Package 'strs'

September 2, 2024

Title 'Python' Style String Functions

Version 0.1.0

Description A comprehensive set of string manipulation functions based on those found in 'Python' without relying on 'reticulate'. It provides functions that intend to (1) make it easier for users familiar with 'Python' to work with strings, (2) reduce the complexity often associated with string operations, (3) and enable users to write more readable and maintainable code that manipulates strings.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports stringi

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/Needs/website pythonicr/pythonicrtemplate

URL https://github.com/pythonicr/strs, https://pythonicr.github.io/strs/

BugReports https://github.com/pythonicr/strs/issues

NeedsCompilation no

Author Garrett Shipley [aut, cre] (<https://orcid.org/0000-0002-0444-0367>)

Maintainer Garrett Shipley <garrett.shipley7@gmail.com>

Repository CRAN

Date/Publication 2024-09-02 14:30:02 UTC

Contents

`strs_capitalize` *Capitalize the first character of each sentence*

Description

This function capitalizes the first character of each string in a given string, based on the specified locale. This is similar to Python's `str.capitalize()` method.

Usage

```
strs_capitalize(string, locale = "en")
```

Arguments

`string`	A character vector where each element is a string to be capitalized.
`locale`	A character string representing the locale to be used for capitalization. Defaults to "en" (English). The locale affects the rules for identifying sentences in the string.

Value

A character vector of the same length as `string`, where each element is the capitalized version of the corresponding element in `string`.

See Also

Python str.capitalize() documentation

Examples

```
strs_capitalize("hello world")
```

`strs_casefold` *Perform case folding on strings*

Description

`strs_casefold` is used to perform case folding on each element of a character vector. This function is particularly useful for case-insensitive string matching and is similar to Python's `str.casefold()` method.

Usage

```
strs_casefold(string)
```

Arguments

`string`	A character vector where each element is a string to be case-folded.

Value

A character vector of the same length as `string`, where each element has been case-folded.

See Also

Python str.casefold() documentation

Examples

```
strs_casefold("HELLO World")
strs_casefold("Äpfel")
```

strs_center *Center a string in a field of a given width*

Description

`strs_center` centers each element of a character vector in a field of a specified width. It pads the string on both sides with a specified character (defaulting to a space). This is similar to Python's `str.center()` method.

Usage

```
strs_center(string, width, fillchar = " ")
```

Arguments

`string`	A character vector where each element is a string to be centered.
`width`	The total width of the field in which the string is to be centered.
`fillchar`	A character used for padding. If not specified, defaults to a space. Only the first character of `fillchar` is used if it is longer than one character.

Value

A character vector of the same length as `string`, where each element has been centered in a field of the specified width.

See Also

Python str.center() documentation

Examples

```
strs_center("hello", 10)
strs_center("world", 10, "*")
```

strs_contains *Check if string contains a substring*

Description

`strs_contains` checks whether each element of a character vector contains a specified substring. This function mirrors the functionality of Python's `str.__contains__()` method.

Usage

```
strs_contains(string, substring)
```

Arguments

`string`	A character vector where each element is a string to be checked.
`substring`	The substring to search for within each element of `string`.

Value

A logical vector of the same length as `string`, with each element indicating whether the corresponding element of `string` contains `substring`.

Examples

```
strs_contains("hello world", "world")
strs_contains(c("apple", "banana", "cherry"), "a")
```

`strs_count` *Count occurrences of a substring in a string*

Description

`strs_count` counts the number of times a specified substring occurs in each element of a character vector. Optionally, the search can be limited to a substring of each element, specified by `start` and `end` positions. This function is similar to Python's `str.count()` method.

Usage

```
strs_count(string, substring, start = 1L, end = -1L)
```

Arguments

`string`	A character vector where each element is a string in which to count occurrences of `substring`.
`substring`	The substring to count within each element of `string`.
`start`	An optional integer specifying the starting position in each element of `string` for the search. Defaults to 1, indicating the start of the string.
`end`	An optional integer specifying the ending position in each element of `string` for the search. The default value of -1 indicates the end of the string.

Value

An integer vector of the same length as `string`, with each element indicating the count of `substring` in the corresponding element of `string`.

See Also

Python str.count() documentation

Examples

```
strs_count("hello world", "o")
strs_count("banana", "na")
strs_count("hello world", "o", start = 6)
strs_count("hello world", "o", end = 5)
```

`strs_endswith` *Check if string ends with a specified suffix*

Description

`strs_endswith` determines whether each element of a character vector ends with a specified suffix. This function is similar to Python's `str.endswith()` method.

Usage

```
strs_endswith(string, suffix)
```

Arguments

`string`	A character vector where each element is a string to be checked.
`suffix`	The suffix to check for at the end of each element of `string`.

Value

A logical vector of the same length as `string`, with each element indicating whether the corresponding element of `string` ends with `suffix`.

See Also

Python str.endswith() documentation

Examples

```
strs_endswith("hello world", "world")
strs_endswith(c("test", "hello", "world"), "ld")
```

strs_expandtabs *Expand tabs in a string to spaces*

Description

strs_expandtabs replaces each tab character (\\t) in a string with a specified number of spaces. This function behaves similarly to Python's str.expandtabs() method.

Usage

```
strs_expandtabs(string, tabsize = 8)
```

Arguments

string	A character vector where each element is a string in which to expand tabs.
tabsize	An integer specifying the number of spaces to replace each tab character with. Defaults to 8.

Value

A character vector of the same length as string, with tabs in each element replaced by tabsize number of spaces.

See Also

Python str.expandtabs() documentation

Examples

```
strs_expandtabs("hello\tworld", 4)
strs_expandtabs("one\ttwo\tthree", 8)
```

strs_find *Find the first occurrence of a substring in a string*

Description

strs_find locates the first occurrence of a specified substring within each element of a character vector. This function is analogous to Python's str.find() method.

Usage

```
strs_find(string, substring)
```

Arguments

`string`	A character vector where each element is a string to search.
`substring`	The substring to find within each element of `string`.

Value

An integer vector of the same length as `string`, with each element representing the starting position of the first occurrence of `substring` in the corresponding element of `string`. If the substring is not found, the function returns NA for that element.

See Also

Python str.find() documentation

Examples

```
strs_find("hello world", "world")
strs_find("hello world", "x")
```

`strs_isalnum` *Check if string is alphanumeric*

Description

`strs_isalnum` checks whether each element of a character vector is alphanumeric. This means that the function tests if all characters in the string are either letters or digits. It is similar to Python's `str.isalnum()` method.

Usage

```
strs_isalnum(string)
```

Arguments

`string`	A character vector to be checked.

Value

A logical vector of the same length as `string`, with each element indicating whether the corresponding element of `string` is completely alphanumeric.

See Also

Python str.isalnum() documentation

Examples

```
strs_isalnum("hello123")
strs_isalnum("hello world")
strs_isalnum("12345")
```

`strs_isalpha` *Check if string contains only alphabetical characters*

Description

`strs_isalpha` checks whether each element of a character vector contains only alphabetical characters. It is similar to Python's `str.isalpha()` method.

Usage

```
strs_isalpha(string)
```

Arguments

`string`	A character vector to be checked.

Value

A logical vector of the same length as `string`, indicating whether each element contains only alphabetical characters.

See Also

Python str.isalpha() documentation

Examples

```
strs_isalpha("hello")
strs_isalpha("hello123")
```

strs_isascii *Check if string contains only ascii characters*

Description

`strs_isascii` determines whether each element of a character vector contains only ASCII characters. It is similar to Python's `str.isascii()` method.

Usage

```
strs_isascii(string)
```

Arguments

`string`	A character vector to be checked.

Value

A logical vector of the same length as `string`, indicating whether each element contains only ASCII characters.

See Also

Python str.isascii() documentation

Examples

```
strs_isascii("hello")
strs_isascii("héllo")
```

strs_isdecimal *Check if string contains only decimal characters*

Description

`strs_isdecimal` checks whether each element of a character vector contains only decimal characters. It is similar to Python's `str.isdecimal()` method.

Usage

```
strs_isdecimal(string)
```

Arguments

`string`	A character vector to be checked.

Value

A logical vector of the same length as `string`, indicating whether each element contains only decimal characters.

See Also

Python str.isdecimal() documentation

Examples

```
strs_isdecimal("12345")
strs_isdecimal("123.45") # FALSE
```

`strs_isdigit` *Check if string contains only digits*

Description

`strs_isdigit` checks whether each element of a character vector contains only digits. It is similar to Python's `str.isdigit()` method.

Usage

```
strs_isdigit(string)
```

Arguments

`string`	A character vector to be checked.

Value

A logical vector of the same length as `string`, indicating whether each element contains only digits.

See Also

Python str.isdigit() documentation

Examples

```
strs_isdigit("12345")
strs_isdigit("123a")
```

strs_islower *Check if string is in lowercase*

Description

strs_islower checks whether each element of a character vector is in lowercase. It is similar to Python's str.islower() method.

Usage

```
strs_islower(string)
```

Arguments

string	A character vector to be checked.

Value

A logical vector of the same length as string, indicating whether each element is entirely in lowercase.

See Also

Python str.islower() documentation

Examples

```
strs_islower("hello")
strs_islower("Hello")
```

strs_isnumeric *Check if string contains only numeric characters*

Description

strs_isnumeric checks whether each element of a character vector contains only numeric characters. It is similar to Python's str.isnumeric() method.

Usage

```
strs_isnumeric(string)
```

Arguments

string	A character vector to be checked.

Value

A logical vector of the same length as `string`, indicating whether each element contains only numeric characters.

See Also

Python str.isnumeric() documentation

Examples

```
strs_isnumeric("12345")
strs_isnumeric("123a") # contains a non-numeric character
```

`strs_isspace` *Check if string contains only whitespace characters*

Description

`strs_isspace` checks whether each element of a character vector contains only whitespace characters. It is similar to Python's `str.isspace()` method.

Usage

```
strs_isspace(string)
```

Arguments

`string`	A character vector to be checked.

Value

A logical vector of the same length as `string`, indicating whether each element contains only whitespace characters.

See Also

Python str.isspace() documentation

Examples

```
strs_isspace("    ")
strs_isspace("hello world")
```

strs_istitle *Check if string is in title case*

Description

strs_istitle checks whether each element of a character vector is title case. This is similar to Python's str.istitle method.

Usage

```
strs_istitle(string)
```

Arguments

string	A character vector where each element is a string to be checked.

Value

A logical vector of the same length as string, indicating whether each element is in title case.

See Also

Python str.istitle() documentation

Examples

```
strs_istitle("This Is Title Case")
strs_istitle("not title case")
strs_istitle("123 Another Example")
```

strs_isupper *Check if string is in uppercase*

Description

strs_isupper checks whether each element of a character vector is in uppercase. It is similar to Python's str.isupper() method.

Usage

```
strs_isupper(string)
```

Arguments

string	A character vector to be checked.

Value

A logical vector of the same length as `string`, indicating whether each element is entirely in uppercase.

See Also

Python str.isupper() documentation

Examples

```
strs_isupper("HELLO")
strs_isupper("Hello")
```

---

strs_join *Join elements into a single string with a separator*

Description

`strs_join` concatenates elements of `iterable` using `sep`. It is similar to Python's `str.join()`.

Usage

```
strs_join(sep, iterable)
```

Arguments

`sep`	A string separator used to join the elements.
`iterable`	A character vector to be joined.

Value

A single string with elements of `iterable` joined by `sep`.

See Also

Python str.join() documentation

Examples

```
strs_join("-", c("hello", "world"))
strs_join("", c("hello", "world"))  # no separator
```

strs_ljust *Left-justify string in a field of a given width*

Description

strs_ljust left-justifies each element of a character vector in a field of a specified width. It is similar to Python's str.ljust() method.

Usage

```
strs_ljust(string, width, fillchar = " ")
```

Arguments

string	A character vector where each element is a string to be left-justified.
width	The total width of the field in which the string is to be left-justified.
fillchar	A character used for padding on the right.

Value

A character vector of the same length as string, with each element left-justified in a field of the specified width.

See Also

Python str.ljust() documentation

Examples

```
strs_ljust("hello", 10)
strs_ljust("world", 10, "*")
```

strs_lower *Convert string to lowercase*

Description

strs_lower converts each element of a character vector to lowercase, based on the specified locale. It is similar to Python's str.lower() method.

Usage

```
strs_lower(string, locale = "en")
```

Arguments

string	A character vector to be converted to lowercase.
locale	A character string representing the locale to be used for the conversion.

Value

A character vector of the same length as string, with each element converted to lowercase.

See Also

Python str.lower() documentation

Examples

```
strs_lower("HELLO WORLD")
strs_lower("Äpfel", locale = "de")
```

---

strs_lstrip *Left strip characters from a string*

Description

strs_lstrip removes leading characters (spaces by default) from each element of a character vector. It is similar to Python's str.lstrip() method.

Usage

```
strs_lstrip(string, chars = NULL)
```

Arguments

string	A character vector where each element is a string to be left-stripped.
chars	An optional string of characters to be removed from the beginning of each element. If NULL, whitespace is removed.

Value

A character vector of the same length as string, with specified characters removed from the beginning of each element.

See Also

Python str.lstrip() documentation

Examples

```
strs_lstrip("    hello world")
strs_lstrip("xxxyhello world", chars = "xy")
```

strs_normalize_whitespace *Normalize whitespace in a string*

Description

strs_normalize_whitespace normalizes the whitespace in each element of a character vector. It trims leading and trailing whitespace and replaces any sequence of whitespace characters within the string with a single space. This function is akin to the typical Python pattern `" ".join(str.split())`.

Usage

```
strs_normalize_whitespace(string)
```

Arguments

string	A character vector where each element is a string in which to normalize whitespace.

Value

A character vector of the same length as string, with whitespace normalized in each element.

Examples

```
strs_normalize_whitespace("  hello   world  ")
strs_normalize_whitespace("\thello\nworld\t")
```

strs_removeprefix *Remove a prefix from a string*

Description

strs_removeprefix removes a specified prefix from the start of each element of a character vector. It is similar to Python's str.removeprefix() method.

Usage

```
strs_removeprefix(string, prefix)
```

Arguments

string	A character vector where each element is a string from which to remove the prefix.
prefix	The prefix to remove.

Value

A character vector of the same length as `string`, with the `prefix` removed from each element.

See Also

Python str.removeprefix() documentation

Examples

```
strs_removeprefix("testString", "test")
strs_removeprefix("hello world", "hello")
```

`strs_removesuffix` *Remove a suffix from a string*

Description

`strs_removesuffix` removes a specified suffix from the end of each element of a character vector. It is similar to Python's `str.removesuffix()` method.

Usage

```
strs_removesuffix(string, suffix)
```

Arguments

`string`	A character vector where each element is a string from which to remove the suffix.
`suffix`	The suffix to remove.

Value

A character vector of the same length as `string`, with the `suffix` removed from each element.

See Also

Python str.removesuffix() documentation

Examples

```
strs_removesuffix("StringTest", "Test")
strs_removesuffix("hello world", "world")
```

strs_replace *Replace substring in a string*

Description

strs_replace replaces all occurrences of a specified substring in each element of a character vector. It is similar to Python's str.replace() method.

Usage

```
strs_replace(string, substring, replacement)
```

Arguments

string	A character vector where each element is a string in which to replace substring.
substring	The substring to be replaced.
replacement	The string to replace substring with.

Value

A character vector of the same length as string, with substring replaced by replacement.

See Also

Python str.replace() documentation

Examples

```
strs_replace("hello world", "world", "there")
strs_replace("banana", "na", "mo")
```

strs_rfind *Find the last occurrence of a substring in a string*

Description

strs_rfind locates the last occurrence of a specified substring within each element of a character vector. It is similar to Python's str.rfind() method.

Usage

```
strs_rfind(string, substring)
```

Arguments

`string`	A character vector where each element is a string to search.
`substring`	The substring to find within each element of `string`.

Value

An integer vector of the same length as `string`, with each element representing the starting position of the last occurrence of `substring` in the corresponding element of `string`. If the substring is not found, the function returns NA for that element.

See Also

Python str.rfind() documentation

Examples

```
strs_rfind("hello world", "o")
strs_rfind("hello world", "x") # not found
```

`strs_rjust` *Right-justify string in a field of a given width*

Description

`strs_rjust` right-justifies each element of a character vector in a field of a specified width. It is similar to Python's `str.rjust()` method.

Usage

```
strs_rjust(string, width, fillchar = " ")
```

Arguments

`string`	A character vector where each element is a string to be right-justified.
`width`	The total width of the field in which the string is to be right-justified.
`fillchar`	A character used for padding on the left.

Value

A character vector of the same length as `string`, with each element right-justified in a field of the specified width.

See Also

Python str.rjust() documentation

Examples

```
strs_rjust("hello", 10)
strs_rjust("world", 10, "*")
```

`strs_rstrip` *Right strip characters from a string*

Description

`strs_rstrip` removes trailing characters (spaces by default) from each element of a character vector. It is similar to Python's `str.rstrip()` method.

Usage

```
strs_rstrip(string, chars = NULL)
```

Arguments

`string`	A character vector where each element is a string to be right-stripped.
`chars`	An optional string of characters to be removed from the end of each element. If NULL, whitespace is removed.

Value

A character vector of the same length as `string`, with specified characters removed from the end of each element.

See Also

Python str.rstrip() documentation

Examples

```
strs_rstrip("hello world    ")
strs_rstrip("hello worldxxx", chars = "x")
```

strs_slice *Slice substrings from a string*

Description

strs_slice extracts substrings from each element of a character vector, specified by start and stop positions. It is similar to Python's slicing syntax for strings, but it uses 1 indexing and stops are inclusive.

Usage

```
strs_slice(string, start = 1L, stop = -1L, ..., step = 1L)
```

Arguments

string	A character vector where each element is a string to slice.
start	An integerish scalar for the starting position for slicing (inclusive).
stop	An integerish scalar for the ending position for slicing (inclusive).
...	Used to force keyword argument usage of step.
step	An integer greater than 0 or equal to -1 for the step size. If -1 is provided, each string will be reversed after slicing operations.

Value

A character vector of the same length as string, with each element being the sliced substring.

Examples

```
strs_slice("hello world", 1, 5)
strs_slice("hello world", 7)
strs_slice("hello world", start = 7, stop = 11)
```

strs_split *Split string into substrings*

Description

strs_split splits each element of a character vector into substrings based on a separator. It is similar to Python's str.split() method.

Usage

```
strs_split(string, sep = " ", maxsplit = -1L)
```

Arguments

`string`	A character vector to split.
`sep`	The separator on which to split the string.
`maxsplit`	The maximum number of splits to perform. If -1, all possible splits are performed.

Value

A list of character vectors, with each vector containing the split substrings from the corresponding element of `string`.

See Also

Python str.split() documentation

Examples

```
strs_split("hello world", " ")
strs_split("one,two,three", ",", maxsplit = 1)
```

`strs_splitlines` *Split string into lines*

Description

`strs_splitlines` splits each element of a character vector into separate lines. It is similar to Python's `str.splitlines()` method.

Usage

```
strs_splitlines(string, keepends = FALSE)
```

Arguments

`string`	A character vector to be split into lines.
`keepends`	A boolean indicating whether to retain line end characters.

Value

A list of character vectors, with each vector containing lines from the corresponding element of `string`.

See Also

Python str.splitlines() documentation

Examples

```
strs_splitlines("hello\nworld\n")
strs_splitlines("line1\r\nline2\n", keepends = TRUE)
```

`strs_startswith` *Check if string starts with a specified prefix*

Description

`strs_startswith` determines whether each element of a character vector starts with a specified prefix. It is similar to Python's `str.startswith()` method.

Usage

```
strs_startswith(string, prefix)
```

Arguments

`string`	A character vector where each element is a string to be checked.
`prefix`	The prefix to check for at the start of each element of `string`.

Value

A logical vector of the same length as `string`, with each element indicating whether the corresponding element of `string` starts with `prefix`.

See Also

Python str.startswith() documentation

Examples

```
strs_startswith("hello world", "hello")
strs_startswith(c("test", "hello", "world"), "te")
```

strs_strip *Strip characters from both ends of a string*

Description

strs_strip removes leading and trailing characters (spaces by default) from each element of a character vector. It is similar to Python's str.strip() method.

Usage

```
strs_strip(string, chars = NULL)
```

Arguments

string	A character vector where each element is a string to be stripped.
chars	An optional string of characters to be removed from both ends of each element. If NULL, whitespace is removed.

Value

A character vector of the same length as string, with specified characters removed from both ends of each element.

See Also

Python str.strip() documentation

Examples

```
strs_strip("    hello world    ")
strs_strip("xxxyhello worldyyy", chars = "xy")
```

strs_swapcase *Swap uppercase and lowercase characters in a string*

Description

strs_swapcase returns a copy of the string with uppercase characters convert to lowercase and visa-versa. It is similar to Python's str.swapcase().

Usage

```
strs_swapcase(string)
```

Arguments

string	A character vector where each element is a string.

Value

A character vector of the same length as `string`, with specified uppercase characters converted to lowercase and visa-versa.

See Also

Python str.swapcase() documentation

Examples

```
strs_swapcase("Hello World")
```

`strs_title` *Convert string to title case*

Description

`strs_title` converts each element of a character vector to title case, based on the specified locale. It is similar to Python's `str.title()` method.

Usage

```
strs_title(string, locale = "en")
```

Arguments

`string`	A character vector to be converted to title case.
`locale`	A character string representing the locale to be used for the conversion.

Value

A character vector of the same length as `string`, with each element converted to title case.

See Also

Python str.title() documentation

Examples

```
strs_title("hello world")
strs_title("guten tag", locale = "de")
```

strs_upper *Convert string to uppercase*

Description

strs_upper converts each element of a character vector to uppercase, based on the specified locale. It is similar to Python's str.upper() method.

Usage

```
strs_upper(string, locale = "en")
```

Arguments

string	A character vector to be converted to uppercase.
locale	A character string representing the locale to be used for the conversion.

Value

A character vector of the same length as string, with each element converted to uppercase.

See Also

Python str.upper() documentation

Examples

```
strs_upper("hello world")
strs_upper("äpfel", locale = "de")
```

Index