|  |  |
| --- | --- |
| 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。或者指定默认值的情况下，返回默认值。例如example\_dict.get(key, “defaultvalue”) |
| 字典添加键值对 | 1. Setdefault方法接受两个参数(key, value)  如果key不存在则设置key对应值为value. 如果key存在则不覆盖设置。  2. dict[key] = value 此方法将字典中的key设置对应值为value。 如果key不存在则设置，如果key存在则覆盖。 |
| 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 |
| 字典的浅拷贝 | Dict.copy()  字典自带方法，需要注意的是浅拷贝。即对于字典中的引用类型不会进行再一次赋值，修改一个副本中引用类型的数据会影响原数据。 |
| 字典的深拷贝 | import copy  copy.deepcopy（Dict）  字典的深拷贝需要借用copy模块然后调用deepcopy方法。由于是深拷贝，所以对于副本中的引用类型也会进行复制，得到另外一个引用，彼此不相互影响。 |

🡪

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

🡪

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。

🡪

Collections.defaultdict(method\_factory) 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. 在实例化时，如果default\_factory存在，则用以初始化构造器，并且当读取的键值不存在时返回一个空的default\_factory所代表数据类型。如果初始化时没有指定default\_factory，其它的功能和dict一样， 当获取的键值不存在时报错。

例如：

from collections import defaultdict

test1 = defaultdict(list) #以list作为default\_factory

test2 = {}

test1[“A”] # 结果时一个空列表，因为即使键” A” 不存在，defaultdict也会根据default\_factory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。

🡪

在遍历字典的时候删除字典键

字典在遍历时进行修改，可能会抛出RuntimeError: dictionary changed size during iteration

例如以下代码：

test = {“A”:” B”, “C”:””}

def refine\_dict(task\_dict):  
 for key in task\_dict.keys():  
 if not task\_dict[key]:  
 del task\_dict[key]

refine\_dict(test) # 在Python3中将抛出异常RuntimeError，Python2可行

如果要达到目的修改为：

test = {“A”:” B”, “C”:””}

def refine\_dict(task\_dict):  
 for key in list(task\_dict.keys()):  
 if not task\_dict[key]:  
 del task\_dict[key]

refine\_dict(test) # 在Python3，Python2可行

🡪

从Python3.6开始，默认的字典类型dict在读取的时候便会保持当初key插入的顺序。当更新一个已有key的对应值时候，字典key顺序不发生改变。当删除key再重新传入一样的key时候，该key会放在该字典的key顺序最后。

Python另外一个字典类型能保持当初key插入的顺序为collections.OrderedDict. collections.OrderedDict.从Python 3.1开始加入collections模块。