sudokutools Documentation

Release 0.1

Maik Messerschmidt

May 01, 2018

Contents

Yet another python sudoku library.

CHAPTER 1

README

You can find a short introduction on the project site.

CHAPTER 2

Contents

2.1 The sudokutools library

Yet another python sudoku library.

sudokutools is a collection of functions and classes, which enable you to read, create, analyze, solve and print sudokus.

Package modules:

- sudokutools.generate: Create new sudokus.
- sudokutools.solve: Low-level solving and checking of sudokus.
- sudokutools.sudoku: Parse, print and compare sudokus.

2.1.1 `sudokutools.sudoku` - Parsing, printing and comparing

Parse, print and compare sudokus.

Classes defined here:

- Sudoku: Represents a sudoku.

Functions defined here:

- column_of(): Returns all coordinates in the column of a given field.
- row_of(): Returns all coordinates in the row of a given field.
- square_of(): Returns all coordinates in the square of a given field.
- surrounding_of(): Returns all surrounding coordinates of a given field.

class sudokutools.sudoku.**Sudoku**

Bases: object

Represents a sudoku.

This class provides methods to read, write, copy and display data to and from a sudoku as well as compare one sudoku with another. It stores 9x9 fields and the numbers and candidates within these fields.

Coordinates for row and column access are values from 0 to 8 (including). Using other values will most likely raise an IndexError (even though negative numbers are supported, they should not be used).

Overview:

Data access (read, write):

- __getitem__()
- __setitem__()
- get_candidates()
- set_candidates()
- remove_candidates()

Copying:

- copy()

Comparing:

- empty()
- __len__()
- __eq__()
- diff()

Printing:

- __str__()
- encode()

Parsing:

- decode()

__eq__(*other*)
Return if other is equal in all fields.

Parameters **other** (Sudoku) – Most likely another *Sudoku* instance, but any object, which can be accessed by other[row, col] is valid.

Returns

True, if all fields are equal and false if not or other is an incompatible type.

Return type bool

__getitem__(*key*)
Return the number in the field referenced by key.

Parameters **key** (*int, int*) – row and column of the requested field. Must be in range(0, 9).

Returns The number in the given field, 0 representing an empty field.

Return type int

Raises IndexError – if the given coordinates are not valid.

__len__()
Return the number of non-empty fields in this sudoku.

Returns The number of non-empty fields within this sudoku.

Return type int

__setitem__(*key*, *value*)
Set the number in the field referenced by key to value.

Parameters

- **key** (*int, int*) – row and column of the requested field. Must be in range(0, 9).
- **value** (*int*) – The number to set the field to, 0 representing an empty field.

Raises IndexError – if the given coordinates are not valid.

__str__()
Return sudoku as a human-readable string.

Returns String representing the sudoku.

Return type str

copy(*include_candidates=False*)
Returns a copy of this sudoku.

Parameters **include_candidates** (*bool*) – Whether to copy candidates as well.

Returns The new sudoku instance.

Return type *Sudoku*

classmethod decode(*s*)
Create a new sudoku from the string s.

Parameters **s** (*str*) – A string representing the sudoku (see below).

Returns The newly created *Sudoku* instance.

Return type *Sudoku*

Examples for s: 000030000005009602008004013020060000703040106000080090210300800306800700000020000
000030000005009602008004013020060000703040106000080090210300800306800700000020000|124,235

Whitespace is ignored while parsing the string, so you can place newlines for better readability.

Each number represents the value of a column. If a row is full, we continue in the next one. So the sudoku above looks like this:

```
      |   3   |
    5 |     9 | 6   2
    8 |     4 |   1 3
------+-------+------
  2   |   6   |
7   3 |   4   | 1   6
      |   8   |   9
------+-------+------
2 1   | 3     | 8
3   6 | 8     | 7
      |   2   |
```

The second string additionally defines candidates. Each set of candidates is separated by ‘,’ so the string above defines the candidates for (0, 0) to be 1, 2 and 4 and for (0, 1) to be 2, 3 and 5

This is the default format, which encode() uses and no other format is supported right now.

diff(*other*)
Iterate through coordinates with different values in other.

Compares each field in other to the corresponding field in self and yield the coordinates, if the number within is different.

Parameters **other** (Sudoku) – Most likely another *Sudoku* instance, but any object, which can be accessed by other[row, col] is valid.

Yields *(int, int)* – row and column of the next different field.

empty()
Iterate through the coordinates of all empty fields.

Yields *(int, int)* – row and column of the next empty field.

encode(*row_sep=”*, *col_sep=”*, *include_candidates=False*)
Return sudoku as a (machine-readable) string.

This method is mainly provided to output sudokus in a machine-readable format, but can be used for creating nicely looking representations as well.

Parameters

- **row_sep** (*str*) – Separator between rows.
- **col_sep** (*str*) – Separator between columns.
- **include_candidates** –

Returns String representing this sudoku.

Return type (str)

For examples of default output string see decode().

get_candidates(*row*, *col*)
Return the candidates of the field at (row, col).

Parameters

- **row** (*int*) – The row of the field.
- **col** (*int*) – The column of the field.

Returns The candidates at (row, col).

Return type frozenset

remove_candidates(*row*, *col*, **candidates*)
Remove the given candidates in the field at (row, col).

Ignores candidates, which are not present in the field.

Parameters

- **row** (*int*) – The row of the field.
- **col** (*int*) – The column of the field.
- **candidates** (*iterable*) – The candidates to remove.

set_candidates(*row*, *col*, *value*)
Set the candidates of the field at (row, col) to value.

Parameters

- **row** (*int*) – The row of the field.

- **col** (*int*) – The column of the field.
- **value** (*iterable*) – The candidates to set the field to.

sudokutools.sudoku.**_quad_without_row_and_column_of**(*row*, *col*)
Return some coordinates in the square of (col, row) as a list.

The coordinates in the same row and column are removed.

This is an internal function and should not be used outside of the sudoku module.

sudokutools.sudoku.**column_of**(*row*, *col*, *include=True*)
Return all coordinates in the column of (col, row) as a list.

Parameters

- **row** (*int*) – The row of the field.
- **col** (*int*) – The column of the field.
- **include** (*bool*) – Whether or not to include (row, col).

Returns

list of pairs (row, column) of all fields in the same column.

Return type list of (int, int)

sudokutools.sudoku.**row_of**(*row*, *col*, *include=True*)
Return all coordinates in the row of (col, row) as a list.

Parameters

- **row** (*int*) – The row of the field.
- **col** (*int*) – The column of the field.
- **include** (*bool*) – Whether or not to include (row, col).

Returns

list of pairs (row, column) of all fields in the same row.

Return type list of (int, int)

sudokutools.sudoku.**square_of**(*row*, *col*, *include=True*)
Return all coordinates in the square of (col, row) as a list.

Parameters

- **row** (*int*) – The row of the field.
- **col** (*int*) – The column of the field.
- **include** (*bool*) – Whether or not to include (row, col).

Returns

list of pairs (row, column) of all fields in the same square.

Return type list of (int, int)

sudokutools.sudoku.**surrounding_of**(*row*, *col*, *include=True*)
Return all surrounding coordinates of (col, row) as a list.

Parameters

- **row** (*int*) – The row of the field.
- **col** (*int*) – The column of the field.

- **include** (*bool*) – Whether or not to include (row, col).

Returns

list of pairs (row, column) of all fields in the same column, row or square.

Return type list of (int, int)

2.1.2 sudokutools.solve - Solving and checking

Low-level solving and checking of sudokus.

Functions defined here:

- bruteforce(): Solves a sudoku using brute force.
- calc_candidates(): Calculates candidates of a field in a sudoku.
- init_candidates(): Sets the candidates for all fields in a sudoku.
- find_conflicts(): Check sudoku for conflicting fields.
- is_unique(): Check if a sudoku has exactly one solution.

sudokutools.solve.**_do_bruteforce**(*sudoku*, *reverse=False*)
Solve sudoku _inplace_ and return the success status.

This is an internal function and should not be used outside of the solve module.

sudokutools.solve.**bruteforce**(*sudoku*, *reverse=False*)
Solve the sudoku using brute force and return a solution or None.

Parameters

- **sudoku** (Sudoku) – The Sudoku instance to solve.
- **reverse** – Solve the sudoku in reverse order, meaning that if a field has multiple valid numbers (a, b, c) use c instead of a. This only creates another solution, if the sudoku is not unique.

Returns The solution of the sudoku. None: If the sudoku is not solvable.

Return type *Sudoku*

sudokutools.solve.**calc_candidates**(*sudoku*, *row*, *col*)
Return a set of candidates of the sudoku at (row, col).

Parameters

- **sudoku** (Sudoku) – The Sudoku instance for which the candidates are calculated.
- **row** (*int*) – The row of the field
- **col** (*int*) – The column of the field.

Returns A set of candidates for the field at (row, col)

Return type set

sudokutools.solve.**find_conflicts**(*sudoku*, **coords*)
Yield conflicts in sudoku at coords.

If coords is empty all possible coordinates will be searched.

Parameters

- **sudoku** (Sudoku) – The Sudoku instance to check.

- **coords** (*iterable of (int, int)*) – The coordinates to search within.

Yields *((int, int), (int, int), int)* –

tuple of coordinate pairs and the offending value.

E.g.: ((2, 3), (2, 6), 2) indicates, that there is a conflict for the fields (2, 3) and (2, 6) because both of them contain a 2.

sudokutools.solve.**init_candidates**(*sudoku*)

Calculate and set all candidates in the sudoku.

Sets all candidates in the sudoku based on the numbers (and nothing else) in the surrounding fields.

Parameters **sudoku** (Sudoku) – The Sudoku instance for which the candidates are calculated.

sudokutools.solve.**is_unique**(*sudoku*)

Check if sudoku has exactly one solution.

Parameters **sudoku** (Sudoku) – The Sudoku instance to check.

Returns Whether or not the sudoku is unique.

Return type bool

2.1.3 sudokutools.generate - Creating new sudokus

Create new sudokus.

Functions defined here:

- create_solution(): Create a complete sudoku without conflicts.
- generate(): Create a new sudoku.

sudokutools.generate.**create_solution**()

Returns a sudoku, without empty or conflicting fields.

Returns The completely filled Sudoku instance.

Return type *Sudoku*

sudokutools.generate.**generate**(*min_count=17*, *symmetry=None*)

Generate a sudoku and return it.

Parameters

- **min_count** (*int*) – Number of fields that must be filled at least. Any number above 81 will raise a ValueError, Any number below 17 makes no sense (but will not cause an error), since unique sudokus must have at least 17 filled fields.
- **symmetry** (*str*) – The kind of symmetry that will be created. Possible values are: None (no symmetry), “rotate-90”, “rotate-180”, “mirror-x”, “mirror-y” and “mirror-xy”.

Returns The generated Sudoku instance.

Return type *Sudoku*

Raises

- ValueError, if symmetry is not a valid argument.
- ValueError, if min_count > 81.

2.2 License

```
MIT License

Copyright (c) 2017-2018 Maik Messerschmidt

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index