simpleconfig Documentation

Release 2.0.0

Kyle Isom

Jun 24, 2019

Contents:

simpleconfig is a Python2/3 module for reading ini-style configuration files.

CHAPTER 1

Introduction

simpleconfig is a Python module for dealing with ini-style configuration files.

For example:

```
>>> config = """
# random version information
version = 1.0.5

; last modified 1 April 2001 by John Doe
; example taken from wikipedia
[owner]
name=John Doe
organization=Acme Widgets Inc.

[database]
# use IP address in case network name resolution is not working
server=192.0.2.62
port=143
file="payroll.dat"
"""
>>> import simpleconfig as sc
>>> cfg = sc.parse_string(config)
>>> cfg['owner']['name']
'John Doe'
>>> cfg.owner.name
'John Doe'
```

CHAPTER 2

INI file format

simpleconfig understands INI files in the following format:

```
[section_name]
key = value

# a comment describing the following section
; some ini files also use this, so it's supported too.
[other_section]
# this comment describes the key
key = value
```

Sections are defined as text inside square brackets. This acts as a namespace of sorts for the keys that follow. For example, in the above configuration, ‘key’ has a different value depending on whether it refers to ‘section_name’ or ‘other_section’. Another example:

```
[ www ]
address = 10.137.4.28
port = 8443
tls = strict

[ db ]
address = 10.137.4.91
port = 5432
tls = strict
```

Address and port, for example, depend on whether they refer to the www server or the db server.

If a key appears before the first section is defined, it goes into a section called ‘default’.

Keys are strings that name a value. They can contain any character that’s allowed in a word, but must be a single word. The parser will trim whitespace from the ends, and will stop parsing a key on the first ‘=’ character.

Values have the same restrictions as keys.

Comments are defined to be any line where the first non-whitespace character is either a ‘#’ or a ‘;’.

CHAPTER 3

Configurations

Configurations are what's returned from the the parsers. They are instances of kutils.dicts.AttrDictDict, which have the following properties in this module:

- attempting to read a non-existent section returns an empty dictionary.
- attempting to read a non-existent key returns an empty string.

This is because of a conscious design choice to avoid throwing exceptions for missing sections; typically, the author handles this with

```
def defaulted(value, default):
        if len(value) is not 0:
                return value
        return default
```

Otherwise, they behave exactly like `dict``s, and can be used anywhere a ``dict` can.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search