**Các phương thức tách chuỗi**

**Phương thức split**

**Cú pháp:**

<chuỗi>.**split(sep=None, maxsplit=-1)**

**Công dụng:** Trả về một list (kiểu dữ liệu sẽ được Kteam giới thiệu ở bài [KIỂU DỮ LIỆU LIST](http://www.howkteam.vn/course/kieu-du-lieu-list-trong-python--phan-2/kieu-du-lieu-list-trong-python--phan-1-1548)) bằng cách chia các phần tử bằng kí tự **sep**.

* Nếu **sep**mặc định bằng None thì sẽ dùng kí tự khoảng trắng.
* Nếu **maxsplit**được mặc định bằng **-1**, Python sẽ không bị giới hạn việc tách, còn không, Python sẽ tách với số lần được cung cấp thông qua maxsplit.

**Ví dụ:**

>>> 'How Kteam K9'.split()

['How', 'Kteam', 'K9']

>>> 'How Kteam K9'.split(maxsplit=1)

['How', 'Kteam K9']

>>> 'How--Kteam--K9'.split('--')

['How', 'Kteam', 'K9']

>>> 'How&Kteam&K9'.split('&')

['How', 'Kteam', 'K9']

**Phương thức rsplit**

**Cú pháp:**

<chuỗi>.**split(sep=None, maxsplit=-1)**

|  |
| --- |
|  |

**Công dụng:**cũng hoàn toàn như phương thức split, có điều là việc tách từ bên **phải sang trái**

**Ví dụ:**

>>> 'How kteam EDUCATION'.rsplit()

['How', 'kteam', 'EDUCATION']

>>> 'How kteam EDUCATION'.rsplit(maxsplit=1)

['How kteam', 'EDUCATION']

**Phương thức splitlines**

<chuỗi>.**splitlines(keppends=False)**

**Công dụng**: hoàn toàn giống với phương thức split, nhưng các phần tử được chia tách bằng “**\n**” (xuống dòng). Nếu keppends được cung cấp giá trị True, các phần tử khi được phân tách cũng sẽ có kí tự “\n” theo sau.

**Ví dụ:**

>>> 'a\nb\nc\nd\ne'.splitlines()

['a', 'b', 'c', 'd', 'e']

>>> 'a\nb\nc\nd\ne'.splitlines(**True**)

['a\n', 'b\n', 'c\n', 'd\n', 'e']

**Phương thức partition**

**Cú pháp:**

<chuỗi>.**partition(sep)**

**Công dụng:**Trả về một tuple với 3 phần tử. Các phần tử đó lần lượt là chuỗi **trước chuỗi sep**, **sep**và **chuỗi sau sep**.

* Trong trường hợp **không tìm thấy sep** trong chuỗi, mặc định trả về giá trị đầu tiên là chuỗi ban đầu và 2 giá trị kế tiếp là chuỗi rỗng.

**Ví dụ:**

>>> 'How kteam vs I hate python team vs Education'.partition('vs')

('How kteam ', 'vs', ' I hate python team vs Education')

>>> 'How kteam vs I hate python team vs Education'.partition('VS')

('How kteam vs I hate python team vs Education', '', '')

**Phương thức rpartition**

**Cú pháp:**

<chuỗi>.**rpartition(sep)**

**Công dụng:** Cách phân chia giống như phương thức partition nhưng lại chia từ **phải qua trái**. Và với **sep**không có trong chuỗi thì sẽ trả về 2 giá trị đầu tiên là chuỗi rỗng và cuối cùng là chuỗi ban đầu.

**Ví dụ:**

>>> 'How kteam vs I hate python team vs free Education'.rpartition('vs')

('How kteam vs I hate python team ', 'vs', ' free Education')

>>> 'How kteam vs I hate python team vs free Education'.rpartition('VS')

('', '', 'How kteam vs I hate python team vs free Education')

**Các phương thức tiện ích**

**Phương thức count**

**Cú pháp:**

<chuỗi>.**count(sub, [start, [end]])**

**Công dụng:**Trả về một số nguyên, chính là số lần xuất hiện của **sub**trong chuỗi. Còn **start**và **end**là số kĩ thuật slicing (lưu ý không hề có bước).

**Ví dụ:**

>>> 'kkkkk'.count('k')

5

>>> 'kkkkk'.count('kk')

2

>>> 'kkkkk'.count('k', 3)

2

>>> 'kkkkk'.count('k', 3, 4)

1

**Phương thức startswith**

**Cú pháp:**

<chuỗi>.**startswith(prefix[, start[, end]])**

**Công dụng:** Trả về  giá trị **True**nếu chuỗi đó bắt đầu bằng chuỗi **prefix**. Ngược lại là **False**.

* Hai yếu tố **start, end**tượng trưng cho việc slicing (không có bước) để kiểm tra với chuỗi slicing đó.

**Ví dụ:**

>>> 'how kteam free education'.startswith('ho')

**True**

>>> 'how kteam free education'.startswith('ha')

**False**

>>> 'how kteam free education'.startswith('ho', 4)

**False**

**Phương thức endswith**

**Cú pháp:**

<chuỗi>.**endswith(prefix[, start[, end]])**

**Công dụng:** Trả về  giá trị **True**nếu chuỗi đó kết thúc bằng chuỗi **prefix**. Ngược lại là **False**.

* Hai yếu tố**start end**tượng trưng cho việc slicing (không có bước) để kiểm tra với chuỗi slicing đó.

**Ví dụ:**

>>> 'how kteam free education'.endswith('n')

**True**

>>> 'how kteam free education'.endswith('ho')

**False**

>>> 'how kteam free education'.endswith('n', 0, 9)

**False**

**Phương thức find**

**Cú pháp:**

<chuỗi>.**find(sub[, start[, end]])**

**Công dụng:**Trả về một số nguyên, là vị trí đầu tiên của sub khi dò từ**trái sang phải** trong chuỗi. Nếu **sub**không có trong chuỗi, kết quả sẽ là **-1**. Vẫn như các phương thức khác, **start end**đại diện cho slicing và ta sẽ tìm trong chuỗi slicing này.

**Ví dụ:**

>>> 'howkteam'.find('h')

0

>>> 'howkteam'.find('k')

3

>>> 'howkteam'.find('l')

-1

>>> 'howkteam'.find('h', 2)

-1

**Phương thức rfind**

**Cú pháp:**

<chuỗi>.**rfind(sub[, start[, end]])**

**Công dụng:**Tương tự phương thức find nhưng tìm từ **phải sang trái**

**Ví dụ:**

>>> 'howkteamhow'.rfind('h')

8

**Phương thức index**

**Cú pháp:**

<chuỗi>.**index(sub[, start[, end]])**

**Công dụng:**Tương tự phương thức find. Nhưng khác biệt là sẽ có lỗi **ValueError** nếu không tìm thấy chuỗi **sub**trong chuỗi ban đầu

**Ví dụ:**

>>> 'abcd'.index('z')

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

ValueError: substring **not** found

**Phương thức rindex**

**Cú pháp:**

<chuỗi>.**rindex(sub[, start[, end]])**

**Công dụng:** Tương tự phương thức rindex. Và cũng khác ở điểm là sẽ có **ValueError**nếu không tìm thấy chuỗi **sub**trong chuỗi ban đầu

**Ví dụ:**

>>> 'abcd'.rindex('z')

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

ValueError: substring **not** found

**Các phương thức xác thực**

**Phương thức islower**

**Cú pháp:**

<chuỗi>.**islower()**

**Công dụng:** Trả về **True**nếu tất cả các kí tự trong chuỗi đều là viết thường. Ngược lại là **False**

**Ví dụ:**

>>> 'python'.islower()

**True**

>>> 'pythoN'.islower()

**False**

**Phương thức isupper**

**Cú pháp:**

<chuỗi>.**isupper()**

**Công dụng:** Trả về **True**nếu tất cả các kí tự trong chuỗi đều là viết hoa. Ngược lại là **False**

**Ví dụ:**

>>> 'HOWKTEAM'.isupper()

**True**

>>> 'HowKteam'.isupper()

**False**

**Phương thức istitle**

**Cú pháp:**

<chuỗi>.**istitle()**

**Công dụng:**Trả về **True**nếu chuỗi đó là một dạng title. Ngược lại là **False**

**Ví dụ:**

>>> 'Free Education'.istitle()

**True**

>>> 'FrEe Education'.istitle()

**False**

**Phương thức isidentifier**

Định danh là một tên xác định một đối tượng duy nhất hay một lớp duy nhất của đối tượng, trong đó "đối tượng" hay lớp có thể là một ý tưởng, một đối tượng vật lý (đếm được), hay vật chất vật lý (không đếm được) (Theo Wikipedia).

Python hỗ trợ một phương thức giúp xác định xem một chuỗi có phải là một định danh hay không. Đó là phương thức isidentifier

**Cú pháp:**

<chuỗi>.**isidentifier()**

**Công dụng:**  Giúp xác định xem một chuỗi có phải là một định danh hay không.

Phương thức isidentifier trả về True khi cả ba điều kiện sau được thỏa mãn:

* Chuỗi phải được bắt đầu bằng dấu gạch dưới (\_) hoặc các kí tự chữ cái
* Chuỗi không được chứa bất kì khoảng trắng nào
* Không được chứa các kí tự đặc biệt (\_, %, $, \_...) ngoại trừ việc kí tự đầu tiên có thể là dấu gạch dưới.

**Ví dụ:**

>>> '\_Client'.isidentifier()

**True**

>>> 'code'.isidentifier()

**True**

>>> '@ads'.isidentifier()

**False**

>>> '25%'.isidentifier()

**False**

**Phương thức isdigit**

**Cú pháp:**

<chuỗi>.**isdigit()**

**Công dụng:**Trả về **True**nếu tất cả các kí tự trong chuỗi đều là những con số từ 0 đến 9

**Lưu ý:**Phương thức này gần giống với **isnumeric**. Nhưng vì liên quan nhiều đến toán nên Kteam sẽ không giới thiệu về phương thức isnumeric và cũng không so sánh sự khác nhau giữa hai phương thức.

**Ví dụ:**

>>> '0123'.isdigit()

**True**

>>> '123'.isdigit()

**True**

>>> '-123'.isdigit()

**False**

**Phương thức isspace**

**Cú pháp:**

<chuỗi>.**isspace()**

**Công dụng:** Trả về **True**nếu tất cả các kí tự trong chuỗi đều là kí tự khoảng trắng

**Ví dụ:**

>>> ' '.isspace()

**True**

>>> ' d '.isspace()

**False**

**Phương thức iskeyword (Thuộc thư viện keyword)**

**Cú pháp:**

import keyword

keyword.iskeyword(<chuỗi>)

**Công dụng**: Trả về True nếu chuỗi đó tương ứng với một từ khóa.

**Ví dụ:**

>>> **import** keyword # Khai báo thư viện keyword

>>> keyword.iskeyword('def')

**True**

>>> keyword.iskeyword('class')

**True**

>>> keyword.iskeyword('clas')

**False**