Package 're'

September 2, 2024

Title 'Python' Style Regular Expression Functions

Version 0.1.0

Description A comprehensive set of regular expression 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 regular expressions, (2) reduce the complexity often associated with regular expressions code, (3) and enable users to write more readable and maintainable code that relies on regular expression-based pattern matching.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports stringi

URL <https://pythonicr.github.io/re/>, <https://github.com/pythonicr/re>

BugReports <https://github.com/pythonicr/re/issues>

Config/Needs/website pythonicr/pythonicrtemplate

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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 14:30:07 UTC

Contents

Index   **11**

---

re_compile   *Create a regular expression object with specific flags*

Description

`re_compile` compiles a regular expression pattern with specified flags. This function allows setting various flags akin to regex modifiers in other programming languages like Python. The flags control various aspects of pattern matching. This function is really just a way to set flag arguments with a constant variable.

Usage

```
re_compile(pattern, IGNORECASE, I, MULTILINE, M, DOTALL, S, VERBOSE, X, NOFLAG)
```

Arguments

pattern	The regular expression pattern to be compiled.
IGNORECASE	Flag to indicate case-insensitive matching.
I	Abbreviation for IGNORECASE.
MULTILINE	Flag to indicate multi-line matching, where ^ and $ match the start and end of each line.
M	Abbreviation for MULTILINE.
DOTALL	Flag to indicate that . (dot) should match any character including newline.
S	Abbreviation for DOTALL
VERBOSE	Flag to allow a more verbose regex syntax, which can include comments and whitespace for readability.
X	Abbreviation for VERBOSE
NOFLAG	Flag to indicate that no flags should be set.

Value

An object of class "Pattern" representing the compiled regular expression with the specified flags.

See Also

[Python re.compile() documentation](#)

Examples

```
pattern <- re_compile("^abc", IGNORECASE)
pattern <- re_compile("end$", M = TRUE)
pattern <- re_compile("a.b", DOTALL = TRUE)
```

---

re_contains   *Check if string contains a regular expression*

Description

`re_contains` checks whether a specified pattern (regular expression) is found within each element of a character vector. If the provided pattern is not already a compiled pattern object, it compiles it using `re_compile`.

Usage

```
re_contains(pattern, string, ...)
```

Arguments

pattern   A regular expression pattern or a compiled pattern object.

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

...   Arguments passed on to [re_compile](#)

                 `IGNORECASE` Flag to indicate case-insensitive matching.

                 `I` Abbreviation for IGNORECASE.

                 `MULTILINE` Flag to indicate multi-line matching, where ^ and $ match the start and end of each line.

                 `M` Abbreviation for MULTILINE.

                 `DOTALL` Flag to indicate that . (dot) should match any character including newline.

                 `S` Abbreviation for DOTALL

                 `VERBOSE` Flag to allow a more verbose regex syntax, which can include comments and whitespace for readability.

                 `X` Abbreviation for VERBOSE

                 `NOFLAG` Flag to indicate that no flags should be set.

Value

A logical vector of the same length as `string`, indicating whether each element contains a match for the pattern.

Examples

```
pattern <- re_compile("^abc", IGNORECASE)
re_contains(pattern, "Abcdef")
re_contains("xyz$", "hello world xyz")
```

---

re_escape   *Escape special characters*

Description

`re_escape` escapes all special characters in a string. This function is useful when you want to treat a string literally in a regular expression context, escaping characters that would otherwise be interpreted as special regex operators.

Usage

```
re_escape(pattern)
```

Arguments

pattern   A character vector where each element is a string in which special regex characters are to be escaped.

Value

A character vector of the same length as `pattern`.

See Also

[Python re.escape() documentation](#)

Examples

```
re_escape("a[bc].*d?")
re_escape(".^$|*+?{}[]()")
```

---

re_findall   *Extract all occurrences of a pattern in a string*

Description

`re_findall` extracts all occurrences of a specified pattern (regular expression) from each element of a character vector. If the provided pattern is not already a compiled pattern object, it compiles it using `re_compile`.

Usage

```
re_findall(pattern, string, ...)
```

Arguments

pattern	A regular expression pattern or a compiled pattern object.
string	A character vector where each element is a string from which to extract matches of the pattern.
...	Arguments passed on to [re_compile](#)

> IGNORECASE Flag to indicate case-insensitive matching.
>
> I Abbreviation for IGNORECASE.
>
> MULTILINE Flag to indicate multi-line matching, where ^ and $ match the start and end of each line.
>
> M Abbreviation for MULTILINE.
>
> DOTALL Flag to indicate that . (dot) should match any character including newline.
>
> S Abbreviation for DOTALL
>
> VERBOSE Flag to allow a more verbose regex syntax, which can include comments and whitespace for readability.
>
> X Abbreviation for VERBOSE
>
> NOFLAG Flag to indicate that no flags should be set.

Value

A list of character vectors, where each vector contains all the matches found in the corresponding element of string.

See Also

[Python re.findall() documentation](#)

Examples

```
pattern <- re_compile("\\b\\w+\\b")
re_findall(pattern, "This is a test.") # Extracts all words
re_findall("\\d+", "123 abc 456")
```

---

re_fullmatch	*Match a pattern against the entire string*

Description

re_fullmatch checks whether each element of a character vector fully matches a specified pattern (regular expression). If the provided pattern is not already a compiled pattern object, it compiles it using re_compile. The function ensures that the entire string matches the pattern from start to end.

Usage

```
re_fullmatch(pattern, string, ...)
```

Arguments

pattern   A regular expression pattern or a compiled pattern object.

string   A character vector where each element is a string to be matched against the
                 pattern.

...   Arguments passed on to re_compile

                 IGNORECASE Flag to indicate case-insensitive matching.

                 I Abbreviation for IGNORECASE.

                 MULTILINE Flag to indicate multi-line matching, where ^ and $ match the start
                     and end of each line.

                 M Abbreviation for MULTILINE.

                 DOTALL Flag to indicate that . (dot) should match any character including new-
                     line.

                 S Abbreviation for DOTALL

                 VERBOSE Flag to allow a more verbose regex syntax, which can include com-
                     ments and whitespace for readability.

                 X Abbreviation for VERBOSE

                 NOFLAG Flag to indicate that no flags should be set.

Value

A list where each element is a character vector containing the full match for the corresponding
element of string, or character(0) if there is no match.

See Also

[Python re.fullmatch() documentation]

Examples

```
pattern <- re_compile("\\d{3}-\\d{2}-\\d{4}")
re_fullmatch(pattern, "123-45-6789") # Full match
re_fullmatch("123-45-6789", "123-45-6789 and more") # No full match
```

---

re_match   *Match a pattern at the start of a string*

Description

re_match checks whether each element of a character vector matches a specified pattern (regular
expression) at the start. If the provided pattern is not already a compiled pattern object, it compiles
it using re_compile. The function ensures that the matching occurs at the beginning of the string.

Usage

```
re_match(pattern, string, ...)
```

Arguments

pattern	A regular expression pattern or a compiled pattern object.
string	A character vector where each element is a string to be matched against the pattern at the beginning.
...	Arguments passed on to re_compile

IGNORECASE Flag to indicate case-insensitive matching.

I Abbreviation for IGNORECASE.

MULTILINE Flag to indicate multi-line matching, where ^ and $ match the start and end of each line.

M Abbreviation for MULTILINE.

DOTALL Flag to indicate that . (dot) should match any character including newline.

S Abbreviation for DOTALL

VERBOSE Flag to allow a more verbose regex syntax, which can include comments and whitespace for readability.

X Abbreviation for VERBOSE

NOFLAG Flag to indicate that no flags should be set.

Value

A list where each element is a character vector containing the match found at the start of the corresponding element of string, or character(0) if there is no match at the start.

See Also

Python re.match() equivalent functionality documentation

Examples

```
pattern <- re_compile("\\d{3}")
re_match(pattern, "123abc")
re_match("abc", "xyzabc")
```

---

re_search	*Search for a pattern in a string*

Description

re_search searches for occurrences of a specified pattern (regular expression) within each element of a character vector. If the provided pattern is not already a compiled pattern object, it compiles it using re_compile.

Usage

```
re_search(pattern, string, ...)
```

Arguments

`pattern`	A regular expression pattern or a compiled pattern object.
`string`	A character vector where each element is a string in which to search for the pattern.
`...`	Arguments passed on to [re_compile](#)

    `IGNORECASE` Flag to indicate case-insensitive matching.

    `I` Abbreviation for IGNORECASE.

    `MULTILINE` Flag to indicate multi-line matching, where `^` and `$` match the start and end of each line.

    `M` Abbreviation for MULTILINE.

    `DOTALL` Flag to indicate that . (dot) should match any character including newline.

    `S` Abbreviation for DOTALL

    `VERBOSE` Flag to allow a more verbose regex syntax, which can include comments and whitespace for readability.

    `X` Abbreviation for VERBOSE

    `NOFLAG` Flag to indicate that no flags should be set.

Value

A list where each element is a character vector containing all matches found in the corresponding element of `string`. If no matches are found, the element will be `character(0)`.

See Also

[Python re.search() documentation](#)

Examples

```
pattern <- re_compile("\\d+")
re_search(pattern, "abc 123 xyz") # Finds "123"
re_search("\\bword\\b", "A sentence with the word.") # Finds "word"
```

`re_split`	*Split a string by a regular expression pattern*

Description

`re_split` splits each element of a character vector into substrings based on a specified pattern (regular expression). If the provided pattern is not already a compiled pattern object, it compiles it using `re_compile`. The function allows for controlling the maximum number of splits performed.

Usage

```
re_split(pattern, string, ..., maxsplit = -1L)
```

Arguments

`pattern`	A regular expression pattern or a compiled pattern object.
`string`	A character vector where each element is a string to be split.
`...`	Arguments passed on to [re_compile](#)
	IGNORECASE Flag to indicate case-insensitive matching.
	I Abbreviation for IGNORECASE.
	MULTILINE Flag to indicate multi-line matching, where ^ and $ match the start and end of each line.
	M Abbreviation for MULTILINE.
	DOTALL Flag to indicate that . (dot) should match any character including newline.
	S Abbreviation for DOTALL
	VERBOSE Flag to allow a more verbose regex syntax, which can include comments and whitespace for readability.
	X Abbreviation for VERBOSE
	NOFLAG Flag to indicate that no flags should be set.
`maxsplit`	The maximum number of splits to perform on each string. If -1L (default), all possible splits are performed.

Value

A list of character vectors, where each vector contains the substrings resulting from splitting the corresponding element of `string`.

See Also

[Python re.split() documentation](#)

Examples

```
pattern <- re_compile("\\s+")
re_split(pattern, "Split this string") # Splits on whitespace
re_split("\\W+", "Split,with!punctuation.morestuff", maxsplit = 2)
```

---

re_sub	*Substitute occurrences of a pattern in a string*

Description

`re_sub` replaces all occurrences of a specified pattern (regular expression) in each element of a character vector with a replacement string. If the provided pattern is not already a compiled pattern object, it compiles it using `re_compile`.

Usage

```
re_sub(pattern, replacement, string, ...)
```

Arguments

pattern	A regular expression pattern or a compiled pattern object.
replacement	The replacement string.
string	A character vector where each element is a string in which the pattern will be replaced.
...	Arguments passed on to [re_compile](#)

     IGNORECASE Flag to indicate case-insensitive matching.

     I Abbreviation for IGNORECASE.

     MULTILINE Flag to indicate multi-line matching, where ^ and $ match the start and end of each line.

     M Abbreviation for MULTILINE.

     DOTALL Flag to indicate that . (dot) should match any character including newline.

     S Abbreviation for DOTALL

     VERBOSE Flag to allow a more verbose regex syntax, which can include comments and whitespace for readability.

     X Abbreviation for VERBOSE

     NOFLAG Flag to indicate that no flags should be set.

Value

A character vector of the same length as string, with all occurrences of the pattern replaced by replacement in each element.

See Also

[Python re.sub() documentation](#)

Examples

```
pattern <- re_compile("\\d+")
re_sub(pattern, "number", "Replace 123 with text.") # Replaces "123" with "number"
re_sub("\\s+", "-", "Split and join") # Replaces spaces with hyphens
```

Index