**41 - string methods – split, strip, enumerate e join**

**<str>.split(sep = ‘ ‘, maxsplit = ‘-1’ )**

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

str.**split**(*sep=Whitespace*, *maxsplit=- 1*)

Return a list of the words in the string, using *sep* as the delimiter string. If *maxsplit* is given, at most *maxsplit* splits are done (thus, the list will have at most maxsplit+1 elements). If *maxsplit* is not specified or -1, 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 (for example, '1,,2'.split(',') returns ['1', '', '2']). The *sep* argument may consist of multiple characters (for example, '1<>2<>3'.split('<>') returns ['1', '2', '3']). Splitting an empty string with a specified separator returns [''].

For example:

>>>

**>>>** '1,2,3'.split(',')

['1', '2', '3']

**>>>** '1,2,3'.split(',', maxsplit=1)

['1', '2,3']

**>>>** '1,2,,3,'.split(',')

['1', '2', '', '3', '']

If *sep* is not specified or is None, a different splitting algorithm is applied: runs of consecutive whitespace are regarded as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace. Consequently, splitting an empty string or a string consisting of just whitespace with a None separator returns [].

For example:

>>>

**>>>** '1 2 3'.split()

['1', '2', '3']

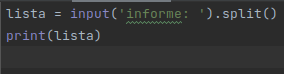
**>>>** '1 2 3'.split(maxsplit=1)

['1', '2 3']

**>>>** ' 1 2 3 '.split()

['1', '2', '3'

Podemos usar split com a função input para obter múltiplas entradas em uma única linha de código:





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

Return a copy of the string with the leading and trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix or suffix; rather, all combinations of its values are stripped:

>>>

**>>>** ' spacious '.strip()

'spacious'

**>>>** 'www.example.com'.strip('cmowz.')

'example'

The outermost leading and trailing *chars* argument values are stripped from the string. Characters are removed from the leading end until reaching a string character that is not contained in the set of characters in *chars*. A similar action takes place on the trailing end. For example:

>>>

**>>>** comment\_string = '#....... Section 3.2.1 Issue #32 .......'

**>>>** comment\_string.strip('.#! ')

'Section 3.2.1 Issue #32'

str.**join**(*iterable*)[¶](https://docs.python.org/3/library/stdtypes.html#str.join)

Return a string which is the concatenation of the strings in iterable. A [TypeError](https://docs.python.org/3/library/exceptions.html#TypeError) will be raised if there are any non-string values in iterable, including [bytes](https://docs.python.org/3/library/stdtypes.html#bytes) objects. The separator between elements is the string providing this method.

