知识点

1. 创建字典

2. 查找字典

3. 遍历字典

4. 添加、修改和删除字典

5. 字典检索

3.字典

字典是一种集合，它不是序列。字典可以看成元素对构成的列表，其中一个元素是键，另一个元素是值。在搜索字典时，首先查找键，当查找到键后就可以直接获取该键对应的值，效率很高，是一种高效的查找方法。

3.字典

3.1创建字典

与列表、元组不同的是，字典是以“{”和“}”定义的，而且字典中每个元素包含两个部分，即键和值。下面给出了一些实例，展示了各种语法，这些语法产生的是相同的字典。

3.字典

3.1创建字典

```
>>> d1=dict({"id":19,"name":"Marry","city":"chongqing"})
>>> d2=dict(id=19,name="Marry",city="chongqing")
>>> d3=dict([("id",19),("nmae","Marry"),("city","chongqing")])
>>> d4=dict(zip(("id","name","city"),(19,"Marry","chongqing")))
>>> d5={"id":19,"name":"Marry","city":"chongqing"}
>>> d1
{'name': 'Marry', 'id': 19, 'city': 'chongqing'}
>>> d2
{'name': 'Marry', 'id': 19, 'city': 'chongqing'}
>>> d3
{'name': 'Marry', 'id': 19, 'city': 'chongqing'}
>>> d4
{'name': 'Marry', 'id': 19, 'city': 'chongqing'}
>>> d5
{'name': 'Marry', 'id': 19, 'city': 'chongqing'}
```

3.字典

3.2查找与反向查找

字典定义好后，可以通过键来查找值，这个操作称为“查找”。

```
>>> d1['id']
19
>>> d1["name"]
'Marry'
>>> d1["chongqing"]
Traceback (most recent call last):
 File "<pyshell#3>", line 1, in <module>
    d1["chongqing"]
KeyError: 'chongqing'
```

3.字典

3.2查找与反向查找

对于字典的操作通常是通过键来查找值，而能不能通过一个给定的值来确定其键呢？

由于字典是一对多的关系，即一个键可能对应多个值，若想根据一个值来确定其键时，只能通过暴力搜索的方法。下面给出一个简单的暴力搜索实例，该段代码接收一个值，并返回映射到该值的键：

```
>>> def reverse_lookup(d,v):
        for k in d:
                if d[k]==v:
                        return k
        raise LookupError()
```

3.字典

3.3遍历字典

用循环语句来遍历字典中的每个元素的键和值，如下所示：

```
>>> for key in d1.keys():
                    print(key,d1[key])

name Marry
id 19
city Chongqing
```

3.字典

3.4添加和修改字典

字典的大小和列表都是动态的，即不需要事先指定其容量大小，可以随时向字典中添加新的键—值对，或者修改现有键所关联的值。添加和修改的方法相同，都是使用“字典变量名[键名]=键值”的形式，主要区分在于字典中是否已存在该键—值对，若存在则为修改，否则为添加。例如：

```
>>> d1["name"]="jason"
>>> d1
{'name': 'jason', 'id': 19, 'city': 'chongqing'}
>>> d1["sex"]="female"
>>> d1
{'sex': 'female', 'name': 'jason', 'id': 19, 'city': 'chongqing'}
```

3.字典

3.4添加和修改字典

例如：

```
>>> d1["name"]="jason"
>>> d1
{'name': 'jason', 'id': 19, 'city': 'chongqing'}
>>> d1["sex"]="female"
>>> d1
{'sex': 'female', 'name': 'jason', 'id': 19, 'city': 'chongqing'}
```

- d1=dict({"id":19,"name":"Marry","city":"chongqing"})，d1中已经存在name键—值对，所以第一个操作是“修改”。d1中原本不存在sex键—值对，所以第二个操作是“添加”。
- 因为字典是无序的，类似于append在尾部添加键—值对的方法是没有任何意义的。

3.字典

3.5字典长度

与列表、元组相同，可以用len()函数返回字典中键的数量，如下所示。

```
>>> len(d1)
4
```

3.字典

3.6字典检索

可以使用in运行符来测试某个特定的键是否在字典中。表达式k in d（d为字典）查找的是键，而不是值。

```
>>> "id" in d1
True
>>> "name" in d1
True
>>> "NO" in d1
False
```

3.字典

3.6字典检索

查看一个值是不是出现在字典中，可以使用方法values，它返回该字典的所有值的一个集合，然后检索当前值是否在集合中即可，例如：

```
>>> d1={'sex': 'female', 'name': 'jason', 'id': 19, 'city':
'chongqing'}
>>> vals=d1.values()
>>> "jason" in vals
True
```

3.字典

3.7删除元素和字典

- 可以使用del语句删除指定键的元素或整个字典；
- 使用clear()方法来删除字典中所有元素；
- 使用pop ()方法删除并返回指定键的元素；
- popitem()弹出随机的项。

3.字典

3.8字典的常用函数

- copy()：返回一个具有相同键—值对的新字典，该新字典是原来字典的一个副本（这个方法实现的是浅拷贝）。
- fromkeys()：使用给定的键建立新的字典，每个默认对应的值为None。
- get()：get方法是一个更宽松的访问字典项的方法。一般来说，如果试图访问字典中不存在的项时会出错。
- items()：items方法将所有的字典项以列表的方式返回，这些列表项中的每一项都来自于(键,值)。

3.字典

3.8字典的常用函数

- keys()：将字典中的键以列表形式返回。
- setdefault()：能够获得与给定键相关联的值，除此之外，setdefault还能在字典中不含有给定键的情况下设定相应的键值。
- update()：利用一个字典项更新另一个字典，若有相同的键存在，则会进行覆盖。

代码练习题

请为如下信息创建字典：
name: 小明
age: 18
gender: True
height: 1.75
weight: 75.5

答案：

```
xiaoming = {"name": "小明",
            "age": 18,
            "gender": True,
            "height": 1.75,
            "weight": 75.5}
```

代码练习题

请为字典xiaoming_dict = {“name”: “小明”}增加键值对age,18。

答案：

```
xiaoming_dict["age"] = 18
```