Bài 11: KIỂU DỮ LIỆU CHUỖI TRONG PYTHON (Phần 5 – Các phương thức chuỗi)

Xem bài học trên website để ủng hộ Kteam: KIỂU DỮ LIỆU CHUỖI TRONG PYTHON (Phần 5 – Các phương thức chuỗi)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Trong bài trước, Kteam đã giới thiệu một vài các PHƯƠNG THỨC CHUỖI TRONG PYTHON.

Ở bài này, chúng ta sẽ tiếp tục tìm hiểu thêm một số phương thức của **KIỂU DỮ LIỆU CHUỖI** trong Python

---

Nội dung

Để đọc hiểu bài này tốt nhất bạn cần:

- Cài đặt sẵn MÔI TRƯỜNG PHÁT TRIỂN CỦA PYTHON.
- Xem qua bài CÁCH CHẠY CHƯƠNG TRÌNH PYTHON.

- Nắm CÁCH GHI CHÚ và BIẾN TRONG PYTHON.
- KIỂU DỮ LIỆU SỐ và KIỂU DỮ LIỆU CHUỖI trong Python.

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Các phương thức tách chuỗi
- Các phương thức tiện ích
- Các phương thức xác thực

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) 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.

```
>>> '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**

```
>>> 'How kteam EDUCATION'.rsplit()
['How', 'kteam', 'EDUCATION']
>>> 'How kteam EDUCATION'.rsplit(maxsplit=1)
['How kteam', 'EDUCATION']
```

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.

```
>>> '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

```
>>> '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).

```
>>> '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.

```
>>> '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 **Flase**.

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

```
>>> '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.

```
>>> '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**

```
>>> '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

```
>>> '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

```
>>> '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**

```
>>> '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**

```
>>> '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**

```
>>> 'Free Education'.istitle()
True
>>> 'FrEe Education'.istitle()
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.

```
>>> '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

```
>>> '     '.isspace()
True
>>> '  d  '.isspace()
False
```

Câu hỏi củng cố

Với chuỗi s bên dưới

```
>>> s = 'aaaAAaaaooaaneu mot Ngay naO Doaaaaaaa'
```

Hãy dùng các phương thức để có được chuỗi s sau đây

```
>>> s
'Neu Mot Ngay Nao Do'
```

Hãy cố gắng làm càng ít dòng code càng tốt.

Đáp án của phần này sẽ được trình bày ở bài tiếp theo.

Tuy nhiên, Kteam khuyến khích bạn tự trả lời các câu hỏi để củng cố kiến thức cũng như thực hành một cách tốt nhất!

Kết luận

Qua bài viết này, bạn đã biết được các PHƯƠNG THỨC CHUỖI, và phần nào đó có các kiến thức tốt về KIỂU DỮ LIỆU CHUỖI.

Ở bài viết sau, Kteam sẽ giới thiệu với các bạn KIỂU DỮ LIỆU LIST TRONG PYTHON.

Cảm ơn bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".