“小帅，我今天参观了‘中国航天展’，感觉探月工程太伟大了！我们可以通过编程，将探月工程的一系列科研成果记录下来，方便记忆和查询吗？”

“小萌，我们可以为嫦娥工程做一本‘字典’，在这本字典中，用‘工程名：嫦娥 5 号’‘发射时间：2020 年 11 月 24 日’‘目标：月球采样返回’……这样的条目来记录。这样查询探月工程就像翻字典一样简单啦！”

数据与数据之间往往存在着有趣的关联，如图 3-1 所示的连线题，是记忆数据之间关联性的常见形式。

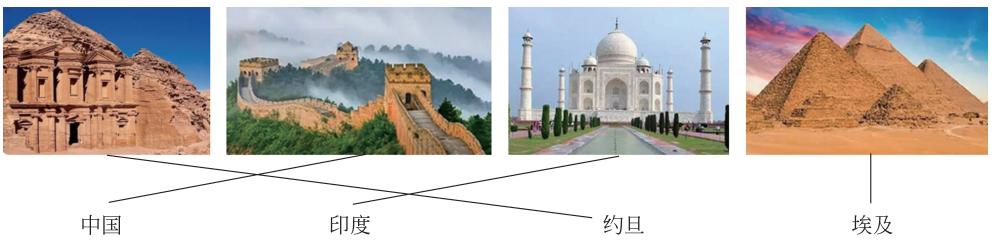

图 3-1　数据关联性示例

有时候离开了关联，甚至会导致数据失去了其原本的含义。例如，可以用一个列表表示科目 ['语文','数学','美术']，用另一个列表表示分数 [85,92,90]，如果分数没有与科目建立关联，那么它就没有明确的意义。所以我们看到的分数表达往往是这样的形式：语文 :85，数学 :92，美术 :90。这种数据之间的关联，被称为“映射”。Python 语言中的字典，就是一种可以存放这种具有映射关系数据的数据类型。

3.1　字典数据类型

Python 用字典保存具有映射关系的数据。字典相当于保存了两组数据，其

中一组数据是关键数据，被称为键（key）；另一组数据可通过 key 来访问，被称为值（value）。字典类型满足了通过书名查询作者、通过国家查询首都、通过商品查询价格等具有关联关系的信息查询需求。

“学会运用字典，我们就可以通过 Python 编程，为‘嫦娥’和‘玉兔’建立百科知识小卡片了。”

在 Python 中，用一对大括号“{ }”存放字典的键值对，其中，键和值之间用冒号“:”连接，键值对之间用逗号“,”隔开，键值对之间是没有顺序的。一个字典中可以有 0 个或者多个键值对。字典的定义形式如图 3-2 所示。

图 3-2　字典数据类型结构

Python 的字典类型具有以下特点。

（1）字典是可变的，即字典中的键值对是可以添加、删除的。

（2）字典以键值对为元素，且字典的元素是无序的。

（3）字典中的键不允许重复，但是值可以重复。

（4）字典中的键必须不可变，比如是数字、字符串、元组等不可变数据类型。字典中的值可以是不可变类型，也可以是列表等可变类型。

“老师，Python 中的字典，与我们使用的《新华字典》都叫‘字典’，它们之间有什么联系吗？”

“小萌，我们查《新华字典》时，是根据偏旁部首或者拼音查找到相应的汉字。而 Python 的字典是根据键去查找值，这两者是不是很相似？用好 Python 的字典，我们就能更好地管理信息。”

【问题 3-1】　在集合中，集合的元素不能重复，例如，设置集合 s={1,1,2,3}，最后集合会自动去除重复的元素，形成的集合形如 {1,2,3}。字典也不允许键的重复，如果有键值重复的字典定义，会出现什么样的结果呢?

```
d = {'作者':'白居易', '朝代':'唐', '作者':'李白'}
```

【问题 3-2】　关于字典的叙述，正确的是（　　）。

A. 字典的键与值都要保证唯一性，不允许重复
B. 列表可以作为字典的键
C. 由键和值组成的键值对构成，是无序的
D. 由键和值组成的键值对构成，是有序的

3.2　字典的创建及访问

1. 创建字典

在程序中创建字典，主要通过下面两种方法。

（1）直接在大括号“{}”中指定若干组键值对，如果大括号中没有键值对，即创建一个空字典。如下代码展示了字典的直接创建过程。

```
>>> d1={}                  # 空字典的创建
>>> type(d1)               # 确定 d1 的数据类型
<class 'dict'>             # 输出 d1 的类型，即字典类型名为 'dict'
```

```
>>> d2 = {2022:'北京',2028:'罗马',2028:'洛杉矶'}
>>> d2
{2022: '北京', 2028: '洛杉矶'}  # 当键值重复时，保留最后一个键值对
>>> d3 = {('A','B'):[90,80],'C':70}
>>> d3      # 元组是不可变类型，可以作为键；列表是可变类型，可以作为值
{('A', 'B'): [90, 80], 'C': 70}
```

（2）使用内置函数 dict( ) 创建字典。若 dict( ) 不加参数，则创建一个空字典，若参数具有映射的参数，则建立与映射关系相一致的具有键值对的字典。

```
>>> dict1 = dict()          # 用 dict() 函数创建空字典
>>> print(dict1)
{}
>>> dict2 = dict(火箭 = '长征5号', 高度=57)  # 具有映射关系的参数
                                            # 创建字典
>>> dict2
{'火箭': '长征5号', '高度': 57}
>>> dict3 = dict([('湖泊','青海湖'),('平均水深（米）',21)])
>>> dict3                   # 实现了将序列创建为字典
{'湖泊': '青海湖', '平均水深（米）': 21}
```

“老师，Python 中的字典与集合都用大括号‘{}’表示，它们之间存在着联系吗？”

“字典和集合都是无序的，字典的键以及集合的元素都具有唯一性且不能是可变的数据类型。所以集合本质上就是没有值（value）的字典。”

2. 访问字典

生活中人们使用字典主要是用于查找生字，Python 中使用字典数据类型，一个主要的目的也是获得字典中的信息，即“访问”字典。Python 中字典的访

问主要有以下几种方式。

（1）根据键访问值。

字典中每个元素表示一种映射关系，将提供的“键”作为索引，可以访问对应的值，如果字典中不存在这个“键”，则会返回错误信息。例如下列代码，演示了对存在于字典中键的访问，以及访问字典中不存在的键而引发的异常。

```
d ={'杭州':'西湖', '北京':'长城', '西安':'兵马俑'}
print("(1)",end =":")
print(d['北京'])            #'北京'是字典中存在的键
print("(2)",end =":")
print(d['天津'])            #'天津'是字典中不存在的键
```

执行结果如下。

```
(1):长城
(2):Traceback (most recent call last):
  File "Demo.py", line 5, in <module>
    print(d['天津'])
KeyError: '天津'
```

（2）使用 get( ) 方法访问值。

为了避免在访问字典时，因为访问的某个键不在字典中而引发异常，可通过 get( ) 方法进行值的访问。get( ) 方法的语法格式如下。

```
dict.get([key[,default = None])
```

其中，key 为要访问的键，若该键存在，则返回对应的值；若该键不存在，则返回默认值。可自行设置需要输出的默认值，default=None 代表不指定默认值时，默认值为 None。下列代码展示了利用 get( ) 方法访问字典的情形。

```
d ={'杭州':'西湖', '北京':'长城', '西安':'兵马俑'}
print("(1):",d. get('北京'))        # 访问存在的键
print("(2):",d. get('天津'))        # 访问不存在的键，默认值为 None
print("(3):",d. get('天津','天津标识景区未记录')) # 自行设定默认值
```

运行结果如下。

```
(1)：长城
(2)：None
(3)：天津标识景区未记录
```

“其实呀，获取字典中键、值以及键值对的方法是很多的。就拿‘访问’键来说，我们通过循环，进行字典的遍历，就能够轻松实现了。”

“上面的例子‘访问’到的都是字典的值，我们有没有方法可以访问字典的键呢？”

for 循环可以遍历字符串、列表、元组，同样也可以遍历字典。在遍历字典时，每次的循环变量其实就是字典的索引值，就是键。以下列代码为例。

```
d ={'杭州':'西湖', '北京':'长城', '西安':'兵马俑'}
for i in d:
    print(i)
```

代码的执行结果如下。

```
杭州
北京
西安
```

【问题 3-3】 用 for 循环遍历字典，可以获取键的信息，既然键与值是映射的关系，那么能否通过 for 循环遍历字典，获得键对应的值的信息呢？请仿照上面的代码，尝试自己动手编程实现吧！

3.3　键值对数据处理

字典除了支持创建和访问等基本操作，作为可变的映射数据类型，字典还拥有丰富的键值对数据的处理能力。

字典除了可以通过键作为索引访问值、通过遍历的方法访问键以外，还提供了三种视图对象，通过它们可以动态访问字典中的数据。这三种视图的获取方法如图 3-3 所示。

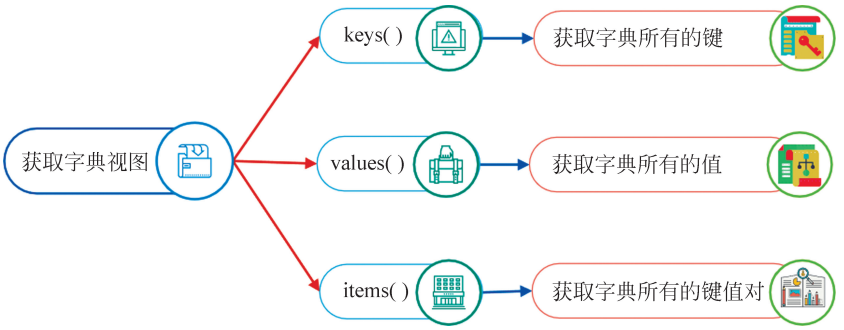

图 3-3　获取字典视图对象的三种方法

若字典名为 d，则 d. keys( )、d. values( )、d. items( ) 都是可以使用 for 循环遍历的可迭代的对象。下列代码用于测试输出上述三种视图对象的类型名。

```
d = {}                  # 创建空字典
print("d. keys 的类型名为 {}".format(type(d. keys())))
print("d. valus 的类型名为 {}".format(type(d. values())))
print("d. items 的类型名为 {}".format(type(d. items())))
```

在输出结果中可以清晰地呈现三种视图的类型名。

```
d. keys 的类型名为 <class 'dict_keys'>
d. valus 的类型名为 <class 'dict_values'>
d. items 的类型名为 <class 'dict_items'>
```

dbook = {' 书名 ':' 平凡的世界 ',' 作者 ':' 路遥 ',' 出版年 ':1990} 是一个定义好的字典，若有如下三个输出语句：

```
    print(dbook.keys())
    print(dbook.values())
    print(dbook.items())
```

则得到如下输出结果。

```
    dict_keys(['书名', '作者', '出版年'])
    dict_values(['平凡的世界', '路遥', 1990])
    dict_items([('书名', '平凡的世界'),('作者', '路遥'),('出版年', 1990)])
```

可见三种视图返回值均为列表，items( ) 方法返回的是列表，它的列表元素为键与值构成的元组。

字典作为可变数据类型，其键值对信息均可进行添加、修改、删除等操作。字典中的键值对添加及修改可以通过赋值的方式直接实现。

若有字典 cd ={'北京':'颐和园'},执行语句 cd['北京'] = '故宫' 后，字典中键 '北京' 对应的值将被改为 '故宫';执行语句 cd['天津'] = '盘山' 后，键值对 '天津':'盘山' 将被添加到字典 cd 中。

字典的删除类命令及方法如表 3-1 所示。

表 3-1　字典删除的命令及方法

方法 / 命令	说　　明	示例 基于字典 d = {'语文':90,'数学':95}
del	删除字典或键值对的命令	(1) del d　# 直接删除整个字典 d (2) del d['语文'] # 删除 '语文':90 (3) del d['英语'] # 将引发 KeyError 错误
pop()	删除指定键值对，至少有 1 个参数	(1) d. pop('数学') # 返回值 95，并删除 '数学':95 键值对 (2) d. pop('英语','查找不到') #'英语' 不是字典中的键，返回 '查找不到'
popitem()	随机删除某个键值对的方法，无参数	d. popitem() 返回结果有可能为 ('数学', 95)。当字典为空时，再运行 popitem() 会引发 KeyError 错误
clear()	清空字典中所有键值对的方法，无参数	d. clear() 执行后 d 将成为一个空字典

练一练

【问题 3-4】　阅读下列代码，请写出运行结果。

```
d = {1:'a', 2:'b', 3:'c'}
del d[1]
d[1] = 'x'
del d[2]
print(d)
```

普遍运用于组合数据类型的 len( ) 函数、in 与 not in 等成员运算符，同样适用于字典数据类型。

len( ) 函数若以字典为参数，则返回字典的长度，即字典中键值对的数量。而成员运算仅适用于判断键是否在字典中，返回值为 True 或者 False。

上述操作是支持字典类型的主要命令及方法，Python 更多关于“字典”的奥秘，等着大家去探索！

“本单元完成了对字典的学习。字典是 Python 唯一的映射类型。字典的访问、遍历、修改、维护等都是围绕键值对展开的。键的唯一性决定了键不能够用可变类型数据来表达。

在课后，请同学们注意对比列表、元组、集合，深入理解字典的特性，为后续课程学习打牢基础。”

习　题

1. 下列选项中，不能生成空字典的是（　　）。

   A. { }　　B. dict( )　　C. {[ ]}　　D. dict([ ])

2. 运行下列代码，输出结果是（　　）。

```
dc = {'中国':'黄河', '埃及':'尼罗河'}
print(len(dc))
```

A. 2　　B. 3　　C. 4　　D. 5

3. 下列对字典的定义，错误的是（　　）。

A. dc = {'姓名':'张柔', '年龄':12}

B. dc = {(1,2):'元组','abc':'字符串'}

C. dc = {{1,2}:'集合',12:'数字'}

D. dc = {12.3:12,35.6:36}

4. 运行下列代码，输出结果是（　　）。

```
d_demo ={"任务":"嫦娥3号", "发射年份":2013, "目标":"巡月"}
print(d_demo.values())
```

A. dict_values(['嫦娥3号', 2013, '巡月'])

B. dict_values(['任务', '发射年份', '目标'])

C. dict_keys(['嫦娥3号', 2013, '巡月'])

D. dict_keys(['任务', '发射年份', '目标'])

5. 已知字典定义为 d_lw = {'小桔灯':'冰心', '故乡':'鲁迅', '春蚕':'茅盾'}，下列语句能够正确执行的是（　　）。

A. print(d_lw.get( ))　　B. print(d_lw.pop( ))

C. print(d_lw.popitem( ))　　D. print(d_lw.del( ))

6. 已知字典 dic1 = {'中国':'北京', '法国':'巴黎', '英国':'伦敦'}，执行下列语句，输出结果中不包括'北京'的是（　　）。

A. dic1.get('中国')　　B. dic1.keys( )

C. dic1.values( )　　D. dic1.items( )

7. 运行下列代码，输出结果是（　　）。

```
d_demo = {'语文':90, '数学':75, '历史':80}
print(80 in d_demo)
```

A. 1　　B. 0　　C. True　　D. False

8. 运行下列代码，输出结果是（　　）。

```
d_1 = {'a':2, 'b':4}
d_2 = d_1
```

```
d_1['a'] = 5
print(d_1['a']+d_2['a'])
```

A. 4　　B. 6　　C. 7　　D. 10

9. 以下关于字典的描述，错误的是（　　）。

A. 字典中元素以键信息为索引访问

B. 字典长度是可变的

C. 字典是键值对的集合

D. 字典中的键可以对应多个值信息

10. 运行下面代码，输出结果是（　　）。

```
d ={'匆匆':'诗歌', '母亲':'散文', '边城':'小说'}
print(d['母亲'], d. get('母亲', '随笔'))
```

A. 散文 随笔　　B. 散文 散文　　C. 随笔 随笔　　D. 散文 诗歌

11. 编程题

有字典定义为：dlog = {'user':'135aba','guest':'test001'}。

用户输入用户名和密码，当用户名与密码和字典中的键值对一致时，显示“登录成功”，否则显示“登录失败”，若登录失败，允许重复输入三次。

输入格式：

在两行中分别输入用户名和密码。

输出格式：

" 登录成功 " 或 " 登录失败 "