|  |  |
| --- | --- |
| 字典创建申明 | Example\_Dict = {key1:value1,key2:value2} |
| 字典值获取 | 1 example\_dict[key] 可以获取key所对应的值，如果key不存在则报错  2 example\_dict.get(key) 可以获取key所对应的值，如果key不存在则不报错返回None |
| 字典更新 | Dict1.update(Dict2) 使用dict2去更新dict1,如果dict2独有的key则加入dict1,如果有dict1与dict2共有，则dict1的key被覆盖 |
| 字典是否存在key | key in dict1 |
| 字典返回数据 | Dict.keys()返回字典所有key  Dict.values()返回字典所有value  Dict.items()返回字典所有键值对入元组中 |
| 字典删除键值对 | Dict.pop(key) 返回删除key对应的value  Del dict[key] 删除，但不返回对应value |

1. 对list of dict进行排序

例如：

dict1 = {"weight”: 1, "value":3}

dict2 = {"weight”: 2, "value":3}

dict3 = {"weight”: 3, "value":3}

list1 = [dict3, dict2, dict1]

此时如果直接进行sorted(list1)将报错， 因为Python使用list默认的行为进行比较。但是所含元素为字典，无法直接进行比较，所以我们要指定sorted的时候使用具体的什么域作为比较的属性。

sorted (list1, key= lambda k: k[‘weight’]) # 指定采用元素k字典中的weight作为排序比较的对象

2.

使用list of tuple自动构建字典

例如：test = [(“A”, “B”), (“C”,” D”)], 则dict(test)将自动构建键为A, B键值为C, D的字典。

3.

Collections.defaultdict() 的使用

Python中通过Key访问字典，当Key不存在时，会引发 ‘KeyError’异常。为了避免这种情况的发生，可以使用collections类中的defaultdict()方法来为字典提供默认值。

该函数返回一个类似字典的对象。defaultdict是Python内建字典类（dict）的一个子类，它重写了方法\_missing\_(key)，增加了一个可写的实例变量default\_factory, 实例变量default\_factory被missing () 方法使用，如果该变量存在，则用以初始化构造器，如果没有，则为None。其它的功能和dict一样。

例如：

from collections import defaultdict

test1 = defaultdict(list)

test2 = {}

test1[“A”] # 结果使一个空列表，因为即使键” A” 不存在，defaultdict也会返回一个空列表，空列表是在构建defaultdict第一个参数指定的. 在接下来我们也可以为test1[“A“]赋值为其他值。

test2[“A”] # 报错，键不存在。

4.