Bài đọc thêm: Phương thức thông dụng của Chuỗi ký tự trong Python

Sử dụng thư viện String ta cần lệnh:

import String

`str.`**`capitalize`**()

Trả về 1 chuỗi bản sao của chuỗi ban đầu, với ký tự đầu tiên được viết hoa, các ký tự sau viết thường.

`str.`**`center`**(*width*[, *fillchar*])

Trả về 1 chuỗi bản sao của chuỗi ban đầu, thêm các ký tự *fillchar (mặc định là khoảng trắng)* vào 2 bên nội dung chuỗi ban đầu sao cho chuỗi mới có chiều dài là width. Nếu width ≤ len(str) thì trả về chuỗi gốc.

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

Đếm số lần xuất hiện không chồng lên nhau của chuỗi sub trong đoạn từ start đến end của chuỗi nguồn.

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

Trả về True nếu chuỗi kết thúc trùng với suffix, ngược lại trả về False

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

Trả về True nếu chuỗi bắt đầu trùng với suffix, ngược lại trả về False

`str.`**`expandtabs`**(*tabsize=8*)

Trả về 1 chuỗi bản sao của chuỗi ban đầu, thay thế các ký tự '\n' trong chuỗi ban đầu bằng khoảng trắng có kích thước bằng với tabsize.

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

Trả về vị trí tìm được đầu tiên của chuỗi sub trong khoảng start đến end của chuỗi ban đầu. Trả về -1 nếu không tìm thấy.

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

Tương tự như find(), như tìm bắt đầu từ cuối chuỗi.

`str.`**`format`**(**args*, ***kwargs*)

Định dạng chuỗi bằng cách thay thế các dấu {} trong chuỗi bằng tham số truyền vào. Xem chi tiết tại bài đọc thêm “Chuỗi ký tự”

`str.`**`index`**(*sub*[, *start*[, *end*]])
Tương tự str.find(), nhưng sẽ báo lỗi ValueError khi không tìm thấy sub.

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

Tương tự như index(), nhưng bắt đầu tìm từ cuối chuỗi.

`str.`**`isalnum`**()
Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là chữ số hoặc chữ cái, ngược lại trả về False.

`str.`**`isalpha`**()

Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là ký tự chữ cái, ngược lại trả về False.

`str.`**`isdecimal`**()

Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là số thập phân, ngược lại trả về False.

`str.`**`isdigit`**()

Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là chữ số, ngược lại trả về False.

`str.`**`islower`**()

Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là ký tự chữ cái thường, ngược lại trả về False.

`str.`**`isupper`**()

Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là ký tự chữ cái in hoa, ngược lại trả về False.

`str.`**`isnumeric`**()

Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là chữ số hoặc hỗ trợ biểu diễn số, ví dụ: U+2155, ngược lại trả về False.

`str.`**`isprintable`**()

Trả về True nếu chuỗi rỗng hoặc tất cả ký tự trong chuỗi đều có thể in được, ngược lại trả về False.

`str.`**`isspace`**()

Trả về True nếu chuỗi có ít nhất một ký tự và tất cả các ký tự trong chuỗi là khoảng trắng, ngược lại trả về False.

`str.`**`lower`**()

Trả về chuỗi bản sao của chuỗi ban đầu, với tất cả ký tự chữ cái được đưa về in thường.

`str.`**`upper`**()

Trả về chuỗi bản sao của chuỗi ban đầu, với tất cả ký tự chữ cái được đưa về in hoa.

`str.`**`lstrip`**([*chars*])
Trả về bản sao của chuỗi ban đầu, với các ký tự của chuỗi trùng với ký tự trong chars sẽ bị loại bỏ. Nếu không có tham số truyền vào, mặc định loại bỏ khoảng trắng ở đầu. Ví dụ:

```
>>> '   spacious   '.lstrip()
'spacious   '
>>> 'www.example.com'.lstrip('cmowz.')
'example.com'
```

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

Tương tự lstrip(), nhưng bắt đầu từ cuối chuỗi.

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

Cắt chuỗi với ký hiệu cắt sep, Trả về một 3-tuple, phần tử đầu tiên của tuple là phần chuỗi trước sep, phần tử thứ hai là sep, phần tử thứ ba là phần chuỗi sau sep.

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

Trả về chuỗi bản sao của chuỗi ban đầu, đã được thay count ký tự old đầu tiên bằng các ký tự new.

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

Tách chuỗi ban đầu thành các chuỗi con, ký hiệu đánh dấu cắt là sep, số lần tối đa cần cắt là maxsplit.

Ví dụ:

```
>>> '1,2,3'.split(',')
['1', '2', '3']
>>> '1,2,3'.split(',', maxsplit=1)
['1', '2,3']
>>> '1,2,,3,'.split(',')
['1', '2', '', '3', '']
```

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

Trả về bản sao của chuỗi ban đầu, đã xóa các ký tự ở đầu và cuối chuỗi mà trùng với ký tự trong tham số chars. Nếu không có tham số, sẽ xóa các khoảng trắng đầu và cuối chuỗi.

Ví dụ:

```
>>> '   spacious   '.strip()
'spacious'
>>> 'www.example.com'.strip('cmowz.')
'example'
```