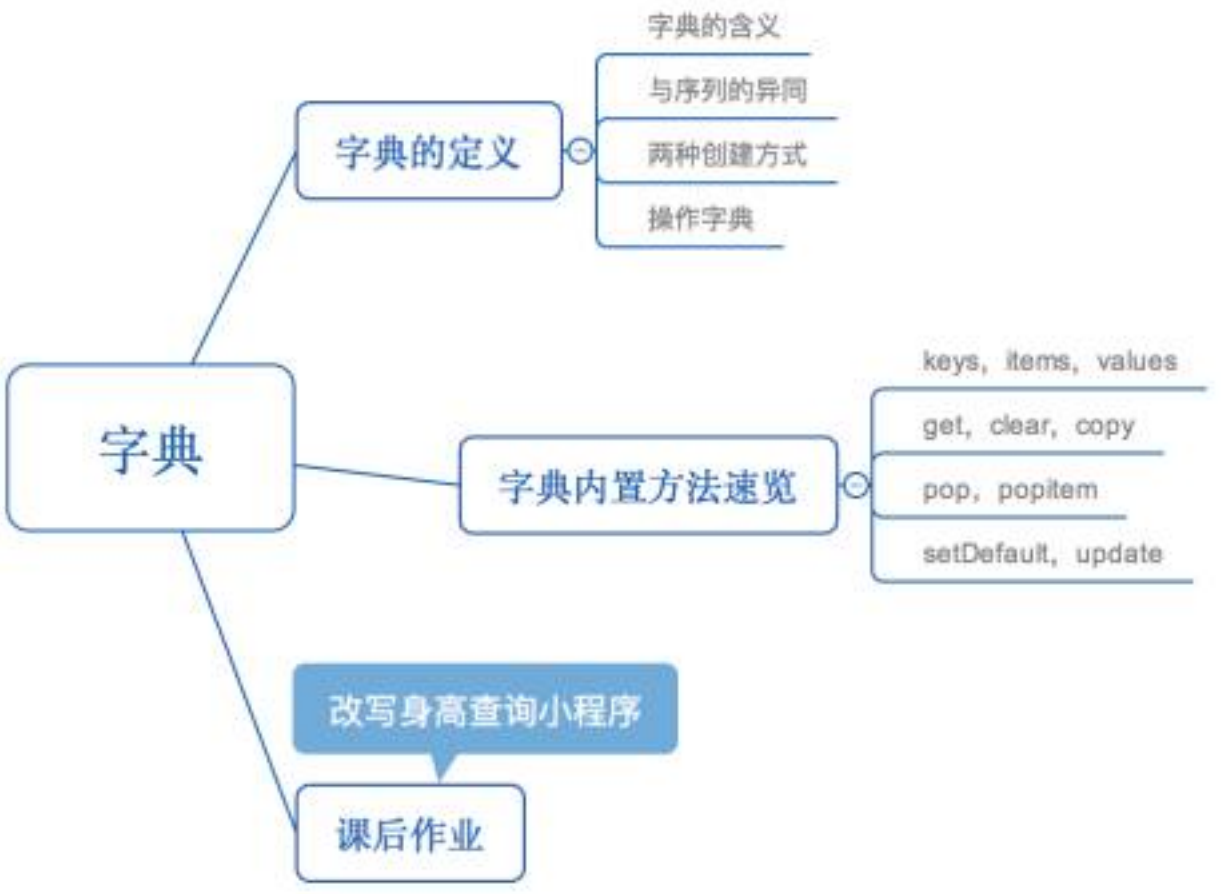
Python字典－购物车说：好伤心啊，就自己，什么也没有了。

Original 2016-11-13 哈哈小 菜猿观世界

通过本节的学习，你将了解以下内容:



看过哈哈小之前课程的伙伴应该还记得之前的一道课后题，是一道查菜猿成员身高的小程序，输入对应的名字，程序将对应的身高显示出来。

记得当时这么一个简单的程序是借助序列，分支与循环等知识点共同完成的。

和和总总大概有十几行代码，那么今天哈哈小告诉大家，通过今天内容的补充，这小程序的实现将缩减到一行。python的“魔性”在今天将又会得到一定的释放，哈哈哈。

所以，今天的主题是：当序列不好使时， 那就想到字典吧?

python诞生之初，其目标就是不断向人类语言迈进，通过这名字“字典”，想必不用多讲，大家也是耳朵熟了，眼睛详了。

没错，python中的字典和我们平时查的字典的概念非常相近。

---

1 两种创建字典的方式

最近看了一篇外文，里面有这么些生词让哈哈小颇感生疏。现在哈哈小要创建一个查找这些单词对应中文意思的字典，以达到学习英语的目的，哈哈～

如果记忆力好的或者说有勤加练习的同学，一定还记得元组，列表的创建，创建时前者上小括号，后者上中括号。那么，要创建一个字典，最简单的方式，要上什么呢?

上大括号，厉害了，我的蛔虫，这都被你猜中了，哈哈～

既然如此，废话绝对绝对绝对不可以不可以不可以多说多说多说，但是重要的话一定要多说。

```
>>> wordlookup={"tranquillity":"安宁","serendipity":"意外新发现","lullaby": "催眠曲","flabbergasted":"目瞪口呆地 "}
```

一个小字典就这样被创建出来了，有发现，字典都是A:B这样的格式成对成对地出现的，这里要明确几个概念，A:B我们把A叫做键（key），B叫做值（value），合起来就叫键值对（key－value），或者我们把A对应B的这种关系叫做映射（记得上节哈哈小也是提过这个概念吧），key和value在python中可以是任何类型的值。上面这个字典里面包含4个这样的键值对，我们就说这个字典有4个元素，或者说有4项（item）。

也可以用这种方式创建一个空字典。

```
>>> empty={}
```

与序列相比，相同点是它们都是python中的数据结构，或者你也可以把它们叫做容器，除此，任何类型的值都可以往这些容器里放。不同点是除了上面所讲，字典是以键值对的形式作为一个元素存放的，并且之所以不把字典叫做序列，是因为字典里面的元素都是无序的。所以被试着用位置值去索引元素。

通过第一种方式熟悉了字典的基本概念，第二种方式也就好理解了。

第二种创建字典的方式是使用工厂函数dict：

在使用一个函数之前，我们一贯的做法是help一下，因为没有谁天生就会怎么用。

```
>>> help(dict)
Help on class dict in module builtins:
class dict(object)
 |  dict() -> new empty dictionary
 |  dict(mapping) -> new dictionary initialized from a mapping object's
 |      (key, value) pairs
 |  dict(iterable) -> new dictionary initialized as if via:
 |      d = {}
 |      for k, v in iterable:
 |          d[k] = v
 |  dict(**kwargs) -> new dictionary initialized with the name=value pairs
 |      in the keyword argument list.  For example:  dict(one=1, two=2)
 |
```

（部分显示）

可以发现，有四种构造字典的方式，第一种是创建一个空字典，第二种方式是传入映射对象，第三种是传入一个带键值对的可迭代对象，第四种是传入的是带关键字的可变参数。

我们用例子来看看四种方式的不同

```
>>> way1=()
>>> way2=dict((("tranquillity","安宁"),("serendipity","意外新发现")))
>>> way3=dict({"tranquillity":"安宁","serendipity":"意外新发现"})
>>> way4=dict(tranquillity="安宁",serendipity="意外新发现")
```

注意：第四种方式的key值不能有双引号，这一点与关键字参数是一致的。

既然，两种创建字典的方式都明白了，这就来用用吧。

> 查

```
>>> wordlookup["tranquillity"]
'安宁'
```

中括号中放入key值，直接索引到value值，so easy～

> 改

```
>>> wordlookup["tranquillity"]="宁静"
>>> wordlookup["tranquillity"]
'宁静'
```

如果key值已存在，则更新value的内容。

> 增

```
>>> wordlookup["hilarious"]="欢闹的"
>>> wordlookup={"tranquillity":"安宁","serendipity":"意外新发现","lullaby": "催眠曲","hilarious": "欢闹的","flabbergasted":"目瞪口呆地 "}
```

如果key值不存在，不会提醒报错，而是创建一个新key-value对。也有发现，因为字典是无序的，加进去一个元素后，最后鬼知道它会处于哪个位置。

> 删

stop，这个我们要借助接下来要讲到的内置方法来实现了。

---

2 内嵌方法速览

在使用一个函数之前，我们一贯的做法是dir一下。

```
>>> dir(dict)
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__-
format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__',
'__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear',
'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
>>>
```

原来有那么多，这里哈哈小要声明的是，这一次我们来速览一下这些方法，但此做法自此打住了。因为，此做法仅希望能起到启蒙带领的作用，除此，大家还能从中感受到方法的传达，就善莫大焉了。可以说编程资料如此之多，知识点如此之多，学习过程中比起知识的收纳，更重要的是方法的探索，而这方法的拾得又来源于对有限知识的不断反思和总结。所以，后面不能一一涵盖的内容还需要大家按照正确的方法躬行亲试哦。

话锋一转，双十一刚过不久，除了焦急地在等待物件的到来同时，想必也有一些小伙伴跑到医院对医生说：我眼睛近视了，医生回他说：近视到什么程度？回答到：近视到看不到钱包里的钱了。

现在假设某h姓狂欢夜前购物车中有这么些商品，蓄势待发，等零点已过，再做处理。

```
>>> cart={"羽绒服":198,"移动硬盘":288,"黄山毛峰": 120,"笔记本": 3549}
```

（key值为商品名，value值为对应的价格）

h某在清理购物车的过程中，有如下活动：

发现零点刚过，笔记本说降100就降100。

```
>>> cart.update({"笔记本": 3449})
>>> cart
{'移动硬盘': 288, '羽绒服': 198, '黄山毛峰': 120, '笔记本': 3449}
>>>
```

果然降价不少，再逛逛看有没有别的实惠的，哎，这个投影仪1280元，中意，不管三七二十一，加到购物车。

```
>>> cart.setdefault("投影仪",1280)
```

```
1280
>>> cart
{'投影仪': 1280, '移动硬盘': 288, '羽绒服': 198, '黄山毛峰': 120, '笔记本': 3449}
>>>
```

算了，今年赚的不多，回家还要报帐，就暂时买这么些吧，确认购买之前，来个备份吧，到时好算帐。

```
>>> cart_copy=cart.copy()
>>> id(cart_copy)
4385810312
>>> id(cart)
4385137160
>>>
```

看看买的热水瓶多少钱。

```
>>> cart.get("热水瓶","你没买啊，哥们，看什么看")
'你没买啊，哥们，看什么看'
>>>
```

哦，记错了，当时商家服务态度太差，没去买。那投影仪总该买了吧。

```
>>> cart.get("投影仪","哥们，如果你没买，你就会看到这条信息")
1280
>>>
```

除了投影仪，看看我还买了啥。

```
>>> cart.keys()
dict_keys(['投影仪', '移动硬盘', '羽绒服', '黄山毛峰', '笔记本'])
>>>
```

不多也不少，主要在价格。

```
>>> cart.items()
dict_items([('投影仪', 1280), ('移动硬盘', 288), ('羽绒服', 198), ('黄山毛峰', 120), ('笔记本', 3449)])
>>>
```

好的，购物车基本信息已经了解了，先确认购买投影仪吧。

```
>>> cart.pop("投影仪")
1280
>>>cart
{'移动硬盘': 288, '羽绒服': 198, '黄山毛峰': 120, '笔记本': 3449}
>>>
```

购物车说：投影仪，拜拜～

再随便确认买个物件，看看是什么。

```
>>> cart.popitem()
('移动硬盘', 288)
>>> cart
{'羽绒服': 198, '黄山毛峰': 120, '笔记本': 3449}
>>>
```

购物车说：移动硬盘，拜拜～

一个一个地，太费事，还赶着睡觉呢，一并清空了吧。

```
>>> cart.clear()

>>> cart
{}
```

购物车说：好伤心啊，就自己，什么也没有了。

噢哦，忘记算总价格了。哦，还好备份了一份，来算一下吧。

```
>>> cart_copy.values()
dict_values([1280, 288, 198, 120, 3449])
```

```
>>> totalprice=0
>>> for eachprice in cart_copy.values():
        totalprice+=eachprice

>>> print("总计：%d元"%totalprice)
总计：5335元
>>>
```

可以看出h某在双十一这一天，总共花费：5335元

> 温馨提示：

上面出现的方法的用法可以以下方式获取：

```
>>> help(dict.clear)
Help on method_descriptor:
clear(...)
    D.clear() -> None.  Remove all items from D.
```

（None. Remove all items from D.返回None, 作用是从D字典中移除所有items。）

掌握这项技能，以后就好办了，哈哈～

---

3 课后作业

记得在文头哈哈小提到过身高查询小程序，没学字典之前，原来的代码长这样的：

通过今天字典的学习，简化这个小程序应该没有什么问题了。

那就去尝试把它改写成一行并试着去使用字典的所有内置方法吧。

结束语：我们不想给你带来多少多少的知识，我们只想尽绵薄之力给你带来可能的智慧的启迪。

持，期待您的喜欢和评论。

Read more