**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](http://www.howkteam.vn/course/kieu-du-lieu-list-trong-python--phan-1/kieu-du-lieu-list-trong-python--phan-2-1549). 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}

**Toán tử “|” với 2 dict**

Cú pháp

<**Dict\_A**> **|**<**Dict\_B**>

**Công dụng:** Trả về một dict mới với các cặp key – value có mặt ở một trong 2 dict. Nếu một key bất kì có trong cả 2 dict, thì giá trị được lấy sẽ là cặp key – value ở Dict\_B

**Ví dụ:**

>>> {'1': 1, '2': 2} | {3: '3', 4: '4'}

{'1': 1, '2': 2, 3: '3', 4: '4'}

>>> {1: '1', 2: '2'} | {2: 'Two'}

{1: '1', 2: 'Two'}

**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'}

1. Nêu sự khác nhau giữa

>>> d = {}

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

và

>>> d = {}

>>> d.update(3)