pyconfigreader Documentation

Release 0.8.1

Aswa Paul

Aug 11, 2020

Contents

A configuration file handler for the most basic stuff in ini files that will get you up and running in no time.

CHAPTER 1

Getting Started

1.1 Installation

Installation is as simple as running this command in the terminal.

```
$ pip install pyconfigreader
```

1.2 QuickStart

To read a configuration file create a file `settings.ini` in the current directory and paste the following content:

```
[main]
reader = configreader
name = pyconfigreader
language = python
versions = [2.7, 3.4, 3.5, 3.6]

[DATA]
language = English
development = None
path = /home/ubuntu/
user = Ubuntu
groups = 1000
```

In a Python console, import `ConfigReader`:

```
>>> from pyconfigreader import ConfigReader
```

Declare the path to your `settings.ini` file:

```
>>> settings_ini = '/path/to/settings.ini'
```

Read the settings file:

```
>>> config = ConfigReader(settings_ini)
```

Get data from the config:

```
>>> config.get('reader')
... 'configreader'
>>> config.get('groups', section='DATA')
... 1000
```

Set values:

```
>>> config.set('key', 'value', section='section')
>>> config.set('number', 4, section='NEW')
```

Save changes:

```
>>> config.save()
```

Close the `ConfigReader` object:

```
>>> config.close()  # Close without saving changes
>>> # or
>>> config.close(save=True)  # Save the changes then close
```

CHAPTER 2

Introduction

pyconfigreader not only makes it easier to work with ini files but also to migrate between JSON and the OS environment. It is possible to transfer variables from one to the other.

The following code examples assume the following sample content in the settings.ini:

```
[main]
reader = configreader
name = pyconfigreader
language = python
versions = [2.7, 3.4, 3.5, 3.6]

[DATA]
language = English
development = None
path = /home/ubuntu/
user = Ubuntu
groups = 1000
```

CHAPTER 3

Getting Values

`ConfigReader` creates a default **main** section from which key-value pairs are read if no section is specified.

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> name = config.get('name')
>>> versions = config.get('versions')
>>> agency = config.get('agency')  # agency is None, it doesn't exist
>>> name
'pyconfigreader'
>>> versions
[2.7, 3.4, 3.5, 3.6]
>>> agency
>>> config.sections  # Get a list of sections
['main', 'DATA']
```

To get the value of a key in a different section, use the `section` parameter

```
>>> config.get('groups', section='DATA')
1000
>>> config.get('language', section='DATA')
'English'
```

By default, `ConfigReader` tries to evaluate the values into Python literals (int, None, list, etc..). In the example above, *groups* from the *DATA* section has been evaluated into an integer. To get a string, set the parameter `evaluate` to False.

```
>>> config.get('groups', section='DATA', evaluate=False)
'1000'
```

If a key does not exist, `ConfigReader` will raise `MissingOptionError`

```
>>> config.get('last_name')
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
```

(continues on next page)

(continued from previous page)

```
    File "/path/to/pyconfigreader/pyconfigreader/reader.py", line 336, in get
      raise MissingOptionError(key, section)
pyconfigreader.exceptions.MissingOptionError: No option 'last_name' in section: 'main'
>>>
```

If, instead, you prefer to get a specific value if none exists, use `default`

```
>>> config.get('last_name', default='Some Name')
'Some Name'
```

From the Python console, it is possible to get a preview of the contents in the settings file. `show()` previews the contents and also returns an OrderedDict of all the data.

```
>>> config.show()


--------------------settings.ini--------------------
                      main
                reader: configreader
                  name: pyconfigreader
              language: python
               version: [2.7, 3.4, 3.5, 3.6]

                      DATA
              language: English
           development: None
                  path: /home/ubuntu/
                  user: Ubuntu
                groups: 1000


-----------------------end-----------------------

OrderedDict([('main', OrderedDict([('reader', 'configreader'), ('name',
↪'pyconfigreader'), ('language', 'python'), ('version', [2.7, 3.4, 3.5, 3.6])])), (
↪'DATA', OrderedDict([('language', 'English'), ('development', None), ('path', '/
↪home/ubuntu/'), ('user', 'Ubuntu'), ('groups', 1000)]))])
```

To hide the preview, set `output` to False

```
>>> config.show(output=False)
OrderedDict([('main', OrderedDict([('reader', 'configreader'), ('name',
↪'pyconfigreader'), ('language', 'python'), ('version', [2.7, 3.4, 3.5, 3.6])])), (
↪'DATA', OrderedDict([('language', 'English'), ('development', None), ('path', '/
↪home/ubuntu/'), ('user', 'Ubuntu'), ('groups', 1000)]))])
```

CHAPTER 4

Setting Values

Setting values is done by passing a *key* and *value* to `set()`. This saves the value to section **main** unless `section` is specified. Unlike Python's ConfigParser order of section then key then value, `pyconfigreader` uses the key then value then section order.

```
>>> config.set('key', 'value')
>>> config.set('version', '2')  # Saved to section `main`
>>> config.set('user', 'root', section='DATA')  # Updates user in section DATA
```

If a section does not exist, it is created on-the-fly and the key-value pair is written to it.

```
>>> config.set('key', 'value', section='Section')
>>> name = config.get('name')
```

As at the moment, the changes are in-memory. To write these changes to the file on the disk close the `ConfigReader` instance with `save` as True or call `save()` which does not close the `ConfigReader` instance.

```
>>> config.close(save=True)  # Save on close
>>> # Or explicitly call
>>> # config.save()
>>> # then later, call
>>> # config.close()
```

In any case you want the changes to be written directly to the file on disk upon setting a value, then set the parameter `commit` to True

```
>>> config.set('name', None, section='DATA', commit=True)
```

In order for `ConfigReader` to be used, the settings.ini file does not have to pre-exist. If the file cannot be found then a new file will be created in the specified path

```
>>> config = ConfigReader()
>>> config.set('tests', 'unit')
>>> config.set('runner', 'default')
```

(continues on next page)

(continued from previous page)

```
>>> config.set('count', 1, section='RUNNER')
>>> config.close(save=True)
```

To set many values at once, it is more convenient to use `set_many()`

```
>>> config = ConfigReader()
>>> data = {'population': 'people', 'sample': True, 'count': 20}
>>> config.set_many(data, section='geography')
>>> config.close(save=True)
```

CHAPTER 5

Removing Keys

Keys can be removed permanently by calling `remove_key()`

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> config.remove_key('reader')  # the reader option is always set by default
>>> # or config.remove_option('reader')
>>> config.set('name', 'first', section='Details')
>>> config.remove_key('name', section='Details')
>>> config.close(save=True)
```

`remove_key()` is aliased by `remove_option()` taking the same arguments.

CHAPTER 6

Sections

You can derive all keys and their values from sections using `get_items()`. This returns an OrderedDict.

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> config.set('name', 'first', section='Details')
>>> config.get_items('Details')
OrderedDict([('name', 'first')])
>>> config.close()  # Or config.close(save=True)
```

Section are created on-the-fly when keys and values are set.

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> config.set('name', 'first', section='Details')  # Save key `name` with value␣
↪`first` in section `Details`
>>> config.close()
```

Sections can also be explicitly remove by calling `remove_section()`

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> config.set('name', 'first', section='Details') # Creates section `Details`
>>> config.remove_section('Details')    # Removes section `Details` plus all the keys␣
↪and values
>>> config.close()
```

CHAPTER 7

Environment Variables

7.1 Get Environment Variables

pyconfigreader makes it possible to load most environment variables into the settings.ini. To achieve this, use `load_env()`

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader('settings.ini')
>>> config.load_env()
```

7.2 Dump to Environment

With **pyconfigreader**, it is easy to pass values to the environment.

```
>>> import os
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> config.set('name', 'first', section='Details')
>>> config.to_env()
>>> os.environ['DETAILS_NAME']
'first'
>>> os.environ['MAIN_READER']
'configreader'
```

By default, `ConfigReader` prepends the key with the section name and then transforms to uppercase before dumping to the environment. For instance, in the above example, *name* in section *Details* generated the environment key `DETAILS_NAME`. If you do not prefer this behaviour and want the key to be saved as is then set `prepend` to False.

```
>>> config.to_env(prepend=False)
```

Alternatively, you can pass your instance of an environment to which the key-value pairs will be dumped.

```
>>> environment = os.environ.copy()
>>> config.to_env(environment=environment)
```

CHAPTER 8

JSON Files

8.1 Load JSON Files

pyconfigreader makes it convenient to load configuration from JSON files. This can be done by calling `load_json()`

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> config.load_json('config.json')  # Load from file config.json
```

As always, a full path to the JSON file is better than just a file name.

By default, `load_json()` loads the JSON configuration to the **main** section. It is possible to set the JSON data to be dumped to a different section - the parameter `section` makes this possible.

```
>>> from pyconfigreader import ConfigReader
>>> with ConfigReader(filename='config.ini') as config:
...     config.load_json('config.json', section='JSONDATA')
```

If you prefer some values to be loaded to a different section in the settings.ini then use the `identifier` parameter. For example, with `identifier` as @ and the following JSON configuration:

```
'@counters': {
             'start': {
                 'name': 'scrollers',
                 'count': 15
             },
             'end': {
                 'name': 'keepers',
                 'count': 5
             }
         }
```

One should expect a section named `counters` that looks like:

```
[counters]
start = {'name': 'scrollers', 'count': 15}
end = {'name': 'keepers', 'count': 5}
```

`load_json()` looks for first-level keys which are prefixed with the `identifier` and turns them into section names with their values being assigned to the same section.

The JSON file encoding can be set using the `encoding` parameter.

8.2 Dump JSON

The settings.ini configuration can be dumped as JSON by calling `to_json()`.

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> json_dump = config.to_json()
```

To dump the data to a JSON file, pass an open writable file to `to_json()`.

```
>>> from pyconfigreader import ConfigReader
>>> config = ConfigReader(filename='config.ini')
>>> with open('config.json', 'w') as f:
...     config.to_json(f)
```

Essentially, any file-like object can have the JSON data dumped into it.

```
>>> from io import StringIO
>>> s_io = StringIO()
>>> with ConfigReader('data.ini') as config:
...     config.to_json(s_io)
```

CHAPTER 9

The pyconfigreader API Reference

9.1 ConfigReader

9.2 Exceptions

CHAPTER 10

Indices and tables

- genindex
- modindex
- search