Python String Utils Documentation

Release 0.6.0

Davide Zanotti

November 08, 2016

Contents

1 Indices and tables 7

Python Module Index 9

Contents:

`string_utils.`**`is_string`**(*obj*)

Checks if an object is a string.

Parameters **`obj`** – Object to test.

Returns True if string, false otherwise.

Return type bool

`string_utils.`**`is_full_string`**(*string*)

Check if a string is not empty (it must contains at least one non space character).

Parameters **`string`** (*str*) – String to check.

Returns True if not empty, false otherwise.

`string_utils.`**`is_url`**(*string*, *allowed_schemes=None*)

Check if a string is a valid url.

Parameters

- **`string`** – String to check.
- **`allowed_schemes`** – List of valid schemes ('http', 'https', 'ftp'...). Default to None (any scheme is valid).

Returns True if url, false otherwise

Return type bool

`string_utils.`**`is_email`**(*string*)

Check if a string is an email.

IMPORTANT NOTES:

By design, the implementation of this checking does not follow the specification for a valid email address, but instead it's based on real world cases in order to match more than 99% of emails and catch user mistakes. For example the percentage sign "%" is a valid sign for an email, but actually no one use it, instead if such sign is found in a string coming from user input (like a web form) is very likely that the intention was to type "5" (which is on the same key on a US keyboard).

You can take a look at "**IsEmailTestCase**" in tests.py for further details.

Parameters **`string`** (*str*) – String to check.

Returns True if email, false otherwise.

Return type bool

`string_utils.`**`is_credit_card`**(*string*, *card_type=None*)

Checks if a string is a valid credit card number. If card type is provided then it checks that specific type, otherwise any known credit card number will be accepted.

Parameters

- **`string`** (*str*) – String to check.
- **`card_type`** (*str*) – Card type.

Can be one of these:

- VISA
- MASTERCARD
- AMERICAN_EXPRESS
- DINERS_CLUB
- DISCOVER
- JCB

or None. Default to None (any card).

Returns True if credit card, false otherwise.

Return type bool

`string_utils.`**`is_camel_case`**(*string*)

Checks if a string is formatted as camel case. A string is considered camel case when:

- it's composed only by letters ([a-zA-Z]) and optionally numbers ([0-9])
- it contains both lowercase and uppercase letters
- it does not start with a number

Parameters **`string`** (*str*) – String to test.

Returns True for a camel case string, false otherwise.

Return type bool

`string_utils.`**`is_snake_case`**(*string*, *separator='_'*)

Checks if a string is formatted as snake case. A string is considered snake case when:

- it's composed only by lowercase letters ([a-z]), underscores (or provided separator) and optionally numbers ([0-9])
- it does not start/end with an underscore (or provided separator)
- it does not start with a number

Parameters

- **`string`** (*str*) – String to test.
- **`separator`** (*str*) – String to use as separator.

Returns True for a snake case string, false otherwise.

Return type bool

`string_utils.`**`is_json`**(*string*)

Check if a string is a valid json.

Parameters **`string`** (*str*) – String to check.

Returns True if json, false otherwise

Return type bool

`string_utils.`**`is_uuid`**(*string*)

Check if a string is a valid UUID.

Parameters **`string`** (*str*) – String to check.

Returns True if UUID, false otherwise

Return type bool

`string_utils.is_ip(string)`

Checks if a string is a valid ip.

Parameters **string** (*str*) – String to check.

Returns True if an ip, false otherwise.

Return type bool

`string_utils.is_palindrome(string, strict=True)`

Checks if the string is a palindrome (https://en.wikipedia.org/wiki/Palindrome).

Parameters

- **string** (*str*) – String to check.
- **strict** (*bool*) – True if white spaces matter (default), false otherwise.

Returns True if the string is a palindrome (like "otto", or "i topi non avevano nipoti" if strict=False),

False otherwise

`string_utils.is_pangram(string)`

Checks if the string is a pangram (https://en.wikipedia.org/wiki/Pangram).

Parameters **string** (*str*) – String to check.

Returns True if the string is a pangram, False otherwise.

`string_utils.is_isogram(string)`

Checks if the string is an isogram (https://en.wikipedia.org/wiki/Isogram).

Parameters **string** (*str*) – String to check.

Returns True if isogram, false otherwise.

`string_utils.is_slug(string, sign='-')`

Checks if a given string is a slug.

Parameters

- **string** (*str*) – String to check.
- **sign** (*str*) – Join sign used by the slug.

Returns True if slug, false otherwise.

`string_utils.words_count(string)`

Returns the number of words contained into the given string.

This method is smart, it does consider only sequence of one or more letter and/or numbers as "words", so a string like this: "! @ # % ... []" will return zero! Moreover it is aware of punctuation, so the count for a string like "one,two,three.stop" will be 4 not 1 (even if there are no spaces in the string).

Parameters **string** (*str*) – String to check.

Returns Number of words.

Return type int

`string_utils.contains_html(string)`

Checks if the given string contains html code. By design, this function is very permissive regarding what to consider html code, don't expect to use it as an html validator, its goal is to detect "malicious" or undesired html tags in the text.

Parameters **string** (*str*) – Text to check

Returns True if string contains html, false otherwise.

Return type bool

`string_utils.`**`camel_case_to_snake`**(*string*, *separator='_'*)

Convert a camel case string into a snake case one. (The original string is returned if is not a valid camel case string)

Parameters

- **string** (*str*) – String to convert.
- **separator** (*str*) – Sign to use as separator.

Returns Converted string.

Return type str

`string_utils.`**`snake_case_to_camel`**(*string*, *upper_case_first=True*, *separator='_'*)

Convert a snake case string into a camel case one. (The original string is returned if is not a valid snake case string)

Parameters

- **string** (*str*) – String to convert.
- **upper_case_first** (*bool*) – True to turn the first letter into uppercase (default).
- **separator** (*str*) – Sign to use as separator (default to "_").

Returns Converted string

Return type str

`string_utils.`**`reverse`**(*string*)

Returns the string reversed ("abc" -> "cba").

Parameters **string** (*str*) – String to revert.

Returns Reversed string.

Return type str

`string_utils.`**`uuid`**()

Generated an UUID string (using uuid.uuid4()).

Returns uuid string.

Return type str

`string_utils.`**`shuffle`**(*string*)

Return a new string containing shuffled items.

Parameters **string** (*str*) – String to shuffle

Returns Shuffled string

Return type str

`string_utils.`**`strip_html`**(*string*, *keep_tag_content=False*)

Remove html code contained into the given string.

Parameters

- **string** (*str*) – String to manipulate.

- **keep_tag_content** (*bool*) – True to preserve tag content, False to remove tag and its content too (default).

Returns String with html removed.

Return type str

`string_utils.prettify(string)`

Turns an ugly text string into a beautiful one by applying a regex pipeline which ensures the following:

•String cannot start or end with spaces

•String cannot have multiple sequential spaces, empty lines or punctuation (except for "?", "!" and ".")

•Arithmetic operators (+, -, /, *, =) must have one, and only one space before and after themselves

•The first letter after a dot, an exclamation or a question mark must be uppercase

•One, and only one space should follow a dot, an exclamation or a question mark

•Text inside double quotes cannot start or end with spaces, but one, and only one space must come first and after quotes (foo" bar"baz -> foo "bar" baz)

•Text inside round brackets cannot start or end with spaces, but one, and only one space must come first and after brackets ("foo(bar)baz" -> "foo (bar) baz")

•Percentage sign ("%") cannot be preceded by a space if there is a number before ("100 %" -> "100%")

•Saxon genitive is correct ("Dave' s dog" -> "Dave's dog")

Parameters **string** – String to manipulate

Returns Prettified string.

Return type str

`string_utils.slugify(string, sign='-')`

Converts a string into a slug using provided join sign. ((**This Is A "Test"!**) -> **this-is-a-test**)

Parameters

- **string** (*str*) – String to convert.
- **sign** (*str*) – Sign used to join string tokens (default to "-").

Returns Slugified string

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

string_utils, 1

C

camel_case_to_snake() (in module string_utils), 4
contains_html() (in module string_utils), 3

I

is_camel_case() (in module string_utils), 2
is_credit_card() (in module string_utils), 1
is_email() (in module string_utils), 1
is_full_string() (in module string_utils), 1
is_ip() (in module string_utils), 3
is_isogram() (in module string_utils), 3
is_json() (in module string_utils), 2
is_palindrome() (in module string_utils), 3
is_pangram() (in module string_utils), 3
is_slug() (in module string_utils), 3
is_snake_case() (in module string_utils), 2
is_string() (in module string_utils), 1
is_url() (in module string_utils), 1
is_uuid() (in module string_utils), 2

P

prettify() (in module string_utils), 5

R

reverse() (in module string_utils), 4

S

shuffle() (in module string_utils), 4
slugify() (in module string_utils), 5
snake_case_to_camel() (in module string_utils), 4
string_utils (module), 1
strip_html() (in module string_utils), 4

U

uuid() (in module string_utils), 4

W

words_count() (in module string_utils), 3