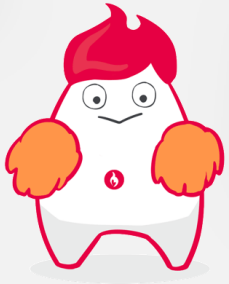
字典添加修改数据的方法



本节课内容

- [] 处理法
- 字典的内置函数 update
- 字典的内置函数setdefault
- 注意事项再强调

[]处理法

字符串，列表，list[0] = 10

- 字典没有索引
- dict[‘name’] = ‘dewei’
- 添加或修改，根据key是否存在所决定

```
In [1]: d = {'name': 'dewei'}

In [2]: d['name'] = 'xiaomu'

In [3]: print(d)
{'name': 'xiaomu'}
```

update的功能与用法

- 添加新的字典，如新字典中有和原字典相同的key，则该key的value会被新字典的value覆盖

update的功能与用法

用法：

dict.update(new_dict) --该函数无返回值

参数：

new_dict：新的字典

```
In [4]: default_dict = {}

In [5]: new_dict = {'name': 'dewei'}

In [6]: default_dict.update(new_dict)

In [7]: default_dict
Out[7]: {'name': 'dewei'}
```

setdefault的功能

- 获取某个key的value，如key不存在于字典中，将会添加key并将value设为默认值,

setdefault的用法

用法：

dict.setdefault(key, value)

参数：

key 需要获取的key

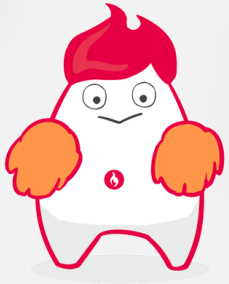
value 如果key不存在，对应这个key存入字典的默认值

```
In [8]: default_dict = {}

In [9]: value = default_dict.setdefault('name', '小慕')

In [10]: print('dict:', default_dict, 'value:', value)
dict: {'name': '小慕'} value: 小慕
```

注意事项再强调

- 字典中每一个key一定是唯一的
- 字典中的数据量没有限制
- 字典中的value可以是任何python的那内置据类型的对象 和自定义的对象

字典的keys函数



本节课内容

- keys 的功能
- keys 的用法

keys功能

- 获取当前字典中所有的键（key）

keys的用法

用法：

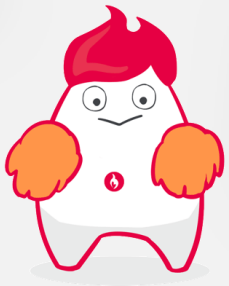
dict.keys() -> 无需传参，返回一个key集合的伪列表

```
In [11]: my_dict = {'name': 'dewei', 'age': 33}

In [12]: my_dict.keys()
Out[12]: dict_keys(['name', 'age'])
```

不具备列表的所有功能

```
In [13]: key_list = list(my_dict.keys())

In [14]: key_list
Out[14]: ['name', 'age']
```

字典的values函数



本节课内容

- values 的功能
- values 的用法

values功能

- 获取当前字典中所有键值对中的值（value）

values的用法

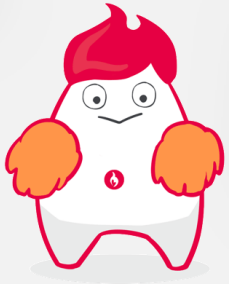
用法：

dict.values() -> 无需传参，返回一个value集合的伪列表

```
In [15]: my_dict = {'name': 'dewei', 'age': 33}

In [16]: my_dict.values()
Out[16]: dict_values(['dewei', 33])
```

不具备列表的所有功能

字典key的获取



本节课内容

- [] 的获取方法
- 字典内置函数 get 获取方法
- [] 与 get的区别

[]的获取方法

```
In [17]: my_dict = {'name': 'dewei', 'age': 33}

In [18]: name = my_dict['name']

In [19]: print(name)
dewei
```

- 字典+中括号内传key，不进行赋值操作 即为获取
- 返回key对应的value值

get功能

- 获取当前字典中指定key的value

get用法

用法：

dict.get(key, default=None)

参数：

key： 需要获取value的key

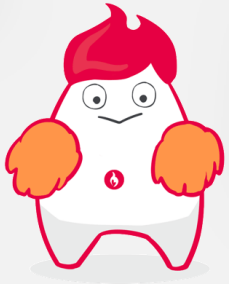
default： key不存在则返回此默认值，默认是None，我们也可以自定义

```
In [20]: my_dict = {'name': 'dewei', 'age': 33}

In [21]: name = my_dict.get('name')

In [22]: print(name)
dewei
```

[]与get的区别

- [] 如果获取的key不存在，则直接报错
- get如果获取的key不存在，则返回默认值
- 所以开发中，优先使用get函数

字典的删除



本节课内容

- clear函数的功能与用法
- pop函数的功能与用法
- del 在 字典中的用法

clear功能

- 清空当前的字典中所有数据

clear用法

用法：

dict.clear() -> 无参数，无返回值

```
In [23]: my_dict = {'name': 'dewei', 'age': 33}

In [24]: my_dict.clear()

In [25]: print(my_dict)
{}
```

pop功能

- 删除字典中指定的key，并将其结果返回，如果key不存在则报错

pop用法

用法：

dict.pop(key)

-- key 希望被删掉的键

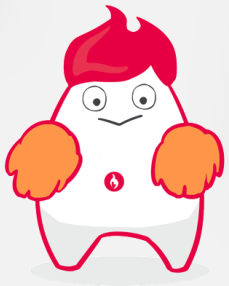
>> 返回这个key对应的值（value）

```
In [26]: my_dict = {'name': 'dewei', 'age': 33}

In [27]: pop_value = my_dict.pop('age')

In [28]: print('pop value:', pop_value, 'my_dict:', my_dict)
pop value: 33 my_dict: {'name': 'dewei'}
```

del 在字典中的用法

```
my_dict = {'name': 'dewei', 'age': 33}

del my_dict['name']
print(my_dict)
>> {'age': 33}

del my_dict
print(my_dict) -> 报错，整个字典对象已被删除
```

字典的复制—copy函数



本节课内容

- copy函数的功能
- copy函数的用法

copy的功能

- 将当前字典复制一个新的字典

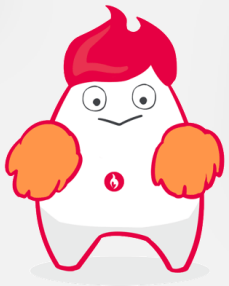
copy的用法

用法：

dict.copy() -> 该函数无参数，返回一个一模一样的内存地址不同的字典

```
In [33]: old_dict = {'name': 'dewei', 'age': 33}

In [34]: new_dict = old_dict.copy()

In [35]: id(new_dict) != id(old_dict)
Out[35]: True
```

字典中的成员判断



本节课内容

- in 与 not in在字典中的用法
- 字典内置的get也可以参与一下

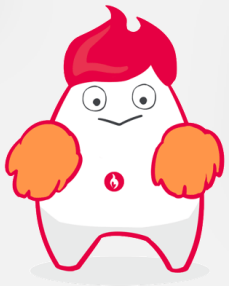
in 与 not in

```
In [39]: test_dict = {'name': 'xiaomu'}

In [40]: 'name' in test_dict
Out[40]: True

In [41]: 'name' not in test_dict
Out[41]: False
```

get用于判断成员存在

```
In [42]: test_dict = {'name': 'xiaomu'}

In [43]: bool(test_dict.get('name'))
Out[43]: True
```

字典中的末尾删除函数--popitem



本节课内容

- popitem的功能
- popitem的用法
- 该函数的注意事项

popitem功能

删除当前字典里末尾一组键值对并将其返回

popitem用法

用法：

dict.popitem() -- 无需传参

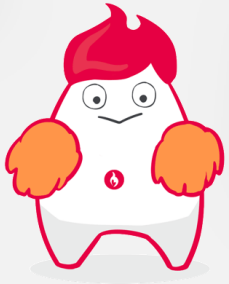
>>返回被删除的键值对，用元组包裹0索引是key，1索引是value

```
In [44]: my_dict = {'name': 'dewei', 'age': 33}

In [45]: my_dict.popitem()
Out[45]: ('age', 33)
```

主意事项

如字典为空，则直接报错

所有数据类型与布尔值的关系



本节课内容

- 字符串，数字，列表，元组，字典，空类型 与 布尔值的关系总结

数据类型与布尔值的关系

- 每一种数据类型，自身的值都有表示True与False
- not 对于一切结果取反

数据类型与布尔值的关系

数据类型	为True	为False
int	非0	0
float	非0.0	0.0
str	len(str) != 0	len(str) == 0 即 ''
list	len(list) != 0	len(list) == 0 即 []
tuple	len(tuple) != 0	len(tuple) == 0 即 ()
dict	len(dict) != 0	len(dict) == 0 即 {}
None	not None	None