
IHH Documentation

Release 0.2.1

Romain Dartigues

Sep 11, 2020

Contents

1	Why?	3
2	Table of contents	5
2.1	Source documentation	5
2.2	Examples	9
2.3	Warnings	13
2.4	Known bugs / limitations	13
3	Indices and tables	15
	Python Module Index	17
	Index	19

Source code [GitLab project](#)

Bug tracker [GitLab issues](#)

License [BSD 3-Clause](#)

Generated Sep 11, 2020

Overview

IHH (I Hate INI hacks) is an attempt to provide simple configuration parsers (for Python) with a dictionary-like interface.

It try to be flexible and let you alter the syntax by sub-classing it.

CHAPTER 1

Why?

Because I Hate INI (initialization) files. I don't need sections, i think `ConfigParser` is a pain to use...

And also because in my opinion configuration files should not be *executed* (ie: i feel bad having a Python file as a configuration system, sure it is *flexible*, but, you know... [if you don't, you probably don't need this]).

2.1 Source documentation

ihih - simple configuration parsers with dictionary-like interface

Source code [GitLab project](#)

License BSD 3-Clause

```
class ihih.IHih(files, ignore_errors=False, *args, **kwargs)
```

Bases: dict

IHH - simple configuration parser

One key/value pair per line.

```
encoding = 'utf8'
```

define the encoding

```
_escape = ' (?<!\\\)(?:\\\\\\\\)* '
```

regexp definition of the escape sequence

```
_escaped_chars = '[\\\'\"\\#/\\\\\\\\]'
```

regexp definition of characters to unconditionally un-escape

```
_separator = '\\='
```

regexp definition of key/value separator

Must be a fixed-width expression.

```
_extract = '^\\s*\\n (?P<key>.+?)\\n \\s*%(separator)s\\s*\\n (?P<value>.*)'
```

extract key = [value] on a single line

```
_quote = '["\\\\"']
```

define what a quote might be

```
_quoted = '%(escape)s(?P<quote>%(quote)s)(?P<value>.*?)%(escape)s(?P=quote)'
```

how to find a quoted value

_bool = `'^(?P<false>0|no|false|off|disabled)|(?P<true>1|yes|true|on|enabled)$'`
 regexp definition of a boolean value (used by `get_bool()`)

__init__(*filenames*, *ignore_errors=False*, **args*, ***kwargs*)
 attempt to parse a list of filenames

Parameters

- **filenames** (*str* or *list(str)*) – one or many path to files
- **ignore_errors** (*bool*) – fail silently on `OSError`
- **args** (*list*) – passed to `dict` constructor
- **kwargs** (*dict*) – passed to `dict` constructor

_comment = `'(\\s*(escape)s(?:\\#|/))'`
 regexp definition of an in-line comment

ignore_errors = `False`
 do not stop on `OSError` when reading sources

reload(*force=False*, *ignore_errors=None*)
 call `parse()` on each configuration file

Parameters

- **force** (*bool*) – force (re)loading of files
- **ignore_errors** (*bool*) – ignore unreadable files

Returns `None`

parse(*filename*, *force=False*, *ignore_errors=None*)
 parse a configuration file

Parameters

- **filename** (*str*) – path to file to parse
- **force** (*bool*) – force (re)loading of files
- **ignore_errors** (*bool*) – ignore unreadable files, default: `ignore_errors`

Returns `bool`

Note: *filename* should be an absolute path.

_unescape(*value*, *quote=None*)
 remove escape prefix on “known escape”

See `_escaped_chars`.

This method attempt to utf8 encode `unicode()` objects.

_handle_fragment(*fragment*, *quote=None*)
 handle a fragment of a value

Provided to help on subclassing.

_comment_at(*value*)
 return the position of the begining of a comment

__parse_value (*value*, *data*)

parse the “value” part of a “key / value”

This function handle the quoted parts and the comments.

Parameters

- **value** (*str*) – value to parse
- **data** – instance supporting += operator

Returns parsed value

Return type type of *data*

__contains__ (*key*)

True if self contains *key*

Note: The *key* will be casted as `text_type()`.

__setitem__ (*key*, *value*)

set item *key* to *value*

Note: The *key* will be casted as `text_type()`.

__getitem__ (*key*)

return *key* value as internal type

You probably want to use one of the following: `get_text()`, `get_float()`, `get_int()`.

Note: The *key* will be casted as `text_type()`.

__delitem__ (*key*)

delete *key* from dict

Note: The *key* will be casted as `text_type()`.

__weakref__

list of weak references to the object (if defined)

get_text (*key*, *default=None*)

return *key* value as `text_type()` or *default* if not found

Note: The *key* will be casted as `text_type()`.

get (*key*, *default=None*)

alias to `get_text()`

get_float (*key*, *default=None*, *errors='strict'*)

return *key* value as `float()` or *default* if not found

If *errors* is “ignore”, return *default* value instead of raising `ValueError` on failure.

Note: The *key* will be casted as `text_type()`.

get_int (*key*, *default=None*, *errors='strict'*, *base=10*)
 return *key* value as `int()` or *default* if not found

If *errors* is “ignore”, return *default* value instead of raising `ValueError` on failure.

Note: The *key* will be casted as `text_type()`.

get_bool (*key*, *default=None*)
 attempt to coerce *key* value to a boolean accordingly `_bool` rules

class `ihih.IHIH` (**args*, ***kwargs*)
 Bases: `ihih.IHIH`

IHIH Interpolate - `IHIH` with variable interpolation

__IHIH__getkey (*key*, *path=None*)
 return *key* value as internal type with interpolated variables

For more informations, see: `__getitem__()`.

_variable = `'%(escape)s\\$ (?P<value>\\w+|%(escape)s\\{ (?P<unquoted>.+?) %(escape)s\\}) '`
 regexp definition of a “variable”

_escaped_chars = `'[\\\\\\\\\\\\\\\\'\\\\"\\\\#\\\\\\\\\\\\\\\\$\\\\{\\\\\\\\}] '`
 regexp definition of characters to unconditionally un-escape

__init__ (**args*, ***kwargs*)
 attempt to parse a list of filenames

Parameters

- **filenames** (*str* or *list(str)*) – one or many path to files
- **ignore_errors** (*bool*) – fail silently on `OSError`
- **args** (*list*) – passed to `dict` constructor
- **kwargs** (*dict*) – passed to `dict` constructor

__setitem__ (*key*, *value*)
 set item *key* to *value*

Note: The *key* will be casted as `text_type()`.

_handle_fragment (*fragment*, *quote=None*)
 search for variables in *fragment*

__getitem__ (*key*)
 return *key* value as internal type

You probably want to use one of the following: `get_text()`, `get_float()`, `get_int()`.

Note: The *key* will be casted as `text_type()`.

`_recursive (value)`
recursive variable handler

Default: empty string

You can overwrite this function when subclassing and chose to return a unexpanded version of the variable, raise an error or make a single, non recursive, lookup.

2.2 Examples

2.2.1 Getting started

Attempt to load a system-wide configuration file, whose settings will be overwritten by a user preferences files.

Missing files are silently ignored.

```
from ihih import IHIH

conf = IHIH(
    (
        '/etc/example.conf',
        os.path.join(os.path.expanduser('~'), '.example.conf')
    ),
    debug='1'
)

if conf.get_float('debug', errors='ignore'):
    print 'i am running in debug mode'
```

2.2.2 Reloading the conf

Assuming *conf* is a *IHIH* instance.

```
# reload on SIGHUP
import signal

signal.signal(signal.SIGHUP, lambda s, f: conf.reload())
```

2.2.3 Configuration format

By default, *IHIH* parse files using the following rules:

- the key is before the first = character
- the value is everything after the first = character
- the value might be empty
- key and value have their leading and trailing spaces stripped
- values can be quoted (between ' or ")
- quoted values have their quotes automatically removed (ie: "my value" becomes my value)
- single quotes are considered as a character

- lines not matching the key / separator / value are ignored
- comments (beginning with a # or //) are ignored and deleted from the value except if they are escaped or quoted
- specials characters (\ ' " # /) can be escaped by prefixing them with a backslash (\) to not be treated specially
- other (non-special) characters preceded by the escape character are not treated specially and the escape character is preserved

By default, *IHHI* parse files accordingly the following rules:

- same-same than *IHHI*
- add dollar (\$) in the special character list
- every word prefixed by a non-escaped dollar and not embraced by single-quotes (') is considered as a variable
- strings beginning with \${ and ending with } are also variables, this let you define variables containing non-word characters such as dots hyphens, or spaces
- variables interpolation is done when using the variable, this let you define (or change) the variable content later
- when a variable is not found, it resolve as an empty string
- variable recursion resolve to an empty string

Which mean that it could parse, to a certain extent (see *Single-line only*), subset of:

- shell script
- *Postfix* main.cf
- Python
- INI (will ignore the sections)

That could be convenient if you have to share a configuration file between scripts, given you pay attention to respect both formats.

Examples of configuration files

Parsing a shell script:

```
# as in shell
FOO="bar"
FOOBAR=foo-$FOO # resolve as: foo-bar
FOOBAR="foo-$FOO" # resolve as: foo-bar
FOOBAR='foo-$FOO' # resolve as: foo-$FOO
BAR=${FOO} # resolve as: bar
ABC="a" 'b' c # resolve as: a b c
C=hello # world # resolve as: hello
D=hello \# world # resolve as: hello # world

# different
DATE=$(date) # resolve as: $(date)
```

Parsing a main.cf:

```
smtpd_banner = $myhostname ESMTP
myhostname = foo.example.net
```

Parsing some Python:

```
# same
a = 'AA'
b = "BB"

# notably different
c = 'A' "B"      # resolve as: A B
d = c            # resolve as: c
```

Parsing an INI file:

```
; section is ignored
[uwsgi]
http-socket = :9090
processes = 4

; different, resolve as: localhost:9000
URL = localhost${http-socket}
```

2.2.4 Examples in the examples directory

You can see / run the examples in the examples directory.

Extending the parsers to parse INI

```
#!/usr/bin/python
# vim:set fileencoding=utf8:
'''INI parsing with ihih - proof of concept

A quick-and-dirty, incomplete, INI parsing proof-of-concept using :mod:`ihih`.
'''

import os
import re
import sys

sys.path.append(os.path.join(os.path.dirname(__file__), '..'))
from ihih import IHIH, IHIHI

class _IHIHI(IHIHI):
    _escaped_chars = r'[\\"'"/\#/\;]'
    _comment = r'(\s*%(escape)s(?:\#|//|\;))'

class IHIHINI(IHIH):
    _section = r'^%(escape)s\[ (?P<section>.+?) %(escape)s\]'
    _escaped_chars = r'[\\"'"/\#/\;]'
    _comment = _IHIHI._comment
    _separator = r'[\=:\:]'
```

(continues on next page)

(continued from previous page)

```

def __init__(self, *args, **kwargs):
    self.r_section = re.compile(
        self._section % {'escape': self._escape},
        re.U
    )

    super(IHHINI, self).__init__(*args, **kwargs)

__setitem__ = dict.__setitem__
__getitem__ = dict.__getitem__

def parse(self, filename, force=False, ignore IOError=True):
    section = None
    try:
        fo = open(filename)
    except IOError:
        if ignore IOError:
            return False
        raise

    for line in fo:
        results = self.r_section.match(line)
        if results:
            section = results.group(1)
            continue
        results = self.r_extract.match(line)
        if results:
            if section is None:
                raise KeyError('not in a section')
            elif section not in self:
                self[section] = _IHHI(())

            self[section][results.group('key')] = results \
                .group('value').rstrip()

    return True

if __name__ == '__main__':
    import tempfile

    with tempfile.NamedTemporaryFile() as tmp:
        tmp.write(['[My section]\nfoodir: $dir/whatever\ndir=frob\n'])
        tmp.write('key = "value" ; a comment')
        tmp.flush()

        conf = IHHINI(tmp.name)

    for section in conf:
        print('%s:' % section)
        for k in conf[section]:

```

(continues on next page)

(continued from previous page)

```
print('\t%s = %s' % (k, conf[section].get_unicode(k)))
```

2.3 Warnings

Warning: They are usage warning, but you are also encouraged to consult the *known bugs and limitations*.

2.3.1 Still in beta

This library is being used in production, but I still lack feedbacks...

Please let me know if you use it, your features requests, bugs, etc.

2.3.2 Default item getter return internal type

You probably want to favor `ihih.IHIH.get()` over `ihih.IHIH.__getitem__()` as the latter return the internal type which might not be suitable for your needs.

2.3.3 Automatic type conversion

This is a key / value, file-based, configuration system; so it forces everything as a string.

Just be aware of that.

2.3.4 File opening failure

Missing configuration files will be silently ignored, *but*, if a configuration file is not readable (permissions errors) or not a file (dead link or directory), it *will* raise an exception, as the user should be notified of this error.

2.4 Known bugs / limitations

If you find some bugs, you are welcome to report them :^)

Please see also the *warnings*.

2.4.1 Partial unicode handling

Unicode is only partially supported, for example it is *not* supported to pre-populate the configuration object with `unicode()`; see *not a true dict*.

It also assumes all files use the same encoding (default to UTF8, or at least ASCII7).

2.4.2 Not a true dict

The configuration objects do not behave like a true `dict`, especially:

No type conversion on some methods

Type conversion is not supported, at least, on:

- pre-population / initialization (ie: `IHIH((), {'a': 'b'})`)
- functions: `pop`, `popitem`, `setdefault`, `update`

```
# this will not work as expected (yet)
conf = IHIH('file.conf', {'pi': 3.14, 'lang': u'', u': 'Chinese'})

# as a workaround, use this method
conf = IHIH('file.conf')
conf['pi'] = 3.14
conf['lang'] = u''
conf[u'] = 'Chinese'

# now the defaults has been set, reparse
conf.reload(force=True)

# or you can alternatively, carefully specify (utf8) strings on the init
conf = IHIH('file.conf', {'pi': '3.14', 'lang': u''.encode('utf8'),
                          u''.encode('utf8'): 'Chinese'})

# now you can
conf['test'] = u'$pi, $lang, $!'

print conf.get_unicode('test') # resolve as: 3.14, , Chinese!
```

2.4.3 Single-line only

It does not, yet, support line-continuation; that mean your configuration value must fit on one line.

CHAPTER 3

Indices and tables

- `genindex`

i

`ihih`, 5

Symbols

`_IHIH__getkey()` (*ihih.IHIH method*), 8
`__contains__()` (*ihih.IHIH method*), 7
`__delitem__()` (*ihih.IHIH method*), 7
`__getitem__()` (*ihih.IHIH method*), 7
`__getitem__()` (*ihih.IHIH method*), 8
`__init__()` (*ihih.IHIH method*), 6
`__init__()` (*ihih.IHIH method*), 8
`__setitem__()` (*ihih.IHIH method*), 7
`__setitem__()` (*ihih.IHIH method*), 8
`__weakref__` (*ihih.IHIH attribute*), 7
`_bool` (*ihih.IHIH attribute*), 5
`_comment` (*ihih.IHIH attribute*), 6
`_comment_at()` (*ihih.IHIH method*), 6
`_escape` (*ihih.IHIH attribute*), 5
`_escaped_chars` (*ihih.IHIH attribute*), 5
`_escaped_chars` (*ihih.IHIH attribute*), 8
`_extract` (*ihih.IHIH attribute*), 5
`_handle_fragment()` (*ihih.IHIH method*), 6
`_handle_fragment()` (*ihih.IHIH method*), 8
`_parse_value()` (*ihih.IHIH method*), 6
`_quote` (*ihih.IHIH attribute*), 5
`_quoted` (*ihih.IHIH attribute*), 5
`_recursive()` (*ihih.IHIH method*), 8
`_separator` (*ihih.IHIH attribute*), 5
`_unescape()` (*ihih.IHIH method*), 6
`_variable` (*ihih.IHIH attribute*), 8

E

`encoding` (*ihih.IHIH attribute*), 5

G

`get()` (*ihih.IHIH method*), 7
`get_bool()` (*ihih.IHIH method*), 8
`get_float()` (*ihih.IHIH method*), 7
`get_int()` (*ihih.IHIH method*), 8
`get_text()` (*ihih.IHIH method*), 7

I

`ignore_errors` (*ihih.IHIH attribute*), 6

`IHIH` (*class in ihih*), 5

`ihih` (*module*), 5

`IHIH` (*class in ihih*), 8

P

`parse()` (*ihih.IHIH method*), 6

R

`reload()` (*ihih.IHIH method*), 6