Package 'pystr'

May 11, 2016

Type Package

Title Python String Methods in R

Version 2.0.0

Date 2016-05-10

Author Nicole White [aut, cre], Oliver Keyes [aut]

URL https://github.com/nicolewhite/pystr

BugReports https://github.com/nicolewhite/pystr/issues

Description String operations the Python way - a package for those of us who miss Python's string methods while we're working in R.

Maintainer Nicole White <nicole.m.white@outlook.com>

License MIT + file LICENSE

LazyData TRUE

Suggests testthat

LinkingTo Rcpp

Imports Rcpp

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-05-11 12:06:48

R topics documented:

`pystr` *Python String Methods in R*

Description

String operations the Python way - a package for those of us who miss Python's string methods while we're working in R. For support, please create a new GitHub issue.

Author(s)

Nicole White

pystr_capitalize *Capitalize a string.*

Description

Return a copy of the string with its first character capitalized and the rest lowercased.

Usage

```
pystr_capitalize(str)
```

Arguments

str	A character vector.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.capitalize

See Also

pystr_title

Examples

```
pystr_capitalize("ONCE UPON A TIME, ")
```

pystr_center *Center a string.*

Description

Return `str` centered in a string of length `width`.

Usage

```
pystr_center(str, width, fillchar = " ")
```

Arguments

str	A character vector.
width	An integer.
fillchar	A character string.

Details

Padding is done using the specified `fillchar` (default is an ASCII space). The original string is returned if `width` is less than or equal to `nchar(str)`.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.center

See Also

`pystr_ljust`, `pystr_rjust`

Examples

```
pystr_center("center me", 15)
pystr_center("center me", 15, "*")
```

---

pystr_count *Count the occurrences of a substring.*

Description

Return the number of non-overlapping occurrences of substring `sub` in the range `start, end`.

Usage

```
pystr_count(str, sub, start = 1, end = max(nchar(str)))
```

Arguments

`str`	A character vector.
`sub`	A character string.
`start`	An integer.
`end`	An integer.

Value

A numeric vector.

References

https://docs.python.org/3/library/stdtypes.html#str.count

Examples

```
pystr_count("ababab", "aba")
pystr_count("abcxyzabc123", "abc")
pystr_count("a--b--c", "--", 4)
pystr_count(c("one", "two", "three"), "e")
```

---

pystr_endswith *Check the suffix of a string.*

Description

Return TRUE if the string str ends with the specified suffix, otherwise return FALSE.

Usage

```
pystr_endswith(str, suffix, start = 1, end = nchar(str))
```

Arguments

str	A character vector.
suffix	A character vector.
start	A numeric vector.
end	A numeric vector.

Details

With optional start, test beginning at that position. With optional end, stop comparing at that position.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.endswith

See Also

pystr_startswith

Examples

```
pystr_endswith("selfie.jpg", ".jpg")
pystr_endswith("selfie.jpg", ".png")
pystr_endswith("hello world", "ello", 1, 5)
```

pystr_find *Find the lowest index of a substring.*

Description

Return the lowest index in the string where substring sub is found, such that sub is contained in the slice `substr(str, start, end)`.

Usage

```
pystr_find(str, sub, start = 1, end = nchar(str))
```

Arguments

str	A character vector.
sub	A character vector.
start	A numeric vector.
end	A numeric vector.

Value

A numeric vector. `-1` indicates sub was not found.

References

https://docs.python.org/3/library/stdtypes.html#str.find

See Also

pystr_rfind

Examples

```
pystr_find("abcdxyzabc", "abc")
pystr_find("abc", "xy")
pystr_find("abcxyzabc", "abc", 4)
```

pystr_format *Format a string.*

Description

Perform a string formatting operation.

Usage

```
pystr_format(str, ...)
```

Arguments

str	A character vector.
...	Parameter values. See details and examples

Details

The string on which this method is called can contain literal text or replacement fields delimited by braces {}. Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.

If ... is a single argument of a data.frame-like object, pystr_format will return an nrow()-length character vector using the column names of the data.frame for the named {placeholder}s.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.format

Examples

```
# Numeric placeholders

pystr_format("Hello {1}, my name is {2}.", "World", "Nicole")
pystr_format("Hello {1}, my name is {2}.", c("World", "Nicole"))
pystr_format("Hello {1}, my name is {2}.", list("World", "Nicole"))

# Named placeholders

pystr_format("Hello {thing}, my name is {name}.", thing="World", name="Nicole")
pystr_format("Hello {thing}, my name is {name}.", c(thing="World", name="Nicole"))
pystr_format("Hello {thing}, my name is {name}.", list(thing="World", name="Nicole"))

# Pass in characters and numbers
```

```
pystr_format("Hello {name}, you have {n} new notifications!", name="Nicole", n=2)

## Placeholders can be used more than once

pystr_format("The name is {last}. {first} {last}.", last="Bond", first="James")

## Pass in a whole data frame, matching by column names

my_cars <- data.frame(car=rownames(mtcars), mtcars)
head(pystr_format("The {car} gets {mpg} mpg (hwy) despite having {cyl} cylinders.", my_cars))

supers <- data.frame(first=c("Bruce", "Hal", "Clark", "Diana"),
                     last=c("Wayne", "Jordan", "Kent", "Prince"),
                     is=c("Batman", "Green Lantern", "Superman", "Wonder Woman"))
pystr_format("{first} {last} is really {is} but you shouldn't call them {first} in public.", supers)
```

`pystr_index` *Find the lowest index of a substring.*

Description

Like `pystr_find` but raises an error if sub is not found.

Usage

```
pystr_index(str, sub, start = 1, end = nchar(str))
```

Arguments

`str`	A character vector.
`sub`	A character vector.
`start`	A numeric vector.
`end`	A numeric vector.

Value

A numeric vector.

References

https://docs.python.org/3/library/stdtypes.html#str.index

See Also

`pystr_rindex`

Examples

```
pystr_index("abcxyzabc", "abc")
pystr_index("abcxyzabc", "abc", 4)
## Not run:
pystr_index("abcxyzabc", "123")

## End(Not run)
```

`pystr_isalnum` *Check if a string is alphanumeric.*

Description

Return `TRUE` if all characters in the string are alphanumeric and there is at least one character, `FALSE` otherwise.

Usage

```
pystr_isalnum(str)
```

Arguments

`str`	A character vector.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.isalnum

See Also

`pystr_isalpha`, `pystr_isnumeric`

Examples

```
pystr_isalnum("abc")
pystr_isalnum("abc123")
pystr_isalnum("abc123!")
pystr_isalnum(c("one", "2", "three!"))
```

pystr_isalpha *Check if a string is alphabetic.*

Description

Return TRUE if all characters in the string are alphabetic and there is at least one character, FALSE otherwise.

Usage

```
pystr_isalpha(str)
```

Arguments

str	A character vector.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.isalpha

See Also

pystr_isalnum, pystr_isnumeric

Examples

```
pystr_isalpha("abc")
pystr_isalpha("abc123")
pystr_isalpha("abc!")
pystr_isalpha(c("one", "2", "three!"))
```

pystr_islower *Check if a string is lowercase.*

Description

Return TRUE if all cased characters in the string are lowercase and there is at least one cased character, FALSE otherwise.

Usage

```
pystr_islower(str)
```

Arguments

str	A character vector.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.islower

See Also

pystr_isupper

Examples

```
pystr_islower("all lowercase!")
pystr_islower("All Lowercase?")
pystr_islower("abc123")
```

---

pystr_isnumeric *Check if a string is numeric.*

Description

Return TRUE if all characters in the string are numeric characters, and there is at least one character, FALSE otherwise.

Usage

```
pystr_isnumeric(str)
```

Arguments

str	A character vector.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.isnumeric

See Also

pystr_isalpha, pystr_isalnum

Examples

```
pystr_isnumeric("123")
pystr_isnumeric("123a")
pystr_isnumeric("123!")
```

`pystr_isspace` *Check if a string is whitespace.*

Description

Return `TRUE` if there are only whitespace characters in the string and there is at least one character, `FALSE` otherwise.

Usage

```
pystr_isspace(str)
```

Arguments

`str`	A character vector.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.isspace

Examples

```
pystr_isspace("    ")
pystr_isspace("  a ")
```

`pystr_istitle` *Check if a string is titlecase.*

Description

Return `TRUE` if the string is a titlecased string and there is at least one character, for example uppercase characters may only follow uncased characters and lowercase characters only cased ones. Return `FALSE` otherwise.

Usage

```
pystr_istitle(str)
```

Arguments

str	A character vector.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.istitle

Examples

```
pystr_istitle("I Am A Title")
pystr_istitle("I Am not A Title")
```

---

pystr_isupper *Check if a string is uppercase.*

Description

Return TRUE if all cased characters in the string are uppercase and there is at least one cased character, FALSE otherwise.

Usage

```
pystr_isupper(str)
```

Arguments

str	A character vector.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.isupper

See Also

pystr_islower

Examples

```
pystr_islower("ALL UPPERCASE!")
pystr_islower("All Uppercase?")
pystr_islower("ABC123")
```

pystr_join *Join the elements of a character vector or list into a string.*

Description

Return a string which is the concatenation of the elements in the iterable `iterable`, where `sep` is the separator between elements.

Usage

```
pystr_join(iterable, sep = "")
```

Arguments

`iterable`	A character vector.
`sep`	A character string.

Value

A character vector or list.

References

https://docs.python.org/3/library/stdtypes.html#str.join

See Also

pystr_split

Examples

```
pystr_join(c("A", "B", "C"))
pystr_join(c(1, 2, 3), "+")
pystr_join(list(c(1, 2, 3), c(4, 5, 6)), ", ")
```

pystr_ljust *Left justify a string.*

Description

Return `str` left justified in a string of length `width`. Padding is done using the specified `fillchar` (default is an ASCII space). The original string is returned if `width` is less than or equal to `nchar(str)`.

Usage

```
pystr_ljust(str, width, fillchar = " ")
```

Arguments

str	A character vector.
width	An integer.
fillchar	A character string.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.ljust

See Also

pystr_rjust

Examples

```
pystr_ljust("left", 10)
pystr_ljust("left", 10, "*")
```

---

pystr_lower *Lowercase a string.*

Description

Return a copy of the string with all the cased characters converted to lowercase.

Usage

```
pystr_lower(str)
```

Arguments

str	A character vector.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.lower

See Also

pystr_upper

Examples

```
pystr_lower("LOWERCASE ME!")
```

`pystr_lstrip` *Left strip a string.*

Description

Return a copy of the string with leading characters removed.

Usage

```
pystr_lstrip(str, chars = " ")
```

Arguments

`str`	A character vector.
`chars`	A character string.

Details

The `chars` argument is a string specifying the set of characters to be removed. If omitted, the `chars` argument defaults to removing whitespace. The `chars` argument is not a prefix; rather, all combinations of its values are stripped.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.lstrip

See Also

`pystr_rstrip`

Examples

```
pystr_lstrip("     spacious    ")
pystr_lstrip("www.example.com", "w.")
```

pystr_maketrans *Create a character map.*

Description

Create a list of character mappings usable for pystr_translate.

Usage

```
pystr_maketrans(x, y)
```

Arguments

x	A string.
y	A string.

Details

The two arguments must be strings of equal length, and in the resulting dictionary, each character in x will be mapped to the character at the same position in y.

Value

A list of character mappings for use in pystr_translate.

References

https://docs.python.org/3/library/stdtypes.html#str.maketrans

See Also

pystr_translate

Examples

```
map = pystr_maketrans("abc", "123")
pystr_translate("a blue cat", map)
```

`pystr_partition` *Partition a string.*

Description

Split the string at the first occurrence of `sep`, and return a list of character vectors containing the part before the separator, the separator itself, and the part after the separator.

Usage

```
pystr_partition(str, sep)
```

Arguments

`str`	A character vector.
`sep`	A character string.

Details

If the separator is not found, return a character vector containing the string itself, followed by two empty strings.

Value

A list of character vectors.

References

https://docs.python.org/3/library/stdtypes.html#str.partition

See Also

`pystr_rpartition`

Examples

```
pystr_partition("onetwothreeonetwothree", "two")
```

pystr_replace *Replace substrings within a string.*

Description

Return a copy of the string with all occurrences of substring `old` replaced by `new`.

Usage

```
pystr_replace(str, old, new, count = nchar(str) + 2)
```

Arguments

`str`	A character vector.
`old`	A character vector.
`new`	A character vector.
`count`	A numeric vector.

Details

If the optional argument `count` is given, only the first `count` occurrences are replaced.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.replace

Examples

```
pystr_replace("123123123", "2", "two")
pystr_replace("123123123", "2", "two", 2)
```

pystr_rfind *Find the highest index of a substring.*

Description

Return the higest index in the string where substring sub is found, such that sub is contained in the slice `substr(str, start, end)`.

Usage

```
pystr_rfind(str, sub, start = 1, end = nchar(str))
```

Arguments

str	A character vector.
sub	A character vector.
start	A numeric vector.
end	A numeric vector.

Value

A numeric vector. `-1` indicates sub was not found.

References

https://docs.python.org/3/library/stdtypes.html#str.rfind

See Also

pystr_find

Examples

```
pystr_rfind("abcdxyzabc", "abc")
pystr_rfind("abc", "xy")
pystr_rfind("abcxyzabc", "abc", 4)
```

pystr_rindex *Find the highest index of a substring.*

Description

Like pystr_rfind but raises an error if sub is not found.

Usage

```
pystr_rindex(str, sub, start = 1, end = nchar(str))
```

Arguments

str	A character vector.
sub	A character vector.
start	A numeric vector.
end	A numeric vector.

Value

A numeric vector.

References

https://docs.python.org/3/library/stdtypes.html#str.rindex

See Also

pystr_index

Examples

```
pystr_rindex("abcxyzabc", "abc")
pystr_rindex("12121212", "12", 4, 6)
## Not run:
pystr_rindex("abcxyzabc", "123")

## End(Not run)
```

pystr_rjust *Right justify a string.*

Description

Return `str` right justified in a string of length `width`. Padding is done using the specified `fillchar` (default is an ASCII space). The original string is returned if `width` is less than or equal to `nchar(str)`.

Usage

```
pystr_rjust(str, width, fillchar = " ")
```

Arguments

`str`	A character vector.
`width`	An integer.
`fillchar`	A character string.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.rjust

See Also

`pystr_ljust`

Examples

```
pystr_rjust("right", 10)
pystr_rjust("right", 10, "*")
```

`pystr_rpartition` *Partition a string from the right.*

Description

Split the string at the last occurrence of `sep`, and return a list of character vectors containing the part before the separator, the separator itself, and the part after the separator.

Usage

```
pystr_rpartition(str, sep)
```

Arguments

`str`	A character vector.
`sep`	A character string.

Details

If the separator is not found, return a character vector containing two empty strings, followed by the string itself.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.rpartition

See Also

`pystr_partition`

Examples

```
pystr_rpartition("onetwothreeonetwothree", "two")
```

pystr_rsplit *Right split a string.*

Description

Return a list of character vectors of the words in the string, using sep as the delimiter string.

Usage

```
pystr_rsplit(str, sep = " ", maxsplit = nchar(str) - 1)
```

Arguments

str	A character vector.
sep	A character string.
maxsplit	A numeric vector.

Details

If maxsplit is given, at most maxsplit splits are done, the rightmost ones. If sep is not specified, any whitespace string is a separator. Except for splitting from the right, pystr_rsplit behaves like pystr_split.

Value

A list of character vectors.

References

https://docs.python.org/3/library/stdtypes.html#str.rsplit

See Also

pystr_split

Examples

```
pystr_rsplit("a--b--c", "--")
pystr_rsplit("a--b--c", "--", 1)
```

pystr_rstrip *Right strip a string.*

Description

Return a copy of the string with trailing characters removed.

Usage

```
pystr_rstrip(str, chars = " ")
```

Arguments

str	A character vector.
chars	A character string.

Details

The chars argument is a string specifying the set of characters to be removed. If omitted, the chars argument defaults to removing whitespace. The chars argument is not a suffix; rather, all combinations of its values are stripped.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.rstrip

See Also

pystr_lstrip

Examples

```
pystr_rstrip("    spacious     ")
pystr_rstrip("www.example.com", ".omc")
```

pystr_split *Split a string.*

Description

Return a list of character vectors of the words in the string, using sep as the delimiter string.

Usage

```
pystr_split(str, sep = " ", maxsplit = nchar(str) - 1)
```

Arguments

str	A character vector.
sep	A character string.
maxsplit	A numeric integer.

Details

If maxsplit is given, at most maxsplit splits are done (thus, the character vector will have at most maxsplit + 1 elements). If maxsplit is not specified, then there is no limit on the number of splits (all possible splits are made). If sep is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings. The sep argument may consist of multiple characters. If sep is not specified, any whitespace string is a separator. Splitting an empty string returns an empty string.

Value

A list of character vectors.

References

https://docs.python.org/3/library/stdtypes.html#str.split

See Also

pystr_join, pystr_rsplit

Examples

```
pystr_split("www.example.com", ".")
pystr_split("123123123", "2", 2)
pystr_split("1,,2,3", ",")
pystr_split("a--b--c", "--")
```

pystr_splitlines *Split a string at linebreaks.*

Description

Return a list of the lines in the string, breaking at line boundaries.

Usage

```
pystr_splitlines(str, keepends = FALSE)
```

Arguments

str	A character vector.
keepends	A logical vector.

Details

Line breaks are not included in the resulting list unless keepends is TRUE. Line breaks include "\n", "\r", and "\r\n".

Value

A list of character vectors.

References

https://docs.python.org/3/library/stdtypes.html#str.splitlines

See Also

pystr_split

Examples

```
pystr_splitlines("First\nSecond\rThird\r\n")
pystr_splitlines("First\nSecond\rThird\r\n", TRUE)
```

pystr_startswith *Check the prefix of a string.*

Description

Return TRUE if the string str starts with the specified prefix, otherwise return FALSE.

Usage

```
pystr_startswith(str, prefix, start = 1, end = nchar(str))
```

Arguments

str	A character vector.
prefix	A character vector.
start	A numeric vector.
end	A numeric vector.

Details

With optional start, test string beginning at that position. With optional end, stop comparing string at that position.

Value

A logical vector.

References

https://docs.python.org/3/library/stdtypes.html#str.startswith

See Also

pystr_endswith

Examples

```
pystr_startswith("www.example.com", "www.")
pystr_startswith("example.com", "www.")
pystr_startswith("www.example.com", "example", 5)
```

pystr_strip *Strip a string.*

Description

Return a copy of the string with the leading and trailing characters removed.

Usage

```
pystr_strip(str, chars = " ")
```

Arguments

str	A character vector.
chars	A character string.

Details

The chars argument is a string specifying the set of characters to be removed. If omitted, the chars argument defaults to removing whitespace. The chars argument is not a prefix or suffix; rather, all combinations of its values are stripped.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.strip

See Also

pystr_lstrip, pystr_rstrip

Examples

```
pystr_strip("   very spacious   ")
pystr_strip("www.example.com", "cmowz.")
```

pystr_swapcase *Swap the case of a string.*

Description

Return a copy of the string with uppercase characters converted to lowercase and vice versa.

Usage

```
pystr_swapcase(str)
```

Arguments

str	A string.

Value

A string.

References

https://docs.python.org/3/library/stdtypes.html#str.swapcase

Examples

```
pystr_swapcase("Swap Me!")
```

pystr_title *Titlecase a string.*

Description

Return a titlecased version of the string where words start with an uppercase character and the remaining characters are lowercase.

Usage

```
pystr_title(str)
```

Arguments

str	A character vector.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.title

Examples

```
pystr_title("make me pretty!")
```

`pystr_translate` *Translate a string.*

Description

Return a copy of `str` where all characters have been mapped through `map`, where `map` can be created with `pystr_maketrans`.

Usage

```
pystr_translate(str, map)
```

Arguments

`str`	A character vector.
`map`	A list of character mappings.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.translate

See Also

`pystr_maketrans`

Examples

```
map = pystr_maketrans("abc", "123")
pystr_translate("a blue cat", map)
```

pystr_upper *Uppercase a string.*

Description

Return a copy of the string with all the cased characters converted to uppercase.

Usage

```
pystr_upper(str)
```

Arguments

str	A character vector.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.upper

See Also

pystr_lower

Examples

```
pystr_upper("uppercase me!")
```

pystr_zfill *Zero-pad a string.*

Description

Return a copy of the string left filled with ASCII '0' digits to make a string of length width.

Usage

```
pystr_zfill(str, width)
```

Arguments

str	A character vector.
width	An integer.

Details

A leading sign prefix (+/-) is handled by inserting the padding after the sign character rather than before. The original string is returned if `width` is less than or equal to `nchar(str)`.

Value

A character vector.

References

https://docs.python.org/3/library/stdtypes.html#str.zfill

Examples

```
pystr_zfill("42", 5)
pystr_zfill("-42", 5)
pystr_zfill("+42", 5)
```

Index