os.path2 Documentation

Release 0.0.4

Sebastian Pawluś

August 23, 2016

Contents

The **os.path** library replacement with simple API.

```
>>> from os import path2 as path

>>> path('/var/log')
/var/log

>>> path('/var', 'log')
/var/log

>>> path('/home/you/file').user
'you'

>>> [(element.user, element.group, element.mod) for element in path('.')]
[('user', 'user', '0664'),
 ('user', 'user', '0664'),
 ('user', 'user', '0664'),
 ('user', 'user', '0664'),
 ('user', 'user', '0664'),
 ('user', 'user', '0664'),
 ('user', 'user', '0664'),
 ('user', 'user', '0775'),
 ('user', 'user', '0664')]
```

CHAPTER 1

Status

Library seems to be pretty stable. Feel free to use it as you want. But this may not be the final version of the API.

CHAPTER 2

Install

You can install it from PyPi, by simply using **pip**:

```
$ pip install os.path2
```

(only if you don't have pip installed), an alternative method use **easy_install**:

```
$ easy_install os.path2
```

to test it launch **python**

```
>>> from os import path2 as path
```

CHAPTER 3

Supported platforms

- Python2.6
- Python2.7
- Python3.3
- PyPy1.9

CHAPTER 4

Source Code

https://github.com/xando/os.path2

CHAPTER 5

API

path.**user**
Path attribute, returns name of the user owner of the path.

```
>>> path('/home/user').user
user
```

path.**group**
Path attribute, returns name of the group owner of the path.

```
>>> path('/etc/').group
root
```

path.**mod**
To get Unix path permissions.

```
>>> path('.').mod
'0775'
```

path.**absolute**
Returns a normalised absolutized version of the pathname path.

```
>>> path('.').absolute
/home/user/Projects/osome
```

Return type path

path.**basename**
Returns the path basename.

```
>>> path('/home/user/Projects/osome').basename
osome
```

Return type path

path.**dir**
Returns the directory path of pathname path.

```
>>> path('/var/log/syslog').dir
/var/log
```

Return type path

path.**a_time**
Return the time of last access of path. The return value is a number giving the number of seconds since the epoch.

```
>>> path('/var/log/syslog').a_time
1358549788.7512302
```

Return type float

path.**m_time**
Return the time of last modification of path. The return value is a number giving the number of seconds since the epoch.

```
>>> path('/var/log/syslog').m_time
1358549788.7512302
```

Return type float

path.**size**
Return the size of path in bytes

```
>>> path('.').size
4096
```

Return type int

path.**exists**
Returns True if path refers to an existing path. Returns False for broken symbolic links.

```
>>> path('/var/log').exists
True
```

Return type bool

path.**is_dir**()
Return True if path is an existing directory. This follows symbolic links, so both is_link() and is_dir() can be true for the same path.

```
>>> path('/var/log').is_dir()
True
```

Return type bool

path.**is_file**()
Return True if path is an existing regular file. This follows symbolic links, so both is_link() and is_file() can be true for the same path.

```
>>> path('/var/log/syslog').is_file()
True
```

Return type bool

path.**mkdir**(*p=False*)

Parameters **p** – if changed will behave like mkdir -p, creating all directories recursively.

```
>>> path('dir').mkdir().exists
True
```

```
>>> path('/home/user/another/dir',p=True).mkdir().exists
True
```

Return type path

`path.`**`rm`**`(`*r=False*`)`

Removing file or directory, **r** parameter needs to be applied to remove directory recursively.

```
>>> path('file').rm()
file
```

```
>>> path('/tmp').rm(r=True)
/tmp
```

Return type path

`path.`**`cp`**`(`*target*`)`

Copy the file or the contents the directory to **target** destination, workrs for files and directories as well.

```
>>> path('dir').cp('dir_copy')
dir_copy
```

```
>>> path('file1').cp('file_copy')
file_copy
>>> path('file1').cp('file_copy').exists
True
```

Return type path

`path.`**`ln`**`(`*target*, *s=True*`)`

Create a link pointing to source named link_name, default call will create symbolic link, change **s=False** to create hard link.

```
>>> path('/tmp/').ln('/home/user/tmp')
'/home/user/tmp'
```

Return type path

`path.`**`unlink`**`()`

Unlink path from the poiting location.

```
>>> path('/home/user/tmp').is_link()
True
>>> path('/home/user/tmp').unlink()
'/home/user/tmp'
```

Return type path

`path.`**`touch`**`()`

Imitates call of Unix's **touch**.

```
>>> path('file').touch()
file
```

Return type path

`path.`**`ls`**(*pattern='*'*, *sort=None*)

Display content of the directory, use **pattern** as filtering parameter, change order by defining **sort** function.

```
>>> path('/var/log').ls()
[/var/log/boot.log, /var/log/dmesg, /var/log/faillog, /var/log/kern.log, /var/log/gdm]
```

```
>>> path('/var/log/').ls('*log')
[/var/log/boot.log, /var/log/faillog, /var/log/kern.log]
```

path('.').ls(sort=lambda x: not x.startswith('_')) [_themes, _build, _static, _templates, Makefile, conf.py, index.rst]

Return type list

`path.`**`ls_files`**(*patern='*'*, *sort=None*)

Returns files inside given path.

```
>>> path('.').ls_files()
[/var/log/boot.log, /var/log/dmesg, /var/log/faillog, /var/log/kern.log]
```

Return type list

`path.`**`ls_dirs`**(*patern='*'*, *sort=None*)

Returns directories inside given path.

```
>>> path('.').ls_dirs()
[/var/log/gdm]]
```

Return type list

`path.`**`walk`**(*pattern='*'*, *r=False*, *sort=None*)

Location walk generator

```
>>> for element in path('.').walk():
        print element
/var/log/boot.log
/var/log/dmesg
/var/log/faillog
/var/log/kern.log
/var/log/gdm
```

Return type generator

`path.`**`chmod`**(*mode*)

To change path access permissions

```
>>> path('test').chmod('0775')
>>> path('test').mod
'0775'
```

`path.`**`open`**(*mode=None*, **args*, ***kwargs*)

Open a file, returning an object of the File Objects.

```
    >>> path('/var/log','syslog').open('r')
    <open file '/var/log/syslog', mode 'r' at 0x294c5d0>
```

5.1 string/unicode methods

Path is also a instance of basestring so all methods implemented for string/unicode should work as well.

```
>>> path('.').absolute().split('/')
['', 'home', 'user', 'Projects', 'os.path2']

>>> path('/home/user/test_tmp_directory').replace('_', '-')
'/home/user/test-tmp-directory'

>>> location = path('/home/user/test_tmp_directory')
>>> location.mv(location.replace('_', '-'))
```