第二章 第四节 Python 列表、元组、集合、字典

Python 列表

Python 集合（数组）

Python 编程语言中有四种集合数据类型：

- 列表（List）是一种有序和可更改的集合。允许重复的成员。
- 元组（Tuple）是一种有序且不可更改的集合。允许重复的成员。
- 集合（Set）是一个无序和无索引的集合。没有重复的成员。
- 词典（Dictionary）是一个无序，可变和有索引的集合。没有重复的成员。

选择集合类型时，了解该类型的属性很有用。

为特定数据集选择正确的类型可能意味着保留含义，并且可能意味着提高效率或安全性。

列表

列表是一个有序且可更改的集合。在 Python 中，列表用方括号编写。

实例

创建列表：

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

访问项目

您可以通过引用索引号来访问列表项：

实例

打印列表的第二项：

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

负的索引

负索引表示从末尾开始，-1 表示最后一个项目，-2 表示倒数第二个项目，依此类推。

实例

打印列表的最后一项：

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

Cherry

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

负的索引

实例

返回第三、第四、第五项:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
Print(thislist[2:5])
```

注释：搜索将从索引 2（包括）开始，到索引 5（不包括）结束。

请记住，第一项的索引为 0。

负索引的范围

如果要从列表末尾开始搜索，请指定负索引:

实例

此例将返回从索引 -4（包括）到索引 -1（排除）的项目:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

更改项目值

如需更改特定项目的值，请引用索引号:

实例

更改第二项:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "mango"
print(thislist)
```

遍历列表

您可以使用 for 循环遍历列表项:

实例

逐个打印列表中的所有项目:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

检查项目是否存在

如需确定列表中是否存在指定的项，请使用 in 关键字：

实例

检查列表中是否存在 “apple”：

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

```
Yes, 'apple' is in the fruits list
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

列表长度

如需确定列表中有多少项，请使用 len() 方法：

实例

打印列表中的项目数：

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

```
3
```

添加项目

如需将项目添加到列表的末尾，请使用 append() 方法：

实例

使用 append() 方法追加项目：

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist
```

```
['apple', 'banana', 'cherry', 'orange'])
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

要在指定的索引处添加项目，请使用 insert() 方法：

实例

插入项目作为第二个位置：

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

```
['apple', 'orange', 'banana', 'cherry']
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

删除项目

有几种方法可以从列表中删除项目：

实例

remove() 方法删除指定的项目：

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
['apple', 'cherry']
```

实例

pop() 方法删除指定的索引（如果未指定索引，则删除最后一项）：

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

```
['apple', 'banana']
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

实例

del 关键字删除指定的索引:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
['banana', 'cherry’]
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

实例

del 关键字也能完整地删除列表:

```
thislist = ["apple", "banana", "cherry"]
Del thislist
```

实例

clear() 方法清空列表:

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

```
[]
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

复制列表

您只能通过键入 list2 = list1 来复制列表，因为：list2 将只是对 list1 的引用，list1 中所做的更改也将自动在 list2 中进行。

有一些方法可以进行复制，一种方法是使用内置的 List 方法 copy()。

实例

使用 copy() 方法来复制列表：

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
Print(mylist)
```

制作副本的另一种方法是使用内建的方法 list()。

实例

使用 list() 方法复制列表：

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

练习：

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
['apple', 'banana', 'cherry']
mylist[0]="sdfg"
print(mylist)
['sdfg', 'banana', 'cherry']
print(thislist)
['apple', 'banana', 'cherry']
new_list = list(thislist)
print(new_list)
['apple', 'banana', 'cherry']
thislist[0]="12345"
print(new_list)
['apple', 'banana', 'cherry']
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

练习

```
import copy
class Foo(object):
    def __init__(self, val):
        self.val = val

    def __repr__(self):
        return f'Foo({self.val!r})'
foo = Foo(1)
a = ['foo', foo]
b = a.copy()
c = a[:]
d = list(a)
e = copy.copy(a)
f = copy.deepcopy(a)
# edit orignal list and instance
a.append('baz')
foo.val = 5
print(f'original: {a}\nlist.copy(): {b}\nslice: {c}\nlist(): {d}\ncopy: {e}\ndeepcopy: {f}')
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

实例

合并两个列表：

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3)
```

```
['a', 'b', 'c', 1, 2, 3]
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

实例

把 list2 追加到 list1 中:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]
for x in list2:
  list1.append(x)
print(list1)
```

```
['a', 'b', 'c', 1, 2, 3]
```

实例

使用 extend() 方法将 list2 添加到 list1 的末尾:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]
list1.extend(list2)
print(list1)
```

```
['a', 'b', 'c', 1, 2, 3]
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

list() 构造函数

也可以使用 list() 构造函数创建一个新列表。

实例

使用 list() 构造函数创建列表：

```
thislist = list(("apple", "banana", "cherry")) # 请注意双括号
print(thislist)
```

```
['apple', 'banana', 'cherry']
```

第二章 第四节 Python 列表、元组、集合、字典

Python 列表

list() 构造函数

列表方法

Python 有一组可以在列表上使用的内建方法。

方法	描述
append()	在列表的末尾添加一个元素
clear()	删除列表中的所有元素
copy()	返回列表的副本
count()	返回具有指定值的元素数量。
extend()	将列表元素（或任何可迭代的元素）添加到当前列表的末尾
index()	返回具有指定值的第一个元素的索引
insert()	在指定位置添加元素
pop()	删除指定位置的元素
remove()	删除具有指定值的项目
reverse()	颠倒列表的顺序
sort()	对列表进行排序

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

元组（Tuple）

元组是有序且不可更改的集合。在 Python 中，元组是用圆括号编写的。

实例

创建元组：

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

访问元组项目

您可以通过引用方括号内的索引号来访问元组项目：

实例

打印元组中的第二个项目：

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

```
Banana
```

负索引

负索引表示从末尾开始，-1 表示最后一个项目，-2 表示倒数第二个项目，依此类推。

实例

打印元组的最后一个项目：

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

```
cherry
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

索引范围

您可以通过指定范围的起点和终点来指定索引范围。

指定范围后，返回值将是带有指定项目的新元组。

实例

返回第三、第四、第五个项目：

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
```

```
('cherry', 'orange', 'kiwi')
```

负索引范围

如果要从元组的末尾开始搜索，请指定负索引：

实例

此例将返回从索引 -4（包括）到索引 -1（排除）的项目：

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])
```

```
('orange', 'kiwi', 'melon')
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

更改元组值

创建元组后，您将无法更改其值。元组是不可变的，或者也称为恒定的。

但是有一种解决方法。您可以将元组转换为列表，更改列表，然后将列表转换回元组。

实例

把元组转换为列表即可进行更改:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
```

```
('apple', 'kiwi', 'cherry')
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

遍历元组

您可以使用 for 循环遍历元组项目。

实例

遍历项目并打印值:

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

```
apple
banana
cherry
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

检查项目是否存在

要确定元组中是否存在指定的项，请使用 in 关键字:

实例

检查元组中是否存在 "apple"：

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

```
Yes, 'apple' is in the fruits tuple
```

元组长度

要确定元组有多少项，请使用 len() 方法:

实例

打印元组中的项目数量:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

```
3
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

添加项目

元组一旦创建，您就无法向其添加项目。元组是不可改变的。

实例

您无法向元组添加项目：

```
thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange" # 会引发错误
print(thistuple)
```

```
 thistuple[3] = "orange" # This will raise an error
TypeError: 'tuple' object does not support item assignment
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

创建有一个项目的元组

如需创建仅包含一个项目的元组，您必须在该项目后添加一个逗号，否则 Python 无法将变量识别为元组。

实例

单项元组，别忘了逗号：

```
thistuple = ("apple",)
print(type(thistuple))
```

```
<class 'tuple'>
```

```
#不是元组
thistuple = ("apple")
print(type(thistuple))
```

```
<class 'str'>
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

删除项目

注释：您无法删除元组中的项目。

元组是不可更改的，因此您无法从中删除项目，但您可以完全删除元组：

实例

del 关键字可以完全删除元组：

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) # 这会引发错误，因为元组已不存在。
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

合并两个元组

如需连接两个或多个元组，您可以使用 + 运算符:

实例

合并这个元组:

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
```

```
('a', 'b', 'c', 1, 2, 3)
```

第二章 第四节 Python 列表、元组、集合、字典

Python 元组

tuple() 构造函数

也可以使用 tuple() 构造函数来创建元组。

实例

使用 tuple() 方法来创建元组：

```
thistuple = tuple(("apple", "banana", "cherry")) # 请注意双括号
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

Python 元组

元组方法

Python 提供两个可以在元组上使用的内建方法。

方法	描述
count()	返回元组中指定值出现的次数。
index()	在元组中搜索指定的值并返回它被找到的位置。

第二章 第四节 Python 列表、元组、集合、字典

Python集合

集合（Set）

集合是无序和无索引的集合。在 Python 中，集合用花括号编写。

实例

创建集合:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

```
{'cherry', 'banana', 'apple’}
```

注释：集合是无序的，因此您无法确定项目的显示顺序。

第二章 第四节 Python 列表、元组、集合、字典

Python集合

您无法通过引用索引来访问 set 中的项目，因为 set 是无序的，项目没有索引。

但是您可以使用 for 循环遍历 set 项目，或者使用 in 关键字查询集合中是否存在指定值。

实例

遍历集合，并打印值:

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
```

```
cherry
banana
apple
```

第二章 第四节 Python 列表、元组、集合、字典

Python 集合

实例

检查 set 中是否存在 “banana”:

```
thisset = {"apple", "banana", "cherry"}
print("banana" in thisset)
```

```
True
```

更改项目

集合

一旦创建，您就无法更改项目，但是您可以添加新项目。

第二章 第四节 Python 列表、元组、集合、字典

Python 集合

添加项目

要将一个项添加到集合，请使用 add() 方法。

要向集合中添加多个项目，请使用 update() 方法。

实例

使用 add() 方法向 set 添加项目：

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```

```
{'banana', 'apple', 'cherry', 'orange'}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 集合

实例

使用 update() 方法将多个项添加到集合中:

```
thisset = {"apple", "banana", "cherry"}
thisset.update(["orange", "mango", "grapes"])
print(thisset)
```

```
{'banana', 'cherry', 'mango', 'orange', 'grapes', 'apple’}
```

获取 Set 的长度

要确定集合中有多少项，请使用 len() 方法。

实例

获取集合中的项目数:

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

```
3
```

第二章 第四节 Python 列表、元组、集合、字典

Python 集合

删除项目

要删除集合中的项目，请使用 remove() 或 discard() 方法。

实例

使用 remove() 方法来删除 “banana”:

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```

```
{'cherry', 'apple’}
```

注释：如果要删除的项目不存在，则 remove() 将引发错误。

实例

使用 discard() 方法来删除 “banana”：

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)
```

```
{'apple', 'cherry’}
```

注释：如果要删除的项目不存在，则 discard() 不会引发错误。

Python 集合

您还可以使用 pop() 方法删除项目，但此方法将删除最后一项。请记住，set 是无序的，因此您不会知道被删除的是什么项目。

pop() 方法的返回值是被删除的项目。

实例

使用 pop() 方法删除最后一项：

```
thisset = {"apple", "banana", "cherry"}
x = thisset.pop()
print(x)
print(thisset)
```

```
banana
{'cherry', 'apple’}
```

注释：集合是无序的，因此在使用 pop() 方法时，您不会知道删除的是哪个项目。

实例

clear() 方法清空集合：

```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)
```

```
set()
```

Python 集合

实例

del 彻底删除集合:

```
thisset = {"apple", "banana", "cherry"}
del thisset
print(thisset)
```

```
Traceback (most recent call last):
  File "demo_set_del.py", line 5, in <module>
    print(thisset) #this will raise an error because the set no longer exists
NameError: name 'thisset' is not defined
```

第二章 第四节 Python 列表、元组、集合、字典

Python 集合

合并两个集合

在 Python 中，有几种方法可以连接两个或多个集合。

您可以使用 union() 方法返回包含两个集合中所有项目的新集合，也可以使用 update() 方法将一个集合中的所有项目插入另一个集合中：

实例

union() 方法返回一个新集合，其中包含两个集合中的所有项目：

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

```
{1, 2, 3, 'b', 'c', 'a'}
```

Python 集合

实例

update() 方法将 set2 中的项目插入 set1 中：

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
```

```
{1, 2, 'c', 'a', 3, 'b’}
```

注释：union() 和 update() 都将排除任何重复项。

还有其他方法将两个集合连接起来，并且仅保留重复项，或者永远不保留重复项，请查看此页面底部的集合方法完整列表。

第二章 第四节 Python 列表、元组、集合、字典

Python 集合

set() 构造函数

也可以使用 set() 构造函数来创建集合。

实例

使用 set() 构造函数来创建集合:

```
thisset = set(("apple", "banana", "cherry")) # 请留意这个双括号
print(thisset)
```

```
{'banana', 'cherry', 'apple'}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 集合

Set 方法

Python 拥有一套能够在集合（set）上使用的内建方法。

方法	描述
add()	向集合添加元素。
clear()	删除集合中的所有元素。
copy()	返回集合的副本。
difference()	返回包含两个或更多集合之间差异的集合。
difference_update()	删除此集合中也包含在另一个指定集合中的项目。
discard()	删除指定项目。
intersection()	返回为两个其他集合的交集的集合。
intersection_update()	删除此集合中不存在于其他指定集合中的项目。
isdisjoint()	返回两个集合是否有交集。
issubset()	返回另一个集合是否包含此集合。
issuperset()	返回此集合是否包含另一个集合。
pop()	从集合中删除一个元素。
remove()	删除指定元素。
symmetric_difference()	返回具有两组集合的对称差集的集合。
symmetric_difference_update()	插入此集合和另一个集合的对称差集。
union()	返回包含集合并集的集合。
update()	用此集合和其他集合的并集来更新集合。

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

字典（Dictionary）

字典是一个无序、可变和有索引的集合。在 Python 中，字典用花括号编写，拥有键和值。

实例

创建并打印字典：

```
thisdict =    {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
print(thisdict)
```

```
'brand': 'Porsche', 'model': '911', 'year': 1963}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

访问项目

您可以通过在方括号内引用其键名来访问字典的项目：

实例

获取 "model" 键的值：

```
x = thisdict["model"]
```

```
911
```

实例

获取 "model" 键的值：

```
x = thisdict.get("model")
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

更改值

您可以通过引用其键名来更改特定项的值:

实例

把 "year" 改为 2019:

```
thisdict =    {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
thisdict["year"] = 2019

print(thisdict)
```

```
{'brand': 'Porsche', 'model': '911', 'year': 2019}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

遍历字典

您可以使用 for 循环遍历字典。

循环遍历字典时，返回值是字典的键，但也有返回值的方法。

实例

逐个打印字典中的所有键名：

```
for x in thisdict:
  print(x)
```

```
brand
model
year
```

Python 字典

实例

逐个打印字典中的所有值：

```
for x in thisdict:
  print(thisdict[x])
```

```
Porsche
911
1963
```

实例

您还可以使用 values() 函数返回字典的值：

```
for x in thisdict.values():
  print(x)
```

```
Porsche
911
1963
```

Python 字典

实例

通过使用 items() 函数遍历键和值：

```
for x, y in thisdict.items():
  print(x, y)
```

```
brand Porsche
model 911
year 1963
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

检查键是否存在

要确定字典中是否存在指定的键，请使用 in 关键字:

实例

检查字典中是否存在 "model":

```
thisdict =    {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

```
Yes, 'model' is one of the keys in the thisdict dictionary
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

字典长度

要确定字典有多少项目（键值对），请使用 len() 方法。

实例

打印字典中的项目数:

```
print(len(thisdict))
```

```
3
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

添加项目

通过使用新的索引键并为其赋值，可以将项目添加到字典中：

实例

```
thisdict =    {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
thisdict["color"] = "red"
print(thisdict)
```

```
{'brand': 'Porsche', 'model': '911', 'year': 1963, 'color': 'red’}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

删除项目

有几种方法可以从字典中删除项目:

实例

pop() 方法删除具有指定键名的项:

```
thisdict =    {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
thisdict.pop("model")
print(thisdict)
```

```
{'brand': 'Porsche', 'year': 1963}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

实例

del 关键字也可以完全删除字典：

```
thisdict =      {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
del thisdict
print(thisdict) #this 会导致错误，因为 "thisdict" 不再存在。
```

实例

clear() 关键字清空字典：

```
thisdict =      {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
thisdict.clear()
print(thisdict)
```

```
{}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

复制字典

您不能通过键入 dict2 = dict1 来复制字典，因为：dict2 只是对 dict1 的引用，而 dict1 中的更改也将自动在 dict2 中进行。

有一些方法可以进行复制，一种方法是使用内建的字典方法 copy()。

实例

使用 copy() 方法来复制字典：

```
thisdict =    {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
mydict = thisdict.copy()
print(mydict)
```

```
{'brand': 'Porsche', 'model': '911', 'year': 1963}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

复制字典

制作副本的另一种方法是使用内建方法 dict()。

实例

使用 dict() 方法创建字典的副本:

```
thisdict =     {
  "brand": "Porsche",
  "model": "911",
  "year": 1963
}
mydict = dict(thisdict)
print(mydict)
```

```
{'brand': 'Porsche', 'model': '911', 'year': 1963}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

嵌套字典

词典也可以包含许多词典，这被称为嵌套词典。

实例

创建包含三个字典的字典:

```
myfamily = {
  "child1" : {
    "name" : "Phoebe Adele",
    "year" : 2002
  },
  "child2" : {
    "name" : "Jennifer Katharine",
    "year" : 1996
  },
  "child3" : {
    "name" : "Rory John",
    "year" : 1999
  }
}
print(myfamily)
```

```
{'child1': {'name': 'Phoebe Adele', 'year': 2002}, 'child2': {'name': 'Jennifer Katharin
e', 'year': 1996}, 'child3': {'name': 'Rory John', 'year': 1999}}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

或者，如果您想嵌套三个已经作为字典存在的字典：

实例

创建三个字典，然后创建一个包含其他三个字典的字典：

```
child1 = {
  "name" : "Phoebe Adele",
  "year" : 2002
}
child2 = {
  "name" : "Jennifer Katharine",
  "year" : 1996
}
child3 = {
  "name" : "Rory John",
  "year" : 1999
}
myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
print(myfamily)
```

```
{'child1': {'name': 'Phoebe Adele', 'year': 2002}, 'child2': {'name': 'Jennifer Katharine', 'year': 1996}, 'child3': {'name': 'Rory J
ohn', 'year': 1999}}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

dict() 构造函数

也可以使用 dict() 构造函数创建新的字典：

实例

```
thisdict = dict(brand="Porsche", model="911", year=1963)
# 请注意，关键字不是字符串字面量
# 请注意，使用了等号而不是冒号来赋值
print(thisdict)
```

```
{'brand': 'Porsche', 'model': '911', 'year': 1963}
```

第二章 第四节 Python 列表、元组、集合、字典

Python 字典

字典方法

Python 提供一组可以在字典上使用的内建方法。

方法	描述
clear()	删除字典中的所有元素
copy()	返回字典的副本
fromkeys()	返回拥有指定键和值的字典
get()	返回指定键的值
items()	返回包含每个键值对的元组的列表
keys()	返回包含字典键的列表
pop()	删除拥有指定键的元素
popitem()	删除最后插入的键值对
setdefault()	返回指定键的值。如果该键不存在，则插入具有指定值的键。
update()	使用指定的键值对字典进行更新
values()	返回字典中所有值的列表