1. 字典

字典是python中唯一的映射类型，采用键值对（key-value）的形式存储数据。python对key进行哈希函数运算，根据计算的结果决定value的存储地址，所以字典是无序存储的，且key必须是可哈希的。可哈希表示key必须是不可变类型，如：数字、字符串、只含不可变类型元素的元组（1,2,3,’abc’）、实现\_\_hash\_\_()方法的自定义对象（因为\_\_hash\_\_()须返回一个整数，否则会出现异常：TypeError: an integer is required）。可以用hash(obj)检测对象是否是可哈希的。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | >>> class HashEnable(object):  ...    def  \_\_hash\_\_(self):  ...         return 1  >>> he = HashEnable()  >>> hash(he)  1  >>> d = {he:1}  >>> d = {['1',2]:2}  Traceback (most recent call last):    File "<stdin>", line 1, in <module>  TypeError: unhashable type: 'list' |

1.1 字典常用操作

（1）创建字典

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | >>> d1 = {}  >>> d2 = {'player':'QVOD','game':'kw'}  >>> d1,d2  ({}, {'player': 'QVOD', 'game': 'kw'})    >>> d3 = dict((['name','alex'],['sex','man']))  >>> d3  {'name': 'alex', 'sex': 'man'}  >>> d33 = d3.copy()  >>> d33  {'name': 'alex', 'sex': 'man'}    >>> d4 = {}.fromkeys(('alex','zhou'),1)  >>> d4  {'alex': 1, 'zhou': 1}  >>> d5 = {}.fromkeys(('alex','zhou'))  >>> d5  {'alex': None, 'zhou': None} |

（2）遍历字典  
ps：访问一个不存在的key时，会发生KeyError异常，访问前可使用in或not in判断一下。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | >>> d = {'name':'alexzhou','sex':'man'}    >>> for key in d:  ...     print '%s,%s' %(key,d[key])  ...  name,alexzhou  sex,man    >>> d['name']  'alexzhou'  >>> d2 = {'name':'alexzhou','age':100}  >>> print 'name: %s,age: %d' %(d2['name'],d2['age'])  name: alexzhou,age: 100    >>> d2['sex']  Traceback (most recent call last):    File "<stdin>", line 1, in <module>  KeyError: 'sex'    >>> 'sex' in d2  False  >>> 'name' in d2  True |

（3）更新字典

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | >>> d = {'name':'alexzhou','age':100}  >>> d['age'] = 88  >>> d  {'age': 88, 'name': 'alexzhou'}  >>> d.pop('age')  88  >>> d  {'name': 'alexzhou'}  >>> d.clear()  >>> d  {} |

1.2 常用内建函数

（1）cmp()  
字典的比较：首先是字典的大小，然后是键，最后是值

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | >>> d1 = {'abc':1,'efg':2}  >>> d2 = {'abc':1,'efg':2,'h':3}  >>> cmp(d1,d2)  -1  >>> d3 = {'ab':1,'efg':2}  >>> cmp(d1,d3)  1  >>> d4 = {'abc':1,'efg':3}  >>> cmp(d1,d4)  -1  >>> d5 = {'abc':1,'efg':2}  >>> cmp(d1,d5)  0 |

（2）len()  
返回键值对的数目

|  |  |
| --- | --- |
| 1  2  3 | >>> d = {'abc':1,'efg':2}  >>> len(d)  2 |

（3）keys()、values() 、items()  
keys()返回一个包含字典所有键的列表  
values()返回一个包含字典所有值的列表  
items()返回一个包含键值元组的列表

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | >>> d = {'name':'alex','sex':'man'}  >>> d.keys()  ['name', 'sex']  >>> d.values()  ['alex', 'man']  >>> d.items()  [('name', 'alex'), ('sex', 'man')] |

（4）dict.get(key,default=None)  
返回字典中key对应的value，若key不存在则返回default

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> d = {'name':'alex','sex':'man'}  >>> d.get('name','not exists')  'alex'  >>> d.get('alex','not exists')  'not exists' |

（5）dict.setdefault(key,default=None)  
若key存在，则覆盖之前的值，若key不存在，则给字典添加key-value对

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | >>> d.setdefault('name','zhou')  'alex'  >>> d  {'name': 'alex', 'sex': 'man'}  >>> d.setdefault('haha','xixi')  'xixi'  >>> d  {'haha': 'xixi', 'name': 'alex', 'sex': 'man'} |

（6）dict.update(dict2)  
将字典dict2的键值对添加到dict

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> d = {'name':'alex','sex':'man'}  >>> d1 = {'age':100,'address':'shenzhen'}  >>> d.update(d1)  >>> d  {'age': 100, 'address': 'shenzhen', 'name': 'alex', 'sex': 'man' |

（7）sorted(dict)  
返回一个有序的包含字典所有key的列表

|  |  |
| --- | --- |
| 1  2 | >>> sorted(d)  ['address', 'age', 'name', 'sex'] |

2. 集合set

python中集合对象（set）是一组无序排列的可哈希的值，包含两种类型：可变集合（set）和不可变集合（frozenset），所以set不是可哈希的，frozenset是可哈希的，能当作字典的键。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | >>> s = set('a')  >>> hash(s)  Traceback (most recent call last):    File "<stdin>", line 1, in <module>  TypeError: unhashable type: 'set'    >>> fs = frozenset('a')  >>> hash(fs)  -1305064881317614714 |

2.1 集合常用操作

（1）创建集合

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | >>> s = set('alexzhou')  >>> s  set(['a', 'e', 'h', 'l', 'o', 'u', 'x', 'z'])  >>> fs = frozenset('alexzhou')  >>> fs  frozenset(['a', 'e', 'h', 'l', 'o', 'u', 'x', 'z']) |

（2）遍历集合

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | >>> for e in s:  ...     print e  ...  a  e  h  l  o  u  x  z |

（3）更新集合(add/update/remove/discard/pop/clear(-=))  
s.add(obj):添加对象obj  
s.update(s1): 用s1中的成员修改s，s现在包含s1的成员  
s.remove(obj):从集合s中删除obj，若obj不存在，则引发KeyError错误  
s.discard(obj): 如果obj是s的成员，则删除obj  
s.pop(): 删除集合s中任意一个对象，并返回  
s.clear(): 删除集合s中所有元素

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | >>> s = set('alexzhou')  >>> s.update('hai')  >>> s  set(['a', 'e', 'i', 'h', 'l', 'o', 'u', 'x', 'z'])  >>> s.add('hai')  >>> s  set(['a', 'hai', 'e', 'i', 'h', 'l', 'o', 'u', 'x', 'z'])  >>> s.remove('hai')  >>> s  set(['a', 'e', 'i', 'h', 'l', 'o', 'u', 'x', 'z'])  >>> s -= set('alex')  >>> s  set(['i', 'h', 'o', 'u', 'z'])  >>> s.pop()  'i'  >>> s  set(['h', 'z', 'u', 'o'])  >>> s.discard('h')  >>> s  set(['z', 'u', 'o'])  >>> s.clear()  >>> s  set([])  >>> fs = frozenset('alexzhou')  >>> fs.add('z')  Traceback (most recent call last):    File "<stdin>", line 1, in <module>  AttributeError: 'frozenset' object has no attribute 'add' |

(4) 集合比较  
s1.issubset(s2)：检测s1是否是s2的子集，是则返回True，否则返回False  
s1.issuperset(s2)：检测s1是否是s2的超集，是则返回True，否则返回False

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | >>> s = set('alexzhou')  >>> fs = frozenset('alexzhou')  >>> s == fs  True  >>> s2 = set('alexzhou')  >>> s == s2  True  >>> s3 = set('alexzhouj')  >>> s > s3  False  >>> s < s3  True  >>> s |

（5）联合union操作(s1|s2,s1.union(s2))  
产生的集合的每个元素至少是其中一个集合的成员。如果左右两边的集合类型相同，则产生的结果是相同的，若不同，则产生的结果跟左操作数相同。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | >>> s1 = set('abc')  >>> fs = frozenset('de')    >>> s1 | fs  set(['a', 'c', 'b', 'e', 'd'])    >>> type(s1 | fs)  <type 'set'>  >>> type(fs | s1)  <type 'frozenset'>    >>> s2 = set('fg')  >>> type(s1 | s2)  <type 'set'>  >>> s1.union(fs)  set(['a', 'c', 'b', 'e', 'd'])  >>> type(s1.union(fs))  <type 'set'="">  >>> type(fs.union(s1))  <type 'frozenset'="">    </type></type> |

（6）交集s1&s2，补集s1-s2，异或s1^s2  
交集：新集合中的元素同时是s1和s2的元素 –> s1.intersection(s2)  
补集：新集合中的元素只属于s1,不属于 –> s1.difference(s2)  
异或：新集合中的元素不能同时属于s1和s2 –> s1.symmetric\_difference(s2)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | >>> fs = frozenset('de')  >>> s = set('def')  >>> s & fs  set(['e', 'd'])  >>> s - fs  set(['f'])  >>> fs - s  frozenset([])  >>> s ^ fs  set(['f'])  >>> s.intersection(fs)  set(['e', 'd'])  >>> s.difference(fs)  set(['f'])  >>> s.symmetric\_difference(fs)  set(['f']) |

原文地址：

<http://codingnow.cn/language/353.html>