

Automatic pdf report from a python script with pandoc - revision 2

ChristianV

19 Mar 2023

Abstract

Export a pdf report (code, outputs) commented with the Markdown embedded in the python script, a bash preprocessing and pandoc. Everything is launched within the python script.

Contents

1	Which problem does this solve?	1
2	How does it work?	1
2.1	The features	1
2.2	The proposed solution	2
2.3	The sources of inspiration	2
2.4	What is missing for the magic complete	3
2.4.1	Code to be added	3
2.4.2	Resources	3
2.4.3	Github repository	3
2.5	Whish list	3
2.6	What does the python script look like ?	3
3	python code	5
3.1	Import modules	5
3.2	Functions	5
3.3	Main code	6
3.3.1	Code section 1	6
3.3.2	Code Output (section 1)	6
3.3.3	Code section 2	6
3.3.4	Code Output (section 2)	6
3.3.5	Code last section	7
3.3.6	Code output (last section)	7
3.3.7	Code of the bash file	7
3.4	Additional output	8
3.5	Test of indented lines	8

Revision 2 : improvement to work with a script formatted with [Black](#) and inspected with [Pylint](#).

1 Which problem does this solve?

To have the possibility to make a report linked to a python script is an interesting function. It allows to keep notes about the script (why, how), to show and share the results. A good location of the report is inside the script itself. A pdf file as documentation has in addition a comfortable readability. Tools like [Pweave](#) offer this functionalities, but my knowledge is inadequate to solve the problems that may occur... or occurred like the last time I updated my Manjaro laptop...

So I came to this solution that uses standard Linux tools, explicit and simple steps.

2 How does it work?

2.1 The features

The expected features are :

- To export a pdf file that includes :
 - The python code cut in chunks
 - All the needed text to explain the goal and outputs of the script
 - Math and formulas
 - The outputs of the script, to the console or graphs (matplotlib)
- The script shall remain directly executable with no modifications
- The only maintained file is the script

2.2 The proposed solution

The script shall include :

- Some codes to store the outputs
- Lines of markdown starting by a mark
- Some lines to include external text files

The mark is a unique sequence of characters starting by “#” (comment in python)

Then the script is preprocessed to :

- remove the first lines (python3 and coding specification)
- remove the mark
- insert the external files
- store the result into an intermediate markdown file

The markdown file is then transformed in pdf with pandoc

2.3 The sources of inspiration

The idea of a mark “#%” comes from pweave, spyder to embedded markdown in the python script.

The first test shown that with, at least, a French keyboard this sequence is so easy to use (need AltGr then shift) and is visually intrusive in editor. Several “#” are not allowed (used by markdown) so I changed to “# ~” as “~” is next key on the left of “#”.

Revision 2 note 1 : the mark was “#~” in version 1 which doesn’t fit with Python coding rule (a comment mark # is followed by a blank). In rev 2 : the mark is “# ~”.

Revision 2 note 2 : a line “# ~ end of Python header, start of markdown py2pdf” is added to separate the Python information of the first lines from the beginning of the Pandoc header.

The script filename.py is copied into an image markdown file temp2.md and then temp.md

The script can remove the first lines and the mark, remove the line after “# ~ no export to pdf from here”. This separator was created in order to have the possibility to limit the length of the pdf file; for example if showing the code or part of it has no interest.

Revision 2 note 3 : the compatibility of the bash file with the original mark “#~” is kept.

See at the end of the document the bash script.

A “magic” call to perl can insert external text files. It is proposed by stackoverflow ie [Embedding one markdown document in another](#)

```
perl -ne 's/^\!\[([.+?)]\]\.*/'cat $1'/e;print' temp.md > filename.md
```

All the found ![textfilepath] in temp.md will be replaced by the textfilepath file content in result.md

This method can be applied to subpart of the file like the header to get more compact script or any other text file to share like a bash file.

Finally the markdown image is exported to pdf with pandoc

```
pandoc -s -o filename.pdf filename.md
```

The intermediate files temp2.md, filename.md are deleted. The temp.md is kept for debug.

There are probably more interesting options of pandoc to use... This will be another task

2.4 What is missing for the magic complete

2.4.1 Code to be added

Some code lines are added to prepare the structure : a directory is created to store the output files.

A local print function allows to print in the console and in a logfile. The logfile name is changed when necessary to separate the output. Inspiration for this comes from stackoverflow [print on console and text file simultaneously python](#)

Some code is also needed to store the desired matplotlib graphs.

The command lines are embedded in a bash file called at the end of the script. The bash file can be in the same directory than the python script. For a more productive configuration, the bash file might be located in one directory, an alias added in ".bashrc". The call to script is then :

```
cmd = "py2pdf scriptname"
subprocess.call(['/bin/bash', '-i', '-c', cmd])
```

"py2pdf" being the alias.

Then in the ".bashrc" file :

```
# my scripts
p="$HOME/0_myscripts/" # or any other path to your scripts
alias py2pdf="$p/py2pdf"
```

2.4.2 Resources

- pandoc : pandoc.org
- python, bash, perl : standard of Linux distribution?
- about sed [Commande sed : utilisation et exemples](#). Sorry it is a page in French but it helps me to write more robust sed commands.

2.4.3 Github repository

The files are under Github [Homeswinghome/py2pdf](#)

2.5 Whish list

- Replace the bash script by python for a better portability (Pandoc is available for Windows)
- Manage versions of the export : it happens a report is replaced by a new trial. Oups!
- Find a way to have a shortcut in an editor to insert the mark (my current editors : gedit / VSodium)

2.6 What does the python script look like ?

Here are some lines from the raw file "scriptpython2pdf.py" (do not confuse with its export to pdf):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# pylint: disable=line-too-long
# end of Python header, start of markdown py2pdf
# ~ ~ ~
# ~ ~ title : Automatic pdf report from a python script with pandoc
# ~ ~ author : ChristianV
# ~ ~ date : 14 Oct 2022
# ~ ~ abstract : Export a pdf report (code, outputs) commented with the Markdown embedded in the python
# ~ ~ lang: en-US
# ~ ~ geometry:
# ~ ~ - margin = 2cm
# ~ ~ - a4paper
# ~ ~ toc: true
# ~ ~ toc_depth: 2
# ~ ~ numbersections: true
# ~ ~ links-as-notes: false
# ~ ~ linkcolor: blue
```

```

# ~ ~ ---
# ~ ~
# ~ ~ # Which problem does this solve?
# ~ ~
# ~ ~ To have the possibility to make a report
# ~ ~..../
# ~ ~ Some more lines here
# ~ ~./...
# ~ ~ Some code is also needed to store the desired matplotlib graphs.
# ~ ~
# ~ ~ The command lines are embedded in a bash file called at the end of the script. In this example the
# ~ ~ '''
# ~ ~ py2pdf scriptname
# ~ ~ '''
# ~ ~ "py2pdf" being the alias.
# ~ ~
# ~ ~ Then in the ".bashrc" file :
# ~ ~ '''bash
# ~ ~ # my scripts
# ~ ~ p="$HOME/0_myscripts/" # or any other path to your scripts
# ~ ~ alias py2pdf="$p/py2pdf"
# ~ ~ '''
# ~ ~
# ~ ~ ### Resources
# ~ ~ * pandoc : [pandoc.org](https://pandoc.org/)
# ~ ~ * python, bash, perl : standard of Linux distribution?
# ~ ~
# ~ ~ # python code
# ~ ~
# ~ ~ ## Import modules
# ~ ~ '''python

# import
import os # for py2pdf
import subprocess # for py2pdf with bash with alias
import matplotlib.pyplot as plt # for the demo
import numpy as np # for the demo

# ~ ~ '''
# ~ ~
# ~ ~ ## Functions
# ~ ~
# ~ ~ This is a demo file. In other applications, for a better readability, those specific functions might
# ~ ~ '''python

def print_twice(log, *args, **kwargs):
    # the double print function (to console and to log)
    print(*args, **kwargs)
    with open(log, "a") as f: # appends to file and closes it when finished
        print(file=f, *args, **kwargs)

# ~ ~..../
# ~ ~ Some more lines here
# ~ ~./...
# ~ ~ '''
# ~ ~ ## Main code
# ~ ~
# ~ ~ ### Code section 1
# ~ ~ '''python

```

```

# lines to be included for py2pdf export
# get this script file name without extension
scriptname = os.path.basename(__file__).split(".")[0]
logfile = "./py2pdf_files/log1.txt" # define the logfile
outputdir() # create the directory to store the output
clearlog(logfile) # clear the logfile (in case script is ran several times)

# code for console output demo
path = os.getcwd() # get the path of the current directory
print_twice(logfile, "Path of the current directory : " + path)
print_twice(logfile, "python script name : ", scriptname)
print_twice(logfile, "Directory content :")
print_twice(logfile, "\n".join(os.listdir(path)))
print_twice(logfile, "Logfile : ", logfile)
# ~ ~ ~ ~ ~
# ~ ~ ### Code Output (section 1)
# ~ ~ ~ ~ ~
# ~ ~ ![[./py2pdf_files/log1.txt]]
# ~ ~ ~ ~ ~
# ~ ~ .... /
# Some more lines here
# ~ ~ ./...
# prepare the command to launch the report creation
# cmd = "./py2pdf " + scriptname # for a local bash
# cmd = "~/0_scripts/py2pdf " + scriptname # for a bash in directory in home
cmd = "py2pdf " + scriptname # alias

# code for console output demo
print_twice(logfile, "launch command : ", cmd) # for demo
print_twice(logfile, "pdf should be available soon") # for demo
# launch the report creation
# os.system(cmd) # to launch the bash file (local or directory in home)
subprocess.call(["/bin/bash", "-i", "-c", cmd]) # to launch the bash file (alias)

```

3 python code

3.1 Import modules

```

# import
import os # for py2pdf
import subprocess # for py2pdf with bash with alias
import matplotlib.pyplot as plt # for the demo
import numpy as np # for the demo

```

3.2 Functions

This is a demo file. In other applications, for a better readability, those specific functions might be included in a library to import in the above section.

```

def print_twice(log, *args, **kwargs):
    '''the double print function (to console and to log)'''
    print(*args, **kwargs)
    with open(log, "a") as f: # appends to file and closes it when finished
        print(file=f, *args, **kwargs)

def outputdir():

```

```

'''create a directory to save outputs'''
directory = "py2pdf_files"
if not os.path.exists(directory):
    # print("create directory")
    os.makedirs(directory)
else:
    pass
    # print("existing directory")

def clearlog(log):
    '''clear logfile'''
    file_to_delete = open(log, "w")
    file_to_delete.close()

```

3.3 Main code

3.3.1 Code section 1

```

# lines to be included for py2pdf export
# get this script file name without extension
scriptname = os.path.basename(__file__).split(".")[0]
logfile = "./py2pdf_files/log1.txt" # define the logfile
outputdir() # create the directory to store the output
clearlog(logfile) # clear the logfile (in case script is ran several times)

# code for console output demo
path = os.getcwd() # get the path of the current directory
print_twice(logfile, "Path of the current directory : " + path)
print_twice(logfile, "python script name : ", scriptname)
print_twice(logfile, "Hello world")
print_twice(logfile, "Logfile : ", logfile)

```

3.3.2 Code Output (section 1)

```

Path of the current directory : /home/christian/py2pdf
python script name : scriptpython2pdf
Hello world
Logfile : ./py2pdf_files/log1.txt

```

3.3.3 Code section 2

From matplotlib site : [Simple plot](#) This simple example plots the signal :

$$s = 1 + \sin(2\pi t)$$

```

# code for matplotlib output demo
# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)
fig, ax = plt.subplots()
ax.plot(t, s)
ax.set(
    xlabel="time (s)", ylabel="voltage (mV)", title="About as simple as it gets, folks"
)
ax.grid()
# this is the key line to store the matplotlib output
fig.savefig("./py2pdf_files/simple_plot.png")
plt.show()

```

3.3.4 Code Output (section 2)

See figure 1

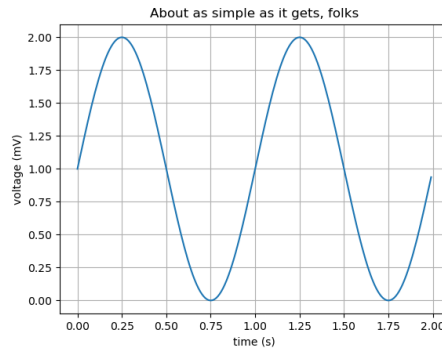


Figure 1: matplotlib output

3.3.5 Code last section

```
# lines to be included for py2pdf export
# create a second logfile
logfile = "./py2pdf_files/loglast.txt" # define the logfile
clearlog(logfile) # clear the logfile (in case script is ran several times)
# prepare the command to launch the report creation
# cmd = "./py2pdf " + scriptname # for a local bash
# cmd = "~/0_scripts/py2pdf " + scriptname # for a bash in directory in home
cmd = "py2pdf " + scriptname # alias

# code for console output demo
print_twice(logfile, "launch command : ", cmd) # for demo
print_twice(logfile, "pdf should be available soon") # for demo
# launch the report creation
# os.system(cmd) # to launch the bash file (local or directory in home)
subprocess.call(["/bin/bash", "-i", "-c", cmd]) # to launch the bash file (alias)

/!\WARNING : no output after this line can be recorded automatically

print("Hurra the pdf is created!")
```

3.3.6 Code output (last section)

```
launch command : py2pdf scriptpython2pdf
pdf should be available soon
```

3.3.7 Code of the bash file

```
py2pdf.sh

#!/bin/bash
echo "python to pdf with pandoc"
ext1=".py"
ext2=".md"
ext3=".pdf"
scriptfile=$1$ext1
mdcopy=$1$ext2
pdfout=$1$ext3
echo "copy to markdown"
# cp $scriptfile temp.md
echo "sed preprocessing"
tac $scriptfile > temp2.md
sed -i '/^# end of Python header, start of markdown py2pdf/, $d' temp2.md
tac temp2.md > temp.md
# rm temp2.md
sed -i '/^#~~ no export to pdf from here/, $d' temp.md
sed -i -e 's/#~~ //1' temp.md
```

```

echo "sed preprocessing PEP compatibility"
sed -i '/^# ~~ no export to pdf from here/, $d' temp.md
sed -i -e 's/^\\s*# ~~ //' temp.md
sed -i -e 's/^\\s*# ~~//' temp.md # added because black remove trailing space
echo "perl preprocessing"
perl -ne 's/^[\\[\\(\\.\\+\\?\\)\\]\\].*/'cat $1'/e;print' temp.md > $mdcopy
echo "pandoc to pdf"
pandoc -s -o $pdfout --filter pandoc-crossref --citeproc $mdcopy
echo "delete temp files"
# rm temp.md # temp.md never erase for debugging
# rm $mdcopy
echo "end of bash"

```

3.4 Additional output

Some pictures or snapshots might be added. It is a manual operation to store them.

```

Fichier  Édition  Affichage  Terminal  Onglets  Aide
Path of the current directory : /home/christian/pweave
python scriptpython2pdf.py
Directory content :
final.md
py2md.sh
parseMd.sh
python
DMLmaterial4.py
scriptpython2pdf.py
scriptpython2pdf.pdf
main.md
scriptpython2pdf
py2pdf_files
Logfile : ./py2pdf_files/log1.txt
launch command : ./py2md.sh scriptpython2pdf
pdf should be available soon
Hurra the pdf is created!
Press any key to continue.
## Additional output
#~~.Some pictures or snapshots might be added..It is a manu

```

Figure 2: terminal

3.5 Test of indented lines

When markdown lines are in a function, the comments are indented.