
rpds.py
Release 0.18.1

Julian Berman

May 06, 2024

CONTENTS

1	Usage	3
1.1	API Reference	3
	Python Module Index	7
	Index	9

Python bindings to the [Rust rpds crate](#) for persistent data structures.

What's here is quite minimal (in transparency, it was written initially to support replacing `pyrsistent` in the [referencing library](#)). If you see something missing (which is very likely), a PR is definitely welcome to add it.

The distribution on PyPI is named `rpds.py` (equivalently `rpds-py`), and thus can be installed via e.g.:

```
$ pip install rpds-py
```

Note that if you install `rpds-py` from source, you will need a Rust toolchain installed, as it is a build-time dependency. An example of how to do so in a `Dockerfile` can be found [here](#).

If you believe you are on a common platform which should have wheels built (i.e. and not need to compile from source), feel free to file an issue or pull request modifying the GitHub action used here to build wheels via `maturin`.

USAGE

Methods in general are named similarly to their `rpds` counterparts (rather than `pypersistent`'s conventions, though probably a full drop-in `pypersistent`-compatible wrapper module is a good addition at some point).

```
>>> from rpds import HashTrieMap, HashTrieSet, List

>>> m = HashTrieMap({"foo": "bar", "baz": "quux"})
>>> m.insert("spam", 37) == HashTrieMap({"foo": "bar", "baz": "quux", "spam": 37})
True
>>> m.remove("foo") == HashTrieMap({"baz": "quux"})
True

>>> s = HashTrieSet({"foo", "bar", "baz", "quux"})
>>> s.insert("spam") == HashTrieSet({"foo", "bar", "baz", "quux", "spam"})
True
>>> s.remove("foo") == HashTrieSet({"bar", "baz", "quux"})
True

>>> L = List([1, 3, 5])
>>> L.push_front(-1) == List([-1, 1, 3, 5])
True
>>> L.rest == List([3, 5])
True
```

1.1 API Reference

class `rpds.HashTrieMap`(*value=None, **kwds*)

__getitem__(*key*, /)

Return `self[key]`.

__iter__()

Implement `iter(self)`.

__len__()

Return `len(self)`.

convert()

discard(*key*)

```
fromkeys(val=None)
get(key, default=None)
insert(key, value)
items()
keys()
remove(key)
update(*maps, **kws)
values()

class rpds.HashTrieSet(value=None)
    __iter__()
        Implement iter(self).
    __len__()
        Return len(self).
    difference(other)
    discard(value)
    insert(value)
    intersection(other)
    remove(value)
    symmetric_difference(other)
    union(other)
    update(*iterables)

class rpds.List(*elements)
    __iter__()
        Implement iter(self).
    __len__()
        Return len(self).
    drop_first()
    first
    push_front(other)
    rest

class rpds.Queue(*elements)
    __iter__()
        Implement iter(self).
```


`__len__()`

Return `len(self)`.

`dequeue()`

`enqueue(value)`

`is_empty`

`peek`

PYTHON MODULE INDEX

r

rpds, 3

Symbols

`__getitem__()` (*rpds.HashTrieMap method*), 3
`__iter__()` (*rpds.HashTrieMap method*), 3
`__iter__()` (*rpds.HashTrieSet method*), 4
`__iter__()` (*rpds.List method*), 4
`__iter__()` (*rpds.Queue method*), 4
`__len__()` (*rpds.HashTrieMap method*), 3
`__len__()` (*rpds.HashTrieSet method*), 4
`__len__()` (*rpds.List method*), 4
`__len__()` (*rpds.Queue method*), 4

C

`convert()` (*rpds.HashTrieMap method*), 3

D

`dequeue()` (*rpds.Queue method*), 5
`difference()` (*rpds.HashTrieSet method*), 4
`discard()` (*rpds.HashTrieMap method*), 3
`discard()` (*rpds.HashTrieSet method*), 4
`drop_first()` (*rpds.List method*), 4

E

`enqueue()` (*rpds.Queue method*), 5

F

`first` (*rpds.List attribute*), 4
`fromkeys()` (*rpds.HashTrieMap method*), 3

G

`get()` (*rpds.HashTrieMap method*), 4

H

`HashTrieMap` (*class in rpds*), 3
`HashTrieSet` (*class in rpds*), 4

I

`insert()` (*rpds.HashTrieMap method*), 4
`insert()` (*rpds.HashTrieSet method*), 4
`intersection()` (*rpds.HashTrieSet method*), 4
`is_empty` (*rpds.Queue attribute*), 5
`items()` (*rpds.HashTrieMap method*), 4

K

`keys()` (*rpds.HashTrieMap method*), 4

L

`List` (*class in rpds*), 4

M

`module`
 rpds, 3

P

`peek` (*rpds.Queue attribute*), 5
`push_front()` (*rpds.List method*), 4

Q

`Queue` (*class in rpds*), 4

R

`remove()` (*rpds.HashTrieMap method*), 4
`remove()` (*rpds.HashTrieSet method*), 4
`rest` (*rpds.List attribute*), 4
`rpds`
 module, 3

S

`symmetric_difference()` (*rpds.HashTrieSet method*), 4

U

`union()` (*rpds.HashTrieSet method*), 4
`update()` (*rpds.HashTrieMap method*), 4
`update()` (*rpds.HashTrieSet method*), 4

V

`values()` (*rpds.HashTrieMap method*), 4