Bài 18: KIỂU DỮ LIỆU DICT TRONG PYTHON - Phần 2

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu Dict trong Python - Phần 2

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 KIỂU DỮ LIỆU DICT trong Python

Ở bài này Kteam sẽ nói về **CÁC PHƯƠNG THỨC CỦA KIỂU DỮ LIỆU DICT** 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, KIỂU DỮ LIỆU SET trong Python.
- KIỂU DỮ LIỆU DICT trong Python

Trong bài này, chúng ta sẽ cùng tìm hiểu những nội dung sau đây

- Giới thiệu về phương thức của kiểu dữ liệu Dict trong Python
- Các phức thức tiện ích
- Các phương thức xử lí

Giới thiệu về phương thức của kiểu dữ liệu Dict trong Python

Kiểu dữ liệu Dict có hỗ trợ một số phương thức và đa số là xử lí các dữ liệu có trong Dict.

Mình mong các bạn sẽ hiểu rõ được các phương thức để sau này áp dụng vào giải quyết các vấn đề với việc viết ít code nhất, hạn chế lỗi nhất.

Các phương thức tiện ích

Phương thức copy

Cú pháp:

```
<Dict>.copy()
```

Công dụng: Giống với phương thức copy trong LIST. Để làm gì thì chắc các bạn cũng có thể suy nghĩ ra.

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> d_2 = d.copy()
>>> d_2
{'team': 'Kteam', (1, 2): 69}
>>> d
{'team': 'Kteam', (1, 2): 69}
```

Phương thức clear

Cú pháp:

```
<Dict>.clear()
```

Công dụng: Loại bỏ tất cả những phần tử có trong Dict

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> d.clear()
>>> d
{}
```

Các phương thức xử lí

Phương thức get

Cú pháp:

```
<Dict>.get(key [,default])
```

Công dụng: Trả về giá trị của khóa **key**. Nếu **key** không có trong Dict thì trả về giá trị default. Default có giá trị mặc định là None nếu chúng ta không truyền vào.

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> d.get('team')
'Kteam'
>>> d.get('a')
>>> d.get('a', 'haha')
'haha'
```

Phương thức items

Cú pháp:

```
<Dict>.items()
```

Công dụng: Trả về một giá trị thuộc lớp dict_items. Các giá trị của dict_items sẽ là một tuple với giá trị thứ nhất là **key**, giá trị thứ hai là **value**.

- Dict_items là một **iterable**.

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> items = d.items()
>>> items
dict_items([('team', 'Kteam'), ((1, 2), 69)])
>>> type(items)
<class 'dict_items'>
>>> list_items = list(items)
>>> list_items
[('team', 'Kteam'), ((1, 2), 69)]
>>> list_items[0]
('team', 'Kteam')
>>> list_items[0][1]
'Kteam'
```

Phương thức keys

Cú pháp:

```
<Dict>.keys()
```

Công dụng: Trả về một giá trị thuộc lớp dict_keys. Các giá trị của dict_keys sẽ là các **key** trong Dict.

- Dict_keys là một **iterable**.

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> keys = d.keys()
>>> keys
dict_keys(['team', (1, 2)])
>>> type(keys)
<class 'dict_keys'>
>>> list_keys = list(keys)
```

```
>>> list_keys
['team', (1, 2)]
>>> list_keys[-2]
'team'
```

Phương thức values

Cú pháp:

```
<Dict>.values()
```

Công dụng: Trả về một giá trị thuộc lớp dict_values. Các giá trị của dict_values sẽ là các **value** trong Dict.

- Dict_values là một iterable.

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> values = d.values()
>>> values
dict_values(['Kteam', 69])
>>> type(values)
<class 'dict_values'>
>>> list_values = list(values)
>>> list_values
['Kteam', 69]
```

Phương thức pop

Cú pháp:

```
<Dict>.pop(key [,default])
```

Công dụng: Bỏ đi phần tử có **key** và trả về **value** của **key** đó. Trường hợp **key** không có trong dict.

- Báo lỗi **KeyError** nếu default là None (ta không thêm vào).
- Trả về default nếu ta thêm default vào.

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> d.pop('team')
'Kteam'
>>> d
{(1, 2): 69}
>>> d.pop('non-exist')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'non-exist'
>>> d.pop('non-exist', 'defaul_value')
'defaul_value'
```

Phương thức popitem

Cú pháp:

```
<Dict>.popitem()
```

Công dụng: Trả về một 2-tuple với **key** và **value** tương ứng bất kì (vấn đề này liên quan đến giá trị của hash của key. Do đó bạn cũng hiểu vì sao key buộc phải là một **hash object**) trong Dict. Và cặp **key-value** sẽ bị loại bỏ ra khỏi Dict.

- Nếu Dict là một empty Dict. Sẽ có lỗi **KeyError**

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> d.popitem()
((1, 2), 69)
>>> d
{'team': 'Kteam'}
>>> d.popitem()
('team', 'Kteam')
>>> d.popitem()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'popitem(): dictionary is empty'
```

Phương thức setdefault

Cú pháp:

```
<Dict>.setdefault(key [,default])
```

Công dụng: Trả về giá trị của **key** trong Dict. Trường hợp **key** không có trong Dict thì sẽ trả về giá trị default. Thêm nữa, một cặp **key-value** mới sẽ được thêm vào Dict với **key** bằng **key** và **value** bằng default.

- Default mặc định là None

Ví dụ:

```
>>> d = {'team': 'Kteam', (1, 2): 69}
>>> d.setdefault('team')
'Kteam'
>>> d.setdefault('non-exist_1')
>>> d
{'team': 'Kteam', (1, 2): 69, 'non-exist_1': None}
>>> d.setdefault('non-exists_2', 'default_value')
'default_value'
>>> d
{'team': 'Kteam', (1, 2): 69, 'non-exist_1': None, 'non-exists_2': 'default_value'}
```

Phương thức update

Cú pháp:

```
<D>.update([E, ]**F)
```

Công dụng: Phương thức giúp bạn cập nhật nội dung cho Dict.

- **F** là một Dict được tạo thành bởi **packing arguments** (khái niệm sẽ được Kteam giải thích ở một bài trong tương lai). Và sẽ thêm vào Dict bằng cách:

```
for k in F: D[k] = F[k]
```

- Nếu **E** được truyền vào và đối tượng **E** có phương thức **keys()**, thì sẽ cập nhật Dict bằng cách:

```
for k in E: D[k] = E[k]
```

- Nếu **E** được truyền vào và đối tượng **E**, đối tượng này có các giá trị là một container chứa hai giá trị thì sẽ cập nhật Dict bằng cách.

```
for k, v in E: D[k] = v
```

Nếu bạn đọc xong và không hiểu gì, thì cũng đừng thất vọng. Kteam sẽ cho bạn vài ví dụ minh họa. Nó rất đơn giản.

Đây là update theo kiểu sử dụng **packing arguments.**

```
>>> d = {'a': 1}
>>> d
{'a': 1}
>>> d.update(b=2,c=3)
>>> d
{'a': 1, 'b': 2, 'c': 3}
```

Đây là cách bạn truyền **E** với **E** là một đối tượng có phương thức **keys**

```
>>> d = {'a': 1}
>>> E = {'b': 2, 'c': 3}
>>> d.update(E)
>>> d
{'a': 1, 'b': 2, 'c': 3}
```

Đây là truyền vào một E với E có các giá chứa hai giá trị

```
>>> d = {'a': 1}
>>> E = [('b', 2), ('c', 3)]
>>> d.update(E)
>>> d
{'a': 1, 'b': 2, 'c': 3}
>>> E_f = (['d', 69], ['e', 96])
>>> d.update(E_f)
>>> d
{'a': 1, 'b': 2, 'c': 3, 'd': 69, 'e': 96}
```

Củng cố bài học

Câu hỏi củng cố

1. Tại sao thay đổi dict2 mà dict1 lại cũng bị thay đổi theo? Hãy cho giải pháp khắc phục

```
>>> dict1 = {'key': 6969}
>>> dict1
{'key': 6969}
>>> dict2 = dict1
>>> dict2
{'key': 6969}
>>> dict2['key'] = 'changed'
>>> dict2
{'key': 'changed'}
>>> dict1
{'key': 'changed'}
```

2. Nêu sự khác nhau giữa

```
>>> d = {}
>>> d.update({'a': 3})



và

>>> d = {}

>>> d.update(3)
```

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 hiểu thêm về DICT qua các phương thức của nó có.

Ở bài viết sau. Kteam hướng dẫn các bạn XỬ LÝ FILE 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**".