

---

# **Set of python scripts to work with GnuCash books Documentation**

***Release 0.1.5***

**sdementen**

November 28, 2016



<b>1</b>	<b>What's new</b>	<b>3</b>
1.1	In development . . . . .	3
<b>2</b>	<b>Documentation</b>	<b>5</b>
2.1	Report creation (Linux and Windows, python >=3.5) . . . . .	5
<b>3</b>	<b>piecash_utilities</b>	<b>11</b>
3.1	piecash_utilities package . . . . .	11
<b>4</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



**Release** 0.1.5

**Date** November 28, 2016

**Authors** sdementen

**Project page** <https://github.com/sdementen/gnucash-utilities>



---

## What's new

---

### 1.1 In development

- system to add python report to gnucash

Contents:





---

## Documentation

---

This project aims to provide a suite of scripts to work on GnuCash files stored in SQL (sqlite3 and Postgres, not tested in MySQL). It depends on [piecash](#) to read/write gnuCash books.

## 2.1 Report creation (Linux and Windows, python >=3.5)

### 2.1.1 Installation & use

You first need to install the gnuCash-utilities with:

```
$ pip install gnuCash-utilities
```

Once installed, you can add python reports to gnuCash by adding python files of the form 'report\_name-of-report.py' to your \$HOME/.gnuCash folder.

Everytime a python report is added or the signature of the report function is modified (change of report metadata, addition/change/removal of an option), you should run the gc\_report script:

```
For windows  
$ gc_report
```

```
For linux  
$ gc_report.py
```

This script generates the scheme wrapper around the python report (it has the same name as the python report file but with a .scm extension) and register the report in the \$HOME/.gnuCash/config.user file.

### 2.1.2 A simple report

The simplest report has the form of

```
from piecash_utilities.report import report, execute_report  
  
@report(  
    title="My simplest report",  
    name="piecash-simple-report",  
    menu_tip="This simple report ever",  
    options_default_section="general",  
)
```

```
def generate_report(
    book_url,
):
    return "<html><body>Hello world from python !</body></html>"

if __name__ == '__main__':
    execute_report(generate_report)
```

The core reporting logic is defined in the function ‘generate\_report’ that:

1. is decorated with the ‘report’ decorator
2. takes one argument ‘book\_url’ which is the book URL
3. takes optional arguments representing the report options
4. returns a string with html. This html is what gnuCash will display as the result of the report execution

**Warning:** The report system provided by the gnuCash-utilities has currently no way to identify the book that is running in gnuCash (this can be fixed if a guile function is able to return the gnuCash URI of the currently opened book). Hence, it uses a hack. It will look in the registry (for windows) or dconf (for linux) to find the last opened file and uses this as the “active gnuCash book” (ie the ‘book\_url’ argument of the ‘generate\_report’ function). This hack will fail a.o. if you work with multiple gnuCash books at the same time.

## 2.1.3 A report with options

If you want to define options for your report, you can do it with type annotations as in

```
from piecash_utilities.report import report, RangeOption, DateOption, StringOption, execute_report
```

```
@report(
    title="My simplest report with parameters",
    name="piecash-simple-report-parameters",
    menu_tip="A simple report with parameters",
    options_default_section="general",
)
def generate_report(
    book_url,
    a_number: RangeOption(
        section="main",
        sort_tag="a",
        documentation_string="This is a number",
        default_value=3),
    a_str: StringOption(
        section="main",
        sort_tag="c",
        documentation_string="This is a string",
        default_value="with a default value"),
    a_date: DateOption(
        section="main",
        sort_tag="d",
        documentation_string="This is a date",
        default_value="(lambda () (cons 'absolute (cons (current-time) 0)))"),
    another_number: RangeOption(
        section="main",
        sort_tag="b",
        documentation_string="This is a number",
```

```
        default_value=3)
):
    return """<html>
<body>
    Hello world from python !<br>
    Parameters received:<br>
    <ul>
        <li>a_number = {a_number}</li>
        <li>a_str = {a_str}</li>
        <li>a_date = {a_date}</li>
        <li>another_number = {another_number}</li>
    </ul>
</body>
</html>""".format(
    a_str=a_str,
    another_number=another_number,
    a_date=a_date,
    a_number=a_number,
)

if __name__ == '__main__':
    execute_report(generate_report)
```

Each option is an additional argument to the ‘generate\_report’ function with its type defined through python type annotations.

Options currently supported are:

- date with DateOption
- float with RangeOption
- str with StringOption

## 2.1.4 A report that access the book

Most of the report will want to access the gnuCash book. You can use piecash to open the book thanks to the ‘book\_url’ argument that the ‘generate\_report’ function gets automatically as illustrated in the following example

```
import piecash

from piecash_utilities.report import report, execute_report

@report(
    title="My simplest report with a book",
    name="piecash-simple-report-book",
    menu_tip="A simple report that opens a book",
    options_default_section="general",
)
def generate_report(
    book_url,
):
    with piecash.open_book(book_url, readonly=True, open_if_lock=True) as book:
        return """<html>
<body>
    Hello world from python !<br>
```

```
        Book : {book_url}<br>
        List of accounts : {accounts}
    </body>
</html>"".format(
    book_url=book_url,
    accounts=[acc.fullname for acc in book.accounts],
)

if __name__ == '__main__':
    execute_report(generate_report)
```

## 2.1.5 A full fledged example with jinja2 to generate the html

You can use the command ‘gc\_create\_report name-of-report’ (under windows) or ‘gc\_create\_report.py name-of-report’ (under linux) to create a set of files ‘report\_name-of-report.py’ and ‘report\_name-of-report.html’ that use the jinja2 templating logic to generate the report. For any moderately complex report, this is the suggested approach.

You can also generate a sample file automatically by executing:

```
For windows
$ gc_report_create name-of-report

For linux
$ gc_report_create.py name-of-report
```

## 2.1.6 Testing your report from the command line

You can test a report by just running the ‘report\_name-of-report.py’ python file and piping the options to it as:

```
$ cat inputs | python report_name-of-report.py
```

with inputs being a file like

```
a_number|3
a_str|with a default value
a_date|1479026587
another_number|3
```

The inputs should be in line with the options required by the report.

## 2.1.7 How does it work ?

The python report mechanism works as following:

- At report creation:
  1. user creates a report by writing a python script as \$HOME/.gnucash/report\_name.py
  2. users launches the gc\_report command that:
    - (a) generates a scheme wrapper as \$HOME/.gnucash/report\_name.scm
    - (b) adds the report to the file \$HOME/.gnucash/config.user to have it loaded at each start of gnucash
- At runtime:

1. gnucash starts, loads \$HOME/.gnucash/config.user and registers the report declared in the .scm files
2. user launches a python report
3. the scheme wrapper is called and:
  - (a) it starts a python subprocess “python report\_name.py”
  - (b) it retrieves and serialises each report option in the format “option\_nameoption\_value” and pipes it to the standard input of the python subprocess
  - (c) the python subprocesses:
    - i. deserialises the options => option arguments
    - ii. retrieves the “last open gnucash book” => book\_url argument
    - iii. calls the generate\_report function with the arguments which returns an HTML string
    - iv. prints the HTML string to the standard output
  - (d) it retrieves the standard output of the python subprocess as the HTML output of the report

The complete api documentation (apidoc) :



---

## piecash\_utilities

---

### 3.1 piecash\_utilities package

#### 3.1.1 Subpackages

**piecash\_utilities.report package**

**Submodules**

**piecash\_utilities.report.options module**

```
class piecash_utilities.report.options.Option (type, section, sort_tag, documentation_string,  
                                              default_value, name=None)
```

Bases: `builtins.object`

An option to be used in a report

**type**

*str* – the type of the option of the form ‘gnc:make-number-range-option’

**section**

*str* – the section/tab where the option should appear in the option dialog

**sort\_tag**

*str* – a string defining the sort order in the tab

**documentation\_string**

*str* – the doc string of the option

**default\_value**

*str* – the default value of the option

**name**

*str* – the name of the variable

**render\_scheme()**

**render\_serialise()**

**parse(value)**

```
class piecash_utilities.report.options.DateOption (is_datetime=False, **kwargs)
```

Bases: `piecash_utilities.report.options.Option`

```
    render_serialise()
    parse(value)
    render_scheme()
class piecash_utilities.report.options.RangeOption(lower=0, upper=10000, decimals=2,
                                                    step_size=0.01, **kwargs)
    Bases: piecash_utilities.report.options.Option
    render_scheme()
class piecash_utilities.report.options.StringOption(**kwargs)
    Bases: piecash_utilities.report.options.Option
    render_scheme()
    parse(value)
```

#### piecash\_utilities.report.report module

```
class piecash_utilities.report.report.Report(name, title, menu_tip, options, op-
                                             tions_default_section,
                                             python_script=None,
                                             function=None)
    Bases: builtins.object
    name = ''
    title = ''
    menu_tip = ''
    options = None
    function = None
    python_script = None
    guid
    generate_scm()
piecash_utilities.report.report.retrieve_template_scm()
piecash_utilities.report.report.generate_sample_report_python()
piecash_utilities.report.report.generate_sample_report_html()
piecash_utilities.report.report.report(options_default_section, title, name, menu_tip)
piecash_utilities.report.report.execute_report(generate_report)
```



## Module contents

`piecash_utilities.report_example` package

## Submodules

`piecash_utilities.report_example.report_simplest` module

`piecash_utilities.report_example.report_simplest_book` module

`piecash_utilities.report_example.report_simplest_parameters` module

## Module contents

### 3.1.2 Submodules

#### 3.1.3 `piecash_utilities.config` module

`piecash_utilities.config.get_latest_file()`

`piecash_utilities.config.get_user_config_path()`

`piecash_utilities.config.update_config_user` (*lines, separator=';; lines automatically added\n;; everything below this line will be scraped'*)

#### 3.1.4 `piecash_utilities.metadata` module

Project metadata

Information describing the project.

#### 3.1.5 Module contents



---

## Indices and tables

---

- *genindex*
- *modindex*
- *search*



## p

- `piecash_utilities`, [13](#)
- `piecash_utilities.config`, [13](#)
- `piecash_utilities.metadata`, [13](#)
- `piecash_utilities.report`, [13](#)
- `piecash_utilities.report.options`, [11](#)
- `piecash_utilities.report.report`, [12](#)
- `piecash_utilities.report_example`, [13](#)
- `piecash_utilities.report_example.report_simplest`,  
[13](#)
- `piecash_utilities.report_example.report_simplest_book`,  
[13](#)
- `piecash_utilities.report_example.report_simplest_parameters`,  
[13](#)



## D

DateOption (class in piecash\_utilities.report.options), 11  
default\_value (piecash\_utilities.report.options.Option attribute), 11  
documentation\_string (piecash\_utilities.report.options.Option attribute), 11

## E

execute\_report() (in module piecash\_utilities.report.report), 12

## F

function (piecash\_utilities.report.report.Report attribute), 12

## G

generate\_sample\_report\_html() (in module piecash\_utilities.report.report), 12  
generate\_sample\_report\_python() (in module piecash\_utilities.report.report), 12  
generate\_scm() (piecash\_utilities.report.report.Report method), 12  
get\_latest\_file() (in module piecash\_utilities.config), 13  
get\_user\_config\_path() (in module piecash\_utilities.config), 13  
guid (piecash\_utilities.report.report.Report attribute), 12

## M

menu\_tip (piecash\_utilities.report.report.Report attribute), 12

## N

name (piecash\_utilities.report.options.Option attribute), 11  
name (piecash\_utilities.report.report.Report attribute), 12

## O

Option (class in piecash\_utilities.report.options), 11  
options (piecash\_utilities.report.report.Report attribute), 12

## P

parse() (piecash\_utilities.report.options.DateOption method), 12  
parse() (piecash\_utilities.report.options.Option method), 11  
parse() (piecash\_utilities.report.options.StringOption method), 12  
piecash\_utilities (module), 13  
piecash\_utilities.config (module), 13  
piecash\_utilities.metadata (module), 13  
piecash\_utilities.report (module), 13  
piecash\_utilities.report.options (module), 11  
piecash\_utilities.report.report (module), 12  
piecash\_utilities.report\_example (module), 13  
piecash\_utilities.report\_example.report\_simplest (module), 13  
piecash\_utilities.report\_example.report\_simplest\_book (module), 13  
piecash\_utilities.report\_example.report\_simplest\_parameters (module), 13  
python\_script (piecash\_utilities.report.report.Report attribute), 12

## R

RangeOption (class in piecash\_utilities.report.options), 12  
render\_scheme() (piecash\_utilities.report.options.DateOption method), 12  
render\_scheme() (piecash\_utilities.report.options.Option method), 11  
render\_scheme() (piecash\_utilities.report.options.RangeOption method), 12  
render\_scheme() (piecash\_utilities.report.options.StringOption method), 12  
render\_serialise() (piecash\_utilities.report.options.DateOption method), 11  
render\_serialise() (piecash\_utilities.report.options.Option method), 11  
Report (class in piecash\_utilities.report.report), 12  
report() (in module piecash\_utilities.report.report), 12

`retrieve_template_scm()` (in module  
`piecash_utilities.report.report`), [12](#)

## S

`section` (`piecash_utilities.report.options.Option` attribute),  
[11](#)

`sort_tag` (`piecash_utilities.report.options.Option` at-  
tribute), [11](#)

`StringOption` (class in `piecash_utilities.report.options`), [12](#)

## T

`title` (`piecash_utilities.report.report.Report` attribute), [12](#)

`type` (`piecash_utilities.report.options.Option` attribute), [11](#)

## U

`update_config_user()` (in module  
`piecash_utilities.config`), [13](#)