Jupman

A template manager for online books made with Jupyter notebooks and NBSphinx doc generator

People That Write a Lot

Jul 09, 2023

Copyright © 2023 by People That Write a Lot.

Jupman is available under the Creative Commons Attribution 4.0 International License, granting you the right to copy, redistribute, modify, and sell it, so long as you attribute the original to People That Write a Lot and identify any changes that you have made. Full terms of the license are available at:

http://creativecommons.org/licenses/by/4.0/

The complete book can be found online for free at:

https://jupman.softpython.org/en/latest/
## CONTENTS

- About .................................................................................................................. 1
- Preface .................................................................................................................... 2

1 Overview .................................................................................................................. 3
  1.1 Contents .............................................................................................................. 3
  1.2 Revisions ........................................................................................................... 3
  1.3 Credits ............................................................................................................... 4

2 Quickstart .................................................................................................................. 5
  2.1 Installation .......................................................................................................... 5
  2.2 Configure ........................................................................................................... 8
  2.3 Building the manual ............................................................................................ 8
  2.4 Publish ............................................................................................................... 9
  2.5 Further steps ...................................................................................................... 9

3 Editing worksheets .................................................................................................... 11
  3.1 Common files .................................................................................................... 11
  3.2 Running Jupyter ................................................................................................. 11
  3.3 Source code for chapters .................................................................................... 12
  3.4 Exercise types .................................................................................................... 12
  3.5 Solution tags ...................................................................................................... 14
  3.6 Directive tags ..................................................................................................... 16
  3.7 Hiding cells ........................................................................................................ 16
  3.8 Info boxes .......................................................................................................... 17
  3.9 Utilities .............................................................................................................. 17
  3.10 Launch unit tests .............................................................................................. 18
  3.11 Python Tutor .................................................................................................... 18
  3.12 Pandas .............................................................................................................. 19
  3.13 Showing function help ..................................................................................... 19
  3.14 Custom js and css ............................................................................................ 20
  3.15 Show table of contents ..................................................................................... 20

4 Customize .................................................................................................................. 21
  4.1 Change website theme ....................................................................................... 21
  4.2 Fonts .................................................................................................................. 21
  4.3 font sizes ........................................................................................................... 22
  4.4 Warning about old versions .............................................................................. 22
  4.5 Fix nbphinx to create rst files .......................................................................... 22
  4.6 Git performance notes ...................................................................................... 22

5 Exams management .................................................................................................... 25
About

A template for online books made with Jupyter\(^1\) notebooks and NBSphinx\(^2\) doc generator.

Features

- inherits generation of static websites from NBSphinx\(^3\), with search, PDF, EPUB
- builds with ReadTheDocs, Github Actions\(^4\), or local Docker emulating ReadTheDocs
- exercises generation from solution templates (both .ipynb and .py)
- zips chapters
- code sharing among chapters (so students don’t need to install dependencies)
- Python Tutor integration (offline, no required dependencies)
- supports hundreds pages and deep nesting
- decent PDF layout
- basic exam management
- optional softpython theme\(^5\)
- documentation and tests\(^6\)
- Apache License v2.0, open source code on Github\(^7\)

Currently lacking

- generated PDF always displays solutions\(^8\)
- Python Tutor doesn’t work in JupyterLab\(^9\)
- more testing for exam management\(^10\)

Requirements

- Python 3.7+
- based upon sphinx-rtd-theme\(^11\) (adds only minimal improvements for better browsing)

Used by

- SoftPython book: english\(^12\) | italian\(^13\) (both 1000+ pdf pages from jupyter notebooks)
- Scientific Programming Lab @ University of Trento, Data Science Master\(^14\) (english, many python exercises and exams in mixed jupyter + python format)

\(^1\) http://jupyter.org
\(^2\) http://nbsphinx.readthedocs.io/
\(^3\) http://nbsphinx.readthedocs.io/
\(^4\) https://github.com/DavidLeoni/readthedocs-to-actions
\(^5\) https://jupman.softpython.org/themed/
\(^6\) https://github.com/DavidLeoni/jupman/tree/master/_test
\(^7\) https://github.com/DavidLeoni/jupman
\(^8\) https://github.com/DavidLeoni/jupman/issues/87
\(^9\) https://github.com/DavidLeoni/jupman/issues/42
\(^10\) https://github.com/DavidLeoni/jupman/issues?q=is%3Aopen+is%3Aissue+label%3Aexams
\(^12\) https://en.softpython.org/
\(^13\) https://it.softpython.org/
\(^14\) https://sciprog.davidleoni.it/
Preface

This book is the result of ... We thank this and that ...
CHAPTER
ONE

OVERVIEW

1.1 Contents

1. Quickstart
2. Editing worksheets
3. Customize
4. Exams management
5. Chapter examples
   1. Python example
   2. Jupyter example
   3. Mixed jupyter and python example
   4. Challenges: example worksheet solution
5. Templates
   1. Past exams
   2. Changelog
6. Tests
   1. Rendering tests
   2. Python Tutor tests

1.2 Revisions

• 19 September 2022: Released v3.5.6
• 4 June 2022: Released v3.5
• 30 April 2022: Released v3.4
• 22 February 2022: Released v3.3
• 16 October 2020: Released v3.2
• 16 January 2020: Released v3.1
• 29 December 2019: Released v3.0

• 24 September 2018: Released v2.0
• 3 August 2018: Released v0.8

1.3 Credits

• This site was made with Jupyter using NBSphinx extension\textsuperscript{16} and Jupman template\textsuperscript{17}.

\textsuperscript{16} \url{http://nbsphinx.readthedocs.io/}
\textsuperscript{17} \url{http://jupman.readthedocs.io/}
Jupman uses NbSphinx\(^\text{18}\) and either ReadTheDocs\(^\text{19}\) or Github Actions\(^\text{20}\)

### 2.1 Installation

(Instructions are for Ubuntu, on Windows may differ)

First, on Github, fork as a template jupman project\(^\text{21}\) to create yours, for example my-project.

Then, on your computer, clone the my-project from Github

You can choose to build either on:

- ReadTheDocs
- Github Actions
- locally with plain Sphinx
- locally with RTD Docker\(^\text{22}\)

(Note Jupman itself is building on both ReadTheDocs and Github Actions only for testing purposes, one is enough)

#### 2.1.1 Building with ReadTheDocs:

**IMPORTANT: choose a name which is NOT already on ReadTheDocs\(^\text{23}\)**

Create a ReadTheDocs account\(^\text{24}\) **using the same name as in Github** so the address in readthedocs will be something like my-project.readthedocs.org.

- Use ReadTheDocs panels to link the project to your Github repository.
- In Admin -> Advanced settings panel, set:
  - *Python interpreter* to *CPython 3.x*
  - *Requirements* to requirements-build.txt

---

\(^{18}\) [http://nbsphinx.readthedocs.io/](http://nbsphinx.readthedocs.io/)

\(^{19}\) [https://readthedocs.org](https://readthedocs.org)

\(^{20}\) [https://github.com/features/actions](https://github.com/features/actions)

\(^{21}\) [https://github.com/DavidLeoni/jupman](https://github.com/DavidLeoni/jupman)

\(^{22}\) [https://github.com/DavidLeoni/readthedocs-to-actions](https://github.com/DavidLeoni/readthedocs-to-actions)

\(^{23}\) [http://readthedocs.org](http://readthedocs.org)

\(^{24}\) [http://readthedocs.org](http://readthedocs.org)
2.1.2 Building with Github Actions:

Configure `.github/workflows/main.yml` on your computer to your needs - you will need to:

1. at the beginning in the `build_docs_job` section there is an `if` which makes the workflow only work in `DavidLeoni/jupman` repository, change it with your project repo and comment the following `needs:`
   `run_tests` line

2. set `RTD_PRJ_NAME`

3. If you want to publish to Github Pages: everything is set, just create an empty branch `gh-pages` in an new `HTML_FOLDER` before committing - from some other folder in your file system:

   ```bash
   git clone YOUR_REPO_ADDRESS HTML_FOLDER
   cd HTML_FOLDER
   git checkout --orphan gh-pages
   git rm -rf .
touch bla
   git add .
git commit -m "init"
git push origin gh-pages
   ```

2.1.3 Local build with Sphinx

Assuming you are on Linux/Mac:

1) Install Python 3.7+

2) Install Jupyter

3) Install required modules:

3.a) In a virtual environment (recommended) - from the root of the project, run:

   ```bash
   ./create-env.sh
   ```

   This will automatically install required modules in `_private/jupman_env` using `python3` system binary.

   If you want to use a particular python binary (note it must already be on your system), run i.e.:

   ```bash
   ./create-env.sh python3.7
   ```

   Afterwards, to activate the environment run:

   - In Windows:

     ```bash
     _private\jupman_env\bin\activate
     ```

   - in Linux/Mac:

     ```bash
     source activate
     ```

   to deactivate (from anywhere):

   ```bash
   deactivate
   ```

---

26 https://pages.github.com/
27 http://jupyter.org/install.html
3.b) without a virtual environment (**not recommended**): From the root of the project, run:

```bash
python3 -m pip install --user -r requirements-build.txt
```

Warning: to have reproducible builds `requirements-build.txt` pinpoints a lot of dependencies, installing without virtual environment may mess up other python programs!

### 2.1.4 Optional - Running tests

To check everything is working, you may want to run the tests.

1. Install test dependencies:

```bash
python3 -m pip install --user -r _test/requirements-test.txt
```

2. Run the tests:

```bash
python3 -m pytest _test/*_test.py
```

### 2.1.5 Optional - Install Jupyter contrib extensions

For a better editing experience like having Table of contents and other things, do the following:

1. install the **Jupyter contrib extensions** package:

   If you have Anaconda:

   ```bash
   conda install -c conda-forge jupyter_contrib_nbextensions
   ```

   If you don't have Anaconda:

   ```bash
   python3 -m pip install --user jupyter_contrib_nbextensions
   ```

2. Install in Jupyter:

   ```bash
   jupyter contrib nbextension install --user
   ```

3. Enable extension:

   For being able to view table of contents while editing notebooks, install toc2 extension:

   ```bash
   jupyter nbextension enable toc2/main
   ```

   For tocs to appear when in a document you will need to press a list button at the right-end of the toolbar.

   (since Jupman 0.8 custom injected tocs are disabled by default)

4. For a nice GUI to install extensions, install the **Jupyter Nbextensions configurator**:

   If you have Anaconda:

   From Anaconda Prompt:

   ```bash
   conda install -c conda-forge jupyter_nbextensions_configurator
   ```

   If you don't have Anaconda:

   [28] https://github.com/ipython-contrib/jupyter_contrib_nbextensions
   [29] https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator
python3 -m pip install --user jupyter_nbextensions_configurator

After installing, enable it:

```
jupyter nbextensions_configurator enable --user
```

and then start Jupyter, in file browser look for a Nbextensions tab

## 2.2 Configure

1. Edit `conf.py` as needed, which is the configuration file for Sphinx. In particular, you **MUST** edit the sections marked with TODO
2. Try launching build:

```
python3 build.py
```

For more info, see related section
3. If everything works fine on your computer, push changes back to Github
4. Go back to ReadTheDocs and try to run a build. Hopefully your project will become available on something like `my-project.readthedocs.org`
5. If you want to grade exams, see Exams management page.

You should now be ready to create your notebooks by launching from the project root:

```
jupyter notebook
```

1. If you wish your notebooks to appear in the generated manual, you have to add them in the `toc.rst` file.

   **NOTE:** the page `toc-page.rst`, which is set to be the `master_doc` of Sphinx, will just load the actual Table of Contents which is in `toc.rst`. It looks a bit convoluted because when it comes to indexes Sphinx is not much reliable, see this issue. We strongly advise not to change these settings!

2. edit the home, which is in the `index.ipynb` file

### 2.3 Building the manual

For a quick build that only produces html:

```
python3 build.py -q
```

Site will be created in `_build/` folder.

For help:

```
python3 build.py -h
```

---

To build everything (html + pdf + epub), go to the console and from the root directory run:

```
python3 build.py
```

NOTE: to generate PDFs you will need to install Latex environment

## 2.4 Publish

Just push to master and Github Actions / ReadTheDocs will do the rest, for example, for jupman is available at addresses:


IMPORTANT: ReadTheDocs WILL *NOT* execute Jupyter notebooks because of this bug

## 2.5 Further steps

See *Editing worksheets, Customize* and if needed *Exams management* pages

---

35 https://github.com/DavidLeoni/softpython/issues/2
EDITING WORKSHEETS

Here we give an overview of how to edit worksheets. More info can be found in Tests notebook.

3.1 Common files

There are a bunch of files common to all worksheets and possibly website

You do not need to change them (except maybe my_lib.py)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Jupyter</th>
<th>Website</th>
<th>Student zips</th>
</tr>
</thead>
<tbody>
<tr>
<td>_jupman_tools.py⁶⁶</td>
<td>back end stuff</td>
<td>X</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>_jupman.py⁶⁷</td>
<td>utilities for work-</td>
<td>X</td>
<td>.</td>
<td>X</td>
</tr>
<tr>
<td>_static/js/jupman.js⁶⁸</td>
<td>Javascript code</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_static/css/jupman.css⁶⁹</td>
<td>CSS</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_static/css/jupman-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>web.css¹¹</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2 Running Jupyter

First of all, run Jupyter from the root directory:

```
jupyter notebook
```

⁶⁶ https://github.com/DavidLeoni/jupman/blob/master/jupman_tools.py
⁶⁷ https://github.com/DavidLeoni/jupman/blob/master/jupman.py
⁶⁸ https://github.com/DavidLeoni/jupman/blob/master/my_lib.py
⁶⁹ https://github.com/DavidLeoni/jupman/blob/master/_static/js/jupman.js
¹¹ https://github.com/DavidLeoni/jupman/blob/master/_static/css/jupman.css
¹¹ https://github.com/DavidLeoni/jupman/blob/master/_static/css/jupman-web.css
3.3 Source code for chapters

Put chapters one per folder, in the root. Any folder which doesn't start with underscore _ or exam/ will be considered a chapter.

During build, each chapter gets automatically zipped and zip goes to _static/generated. So for example, python-example/ produces a zip called _static/generated/python-example.zip, which will have these contents:

```
python-example
  _static
    js
      jupman.js
toc.js
    css
      jupman.css
    img
      cc-by.png
python-example.ipynb
lab.py
lab_test.py
lab_sol.py
jupman.py
my_lib.py
```

The zip folder structure will be a merge of chapter files and files shared by all chapters which are specified in exercises_common_files variable in conf.py. Since the root in the zip becomes the chapter itself, jupman will process .py and .ipynb files for fixing eventual relative imports. Markdown, HTML and CSS links in ipynb will also be adjusted.

Exercise files can be automatically generated from solutions, as we will see next.

3.4 Exercise types

There can be three kinds of exercises: exercises in Python files, exercises in Jupyter files and mixed jupyter and Python exercises.

You can automatically generate an exercise from a solution file by stripping text marked with special tags. You can inspect generated files in _build/jupman/ directory

On the website, students will be able to see solutions by clicking on appropriate buttons.

In the zips to download, two versions of files will provided, one without solution and one with solutions (in exam modality of course no solution will be shipped)
3.4.1 Exercises in Python files

See python-example/python-example.ipynb

In this type of exercises, typically you have a Jupyter file (like python-example.ipynb) that describes the exercise and then the actual exercises are in Python files.

If there is a solution file FILE_sol.py ending in _sol.py but no corresponding exercise file FILE.py without the _sol:

then Jupman will try to generate FILE.py one from FILE_sol.py. To do so, it will look for tags to strip inside the solution file.

If there is already an exercise file like this:

- python_intro.py
- python_intro_sol.py

Jupman will just copy the existing file.

3.4.2 Exercises in Jupyter files

See example: jupyter-example/jupyter-example-sol.ipynb

This type of exercises stay in a Jupyter notebook itself.

If there is a notebook ending in -sol.ipynb, the following applies (WARNING: for ipynb files we use dash -, not the underscore _):

- the notebook must contain tags to strip
- exercises derived will have 'EXERCISES' appended to the title (the word can be customized in conf.py - you might need to translate it)

3.4.3 Mixed exercises in Jupyter and Python files

See jup-and-py-example/jup-and-py-example-sol.ipynb

3.4.4 Challenges

This is an experimental feature, current implementation is subject to change.

Challenges are solutions which remain unpublished and from which exercises are generated in the same original older where the solution resides (not only in the zip!). Challenge files can be both Jupyter notebooks or Python files, ending in -chal-sol.ipynb or _chal_sol.py.

The idea is that challenges solutions are gitignored, and exercises are manually generated by calling jupman.generate_exercise() inside a Jupyter notebook like this:

```python
#jupman-purge
import sys; sys.path.append('./'); from conf import jm;
jm.generate_exercise('great_chal_sol.py')
#/jupman-purge
```

3.4. Exercise types
It is a bit laborious but the idea is that typically you will also want to run and see tests results in Jupyter notebook so you can do it in the same final cell, which you will also probably want to set in cell metadata "nbsphinx": "hidden"

- the solution notebook must contain tags to strip and have SOLUTIONS at the end of the title (the word can be customized in conf.py - you might need to translate it)

### 3.5 Solution tags

Presence of these tags marks a cell as a solution.

Start tags begin with a # while end tags begin with a#

#### 3.5.1 jupman-raise

Replaces code inside with an Exception (text is customizable in conf.py). Be careful to position the comment exactly with the indentation you want the raise to appear. For example:

```python
def add(x,y):
    #jupman-raise
    return x + y
    #=>jupman-raise
```

becomes

```python
def add(x,y):
    raise Exception('TODO IMPLEMENT ME !')
```

#### 3.5.2 jupman-strip

Just strips code inside exercises

```python
def f(x):
    print(x)
    #jupman-strip

def help_func(x,y):
    return x - y
    #=>jupman-strip

def g(y):
    return y
```

becomes

```python
def f(x):
    print(x)

def g(y):
    return y
```
### 3.5.3 write here

This special tag for python code erases whatever is found afterwards the `# write here` line

- you can put how many spaces you want in the comment
- phrase can be customized in `conf.py`

```python
w = 5
# write here fast please
x = 5 + w
y = 2 + x
```

becomes

```python
w = 5
# write here fast please
```

### 3.5.4 SOLUTION

In a code cell, if you put `# SOLUTION` at the beginning the whole cell content gets deleted (`# SOLUTION` string included).

- Word can be customized in `conf.py`

```python
# SOLUTION
def f():
    print('hello')
```

becomes nothing:

```python
[ ]:
```

### 3.5.5 QUESTION - ANSWER

In a markdown cell, everything in a cell with `**ANSWER**` inside will be stripped.

- Markdown can be customized in `conf.py`

**QUESTION:** Describe why iPhone n+1 is better than iPhone n
**ANSWER:** it costs more

Becomes:

**QUESTION:** Describe why iPhone n+1 is better than iPhone n

[ ]:
3.6 Directive tags

Some tags change the preprocessor behaviour. They are applied before solution tags.

3.6.1 jupman-purge

Eliminate content both from exercises AND solutions. Can be helpful when you have code which creates expected output, like images or python data - the idea is to completely remove code so so students don’t accidentally copy-paste it or uncomment it.

- **jupman-purge-input**: purges only cell source
- **jupman-purge-output**: purges only cell output
- **jupman-purge-io**: purges both input and output

```
jupman-purge purges only a span:

x=5
#jupman-purge
plt.savefig('expected_image.png')
jupman.save_py('expected_output_db.py', ['big', 'data', 'structure']*1000)
#/jupman-purge
x=6
```

becomes

```
x=5
x=6
```

3.6.2 jupman-preprocess

By default only notebooks solutions (ending in -sol.ipynb) are preprocessed before html conversion begins. If you want to force preprocessing on a particular non-solution notebook, add this in the first cell:

```
#jupman-preprocess
```

3.7 Hiding cells

A way to hide cells (like for example the import jupman code) is by clicking View->Cell toolbar -> Edit metadata and adding "nbsphinx": "hidden" to the JSON (see also original [NB Sphinx docs] and [Toggleable cells in Jupman tests]).

**NOTE 1**: As of NB Sphinx 2.17, it is not possible to hide only cell text but not the output.

---

42 [https://nbsphinx.readthedocs.io/en/0.2.14/hidden-cells.html#Hidden-Cells](https://nbsphinx.readthedocs.io/en/0.2.14/hidden-cells.html#Hidden-Cells)
### 3.8 Info boxes

Supported boxes are inherited from NBSphinx with div classes "alert alert-info", "alert alert-warning"

See Rendering tests[^43] for examples.

Plus we add jupman-alert-principle: some alerts to be often reminded can be preceded with an empty div having class jupman-alert-principle followed by a regular alert box, so they will display as you want on the website and as fallback boxes in the pdf (did this way as we can't add classes nor other attributes, tried also data-jupman html attributes with no success)

**NOTE:** default colors are indicative and minimal on purpose, for a better view see sofpython themed version[^44]

**Recommended approach:** The typical principle alert should be brief and may have a link to more substantial text, with a short line under it. If you need more explicative text, put it outside:

```html
<div class="jupman-alert-principle"></div>
<div class="alert alert-info">
[IV PRINCIPLE](https://jupman.softpython.org/principles.html#IV-PRINCIPLE): **You shall write tests!**
Who does **not** writes tests, falls into _Debugging Hell_!
</div>
```

**IV PRINCIPLE[^45]: You shall write tests!**

Who does **not** writes tests, falls into *Debugging Hell*!

### 3.9 Utilities

**NOTE:** not mandatory, it's mostly intended to tweak notebooks downloaded locally. Should be avoided in notebooks meant for students, as it's more likely it will mess their configurations - also, they might copy the notebooks without knowing they contain the custom js and use weird extensions which could generate conflicts (such as double toc)

For notebooks in the root folder:

```python
import jupman
```

Worksheets in in subfolders can use `sys.path` to locate the module

```python
import sys
sys.path.append('../')
import jupman
```

Some reason for this ugliness is reported in [this issue][46].

[^43]: http://127.0.0.1:8888/notebooks/jupman/jupman-3.5.1/manual/tests.ipynb#Info/Warning-Boxes
[^45]: https://jupman.softpython.org/principles.html#IV-PRINCIPLE
[^46]: https://github.com/DavidLeoni/jupman/issues/12
3.10 Launch unit tests

Inside worksheets you can run unittest tests.
To run all the tests of a test class, write like this

```
jupman.run(NameOfTheTestClass)
```

To run a single method, write like this:

```
jupman.run(NameOfTheTestClass.nameOfTheMethod)
```

3.11 Python Tutor

Among the various ways to embed Python Tutor, we decided to implement a special `jupman.pytut()` method. First you need to import the jupman module:

```
import sys
sys.path.append('../')
import jupman
```

Then you can put a call to `jupman.pytut()` at the end of a cell, and the cell code will magically appear in python tutor in the output (except the call to `pytut()` of course). To see Python Tutor you don’t need to be online

```
x = [5, 8, 4]
y = {3: 9}
z = [x]
jupman.pytut()
```

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

Beware of variables which were initialized in previous cells, they won’t be available in Python Tutor and you will get an error:

```
w = 8
```

```
x = w + 5
jupman.pytut()
```

Traceback (most recent call last):
  File "./jupman.py", line 2453, in _runscript
    self.run(script_str, user_globals, user_globals)
  File "/usr/lib/python3.7/bdb.py", line 578, in run
    exec(cmd, globals, locals)
  File "<string>", line 2, in <module>
NameError: name 'w' is not defined
```

```
<IPython.core.display.HTML object>
```
Correctly rendering pandas in PDFs is not so easy (see issue[^47]), so far we created this little function which sometimes is handy:

```python
import pandas as pd

lista = [['Rosanna', 'Gippalanda', 26, 100, 500, 300, 600, 600, 100, 300, 600, 300, 200, 400, 200, 300, 400, 500],
         ['Matilda', 'Zampola', 10, 500, 200, 300, 500, 400, 300, 200, 500, 300, 200, 400, 200, 300, 400, 500],
         ['Mario', 'Cipolli', 25, 300, 500, 100, 500, 300, 500, 100, 500, 300, 200, 600, 300, 300, 300, 300, 100, 500],
         ['Ugo', 'Sgarapirri', 30, 100, 400, 200, 500, 300, 200, 600, 300, 300, 200, 400, 200, 300, 400, 500]
]

df = pd.DataFrame(lista, columns=['Name', 'Surname', 'Age', *[f'Par{i}'] for i in range(1,16)])
df # web
```

![Image for PDF](image.png)

```python
import jupman
jupman.draw_df(df) # image for pdf
```

3.13 Showing function help

Python help is already quite good, but adds two useless extra lines and only works as a print, so we defined `jupman.get_doc`:

```python
print(jupman.get_doc(jupman.get_doc))
```

```python
def get_doc(fun):
    """
    Returns the help of a function formatted in a faithful manner
    """
```

[^47]: https://github.com/DavidLeoni/jupman/issues/69
3.14 Custom js and css

If you need custom js and/or css in a notebook, you can inject it by running

```python
def jupman.init()
```

in the first cell, it will inject `jupman.js` and `jupman.css`

3.15 Show table of contents

Since 0.8, custom toc is disabled, try instead `installing toc2 extension`. If you still want the jupman toc (not recommended), execute

```python
def jupman.init(toc=True)
```

it will create the sidebar even when editing in Jupyter. To refresh the sidebar, just rerun the cell.

Note: hiding the `jupman.init` code cell will prevent the build system to embed the Javascript file `jupman.js` inside the page in the HTML website: this is still fine as it is fetched separately by settings in `conf.py`. 
4.1 Change website theme

If you want to change site colors and other changes, copy/edit `_static/css/jupman-web.css` and set it in `conf.html_css_files`:

```python
html_css_files = [
    'css/jupman.css',  # shared among jupyter and website
    'css/jupman-web.css',  # only on website
    #'css/softpython-theme.css',  # uncomment to activate
    #'css/scifi-theme.css',
]
```

4.2 Fonts

Fonts are a bit of a complex topic

TODO this part is just a collection of personal notes

- The missing guide to font formats
- RTD Code font issue on github

Tools:

Comprehensive article: https://www.useragentman.com/blog/2011/02/20/converting-font-face-fonts-quickly-in-any-os/
and https://www.useragentman.com/blog/the-css3-font-converter/

https://github.com/zoltan-dulac/css3FontConverter

woff2

https://github.com/google/woff2

sfnt2woff

```
sudo apt-get install libbrotli-dev
sfnt2woff SomeFont.otf
```

mkeot

---

49 https://creativemarket.com/blog/the-missing-guide-to-font-formats
50 https://github.com/readthedocs/sphinx_rtd_theme/issues/524
sudo apt-get install eot-utils
mkeot SomeFont.otf > SomeFont.eot

or https://github.com/wget/ttf2eot

FontForge (GUI and scriptable)
sudo apt-get install fontforge

4.3 font sizes

https://chiamakaikanyi.dev/sizing-in-css-px-vs-em-vs-rem/

4.4 Warning about old versions

ReadTheDocs has a mechanism\(^{51}\) to warn the user if he’s looking at an old version of the site, but we found it doesn’t work much for course-based documentation. So for versioning we think it’s better to adopt a mixed git branch / tags development model, and we added a template warning to show in old branches. To enable it in an old branch, just edit \_templates\_breadcrumbs.html\(^{52}\) as needed.

4.5 Fix nbshinx to create rst files

Sometimes nbshinx does not report properly RST conversion errors (see bug\(^{53}\)). As a hacky workaround, you might take the nbshinx.py from ~/.local/lib/python3.5/site-packages/., make a copy of it in your project home and patch it like this\(^{54}\) When you call sphinx, it will generate RST files in \_build\_jupman\_rst/.

Of course, things can be cleaner using a virtual env with venv\(^{55}\)

4.6 Git performance notes

Current suggested setup for hosting on Github is creating branch gh-pages and using Github Actions to populate it with html, zips, pdf and epub files. While keeping all that stuff versioned may seem pretty inefficient, apparently git is pretty good\(^ {56}\) at compressing binary files

The size of .git repo for a 1000 pdf page project SoftPython with 300 commits and 100 MB of code is:

| .git: 183 MB |

By truncating gh-pages to last commit and garbage collecting, we get:

\(^{51}\) https://docs.readthedocs.io/en/latest/versions.html
\(^{52}\) https://github.com/DavidLeoni/jupman/blob/master/_templates/breadcrumbs.html
\(^{53}\) https://github.com/DavidLeoni/jupman/issues/9
\(^{54}\) https://github.com/DavidLeoni/jupman/commit/0f332629ce4e2b0186c954c55ea7fa67992ace9#diff-bd3d9c4d2e80ed83fd2443d1301aa65bR649
\(^{55}\) https://docs.python.org/3/library/venv.html
\(^{56}\) https://stackoverflow.com/a/48305739
If we completely remove gh-pages branch, we get:

\[ \text{.git: 68.7 MB} \]

So gh-pages size is:

- one commit: 70.3 MB
- 300 commits: 114.3 MB

which is not even double than source code git size.

If the repo gets really huge, in order to to shrink it some git knowledge is required.

**If the repo is served from another server** and you want to truncate that server git repo:

On that server console:

1. first make sure you are on gh-pages branch:

```
git checkout gh-pages
```

2. truncates previous commits:

```
git fetch --depth=1 origin gh-pages
```

3. removes various links around which may still point to old commits:

```
git reflog expire --expire-unreachable=now --all
```

4. actually deletes from disk old commits:

```
git gc --aggressive --prune=all
```

Note the result of truncation cannot be pushed back to origin as git would complain it is a shallow branch.
Jupman comes with a script to manage exams called `exam.py`, which allows to manage the full cycle of an exam.

### 5.1 What is an exam

**Exam text** is represented as Jupyter notebooks, which are taken from `_templates/exam/solutions/exam-yyyy-mm-dd.ipynb`

**Exercises for students**: they are supposed to be the exam notebook itself and / or plain python files (or the notebook itself) plus unit tests and relative solutions.

**Marks spreadsheet**: By default there is also an LibreOffice spreadsheet to give marks, in case you need it.

When you initialize an exam with the `init` command, for example for date `2000-12-31`, all the presets in `_templates/exam/` are copied to `private/2000-12-31/` and `private/2000-12-31/solutions`. Presets can be changed at will to suit your needs. When packaging, student zip is assembled in `private/2000-12-31/student-zip`

System is flexible enough so you can privately work on next exams in `private/` folder and still being able to publish modifications to main website. After an exam, you can copy the private exam to the public folders in `past-exams/`.

### 5.2 Exam commands

To see the help:

```
python3 exam.py -h
```

To see help for a particular subcommand, like i.e. `init`, type the subcommand followed by `-h`:

```
python3 exam.py init -h
```

Running commands should be quite self-explanatory.

**NOTE**: as of today (Dec 2019) software may contain bugs, but at least we check for major misuses (like trying to overwrite existing exams).

In the file `create-exam-example.sh` there is a typical run of the script, which creates the example exam for date `2000-12-31`. Notice it might ask you to delete the existing `2000-12-31` exam, if it does just follow the instructions. Here’s the output:
> ./create-exam-example.sh
python3 exam.py init 2000-12-31
Detected release from git: 3.2.0-3-g30a995c
No GOOGLE_ANALYTICS environment variable was found, skipping it

You can now edit Python solutions, tests, exercises and exam notebook here:

  _private/2000-12-31/solutions

DONE.

python3 exam.py package 2000-12-31
Detected release from git: 3.2.0-3-g30a995c
No GOOGLE_ANALYTICS environment variable was found, skipping it
Cleaning _private/2000-12-31/server/jupman ...
Copying exercises to _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/
  Copying code
       from _private/2000-12-31/solutions
to _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/
    Generating _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/trees.py
    Writing _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/example.txt
    Generating _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/lists.py
    Writing (patched) _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/trees_test.py
    Writing (patched) _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/lists_test.py
    Creating dir _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/img
    Writing _private/2000-12-31/student-zip/jupman-2000-12-31-FIRSTNAME-LASTNAME-ID/img/mountains.jpg
Building pdf ...
  Writing jupman.py
  Writing my_lib.py
  Writing _static/img/cc-by.png
  Writing _static/js/jupman.js
  Writing _static/css/jupman.css
  Writing _static/js/toc.js
  Writing _static/js/pytutor-embed.bundle.min.js
Wrote _private/2000-12-31/server/jupman-2000-12-31-exam.zip

DONE.

------- Simulating some shipped exams...
mkdir -p _private/2000-12-31/shipped/john-doe-112233
  cp _templates/exam/solutions/lists_sol.py _templates/exam/solutions/lists_test.py _
  _templates/exam/solutions/trees_sol.py _templates/exam/solutions/trees_test.py _
  _private/2000-12-31/shipped/john-doe-112233
mkdir -p _private/2000-12-31/shipped/jane-doe-445566

(continues on next page)
cp _templates/exam/solutions/lists_sol.py _templates/exam/solutions/lists_test.py _
templates/exam/solutions/trees_sol.py _templates/exam/solutions/trees_test.py _
_private/2000-12-31/shipped/jane-doe-445566
------- Done with shipped exams simulation, time to grade ...

```bash
python3 exam.py grade 2000-12-31
Detected release from git: 3.2.0-3-g0a995c
No GOOGLE_ANALYTICS environment variable was found, skipping it
Copying Python files to execute and eventually grade in _private/2000-12-31/graded/
john-doe-112233/graded
Copying original shipped files (don't touch them!) in _private/2000-12-31/graded/
john-doe-112233/shipped
Copying Python files to execute and eventually grade in _private/2000-12-31/graded/
jane-doe-445566/graded
Copying original shipped files (don't touch them!) in _private/2000-12-31/graded/
jane-doe-445566/shipped
DONE.
```

```bash
python3 exam.py zip-grades 2000-12-31
Detected release from git: 3.2.0-3-g0a995c
No GOOGLE_ANALYTICS environment variable was found, skipping it
You can now find zips to send to students in _private/2000-12-31/graded
DONE.
```

```bash
python3 exam.py publish 2000-12-31
Detected release from git: 3.2.0-3-g0a995c
No GOOGLE_ANALYTICS environment variable was found, skipping it
Copying solutions to exams/2000-12-31/solutions
Copying exam PDF text
Exam Python files copied.
You can now manually build and run the following git instructions to publish the_
exam.
./build.py
  git status # just to check everything is ok
  git add .
  git commit -m 'published 2000-12-31 exam'
  git push
DONE.
```

Finished example exam run !!

[ ]:
6.1 Python example

6.1.1 Download exercises zip

Browse files online

Example of notebook for exercises in Python files

6.1.2 What to do

- unzip exercises in a folder, you should get something like this:

```
python-example
    python-example.ipynb
    lab1.py
    lab1_test.py
    lab1_sol.py
    lab2.py
    lab2_test.py
    lab2_sol.py
    jupman.py
    my_lib.py
```

- open the editor of your choice (for example Visual Studio Code, Spyder or PyCharme), you will edit the files lab1.py and lab2.py
- Go on reading this notebook, and follow instructions inside.

Let’s begin

You are going to program a simulator of bouncing clowns. To do so, we are going to load this module:

```
[2]: import local
```

```
[3]: local.gimme(5)
    It was a 5 indeed
```

57 https://github.com/DavidLeoni/jupman/tree/master/python-example
Download test data
Local file:
- example.txt
- example.csv

6.1.3 Global image

6.1.4 Local exercise image

6.1.5 Python tutor

```
[4]: x = [1, 2, 3]
y = 6

jupman pytut()
```

```
[4]: <IPython.core.display.HTML object>
```

```
[5]: y = [1, 2, 3]

jupman pytut()
```

```
[5]: <IPython.core.display.HTML object>
```

Start editing lab1.py in text editor
6.1.6 add

Implement add function:

```
[7]: add(3,5)
[7]: 8
```

6.1.7 sub

Implement sub function

```
[8]: sub(7,4)
[8]: 3
```

6.2 Jupyter example

6.2.1 Download exercises zip

Browse files online\(^{58}\)

Example of notebook for exercises in Jupyter files.

For python files based example and more, see [Python example](https://github.com/DavidLeoni/jupman/tree/master/jupyter-example)

6.2.2 What to do

- unzip exercises in a folder, you should get something like this:

  jupyter-example
  jupyter-example.ipynb
  jupyter-example-sol.ipynb
  jupman.py
  my_lib.py

  **WARNING:** to correctly visualize the notebook, it MUST be in an unzipped folder!

  - open Jupyter Notebook from that folder. Two things should open, first a console and then browser. The browser should show a file list: navigate the list and open the notebook `jupyter-example/jupyter-example.ipynb`
  - Go on reading that notebook, and follow instructions inside.

\(^{58}\) [https://github.com/DavidLeoni/jupman/tree/master/jupyter-example](https://github.com/DavidLeoni/jupman/tree/master/jupyter-example)
Shortcut keys:

- to execute Python code inside a Jupyter cell, press Control + Enter
- to execute Python code inside a Jupyter cell AND select next cell, press Shift + Enter
- to execute Python code inside a Jupyter cell AND create a new cell afterwards, press Alt + Enter
- If the notebooks look stuck, try to select Kernel -> Restart

```python
# REMEMBER TO IMPORT jupman!
# This cell needs to be executed only once, you can usually find it at the beginning of the worksheets
import jupman
```

```python
x = [1, 2, 3]
y = x
jupman.pytut()  # <IPython.core.display.HTML object>
```

```python
y = [1, 2, 3]
w = y[0]
jupman.pytut()  # <IPython.core.display.HTML object>
```

### 6.2.3 Exercise 1

Implement `inc` function:

```python
def helper(x):
    return x + 1
def inc(x):
    return helper(x)
</div>
```

```python
def inc(x):
    raise Exception('TODO IMPLEMENT ME !')
```
6.2.4 Exercise 2

Implement upper function

```
[6]:
def helper2(x):
    return x.upper()

def upper(x):
    return helper2(x)
```

Exercise 3

Note everything after the ‘write here’ comment will be discarded. Note you can put how many spaces you want in the comment

```
[7]:
w = 5
# write here
x = 5 + 6
y = 6.4
z = x / y
```

```
[7]:
w = 5
# write here
```
Exercise 4

Shows how to completely remove the content of a solution cell (including the solution comment)

**EXERCISE:** write a function that prints ‘hello’

```python
[8]: # SOLUTION
    "def f():
        print('hello')"

</div>

Exercise 5

Shows the QUESTION / ANSWER feature. All content in ‘ANSWER:’ cell will be stripped

**QUESTION:** Describe why iPhone n + 1 is better than iPhone n

```html
[8]: "<a class="jupman-sol" onclick="jupman.toggleSolution(this);" data-jupman-show="Show answer" data-jupman-hide="Hide">Show answer</a>"</html>

**ANSWER:** it costs more

```html
</div>

6.2.5 Conclusion

bla bla

Relative image test, Markdown format:

![Creative Commons License](https://example.com/license)

Relative image test, HTML `img` tag:

![Creative Commons License](https://example.com/license)

Relative link test, Markdown format:

*Back to index*

Relative link test, HTML `a` tag:

[Back to index](https://example.com)
6.3 Jupyter example with custom js and css

6.3.1 Download exercises zip

Browse files online\(^{59}\)

Example of notebook for exercises in Jupyter files which calls `jupman.init()` for injecting `jupman.js` and `jupman.css` while editing in Jupyter.

Note calling `init()` is **not** mandatory. See also Common files\(^{60}\) and Custom-js-and-css\(^{61}\).

6.3.2 Exercise 1

Implement `inc` function:

```
[2]:
def helper(x):
    return x + 1

def inc(x):
    return helper(x)
```

```
[2]:
def inc(x):
    raise Exception('TODO IMPLEMENT ME !')
```

6.3.3 Exercise 2

Implement `upper` function

```
[3]:
def helper2(x):
    return x.upper()

def upper(x):
    return helper2(x)
```

\(^{59}\) https://github.com/DavidLLeoni/jupman/tree/master/jupyter-example
\(^{60}\) https://jupman.softpython.org/en/latest/manual/editing.html#Common-files
6.4 Jupyter and Python example

6.4.1 Download exercises zip

Browse files online^62^ Most complex example of a notebook with exercises both in Jupyter and Python files, and ‘advanced’ features

6.4.2 What to do

- unzip exercises in a folder, you should get something like this:

```
jup-and-py-example
    jup-and-py-example.ipynb
    jup-and-py-example_sol.ipynb
    lab.py
    lab_test.py
    lab_sol.py
```

- open the editor of your choice (for example Visual Studio Code, Spyder or PyCharm), and edit `lab.py` file
- Go on reading this notebook, and follow instructions inside.

Let’s begin

You are going to program a simulator of bouncing clowns. To do so, we are going to load this module:

```
[2]: import local
[3]: local.gimme(5)
            It was a 5 indeed
```

Download test data

Local file:

- example.txt
- example.csv

6.4.3 Global image

6.4.4 Local exercise image

6.4.5 Python tutor

```python
[4]:
x = 5
y = 6
z = x + y

jupman.pytut()

[4]: <IPython.core.display.HTML object>
```

6.4.6 Exercise in Jupyter

Implement this function:

```python
[5]: def hello(s):
    return ['hello',s]*1000

hello_db = hello("Guybrush")
```

(continues on next page)
```python
[5]: def hello(s):
    raise Exception('TODO IMPLEMENT ME !')

hello_db = hello("Guybrush")

hello_db[:10]
```

Full expected output is in file `expected_output_db.py`, if you can’t manage to solve the exercise, as a last resort you can type: `from expected_hello_db import *` (DO NOT copy-paste file content, it would probably mess Jupyter up)

```python
[6]: from expected_hello_db import *
expected_hello_db[:10]
```

```
[6]: ['hello',
   'Guybrush',
   'hello',
   'Guybrush',
   'hello',
   'Guybrush',
   'hello',
   'Guybrush',
   'hello',
   'Guybrush']
```

Other example:

```python
[7]: hello_db2 = hello("Threepwood")

hello_db2[:10]
```

```
[7]: ['hello',
   'Guybrush',
   'hello',
   'Guybrush',
   'hello',
   'Guybrush',
   'hello',
   'Guybrush',
   'hello',
   'Guybrush']
```
6.4.7 Exercise using previous output

Write some code which says hello 3 times using previous function.

```
[8]: print(hello('Guybrush')[6])
['hello', 'Guybrush', 'hello', 'Guybrush', 'hello', 'Guybrush']
```

6.4.8 Question in Jupyter

**QUESTION:** Why learn coding?

**ANSWER:** So they pay me more

```
x + 1
```

Some other comment

Some nasty formatting

even more formatting

</div>

6.4.9 Exercise in Python

Start editing `lab.py` in text editor

```
[9]: from lab_sol import *
```
6.4.10 add

Implement add function:

```
add(3, 5)
```

```
8
```

6.4.11 sub

Implement sub function

```
sub(7, 4)
```

```
3
```

6.4.12 Fine grained purging

This cell input will be completely removed

```
print("This cell output will be completely removed")
```

```

```

6.5 Big sub chapter

6.5.1 Big docs example 1

reasonable header

Bla bla

reasonable sub header

Bla bla

Reasonable subsub header

Bla bla
reasonable header
Bla bla

reasonable sub header
Bla bla

Reasonable subsub header
Bla bla

header with long title
Bla bla

sub header with long title
Bla bla

header with long title
sub header with long title
subsub header with long title
header with long title
sub header with long title

header with extra super long title
sub header with extra super long title
sub sub header with extra super long title
header with extra super long title
sub header with extra super long title
header with extra super long title
sub header with extra super long title
header with extra super long title

6.5. Big sub chapter
sub header with extra super long title

header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header

header

sub header

header

sub header

reasonable header

reasonable sub header

reasonable header

reasonable sub header

header with long text

sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

6.5. Big sub chapter
sub header with extra super long text

header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
6.5.2 Big docs example 2

reasonable header
reasonable sub header
reasonable header
reasonable sub header
header with long text
sub header with long text
header with long text
sub header with long text
header with long text
sub header with long text
header with extra super long text
sub header with extra super long text
header with extra super long text
sub header with extra super long text
header with extra super long text
sub header with extra super long text
sub header with extra super long text
header with extra super long text
sub header with extra super long text
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
reasonable header
reasonable sub header
reasonable header
reasonable sub header
header with long text
sub header with long text
header with long text
sub header with long text
header with long text
sub header with long text
header with extra super long text
sub header with extra super long text
header with extra super long text
sub header with extra super long text
header with extra super long text

6.5. Big sub chapter
6.5.3 Big sub chapter A

Big docs example A1
reasonable header
reasonable sub header
reasonable header
reasonable sub header
header with long text
sub header with long text
header with long text
sub header with long text
header with long text
sub header with long text
header with extra super long text
Chapter 6. Chapter examples
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
reasonable header
reasonable sub header
reasonable header
reasonable sub header
header with long text
sub header with long text
header with long text
sub header with long text
header with long text

6.5. Big sub chapter
sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header
Big docs example A2
reasonable header
reasonable sub header
reasonable header
reasonable sub header
header with long text
sub header with long text
header with long text
sub header with long text
header with long text
sub header with long text
header with extra super long text
sub header with extra super long text
header with extra super long text
sub header with extra super long text
header with extra super long text
sub header with extra super long text
header with extra super long text
sub header with extra super long text
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
reasonable sub header

header with long text

sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header

sub header

header

sub header

header

sub header
6.5. Big sub chapter
Big sub chapter A.A

Big docs example AA1

reasonable header

reasonable sub header

reasonable header

reasonable sub header

header with long text

sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header
6.5. Big sub chapter
6.5. Big sub chapter
Big docs example AA2
header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header
sub header

header

sub header

header

sub header

header

sub header

header

sub header

reasonable header

reasonable sub header

reasonable header

reasonable sub header

header with long text

sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text
header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header
Big sub chapter A.B

Big docs example AB1

reasonable header

reasonable sub header

reasonable header

reasonable sub header

header with long text

sub header with long text

header with long text
sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header
Big sub chapter A.C

Big docs example AC1

reasonable header

reasonable sub header

reasonable header

reasonable sub header

header with long text

sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header
header

sub header

header

sub header

reasonable header

reasonable sub header

reasonable header

reasonable sub header

header with long text

sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

6.5. Big sub chapter
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header
sub header
header

sub header

header

sub header

header

sub header

[ ]:

Big docs example AC2

reasonable header

reasonable sub header

reasonable header

reasonable sub header

header with long text

sub header with long text

header with long text

sub header with long text

header with long text

sub header with long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

6.5. Big sub chapter
header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header
[ ]:

reasonable header
reasonable sub header
reasonable header
reasonable sub header

header with long text
sub header with long text
header with long text
sub header with long text
header with long text
sub header with long text

6.5.  Big sub chapter
header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header with extra super long text

sub header with extra super long text

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header

sub header

header
6.5.4 Big sub chapter B

Big docs example AB1
reasonable header
reasonable sub header
reasonable header
reasonable sub header
header with long text
6.5. Big sub chapter
Chapter 6. Chapter examples
6.5. Big sub chapter
6.6 Example Challenge

6.6.1 Download exercises zip

Browse files online\(^{63}\)

This notebook has no solution!

**We published solution on Github\(^ {64}\) only for example purposes**, but normally all other files ending in `-chal-sol` or `_chal_sol` will be gitignored.

```
[3]: from example_chal_sol import f
    f(3)

[3]: 3

[4]: def wow(x):
    raise Exception('TODO IMPLEMENT ME !')

[ ]:
```

\(^{63}\) https://github.com/DavidLeoni/jupman/tree/master/challenge-example

\(^{64}\) https://github.com/DavidLeoni/jupman/blob/master/challenge-example/example-chal-sol.ipynb
7.1 Changelog

Jupman: A template for online manuals made with Jupyter notebooks.
jupman.softpython.org

7.1.1 July 9th, 2023 - 3.5.7

Added:

• Softpython theme eye candy #122
  – softpython-theme-textures.css with manually embedded base64 images
  – jupman-alert-principle in jupman-web.css as backbone for commandments
• logo, favicon #123
• explicitly stated all dependencies in requirements-build.txt, added create_env script #82
• improved manual

Fixed:

• relative paths in cell outputs for zips #119
• now using relative js and css imports in jupman.init #117
• Python Tutor:
  – now shows unnest data structures by default #132
  – now shows in cloned cells #126 (changed stable ids #107)
  – sets are now correctly displaying with grey header #125
  – tutor Visual Studio Code no longer appears with grey text #136

Removed:

• _private/README.md

65 https://jupman.softpython.org
7.1.2 August 11th, 2022 - 3.5.4

- Python Tutor:
  - added credits #111
  - visualized vertical separator bar #110
  - fixed red arrow misalignment #105
  - stable ids #107
  - removed ‘Customize visualization’ #108
- Github actions: automatically build themed version #100
- SoftPython theme #92: various fixes
- Fixed relative html a, img with attributes in markdown not working in zip: #113

7.1.3 June 4th 2022 - 3.5

- generated html can be really used offline #96, also fixes wrong math symbols with local build #3
- automated testing on github actions #99
- virtual env install, pinpointed build depenedencies #82
- fixed text overflow on smartphones #94, fixed softpython theme flag
- github actions: always reset html output #98

7.1.4 April 28th 2022 - 3.4

- fixed softpython theme font size #92
- restructured manual

7.1.5 February 25th 2022 - 3.3

Implemented:

- jupman-preprocess #64
- big docs support #77
- Challenges support (suboptimal but usable) #56
- jupman-purge #59
- jupman.draw_text for #66
- jupman.save_pyfunction
- jupman.get_doc as nice way to print function documentation #68
- jupman.draw_text to show some ASCII characters in local build #66
- jupman.mem_limit for Linux #62
- Home link should point to index.html #71
- optional parameter conf to jmt.init
• deterministic zip #60 (requires python 3.7)

Fixed:
• Notebook validation failed: Non-unique cell id error #78
• exam pdf build breaks when using images #61

Defined:
• how to use custom anchors #80
• how to have single pages like References at menu bottom #70

7.1.6 October 17th 2020 - 3.2

• added optional build on Github Actions
• solutions are finally hidden on the website, with a click-to-show button!
• introduced generic jupman-toggleable and specific jupman-sol CSS classes
• improved menu navigation
• added softpython theme
• images are now shown centered in HTML
• moved to jupman.softpython.org
• fixed write here tag not preserving the line
• deprecated jupman_tools.ignore_spaces in favor of tag_regex
• updated nbsphinx to 0.7.1
• updated sphinx_rtd_theme to 0.4.3
• updated sphinx to 2.3.1
• updated pygments to 2.7.1

7.1.7 January 16th 2020 - 3.1

• removed jupman.init root parameter
• bugfixes
• upgraded nbsphinx from 0.3.4 to 0.5.0
• upgraded sphinx_rtd_theme from 0.2.5b1 to 0.4.3
• upgraded sphinx from 1.7.6 to 2.3.1
• upgraded recommonmark from 0.4.0 to 0.6.0
7.1.8 December 29th 2019 - 3.0

- much simplified folder structure
  - Issue 33
- removed solutions from header requirement
  - Issue 32
- introduced tests (pytest, hypothesis)
- removed old_news in favor of changelog.md
- Latex:
  - much better PDF cover
  - using xelatex
  - set up unicode mappings
- several fixes

7.1.9 September 24th 2018 - 2.0

- now using index.ipynb as home. Hurray!

7.1.10 September 19th 2018 - 1.0

- fixed build.py
- added html templates examples
- cleaned toc (was showing too much when loading)

7.1.11 August 26th 2018 - 0.9

- implemented generation of exercises from solutions [Issue 14](https://github.com/DavidLeoni/jupman/issues/14)
- reverted to old jupman.init() code Issue 12

7.1.12 August 12th 2018 - 0.8

- Prepended all functions in jupman.py with jupman_
- replaced index with proper homepage. see Issue 11
  - from now on you need home.ipynb file, because replacing index.rst is a nightmare!
  - new index.rst is just a placeholder which simply redirects to home.html. Do not modify it.
  - put the toctree in toc.rst

---

66 [https://github.com/DavidLeoni/jupman/issues/33](https://github.com/DavidLeoni/jupman/issues/33)
68 [https://github.com/DavidLeoni/jupman/issues/12](https://github.com/DavidLeoni/jupman/issues/12)
• exercises ipynb can now stay in exercises/ folder; when exercises are zipped, jupman automatically adds the zip the required site files. see Issue 12

• Tried %run at beginning of notebooks, without much satisfaction (see discussion in issue 12):
  • disabled toc by default in html files. To enable it, in python use %run -i ../../../jupman --toc
  • renamed past-exams directory from 'past-exams' to 'exams'
  • created info, error, warn, fatal functions to conf.py
  • introduced new variable exercise_common_files in conf.py for common files to be zipped
  • added pages exam-project, markdown, project-ideas.
  • added cc-by.png
  • renamed changelog.txt to changelog.md
  • now using templates with curly brackets in in templating, like _JM_{some_property}
  • jupman.js : now when manually saving html in Jupyter, resulting html correctly hides cells
  • Fixes https://github.com/DavidLeoni/jupman/issues/2: now toc is present in local build for pdfs

7.1.13 August 3rd 2018 - 0.7

• added jupman.py pytut() for displaying Python tutor in the cells
• added jupman.py toc=False option to jupman.py init to disable toc
• removed jupman.py useless networkx import from
• fixed usage indentation
• added changelog.txt

7.2 Past Exams

[ ]:

7.3 Exam project

**For general (credits, attendance) info, see course description

Delivery times

Ideas for possible projects: See here

Last update: TODO

In short:

70 https://github.com/DavidLeoni/jupman/issues/12
71 https://github.com/DavidLeoni/jupman/issues/12
7.3.1 What to do

First of all: send by email to TODO@TODO.COM a brief description of the project, to decide what to do. I will create a Google doc to keep track of progresses and/or problems found.

Once the project is defined, go on like this:

1 - Download zip with template (view online files TODO72)

After unzipped, you will find a folder named NAME-SURNAME-ID, with these files inside:

- NAME-SURNAME-ID
  - project.ipynb
  - markdown.ipynb
  - requirements.txt
  - img
    - example.png

2 - Rename the folder NAME-SURNAME-ID with your data
3 - run Jupyter from the folder you just renamed
4 - edit file project.ipynb, closely following the indications in the following technical requirements
5 - Once done, send project by email to TODO@TODO.COM

7.3.2 Technical requirements

Write in Markdown

Python code

requirements.txt file

Graphical interfaces

Be careful to

7.4 Project ideas

7.4.1 TODO

Last update: TODO

72 https://www.GITHUB.TODO
7.4.2 Introduction

7.5 Jupman Project

PUT:
TITLE
NAME-ID
DATAE

7.5.1 Introduction

Bla bla

7.5.2 Data sources

Bla bla

7.5.3 Data cleaning and integration

Bla bla

7.5.4 Analysis

Bla bla

7.5.5 Problems found

Bla bla

7.5.6 Conclusioni

Bla bla
7.6 Markdown

Briefly explain why markdown is so great.

[ ]:
8.1 Rendering tests

Shows various recommendations and cornercases, see also softpython themed\(^{73}\) version.
The page Title has one sharp, sections always have two sharps.

8.1.1 Section 1

bla bla

8.1.2 Section 2

Subsections always have three sharps

Subsection 1

bla bla

Subsection 2

bla bla

8.1.3 Python code

\[
\begin{align*}
\text{x} & = 4 \\
\text{lst} & = [1, 2, 4]
\end{align*}
\]

\(^{73}\) https://jupman.softpython.org/themed/manual/tests.html
8.1.4 Python in Markdown

\[
x = 4
\]

\[
\text{lst} = [1, 2, 4]
\]

8.1.5 Other quotes

I’m quoted with > symbol on multiple lines. **Am I readable?**

Somebody asked

I'm quoted with **spaces** on multiple lines
Am I readable?

Generic quote with backticks
Can you read it?
Is it nice?

```
# bash quote with backticks
cd foo
ls

cat wiz.txt
```

8.1.6 Download links

Files manually put in _static_:

- Download trial.odt
- Download trial.pdf

Files in arbitrary folder position:

- Download requirements.txt

**NOTE:** download links are messy, see issue 8\(^74\)

8.1.7 Links to HTML

- Link to trial.html

\(^74\) https://github.com/DavidLeoni/jupman/issues/8
8.1.8 Info/Warning Boxes

Inherited from NBSphinx:

Until there is an info/warning extension for Markdown/CommonMark, such boxes can be created by using HTML elements like the following

For this to work reliably, you should obey the following guidelines:

- The class attribute has to be either "alert alert-info" or "alert alert-warning", other values will not be converted correctly.
- No further attributes are allowed.
- For compatibility with CommonMark, you should add an empty line between the `<div>` start tag and the beginning of the content.

Note: This is an info, mixed bold!

Note: This is a warn, mixed bold!

Note: This is a long info, plain title, single paragraph
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.

This is a long warn, plain title, single paragraph
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.

Note: This is a long info, plain title, multiple paragraph
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

This is a long warn, plain title, multiple paragraph
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Note: This is a long info, plain title, multiple paragraph with code

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

```python
for x in range(4):
    if x in [8, 3, 1, 5]:
        print(x)
```

This is a long warn, plain title, multiple paragraph, code

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

```python
for x in range(4):
    if x in [8, 3, 1, 5]:
        print(x)
```

Note: This is a long info, bold title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.

This is a long warn, plain title!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquip ex ea commodo consequat.
**jupman-alert-principle**

**NOTE**: default colors are indicative and minimal on purpose, for a better view see [softpython themed version](https://jupman.softpython.org/themed/manual/tests.html#jupman-alert-principle).

**Recommended approach**: The typical principle alert should be brief and may have a link to more substantial text, with a short line under it. If you need more explicative text, put it outside:

```html
<div class="jupman-alert-principle"></div>
<div class="alert alert-info">
[IV PRINCIPLE](https://jupman.softpython.org/principles.html#V-PRINCIPLE): **You shall write tests!**

Who does **not** writes tests, falls into _Debugging Hell_!
</div>
```

**IV PRINCIPLE**: You shall write tests!

Who does not writes tests, falls into Debugging Hell!

**Normal weight font:**

**IV PRINCIPLE**: You shall write tests!

Who does not writes tests, falls into Debugging Hell!

**Shorter principle recall, bold:**

**IV PRINCIPLE**: You shall write tests!

**Shorter principle recall, normal font weight:**

**IV PRINCIPLE**: You shall write tests!

**Shorter principle recall, all bold**: (doesn’t work: shows artifacts)

`IV PRINCIPLE <https://jupman.softpython.org/principles.html#IV-PRINCIPLE>`__: You shall write tests!

**Longer principle with text outside the box** (recommended):

**VII PRINCIPLE**: You shall never ever add nor remove elements from a sequence you are iterating with a *for*!

Better to keep explanation text and code outside

---

75 https://jupman.softpython.org/themed/manual/tests.html#jupman-alert-principle
76 https://jupman.softpython.org/principles.html#IV-PRINCIPLE
77 https://jupman.softpython.org/principles.html#IV-PRINCIPLE
78 https://jupman.softpython.org/principles.html#IV-PRINCIPLE
79 https://jupman.softpython.org/principles.html#IV-PRINCIPLE
80 https://jupman.softpython.org/principles.html#VII-PRINCIPLE

---

8.1. Rendering tests
```python
lst = ['a', 'b', 'c']
for x in lst:
    lst.add(x)  # aaargh
```

**Longer principle with text inside the box** (not recommended):

\[\text{VII PRINCIPLE}^{81}: \text{You shall never ever add nor remove elements from a sequence you are iterating with a for!}\]

This is a **way too long** principle only for **testing** purposes, typically it’s much better keeping long stuff outside.

```python
lst = ['a', 'b', 'c']
for x in lst:
    lst.add(x)  # aaargh
```

**Longer principle with text inside the box, all bold** (not recommended, doesn’t work: shows artifacts):

`\text{VII PRINCIPLE}^{81}: \text{You shall never ever add nor remove elements from a sequence you are iterating with a for!}\`

This is a **way too long** principle only for **testing** purposes, typically it’s much better keeping long stuff outside.

```python
lst = ['a', 'b', 'c']
for x in lst:
    lst.add(x)  # aaargh
```

### 8.1.9 Math

For math stuff, see [nbsphinx docs]^{82}

Here we put just some equation to show it behaves fine in Jupman

This is infinity: $\infty$

### 8.1.10 Unicode

Unicode characters should display an HTML, but with latex you might have problems, and need to manually map characters in conf.py

You should see a star in a black circle: \text{⭐}

You should see a check: \text{✓}

Table characters: | — —

---

81 [https://jupman.softpython.org/principles.html#VII-PRINCIPLE](https://jupman.softpython.org/principles.html#VII-PRINCIPLE)

82 [https://nbsphinx.readthedocs.io/en/0.2.14/markdown-cells.html#Equations](https://nbsphinx.readthedocs.io/en/0.2.14/markdown-cells.html#Equations)
8.1.11 Tables

A small table

<table>
<thead>
<tr>
<th>Name</th>
<th>Surname</th>
<th>Age</th>
<th>Occupation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Augustus</td>
<td>Gloop</td>
<td>40</td>
<td>Software Engineer</td>
</tr>
<tr>
<td>Veruca</td>
<td>Salt</td>
<td>34</td>
<td>Writer</td>
</tr>
<tr>
<td>Violet</td>
<td>Beauregarde</td>
<td>37</td>
<td>Accountant</td>
</tr>
</tbody>
</table>

An extra large table

<table>
<thead>
<tr>
<th>Why</th>
<th>do</th>
<th>you</th>
<th>need</th>
<th>so</th>
<th>much</th>
<th>chocolate?</th>
<th>I don’t</th>
<th>understand.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oompa</td>
<td>Loompa,</td>
<td>do</td>
<td>ba</td>
<td>dee</td>
<td>doo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oompa</td>
<td>Loompa,</td>
<td>do</td>
<td>ba</td>
<td>dee</td>
<td>doo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I've</td>
<td>got</td>
<td>a</td>
<td>perfect</td>
<td>puzzle</td>
<td>for</td>
<td>you.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If</td>
<td>you</td>
<td>are</td>
<td>wise</td>
<td>you'll</td>
<td>listen</td>
<td>to</td>
<td>me.</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>do</td>
<td>you</td>
<td>get</td>
<td>when</td>
<td>you</td>
<td>guzzle</td>
<td>down</td>
<td>sweets?</td>
</tr>
</tbody>
</table>

A table with extra large text

Note with default style text overflows, we fix this in themes

<table>
<thead>
<tr>
<th>Who</th>
<th>do you</th>
<th>blame?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gum chewing’s fine when it’s once in a while.</td>
<td>It stops you from smoking and brightens your smile.</td>
<td>But it’s repulsive revolting and wrong.</td>
</tr>
<tr>
<td>Blaming the kids is a lie and a shame</td>
<td>You know exactly who’s to blame:</td>
<td>The mother and the father!</td>
</tr>
</tbody>
</table>

A table with links

Some of these links in markdown tables may show ill-formatted in HTML output

Test for issue 137

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td>lunghezza</td>
<td>len(str)</td>
<td>int</td>
<td>Ritorna la lunghezza della stringa</td>
</tr>
<tr>
<td>indice</td>
<td>str[int]</td>
<td>str</td>
<td>Legge il carattere all’indice specificato</td>
</tr>
<tr>
<td>concatenazione</td>
<td>str1 + str2</td>
<td>str</td>
<td>Concatena due stringhe</td>
</tr>
<tr>
<td>inclusione</td>
<td>str1 in str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>

83 https://github.com/DavidLeoni/jupman/issues/137

8.1. Rendering tests
## Jupman, Release 3.5.7

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>inclusione &lt;#Operatore-in&gt;</code></td>
<td>str1 in str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>inclusione &lt;#Operatore-in&gt;</code></td>
<td>str1 inabc str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>

with `#Operatore-inxyz` link and `inabc` operator:

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td>inclusione</td>
<td>str1 inabc str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>

with `#Operatore-inxyz` link and `in` operator:

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td>inclusione</td>
<td>str1 in str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>

with `#Operatore-in` (with end space) link and `in` operator:

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>inclusione &lt;#Operatore-in&gt;</code></td>
<td>str1 in str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>

with `#Operaz` link and `in` operator:

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td>inclusione</td>
<td>str1 in str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>

Workaround with `\`<a>`\` tags

… doesn’t work :-(

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Sintassi</th>
<th>Risultato</th>
<th>Significato</th>
</tr>
</thead>
<tbody>
<tr>
<td>lunghezza</td>
<td>len(str)</td>
<td>int</td>
<td>Ritorna la lunghezza della stringa</td>
</tr>
<tr>
<td>indice</td>
<td>str[int]</td>
<td>str</td>
<td>Legge il carattere all’indice specificato</td>
</tr>
<tr>
<td>concatenazione</td>
<td>str1 + str2</td>
<td>str</td>
<td>Concatena due stringhe</td>
</tr>
<tr>
<td>inclusione</td>
<td>str1 in str2</td>
<td>bool</td>
<td>Controlla se la stringa è presente in un’altra stringa</td>
</tr>
</tbody>
</table>
Leggere caratteri
bla bla

Sostituire caratteri
bla bla

Operatore in
bla bla

Opera in
bla bla

Operaz in
bla bla

Operatore inxyz
bla bla

8.1.12 Lists

• Home
  – Contents
  – Revisions

• More
  – Quickstart
  – Configure
    • Building the manual
    • Publish
    • Further steps
  – Installation
8.1.13 Images

SVG Images

SVG images work in notebook, but here it is commented since it breaks Latex, see issue\(^84\)

```latex
![An image](../img/cc-by.svg)
```

This one also doesn’t work (and shows ugly code in the notebook anyway)

```python
from IPython.display import SVG
SVG(filename='../img/cc-by.svg')
```

PNG Images

![Jupyter Nb](../img/jupman/notebook_icon.png)

Inline images - pure markdown

```markdown
Bla ![A PNG image md](../_static/img/jupman/notebook_icon.png) bli blo
```

Inline images - markdown and img

```markdown
bla <img alt="markimg84545" style="display:inline" src="../_static/img/jupman/_notebook_icon.png"> bli blo
```

\(^84\) https://github.com/DavidLeoni/jupman/issues/1
**Img class**

If we pass a class, it will to be present in the website:

```
<img alt="markimg7325" class="jupman-inline-img" src="../_static/img/jupman/notebook_icon.png">
```

This should be inline

---

**A picture with alternate text**

---

### 8.1.14 Expressions list

Highlighting **does** work both in Jupyter and Sphinx

Three quotes, multiple lines - Careful: put **exactly 4 spaces** indentation

1. \[2,3,1\] != "[2,3,1]"

2. \[4,8,12\] == [2*2,"4*2",6*2]

3. [[]:] == []

Three quotes, multiple lines, more compact - works in Jupyter, **doesn’t** in Sphinx

1. python [2,3,1] != "[2,3,1]"
2. python \[4,8,12\] == \[2*2,"4*2",6*2\]
3. python [][:]== []

Highlighting \textbf{doesn’t} work in Jupyter neither in Sphinx:

Three quotes, single line
1. python \[2,3,1\] != \["2",3,1\]
2. python \[4,8,12\] == \[2*2,"4*2",6*2\]
3. python [][:]== "[]"

Single quote, single line
1. python \[2,3,1\] != \["2",3,1\]
2. python \[4,8,12\] == \[2*2,"4*2",6*2\]
3. python [][:]== "[]"

\section*{8.1.15 Toggable cells}

There are various ways to have toggable cells.

\textbf{Show/hide exercises (RECOMMENDED)}

If you need clickable show/hide buttons for exercise solutions, see here: \url{Editing - Exercise types}\textsuperscript{85}. It manages comprehensively use cases for display in website, student zips, exams, etc.

If you have other needs, we report here some test we made, but keep in mind this sort of hacks tend to change behaviour with different versions of jupyter.

\textbf{Toggling with Javascript}

- Works in MarkDown
- Works while in Jupyter
- Works in HTML
- Does not show in Latex (which might be a good point, if you intend to put somehow solutions at the end of the document)
- NOTE: after creating the text to see the results you have to run the initial cell with jupman.init (as for the toc)
- NOTE: you can’t use Markdown block code since of Sept 2017 doesn’t show well in HTML output

\textsuperscript{85} \url{https://jupman.softpython.org/en/latest/manual/editing.html#Exercise-types}
HTML details in Markdown, code tag

- Works while in Jupyter
- Doesn’t work in HTML output
- as of Sept Oct 2017, not yet supported in Microsoft browsers

Click here to see the code

```python
question = raw_input("What?")
answers = random.randint(1,8)
if question == "":
    sys.exit()
```

HTML details in Markdown, Markdown mixed code

- Works while in Jupyter
- Doesn’t work in HTML output
- as of Sept Oct 2017, not yet supported in Microsoft browsers

Click here to see the code

```python
question = raw_input("What?")
answers = random.randint(1,8)
if question == "":
    sys.exit()
```

HTML details in HTML, raw NBConvert Format

- Doesn’t work in Jupyter
- Works in HTML output
  - NOTE: as of Sept Oct 2017, not yet supported in Microsoft browsers
- Doesn’t show at all in PDF output

Some other Markdown cell Afterwards ….

8.1.16 Stripping answers

For stripping answers examples, see jupyter-example/jupyter-example-sol. For explanation, see editing

8.1.17 Files in templates

Since Dec 2019 they are not accessible see issue 1086, but it is not a great problem, you can always put a link to Github, see for example exam-yyyy-mm-dd.ipynb87

86 https://github.com/DavidLeoni/jupman/issues/10
8.1.18 Inline text style

For colors on the fly you can use inline style in markdown. Works on:

- website: yes
- Jupyter: no
- pdf: no

```html
<style>
  .be-fancy {
    color: red;
  }
</style>

<div class="be-fancy">
  This should display in red
</div>

This should display in red

8.1.19 Formatting problems

Characters per line

Python standard for code has limit to 79, many styles have 80 (see Wikipedia\(^{88}\)). We can keep 80 like this, which should not display a scrollbar:

Plain:

```
--------------------------------------------------------------------------------
```

As python markup:

```
print ('-'*80)
```

This should not display a scrollbar:

```
[3]: `-'*78
[3]: '--------------------------------------------------------------------------------'
```

This should not display a scrollbar:

```
[4]: print ('-'*80)
[5]: `-'*80
[5]: '--------------------------------------------------------------------------------'
```

On website this may display a scroll bar, because it will actually print ' apexes plus the dashes

\(^{88}\) https://en.wikipedia.org/wiki/Characters_per_line
Note errors hold 75 dashes:

Plain:

```
ZeroDivisionError: division by zero
```

As python markup:

```
ZeroDivisionError: division by zero
```

[6]: `len('-----------------------------------------------')`

```
75
```

**Very long input**

In Jupyter: default behaviour, show scrollbar

On the website: default behaviour, don’t show scrollbar

Test for issue 122\(^89\) softpython theme eye candy scrollbars

```
[7]:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

\(^{89}\) [https://github.com/DavidLeoni/jupman/issues/122](https://github.com/DavidLeoni/jupman/issues/122)
80

[7]: 80
Very large input

In Jupyter: default behaviour, show scrollbar

On the website: default behaviour, show scrollbar

```
[8]: # line with an exceedingly long comment line with an exceedingly long comment line with an...  
  -> with an exceedingly long comment line with an exceedingly long comment line with an...  
  -> exceedingly long comment line with an exceedingly long comment

# line with an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...  
-> an out-of-this-world long comment line with an an out-of-this-world long comment...  
-> line with an an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...  
-> an out-of-this-world long comment line with an an out-of-this-world long comment...  
-> line with an an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...  
-> an out-of-this-world long comment line with an an out-of-this-world long comment...  
-> line with an an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...  
-> an out-of-this-world long comment line with an an out-of-this-world long comment...  
-> line with an an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...  
-> an out-of-this-world long comment line with an an out-of-this-world long comment...  
-> line with an an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...  
-> an out-of-this-world long comment line with an an out-of-this-world long comment...  
-> line with an an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...  
-> an out-of-this-world long comment line with an an out-of-this-world long comment...  
-> line with an an out-of-this-world long comment line with an an out-of-this-world...  
-> long comment line with an an out-of-this-world long comment line with an an out-of-  
-> this-world long comment line with an an out-of-this-world long comment line with an...
```

(continues on next page)
Very long and large input

In Jupyter: default behaviour, show horizontal scrollbar

On the website: default behaviour, show horizontal scrollbar

Test for issue 122\(^9\) softpython theme eye candy scrollbars

\[9\]:
```python
# xxxxxxxx
# xxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

\(^{90}\) https://github.com/DavidLeoni/jupman/issues/122
Very long HTML (and long code line)
Should expand in vertical as much as it wants.

[10]: %html

```html
<iframe width="100%" height="1300px" frameborder="0" src="https://umap.openstreetmap.fr/en/map/mia-mappa-agritur_182055?scaleControl=false&miniMap=false&scrollWheelZoom=false&zoomControl=true&allowEdit=false&moreControl=true&searchControl=null&tilelayersControl=null&embedControl=null&datalayersControl=true&onLoadPanel=undefined&captionBar=false#11/46.0966/11.4024"></iframe>
```

Very long output

In Jupyter: by clicking, you can collapse
On the website: a scrollbar should appear

[11]:

```python
for x in range(150):
    print('long output ...', x)
```

long output ... 0
long output ... 1
long output ... 2
long output ... 3
long output ... 4
8.1. Rendering tests

long output ... 62
long output ... 63
long output ... 64
long output ... 65
long output ... 66
long output ... 67
long output ... 68
long output ... 69
long output ... 70
long output ... 71
long output ... 72
long output ... 73
long output ... 74
long output ... 75
long output ... 76
long output ... 77
long output ... 78
long output ... 79
long output ... 80
long output ... 81
long output ... 82
long output ... 83
long output ... 84
long output ... 85
long output ... 86
long output ... 87
long output ... 88
long output ... 89
long output ... 90
long output ... 91
long output ... 92
long output ... 93
long output ... 94
long output ... 95
long output ... 96
long output ... 97
long output ... 98
long output ... 99
long output ... 100
long output ... 101
long output ... 102
long output ... 103
long output ... 104
long output ... 105
long output ... 106
long output ... 107
long output ... 108
long output ... 109
long output ... 110
long output ... 111
long output ... 112
long output ... 113
long output ... 114
long output ... 115
long output ... 116
long output ... 117
long output ... 118

(continues on next page)
Very large output

In Jupyter: automatically returns carriage

On the website: horiz scrollbar should appear

```python
[12]: print(', '.join([str(x) for x in range(150)]))
```

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 
88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 
112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 
133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149
Very long output and large output

In Jupyter: by clicking, you can collapse

On the website: both scrollbars should appear

```
[13]:  for x in range(150):
      print('long and large output ...', 'z'*x)
long and large output ...
long and large output ... z
long and large output ... zz
long and large output ... zzz
long and large output ... zzzz
long and large output ... zzzzz
long and large output ... zzzzzz
long and large output ... zzzzzzz
long and large output ... zzzzzzzz
long and large output ... zzzzzzzzz
long and large output ... zzzzzzzzzz
long and large output ... zzzzzzzzzzz
long and large output ... zzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
long and large output ... zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
(continues on next page)```
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
long and large output ...
(continues from previous page)

(continues on next page)
8.1.20 Python tutor tests

See Pytutor page

8.2 Python Tutor tests

There are various ways to embed Python tutor, first we put the recommended one.

RECOMMENDED: You can put a call to `jupman.pytut()` at the end of a cell, and the cell code will magically appear in python tutor in the output (except the call to `pytut()` of course). Does not need internet connection.

[1]:
```python
import sys
sys.path.append('..')
import jupman
```

[2]:
```python
lst = [5, 8, 4, 10, 30, 20, 40, 50, 60, 70, 20, 30]
for x in lst:
    y = x * 2
jupman.pytut()
```

[2]: <IPython.core.display.HTML object>

8.2.1 Scope

BEWARE of variables which were initialized in previous cells, they WILL NOT be available in Python Tutor:

[3]:
```python
w = 8
```

[4]:
```python
x = w + 5
jupman.pytut()
```

Traceback (most recent call last):
  File "./jupman.py", line 2453, in _runscript
    self.run(script_str, user_globals, user_globals)
(continues on next page)
8.2.2 Data types

Data display review:

```python
[5]:
word = "wonder"
lst = [9,7,8]
t = (6,2,4,3)
mixed_set = {3,'b','e',9, 'm', 'n'}
numbers_set = {3,4,9}
char_set = {'r','a','h'}
d = {'a' : 8,
     'b' : 4,
     'c' : 5}
empty_str = ""
empty_tuple = ()
empty_set = set()
empty_dict = {}

#import numpy  # not supported!

def f(par):
    print(par)

class C:
    def hello():
        print('ciao')

        def __init__(self, name, age):
            self.name = name
            self.age = age

m = C.hello
g = f
c = C('blah',30)
jupman.pytut()
```

[5]: <IPython.core.display.HTML object>
8.2.3 Window overflow

When too much right space is taken, it might be difficult to scroll:

8.2.4 pytut execution

Some cells might execute in Jupyter but not so well in Python Tutor, due to its inherent limitations:

8.2.5 Infinite loops

Since execution occurs first in Jupyter and then in Python tutor, if you have an infinite loop no Python Tutor instance will be spawned:

---

91 https://github.com/pgbovine/OnlinePythonTutor/blob/master/unsupported-features.md
### 8.2.6 Resizability

Long vertical and horizontal expansion should work:

```python
x = {0:'a'}
for i in range(1,30):
    x[i] = x[i-1]+str(i*10000)
jupman.pytut()
```

With multiple visualizations, arrows shouldn’t cross from one to the other even if underlying script is loaded multiple times (relates to visualizerIdOverride)

```python
x = [1,2,3]
jupman.pytut()
```

### 8.2.8 Print output

With only one line of print, Print output panel shouldn’t be too short:

```python
print("hello")
jupman.pytut()
```

### 8.2.9 Alignment test 1

Test for [https://github.com/DavidLeoni/jupman/issues/105](https://github.com/DavidLeoni/jupman/issues/105)

Before fixing red arrow was badly misaligned and went way past the function call.

```python
def f(mat, i):
    """ Do something"
    row = []
    row.append(mat[i])
    return row
def([2,3,4]), 0)
jupman.pytut()
```
8.2.10 Alignment test 2

Check the arrow stays on right line even in long code.

```python
[13]: def f(mat, i):
    """ Do something else"
    row = []
    row.append(mat[i])
    return row

m = [
    ['a','b'],
    ['c','d'],
    ['a','e'],
]

#row = f(m,0)

x = 3
if x == x:
    m = [
        [1,2,3],
        [4,3]
    ]
print("zim")

y = 1
if y == y:
    m = [
        [1,2,3],
        [4,3]
    ]
print("zam")
```

jupman.pytut()

zim
zam

[13]: <IPython.core.display.HTML object>

8.2.11 Cloned cell test

Tests PythonTutor output is correctly displayed when cells are cloned (solved issue\(^{92}\))

```python
[14]: maremma = 123
jupman.pytut()

[14]: <IPython.core.display.HTML object>
```

Cloned cell:

```python
[15]: maremma = 123
jupman.pytut()
```

\(^{92}\) https://github.com/DavidLeoni/jupman/issues/126
8.2.12 Don’t nest data structures

By default data structures shouldn’t be displayed as nested (which is the actual default on Python Tutor site). Test for this issue\(^3\):

```python
listA = [['Keep', 'an'],
         ['eye', 'on'],
         ['the', 'arrow', 'tips']
]
listB = listA.copy()
jupman.pytut()
```

8.2.13 Force nested data structures

```python
listA = [['Keep', 'an'],
         ['eye', 'on'],
         ['the', 'arrow', 'tips']
]
listB = listA.copy()
jupman.pytut(disableHeapNesting=False)
```

8.2.14 Function pointers

Containers should always contain links to functions, not inlined names

```python
def f():
    print('hello')
def g():
    print('world')
fs = [f, g]
jupman.pytut()
```

\(^3\) [https://github.com/DavidLeoni/jupman/issues/132](https://github.com/DavidLeoni/jupman/issues/132)
8.2.15 Errors - Code after pytut

```
x = 3
jupman.pytut()
print("ciao")
```

ERROR: the call to jupman.pytut() must be the last in the cell, instead, found this code afterwards:

```
print("ciao")
ciao
```

8.2.16 Errors - pytut double call

```
x = 3
jupman.pytut()
jupman.pytut()
```

ERROR: There should only be one call to jupman.pytut(), found 2 instead

8.2.17 Errors - Nothing to show

```
jupman.pytut()
```

Nothing to show! You have to put ALL the code IN THE SAME cell as pytut() right before its call.

Example:

```
x = 5
y = x + 3
jupman.pytut()
```

8.2.18 Spaces in attributes

Check spaces don't affect jupman.pytut() call strip

```
la = [['a', 'b'], ['c', 'd']]
jupman.pytut( disableHeapNesting = False )
```

```
<IPython.core.display.HTML object>
```
8.2.19 Alternative: HTML magics

Another option is to directly paste Python Tutor iframe in the cells, and use Jupyter %%HTML magics command. HTML should be available both in notebook and website - of course, requires an internet connection.

Beware: you need the HTTPS!

```bash
[23]: %%HTML
<iframe width="800" height="300" frameborder="0"
    src="https://pythontutor.com/iframe-embed.html#code=x+5%0Ay+10%0Az+x+y&cumulative=false&py=3&curInstr=3">
</iframe>
<IPython.core.display.HTML object>
```

8.2.20 Alternative: NBTutor

To show Python Tutor in notebooks, there is already a jupyter extension called NBTutor\(^{94}\), afterwards you can use magic ```nbtutor``` to show the interpreter.

Unfortunately, it doesn’t show in the generated HTML :-/

```bash
[24]: %reload_ext nbtutor
[25]: %nbtutor
    for x in range(1,4):
        print("ciao")
    x=5
    y=7
    x +y

    ciao
ciao
ciao
[25]: 12
```

\(^{94}\) [https://github.com/lgpage/nbtutor](https://github.com/lgpage/nbtutor)
REFERENCES

Shows how to put a single page at the bottom of the sidebar, visible without being inside a section. See this issue\footnote{https://github.com/DavidLeoni/jupman/issues/70}