` without any arguments lists all environment variables.

```
# The line below sets the environment variable OMP_NUM_THREADS
%env OMP_NUM_THREADS=4
```

```
env: OMP_NUM_THREADS=4
```

The`\%run` magic runs a scipt along the stated path and prints the results to the cell output. Useful to run external scripts not coded in the notebook itself. Just be sure to copy the script with the notebook. The next cell writes a script to the current directory and then the following cell executes it.

```
%%file helloipython.py
print('Hello IPython!')
```

```
Overwriting helloipython.py
```

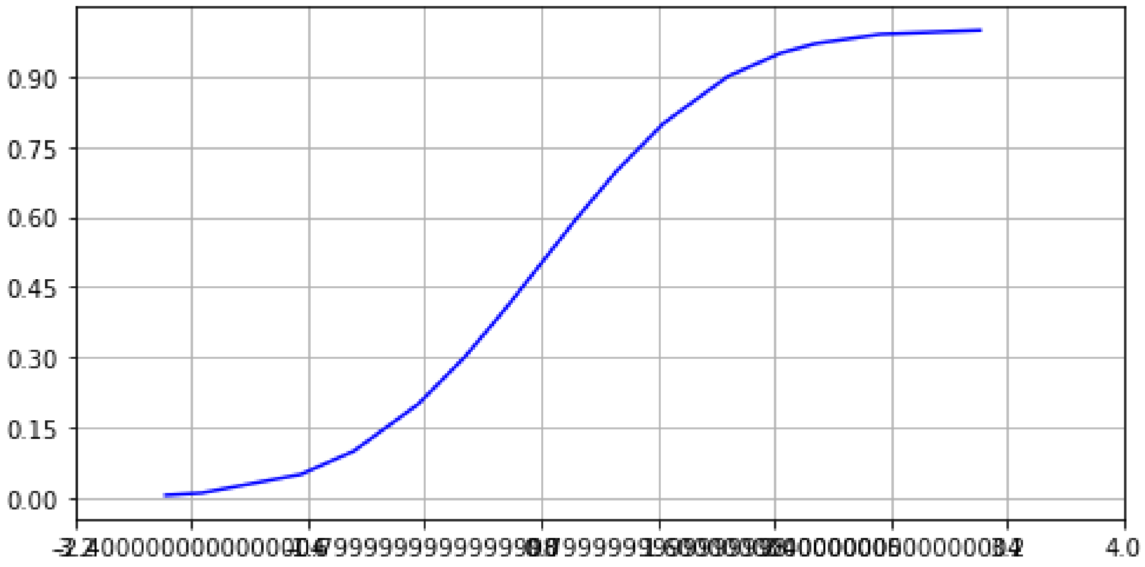
```
%run helloipython.py
```

```
Hello IPython!
```

`\%run` can also run Jupyter/IPython notebooks and insert the output of the notebook into the current result cell.

```
%run ./PlayStats.ipynb
```

```
[[ 0.005       -2.5758293 ]
 [ 0.01       -2.32634787]
 [ 0.05       -1.64485363]
 [ 0.1         -1.28155157]
 [ 0.2         -0.84162123]
 [ 0.3         -0.52440051]
 [ 0.4         -0.2533471 ]
 [ 0.5         -0.         ]
 [ 0.6          0.2533471 ]
 [ 0.7          0.52440051]
 [ 0.8          0.84162123]
 [ 0.9          1.28155157]
 [ 0.95         1.64485363]
 [ 0.97         1.88079361]
 [ 0.99         2.32634787]
 [ 0.9987       3.01145376]]
```



```
Stored 'table' (ndarray)
```

The %store command lets you pass variables between two different notebooks. In the PlayStats.ipynb the `table` variable is stored as follows:

```
# store the variable in the server for another notebook to read it
\%store table
```

If the other notebook has been executed in the present Jupyter session, the data can be retrieved in this notebook by

```
\%store -r table
print(table)
```

The %timeit magic can time the execution of a an expression. It can be one line or multiline statement. In a one liner we can pass through multiple ones separated by semicolon.

http://pynash.org/2013/03/06/timing-and-profiling.html[58]

```
%timeit range(100)
```

```
129 ns  1.61 ns per loop (mean  std. dev. of 7 runs, 10000000 loops ↵
   each)
```

`\% pastebin 'file.py'` to upload code to pastebin and get the url returned.

`\% bash` to run cell with bash in a subprocess.

`\%mprun & \%memit`: See how much memory a script uses (line-by-line, or averaged over a bunch of runs)

`\%prun statement\_name` will give you an ordered table showing you the number of times each internal function was called within the statement, the time each call took as well as the cumulative time of all runs of the function.

`\%\% HTML`: to render the cell as HTML. So you can even embed an image or other media in your notebook

```
\%\%HTML
<img src="http://storage.proboards.com/6578018/thumbnailer/cQSQRzgbljQLzGPJ3hfZ.jpg
```

`\%\%latex` to render cell contents as LaTeX, see here[59]

$$\begin{aligned}
\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial\vec{\mathbf{E}}}{\partial t} &= \frac{4\pi}{c}\vec{\mathbf{j}} \\
\nabla \cdot \vec{\mathbf{E}} &= 4\pi\rho \\
\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial\vec{\mathbf{B}}}{\partial t} &= \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} &= 0
\end{aligned} \tag{2.1}$$

```
from IPython.display import Latex
Latex(r"""\begin{eqnarray}
\nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf↵
   {E}}}{\partial t} & = \frac{4\pi}{c}\vec{\mathbf{j}} \\
\nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
\nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\↵
   mathbf{B}}}{\partial t} & = \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} & = 0
\end{eqnarray}""")
```

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial\vec{\mathbf{E}}}{\partial t} = \tfrac{4\pi}{c}\vec{\mathbf{j}} \tag{2.2}$$

$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho \tag{2.3}$$

$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial\vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}} \tag{2.4}$$

$$\nabla \cdot \vec{\mathbf{B}} = 0 \tag{2.5}$$

2.19 Magics for using IPython for numeric and scientific work

Early tutorials advised that you start the IPython kernel with the `--pylab=inline` option to load a bunch of packages and see Matplotlib graphs in the browser. As from IPython 2.0, the server advises you not to use this option, because it pre-loads a number of modules and packages that may not be required and may even interfere with your intended work.

Don't use Then there is the `\%pylab` magic command, which essentially does the same as the `--pylab=inline` option. The full command is [60]. When using this magic, IPython loads numpy and matplotlib. The following libraries are imported in this magic:

```
import numpy
import matplotlib
from matplotlib import pylab, mlab, pyplot
np = numpy
plt = pyplot

from IPython.display import display
from IPython.core.pylabtools import figsize, getfigs

from pylab import *
from numpy import *
```

Clearly the last two imports could potentially cause namespace conflicts with other modules. This is because `import *` is not good practice[61]. If you use the form `\%pylab --no-import-all` the last two * imports will not be executed. So the `\%pylab` magic command could be something like `\%pylab --no-import-all inline` to get inline plots in the notebook.

But this method still clutters the IPython interactive namespace with global pylab names, potentially causing problems.

Use Do use the `\%matplotlib [gtk|gtk3|inline|osx|qt|qt4|tk|wx]` magic to define the Matplotlib plotting backend without importing anything into the IPython interactive namespace. For the full discussion see here[62]. So the preferred method to get Matplotlib graphics inline in the notebook is to use the magic command

`\%matplotlib inline`

After using this magic, you still have to import numpy manually.

Note that the file format to which Matplotlib renders a graphic can be set by the following magic (svg, png or high resolution png):

`\%config InlineBackend.figure\_format = 'svg'`

`\%config InlineBackend.figure\_format = 'png'`

`\%config InlineBackend.figure\_format = 'retina'`

`https://stackoverflow.com/questions/17582137/ipython-notebook-svg-figures-by-defau`

2.20 Results display in Results Cell

Normally IPython will display the *last* unassigned result from the cell in the result cell. You can modify the `ast\_note\_interactivity` kernel option to make jupyter do this for all unassigned variables.

If you want to set this behaviour for all instances of Jupyter (Notebook and Console), simply create a file `~/.ipython/profile\_default/ipython\_config.py` with the lines below.

```
c = get\_config()
# Run all nodes interactively
c.InteractiveShell.ast\_node\_interactivity = "all"
```

https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

```
5
a = 6
7
```

```
5
```

```
7
```

2.21 Writing functions in other languages

This example is taken from Josh Devlin[64], thanks Josh!

You can write functions in cython or fortran and use those directly from python code. First you'll need to install: pip install cython fortran-magic Then

```
\%load\_ext Cython

\%\%cython
def myltiply\_by\_2(float x):
    return 2.0 * x
```

Then in some further-down cell: myltiply_by_2(23.)

Or

```
\%load\_ext fortranmagic

\%\%fortran
subroutine compute\_fortran(x, y, z)
    real, intent(in) :: x(:), y(:)
    real, intent(out) :: z(size(x, 1))

    z = sin(x + y)

end subroutine compute\_fortran
```

Then in some further-down cell:

`compute\_fortran([1, 2, 3], [4, 5, 6])`

`http://arogozhnikov.github.io/2015/11/29/using-fortran-from-python.html`[65]

2.22 IPython notebook extensions

2.22.1 contrib nbextensions

The following repository contains a collection of extensions that add functionality to the Jupyter notebook. There are several extensions, best visit the repo for more information.

https://github.com/ipython-contrib/jupyter_contrib_nbextensions

https://github.com/ipython-contrib/IPython-notebook-extensions

https://github.com/ipython/ipython/wiki/Extensions-Index

The following commands will install the extensions, as well as a menu based configurator that will help you browse and enable the extensions from the main Jupyter notebook screen.

Method 1

https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator

Method 2 The install instructions are taken from Josh Devlin[66]. There is a risk that the following installation may not succeed on your Jupyter installation, depending on software version status.

pip install –upgrade https://github.com/ipython-contrib/jupyter_contrib_nbextensions/tarball/master pip install –upgrade jupyter_nbextensions_configurator jupyter contrib nbextension install –user jupyter nbextensions_configurator enable –user

You can install Nbextensions any time from your command line like this

```
conda install -c conda-forge jupyter\_contrib\_nbextensions
conda install -c conda-forge jupyter\_nbextensions\_configurator
jupyter contrib nbextension install --user
```

Once they're installed, you'll see an Nbextensions tab. Explore away!

2.22.2 General notes

The IPython architecture supports the installation of extension packages to add new functionality to notebooks.

http://ipython.org/ipython-doc/dev/config/extensions/[67]

Several extensions exist to access other software tools from within IPython:

http://jupyter.cs.brynmawr.edu/hub/dblank/public/Jupyter\%20Help.ipynb[68]

http://ipython.org/ipython-doc/dev/config/extensions/[69]

http://nbviewer.ipython.org/github/jrjohansson/ipython-asymptote/blob/master/Asymptote-examples.ipynb[70] for Asymptote[71]

http://www2.ipp.mpg.de/~mkraus/python/tikzmagic.py[72] and http://www2.ipp.mpg.de/~mkraus/python/tikzmagic_test.ipynb[73] for tikz[74].

On Windows, when using Anaconda, the notebook extensions are installed here:

C:\Anaconda\share\jupyter\nbextensions

On raw Python installations the notebook extensions appear to be installed here:

```
C:\Users\YourUserName\.ipython\nbextensions
C:\Users\YourUserName\.ipython\profile\_default\static\custom
```

2.22.3 ICalico spell checker

The ICalico spell checker (thanks Doug Blank!) checks spelling and underlines words that appear incorrect. The spell checker is implemented in JavaScript and works on markdown cells in edit mode. It points out spelling errors but do not offer corrections at current. The word list is US English, so it is not very friendly towards UK English. The dictionary can be changed, see below.

To use the spell checker open a markdown cell for editing and click on the 'tickmark' button on the toolbar. The tickmark button will only be present if you installed and activated the spell checker, and then restarted the jupyter server.

The instructions are somewhat confusing because of different versions of the Jupyter notebook and different versions (and repository locations) of the extension.

To install and activate the extension follow the YouTube video below, or the instructions at `http://calicoproject.org/Icalico`[75] and modified here[76]. The installation requires at least the first two steps:

1. Download the extension (do this once only) - I am not sure which to use for the different combinations of Jupyter and repos:

2. Let Jupyter download it for you:

   ```
   \verb+      !jupyter nbextension install https://github.com/Calysto/notebook-
   ```

   Let Jupyter download it for you:

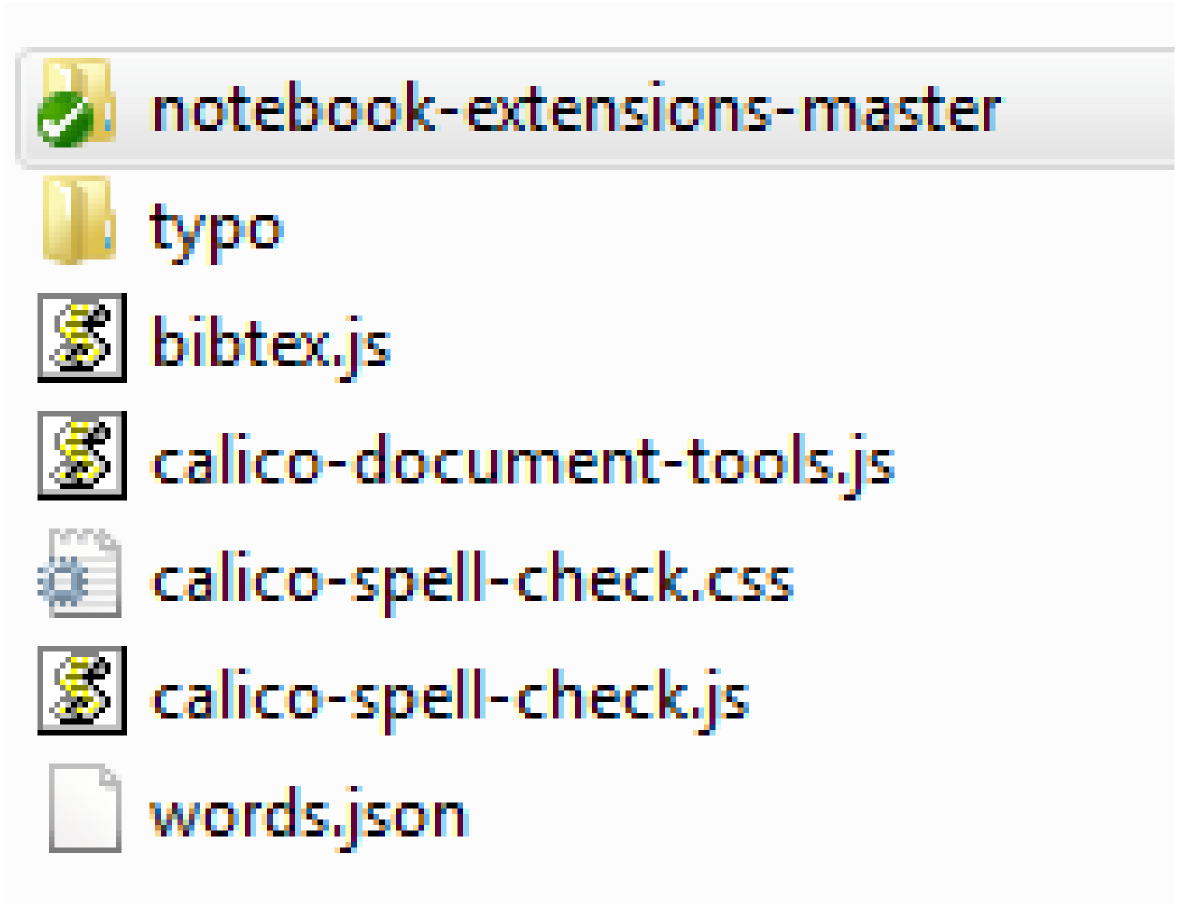
   ```
   \verb+      !jupyter nbextension install https://github.com/Calysto/notebook-
   ```

   or

   ```
   \verb+      !jupyter nbextension install https://github.com/Calysto/notebook-
   ```

3. Download manually: Clone the repo at `https://github.com/Calysto/notebook-extensions` into the local directory `C:\ProgramData\jupyter\nbextensions\notebook-extensions-maste` Note that you need to clone to a different directory name than is the repo name. Make a copy of the files shown below, to one level lower than where you cloned, in `C:\ProgramData\jupyter\nbexten`

   Download manually: Clone the repo at `https://github.com/Calysto/notebook-extensions` into the local directory `C:\ProgramData\jupyter\nbextensions\notebook-extensions-maste` Note that you need to clone to a different directory name than is the repo name. Make a copy of the files shown below, to one level lower than where you cloned, in `C:\ProgramData\jupyter\nbexten`


notebook-extensions-master
typo
bibtex.js
calico-document-tools.js
calico-spell-check.css
calico-spell-check.js
words.json

4. Activate it by executing the following:

   !jupyter nbextension enable calico-document-tools

5. To activate the spell checker for Jupyer 4.x notebooks, edit the file `C:\Users\YourUserName\.jupyter` in your Jupyter profile and conform that the following load_extensions commands are present (add in the appropriate place if necessary): "load_extensions": "calico-spell-check":true, "calico-document-tools": true, "calico-cell-tools":true

The ICalico spell checker discussion takes place here:

`https://github.com/ipython/ipython/issues/3216`[77]

```
from IPython.display import YouTubeVideo
# a talk about the ICalico spell checker extension
YouTubeVideo('Km3AtRynWFQ')
```

2.22.4 UK English spell checker

I have not yet had the time to figure out how to do this for Jupyter 4

Marco Pinto[78] maintains a UK English hunspell-style list here[79]. To implement the UK dictionary in an Anaconda Jupyter installation on Windows:

1. Download the two files `en-GB.aff` and `en-GB.dic` into the (new) folder `C:\Anaconda\share\jupyter`

2. Rename the two files to use underscore instead of dash/hyphen (look at the en_US equivalent). Change `en-GB.aff` to `en\_GB.aff` and change `en-GB.dic` to `en\_GB.dic`.

3. Edit the file `C:\Anaconda\share\jupyter\nbextensions\calico-spell-check.js` to replace `var lang = "en\_US";` with `var lang = "en\_GB";`.

It should be a simple matter to find the equivalent directories in Linux.

So now at least we have a UK dictionary, the remaining work is to add new words. Marco Pinto's GUI-based tool at `http://marcoagpinto.cidadevirtual.pt/proofingtoolgui.html`[80] is exceptionally suitable tool for this purpose. Doug Blank also pointed out: "Also, one can add words to the `calico-spell-check extension` by making a JSON object in a file named `words.json` and putting it next to `calico-spell-check.js`."

2.23 Multicursor editor support

Jupyter supports mutiple cursors, similar to Sublime Text. Simply click and drag your mouse while holding down `Alt`.

2.24 Optimising with IPython

`https://robotwhale.wordpress.com/2014/08/17/optimization-with-ipython/`[81]

2.25 Local files

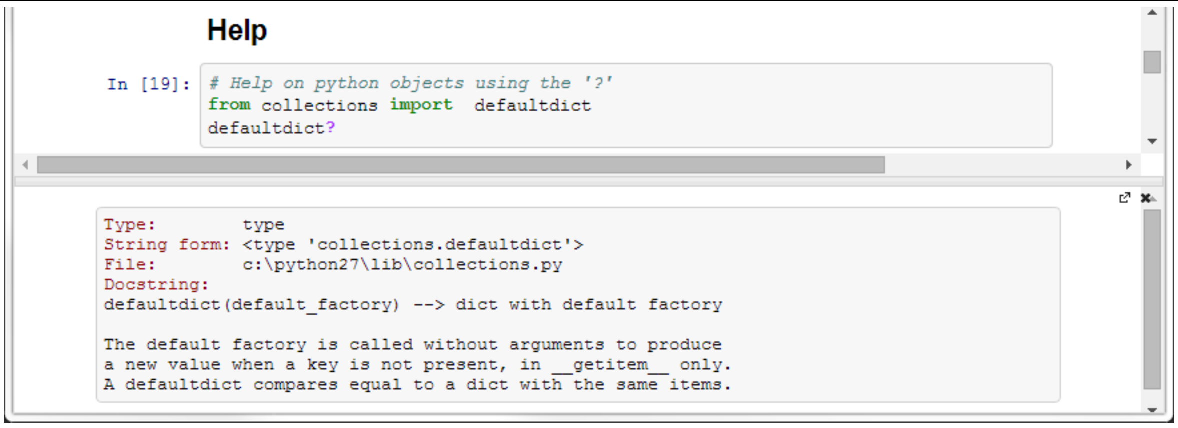
If you have local files in your Notebook directory, you can refer to these files in Markdown cells via relative URLs that are prefixed with files/ as in:

files/[subdirectory/]filename

2.26 Help

Introspection help is available by typing the object's name followed by a question mark, then execute the cell. It will print details about the object, including docstrings, function call argument) and class constructor details. Double click on the divider to close the help console.

```
from collections import  defaultdict
# defaultdict?
display(Image(filename='images/introspection.png'))
```



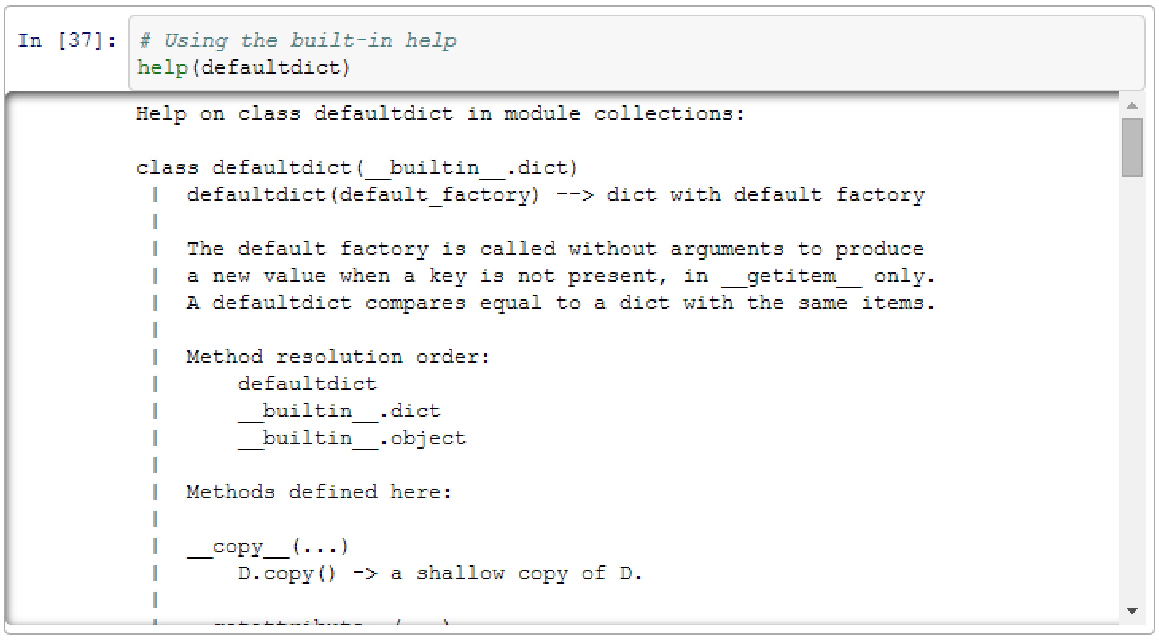
Help

```
In [19]: # Help on python objects using the '?'
         from collections import  defaultdict
         defaultdict?
```

```
Type:        type
String form: <type 'collections.defaultdict'>
File:        c:\python27\lib\collections.py
Docstring:
defaultdict(default_factory) --> dict with default factory

The default factory is called without arguments to produce
a new value when a key is not present, in __getitem__ only.
A defaultdict compares equal to a dict with the same items.
```

You can also access the built in Python help system, by typing help(objectname), and then execute the cell.

```
# help(defaultdict)
display(Image(filename='images/python-help.png'))
```

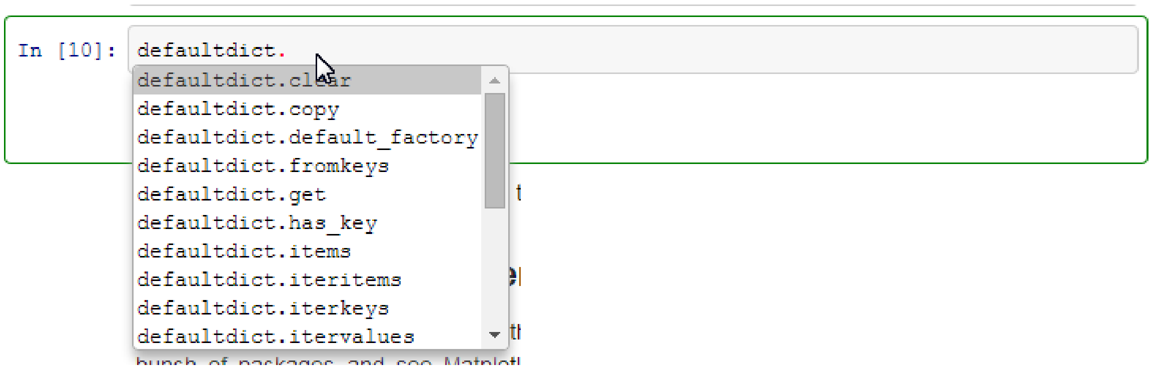


```
In [37]: # Using the built-in help
         help(defaultdict)
```

```
Help on class defaultdict in module collections:

class defaultdict(__builtin__.dict)
 |  defaultdict(default_factory) --> dict with default factory
 |
 |  The default factory is called without arguments to produce
 |  a new value when a key is not present, in __getitem__ only.
 |  A defaultdict compares equal to a dict with the same items.
 |
 |  Method resolution order:
 |      defaultdict
 |      __builtin__.dict
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  __copy__(...)
 |      D.copy() -> a shallow copy of D.
 |
```

Tab-completion help is also available, just press TAB after the period

```
# defaultdict.
display(Image(filename='images/tabcompletion.png'))
```



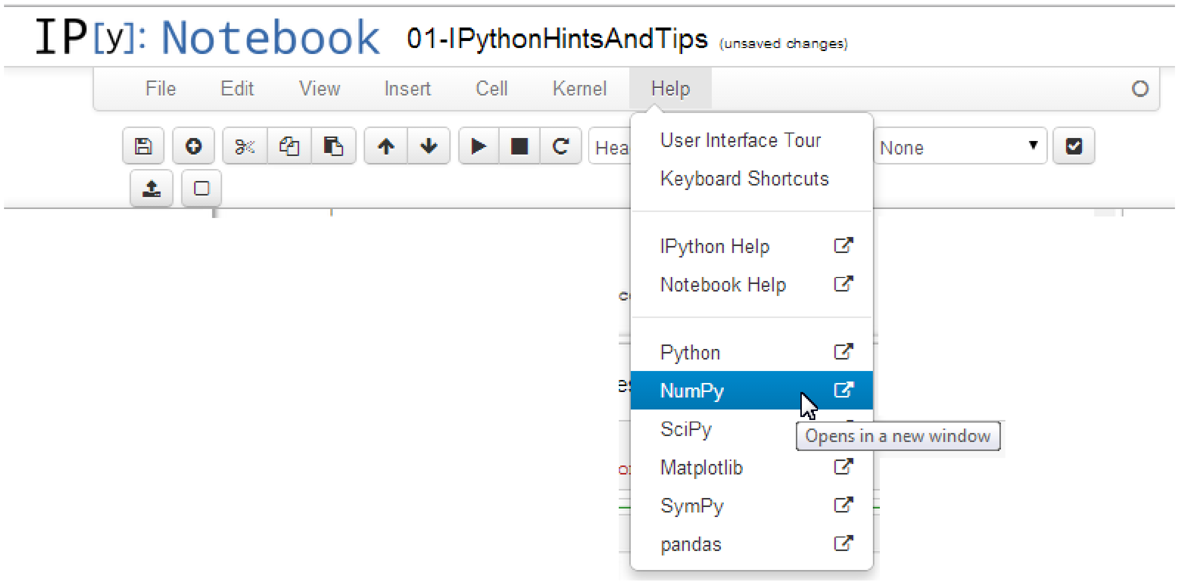
```
In [10]: defaultdict.
         defaultdict.clear
         defaultdict.copy
         defaultdict.default_factory
         defaultdict.fromkeys
         defaultdict.get
         defaultdict.has_key
         defaultdict.items
         defaultdict.iteritems
         defaultdict.iterkeys
         defaultdict.itervalues
```

You can also obtain the docstring by prepending a function with a question mark and then executing.

```
?str.replace()
```

The IPython *Help* menu item has links to IPython and the scientific packages

```
display(Image(filename='images/ipythonhelp.png'))
```



2.27 Notebooks remember, but not the way you might think

The notebook visible in the browser is not really the notebook, it is only a rendering of the notebook. The actual notebook and its data resides in the IPython kernel. The browser is connected to the kernel via a zmq channel, for the purpose of rendering and input. You can close all the browser windows, but the code and data still resides in the kernel process. As long as the kernel is running you can open a new browser window at 127.0.0.1:8888 and the notebook will be displayed.

The results (including graphs and images) from previous runs are recorded in the notebook, so when it is rendered the previously stored results are shown. The previous results are overwritten only if the cell is executed again.

Most important, the results from a previous cell execution remains in the kernel, even if the cell is removed or moved up in the notebook. It happens from time to time that you execute a cell in a given location, creating some data in the kernel. The next cell uses this information in a calculation. So far, so good. Now you decide to re-organise the notebook and move the 'next cell' to a position earlier in the notebook, even before the cell that created the information in the first place. The newly moved cell still works in its new location, because the information is still in memory. At the end of work, you save and close the notebook and exit the kernel.

Tomorrow you start a new kernel and load the notebook. To ensure that all is fresh and well, you decide to execute all cells in the notebook. But now the moved cell does not work, because its input information is not yet created - it is only created a few cells into the future. Yet it did work yesterday (because of results still remained in memory after moving the cell). But today it fails because there is no memory of something that is yet to come.

It is important to understand that the concept of the non-existence of data prior to cell execution does not apply if you move cells around in the notebook. All results are remembered and accessible by cells *linearly before* the cell that created the information.

Similarly, remember that a cell changes information for prosperity, also for subsequent runs of the same cell. For example, suppose we assign a value to a variable in the next cell and in the cell thereafter increment the value. If the first cell is executed once and the second cell is executed repeatedly, the value will increase much more than it would be if the program was executed from start to finish with each cell executed once only.

```
#run this cell once
a = 5
```

```
#run this cell several times
a = a + 1
print(a)
```

```
6
```

A similar error occurs in Pandas if the first cell creates a dataframe, and the second cell adds a new column to the dataframe. If the second cell is executed a number of times, many columns will be added, which was not the intention with the code *if executed linearly from start to end*.

Therefore be careful with cells that modifies its own input data - such data is not static and changes with each invocation of the cell.

2.28 Reloading imports

Python's import facility loads a file only once, if the `import` is encountered again for the same module, the import is ignored. If you are actively developing the the module to be imported the module contents changes all the time and these changes must be imported to see their effect. In this case you want to force another import execution. This can be done with the `\%load\_ext autoreload` magic command. If the extension is already loaded, Ipython may complain, so I use `\%reload\_ext autorelo` which attempts to load or reload, as appropriate.

`http://ipython.org/ipython-doc/rel-1.1.0/config/extensions/autoreload.html`[82]

```
%reload_ext autoreload
%autoreload 2
import numpy as np
```

2.29 Clearing the IPython memory

If for some reason you need to clear out all memory in your notebook this can be done by selecting the `Kernel Restart` menu option. This will clear the kernel memory, including the memory of all open notebooks. So use this avenue with care. After restarting the kernel, all notebooks must be re-run from start to build the information again.

You can use the Cell | All Output | Clear menu option to *remove all output* from a notebook. It will be much smaller, but also empty of any output or embedded media. To see the full notebook with all calculation results, you would have to run all cells again.

2.30 Markdown (MD) language syntax

Markdown is a simplified syntax for simple text layout. It allows you to embed simple formatting commands in a regular text file. In an ASCII text editor you will see the markup but not formatted. You can see the formatted version in a local or online markdown editor.

1. https://en.wikipedia.org/wiki/Markdown
2. https://daringfireball.net/projects/markdown/
3. https://help.github.com/articles/markdown-basics/

You can write markdown in any text editor or in a dedicated markdown editor:

1. https://notepad-plus-plus.org/
2. https://atom.io/
3. http://www.sitepoint.com/best-markdown-editors-windows/
4. http://www.sublimetext.com/ (paid software)
5. http://markdownpad.com/ (paid software)

IPython uses the markdown syntax for text in its non-code cells. When IPython creates a new cell it is a code cell by default, you must remember to change it to a markdown cell on the menu.

You can also edit markdown online here:

1. https://github.com/benweet/stackedit
2. http://dillinger.io/
3. http://markdownlivepreview.com/

One confusing matter is the fact there are different variants of markdown. For more details on the syntax see

`https://stackoverflow.com/editing-help`[83] `http://nbviewer.ipython.org/github/ipython/ipython/blob/master/examples/notebooks/Part\%204\%20-\%20Markdown\%20Cells.ipynb`[84] `http://johnmacfarlane.net/pandoc/demo/example9/pandocs-markdown.html`[85]

`http://johnmacfarlane.net/pandoc/README.html`[86]

`http://daringfireball.net/projects/markdown/`[87]

`http://daringfireball.net/projects/markdown/basics`[88]

`http://daringfireball.net/projects/markdown/syntax`[89]

`https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet`[90]

`http://support.iawriter.com/help/kb/general-questions/markdown-syntax-reference-gu`

You can also use embedded raw HTML into a markdown page, but you need to type a lot more than in pure markdown.

IPython uses the Pandoc converter to convert between different markup languages, so my guess is that it is prudent to work with the Pandoc variant of MD syntax (which is close to the original). The other major MD variant is the github MD syntax - which is somewhat different.

To force a newline you must end the current line with two spaces.

If your markdown does not want to work right, first try to leave a blank line before the offending text.

2.30.1 Markdown Cells

Type this to get the output shown below:

```
Markdown basics: lists, markup and code

* list item
* list item
  * nested <font size="3" color="red">list item</font>


* *italics*
* **bold**
* ‘fixed\_font‘

You can embed code meant for illustration instead of execution in Python - use fou

    def hello\_ipython():
        print "Hello IPython!"
```

Markdown basics: lists, markup and code

- list item
- list item
- nested list item
- *italics*
- **bold**
- `fixed\_font`

You can embed code meant for illustration instead of execution in Python - use four spaces before:

```
def hello\_ipython():
    print "Hello IPython!"
```

A key and important point is that IPython markdown can take embedded HTML of any form.

Therefore in applications where the markdown syntax it too weak, use HTML. Be aware however that not all of the HTML constructs can be converted to some of the conversion output formats (e.g., embedding video in a LaTeX document).

Type this to get the output shown below:

```
Markdown cells can also contain HTML

<p>Markdown basics: lists, markup and code</p>

<ul>
<li>list item</li>
<li><p>list item</p>

<ul>
<li>nested <font size="3" color="red">list item</font></li>
</ul></li>
<li><p><em>italics</em></p></li>
<li><strong>bold</strong></li>

<li><code>fixed\_font</code></li>
</ul>

<p>Code examples:</p>

<pre><code>def hello\_ipython():
    print "Hello IPython!"
</code></pre>
```

Markdown cells can also contain HTML

Markdown basics: lists, markup and code

- list item
- list item
  - nested list item
- *italics*
- **bold**
- `fixed\_font`

Code examples:

```
def hello\_ipython():
    print "Hello IPython!"
```

Type this to get the output shown below:

```
Using math mode (anything delimited before and after by single or double ‘$‘ symbol
```

```
\\begin\{equation\} D\_\{KL\}\(P||Q\) = \\sum\\limits\_\{i\}ln \(\\frac\{P\(i\)\}\
```

Using math mode (anything delimited before and after by single or double $ symbols is interpreted as LaTeX math:

$$D_{KL}(P||Q) = \sum_{i} ln(\frac{P(i)}{Q(i)})P(i) \tag{2.6}$$

$$e^{i\pi} + 1 = 0 \tag{2.7}$$

2.30.2 Github markdown can also be used

The Notebook webapp support Github flavored markdown meaning that you can use triple backticks for code blocks

```
‘‘‘python
print "Hello World"
‘‘‘

‘‘‘javascript
console.log("Hello World")
‘‘‘
```

gives

```
print "Hello World"
```

```
console.log("Hello World")
```

And a table like this :

```
| This | is   |
|------|------|
|   a  | table|
```

gives

This	is
a	table

2.30.3 Citations and links

Markdown syntax does not have any means to provide citations as often required in formal documentation.

Markdown hyperlinks to external websites: pyradi[92] Python package. You can provide the display text, the ling address and amouse-over attribute.

HTML anchors can be used to create links to other locations in the same file. Put an anchor definition in the place you want to link to, using this format: `<a name="MyAnchor">`. It seems the best place to put this anchor is in a dedicated markdown cell immediatelty in front of where you want to link to, it does not work in a cell with other text. From elsewhere in the file link to this anchor by using the following format

`<a href="#MyAnchor">my anchor</a>`

It is also possible to attach anchors to headers as explained in `https://stackoverflow.com/questions/16630969/ipython-notebook-anchor-link-to-refer-a-cell-directly-from-outs`

`http://nbviewer.ipython.org/github/ipython/nbconvert-examples/blob/master/citations/Tutorial.ipynb`[94] - LaTeX citations in the notebook.

The LaTeX docs with template control[95] script provides full citation support when creating PDF documents.

2.30.4 Cross linking and table of contents

IPython does not yet support the automatic creation of a table of contents, but it can be added manually by placing an HTML anchor immediately before the section heading and then linking to the anchor. Note that in IPYthon the browser's 'back arrow' won't navigate back to the previous location in the page.

```
# Table of Contents

- [Overview](#Overview)
- [Learning Python and hints and tips](#LearningPythonandhintsandtips)
- [Introducing Python for scientific work](#IntroducingPythonforscientificwork)

<a name="Overview"></a>
##Overview

This notebook provides a brief summary ....

<a name="LearningPythonandhintsandtips"></a>
##Learning Python, and hints and tips

There are many free ....

<a name="IntroducingPythonforscientificwork"></a>
##Introducing Python for scientific work

A very good introduction to Python for scientific work ....
```

Table of Contents

2.31 IPython's rich display system

IPython notebook can use all of the modern browsers' capabilities. This is called the rich display system, see here.[99]

2.31.1 Plotting

Plotting packages and options

Scientific Plotting in Python[100]

Interactive Plotting in IPython Notebook (Part 1/2): Bokeh[101]

Interactive Plotting in IPython Notebook (Part 2/2): Plotly[102]

Overview of Python Visualization Tools [103]

Comparing Python web visualization libraries[104]

matplotlib matplotlib[105] Gallery[106] Examples[107] docs[108]

bokeh Welcome to bokeh[109] Gallery[110] Quickstart[111] Tutorials[112] User Guide[113]

plotly What is Plotly.js[114] Plotly.js Open-Source Announcement[115] Getting Started: Plotly for Python[116] Plotly Offline[117] User Guide[118] Plotly's Python API User Guide[119] Example 3-D plot[120]

2.31.2 Matplotlib in IPython notebook

IPython can inline `matplotlib plots`.

`http://nbviewer.ipython.org/urls/raw.github.com/jakevdp/matplotlib\_pydata2013/master/notebooks/05\_Animations.ipynb`[121] - animations

`http://nbviewer.ipython.org/urls/raw.github.com/jakevdp/matplotlib\_pydata2013/master/notebooks/03\_Widgets.ipynb`[122] - interactivity

`http://nbviewer.ipython.org/github/dpsanders/matplotlib-examples/blob/master/colorline.ipynb`[123] - multi-colour lines

```
%matplotlib inline

import pylab as pl
import numpy as np

t = np.arange(0.0, 2.0, 0.01)
s = np.sin(2*np.pi*t)
```

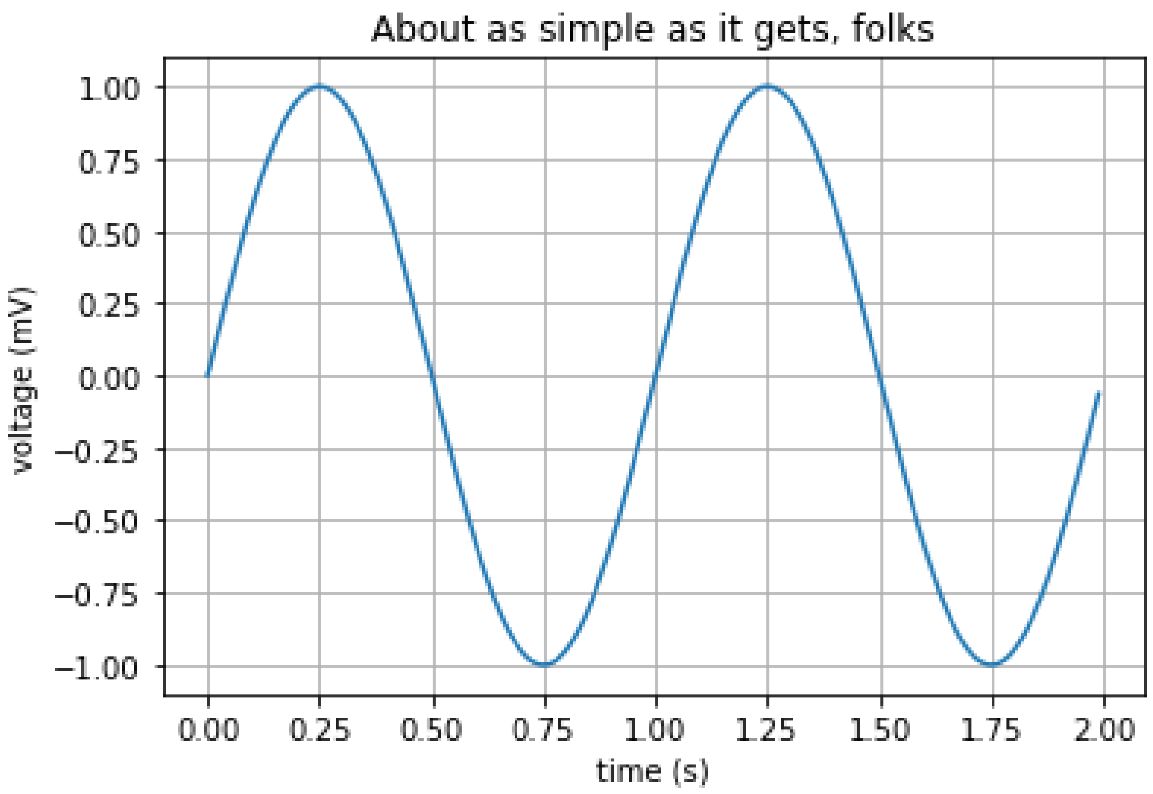
```
pl.plot(t, s)
pl.xlabel('time (s)')
pl.ylabel('voltage (mV)')
pl.title('About as simple as it gets, folks')
pl.grid(True)
# savefig("test.png")
# show()
```

```
[<matplotlib.lines.Line2D at 0x17ad7971580>]
```

```
Text(0.5, 0, 'time (s)')
```

```
Text(0, 0.5, 'voltage (mV)')
```

```
Text(0.5, 1.0, 'About as simple as it gets, folks')
```



By just importing seaborn, the Matplotlib graphs is given a different style. If seaborn is not installed do `conda install seaborn`.

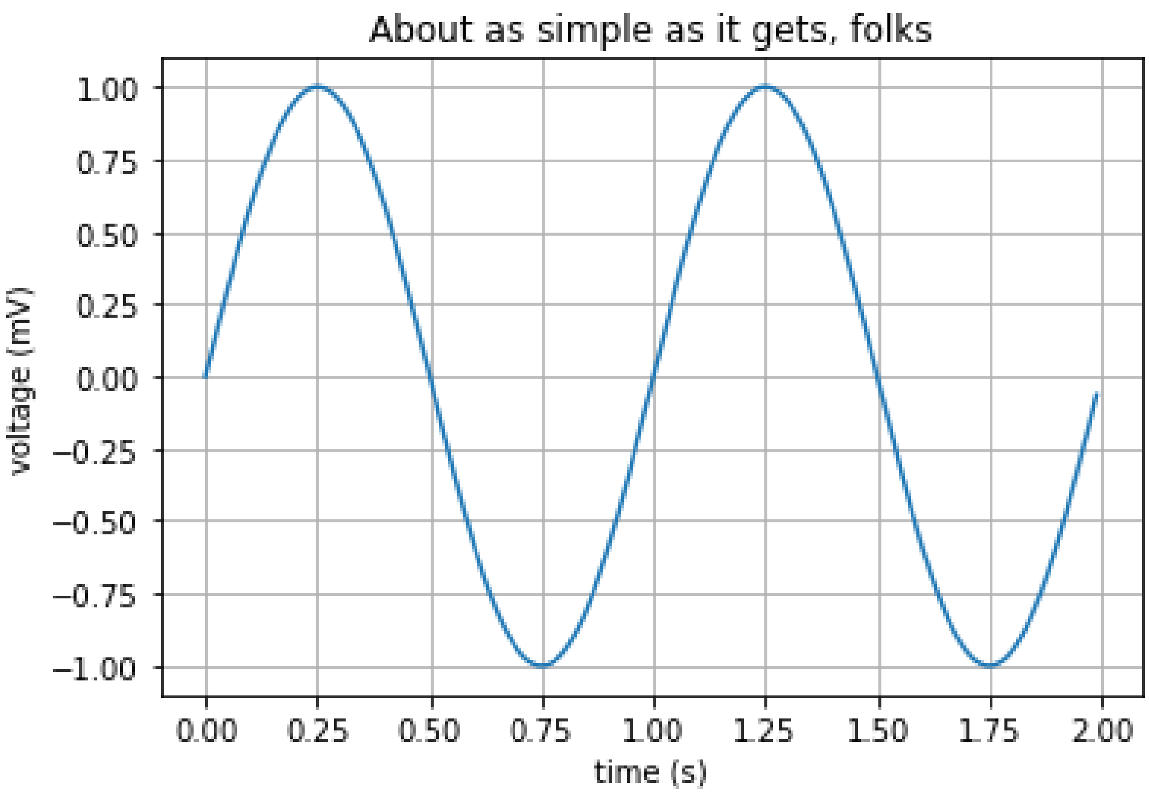
```
import seaborn as sns

t = np.arange(0.0, 2.0, 0.01)
s = np.sin(2*np.pi*t)
pl.plot(t, s)
pl.xlabel('time (s)')
pl.ylabel('voltage (mV)')
pl.title('About as simple as it gets, folks')
pl.grid(True)
# savefig("test.png")
# show()
```

```
[<matplotlib.lines.Line2D at 0x17ad7c503d0>]
```

```
Text(0.5, 0, 'time (s)')
```

```
Text(0, 0.5, 'voltage (mV)')
```

```
Text(0.5, 1.0, 'About as simple as it gets, folks')
```



2.31.3 Matplotlib in qt window

Use the ipython magic command pylab to control the graphing backend, switch between inline and qt as required.

`http://stackoverflow.com/questions/14261903/how-can-i-open-the-interactive-matplot`
14277370#14277370[124]

2.31.4 Seaborn distribution plotting

The seaborn[125] package provides special means to plot distributions

`http://nbviewer.ipython.org/github/mwaskom/seaborn/blob/master/examples/plotting\`
`_distributions.ipynb`[126]

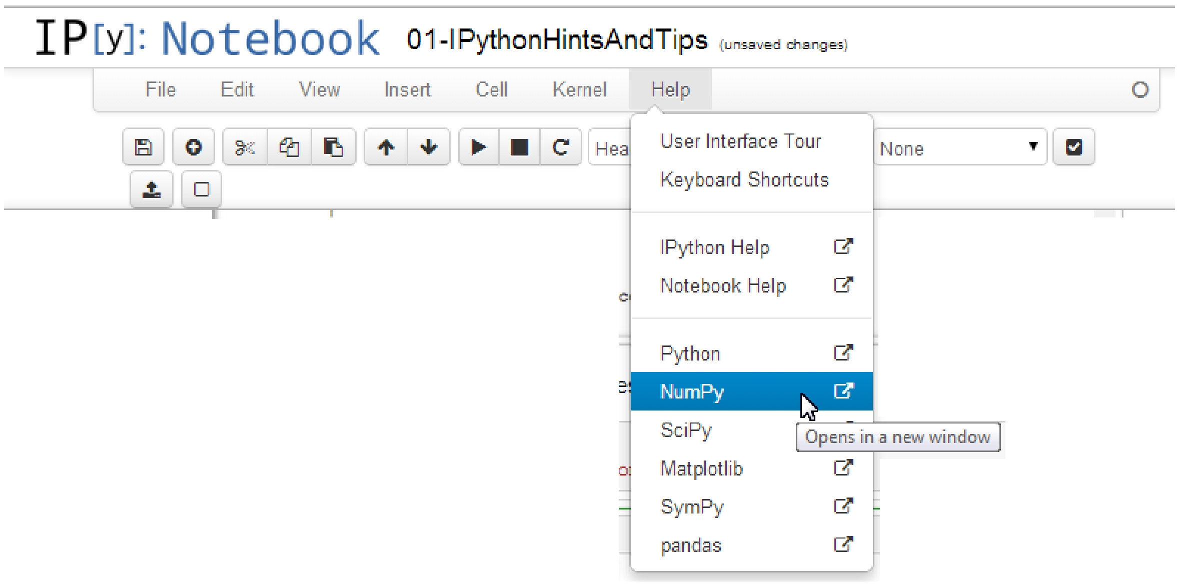
2.31.5 Including images into the notebook

There are (at least) two ways to include images in the notebook.

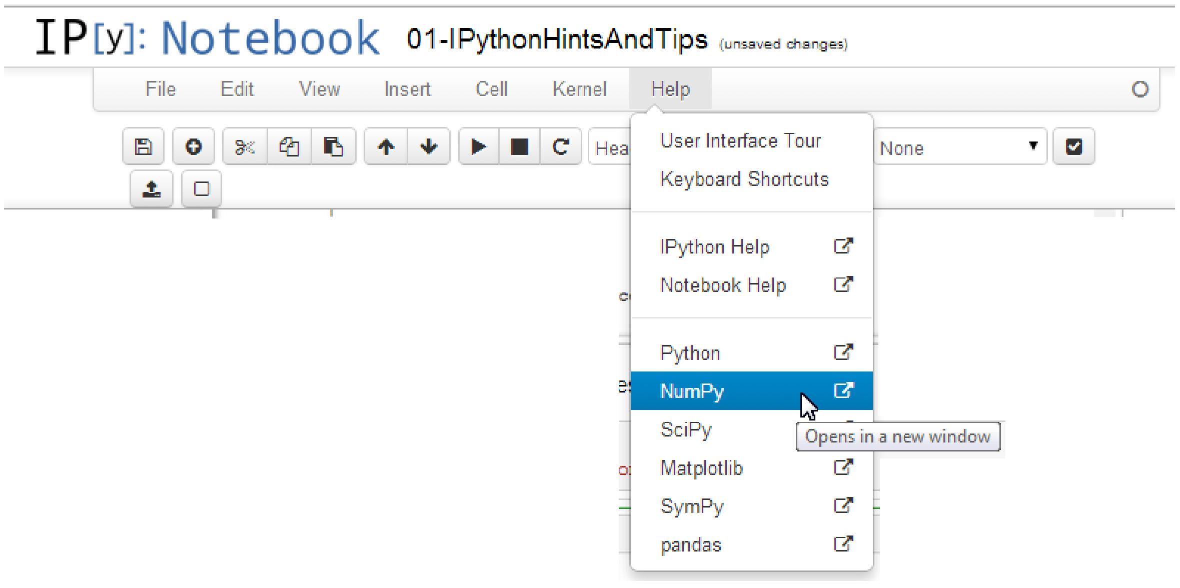
```
display(Image(filename='images/portalpage.png'))
HTML('<img src="files/images/portalpage.png" width=600 height=600/>')
```

The first form includes the image in its natural size, but the size can be adjusted by width and height function parameters. The second form injects HTML code and, likewise, allows you to set the image size.

```
HTML('<img src="images/ipythonhelp.png" width=400 height=200/>')
```

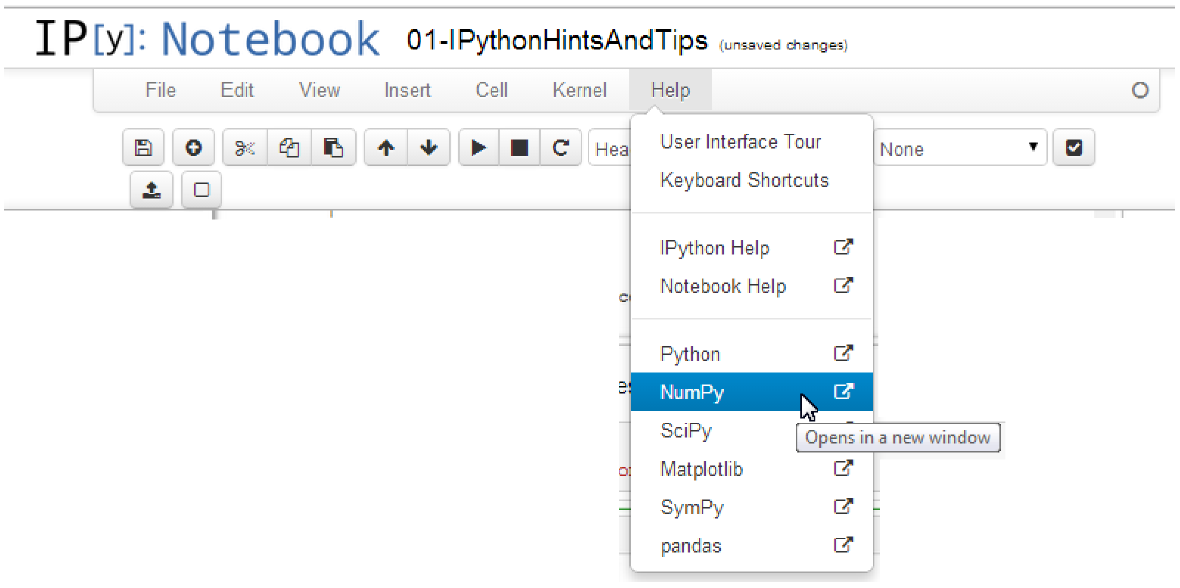


```
display(Image(filename='images/ipythonhelp.png', width=250, height=250)↵
    )
```



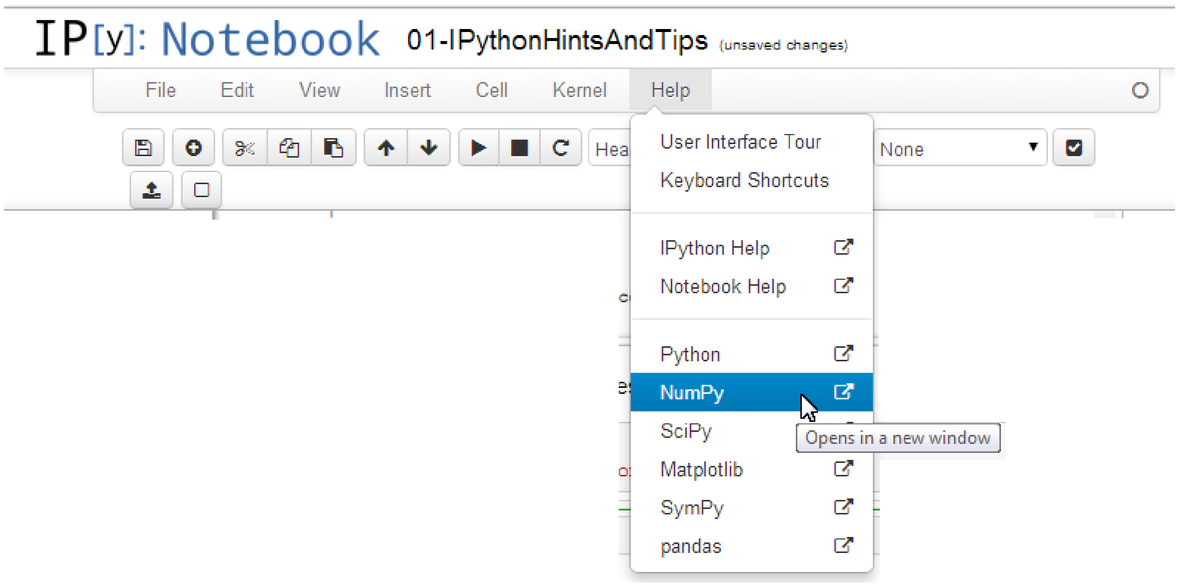
Images can also be included as markdown by using the following format

```
!['replacement test'](images/ipythonhelp.png)
```



Images can also be included as markdown by using the following format

`<img src="images/ipythonhelp.png" width="200">`



2.31.6 Embedded vs non-embedded images

As of IPython 0.13, images are embedded by default for compatibility with QtConsole, and the ability to still be displayed offline.

http://www.slideviper.oquanta.info/nbcreveal/sky_test.html?theme=sky#/7[127]

```
# by default Image data are embedded
picUrl = 'https://raw.githubusercontent.com/NelisW/pyradi/master/pyradi↵
   /doc/_images/pyradi.png'
Embed  = Image(picUrl)
display(Embed)

# if kwarg `url` is given, the embedding is assumed to be false
# SoftLinked = Image(url=picUrl)
# In each case, embed can be specified explicitly with the `embed` ↵
   kwarg
# ForceEmbed = Image(url=picUrl, embed=True)
```

2.31.7 Embedding other media

SVG graphic.

```
from IPython.display import SVG
SVG(filename='images/solidangleflatplate.svg')
```

```
<IPython.core.display.SVG object>
```

2.31.8 Embed a video from YouTube.

```
from IPython.display import YouTubeVideo
# a talk about IPython at Sage Days at U. Washington, Seattle.
# Video credit: William Stein.
if False:
    YouTubeVideo('1j_HxD4iLn8')
```

2.31.9 Embed an external web page.

```
if False:
    HTML('<iframe src=https://en.wikipedia.org/wiki/Einstein width=700 ↵
       height=350></iframe>')
```

2.31.10 Embed a video from local file system

The following shell shows a recording of the Mayavi display. The file is locally saved in the notebook. It seems that the format must be webm format, the other formats (avi, mp4) did not work.

```
# display a locally saved video file.
# it seems that only webm format works here
import io
import base64
from IPython.core.display import HTML

filename = './images/interpolationSphere.webm'

# video = io.open(filename, 'r+b').read()
# encoded = base64.b64encode(video)
# HTML(data='''<video alt="Data set video" controls>
#              <source src="data:video/mp4;base64,{0}" type="video/mp4" ↵
   />
#              </video>'''.format(encoded.decode('ascii')))

HTML("""
<div align="middle">
<video width="40%" controls>
      <source src="{}" type="video/mp4">
</video></div>""".format(filename))
```

2.32 Interactive widgets

IPython includes an architecture for interactive widgets that tie together Python code running in the kernel and JavaScript/HTML/CSS running in the browser. These widgets enable users to explore their code and data interactively. For details see

http://nbviewer.ipython.org/github/ipython/ipython/blob/master/examples/Interactive
%20Widgets/Index.ipynb[128]

https://github.com/ipython/ipython/tree/master/examples/Interactive\%20Widgets[129]

https://www.youtube.com/watch?v=VaV10VNZCLA[130]

https://www.youtube.com/watch?v=vE_CJTen15M[131]

https://www.youtube.com/watch?v=o7Tb7YhJZR0[132] https://www.youtube.com/watch?v=wxVx54ax47s[133]

https://github.com/ipython/ipython/wiki/Widgets[134]

The following examples are taken from

https://github.com/ipython/ipython-in-depth/tree/master/examples/Interactive\
%20Widgets[135]

Upgrading widgets from IPython 2 to 3:

http://ipython.org/ipython-doc/3/whatsnew/version3_widget_migration.html[136]

IPython widgets in Jupyter/IPython 4

https://keminglabs.com/print-the-docs-pdfs/1518024.pdf[137]

https://github.com/ipython/ipywidgets/blob/master/examples/notebooks/Widget\
%20Basics.ipynb[138]

The widget ecosystem changes frequently, the following code may not work....

```
# import IPython.html.widgets as widgets
from IPython.display import display
import ipywidgets
from ipywidgets import widgets

[n for n in dir(ipywidgets) if n[0] == n[0].upper() and not n[0] == '_'↵
   ]
```

```
['Accordion',
 'AppLayout',
 'Audio',
 'BoundedFloatText',
 'BoundedIntText',
 'Box',
 'Button',
 'ButtonStyle',
 'CallbackDispatcher',
 'Checkbox',
 'Color',
 'ColorPicker',
 'Combobox',
 'Controller',
 'CoreWidget',
 'DOMWidget',
 'DatePicker',
 'Datetime',
 'Dropdown',
 'FileUpload',
 'FloatLogSlider',
 'FloatProgress',
 'FloatRangeSlider',
 'FloatSlider',
 'FloatText',
 'GridBox',
 'GridspecLayout',
 'HBox',
 'HTML',
 'HTMLMath',
 'Image',
 'IntProgress',
 'IntRangeSlider',
 'IntSlider',
 'IntText',
 'Label',
 'Layout',
 'NumberFormat',
 'Output',
 'Password',
 'Play',
 'RadioButtons',
 'Select',
 'SelectMultiple',
 'SelectionRangeSlider',
 'SelectionSlider',
 'SliderStyle',
 'Style',
 'Tab',
 'Text',
 'Textarea',
 'ToggleButton',
 'ToggleButtons',
 'ToggleButtonsStyle',
 'TwoByTwoLayout',
 'VBox',
 'Valid',
 'ValueWidget',
 'Video',
 'Widget']
```

```
xx = widgets.FloatSlider(
    value=7.5,
    min=5.0,
    max=10.0,
    step=0.1,
    description='Test:',
)

y = ipywidgets.Checkbox(
    description='Check me',
    value=True,
)

w = ipywidgets.Dropdown(
    options = [ 'test 1', 'option 2', 'selection 3',],
    value='option 2',
    description='Number:',
)

#use ordered dic to get required sorting sequence
from collections import OrderedDict
foclens = [ 0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1.,   1.1,  1.2, ↵
    1.3,  1.4,  1.5]
m = ipywidgets.Dropdown(
    options = OrderedDict([(str(x), str(x)) for x in foclens]) ,
    value='0.4',
    description='Focal length:',
)



from IPython.display import display
```

```
display(xx)
display(y)
display(w)
display(m)
```

```
FloatSlider(value=7.5, description='Test:', max=10.0, min=5.0)
```

```
Checkbox(value=True, description='Check me')
```

```
Dropdown(description='Number:', index=1, options=('test 1', 'option 2',↵
    'selection 3'), value='option 2')
```

```
Dropdown(description='Focal length:', index=1, options=OrderedDict↵
   ([('0.3', '0.3'), ('0.4', '0.4'), ('0.5', '0...
```

```
# http://stackoverflow.com/questions/28529157/dynamically-changing-↵
   dropdowns-in-ipython-notebook-widgets-and-spyre
# from IPython.html import widgets
from IPython.display import display

geo={'USA':['CHI','NYC'],'Russia':['MOW','LED']}

def print_city(city):
    print(city)

def select_city(country):
    cityW.options = geo[country]


scW = ipywidgets.Select(options=geo.keys())
init = scW.value
cityW = ipywidgets.Select(options=geo[init])
j = ipywidgets.interactive(print_city, city=cityW)
i = ipywidgets.interactive(select_city, country=scW)
display(i)
display(j)
```

```
interactive(children=(Select(description='country', options=('USA', '↵
   Russia'), value='USA'), Output()), _dom_c...
```

```
interactive(children=(Select(description='city', options=('CHI', 'NYC')↵
   , value='CHI'), Output()), _dom_classes...
```

The following two cells illustrate how a slider is used in the `widgets.interactive` function to test the value of the slider and then do something with the value. The example below shows how to pass 'fixed' or non-widget parameters to the function. Any number of such widgets may be passed, but they must all be named.

For more examples see the links shown above. An example of interactive image segmentation is shown in notebook '10-ImageUtilities' in this series.

```
def doSomething(scale, thx):
    print('scale={} thx={} product={}'.format(scale, thx, scale * thx))
    return (scale, thx)
```

```
scale = 5.0
v = ipywidgets.interactive(doSomething, scale=ipywidgets.fixed(scale),
                        thx=ipywidgets.FloatSlider(value=128, min=0.0, ↵
                            max=255.0, step=1))
display(v)
```

```
interactive(children=(FloatSlider(value=128.0, description='thx', max↵
   =255.0, step=1.0), Output()), _dom_classe...
```

```
form = widgets.VBox()
first = widgets.Text(description="First Name:")
last = widgets.Text(description="Last Name:")

students = widgets.VBox(visible=True, children=[
    widgets.Checkbox(description="Student1:", value=False),
    widgets.Checkbox(description="Student2:", value=False),
    ])

student = widgets.Checkbox(description="Student:", value=False)

school_info = widgets.VBox(visible=False, children=[
    widgets.Text(description="School:"),
    widgets.IntText(description="Grade:", min=0, max=12)
    ])

pet = widgets.Text(description="Pet's Name:")

form.children = [first, last, student, students, school_info, pet]

display(form)

def on_student_toggle(name, value):
    if value:
        school_info.visible = True
    else:
        school_info.visible = False

student.observe (on_student_toggle, 'value')
students.children[0].observe(on_student_toggle, 'value')
students.children[1].observe(on_student_toggle, 'value')
```

```
VBox(children=(Text(value='', description='First Name:'), Text(value↵
   ='', description='Last Name:'), Checkbox(v...
```

```
form = widgets.VBox()
first = widgets.Text(description="First Name:")
last = widgets.Text(description="Last Name:")

student = widgets.Checkbox(description="Student:", value=False)

school_info = widgets.VBox(visible=False, children=[
    widgets.Text(description="School:"),
    widgets.IntText(description="Grade:", min=0, max=12)
    ])

pet = widgets.Text(description="Pet's Name:")
form.children = [first, last, student, school_info, pet]
display(form)

def on_student_toggle(name, value):
    if value:
        school_info.visible = True
    else:
        school_info.visible = False
student.observe(on_student_toggle, 'value')
```

```
VBox(children=(Text(value='', description='First Name:'), Text(value↵
   ='', description='Last Name:'), Checkbox(v...
```

```
from IPython.display import display

float_range = widgets.FloatSlider()
string = widgets.Text(value='hi')
container = widgets.Box(children=[float_range, string])

container.border_color = 'red'
container.border_style = 'dotted'
container.border_width = 3
display(container) # Displays the `container` and all of it's children.

def on_string_change(name, value):
    print(value)

# string.on_trait_change(on_string_change,'value')
string.on_submit(on_string_change,'value')
```

```
Box(children=(FloatSlider(value=0.0), Text(value='hi')))
```

The following is an example by Ketcheson, Ahmadia and Granger taken from

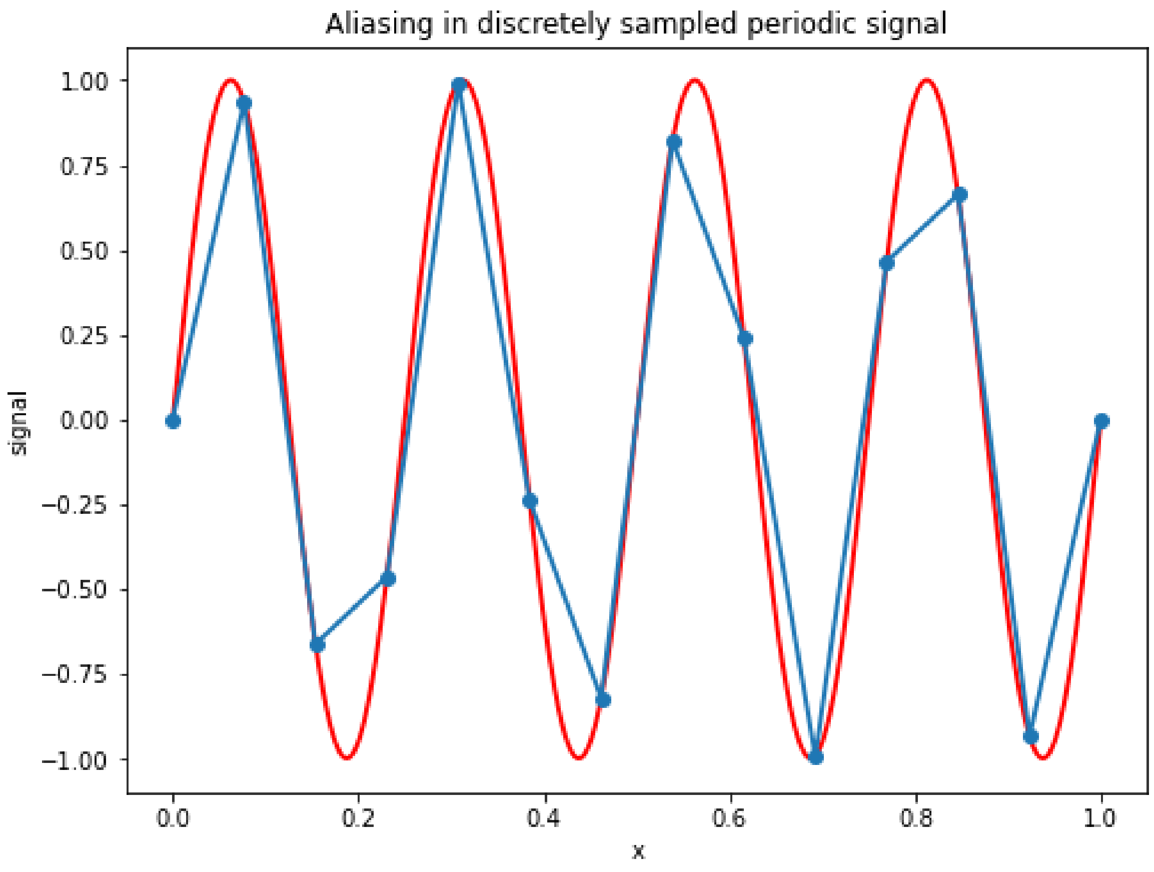
`http://www.nature.com/news/ipython-interactive-demo-7.21492?article=1.16261`[139]

It demonstrates aliasing during sampling of a signal. To see the effects of aliasing:

1. Run the next cell, then set the `grid\_points` slider to 13.
2. Move the `frequency` slider to values above 10.
3. As the frequency increases, the measured signal (blue) has a lower frequency than the real one (red).

```
# Import matplotlib (plotting) and numpy (numerical arrays).
# This enables their use in the Notebook.
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Import IPython's interact function which is used below to
# build the interactive widgets
from ipywidgets import interact#, interactive, fixed, interact_manual
# import ipywidgets as widgets


def plot_sine(frequency=4.0, grid_points=12, plot_original=True):
    """
    Plot discrete samples of a sine wave on the interval ``[0, 1]``.
    """
    x = np.linspace(0, 1, grid_points + 2)
    y = np.sin(2 * frequency * np.pi * x)
    xf = np.linspace(0, 1, 1000)
    yf = np.sin(2 * frequency * np.pi * xf)

    fig, ax = plt.subplots(figsize=(8, 6))
    ax.set_xlabel('x')
    ax.set_ylabel('signal')
    ax.set_title('Aliasing in discretely sampled periodic signal')

    if plot_original:
        ax.plot(xf, yf, color='red', linestyle='solid', linewidth=2)

    ax.plot(x,  y,  marker='o', linewidth=2)

# The interact function automatically builds a user interface for ↵
   exploring the
# plot_sine function.
interact(plot_sine, frequency=(1.0, 22.0, 0.5), grid_points=(10, 60, 1)↵
   , plot_original=True);
```

```
interactive(children=(FloatSlider(value=4.0, description='frequency', ↵
   max=22.0, min=1.0, step=0.5), IntSlider(...
```



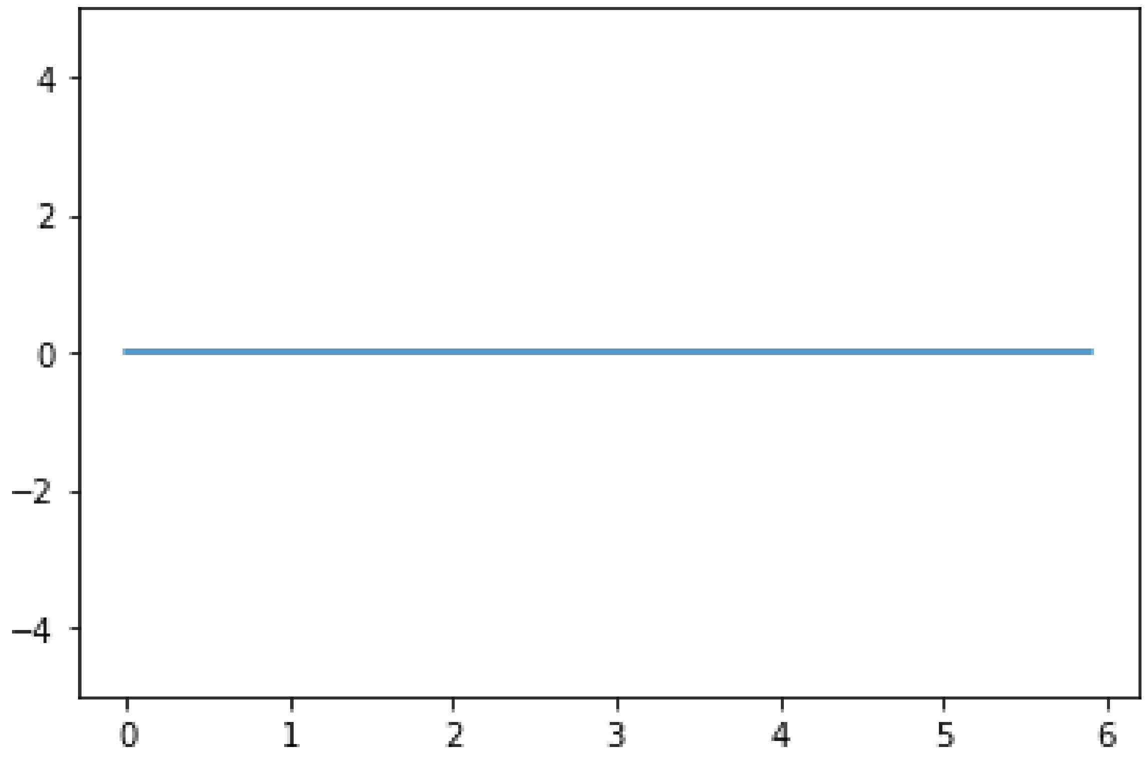
An example at `https://github.com/NelisW/ComputationalRadiometry/blob/master/10-ImageUtilities.ipynb`[140] shows how to use interactive widgets when segmenting an image.

https://stackoverflow.com/questions/47102564/how-to-set-interact-arguments-programmatically

```
n_weights = 10
weight_sliders = [widgets.FloatSlider(value=0,min=-2,max=2,step=0.1,↵
   description=f's{i}',
                                              disabled=False,continuous_update=↵
                                                 False,orientation='horizontal'↵
                                                 ,
                                             readout=True,readout_format='.2f')↵
                                                 for i in range(n_weights)]

def PlotSuper(**kwargs):
    def f(x):
        y=0
        for i,weight in enumerate(kwargs.values()):
            if i==0:
                y+=weight
            else:
                y+=weight*np.sin(x*i)
        return y
    vf = np.vectorize(f)
    xx= np.arange(0,6,0.1)
    plt.plot(xx,vf(xx))
    plt.gca().set_ylim(-5,5)

kwargs = {f's{i}':slider for i,slider in enumerate(weight_sliders)}
interact(PlotSuper,**kwargs)
```

```
interactive(children=(FloatSlider(value=0.0, continuous_update=False, ↵
   description='s0', max=2.0, min=-2.0), Fl...
```

```
<function __main__.PlotSuper(**kwargs)>
```



https://github.com/jupyter-widgets/ipywidgets/blob/master/docs/source/examples/Widget%20Styling.ipynb shows how to style the widgets using css.

```
from ipywidgets import Button, Layout

b = Button(description='(50% width, 80px height) button',
           layout=Layout(width='50%', height='80px'))
b
```

```
Button(description='(50% width, 80px height) button', layout=Layout(↵
    height='80px', width='50%'), style=ButtonS...
```

```
c = Button(description='Another button with the same layout', layout=b.↵
    layout)

c
```

```
Button(description='Another button with the same layout', layout=Layout↵
    (height='80px', width='50%'), style=But...
```

```
from ipywidgets import Button, HBox, VBox

words = ['correct', 'horse', 'battery', 'staple']
items = [Button(description=w) for w in words]
left_box = VBox([items[0], items[1]])
right_box = VBox([items[2], items[3]])
HBox([left_box, right_box])
```

```
HBox(children=(VBox(children=(Button(description='correct', style=↵
    ButtonStyle()), Button(description='horse', ...
```

```
from ipywidgets import IntSlider, Label
IntSlider(description=r'\(\int_0^t f\)')
```

```
IntSlider(value=0, description='\\(\\int_0^t f\\)')
```

```
from ipywidgets import Layout, Button, Box

items_layout = Layout( width='auto')     # override the default width ↵
    of the button to 'auto' to let the button grow

box_layout = Layout(display='flex',
                    flex_flow='column',
                    align_items='stretch',
                    border='solid',
                    width='50%')

words = ['correct', 'horse', 'battery', 'staple']
items = [Button(description=word, layout=items_layout, button_style='↵
    danger') for word in words]
box = Box(children=items, layout=box_layout)
box
```

```
Box(children=(Button(button_style='danger', description='correct', ↵
    layout=Layout(width='auto'), style=ButtonSt...
```

```
from ipywidgets import Layout, Button, Box, VBox

# Items flex proportionally to the weight and the left over space ↵
    around the text
```

```
items_auto = [
    Button(description='weight=1; auto', layout=Layout(flex='1 1 auto',↵
        width='auto'), button_style='danger'),
    Button(description='weight=3; auto', layout=Layout(flex='3 1 auto',↵
        width='auto'), button_style='danger'),
    Button(description='weight=1; auto', layout=Layout(flex='1 1 auto',↵
        width='auto'), button_style='danger'),
 ]

# Items flex proportionally to the weight
items_0 = [
    Button(description='weight=1; 0%', layout=Layout(flex='1 1 0%', ↵
       width='auto'), button_style='danger'),
    Button(description='weight=3; 0%', layout=Layout(flex='3 1 0%', ↵
       width='auto'), button_style='danger'),
    Button(description='weight=1; 0%', layout=Layout(flex='1 1 0%', ↵
       width='auto'), button_style='danger'),
 ]
box_layout = Layout(display='flex',
                    flex_flow='row',
                    align_items='stretch',
                    width='70%')
box_auto = Box(children=items_auto, layout=box_layout)
box_0 = Box(children=items_0, layout=box_layout)
VBox([box_auto, box_0])
```

```
VBox(children=(Box(children=(Button(button_style='danger', description↵
   ='weight=1; auto', layout=Layout(flex='1...
```

```
from ipywidgets import Layout, Button, Box, FloatText, Textarea, ↵
   Dropdown, Label, IntSlider

form_item_layout = Layout(
    display='flex',
    flex_flow='row',
    justify_content='space-between'
)

form_items = [
    Box([Label(value='Age of the captain'), IntSlider(min=40, max=60)],↵
        layout=form_item_layout),
    Box([Label(value='Egg style'),
         Dropdown(options=['Scrambled', 'Sunny side up', 'Over easy'])↵
            ], layout=form_item_layout),
    Box([Label(value='Ship size'),
         FloatText()], layout=form_item_layout),
    Box([Label(value='Information'),
         Textarea()], layout=form_item_layout)
]

form = Box(form_items, layout=Layout(
    display='flex',
    flex_flow='column',
    border='solid 2px',
    align_items='stretch',
    width='50%'
))
form
```

```
Box(children=(Box(children=(Label(value='Age of the captain'), ↵
   IntSlider(value=40, max=60, min=40)), layout=La...
```

```
from ipywidgets import Layout, Button, Box

item_layout = Layout(height='100px', min_width='40px')
items = [Button(layout=item_layout, description=str(i), button_style='↵
   warning') for i in range(40)]
box_layout = Layout(overflow_x='scroll',
                    border='3px solid black',
                    width='500px',
                    height='',
                    flex_flow='row',
                    display='flex')
carousel = Box(children=items, layout=box_layout)
VBox([Label('Scroll horizontally:'), carousel])
```

```
VBox(children=(Label(value='Scroll horizontally:'), Box(children=(↵
   Button(button_style='warning', description='...
```

```
def makeplot(title,display_trend,marker,amplitude,step_size,periods,↵
   noise_scale,offset,trend):
    pass

def interact_hookup(f, controls):
    from ipywidgets import Output
    out = Output()
    def observer(change):
        out.clear_output()
        kwargs = {k:v.value for k,v in controls.items()}
        with out:
            f(**kwargs)
    for k,w in controls.items():
        w.observe(observer, 'value')
    observer(None)
    return out

w = dict(
      title=widgets.Text(value='Hello World', placeholder='Type ↵
         something', description='Title:', disabled=False),
      display_trend=widgets.ToggleButton(value=False, description='↵
         Display Trend', icon='check'),
      marker=widgets.RadioButtons(options=['x', 'o', '.'], value='x', ↵
         description='Marker:'),
      amplitude=widgets.FloatSlider(value=1, min=-5, max=5, description↵
         ='Amplitude:'),
      step_size=widgets.FloatSlider(value=0.1, min=0.01, max=0.1, step↵
         =0.01, description='Step size:'),
      periods=widgets.FloatSlider(value=5, min=1, max=20, description='↵
         Periods:'),
      noise_scale=widgets.FloatSlider(value=0.1, min=0.01, max=2, ↵
         description='Noise:'),
      offset=widgets.FloatSlider(value=0, min=-5, max=5, description='↵
         Offset:'),
      trend=widgets.FloatSlider(value=1, min=-5, max=5, description='↵
         Trend:'),
    )

output = interact_hookup(makeplot, w)
```

```
UI = VBox([
    HBox([
        VBox([
            w['title'],
            w['display_trend'],
            w['marker'],
        ]),
        VBox([
            w['amplitude'],
            w['step_size'],
            w['periods'],
            w['noise_scale'],
            w['offset'],
            w['trend'],
        ])
    ]),
    output
])


display(UI)
```

```
VBox(children=(HBox(children=(VBox(children=(Text(value='Hello World', ↵
   description='Title:', placeholder='Type...
```

2.32.1 Interdependent widgets

The softmax function is used in neural networks. Suppose we have a network with four neurons, and four corresponding weighted inputs, which we'll denote z_1^L, z_2^L, z_3^L, and z_4^L.

According to this function, the activation a_j^L of the j output neuron is

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \tag{2.8}$$

where in the denominator we sum over all the inputs z_j^L.

As you increase any one component, its output will increase

Shown below are adjustable sliders showing possible values for the weighted inputs, and a graph of the corresponding output activations. A good place to start exploration is by using the bottom slider to increase z_4^L. As you increase z_4^L, you'll see an increase in the corresponding output activation, a_4^L, and a decrease in the other output activations. Similarly, if you decrease z_4^L then a_4^L will decrease, and all the other output activations will increase. In fact, if you look closely, you'll see that in both cases the total change in the other activations exactly compensates for the change in a_4^L. The reason is that the output activations are guaranteed to always sum up to 1.

In the code below there is a direct match between each slider and progress bar next to it. The sliders and progress bars are created in a dict comprehension, using the same keys for sliders and progress bars. These widgets have global scope and are available inside the softmax function.

The widgets are displayed manually (not automatically in interact) with the idea that these will be updated later.

`ipywidgets.interact` automatically displays the widgets when invoked, but we already displayed the widgets. Hence, the `interactive` function is called rather than `interact`, because `interactive` does not display/show the widgets.

```
import numpy as np
from ipywidgets import HBox,VBox,Button,FloatSlider,FloatProgress,↵
   interactive

# set up the widgets with precalculated values
# these sliders and prog bars are visible and are updated below in the ↵
   softmax function
sliders = {'1':[2.5,0.31], '2':[-1.,0.009], '3':[3.2,0.633], '4'↵
   :[0.5,0.043]}
sld = {key:FloatSlider(min=-5.0, max=+5.0, value=f'{sliders[key][0]}', ↵
   step=0.05,description=f'$z^L_{key}$') for key in sliders}
prb = {key:FloatProgress(value=f'{sliders[key][1]}',min=0,max=1.0,step↵
   =0.01,description=f'$a^L_{key}$',bar_style='info',orientation='↵
   horizontal') for key in sliders}

# build and display the widget grid in pairs of sliders and prog bars
lstD = [HBox([sld[key], prb[key]]) for key in sld]
display(VBox(lstD))

# function is invoked if any of the sliders change
# and the result is used to change the progress bar
def softmax(**lstZ):
    sum = 0
    for key in lstZ:
        sum += np.exp(lstZ[key])
    for key in lstZ:
        prb[key].value = np.exp(lstZ[key])/sum

#  `interactive` does not display/show the widgets, already done above.
w = interactive(softmax, **sld )
```

```
VBox(children=(HBox(children=(FloatSlider(value=2.5, description='$z^↵
   L_1$', max=5.0, min=-5.0, step=0.05), Flo...
```

3 The following information is somewhat esoteric, you need not go into this

3.0.1 Simple progress bar

Note that clear_output wipes the entire cell output, including previous output

https://mikulskibartosz.name/how-to-display-a-progress-bar-in-jupyter-notebook-47bd4c2944bf

```
def update_progress(progress, bar_length=20):

    from IPython.display import clear_output

    if isinstance(progress, int):
        progress = float(progress)
    if not isinstance(progress, float):
        progress = 0
    if progress < 0:
        progress = 0
    if progress >= 1:
        progress = 1
    block = int(round(bar_length * progress))
    clear_output(wait = True)
    text = "Progress: [{0}] {1:.1f}%".format( "#" * block + "-" * (↲
       bar_length - block), progress * 100)
    print(text)
```

Test:

```
import time

print('before')

#Replace this with a real computation
number_of_elements = 10
for i in range(number_of_elements):
    time.sleep(0.1)

    # progress must be a float between 0 and 1
    update_progress((i+1) / number_of_elements,bar_length=40)

print('after')
```

```
Progress: [########################################] 100.0%
after
```

```
import pyradi.ryutils as ryutils

import time

print('before')

#Replace this with a real computation
number_of_elements = 10
for i in range(number_of_elements):
    time.sleep(0.1)
    # progress must be a float between 0 and 1
    ryutils.update_progress((i+1) / number_of_elements,bar_length=40)

print('after')
```

```
Progress: [########################################] 100.0%
after
```

3.0.2 Notebook file format

From http://ipython.org/ipython-doc/stable/notebook/nbconvert.html#notebook-json-fi

Binary data such as figures are also saved directly in the JSON file. This provides convenient single-file portability, but means that the files can be large; a diff of binary data is also not very meaningful. Since the binary blobs are encoded in a single line, they affect only one line of the diff output, but they are typically very long lines. You can use the Cell | All Output | Clear menu option to remove all output from a notebook prior to committing it to version control, if this is a concern.

3.0.3 Reading json files in IPython

The following code reads this file and prints the first five cells.

```
import nbformat

nb = nbformat.read('01-IPythonHintsAndTips.ipynb', as_version=4)
```

```
nb.cells[0:5]
```

```
[{'cell_type': 'markdown',
  'metadata': {'run_control': {'frozen': False, 'read_only': False}},
  'source': 'This notebook forms part of a series on [computational ↵
     optical radiometry](https://github.com/NelisW/↵
     ComputationalRadiometry#computational-optical-radiometry-with-↵
     pyradi).  The notebooks can be downloaded from [Github](https://↵
     github.com/NelisW/ComputationalRadiometry#computational-optical-↵
     radiometry-with-pyradi). These notebooks are constantly revised ↵
     and updated, please revisit from time to time.  \n\nThis notebook ↵
     was written several Ipython/Jupyter generations ago, some ↵
     information may be no longer applicable.\n'},
 {'cell_type': 'markdown',
  'metadata': {'run_control': {'frozen': False, 'read_only': False}},
  'source': '# 1 Jupyter / IPython notebook hints and tips '},
 {'cell_type': 'markdown',
  'metadata': {'run_control': {'frozen': False, 'read_only': False}},
  'source': 'The date of this document and module versions used in this↵
      document are given at the end of the file.  \nFeedback is ↵
     appreciated: neliswillers at gmail dot com. '},
 {'cell_type': 'markdown',
  'metadata': {'run_control': {'frozen': False, 'read_only': False}},
  'source': '# Jupyter and IPython'},
 {'cell_type': 'markdown',
  'metadata': {'run_control': {'frozen': False, 'read_only': False}},
  'source': 'From <http://ipython.org/>:\n\n"IPython is a growing ↵
     project, with increasingly language-agnostic components. IPython ↵
     3.x will be the last monolithic release of IPython, containing the↵
      notebook server, qtconsole, etc. The language-agnostic parts of ↵
     the project: the notebook format, message protocol, qtconsole, ↵
     notebook web application, etc. will move to new projects under the↵
      name [Jupyter](https://jupyter.org/). IPython itself will return ↵
     to being focused on interactive Python, part of which will be ↵
     providing a Python kernel for Jupyter. IPython 3.0 contains some ↵
     indications of the project transition, including the logo in the ↵
     notebook web UI being that of Jupyter."\n\nIn this document, all ↵
     references to IPython refer to the IPython kernel, running on top ↵
     of Jupyter.\n\nIPython versions 2.x use the nbformat 3 file format↵
     .  \nIPython versions 3.x use the nbformat 4 file format.  \nTo ↵
     convert from nb4 to nb3 format (from with Jupyter IPython 3 ↵
     installed) type:  \n\n    ipython nbconvert --to notebook --↵
     nbformat 3 mynotebook.ipynb\n\n\n\n<http://ipython.org/ipython-doc↵
     /3/notebook/nbformat.html#nbformat>  \n<http://ipython.org/ipython↵
     -doc/3/whatsnew/version3.html>  \n'}]
```

3.0.4 Running notebook servers

This document describes how you can secure a notebook server and how to run it on a public interface:

`http://ipython.org/ipython-doc/rel-1.1.0/interactive/public\_server.html`[142]

3.0.5 Markdown formatting in dynamic output display

`http://catherinedevlin.blogspot.com/2013/06/easy-html-output-in-ipython-notebook.html`[143]

For this to work, you first have to install the Python markdown package (assuming you have the pip python package installer):

`pip install markdown`

Use the following function to render a Python string in markdown syntax to display in IPython:

```python
import markdown
class MD(str):
    def _repr_html_(self):
        return markdown.markdown(self)
```

```python
import math
a = 2
MD("""
Dynamic demonstration
--------------
This is a mixture of markdown **and** html:<br>
The square root of {0} <font color="green">used to be</font> somewhere ↵
   near {1}""".format(a,math.sqrt(a)))
```

3.1 Dynamic demonstration

This is a mixture of markdown **and** html:

The square root of 2 used to be somewhere near 1.4142135623730951

3.1.1 HTML formatting in dynamic output display

Use HTML to format the output of your code `http://python.6.x6.nabble.com/Printing-HTML-within` `html`[144]

```
from IPython.display import display, HTML
for x in range(3):
    display(HTML("<p><i>Length</i> <b>" + str(x) + "</b>"))
```

Length **0**

Length **1**

Length **2**

3.1.2 Displaying tables in HTML

`http://calebmadrigal.com/display-list-as-table-in-ipython-notebook/`[145]

```
class ListTable(list):
    """ Overridden list class which takes a 2-dimensional list of
        the form [[1,2,3],[4,5,6]], and renders an HTML Table in
        IPython Notebook. """

    def _repr_html_(self):
        html = ["<table>"]
        for row in self:
            html.append("<tr>")

            for col in row:
                html.append("<td>{0}</td>".format(col))

            html.append("</tr>")
        html.append("</table>")
        return ''.join(html)
```

```
import random
table = ListTable()
table.append(['x', 'y', 'x-y', '(x-y)**2'])
for i in range(7):
    x = random.uniform(0, 10)
    y = random.uniform(0, 10)
    table.append([x, y, x-y, (x-y)**2])

table
```

x	y	x-y	(x-y)**2
7.961474434798401	6.865343639904734	1.096130794893667	1.201502719514222
5.201977519140058	7.346595757886568	-2.14461823874651	4.599387389964183
4.696815945296278	3.42530669581821	1.2715092494780675	1.6167357715082786
6.2854048732677565	9.328224476424943	-3.042819603157186	9.258751137357656
1.337242255262987	6.186116388698062	-4.848874133435075	23.51158036189575
7.040166772972675	3.984930885436305	3.0552358875363703	9.334466328490153
3.941562299040963	1.4756150197404327	2.46594727930053	6.080895984289687

3.2 Fine-tuning IPython typographic output appearance

Changing the fonts, colours and layout to suit your own style.

http://slendrmeans.wordpress.com/2012/12/05/better-typography-for-ipython-notebook

http://zulko.wordpress.com/2013/04/14/customize-your-ipython-notebook-with-css/[147

3.3 Adding IPython display output to existing objects

http://nbviewer.ipython.org/url/github.com/ipython/ipython/raw/master/examples/notebooks/Custom\%20Display\%20Logic.ipynb[148]

For example, define a function that pretty-prints a polynomial as a LaTeX string:

```
def poly2latex(p):
    terms = ['%.2g' % p.coef[0]]
    if len(p) > 1:
        term = 'x'
        c = p.coef[1]
        if c!=1:
            term = ('%.2g ' % c) + term
        terms.append(term)
    if len(p) > 2:
        for i in range(2, len(p)):
            term = 'x^%d' % i
            c = p.coef[i]
            if c!=1:
                term = ('%.2g ' % c) + term
            terms.append(term)
    px = '$P(x)=%s$' % '+'.join(terms)
    dom = r', domain: $[%.2g,\ %.2g]$' % tuple(p.domain)
    return px+dom
```

```
import numpy as np

p = np.polynomial.Polynomial([1,2,3], [-10, 10])
from IPython.display import Latex
Latex(poly2latex(p))
```

$P(x)=1+2x+3x^2$, domain: $[-10,\ 10]$

But you can instruct IPython to use default display as follows:

```
ip = get_ipython()
latex_formatter = ip.display_formatter.formatters['text/latex']
latex_formatter.for_type_by_name('numpy.polynomial.polynomial',
                                'Polynomial', poly2latex)
```

```
p2 = np.polynomial.Polynomial([-20, 71, -15, 1])
p2
```

$P(x) = -20 + 71x + -15x^2 + x^3$, domain: $[-1,\ 1]$

3.4 Making slides from IPython notebooks

IPython is the tool of choice for presentations at Python conferences today - you hardly see a slideshow that was not made with IPython.

`http://www.slideviper.oquanta.info/tutorial/slideshow\_tutorial\_slides.html#/`[149]

`http://www.damian.oquanta.info/posts/make-your-slides-with-ipython.html`[150]

`http://nbviewer.ipython.org/github/fperez/nb-slideshow-template/blob/master/install-support.ipynb`[151]

`https://hannes-brt.github.io/blog/2013/08/11/ipython-slideshows-will-change-the-wa`

Now one of the coolest new features are the Reveal.js based slideshows. Here[153] is an example by the developer of the slideshow feature Damián Avila which shows how to turn any IPython Notebook into a slideshow and how to include math, images, videos, tables, etc.

`http://www.slideviper.oquanta.info/tutorial/slideshow\_tutorial\_slides.html?transition=none#/`[154]

`http://hannes-brt.github.io/blog/2013/08/11/ipython-slideshows-will-change-the-way`
- hiding code in slide shows.

`http://nbviewer.ipython.org/urls/gist.github.com/damianavila/5970218/raw/766d41eab`
`using\_local\_reveal.ipynb`[156] - local copy of reveal.js

`https://www.youtube.com/watch?v=rBS6hmiK-H8`[157] - youtube video

First (1) create the IPython notebook as a regular notebook, then (2) change each cell's metadata to set its slideshow status, then (3) save the notebook, (4) convert the notebook to the slideshow format, and (5) serve the slideshow html file on an http server. Some of these steps are described next.

Step (2): Click on the the "Cell Toolbar" dropdown combobox: select the "Slideshow" option. On the top-right side of *each* cell will appear a dropdown combobox where you can define the slideshow status of that specific cell. Select the appropriate type for each cell.

Steps (4) and (5): The slide show runs in a reveal.js javascript environment, but requires that the file be served on an http server. In order to convert the notebook to slide show and serve in a browser, type the following command:

```
ipython nbconvert --to slides --post serve filename
```

where `filename` is the name of the ipython notebook you want to convert to a slide show.

3.5 Blogging with IPython

IPython is also used to created blogging pages:

`http://blog.fperez.org/2012/09/blogging-with-ipython-notebook.html`[158]

`http://www.damian.oquanta.info/`[159]

`http://brunettoziosi.eu/posts/blogging-with-nikola-ipython-github.html`[160]

`http://www.davidketcheson.info/2012/10/11/blogging\_ipython\_notebooks\_with\_jekyll.html`[161]

3.6 Customising the IPython notebook

The notebook is primarily rendered in HTML in the browser and when exported to HTML. As an HTML product it can be customised in terms of layout, font, colours and other elements of style. Likewise the exports to other formats, such as LaTeX, can also be similarly customised to a particular look and feel.

`http://slendermeans.org/better-typography-for-ipython-notebooks.html`[162]

`http://zulko.wordpress.com/2013/04/14/customize-your-ipython-notebook-with-css/`[163

`http://nbviewer.ipython.org/github/Carreau/posts/blob/master/Blog1.ipynb`[164]

3.7 Publishing your notebooks

If a notebook file (.ipynb) is available somewhere on the web, you can paste the URL into a text box on this website http://nbviewer.ipython.org/ and it will render the notebook for you, returning a URL to the HTML rendered file. This new URL can be embedded as a hyperlink in an HTML file - when the user clicks on the link, the browser will display the rendered notebook. This is how the notebooks referred to in the next section are rendered.

`http://developer.rackspace.com/blog/bookstore-for-ipython-notebooks.html`[165]

3.7.1 More HTML formatting

`http://stackoverflow.com/questions/13770394/ipython-notebook-make-output-cells-lik`

```
htmlContent = ''

def header(text):
    raw_html = '<h1>' + str(text) + '</h1>'
    return raw_html

def box(text):
    raw_html = '<div style="border:1px dotted black;padding:2em;">'+str↵
       (text)+'</div>'
    return raw_html

def addContent(raw_html):
    global htmlContent
    htmlContent += raw_html


# Example
addContent( header("This is an autogenerated header") )
addContent( box("This is some text in a box") )

from IPython.core.display import HTML
HTML(htmlContent)
```

4 This is an autogenerated header

4.1 Class Descriptors

https://twitter.com/jakevdp/status/1121873857973870592/photo/1

1. The instance method (aka plain function) remains unbound when retrieved from the class so `C.method.\_\_get\_\_` just returns itself. By contrast, the class method gets bound to the class in that case, so it returns a new bound method object. In the first case both calls point to the same thing. In the second case since it is a classmethod, every call call creates a new instance of the method. I think this kind of explains the whole idea behind classmethods.
2. In the first cell, `C.method` evaluates to `C.\_\_dict\_\_["method"]` ; in second cell it evaluates to `C.\_\_dict\_\_["method"].\_\_get\_\_(C, None)` which turns new objects each time
3. In the first case, `C.method` points to the `method()` function. In the second case, because the `method()` function is decorated with a `@classmethod` decorator, `C.method` returns an instance of bound method object. Every time you write `C.method` a new object is created.

To add to your confusion, `id(Example.clsmethod) == id(Example.clsmethod)`, because method objects use a free list. After the first id() call, the object gets deallocated and the second call can reuse the same object.

```python
class C:
    def method(self):
        pass

C.method is C.method
```

```
True
```

```python
class C:
    @classmethod
    def method(cls):
        pass

print(C.method is C.method)

print(id(C.method)==id(C.method))

a = C.method
b = C.method
print(id(a)==id(b))
```

```
False
True
False
```

```python
C.__dict__
```

```
mappingproxy({'__module__': '__main__',
              'method': <classmethod at 0x17ad848dac0>,
              '__dict__': <attribute '__dict__' of 'C' objects>,
              '__weakref__': <attribute '__weakref__' of 'C' objects>,
              '__doc__': None})
```

```
print(type(C.method))
print(type(C.__dict__['method']))
```

```
<class 'method'>
<class 'classmethod'>
```

4.2 Class and Instance Attributes

https://twitter.com/jakevdp/status/1120898594519650304?s=09

Tricky Python bug I just hit due to an incorrect mental model of class & instance attributes Solution is to use `self.\_\_class\_\_.\_num\_instances` = 1+.

```
class Foo:
    _num_instances = 0
    def __init__(self):
        self._num_instances += 1
#          self.__class__._num_instances += 1

f1 = Foo()
f2 = Foo()
print(Foo._num_instances)
```

```
0
```

4.3 Python and module versions, and dates

```
try:
    import pyradi.ryutils as ryutils
    print(ryutils.VersionInformation('matplotlib,numpy,pyradi,scipy,↵
       pandas'))
except:
    print("pyradi.ryutils not found")
```

```
Software versions
Python:   3.8.3 64bit [MSC v.1916 64 bit (AMD64)]
IPython:   7.26.0
OS:   Windows 10 10.0.19041 SP0
matplotlib:   3.4.3
numpy:   1.20.3
pyradi:   1.1.4
scipy:   1.7.1
pandas:   1.3.2
Mon Aug 23 20:22:38 2021 South Africa Standard Time
```

BIBLIOGRAPHY

[1] [Online]. Available: https://github.com/NelisW/ComputationalRadiometry#computational-optical-radiometry-with-pyradi

[2] [Online]. Available: https://github.com/NelisW/ComputationalRadiometry#computational-optical-radiometry-with-pyradi

[3] [Online]. Available: https://jupyter.org/

[4] [Online]. Available: http://ipython.org/ipython-doc/3/notebook/nbformat.html#nbformat

[5] [Online]. Available: http://ipython.org/ipython-doc/3/whatsnew/version3.html

[6] [Online]. Available: https://github.com/barbagroup/jupyter-tutorial

[7] [Online]. Available: http://nbviewer.jupyter.org/github/barbagroup/jupyter-tutorial/blob/master/3--Jupyter%20like%20a%20pro.ipynb

[8] [Online]. Available: http://nbviewer.ipython.org/urls/raw.github.com/ellisonbg/talk-strata2013/master/StrataIPythonSlides.ipynb

[9] [Online]. Available: http://nbviewer.ipython.org/

[10] [Online]. Available: https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks

[11] [Online]. Available: http://docs.continuum.io/anaconda/install.html

[12] [Online]. Available: http://ipython.org/ipython-doc/dev/interactive/notebook.html

[13] [Online]. Available: http://nbviewer.ipython.org/urls/raw.github.com/Unidata/tds-python-workshop/master/ipython-notebook.ipynb

[14] [Online]. Available: http://blog.safaribooksonline.com/2013/12/12/start-ipython-notebook/

[15] [Online]. Available: http://ipython.org/ipython-doc/stable/notebook/index.html

[16] [Online]. Available: http://www.astro.washington.edu/users/vanderplas/Astr599/notebooks/03_IPython_intro

[17] [Online]. Available: https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook#gs.KMCTS4w

[18] [Online]. Available: https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/

[19] [Online]. Available: http://www.cs.princeton.edu/courses/archive/spr05/cos126/cmd-prompt.html

[20] [Online]. Available: http://www.bleepingcomputer.com/tutorials/windows-command-prompt-introduction/

[21] [Online]. Available: http://stackoverflow.com/questions/1077814/assigning-a-shortcut-to-open-cmd-here

[22] [Online]. Available: https://stackoverflow.com/questions/60904/how-can-i-open-a-cmd-window-in-a-specific-location

[23] [Online]. Available: https://raw.githubusercontent.com/NelisW/ComputationalRadiometry/master/cmd-here/cmd-window-here.reg

[24] [Online]. Available: http://www.json.org/

[25] [Online]. Available: http://ipython.org/ipython-doc/dev/interactive/notebook.html#starting-the-notebook-server

[26] [Online]. Available: #NotebooksRemember

[27] [Online]. Available: http://ipython.org/ipython-doc/rel-1.0.0/interactive/nbconvert.html

[28] [Online]. Available: http://blog.fperez.org/2012/09/blogging-with-ipython-notebook.html

[29] [Online]. Available: http://nbviewer.ipython.org/url/www.damian.oquanta.info/posts/blogging-with-nikola-and-ipython.ipynb

[30] [Online]. Available: http://www.slideviper.oquanta.info/tutorial/slideshow_tutorial_slides.html?transition=none#/

[31] [Online]. Available: http://www.damian.oquanta.info/

[32] [Online]. Available: http://nbviewer.ipython.org/github/Carreau/posts/blob/master/06-NBconvert-Doc-Draft.ipynb

[33] [Online]. Available: https://github.com/NelisW/ipynb2tex

[34] [Online]. Available: https://github.com/NelisW/ipynb2tex/blob/master/test2LaTeX-wp.pdf?raw=true

[35] [Online]. Available: http://blog.juliusschulz.de/blog/ultimate-ipython-notebook

[36] [Online]. Available: http://www.inkscape.org/en/

[37] [Online]. Available: http://ipython.org/ipython-doc/rel-1.0.0/interactive/nbconvert.html

[38] [Online]. Available: https://stackoverflow.com/questions/19659864/ipython-nbconvert-and-latex-use-eps-instead-of-png-for-plots

[39] [Online]. Available: https://stackoverflow.com/questions/19600234/nbconvert-pdf-latex-page-formatting-ipython

[40] [Online]. Available: https://stackoverflow.com/questions/19524554/suppress-code-in-nbconvert-ipython

[41] [Online]. Available: http://nbviewer.ipython.org/

[42] [Online]. Available: http://nbviewer.ipython.org/

[43] [Online]. Available: http://nbviewer.ipython.org/

[44] [Online]. Available: https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/

[45] [Online]. Available: http://nbviewer.ipython.org/github/ipython/ipython/blob/master/examples/notebooks/Cell%20Magics.ipynb

[46] [Online]. Available: http://ipython.org/ipython-doc/dev/interactive/tutorial.html#magic-functions

[47] [Online]. Available: http://damontallen.github.io/IPython-quick-ref-sheets/

[48] [Online]. Available: http://ipython.org/

[49] [Online]. Available: http://andrew.gibiansky.com/blog/ipython/ipython-kernels/

[50] [Online]. Available: http://ipython.org/ipython-doc/stable/development/messaging.html

[51] [Online]. Available: https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/

[52] [Online]. Available: https://ipython.org/ipython-doc/3/interactive/magics.html#magic-reset_selective

[53] [Online]. Available: https://ipython.org/ipython-doc/dev/interactive/magics.html#magic-edit

[54] [Online]. Available: http://ipython.org/ipython-doc/1/config/editors.html

[55] [Online]. Available: http://stackoverflow.com/questions/15681153/external-editor-for-ipython-notebook

[56] [Online]. Available: http://stackoverflow.com/questions/3438531/ipython-workflow-edit-run

[57] [Online]. Available: http://stackoverflow.com/questions/28309430/edit-ipython-cell-in-an-external-editor

[58] [Online]. Available: http://pynash.org/2013/03/06/timing-and-profiling.html

[59] [Online]. Available: http://nbviewer.ipython.org/github/ipython/ipython/blob/1.x/examples/notebooks/Part%205%20-%20Rich%20Display%20System.ipynb#LaTeX

[60] [Online]. Available: http://ipython.org/ipython-doc/stable/api/generated/IPython.core.magics.pylab.html

[61] [Online]. Available: https://stackoverflow.com/questions/2386714/why-is-import-bad

[62] [Online]. Available: http://carreau.github.io/posts/10-No-PyLab-Thanks.ipynb.html

[63] [Online]. Available: https://stackoverflow.com/questions/17582137/ipython-notebook-svg-figures-by-default

[64] [Online]. Available: https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/

[65] [Online]. Available: http://arogozhnikov.github.io/2015/11/29/using-fortran-from-python.html

[66] [Online]. Available: https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/

[67] [Online]. Available: http://ipython.org/ipython-doc/dev/config/extensions/

[68] [Online]. Available: http://jupyter.cs.brynmawr.edu/hub/dblank/public/Jupyter%20Help.ipynb

[69] [Online]. Available: http://ipython.org/ipython-doc/dev/config/extensions/

[70] [Online]. Available: http://nbviewer.ipython.org/github/jrjohansson/ipython-asymptote/blob/master/Asymptote-examples.ipynb

[71] [Online]. Available: http://asymptote.sourceforge.net/

[72] [Online]. Available: http://www2.ipp.mpg.de/~mkraus/python/tikzmagic.py

[73] [Online]. Available: http://www2.ipp.mpg.de/~mkraus/python/tikzmagic_test.ipynb

[74] [Online]. Available: http://www.texample.net/tikz/examples/

[75] [Online]. Available: http://calicoproject.org/Icalico

[76] [Online]. Available: http://stackoverflow.com/questions/32046241/how-to-add-automatically-extension-to-jupiter-ipython-notebook

[77] [Online]. Available: https://github.com/ipython/ipython/issues/3216

[78] [Online]. Available: http://marcoagpinto.cidadevirtual.pt/proofingtoolgui.html

[79] [Online]. Available: https://github.com/marcoagpinto/aoo-mozilla-en-dict/tree/master/en_GB%20%28Marco%20Pinto%29

[80] [Online]. Available: http://marcoagpinto.cidadevirtual.pt/proofingtoolgui.html

[81] [Online]. Available: https://robotwhale.wordpress.com/2014/08/17/optimization-with-ipython/

[82] [Online]. Available: http://ipython.org/ipython-doc/rel-1.1.0/config/extensions/autoreload.html

[83] [Online]. Available: https://stackoverflow.com/editing-help

[84] [Online]. Available: http://nbviewer.ipython.org/github/ipython/ipython/blob/master/examples/notebooks/Part%204%20-%20Markdown%20Cells.ipynb

[85] [Online]. Available: http://johnmacfarlane.net/pandoc/demo/example9/pandocs-markdown.html

[86] [Online]. Available: http://johnmacfarlane.net/pandoc/README.html

[87] [Online]. Available: http://daringfireball.net/projects/markdown/

[88] [Online]. Available: http://daringfireball.net/projects/markdown/basics

[89] [Online]. Available: http://daringfireball.net/projects/markdown/syntax

[90] [Online]. Available: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

[91] [Online]. Available: http://support.iawriter.com/help/kb/general-questions/markdown-syntax-reference-guide

[92] [Online]. Available: https://code.google.com/p/pyradi/

[93] [Online]. Available: https://stackoverflow.com/questions/16630969/ipython-notebook-anchor-link-to-refer-a-cell-directly-from-outside

[94] [Online]. Available: http://nbviewer.ipython.org/github/ipython/nbconvert-examples/blob/master/citations/Tutorial.ipynb

[95] [Online]. Available: #LaTeXdocswithtemplatecontrol

[96] [Online]. Available: #Overview

[97] [Online]. Available: #LearningPythonandhintsandtips

[98] [Online]. Available: #IntroducingPythonforscientificwork

[99] [Online]. Available: http://nbviewer.ipython.org/urls/raw.github.com/ipython/ipython/1.x/examples/notebooks/Part%205%20-%20Rich%20Display%20System.ipynb

[100] [Online]. Available: http://www.pycon.se/2014/assets/slides/Plotly-Pycon-Sweden.pdf

[101] [Online]. Available: https://pyscience.wordpress.com/2014/09/01/interactive-plotting-in-ipython-notebook-part-12-bokeh/

[102] [Online]. Available: https://pyscience.wordpress.com/2014/09/02/interactive-plotting-in-ipython-notebook-part-22-plotly-2/

[103] [Online]. Available: http://pbpython.com/visualization-tools-1.html

[104] [Online]. Available: http://nipunbatra.github.io/2014/04/comparing-python-web-visualization/

[105] [Online]. Available: http://matplotlib.org/

[106] [Online]. Available: http://matplotlib.org/gallery.html

[107] [Online]. Available: http://matplotlib.org/examples/index.html

[108] [Online]. Available: http://matplotlib.org/contents.html#

[109] [Online]. Available: http://bokeh.pydata.org/en/latest/

[110] [Online]. Available: http://bokeh.pydata.org/en/latest/docs/gallery.html

[111] [Online]. Available: http://bokeh.pydata.org/en/latest/docs/quickstart.html

[112] [Online]. Available: http://nbviewer.ipython.org/github/bokeh/bokeh-notebooks/blob/master/index.ipynb#Tutorial

[113] [Online]. Available: http://bokeh.pydata.org/en/latest/docs/user_guide.html

[114] [Online]. Available: https://plot.ly/javascript/

[115] [Online]. Available: https://plot.ly/javascript/open-source-announcement/

[116] [Online]. Available: https://plot.ly/python/getting-started/

[117] [Online]. Available: https://plot.ly/python/offline/

[118] [Online]. Available: https://plot.ly/python/user-guide/

[119] [Online]. Available: http://nbviewer.ipython.org/github/plotly/python-user-guide/blob/master/Index.ipynb

[120] [Online]. Available: https://plot.ly/~chriddyp/1780.embed

[121] [Online]. Available: http://nbviewer.ipython.org/urls/raw.github.com/jakevdp/matplotlib_pydata2013/master/notebooks/05_Animations.ipynb

[122] [Online]. Available: http://nbviewer.ipython.org/urls/raw.github.com/jakevdp/matplotlib_pydata2013/master/notebooks/03_Widgets.ipynb

[123] [Online]. Available: http://nbviewer.ipython.org/github/dpsanders/matplotlib-examples/blob/master/colorline.ipynb

[124] [Online]. Available: http://stackoverflow.com/questions/14261903/how-can-i-open-the-interactive-matplotlib-window-in-ipython-notebook/14277370#14277370

[125] [Online]. Available: https://github.com/mwaskom/seaborn

[126] [Online]. Available: http://nbviewer.ipython.org/github/mwaskom/seaborn/blob/master/examples/plotting_distributions.ipynb

[127] [Online]. Available: http://www.slideviper.oquanta.info/nbcreveal/sky_test.html?theme=sky#/7

[128] [Online]. Available: http://nbviewer.ipython.org/github/ipython/ipython/blob/master/examples/Interactive%20Widgets/Index.ipynb

[129] [Online]. Available: https://github.com/ipython/ipython/tree/master/examples/Interactive%20Widgets

[130] [Online]. Available: https://www.youtube.com/watch?v=VaV10VNZCLA

[131] [Online]. Available: https://www.youtube.com/watch?v=vE_CJTen15M

[132] [Online]. Available: https://www.youtube.com/watch?v=o7Tb7YhJZR0

[133] [Online]. Available: https://www.youtube.com/watch?v=wxVx54ax47s

[134] [Online]. Available: https://github.com/ipython/ipython/wiki/Widgets

[135] [Online]. Available: https://github.com/ipython/ipython-in-depth/tree/master/examples/Interactive%20Widgets

[136] [Online]. Available: http://ipython.org/ipython-doc/3/whatsnew/version3_widget_migration.html

[137] [Online]. Available: https://keminglabs.com/print-the-docs-pdfs/1518024.pdf

[138] [Online]. Available: https://github.com/ipython/ipywidgets/blob/master/examples/notebooks/Widget%20Basics.ipynb

[139] [Online]. Available: http://www.nature.com/news/ipython-interactive-demo-7.21492?article=1.16261

[140] [Online]. Available: https://github.com/NelisW/ComputationalRadiometry/blob/master/10-ImageUtilities.ipynb

[141] [Online]. Available: http://ipython.org/ipython-doc/stable/notebook/nbconvert.html#notebook-json-file-format

[142] [Online]. Available: http://ipython.org/ipython-doc/rel-1.1.0/interactive/public_server.html

[143] [Online]. Available: http://catherinedevlin.blogspot.com/2013/06/easy-html-output-in-ipython-notebook.html

[144] [Online]. Available: http://python.6.x6.nabble.com/Printing-HTML-within-IPython-Notebook-IPython-specific-prettyprint-td5016624.html

[145] [Online]. Available: http://calebmadrigal.com/display-list-as-table-in-ipython-notebook/

[146] [Online]. Available: http://slendrmeans.wordpress.com/2012/12/05/better-typography-for-ipython-notebooks/

[147] [Online]. Available: http://zulko.wordpress.com/2013/04/14/customize-your-ipython-notebook-with-css/

[148] [Online]. Available: http://nbviewer.ipython.org/url/github.com/ipython/ipython/raw/master/examples/notebooks/Custom%20Display%20Logic.ipynb

[149] [Online]. Available: http://www.slideviper.oquanta.info/tutorial/slideshow_tutorial_slides.html#/

[150] [Online]. Available: http://www.damian.oquanta.info/posts/make-your-slides-with-ipython.html

[151] [Online]. Available: http://nbviewer.ipython.org/github/fperez/nb-slideshow-template/blob/master/install-support.ipynb

[152] [Online]. Available: https://hannes-brt.github.io/blog/2013/08/11/ipython-slideshows-will-change-the-way-you-work/

[153] [Online]. Available: http://www.slideviper.oquanta.info/tutorial/slideshow_tutorial_slides.html?transition=none#/

[154] [Online]. Available: http://www.slideviper.oquanta.info/tutorial/slideshow_tutorial_slides.html?transition=none#/

[155] [Online]. Available: http://hannes-brt.github.io/blog/2013/08/11/ipython-slideshows-will-change-the-way-you-work/

[156] [Online]. Available: http://nbviewer.ipython.org/urls/gist.github.com/damianavila/5970218/raw/766d41eab9a16850a2a4447f14e93e7ed88f6b08/using_local_reveal.ipynb

[157] [Online]. Available: https://www.youtube.com/watch?v=rBS6hmiK-H8

[158] [Online]. Available: http://blog.fperez.org/2012/09/blogging-with-ipython-notebook.html

[159] [Online]. Available: http://www.damian.oquanta.info/

[160] [Online]. Available: http://brunettoziosi.eu/posts/blogging-with-nikola-ipython-github.html

[161] [Online]. Available: http://www.davidketcheson.info/2012/10/11/blogging_ipython_notebooks_with_jekyll.html

[162] [Online]. Available: http://slendermeans.org/better-typography-for-ipython-notebooks.html

[163] [Online]. Available: http://zulko.wordpress.com/2013/04/14/customize-your-ipython-notebook-with-css/

[164] [Online]. Available: http://nbviewer.ipython.org/github/Carreau/posts/blob/master/Blog1.ipynb

[165] [Online]. Available: http://developer.rackspace.com/blog/bookstore-for-ipython-notebooks.html

[166] [Online]. Available: http://stackoverflow.com/questions/13770394/ipython-notebook-make-output-cells-like-markdown