



# Python编程案例教程

# 第6章 字典和集合

## 本章导读

在实际开发过程中，我们会遇到需要将相关数据关联起来的情况，例如，处理学生的学号、姓名、年龄、成绩等信息。另外，还会遇到需要将一些能够确定的不同对象看成一个整体的情况。Python提供了字典和集合这两种数据结构来解决上述问题。

本章首先介绍字典的创建和基本操作方法，然后介绍字典的遍历和嵌套，接着介绍集合的概念以及常用操作方法，最后通过两个典型案例的分析和实现，让读者进一步掌握在程序中使用字典和集合的方法。



# 学习目标

- 掌握字典的创建方法
- 掌握字典的基本操作方法
- 掌握字典的遍历方法
- 掌握字典的嵌套使用方法
- 理解集合的概念
- 掌握集合的创建及基本操作方法

The background of the slide features an abstract design composed of overlapping, semi-transparent red polygons of various shades, creating a dynamic, layered effect on the left side of the frame.

## 6.1 字典的创建和访问

## 6.2 字典的基本操作

## 6.3 字典的遍历

## 6.4 嵌套

## 6.5 集合

## 6.6 典型案例

# 6.1 字典的创建和访问

---

◆ 6.1.1 字典的创建

◆ 6.1.2 字典的访问

### 6.1.1 字典的创建

字典是Python中常用的一种数据存储结构，它是由“键-值”对组成，每个“键-值”对称为一个元素，每个元素表示一种映射或对应关系。

- ◆ “键” 可以是Python中任意不可变数据，如整数、实数、复数、字符串、元组等类型，但不能使用列表、集合、字典或其他可变类型作为字典的“键”。
- ◆ “值” 可以取任意数据类型。



## 6.1.1 字典的创建

### ► 1 . 直接赋值创建字典

直接赋值创建字典的一般格式如下：

```
变量名 = {键1:值1, 键2:值2, 键3:值3,...}
```



**例如：** 创建一个学生信息字典，包括学生学号、姓名和性别三个元素。

```
stu_info = {'num':'20180101', 'name':'Liming', 'sex':'male'} #创建字典
```

## 6.1.1 字典的创建

### ► 1. 直接赋值创建字典

提示

字典中元素打印出来的顺序与创建时的顺序不一定相同，这是因为字典中各个元素并没有前后顺序。——不属于序列结构



**例如：** 创建一个学生信息字典，包括学生学号、姓名和性别三个元素。

```
>>>stu_info = {'num':'20180101', 'name':'Liming', 'sex':'male'} #创建字典
>>>stu_info #查看字典
{'name':'Liming', 'num':'20180101', 'sex':'male'}
```



## 6.1.1 字典的创建

### ► 2 . 使用内置函数dict()创建字典

dict()参数形式:

- 其他 “字典” ；
- “(键,值)” 对的元组序列 ；
- 关键字参数
- zip函数组合两个列表

## 6.1.1 字典的创建

### ► 2 . 使用内置函数dict()创建字典

例：使用内置函数dict()创建字典。

```
stu_info1 = dict( {'num':'20180101', 'name':'Liming', 'sex':'male'})    #通过其他字典创建
stu_info2 = dict([('num', '20180101'), ('name', 'Liming'), ('sex', 'male')])    #通过“(键,值)”
对的序列创建
stu_info3 = dict(num = '20180101', name = 'Liming', sex = 'male')    #通过关键字参数创建
stu_info4 = dict(zip(['num', 'name', 'sex'], ['20180101', 'Liming', 'male']))    #通过dict和zip
结合创建。只能是2个列表，不能3个；列表长度不一，按段的处理
if stu_info1 == stu_info2 == stu_info3 == stu_info4 :                #判断五个变量是否相等
    print("创建字典的5种方式相同")                                    #如果相同输出提示符
else:                                                                    #如果不相同
    print("创建字典的5种方式不相同")                                    #输出提示符
```

## 6.1.1 字典的创建

### 知识库

zip()函数将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，元素个数与最短的列表一致，然后返回由这些元组组成的zip对象。

例如：

```
>>>list_str = ['a', 'b', 'c', 'd']
```

```
>>>list_num = [1, 2, 3, 4]
```

```
>>>list_new = zip(list_str, list_num)
```

```
>>>print("zip结果(列表)：", list(list_new))
```

```
zip结果(列表)： [('a', 1), ('b', 2), ('c', 3), ('d', 4)]
```

```
#创建列表并赋值
```

```
#创建列表并赋值
```

```
#打包为元组组成的zip对象
```

```
#用list()函数转换为列表输出
```

## 6.1.1 字典的创建

### ► 3 . 使用fromkeys()方法创建字典

在Python中，当所有键对应同一个值时，可使用fromkeys()方法创建字典。

```
dict.fromkeys(seq[,value])
```

◆seq为字典“键”值列表

◆value为设置键序列（seq）的值，省略时默认为None

例如：

```
>>>stu_age1=dict.fromkeys(['Wangwu','Zhangsan'])
```

```
>>>stu_age1                                     #输出stu_age1
```

```
{'Wangwu':None, 'Zhangsan':None}
```

```
>>>stu_age2=dict.fromkeys(['Wangwu','Zhangsan'],'18')
```

```
>>>stu_age2                                     #输出stu_age2
```

```
{'Wangwu':'18', 'Zhangsan':'18'}
```

## 6.1.1 字典的创建

### 知识库

字典中的“键”是唯一的，创建字典时若出现“键”相同的情况，则后定义的“键-值”对将覆盖先定义的“键-值”对。



例如：

```
>>>x = {'a':1, 'b':2, 'b':3}
```

#直接赋值创建字典x

```
>>>x
```

#输出字典x

```
{'a':1,'b':3}
```

## 6.1.1 字典的创建

### ▶ 4 . json字符串创建字典

例如：

```
import json
stu_info={'num':'中文','name':'Yinbing','age':'17'}
jsonStr=json.dumps(stu_info,ensure_ascii=False)
print(jsonStr)
NewDic=json.loads(jsonStr.encode("utf-8"))#创建字典
print(type(NewDic))
```

## 6.1.2 字典的访问

### ► 1. 根据键访问值

字典中的每个元素表示一种映射关系，将提供的“键”作为下标可以访问对应的“值”，如果字典中不存在这个“键”则会抛出异常。其语法格式如下：

例如：`字典变量名[键]`

```
>>>stu_info = {'num':'20180105', 'name':'Yinbing', 'sex':'male'} #创建字典
```

```
>>>stu_info['num'] #根据num访问学号 '20180105'
```

```
>>>stu_info['age'] #指定的键不存在抛出异常
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

## 6.1.2 字典的访问

### ► 2. 使用get()方法访问值

在访问字典时，若**不确定**字典中是否有某个键，可通过**get()方法**进行获取，若该键存在，则返回其对应的值，若不存在，则返回默认值。其语法格式如下：

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

◆ dict为被访问字典名

◆ key是要查找的键

◆ default定义默认值，如果指定键的值不存在，返回该默认值，当default为空时，返回None



## 6.1.2 字典的访问

### ► 2 . 使用get()方法访问值

例如 :

```
>>>stu_info.get('name')           #get()获取学生姓名
'Yinbing'

>>>stu_info.get('age')              #get()获取学生年龄，返回值为None
>>>print(stu_info.get('age'))       #输出返回值None
None

>>>stu_info.get('age',18)           #设置返回默认值为18
18
```

The background of the slide is an abstract composition of various shades of red and pink. It features a complex pattern of overlapping triangles and polygons, creating a sense of depth and movement. The colors range from deep, dark reds to lighter, almost white pinks, with some areas showing a gradient effect. The overall style is modern and geometric.

## 6.2 字典的基本操作

## ► 1. 修改和添加元素

当以指定“键”为下标为字典元素赋值时，有两种含义：

- （1）若该“键”在字典中存在，则表示修改该“键”对应的值；
- （2）若该“键”不存在，则表示添加一个新的“键-值”对，也就是添加一个新元素到字典中。

## ► 1. 修改和添加元素

例如：

```
>>> stu_info={'num':'20180105','name':'Yinbing','age':'17'}      #创建字典
>>> stu_info['age']='18'                                           #修改age的值
>>> stu_info['age']                                                #输出修改后的age值
'18'
>>> stu_info['sex']='male'                                         #添加学生性别
>>> stu_info                                                       #输出添加后的字典
{'num': '20180105', 'name': 'Yinbing', 'age': '18', 'sex': 'male'}
```

## ► 2 . 删除元素

要删除字典中的元素或整个字典，可以使用del命令、clear()、pop()和popitem()方法。

1) **del命令**：可根据“键”删除字典中的元素。

例如：

```
>>> stu_info={'num':'20180105','name':'Yinbing','age':'17'}      #创建字典
>>> del stu_info['age']                                           #删除age及其值
>>> stu_info                                                       #输出删除后的字典
{'num': '20180105', 'name': 'Yinbing'}
```

2) `clear()`方法：用于清除字典中的所有元素。

```
dict.clear()
```

◆ `dict`为要被清空的字典名

◆ 该方法不包含任何参数，也没有返回值



例如：

```
>>>stu_info={'num':'20180105','name':'Yinbing','age':'17'}
```

#创建字典

典

```
>>>stu_info.clear()
```

#清空字典

```
>>>stu_info
```

#输出清空后的字典

3) `pop()`方法：用于获取指定“键”的值，并将这个“键-值”对从字典中移除。

`dict.pop(key[,default])`

◆ `dict`为要被删除元素的字典名

◆ `key`是要被删除的键

◆ `default`是默认值，当字典中没有要被删除的`key`时，该方法返回指定的默认值

例如：

```
>>> stu_info={'num':'20180105','name':'Yinbing','age':'17'}#创建字典
>>> stu_info.pop('age') #返回并删除age “键-值” 对
'17'
>>> stu_info.pop('age',18) #无指定键，返回默认值18
```

4) `popitem()`方法：用于随机获取一个“键-值”对，并将其删除。

`dict.popitem()`

◆ `dict`为要被删除元素的字典名

◆ 该方法无参数，返回值为一个随机的“键-值”对

例如：

```
>>> stu_info={'num':'20180105','name':'Yinbing','age':'17'} #创建字典
```

```
>>> stu_info.popitem() #随机返回某“键-值”对并删除
```

```
('age', '17')
```

```
>>> stu_info #输出字典
```

```
{'num': '20180105', 'name': 'Yinbing'}
```



## ► 3 . 更新字典

`update()`方法：可以将新字典的“键-值”对一次性全部添加到当前字典中，如果两个字典中存在相同的“键”，则以新字典中的“值”为准更新当前字典。

`dict.update(dict2)`

- ◆ dict为当前字典
- ◆ dict2为新字典

## ► 3 . 更新字典

例如：

```
>>>stu_info = {'num':'20180105', 'name':'Yinbing', 'age':'17'}    #创建字典
>>>stu_info.update({'age':'18', 'sex':'male'}) #修改age的值，同时添加新元素
>>>stu_info                                                         #输出字典
{'num':'20180105', 'name':'Yinbing', 'age':'18', 'sex':'male'}
```

## ► 4 . 复制字典

复制字典可调用copy()方法，copy()方法返回字典的浅复制。

dict.copy()

- ◆ dict为需要复制的字典
- ◆ 该方法无参数，返回值为一个新字典

例如：

```
>>> stu_info={'num':'20180105','name':'Yinbing','age':'17'}  
>>> stu_info.copy()  
{'num': '20180105', 'name': 'Yinbing', 'age': '17'}
```

## ► 4 . 复制字典

### 知识库

在Python3中，可以用三种方法复制字典：直接赋值、浅复制和深复制。

( 1 ) 直接赋值：对象的引用。

( 2 ) 浅复制 ( `copy()`方法 )：拷贝父对象，引用对象内部的子对象。

( 3 ) 深复制 ( `deepcopy()`方法 )：`copy`模块的`deepcopy()`方法，完全复制父对象及其子对象。

例：直接赋值、浅复制和深复制的区别。

```
dict1 = {'user':'runoob','num':[1,2,3]}
```

```
dict2 = dict1                #引用对象
```

```
dict3 = dict1.copy() #浅复制，深复制父对象，子对象不复制，还是引用
```

```
import copy
```

```
dict4 = copy.deepcopy(dict1) #深复制，完全复制父对象及其子对象
```

```
dict1['user'] = 'root'        #将dict1中键为'user'的值改为'root'
```

```
dict1['num'].remove(1)        #移除dict1中键为'num'的列表值中的1
```

```
print('dict1=',dict1)
```

```
print('dict2=',dict2)
```

```
print('dict3=',dict3)
```

```
print('dict4=',dict4)
```

```
dict1= {'user': 'root', 'num': [2, 3]}
dict2= {'user': 'root', 'num': [2, 3]}
dict3= {'user': 'runoob', 'num': [2, 3]}
dict4= {'user': 'runoob', 'num': [1, 2, 3]}
[Finished in 0.2s]
```

## ► 4 . 复制字典

### 知识库

在Python3中，可以用三种方法复制字典：直接赋值、浅复制和深复制。

( 1 ) 直接赋值：对象的引用。

( 2 ) 浅复制 ( `copy()`方法 )：拷贝父对象，引用对象内部的子对象。

( 3 ) 深复制 ( `deepcopy()`方法 )：`copy`模块的`deepcopy()`方法，完全复制父对象及其子对象。

## ► 5补充 . 如何克隆字典结构

### 知识库

```
keyList=srcDict.keys()
```

```
NewDict=dict.fromkeys(keyList,None)
```

## **6.3 字典的遍历**

---

- ◆ 6.3.1 遍历字典中所有的“键-值”对
- ◆ 6.3.2 遍历字典中所有的键
- ◆ 6.3.3 遍历字典中所有的值



### 6.3.1 遍历字典中所有的“键-值”对

遍历字典中所有的“键-值”对需要用到`items()`方法，该方法以列表形式返回`dict.items()`

例：遍历字典中所有的“键-值”对。

```
stu_class = { 'Mary':'C', 'Jone':'Java', 'Lily':'Python', 'Tony':'Python' }  
for name, cla in stu_class.items():      #遍历 “键-值” 对  
    print(name,'选修的是',cla)          #输出每个值
```

## 6.3.2 遍历字典中所有的键

当不需要使用字典中的值时，可使用`keys()`方法只遍历字典中的键，该方法以列表返回一个字典中所有的键。

`dict.keys()`

例：遍历字典中所有的键。

```
stu_class = { 'Mary':'C', 'Jone':'Java', 'Lily':'Python', 'Tony':'Python' }  
for name in stu_class.keys():  
    print(name)
```

#遍历字典所有的键  
#输出每个键

```
Mary  
Jone  
Lily  
Tony  
[Finished in 0.2s]
```

### 6.3.3 遍历字典中所有的值

当只关心字典所包含的**值**时，可使用**values()**方法，该方法以列表形式返回字典中所有的值。

**dict.values()**

例：遍历字典中所有的值。

```
stu_class = { 'Mary':'C', 'Jone':'Java', 'Lily':'Python', 'Tony':'Python' }
```

```
print('以下课程已被选择：')
```

```
for cla in stu_class.values():
```

```
    print(cla)
```

#遍历字典所有的值

#输出每个值

```
以下课程已被选择：
C
Java
Python
Python
[Finished in 0.2s]
```

## 6.4 嵌套

- ◆ 6.4.1 在列表中嵌套字典
- ◆ 6.4.2 在字典中嵌套列表
- ◆ 6.4.3 在字典中嵌套字典

## 6.4.1 在列表中嵌套字典

例：在列表中嵌套字典。

```
student_info1 = {'name':'Wangmi', 'sex':'F', 'age':'15'}
```

```
student_info2 = {'name':'Linmei', 'sex':'M', 'age':'14'}
```

```
student_info3 = {'name':'Chenhui', 'sex':'F', 'age':'14'}
```

```
student = [student_info1, student_info2, student_info3] #创建包含三个学生的列表
```

```
for s in student:
```

```
#遍历列表
```

```
    print(s)
```

```
{'name': 'Wangmi', 'sex': 'F', 'age': '15'}  
{'name': 'Linmei', 'sex': 'M', 'age': '14'}  
{'name': 'Chenhui', 'sex': 'F', 'age': '14'}  
[Finished in 0.2s]
```



7 lines, 355 characters selected

Tab Size: 4

## 6.4.2 在字典中嵌套列表

例：在字典中存储列表。

```
stu_class = {  
    'Mary':['C','Math'],  
    'Jone':['Java','Art'],  
    'Lily':['Python'],  
    'Tony':['Python','Mysql','Math']  
}
```

#定义字典并赋值，字典中的值为列表

```
for name,cla in stu_class.items():
```

#遍历字典所有的元素

```
    print(name,'选的课程是:',)
```

#输出键

```
        for c in cla:
```

#遍历列表

```
            print(c)
```

#输出列表中的值

```
Mary 选的课程是:  
C  
Math  
Jone 选的课程是:  
Java  
Art  
Lily 选的课程是:  
Python  
Tony 选的课程是:  
Python  
Mysql  
Math  
[Finished in 0.2s]
```

10 lines, 303 characters selected

### 6.4.3 在字典中嵌套字典

例：在字典中嵌套字典。

```
stu_info = {  
    'WangMi':{'sex':'F','age':'15'},  
    'LinMei':{'sex':'M','age':'14'},  
    'ChenHui':{'sex':'F','age':'14'}  
}  
#定义字典并赋值  
for name, stu in stu_info.items():  
    #遍历字典所有元素  
    print(name, '性别', stu['sex'], '年龄', stu['age'])  
    #输出键和值
```

```
WangMi 性别 F 年龄 15  
LinMei 性别 M 年龄 14  
ChenHui 性别 F 年龄 14  
[Finished in 0.2s]
```

# 6.5 集 合

◆ 6.5.1 集合的创建

◆ 6.5.2 集合的基本操作



集合（set）与数学中集合的概念一致，即包含0个或多个数据项的无序组合。

- ◆ 集合中的元素不可重复
- ◆ 元素类型只能是固定数据类型，如整数、浮点数、字符串、元组等
- ◆ 不能是列表、字典和集合等可变数据类型

## 知识库

Python提供了一个内置函数hash()来计算对象的哈希值，凡是无法计算哈希值（调用内置函数hash()时抛出异常）的对象都不能作为集合的元素，也不能作为字典对象的“键”。

```
>>>hash('Python')  
-5336376190899570360
```

#计算字符串的哈希值

```
>>>hash(1)  
1
```

#计算整型数据的哈希值

```
>>>hash([1,2,3])
```

#计算列表的哈希值，抛出异常

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: unhashable type: 'list'
```

## 6.5.1 集合的创建

### ► 1. 直接创建

创建集合的方式很简单，只需将逗号分隔的不同元素使用大括号括起来即可。

例如：

```
>>>a_set = {1, 2, 3, 4}                #创建集合并赋值
>>>a_set                                #输出集合a_set
{1, 2, 3, 4}
>>>b_set = {2, 1, 3, 4, 1, 2}          #创建集合并赋值
>>>b_set                                #输出集合b_set
{1, 2, 3, 4}
```

由于集合元素是无序的，集合的打印效果与定义顺序可以不一致，且由于集合元素独一无二，使用集合类型能够过滤掉重复元素。

## 6.5.1 集合的创建

### ► 2. 使用set()函数

**set()函数**：将列表、元组等其他类型的数据转换为集合，如果原来的数据中存在重复元素，则在转换为集合时会将其删除。

例如：

```
>>>x = set('runoob')           #将字符串转换为集合
>>>x                           #输出集合x {'o', 'u', 'n', 'b', 'r'}
>>>y = set(['g', 'o', 'o', 'g', 'l', 'e'])   #将列表转换为集合
>>>y                           #输出集合y {'g', 'o', 'e', 'l'}
>>>z = set()                   #空集合
>>>z                           #输出集合z
set()
```

## 6.5.1 集合的创建

### 提示

集合类型与其他类型最大的不同在于它不包含重复元素，因此，当需要对一维数据进行去重处理时，一般可通过集合来完成。

## 6.5.1 集合的创建

例：将输出数据去除重复项后输出。

```
stu_class = {  
    'Mary':'C',  
    'Jone':'Java',  
    'Lily':'Python',  
    'Tony':'Python'  
}
```

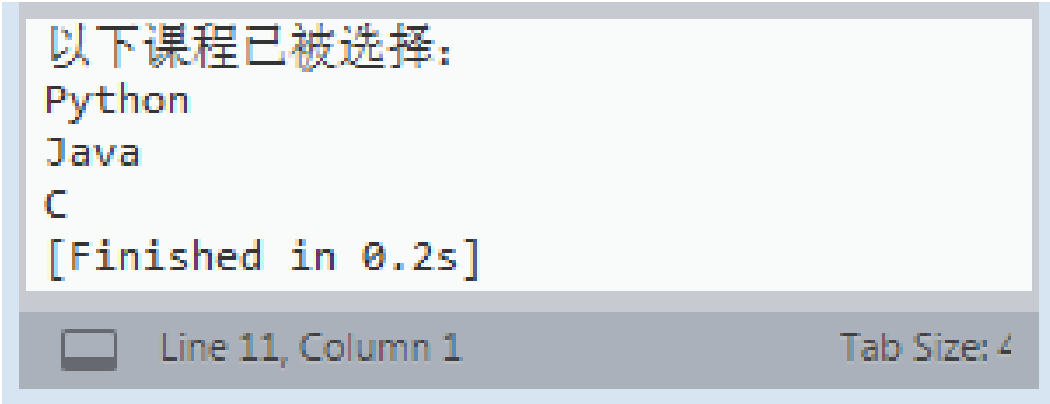
#定义字典并赋值

```
print('以下课程已被选择：')
```

```
for cla in set(stu_class.values()): #遍历字典所有的值，调用set()将列表转换为集合从而去除重复项
```

```
    print(cla)
```

#输出每个值



```
以下课程已被选择:  
Python  
Java  
C  
[Finished in 0.2s]
```



Line 11, Column 1

Tab Size: 4

## 6.5.2 集合的基本操作

### ► 1. 添加与删除集合元素

操作函数或方法	描 述
<b>S.add(x)</b>	如果数据项x不在集合S中，将x添加到S中
<b>S.update(T)</b>	合并集合T中的元素到当前集合S中，并自动去除重复元素
<b>S.pop()</b>	随机删除并返回集合中的一个元素，如果集合为空则抛出异常
<b>S.remove(x)</b>	如果x在集合S中，移除该元素；如果x不存在则抛出异常
<b>S.discard(x)</b>	如果x在集合S中，移除该元素；如果x不存在不报错
<b>S.clear()</b>	清空集合

## 6.5.2 集合的基本操作

例如：

```
>>>s = {1, 2, 3}           #创建集合并赋值
>>>s.add(4)                #添加元素
>>>s.update({4, 5, 6})     #更新当前集合，自动忽略重复元素
>>>s                       #输出集合{1, 2, 3, 4, 5, 6}
>>>s.discard(5)            #删除元素，不存在则忽略该操作
>>>s                       #输出集合{1, 2, 3, 4, 6}
>>>s.remove(5)             #删除元素，不存在则抛出异常
KeyError:5
>>>s.pop()                #删除并返回一个元素 1
```



## 6.5.2 集合的基本操作

### ► 2. 集合运算

内置函数len()、max()、min()、sorted()等也适用于集合，另外，Python集合还支持数学意义上的交集、并集、差集、补集等运算。

操作符	描述
$S \& T$	交集，返回一个新集合，包括同时在集合S和T中的元素
$S   T$	并集，返回一个新集合，包括集合S和T中的所有元素
$S - T$	差集，返回一个新集合，包括在集合S中但不在集合T中的元素
$S \wedge T$	补集，返回一个新集合，包括集合S和T中的元素，但不包括同时在集合S和T中的元素
$S \leq T$	如果S与T相同或S是T的子集，返回True，否则返回False，可以用 $S < T$ 判断S是否是T的真子集
$S \geq T$	如果S与T相同或S是T的超集，返回True，否则返回False，可以用 $S > T$ 判断S是否是T的真超集

## 6.5.2 集合的基本操作

例如：

```
>>>a_set = {1, 2, 3, 4, 5}
>>>b_set = {1, 2, 6, 7, 8}
>>>a_set & b_set
>>>a_set | b_set
>>>a_set - b_set
>>>a_set ^ b_set
>>>x = {1, 2, 3}
>>>y = {2, 3}
>>>z = {1, 2, 4}
>>>x >= y
>>>x <= z
```

#创建集合并赋值

#创建集合并赋值

#交集{1, 2}

#并集{1, 2, 3, 4, 5, 6, 7, 8}

#差集{3, 4, 5}

#补集{3, 4, 5, 6, 7, 8}

#创建集合并赋值

#创建集合并赋值

#创建集合并赋值

#y是否为x的子集True

#x是否为z的子集False

The background of the slide features an abstract design with overlapping red and pink geometric shapes, primarily triangles and polygons, creating a dynamic and modern look. The colors range from deep red to light pink, with some areas appearing more translucent than others.

## 6.6 典型案例

```
count =0
passwd=123
dict1={'alex':[passwd,count],'Tom':[passwd,count]}
while True:
    name = input("please input your name:")
    password = int(input("please input your password:"))
    if name not in dict1.keys():
        print("name %s is not in dict"%name)
        break
    if dict1[name][1] > 2:
        print("the name %s locked"%name)
        break
    if password == dict1[name][0]:
        print("login ok")
        break
    else:
        print("name or passwd error")
        dict1[name][1] +=1
```

#定义count变量并赋初值为0

#定义passwd变量并赋初值为123

#定义字典用于存储用户信息

#开始循环

#输入用户名

#输入密码

#如果输入的用户名不在字典中

#输出提示语

#跳出循环

#如果次数大于2

#输出被锁定提示信息

#跳出循环

#如果输入的密码正确

#输出登录成功提示语

#跳出循环

#密码输入错误

#输出提示语

#次数加1

## 6.6.1 登录验证

### 程序运行效果

```
6-10.py x PL* [python] x
please input your name:Tom
please input your password:123
login ok

***Repl Closed***
```

```
6-10.py x PL* [python] x
please input your name:Jack
please input your password:123
name Jack is not in dict

***Repl Closed***
```

```
6-10.py x *REPL* [python] x
please input your name:Tom
please input your password:456
name or passwd error
please input your name:Tom
please input your password:489
name or passwd error
please input your name:Tom
please input your password:458
name or passwd error
please input your name:Tom
please input your password:123
the name Tom locked

***Repl Closed***
```

# 典型案例

```
data = {
    '北京': {
        '昌平': {
            '沙河': ['沙河机场', '链家'],
            '天通苑': ['北方明珠', '天通尾货']
        },
        '朝阳': {
            '花家地': ['朝阳公园', '望京soho'],
            '北小河': ['北小河公园', '北京中学']
        }
    },
    '上海': {
        '虹桥': {
            '虹桥机场': ['超市', '特产店', '水吧'],
            '东方明珠': ['电影院', '游泳馆', '餐馆']
        },
        '浦东': {
            '景秀路': ['世纪公园', '立交桥'],
            '中环路': ['鲁迅公园', '同济大学']
        }
    },
    '河北': {
        '石家庄': {
            '行唐': ['东正', '阳关'],
            '赵县': ['赵州桥', '高村乡']
        },
        '唐山': {
            '滦南县': ['司各庄镇', '安各庄镇'],
            '玉田县': ['玉田镇', '亮甲店镇']
        }
    }
}
```

级菜

三  
县  
及  
呈  
序

```
while True:
    for i in data:                                     #打印第一级列表
        print(i)
        choice = input("请选择省或直辖市(退出请按q):")
        if choice in data:                             #如果在第一级列表里则进入下一级列表
            while True:
                for i2 in data[choice]:                 #打印第二级列表
                    print(i2)
                    choice2 = input("请选择(退出请按q返回省或直辖市列表请按b):")
                    if choice2 in data[choice]:         #如果在第二级列表里则进入下一级
                        while True:
                            for i3 in data[choice][choice2]: #打印第三级列表
                                print(i3)
                                choice3=input("请选择(退出请按q返回上一级列表请按b):")
                                if choice3 in data[choice][choice2]:
                                    for i4 in data[choice][choice2][choice3]:
                                        print(i4)
                                        choice4 = input("已经到达最后一级(退出请按q返回上一级列表请按b):")
                                        if choice4 == 'b':
                                            continue
                                        elif choice4 == 'q':
                                            exit()
                                    elif choice3 == 'b':                 #从第三级返回第二级
                                        break
                                    elif choice3 == 'q':
                                        exit()
                                elif choice2 == 'b':                 #从第二级返回第一级
                                    break
                                elif choice2 == 'q':
                                    exit()
                            elif choice == 'q':
                                exit()
```

## 6.6.2 三级菜单

程序运行效果

```
6-12.py x *REPL* [python] x
北京
上海
河北
请选择省或直辖市(退出请按q):北京
昌平
朝阳
请选择(退出请按q返回省或直辖市列表请按b):昌平
沙河
天通苑
请选择(退出请按q返回上一级列表请按b):沙河
沙河机场
链家
已经到达最后一级(退出请按q返回上一级列表请按b):b
沙河
天通苑
请选择(退出请按q返回上一级列表请按b):q

***Repl Closed***
```

**感谢您的观看**

