configrw

Release 1.1.0

May 16, 2021

Contents

ConfigRW is a simple python module which read and write config files based on key-value or INI-structure.

CHAPTER 1

Key features

- Maximum preserves formatting of the source file (indents, spaces, comments, etc)
- Support non-section (for access for simple config files based on key-value)
- Inserting an option on an arbitrary string in the section
- Support multiple values of option
- Support options without values
- Support comments in a section
- Support indentation for options, values
- Secure file rewriting. Using **.new* file on write, then renamed to original filename

CHAPTER 2

Table of contents

2.1 Installation

Installation package to user python-library directory:

```
$ pip install --user configrw
```

Or you can install package to global system directory of python libraries (not recommended):

```
$ sudo pip install configrw
```

2.2 Quick start

In next examples we will use the following INI file:

```
# This is comment
this is option = this is value
second option  = -100

[ SECTION1 ] # comment
    option1 = 100
    option2 = 200
    # comment
    option3 = 1.2

[ section2 ]
    param1 = 'str'
    param2 = 0 # comment
    parameter_without_value

[section3]
```

(continues on next page)

(continued from previous page)

```
    extensions =
        # comment1
        ext1
        # comment2
        ext2
        ext3
```

2.2.1 Access to non-section area

This is features needed if you want use simple key-value of config file

```
from configrw import Config

config = Config.from_file('/path/to/file')

section = config[None]              # Getting non-section
value = section['this is option']   # Getting the value
section['this is option'] = None    # Setting the value
del section['second option']        # Deleting the option
```

2.2.2 Access to section area

This is features needed if you want use INI config file

```
value = config['SECTION1']['option2']       # Getting the value
config['SECTION1']['option2'] = 0           # Setting the value
config['SECTION1']['option3'] = 300         # Adding new option to section

config['section3']['extensions'].append('ext4')     # Adding new value to multiple␣
↪values
config['section3']['extensions'].insert('ext0', 0)  # Inserting new value
config['section3']['extensions'][0] = 'extension0'  # Changing single value of␣
↪multiple values

config.write('/path/to/file')               # Saving config to file

# Render config to screen
for line in config.to_text():
    print(line)
```

INI-file after changes:

```
# This is comment
this is option

[ SECTION1 ] # comment
    option1 = 100
    option2 = 0
    # comment
    option3 = 300

[ section2 ]
    param1 = 'str'
```

(continues on next page)

(continued from previous page)

```
    param2 = 0 # comment
    parameter_without_value

[section3]
    extensions =
        extension0
        # comment1
        ext1
        # comment2
        ext2
        ext3
        ext4
```

2.3 Load config

You can load configuration from file:

```
from configrw import Config


config = Config.from_file('./path/to/file')
```

or load config from string:

```
config = Config.from_str("""
[section1]
option1 = 100
option2 = 200

[section2]
option1 = 300
""")
```

or create new empty config:

```
config = Config()
```

2.4 Management of sections

2.4.1 Checking if has a section

For checking existing section you can use method `has_section()`:

```
if not config.has_section('section1'):
    print('Not exists')
```

or use iteration:

```
if not 'section1' in config:
    print('Not exists')
```

2.4.2 Add new section

Adding a new section or reset exist section:

```
new_section = config.add_section('section1')
```

Also, you can add new section with a leading spaces or add comment after section name:

```
new_section = config.add_section('section1', text_before='    ', text_after=' # This␣
↪is comment')
```

2.4.3 Get an section

Get non-section area for management of simple configs:

```
non_section = config[None]
non_section = config.get_section()
```

Also you can get an existing section in two different ways:

```
section1 = config['section1']
section1 = config.get_section('section1')
```

Note: if section do not exist then raised `KeyError`

2.4.4 Remove an section

```
del config['section1']
```

or

```
config.remove_section('section1')
```

2.5 Management of items of section

2.5.1 Set or add new an option

```
section1 = config.add_section('section1', text_after=' # this is comment')

section1['option1'] = 'value1'
section1['option_without_value'] = None
section1.set_option('option2', 200)
```

or you can adding option with custom separator or position:

```
section1.set_option('option1', 100, sep=' == ', pos=0)
section1.set_option('option_without_value', pos=0)
```

Result:

```
[section1] # this is comment
option_without_value
option1 == 100
option2 = 200
```

Also, you can set option to non-section area:

```
non_section = config[None]
non_section['global_parameter'] = 1
```

Result:

```
global_parameter = 1
[section1] # this is comment
option_without_value
option1 == 100
option2 = 200
```

You can set option with custom indentation:

```
section1['    option1'] = 100
section1.set_option('    option1', 100)

section1['    option_without_value'] = None
section1.set_option('    option_without_value')
```

Result:

```
global_parameter = 1
[section1] # this is comment
    option_without_value
    option1 = 100
option2 = 200
```

2.5.2 Get values of option

You can get an option value in three different ways:

```
value = section1['option1']
value = section1.get_value('option1')
value = section1.get_option('option1')['value']
```

Note: Also you can get value by index at section. See the chapter next.

2.5.3 Remove an option

```
del section1['option2']
```

or

```
section1.remove_option('option2')
```

Note: Also you can delete an option by index. See the chapter next.

2.5.4 Add a new item

Instead of using `set_option()` method, you can use the low-level method `add_item()` to add empty lines, comments to the section:

```
# append new option:
section1.add_item({'key': '    option2', 'sep': ' = ', 'value': 2})
# insert new option
section1.add_item({'key': '    option0', 'sep': ' = ', 'value': 0}, pos=0)
# insert empty line
section1.add_item('', pos=0)
# append empty line
section1.add_item('')
# insert comment
section1.add_item('    # This is comment for option1', 3)
```

Result:

```
global_parameter = 1
[section1] # this is comment

    option0 = 0
    option_without_value
    # This is comment for option1
    option1 = 100
    option2 = 2
```

2.5.5 Get an item by index

```
comment = section1[3]
option1_value = section1[4]['value']
last_item = section1[len(section1) - 1]
```

2.5.6 Remove an item by index

```
del section1[0]
del section1[5]
del section1[4]
```

Result:

```
global_parameter = 1
[section1] # this is comment
    option0 = 0
    option_without_value
    # This is comment for option1
    option1 = 100
```

2.6 Print data

```
print(config)
```

Output:

```
[
    {'key': 'global_parameter', 'sep': ' = ', 'value': 1}
],
{'section1':
    [
        {'key': '    option0', 'sep': ' = ', 'value': 0},
        {'key': '    option_without_value', 'sep': None, 'value': None},
        '    # This is comment for option1',
        {'key': '    option1', 'sep': ' = ', 'value': 100}
    ]
}
```

```
print(config['section1'])
```

Output:

```
[
    {'key': '    option0', 'sep': ' = ', 'value': 0},
    {'key': '    option_without_value', 'sep': None, 'value': None},
    '    # This is comment for option1',
    {'key': '    option1', 'sep': ' = ', 'value': 100}
]
```

Also you can use method `to_text()` for all config or for a section:

```
for line in config.to_text():
    print(line)
```

Output as text format:

```
global_parameter = 1
[section1] # this is comment
    option0 = 0
    option_without_value
    # This is comment for option1
    option1 = 100
```

2.7 Write data to a file

```
config.write('/path/to/file')
```

If file was opened, then method `write()` can be just call without a parameter:

```
config.write()
```