pyRestTable Documentation

Release 2017.2.0+0.ga7f8967.dirty<2017-02-15T10:04:33-0600>

Pete R. Jemian

February 15, 2017

Contents

1 Usage **3**

2 Installation **5**

3 Examples **7**
- 3.1 Interactive example with *ipython* . . . 7
- 3.2 *simple* (default) . . . 8
- 3.3 *plain* . . . 9
- 3.4 *grid* (*complex*) . . . 9
- 3.5 *list-table* . . . 10
- 3.6 Complicated example . . . 11
- 3.7 Example using XML source data from a URL . . . 13

4 pyRestTable **17**
- 4.1 source code documentation . . . 17

5 Change History **19**
- 5.1 Production . . . 19

6 License **21**

7 Features **23**

8 Indices and tables **25**

Python Module Index **27**

Format a nice table in reST (reStructuredText) from Python.

Each cell may have multiple lines, separated by a newline. The content of each cell will be rendered as str(cell). At present, *pyRestTable* only supports tables with content that does not span any cells (no rowspans or columnspans).

author Pete R. Jemian

email prjemian@gmail.com

docs http://pyRestTable.readthedocs.io

URL https://github.com/prjemian/pyRestTable

PyPI https://pypi.python.org/pypi/pyRestTable/

TODO list https://github.com/prjemian/pyRestTable/issues

build

coverage

version 2017.2.0+0.ga7f8967.dirty

release 2017.2.0+0.ga7f8967.dirty<2017-02-15T10:04:33-0600>

published February 15, 2017

CHAPTER 1

Usage

pyRestTable provides support for writing tables in the format of reStructured Text [1] from Python programs. (It provides no command-line or GUI program itself – no **"entry points"**; it should be used within a Python program.)

- Import the pyRestTable package
- Create the `Table` instance
- Set the list of column labels (either `labels.append()` or `addLabel()`)
- Append the list of column cells for each row (either `rows.append([])` or `addRow()`)
- Render the table with `reST()` (default table format is `simple`)

Examples are provided to demonstrate usage.

[1] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html

CHAPTER 2

Installation

available for installation from PyPI via standard installers for Python 2.7 or Python 3.0+:

```
$ pip install pyRestTable
```

or:

```
$ easy_install -U pyRestTable
```

or download the tarball from GitHub and use setuptools to install:

```
$ tar xzf pyRestTable-2015-1111-1.tar.gz
$ cd pyRestTable-2015-1111-1
$ python setup.py install
```

CHAPTER 3

Examples

Examples are provided to demonstrate usage.

3.1 Interactive example with *ipython*

```
In [1]: import pyRestTable

In [2]: pyRestTable.__long_description__

Out[2]: 'Format a nice table in reST (reStructuredText ) from Python'

In [3]: pyRestTable.__version__

Out[3]: '2015-1111-1'

In [4]: t = pyRestTable.Table()

In [5]: t.labels = ['x', 'y']

In [6]: t.rows.append([1,2])

In [7]: print(t.reST())

= =
x y
= =
1 2
= =
```

which displays as:

x	y
1	2

The same table may be rendered in the *grid* reST format:

```
In [8]: print(t.reST(fmt='grid'))

+---+---+
| x | y |
+===+===+
| 1 | 2 |
+---+---+
```

which displays as:

x	y
1	2

The same table may be rendered in the *list-table* reST format:

```
In [9]: print(t.reST(fmt='list-table'))

.. list-table::
   :header-rows: 1
   :widths: 1 1

   * - x
     - y
   * - 1
     - 2
```

which displays as:

x	y
1	2

3.2 *simple* (default)

see http://docutils.sourceforge.net/docs/ref/rst/directives.html#tables

These python commands:

```
import pyRestTable
t = pyRestTable.Table()
t.labels = ('one', 'two', 'three' )
t.rows.append( ['1,1', '1,2', '1,3',] )
t.rows.append( ['2,1', '2,2', '2,3',] )
t.rows.append( ['3,1', '3,2', '3,3',] )
t.rows.append( ['4,1', '4,2', '4,3',] )
print(t.reST())
```

build this table source code:

```
=== === =====
one two three
=== === =====
1,1 1,2 1,3
2,1 2,2 2,3
3,1 3,2 3,3
4,1 4,2 4,3
=== === =====
```

which is rendered as:

one	two	three
1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3
4,1	4,2	4,3

3.3 *plain*

These python commands:

```
import pyRestTable
t = pyRestTable.Table()
t.labels = ('one', 'two', 'three' )
t.rows.append( ['1,1', '1,2', '1,3',] )
t.rows.append( ['2,1', '2,2', '2,3',] )
t.rows.append( ['3,1', '3,2', '3,3',] )
t.rows.append( ['4,1', '4,2', '4,3',] )
print(t.reST(fmt='plain'))
```

build this table source code:

```
one two three
1,1 1,2 1,3
2,1 2,2 2,3
3,1 3,2 3,3
4,1 4,2 4,3
```

The *plain* format is useful when generating very compact tables as text.

3.4 *grid* (*complex*)

see http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#grid-tables

These python commands:

```
import pyRestTable
t = pyRestTable.Table()
t.labels = ('one', 'two', 'three' )
t.rows.append( ['1,1', '1,2', '1,3',] )
t.rows.append( ['2,1', '2,2', '2,3',] )
t.rows.append( ['3,1', '3,2', '3,3',] )
t.rows.append( ['4,1', '4,2', '4,3',] )
print(t.reST(fmt='grid'))
```

build this table in reST source code:

```
+-----+-----+-------+
| one | two | three |
+=====+=====+=======+
| 1,1 | 1,2 | 1,3   |
+-----+-----+-------+
| 2,1 | 2,2 | 2,3   |
+-----+-----+-------+
| 3,1 | 3,2 | 3,3   |
+-----+-----+-------+
| 4,1 | 4,2 | 4,3   |
+-----+-----+-------+
```

which is rendered as:

one	two	three
1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3
4,1	4,2	4,3

Note: API Changes

- version 2015.1111.01

  In versions previous to 2015.1111.01, the `complex` output table format was supported:

```
print t.reST(fmt='complex')
```

  The `complex` output format has been aliased `grid` to be consistent with the docutils [1] documentation:

```
print(t.reST(fmt='grid'))
```

  The two commands are identical (except the latter is upgraded for compatibility with Python v3). To preserve existing code, no plans are made to deprecate the `complex` name.

3.5 *list-table*

see http://docutils.sourceforge.net/docs/ref/rst/directives.html#list-table

These python commands:

```
import pyRestTable
t = pyRestTable.Table()
t.labels = ('one', 'two', 'three' )
t.rows.append( ['1,1', '1,2', '1,3',] )
t.rows.append( ['2,1', '2,2', '2,3',] )
t.rows.append( ['3,1', '3,2', '3,3',] )
t.rows.append( ['4,1', '4,2', '4,3',] )
print(t.reST(fmt='list-table'))
```

build this table source code:

```
.. list-table::
  :header-rows: 1
  :widths: 3 3 5

  * - one
    - two
    - three
  * - 1,1
    - 1,2
    - 1,3
  * - 2,1
    - 2,2
    - 2,3
  * - 3,1
```

[1] docutils: http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html

```
    - 3,2
    - 3,3
  * - 4,1
    - 4,2
    - 4,3
```

which is rendered as:

one	two	three
1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3
4,1	4,2	4,3

3.6 Complicated example

These python commands setup the table:

```
import pyRestTable
t = pyRestTable.Table()
t.addLabel('Name\nand\nAttributes')
t.addLabel('Type')
t.addLabel('Units')
t.addLabel('Description\n(and Occurrences)')
t.addRow( ['one,\ntwo', "buckle my", "shoe.\n\n\nthree,\nfour", "..."] )
t.addRow( ['class', 'NX_FLOAT', '', None, ] )
t.addRow( range(0,4) )
t.addRow( [None, {'a': 1, 'b': 'dreamy'}, 1.234, range(3)] )
t.setLongtable()
t.setTabularColumns(True, 'l L c r'.split())
```

Here, we assert more control over the table format using `setLongtable()` and `setTabularColumns()` configuration options.

3.6.1 Format: *print(t.reST(fmt='simple'))*

this reST code:

```
.. tabularcolumns:: |l|L|c|r|
    :longtable:

========== ====================== ====== ================
Name       Type                   Units  Description
and                                      (and Occurrences)
Attributes
========== ====================== ====== ================
one,       buckle my              shoe.  ...
two

                                  three,
                                  four
class      NX_FLOAT                      None
0          1                      2      3
None       {'a': 1, 'b': 'dreamy'} 1.234 [0, 1, 2]
========== ====================== ====== ================
```

is rendered as:

Name	Type	Units	Description
and			(and Occurrences)
Attributes			
one,	buckle my	shoe.	...
two		three, four	
class	NX_FLOAT		None
0	1	2	3
None	{'a': 1, 'b': 'dreamy'}	1.234	[0, 1, 2]

3.6.2 Format: *print(t.reST(fmt='grid'))*

this reST code:

```
.. tabularcolumns:: |l|L|c|r|
    :longtable:

+------------+-------------------------+--------+--------------------+
| Name       | Type                    | Units  | Description        |
| and        |                         |        | (and Occurrences)  |
| Attributes |                         |        |                    |
+============+=========================+========+====================+
| one,       | buckle my               | shoe.  | ...                |
| two        |                         |        |                    |
|            |                         |        |                    |
|            |                         | three, |                    |
|            |                         | four   |                    |
+------------+-------------------------+--------+--------------------+
| class      | NX_FLOAT                |        | None               |
+------------+-------------------------+--------+--------------------+
| 0          | 1                       | 2      | 3                  |
+------------+-------------------------+--------+--------------------+
| None       | {'a': 1, 'b': 'dreamy'} | 1.234  | [0, 1, 2]          |
+------------+-------------------------+--------+--------------------+
```

is rendered as:

Name and Attributes	Type	Units	Description (and Occurrences)
one, two	buckle my	shoe. three, four	...
class	NX_FLOAT		None
0	1	2	3
None	{'a': 1, 'b': 'dreamy'}	1.234	[0, 1, 2]

3.6.3 Format: *print(t.reST(fmt='list-table'))*

this reST code:

```
.. list-table::
   :header-rows: 1
   :widths: 10 23 6 17

   * - Name
       and
       Attributes
     - Type
```

```
      - Units
      - Description
        (and Occurrences)
    * - one,
        two
      - buckle my
      - shoe.


        three,
        four
      - ...
    * - class
      - NX_FLOAT
      -
      -
    * - 0
      - 1
      - 2
      - 3
    * - None
      - {'a': 1, 'b': 'dreamy'}
      - 1.234
      - [0, 1, 2]
```

is rendered as:

Name and Attributes	Type	Units	Description (and Occurrences)
one, two	buckle my	shoe. three, four	...
class	NX_FLOAT		
0	1	2	3
None	{'a': 1, 'b': 'dreamy'}	1.234	[0, 1, 2]

3.7 Example using XML source data from a URL

Another example (*cansas.py* in the source distribution) shows how content can be scraped from a URL that provides XML (using the *lxml* package) and written as a reST table. This particular XML uses a namespace which we setup in the variable `nsmap`:

```
#!/usr/bin/env python

import io
import sys
from lxml import etree
try:
    # python 3
    from urllib.request import urlopen
except ImportError as _exc:
    # python 2
    from urllib2 import urlopen
sys.path.insert(0, '..')
from pyRestTable import Table

SVN_BASE_URL = 'http://www.cansas.org/svn/1dwg/trunk'
```

```
GITHUB_BASE_URL = 'https://raw.githubusercontent.com/canSAS-org/1dwg/master'
CANSAS_URL = '/'.join((GITHUB_BASE_URL, 'examples/cs_af1410.xml'))


def main():
    nsmap = dict(cs='urn:cansas1d:1.1')

    r = urlopen(CANSAS_URL).read().decode("utf-8")
    doc = etree.parse(io.StringIO(r))

    node_list = doc.xpath('//cs:SASentry', namespaces=nsmap)
    t = Table()
    t.labels = ['SASentry', 'description', 'measurements']
    for node in node_list:
        s_name, count = '', ''
        subnode = node.find('cs:Title', namespaces=nsmap)
        if subnode is not None:
            s = etree.tostring(subnode, method="text")
            s_name = node.attrib['name']
            count = len(node.xpath('cs:SASdata', namespaces=nsmap))
        title = s.strip().decode()
        t.rows += [[s_name, title, count]]

    return t


if __name__ == '__main__':
    table = main()
    # use "complex" since s_name might be empty string
    print(table.reST(fmt='complex'))
```

The output from this code:

```
10 SASentry elements in http://www.cansas.org/svn/1dwg/trunk/examples/cs_af1410.xml

+-----------+--------------------------------------+--------------+
| entry     | description                          | measurements |
+===========+======================================+==============+
| AF1410:10 | AF1410-10 (AF1410 steel aged 10 h)   | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:8h | AF1410-8h (AF1410 steel aged 8 h)    | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:qu | AF1410-qu (AF1410 steel aged 0.25 h) | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:cc | AF1410-cc (AF1410 steel aged 100 h)  | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:2h | AF1410-2h (AF1410 steel aged 2 h)    | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:50 | AF1410-50 (AF1410 steel aged 50 h)   | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:20 | AF1410-20 (AF1410 steel aged 20 h)   | 1            |
+-----------+--------------------------------------+--------------+
| AF1410:5h | AF1410-5h (AF1410 steel aged 5 h)    | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:1h | AF1410-1h (AF1410 steel aged 1 h)    | 2            |
+-----------+--------------------------------------+--------------+
| AF1410:hf | AF1410-hf (AF1410 steel aged 0.5 h)  | 2            |
+-----------+--------------------------------------+--------------+
```

The resulting table is shown:

10 SASentry elements in http://www.cansas.org/svn/1dwg/trunk/examples/cs_af1410.xml

entry	description	measurements
AF1410:10	AF1410-10 (AF1410 steel aged 10 h)	2
AF1410:8h	AF1410-8h (AF1410 steel aged 8 h)	2
AF1410:qu	AF1410-qu (AF1410 steel aged 0.25 h)	2
AF1410:cc	AF1410-cc (AF1410 steel aged 100 h)	2
AF1410:2h	AF1410-2h (AF1410 steel aged 2 h)	2
AF1410:50	AF1410-50 (AF1410 steel aged 50 h)	2
AF1410:20	AF1410-20 (AF1410 steel aged 20 h)	1
AF1410:5h	AF1410-5h (AF1410 steel aged 5 h)	2
AF1410:1h	AF1410-1h (AF1410 steel aged 1 h)	2
AF1410:hf	AF1410-hf (AF1410 steel aged 0.5 h)	2

CHAPTER 4

pyRestTable

author Pete R. Jemian

version 2017.2.0+0.ga7f8967.dirty

release 2017.2.0+0.ga7f8967.dirty<2017-02-15T10:04:33-0600>

published February 15, 2017

4.1 source code documentation

Format a nice table in reST (restructured text)

User Interface	Description
Table	Construct a table in reST
addLabel()	add label for one additional column
addRow()	add list of items for one additional row
setLongtable()	set *longtable* attribute
setTabularColumns()	set *use_tabular_columns* & *alignment* attributes
reST()	render the table in reST format

class pyRestTable.rest_table.**Table**

Construct a table in reST (no row or column spans).

Parameters

- **use_tabular_columns** (*bool*) – if True, embed table in Sphinx *'.. tabularcolumns:: |%s|' % alignment'* role
- **alignment** (*[str]*) – with *use_tabular_columns*, each list item is a column format string, as specified by LaTeX *tabulary* package format: http://sphinx-doc.org/markup/misc.html?highlight=tabularcolumns#directive-tabularcolumns
- **longtable** (*bool*) – with *use_tabular_columns*, if True, add Sphinx *:longtable:* directive

addLabel(*text*)

add label for one additional column

Parameters **text** (*str*) – column label text

Return int number of labels

addRow(*list_of_items*)

add list of items for one additional row

Parameters **list_of_items** (*[obj]*) – list of items for one complete row

Return int number of rows

find_widths()
measure the maximum width of each column, considering possible line breaks in each cell

grid_table(*indentation=''*)
render the table in *grid* reST format

list_table(*indentation=''*)
render the table in *list-table* reST format:

See http://docutils.sourceforge.net/docs/ref/rst/directives.html

Table 4.1: Frozen Delights!

Treat	Quantity	Description
Albatross	2.99	On a stick!
Crunchy Frog	1.49	If we took the bones out, it wouldn't be crunchy, now would it?
Gannet Ripple	1.99	On a stick!

plain_table(*indentation=''*)
render the table in *plain* reST format

reST(*indentation=''*, *fmt='simple'*)
render the table in reST format

setLongtable(*state=True*)
set *longtable* attribute

Parameters **longtable** (*bool*) – True | False

setTabularColumns(*state=True*, *column_spec=[]*)
set *use_tabular_columns* & *alignment* attributes

Parameters

- **state** (*bool*) – True | False
- **column_spec** (*[str]*) – list of column specifications

simple_table(*indentation=''*)
render the table in *simple* reST format

pyRestTable.rest_table.**example_basic**()
basic example table

pyRestTable.rest_table.**example_complicated**()
complicated example table

pyRestTable.rest_table.**example_minimal**()
minimal example table

CHAPTER 5

Change History

5.1 Production

2017.2.0

- #9 provide default rendering: *t = Table(); ...; str(t)*
- #8 add a plain table output

2016.1024.0 unit tests, repaired python 3 support

2016.1003.1 support python 3 AND python 2

2015.1115.0 add support methods, such as *addLabel* and *addRow*

2015.1111.01 support output as ReST *list-table* directive

2014.0710.01 provide docs at http://pyRestTable.readthedocs.org

2014.0430.01 release after forking from previous home

CHAPTER 6

License

```
Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and

Section 1  Definitions.

    Adapted Material means material subject to Copyright and Similar Rights that is derived from or b
    Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contri
    Copyright and Similar Rights means copyright and/or similar rights closely related to copyright i
    Effective Technological Measures means those measures that, in the absence of proper authority, m
    Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitation
    Licensed Material means the artistic or literary work, database, or other material to which the l
    Licensed Rights means the rights granted to You subject to the terms and conditions of this Publi
    Licensor means the individual(s) or entity(ies) granting rights under this Public License.
    Share means to provide material to the public by any means or process that requires permission un
    Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of
    You means the individual or entity exercising the Licensed Rights under this Public License. Your

Section 2  Scope.

    License grant.
        Subject to the terms and conditions of this Public License, the Licensor hereby grants You a
            reproduce and Share the Licensed Material, in whole or in part; and
            produce, reproduce, and Share Adapted Material.
        Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations appl
        Term. The term of this Public License is specified in Section 6(a).
        Media and formats; technical modifications allowed. The Licensor authorizes You to exercise t
        Downstream recipients.
            Offer from the Licensor  Licensed Material. Every recipient of the Licensed Material auto
            No downstream restrictions. You may not offer or impose any additional or different terms
        No endorsement. Nothing in this Public License constitutes or may be construed as permission

    Other rights.
        Moral rights, such as the right of integrity, are not licensed under this Public License, nor
        Patent and trademark rights are not licensed under this Public License.
        To the extent possible, the Licensor waives any right to collect royalties from You for the e

Section 3  License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

    Attribution.
```

```
        If You Share the Licensed Material (including in modified form), You must:
            retain the following if it is supplied by the Licensor with the Licensed Material:
                identification of the creator(s) of the Licensed Material and any others designated t
                a copyright notice;
                a notice that refers to this Public License;
                a notice that refers to the disclaimer of warranties;
                a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
            indicate if You modified the Licensed Material and retain an indication of any previous m
            indicate the Licensed Material is licensed under this Public License, and include the tex
        You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the mediu
        If requested by the Licensor, You must remove any of the information required by Section 3(a)
        If You Share Adapted Material You produce, the Adapter's License You apply must not prevent r

Section 4  Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed

    for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, an
    if You include all or a substantial portion of the database contents in a database in which You h
    You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under th

Section 5  Disclaimer of Warranties and Limitation of Liability.

    Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offe
    To the extent possible, in no event will the Licensor be liable to You on any legal theory (inclu

    The disclaimer of warranties and limitation of liability provided above shall be interpreted in a

Section 6  Term and Termination.

    This Public License applies for the term of the Copyright and Similar Rights licensed here. Howev

    Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:
        automatically as of the date the violation is cured, provided it is cured within 30 days of Y
        upon express reinstatement by the Licensor.
    For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to
    For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate term
    Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7  Other Terms and Conditions.

    The Licensor shall not be bound by any additional or different terms or conditions communicated b
    Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein

Section 8  Interpretation.

    For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce
    To the extent possible, if any provision of this Public License is deemed unenforceable, it shall
    No term or condition of this Public License will be waived and no failure to comply consented to
    Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver
```

CHAPTER 7

Features

- create *simple*, *plain*, *grid* (also known as *complex*), and *list-table* reST formatted tables
- defines table cells through Python lists, row-by-row
- use with Python 2 or Python 3

  see http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#tables

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

p

pyRestTable.rest_table, 17

A

addLabel() (pyRestTable.rest_table.Table method), 17
addRow() (pyRestTable.rest_table.Table method), 17

E

example_basic() (in module pyRestTable.rest_table), 18
example_complicated() (in module pyRestTable.rest_table), 18
example_minimal() (in module pyRestTable.rest_table), 18

F

find_widths() (pyRestTable.rest_table.Table method), 18

G

grid_table() (pyRestTable.rest_table.Table method), 18

L

list_table() (pyRestTable.rest_table.Table method), 18

P

plain_table() (pyRestTable.rest_table.Table method), 18
pyRestTable.rest_table (module), 17

R

reST() (pyRestTable.rest_table.Table method), 18

S

setLongtable() (pyRestTable.rest_table.Table method), 18
setTabularColumns() (pyRestTable.rest_table.Table method), 18
simple_table() (pyRestTable.rest_table.Table method), 18

T

Table (class in pyRestTable.rest_table), 17