**Métodos de *String* em Python**

str.**capitalize**()

Retorna primeira letra em maiúscula e o resto em minúscula.

str.**casefold**()

Deixa tudo em minúsculas, até letras estrangeiras.

str.**center**(*width*[, *fillchar*])

Retorna uma string centralizada. Recebe dois parâmetros(width e fillchar).

str.**count**(*sub*[, *start*[, *end*]])

Retorna a contagem de um termo ou letra na string.

str.**encode**(*encoding="utf-8"*, *errors="strict"*)

Retorna uma string codificada em bytes. O parâmetro enconding é o padrão utf-8, e o parâmetro errors podem ser: strict, ignore, replace, 'xmlcharrefreplace' e 'backslashreplace'.

str.**endswith**(*suffix*[, *start*[, *end*]])

Retorna um valor booleano: True se a string termina com o sufixo dado ou Falso se não.

str.**expandtabs**([*tabsize*])

Substitui os tabs por espaços.

str.**find**(*sub*[, *start*[, *end*]])

Retorna o número da posição da substring a ser achada.

str.**format**( )

Retorna uma cópia da string em que cada campo de substituição é substituído pelo valor da string do argumento correspondente: >>> ‘A soma de 1 + 2 é { }” .format(1+2)

str.**format\_map**(*mapping*)

Semelhante ao método anterior, contudo usado diretamente ao invés de copiado para um dicionário.

>>> ‘{nome} nasceu na {pais}’ .format\_map(nome=’Guido’, pais=’Italia’)

str.**index**(*sub*[, *start*[, *end*]])

Como o método find() mas retorna um erro ValueError quando a substring não é achada.

str.**isalnum**()

Retorna True se todos os caracteres são alfanuméricos.

str.**isalpha**()

Retorna True se todos os caracteres são alfabéticos.

str.**isdecimal**()

Retorna True se todos os caracteres são decimais.

str.**isdigit**()

Retorna True se todos os caracteres são dígitos.

str.**isidentifier**()

Retorna True se a string é um identificador válido de acordo com a língua definida.

str.**islower**()

Retorna True se todos os caracteres são minúsculos.

str.**isnumeric**()

Retorna True se todos os caracteres são numéricos.

str.**isprintable**()

Retorna True se todos os caracteres são imprimíveis ou a string estiver vazia.

str.**isspace**()

Retorna True se há somente espaços na string.

str.**istitle**()

Retorna True se a string tem Maiúsculas iniciais.

str.**isupper**()

Retorna True se todos os caracteres são maiúsculos.

str.**join**(*iterable*)

Concatena strings.

str.**ljust**(*width*[, *fillchar*])

Justitica à esquerda.

str.**lower**()

Converte em minúsculas.

str.**lstrip**([*chars*])

Remove os espaços à esquerda da string.

str.**maketrans**(*x*[, *y*[, *z*]])

Retorna uma tabela de tradução para usar com o método str.translate(). Se houver apenas um parâmetro, deve ser um dicionário {(‘y’:’x’)} mapeando integers ou strings. As chaves são convertidas em ordinais.

str.**partition**(*sep*)

Separa a string na primeira ocorrência do parâmetro sep e retorna uma tupla contendo a parte anterior ao separador, o separador e a parte posterior ao separador.

str.**replace**(*old*, *new*[, *count*])

Substitui termos ou letras passados como parâmetros old e new.

str.**rfind**(*sub*[, *start*[, *end*]])

Retorna o índice mais alto na string em que a substring sub é encontrada, de modo que sub esteja contido em s[start:end] . Os argumentos opcionais start e end são interpretados como em notação de fatia. Retorna -1 em caso de falha.

str.**rindex**(*sub*[, *start*[, *end*]])

Como [rfind(),](https://docs.python.org/3.3/library/stdtypes.html?highlight=split" \l "str.rfind" \o "str.rfind) mas gera [ValueError](https://docs.python.org/3.3/library/exceptions.html" \l "ValueError" \o "ValorErro) quando a substring sub não é encontrada.

str.**rjust**(*width*[, *fillchar*])

Justifica à direita.

str.**rpartition**(*sep*)

Divide a string na última ocorrência de sep e retorne uma tupla de 3 contendo a parte antes do separador, o próprio separador e a parte após o separador. Se o separador não for encontrado, retorne uma tupla 3 contendo duas strings vazias, seguidas pela própria string.

str.**rsplit**(*sep=None*, *maxsplit=-1*)

Retorna uma lista das palavras na string, usando sep como a string delimitadora. Se a *divisão* máxima for fornecida, no máximo *as divisões* de divisão máxima são feitas, as mais à direita . Se sep não for especificado ou None , qualquer string de espaço em branco será um separador. Exceto para dividir da direita, [rsplit()](https://docs.python.org/3.3/library/stdtypes.html?highlight=split" \l "str.rsplit" \o "str.rsplit) se comporta como [split()](https://docs.python.org/3.3/library/stdtypes.html?highlight=split" \l "str.split" \o "str.split) que é descrito em detalhes abaixo.

str.**rstrip**([*chars*])

Remove os espaços à direita.

str.**split**(*sep=None*, *maxsplit=-1*)

Divide uma string segundo o parâmetro de separador dado. Sep é caractere separador, e maxsplit é o delimitador de quantidade de separação.

str.**splitlines**([*keepends*])

Retorna uma lista das linhas na string, quebrando nos limites da linha. Este método usa a abordagem [universal](https://docs.python.org/3.3/glossary.html#term-universal-newlines) de novas linhas para dividir linhas de ting. As quebras de linha não são incluídas na lista resultante, a menos que keepends seja fornecido e verdadeiro. Por exemplo, 'ab c\n\nde fg\rkl\r\n'. split lines() retorna ['ab c', '', 'de fg', 'kl'] , enquanto a mesma chamada com split lines(True) retorna ['ab c\n', '\n', 'de fg\r', 'kl\r\n'] . Ao contrário de [split](https://docs.python.org/3.3/library/stdtypes.html?highlight=split" \l "str.split" \o "str.split)[()](https://docs.python.org/3.3/library/stdtypes.html?highlight=split#str.split) quando uma string delimitadora sep é fornecida, esse método retorna uma lista vazia para a string vazia e uma quebra de linha terminal não resulta em uma linha extra.

str.**startswith**(*prefix*[, *start*[, *end*]])

Retorna True se a string inicia com o prefixo dado como parâmetro.

str.**strip**([*chars*])

Retorne uma cópia da string com os caracteres iniciais e finais removidos. O argumento chars é uma string que especifica o conjunto de caracteres a ser removido. Se omitido ou None , o argumento chars assume como padrão a remoção de espaços em branco. O argumento chars não é um prefixo ou sufixo; em vez disso, todas as combinações de seus valores são removidas:

str.**swapcase**()

Retorna uma cópia da string com caracteres maiúsculos convertidos em minúsculos e vice-versa. Observe que não é necessariamente verdade que s.swapcase().swapcase() == s .

str.**title**()

Retorna uma versão em maiúsculas da string em que as palavras começam com um caractere maiúsculo e os caracteres restantes são minúsculos. O algoritmo usa uma definição simples independente de linguagem de uma palavra como grupos de letras consecutivas. A definição funciona em muitos contextos, mas significa que apóstrofos em contrações e possessivos formam limites de palavras, o que pode não ser o resultado desejado:

str.**translate**(*map*)

Retorna uma cópia do s onde todos os caracteres foram mapeados através do mapa que deve ser um dicionário de ordinais Unicode (inteiros) para ordinais Unicode, strings ou None . Caracteres não mapeados são deixados intocados. Os caracteres mapeados para Nenhum são excluídos.

Você pode usar [str.maketrans()](https://docs.python.org/3.3/library/stdtypes.html?highlight=split" \l "str.maketrans" \o "str.maketrans) para criar um mapa de tradução de mapeamentos de caractere para caractere em diferentes formatos.

str.**upper**()

Retorna uma cópia da string com todos os caracteres em maiúsculas [[4]](https://docs.python.org/3.3/library/stdtypes.html?highlight=split#id13) convertidos em maiúsculas. Observe que str.upper().isupper() pode ser False se s contiver caracteres sem maiúsculas ou se a categoria Unicode do(s) caractere(s) resultante(s) não for “Lu” (Letter, maiúsculo), mas, por exemplo, “Lt” (Letter, caixa de título).

str.**zfill**(*width*)

Retorna a string numérica preenchida com zeros à esquerda em uma string de comprimento largura . Um prefixo de sinal é tratado corretamente. A string original é retornada se width for menor ou igual a len(s) .