9. Set và FrozenSet

Kiểu dữ liệu "set" đã là một phần của Python kể từ phiên bản 2.4. Một “set” sẽ chứa các tập các đối tượng bất biến (immutable objects), khác nhau (unique) và không được lập chỉ mục (indexing). Như vậy nó không giống “list” hay “tuple”, kiểu dữ liệu mà có thể chứa tùy tiện các đối tượng giống nhau và có thể indexing.

Set có thể được tạo từ kiểu dữ liệu tuần tự như string, list, hay tuple. Trong các đoạn mã trong bài viết này sẽ được thực hiện ngay trên CMD không qua IDE.

```
>>> s = set([1,2,3,4,1,2,3,4])

>>> print s

set([1, 2, 3, 4])
```

Dữ liệu kiểu “set” không hỗ trợ indexing

```
>>> s[1]

Traceback (most recent call last):

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

TypeError: 'set' object does not support indexing
```

“set” không chứa các đối tượng có thể thay đổi được (mutable objects), như list chẳng hạn.

```
>>> s = set([[1,2,3],[4,1,2,3,4]])

Traceback (most recent call last):

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

TypeError: unhashable type: 'list'

>>>
```

Mặc dù, set không chứa mutable object nhưng ta có thể thêm phần tử vào set. Tuy nhiên frozenset thì không thể.

```
>>> s

set([1, 2, 3, 4])

>>> s.add(10)

>>> s

set([1, 2, 3, 4, 10])

>>> s = frozenset([1,2,3,4,1,2,3,4])

>>> s

frozenset([1, 2, 3, 4])

>>> s.add(10)

Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute 'add'
>>>
```

Kể từ Python 2.6 chúng ta có thể khai báo một set mà không nhất thiết phải sử dụng hàm set. Thay vì vậy có thể dùng {} để tạo một set.

```
>>> s = {1,2,3,4,5,6,6,7}
>>> s
set([1, 2, 3, 4, 5, 6, 7])
>>>
```

Các hàm hữu ích trong set

Tên	Ý nghĩa và ví dụ
add(element)	Thêm phần tử vào tập “set”
clear()	Xóa hết phần tử trong tập “set” `>>> s` `set([1, 2, 3, 4, 5, 6, 7])` `>>> s.clear()` `>>> s` `set([])` `>>>`
copy()	Trả về một copy nông (shallow copy) của set gốc `>>> s = {1,2,3,4,5,6}` `>>> copyOfs = s.copy()` `>>> s` `set([1, 2, 3, 4, 5, 6])` `>>> s.clear()` `>>> copyOfs` `set([1, 2, 3, 4, 5, 6])`

	Chú ý: Nếu chúng ta chỉ sử dụng phép gán mà không dùng lệnh copy(), bản chất là dùng 2 cái tên khác nhau nhưng cùng trỏ đến cùng một vùng nhớ dữ liệu. >>> copyOfs = s >>> s.clear() >>> copyOfs Set([]) >>>
difference()	Trả về những phần tử khác nhau của hai tập, cụ thể như ví dụ sau: >>> a = {1,2,3,4} >>> b = {3,4,5,6} >>> a.difference(b) set([1, 2]) >>> b.difference(a) set([5, 6]) >>> Thay vì dùng “difference()” có thể dùng toán tử “-“ >>> a - b set([1, 2]) >>> b - a set([5, 6]) >>>
difference_update()	Hàm difference_update() loại bỏ tất cả các phần tử của tập khác từ tập hợp này. x.difference_update (y) là giống như "x = x - y". ví dụ >>> a set([1, 2, 3, 4]) >>> b set([3, 4, 5, 6]) >>> a.difference_update(b)

	`` >>> a set([1, 2]) >>> ```
discard(e)	Loại bỏ phần tử e nếu nó tồn tại trong tập hợp. Nếu không set sẽ không thay đổi. ``` >>> a set([1, 2]) >>> a.remove(1) >>> a set([2]) >>> a.remove(10) >>> a set([2]) ```
remove(e)	Hoạt động giống như discard(). Ngoại trừ việc nếu e không nằm trong set thì sẽ báo lỗi KeyError ``` >>> a set([1, 2]) >>> a.remove(10) Traceback (most recent call last): File "<stdin>", line 1, in <module> KeyError: 10 >>> ```
intersection(s)	Trả về các phần tử giống nhau của hai tập hợp. ví dụ ``` >>> a = {1,2,3,4,5,6} >>> b = {4,5,6,7,8,9} >>> a.intersection(b) set([4, 5, 6]) ```

	>>> #Có thể thay thế hàm này bằng toán tử "&" ...a & b set([4, 5, 6]) >>>
isdisjoint()	Trả về True nếu cả hai tập hợp không có phần tử nào giống nhau
issubset()	x.issubset(y) trả về True, nếu x là môt tập con của y. "<=" và “>=” là hai toán tử thể hiện “tập con của” và “ tập chứa của”. “<” để kiểm tra xem có phải là tập con thực hay không. Quan sát ví dụ sau: >>> x = {1,2,3,4,5} >>> y = {1,2,3} >>> x.issubset(y) False >>> y.issubset(x) True >>> x<=y False >>> y<=y True >>> y<=x True >>> y<x True >>> x<x False >>>
issuperset()	Tương tự như issubset() nhưng ngược lại vai trò của x và y.

pop()	Trả về và xóa một phần tử trong “set”, không sẽ báo lỗi nếu tập rỗng. (Hình dung tương tự như lấy phần tử từ stack hoặc queue trong C/C++) >>> s = {1,2} >>> s.pop() 1 >>> s.pop() 2 >>> s.pop() Traceback (most recent call last): File "<stdin>", line 1, in <module> KeyError: 'pop from an empty set' >>>

Kết Luận

Qua bài này chúng tôi đã giới thiệu đến các bạn thêm một kiểu dữ liệu nữa trong Python đó là set. Tại sao lại cần sinh ra set trong khi chúng ta đã có list hay tuble?. Câu trả lời nằm trong chính những thể hiện riêng biệt của set, cái mà không tồn tại trong những kiểu dữ liệu khác. Set chứa tập các đối tượng không theo một thứ tự nào cả, các đối tượng là duy nhất và không thể thay đổi trong set. Tuy nhiên set vẫn có thể thay đổi, có thể thêm hoặc xóa bớt các đối tượng. Frozenset, cùng mang các đặc điểm như set nhưng chúng ta không thể thay đổi frozenset cũng như các phần tử trong nó.

Python cũng hỗ trợ nhiều hàm thực thi trên set, các bạn cần nắm được chúng để thao tác với set được linh hoạt. Trong bài tiếp theo chúng tôi sẽ giới thiệu với các bạn một kiểu dữ liệu mới nữa, kiểu Dictionary.