Bài 16: KIỂU DỮ LIỆU SET TRONG PYTHON

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu Set trong Python

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 các bài trước, Kteam đã giới thiệu đến bạn một số container của Python.

Ở bài này Kteam sẽ giới thiệu tới bạn một container khác đó chính **KIỂU DỮ LIỆU SET** 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, KIỂU DỮ LIỆU CHUỖI trong Python
- KIỂU DỮ LIỆU LIST, KIỂU DỮ LIỆU TUPLE trong Python.

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

- Giới thiệu về Set trong Python
- Cách khởi tạo Set
- Một số toán tử với Tuple trong Python

- Indexing và cắt Set trong Python
- Các phương thức của Set
- Set không phải là một hash object

Giới thiệu về Set trong Python

Set là một container, tuy nhiên không được sử dụng nhiều bằng LIST hay TUPLE.

Một Set gồm các yếu tố sau:

- Được giới hạn bởi cặp ngoặc **{}**, tất cả những gì nằm trong đó là những phần tử của Set.
- Các phần tử của Set được phân cách nhau ra bởi dấu phẩy (**,**).
- Set không chứa nhiều hơn 1 phần tử trùng lặp

Set chỉ có thể chứa các **hashable object** nhưng chính nó **không phải** là một **hashable object**. Do đó, bạn không thể chứa một set trong một set.

Ví dụ:

```
>>> set_1 = {69, 96}
>>> set_1
{96, 69}
>>> type(set_1) # kiểu set thuộc lớp set
<class 'set'>
>>> set_2 = {'How Kteam'}
>>> set_2
{'How Kteam'}
>>> set_3 = {(69, 'Free Education'), (1, 2, 3)}
>>> set_3
{(69, 'Free Education'), (1, 2, 3)}
>>> set_4 = {[1, 2], [3, 4]}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
>>> set_5 = {(1, 2, ['How Kteam'])}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
TypeError: unhashable type: 'list'
>>> set_6 = {1, 2, {'HowKteam'}}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'set'
```

Cách khởi tạo Set

Sử dụng cặp dấu ngoặc {} và đặt giá trị bên trong

Cú pháp:

> **{<giá trị thứ nhất>, <giá trị thứ hai>, .., <giá trị thứ n – 1>, <giá trị thứ n>}**

Lưu ý: Khi khởi tạo bằng cách này, ít nhất phải có một giá trị.

Ví dụ:

```
>>> set_ = {1, 2, 3, 4}
>>> set_
{1, 2, 3, 4}
>>> set_1 = {1, 1, 1} # các giá trị trùng lặp bị loại bỏ
>>> set_1
{1}
>>> empty_set = {}  # thử khởi tạo set rỗng
>>> empty_set
{}
>>> type(empty_set) # không phải là set
<class 'dict'>
```

Sử dụng Set Comprehension

```
>>> set_1 = {value for value in range(3)}
>>> set_1
{0, 1, 2}
```

Sử dụng constructor Set

Cú pháp:

```
set(iterable)
```

Công dụng: Giống hoàn toàn với việc bạn sử dụng constructor **List**. Khác biệt duy nhất là constructor Set sẽ tạo ra một Set.

Ví dụ:

```
>>> set_1 = set((1, 2, 3))
>>> set_1
{1, 2, 3}
>>> set_2 = set('How Kteam')
>>> set_2 # set không quan tâm đến vị trí của các phần tử
{'o', ' ', 'a', 'm', 'H', 'K', 't', 'w', 'e'}
>>> set_3 = set('aaaaaaaaa')
>>> set_3
{'a'}
>>> set_4 = set([1, 6, 8, 3, 1, 1, 3, 6])
{8, 1, 3, 6}
>>> empty_set = set() # cách bạn tạo được empty set
>>> empty_set
set()
```

Một số toán tử với Set trong Python

Nhằm giúp các bạn dễ hiểu hơn về các toán tử với Set trong Python, Kteam minh họa các set dưới dạng biểu đồ Venn, với S1, S2 tương ứng các Set1, Set2 chứa các phần tử.

Toán tử in

Cú pháp:

```
value in <Set>
```

Công dụng: Kết quả trả về là True nếu value xuất hiện trong Set. Ngược lại sẽ là False

Ví dụ:

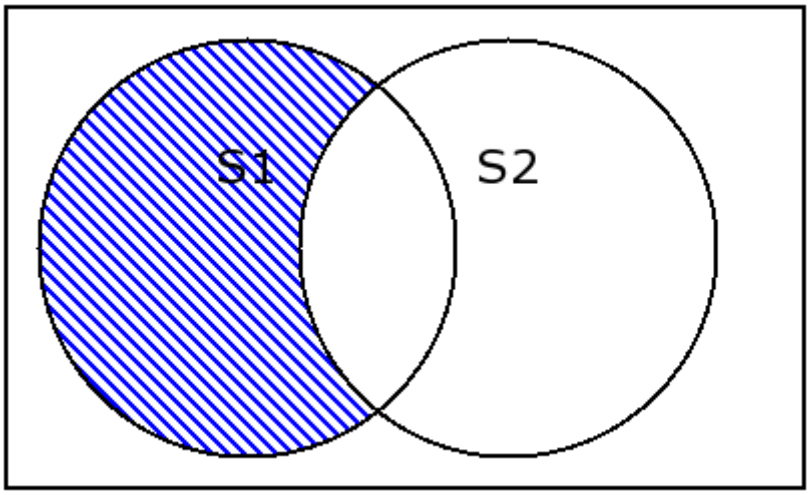
```
>>> 1 in {1, 2, 3}
True
>>> 4 in {'a', 'How Kteam', 5}
False
```

Toán tử -

Cú pháp:

```
<Set1> - <Set2>
```

Công dụng: Kết quả trả về là một Set gồm các phần tử chỉ tồn tại trong Set1 mà không tồn tại trong Set2



Ví dụ:

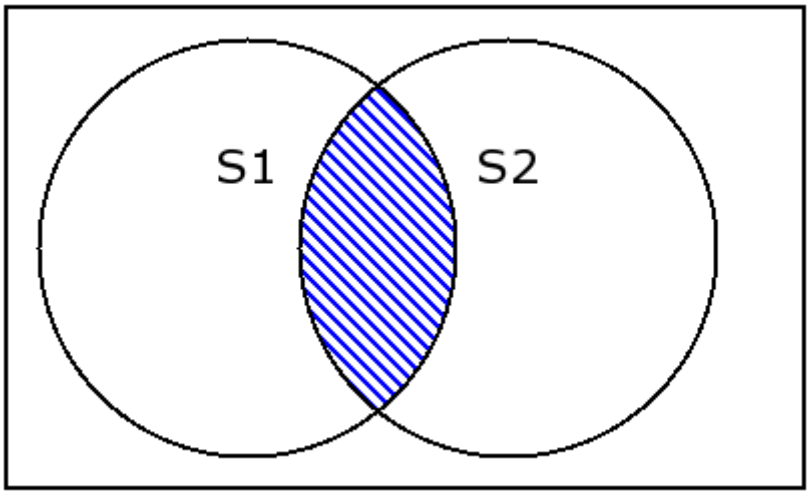
```
>>> {1, 2, 3} – {2, 3}
{1}
>>> {1, 2, 3} - {4}
{1, 2, 3}
>>> {1, 2, 3} - {1, 2, 3}
set()
>>> {1, 2, 3} - {1, 2, 3, 4}
set()
```

Toán tử &

Cú pháp:

```
<Set1> & <Set2>
```

Công dụng: Kết quả trả về là một Set chứa các phần tử vừa tồn tại trong Set1 vừa tồn tại trong Set2



Ví dụ:

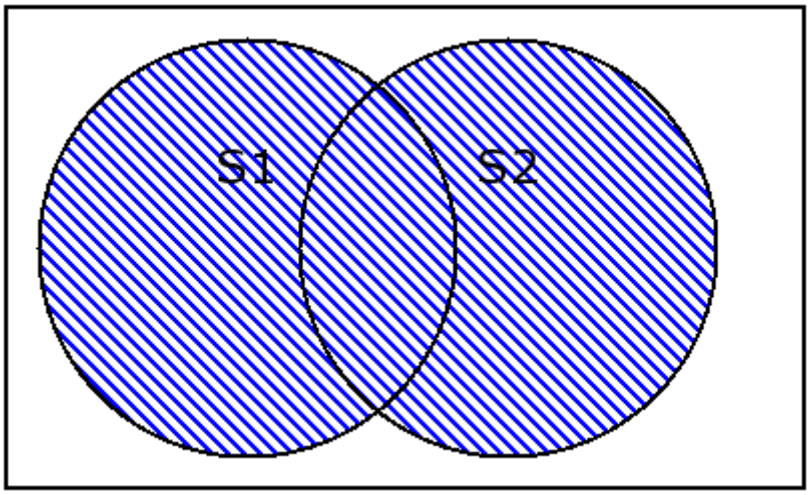
```
>>> {1, 2, 3} & {4, 5}
set()
>>> {1, 2, 3} & {1, 4, 5}
{1}
>>> {1, 2, 3} & {1, 2, 3}
{1, 2, 3}
```

Toán tử |

Cú pháp:

```
<Set1> | <Set2>
```

Công dụng: Kết quả trả về là một Set chứa tất cả các phần tử tồn tại trong hai Set



Ví dụ:

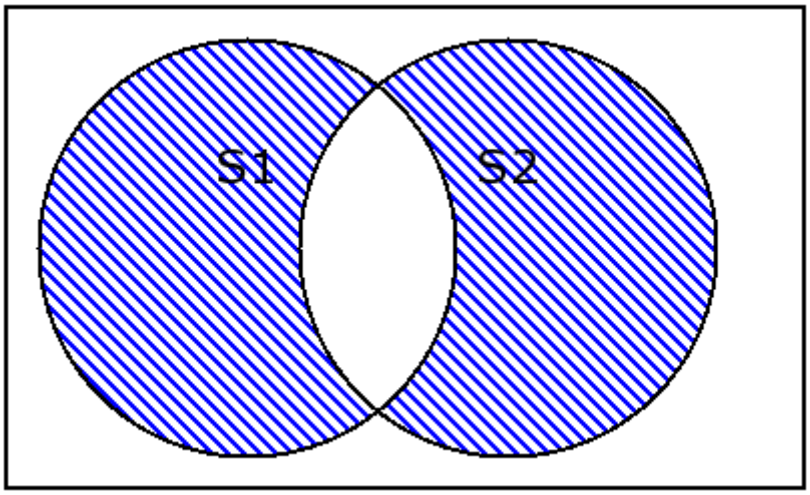
```
>>> {1, 2, 3} | {1, 2, 3}
{1, 2, 3}
>>> {1, 2, 3} | {4, 5}
{1, 2, 3, 4, 5}
```

Toán tử ^

Cú pháp:

```
<Set1> ^ <Set2>
```

Công dụng: Kết quả trả về là một Set chứa tất cả các phần tử chỉ tồn tại ở một trong hai Set



Ví dụ:

```
>>> {1, 2, 3} ^ {4, 5}
{1, 2, 3, 4, 5}
>>> {1, 2, 3} ^ {1, 2,  3}
set()
>>> {1, 2, 3} ^ {1, 4}
{2, 3, 4}
```

Indexing và cắt Set trong Python

Ở trên Kteam đã đề cập về việc set không quan tâm đến vị trí của phần tử nằm trong set. Nên, việc indexing và cắt set trong Python không được hỗ trợ.

Các phương thức của Set

Set cũng có khá nhiều phương thức. Nhưng Kteam chỉ giới thiệu một số phương thức cơ bản.

Phương thức clear

Cú pháp:

```
<Set>.clear()
```

Công dụng: Loại bỏ hết tất cả các phần tử có trong Set

Ví dụ:

```
>>> set_ 1= {1, 2}
>>> set_1.clear()
>>> set_1
set()
```

Phương thức pop

Cú pháp:

```
<Set>.pop()
```

Công dụng: Kết quả trả về một giá trị được lấy ra từ Set, đồng thời loại bỏ giá trị đã lấy ra khỏi Set ban đầu

- Nếu là set rỗng, sẽ có lỗi

Ví dụ:

```
>>> set_1 = {1, 2}
>>> set_1.pop()
1
>>> set_1
```

```
{2}
>>> set_1.pop()
2
>>> set_1
set()
>>> set_1.pop()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'pop from an empty set'
```

Lưu ý: trong một số trường hợp, bạn sẽ pop được các giá trị từ set ra từ bé đến lớn. Nhưng đó không phải bản chất của nó, việc pop này liên quan đến các giá trị của hàm hash trong của các phần từ. Đó là lí do set chỉ chứa các phần tử là các hashable object. Vì kiến thức này không quan trọng ở mức cơ bản nên Kteam xin phép được bỏ qua.

Phương thức remove

Cú pháp:

```
<Set>.remove(value)
```

Công dụng: Loại bỏ giá trị value ở trong Set. Nếu như value không ở trong Set, thông báo lên lỗi **KeyError**.

Ví dụ:

```
>>> a = {1, 2}
>>> a.remove(1)
>>> a
{2}
>>> a.remove(3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
KeyError: 3
```

Phương thức discard

Cú pháp:

```
<Set>.discard(value)
```

Công dụng: Loại bỏ giá trị value ở trong Set. Nếu như value không ở trong Set, thì sẽ bỏ qua.

Ví dụ:

```
>>> a = {1, 2}
>>> a.discard(1)
>>> a
{2}
>>> a.discard(4)
>>> a
{2}
```

Phương thức copy

Cú pháp:

```
<Set>.copy()
```

Công dụng: Trả về một bản sao của Set

Ví dụ:

```
>>> a = {1, 2}
>>> b = a.copy()
>>> b
{1, 2}
>>> a
{1, 2}
```

Phương thức add

Cú pháp:

```
<Set>.add(value)
```

Công dụng: Thêm value vào trong set. Nếu như value đã có trong Set thì bỏ qua.

Ví dụ:

```
>>> a = {1, 2}
>>> a.add(3)
>>> a
{1, 2, 3}
>>> a.add(2)
>>> a
{1, 2, 3}
```

Set không phải là một hash object

Đúng như vậy! Điều đó có thể chứng minh theo hai cách:

Ở ví dụ dưới, bạn cũng thấy, ta đã thay đổi nội dung của set nhưng id của set vẫn là id ban đầu

Ví dụ:

```
>>> a = {1, 2}
>>> id(a)
52255360
>>> a.add(3)
>>> id(a)
52255360
```

Thêm nữa, set không thể chứa một set khác

```
>>> a = {1, 2}
>>> b = {a}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'set'
```

Củng cố bài học

Câu hỏi củng cố

Giải thích lí do tại sao lại có sự thay đổi ở set a? Cho giải pháp khắc phục?

```
>>> a = {1, 2}
>>> b = a
>>> b.clear()
>>> a # tại sao lại trở thành set rỗng?
set()
```

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

Bài viết này đã giới thiệu cho các bạn KIỂU DỮ LIỆU SET TRONG PYTHON.

Ở bài sau, Kteam sẽ nói về một kiểu dữ liệu khác của Python chính là KIỂU DỮ LIỆU DICT (DICTIONARY).

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**".