

Package ‘Dict’

July 21, 2025

Title R6 Based Key-Value Dictionary Implementation

Version 0.1.0

Description A key-value dictionary data structure based on R6 class which is designed to be similar usages with other languages dictionary (e.g. 'Python') with reference semantics and extendabilities by R6.

URL <https://github.com/five-dots/Dict>

BugReports <https://github.com/five-dots/Dict/issues>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports R6, dplyr, magrittr, purrr, rlang,

Suggests testthat

RoxygenNote 7.1.0

NeedsCompilation no

Author Shun Asai [aut, cre, cph]

Maintainer Shun Asai <syun.asai@gmail.com>

Repository CRAN

Date/Publication 2020-06-02 15:10:10 UTC

Contents

Dict	2
Index	7

Dict

*R6 Based Key-Value Dictionary Implementation***Description**

A key-value dictionary data structure based on R6 class which is designed to be similar usages with other languages dictionary (e.g. Python) with reference semantics and extendabilities by R6.

Usage

```
dict(..., .class = "any", .overwrite = TRUE)
```

Arguments

<code>...</code>	Any length of key and value pairs. If you would like to use a not valid R name as a key, you must wrap it by backquotes or convert it using make.names .
<code>.class</code>	A character scalar of value object's class. It must be an output from class . If "any" (default), value can contain any type of object.
<code>.overwrite</code>	A logical scalar whether to overwrite the value if the key is overlapped.

Value

A Dict class object.

Active bindings

`items` A `tbl_df` of the dictionary items.
`keys` A character vector of the dictionary keys.
`values` A list of of the dictionary values.
`length` A integer scalar of the items length.
`class` A character scalar of value class.
`overwrite` A logical scalar whether to overwrite value if key is overlapped.

Methods**Public methods:**

- [Dict\\$new\(\)](#)
- [Dict\\$print\(\)](#)
- [Dict\\$add\(\)](#)
- [Dict\\$has\(\)](#)
- [Dict\\$get\(\)](#)
- [Dict\\$remove\(\)](#)
- [Dict\\$sort\(\)](#)
- [Dict\\$clear\(\)](#)

- `Dict$clone()`

Method `new()`: Construct a new Dict object.

Usage:

```
Dict$new(..., .class = "any", .overwrite = TRUE)
```

Arguments:

... Any length of key and value pairs. If you would like to use a not valid R name as a key, you must wrap it by backquotes or convert it using `make.names`.

`.class` A character scalar of value object's class. It must be an output from `class`. If "any" (default), value can contain any type of object.

`.overwrite` A logical scalar whether to overwrite the value if the key is overlapped.

Returns: A Dict class object.

Examples:

```
ages <- Dict$new(
  Charlie = 40L,
  Alice = 30L,
  Bob = 25L,
  .class = "integer",
  .overwrite = TRUE
)
```

Method `print()`: Print Dict items which is a `tbl_df-class` object by tibble package.

Usage:

```
Dict$print(...)
```

Arguments:

... Additional arguments passed to `print.tbl`.

Returns: Dict object by `invisible(self)`.

Examples:

```
ages$print(n = Inf)
```

Method `add()`: Add key-value objects to the dictionary.

Usage:

```
Dict$add(...)
```

Arguments:

... Any length of key and value pairs. If you would like to use a not valid R name as a key, you must wrap it by backquotes or convert it using `make.names`.

Returns: Dict object by `invisible(self)`.

Examples:

```
ages$add(John = 18L)
ages["John"] <- 18L
```

Method `has()`: Check if the object contains the key.

Usage:

```
Dict$has(key = NULL)
```

Arguments:

key A character scalar of the dictionary key.

Returns: A logical scalar.

Examples:

```
ages$has("Bob")
```

Method `get()`: Retrieves object with a key from the dictionary.

Usage:

```
Dict$get(key = NULL, default = NULL)
```

Arguments:

key A character scalar, integer scalar of items index or NULL. If key is NULL and items is not empty, the first value is returned.

default A default value returned, if the key is not found. Default is NULL.

Returns: A object with the key.

Examples:

```
ages$get("Bob")
```

```
ages["Bob"]
```

```
ages[3] # also by integer index
```

Method `remove()`: Removes a key-value from the dictionary by a key. If the key is a not valid key, this function throw an error. Use `self$has()` to check key availability.

Usage:

```
Dict$remove(key = NULL)
```

Arguments:

key A character scalar of the dictionary key.

Returns: Dict object by `invisible(self)`.

Examples:

```
ages$remove("Bob")
```

Method `sort()`: Sort dictionary by keys.

Usage:

```
Dict$sort(desc = FALSE)
```

Arguments:

desc A logical scalar whether to sort in descending order. Default is FALSE.

Returns: Dict object by `invisible(self)`.

Examples:

```
ages$sort()
```

Method `clear()`: Clear dictionary.

Usage:

```
Dict$clear()
```

Returns: Dict object by invisible(self).

Examples:

```
ages$clear()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
Dict$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## -----
## Method `Dict$new`
## -----

ages <- Dict$new(
  Charlie = 40L,
  Alice = 30L,
  Bob = 25L,
  .class = "integer",
  .overwrite = TRUE
)

## -----
## Method `Dict$print`
## -----

ages$print(n = Inf)

## -----
## Method `Dict$add`
## -----

ages$add(John = 18L)
ages["John"] <- 18L

## -----
## Method `Dict$has`
## -----

ages$has("Bob")

## -----
## Method `Dict$get`
## -----

ages$get("Bob")
ages["Bob"]
```

```
ages[3] # also by integer index

## -----
## Method `Dict$remove`
## -----

ages$remove("Bob")

## -----
## Method `Dict$sort`
## -----

ages$sort()

## -----
## Method `Dict$clear`
## -----

ages$clear()
```

Index

`class`, [2](#), [3](#)

`Dict`, [2](#)

`dict (Dict)`, [2](#)

`make.names`, [2](#), [3](#)