

python 数据结构 , 原来不难

Python 实战圈

Python 实战圈

学python，来实战圈

目录

什么是数据结构	3
什么是列表	3
列表的基本操作	3
访问列表元素	4
修改列表元素	5
添加列表元素	6
删除列表元素	7
列表排序	9
列表的高级用法	10
列表切片	10
元组	13
项目实战：《延禧攻略》之魏璎珞请客之道	14
什么是字典	15
字典的特性	16
字典的基本操作	17
访问字典	17
添加键值对	17
修改键值对	18
删除键值对	18
创建空字典	19
字典内置函数	19
字典与列表结合	21
字典列表	21
在字典中存储列表	22
在字典中存储字典	22
项目练习：《扶摇》之演员简介	23
扫码加入 Python 实战圈	24

什么是数据结构

数据结构是相互之间存在一种或多种特定关系的集合。也可以理解为，数据结构就是将数据按照某种方式组合在一起的结构。这里的数据也就是 python 中的基本数据类型数据，比如整型、浮点数、字符串等。在 Python 中，常见的内置数据结构（也就是自带的）是列表、元组、字典等，在 python 的第三方包中还有其他数据结构，比如 numpy 中的 dataframe or series。接下来的内容重点介绍 python 中的内置数据结构，但是在介绍的过程中，我们使用了很多 print()函数作为进一步的解释说明，希望大家能认真阅读一下。

什么是列表

列表是由一系列按特定顺序排列的元素组成。也就是列表是有序集合。在 Python 中，用方括号（[]）来表示列表，并用逗号来分隔其中的元素。可以给列表起一个名字，并且使用（=）把列表名字和列表关联起来，这就叫做列表赋值。具体如下：

#语法定义：

列名名字 = [元素 1, 元素 2, 元素 3,]

例子：

```
#定义一个列表
```

```
#python 实战圈成员列表
```

```
names_python_pc = ['陈升','刘德华','杨幂','TFboys']
```

```
print(fPython 实战圈的成员有： {names_python_pc})
```

输出结果为：

```
Python 实战圈的成员有： ['陈升','刘德华','杨幂','TFboys']
```

其中：列表中的元素个数是动态的，也就是可以随意添加和删除。这点是与字符串的本质区别：字符串是不能修改，列表是可变的。

列表的基本操作

在 Python 中，type()函数被用来查看变量的类型。只有知道了变量的类型才能对其进行相应的操作，因为不同的数据类型有不同的操作，比如字符串有自己独特的一系列操

作。同样，我们使用该函数查看列表在 python 中的类型表示为<class 'list'>，具体如下：

例子：

```
#查看变量的类型
```

```
print('names_python_pc 的数据类型是：',type(names_python_pc))
```

输出结果：

```
names_python_pc 的数据类型是： <class 'list'>
```

在实际项目中，变量的各种类型都会用到，当看到<class 'list'>就表示是列表，才可以对其进行列表的各种操作。列表常见的操作有访问元素、添加元素、修改元素、删除元素以及列表排序等等。这些操作中经常使用的两个术语是函数和方法，我们需要知道他们的区别。函数是独自的一个功能单元，直接可以使用，比如函数 len(列表名)求列表的元素长度；而方法是依附于对象的，调用方法是对象.方法()，比如列表名.sort()，对列表进行排序，方法是面向对象的一个重要概念。无论是方法还是函数，随着我们使用的频率增加，自然而然就记住了，不需要刻意的去背诵。

访问列表元素

列表是有序的，每一个元素都自动带有一个位置信息，也就是索引。在编程语言中，无论 python 还是其他的语言，索引都是从 0 开始，而不是 1.第 0 个索引对应的元素就是第一个元素，以此类推，比如在列表 names_python_pc 中，第 0 个索引对应的列表元素就是‘陈升’；第三个索引，也就是最后一个元素对应的是‘TFboys’。



Index ▲	Type	Size	Value
0	str	1	陈升
1	str	1	刘德华
2	str	1	杨幂
3	str	1	TFboys

访问列表元素的方法是根据索引访问，只需要指出索引号即可。

语法：

列表名[索引号]

例子：

```
#根据索引访问列表元素，并且赋值给变量 three_str  
three_str = names_python_pc[2]  
  
#直接打印 ( print ) 列表元素或根据变量打印，项目中经常用到  
print(names_python_pc[2])  
print('列表中第三个元素是：{}'.format(three_str))
```

输出结果为：

杨幂

列表中第三个元素是：杨幂

列表中最后一个元素的方法有两个：第一个；通过索引号[-1]来获取。这个特殊的语法特别有用，尤其在项目中，不知道一个 excel 文件具体有多少列，但是我们记得最后一列是想要获取的信息，此时就可以使用该方法；第二个：明确知道列表有多少列，使用最后一列的索引即可。

例子

```
#两种方法访问最后一个元素  
names_python_pc[-1]  
print('使用第一种方法，获得列表最后一个元素是{}'.format(names_python_pc[-1]))  
  
names_python_pc[3]  
print('使用第二种方法，获得列表最后一个元素是{}'.format(names_python_pc[3]))
```

结果为：

使用第一种方法，获得列表最后一个元素是 TFboys

使用第二种方法，获得列表最后一个元素是 TFboys

修改列表元素

修改列表元素与访问列表元素一样，根据索引即可修改元素的值。

语法：

列表名[index] = '新的值'

例子：

```
#修改第三个元素的值
```

```
names_python_pc[2] = '扶摇'
```

```
print('修改后的成员列表:{}'.format(names_python_pc))
```

结果为：

修改后的成员列表:['魏璎珞', '陈升', '扶摇', '杨幂', 'TFboys', '傅恒']

添加列表元素

列表是可变的。在列表中添加元素分为两种情况：

第一种：在指定位置插入一个元素，用到的方法是：

#insert 方法

```
insert(index,x)
```

index 是准备插入到其前面的那个元素的索引; x 为需要插入的元素。

例子：

```
print('原来的成员列表: {}'.format(names_python_pc))
```

```
names_python_pc.insert(0,'魏璎珞')
```

```
print('插入新的成员以后的列表: {}'.format(names_python_pc))
```

结果为：

原来的成员列表：['陈升', '刘德华', '杨幂', 'TFboys']

插入新的成员以后的列表：['魏璎珞', '陈升', '刘德华', '杨幂', 'TFboys']

第二种：在列表的末位添加元素,用到的方法是：

#append(x) x 为需要插入的元素

例子：

```
#append(x)
```

```
print('原来的成员列表: {}'.format(names_python_pc))
```

```
names_python_pc.append('傅恒')
```

```
print('插入新的成员以后的列表: {}'.format(names_python_pc))
```

结果为:

原来的成员列表: ['魏瓔珞', '陈升', '刘德华', '杨幂', 'TFboys']

插入新的成员以后的列表: ['魏瓔珞', '陈升', '刘德华', '杨幂', 'TFboys', '傅恒']

其中, 在项目开发中, 第二种方法经常被用来构建一个新的列表。首先, 创建一个空的列表, 然后在程序运行的过程中使用 `append()` 方法添加元素。

例子:

#构建新的列表

```
yan_xi_gong_luo = []
```

```
yan_xi_gong_luo.append('皇上')
```

```
yan_xi_gong_luo.append('富察皇后')
```

```
yan_xi_gong_luo.append('高贵妃')
```

```
yan_xi_gong_luo.append('纯妃')
```

```
print('使用 append()方法构建列表: {}'.format(yan_xi_gong_luo))
```

结果为:

使用 `append()` 方法构建列表: ['皇上', '富察皇后', '高贵妃', '纯妃']

删除列表元素

在项目中, 我们经常需要删除列表中的元素。python 可以根据索引值删除, 也可以根据元素值删除。

如果我们记得要删除的元素的位置, 则可以根据索引值删除, 用到的是语句 `del()` 或者方法 `pop`。语句 `del(index)` 根据索引值删除元素, 并且删除后不可以赋值给任何变量; 方法 `pop()` 删除列表尾部的元素, 或者 `pop(index)` 感觉索引值删除, 但是 `pop` 方法删除后的元素可以赋值给变量。这就是两者的最大区别。

语法:

```
del 列表名[indx]
```

```
列表名.pop() 或者 列表名.pop(index)
```

例子:

#删除列表中的魏璎珞

```
del names_python_pc[0]
```

```
print('del 语句删除列表中的魏璎珞后的列表是{}'.format(names_python_pc))
```

#POP 方法删除列表中的傅恒

```
delete_name = names_python_pc.pop()
```

```
print(f'pop 方法删除的元素是 {delete_name}')
```

#根据位置删除 扶摇

```
delete_name_index = names_python_pc.pop(1)
```

```
print(f'pop 根据索引删除的元素值是 {delete_name_index}')
```

结果为:

```
del 语句删除列表中的魏璎珞后的列表是['陈升', '扶摇', '杨幂', 'TFboys', '傅恒']
```

```
pop 方法删除的元素是 傅恒
```

```
pop 根据索引删除元素值是 扶摇
```

如何我们不记得要删除的列表元素的位置，只是记得值，可以采用的方法是 `remove()`。如果列表中有多个类似的值，则 `remove()` 方法一次只能删除一个。

语法:

```
列表名.remove('值')
```

例子:

```
print("原来的列表是:", names_python_pc)
```

```
#删除列表中的 TFboys
```

```
names_python_pc.remove('TFboys')
```

```
print(f'删除后的列表是 {names_python_pc}')
```

结果为:

```
原来的列表是: ['陈升', '杨幂', 'TFboys']
```

```
删除后的列表是 ['陈升', '杨幂']
```


列表排序

很多时候，我们需要对列表中的元素进行排序，然后进行运算。列表排序分为永久性排序和临时性排序。永久性排序是真正修改列表元素的排列顺序，用到的方法是 `sort()`，默认为升序，如果是降序，添加参数 `reverse=True`；而临时性排序是不改变原来的排列顺序，用到的函数是 `sorted()`

语法：

永久排序：列表名.`sort()`

临时性排序: `sorted(列表名)`

除了列表排序，还有很多其他重要用法，比如方法 `copy()`复制列表、方法 `len()`求列表长度、函数 `reverse()`反转列表等

例子：

"""

`copy` 方法复制一个物理对象，而非视图对象；

`count` 方法计数；

`index` 方法返回索引位置；

`reverse` 方法实现元素颠倒；

`sort` 方法排序；

"""

`L1=['a','a','b','c','d','e','f','g']`

`L2=L1.copy()`#复制 list

`print('复制列表: ',L2)`

`print('统计列表中 a 出现的次数',L1.count('a'))`#计“a”出现次数

`print('B 所在的位置',L1.index('b'))`#返回 b 的索引位置

`L1.reverse()`#颠倒元素

`print('颠倒元素的顺序:',L1)`

`L1.sort()`#默认升序排序

`print('升序排列元素',L1)`

`L1.sort(reverse=True)`#降序

`print('降序排列元素',L1)`

`print('L1 长度为:',len(L1))`#确保列表的长度

结果为:

复制列表: ['a', 'a', 'b', 'c', 'd', 'e', 'f', 'g']

统计列表中 a 出现的次数 2

B 所在的位置 2

颠倒元素的顺序: ['g', 'f', 'e', 'd', 'c', 'b', 'a', 'a']

升序排列元素 ['a', 'a', 'b', 'c', 'd', 'e', 'f', 'g']

降序排列元素 ['g', 'f', 'e', 'd', 'c', 'b', 'a', 'a']

L1 长度为: 8

列表的高级用法

列表切片

列表切片是处理列表的部分元素，也就是把整个列表切开。它是整个列表中的重点内容，在以后的 python 项目中经常使用到。另外，Python 中符合序列的有序序列都支持切片（slice），例如列表，字符串，元组。

语法：

格式：【start:end:step】

start: 起始索引，从 0 开始，-1 表示结束

end: 结束索引

step: 步长，end-start，步长为正时，从左向右取值。步长为负时，反向取值

注意切片的结果不包含结束索引，即不包含最后的一位，-1 代表列表的最后一个位置索引

例子：

```
'''
```

切片

```
'''
```

```
name_fuyao = ['扶摇','周叔','国公','无级太子','医圣','非烟殿主','穹苍']
```

```
#指定开始和结束位置，注意不包括最后的位置元素
```

```
print('扶摇电视列表中第三个到第五个人物的名字:',name_fuyao[2:5])
```

```
#不指定开始的位置，则默认从头开始
```

```
print('扶摇电视列表中前 5 个人物名字:',name_fuyao[:5])
```

```

#不指定结束的位置，则从开始位置到结束
print('扶摇电视列表从 6 个人物的名字:',name_fuyao[5:])

#开始和结束位置都不指定
print('扶摇电视列表中的名字:',name_fuyao[:])

#负数索引表示返回距离列表末位相应距离的元素，也就是取列表中后半部分的元素
print('扶摇电视列表中最后三个人物的名字:',name_fuyao[-3:])

#取偶数位置的元素
print('扶摇电视列表中偶数位置的人物是:',name_fuyao[::2])

#取奇数位置的元素
print('扶摇电视列表中偶数位置的人物是:',name_fuyao[1::2])

#逆序列表，相当于 reversed(list)
print('扶摇电视列表中人物颠倒顺序:',name_fuyao[::-1])

#在某个位置插入多个元素
#也可以用同样的方法插入或者删除多个元素
name_fuyao[3:3]=['玄机','太渊','天煞']
print('扶摇电视列表中人物变为:',name_fuyao)

#复制列表,相当于 copy () ,复制以后的新的列表是一个新的，可以对其操作
#new_name_fuyao = name_fuyao 这样的操作是变量赋值，也就是同一个值给了两个变量，一个改变了值，另外的两个跟着修改了。
new_name_fuyao = name_fuyao[:]
print('新的列表元素:{}'.format(new_name_fuyao))

```

结果为:

扶摇电视列表中第三个到第五个人物的名字: ['国公', '无级太子', '医圣']

扶摇电视列表中前 5 个人物名字: ['扶摇', '周叔', '国公', '无级太子', '医圣']

扶摇电视列表从 6 个人物的名字: ['非烟殿主', '穹苍']

扶摇电视列表中的名字: ['扶摇', '周叔', '国公', '无级太子', '医圣', '非烟殿主', '穹苍']

扶摇电视列表中最后三个人物的名字: ['医圣', '非烟殿主', '穹苍']

扶摇电视列表中偶数位置的人物是: ['扶摇', '国公', '医圣', '穹苍']

扶摇电视列表中偶数位置的人物是: ['周叔', '无级太子', '非烟殿主']

扶摇电视列表中人物颠倒顺序: ['穹苍', '非烟殿主', '医圣', '无级太子', '国公', '周叔', '扶摇']

扶摇电视列表中人物变为: ['扶摇', '周叔', '国公', '玄机', '太渊', '天煞', '无级太子', '医圣', '非烟殿主', '穹苍']

新的列表元素: ['扶摇', '周叔', '国公', '玄机', '太渊', '天煞', '无级太子', '医圣', '非烟殿主', '穹苍']

列表其他常用用法:

1. list 函数: 根据字符串创建列表, 比如吧 hello, 变成列表.

语法:

```
list('字符串')
```

例子:

```
print('把字符串 hello 变成列表结构:', list('hello'))
```

```
把字符串 hello 变成列表结构: ['h', 'e', 'l', 'l', 'o']
```

2. join() 方法用于将序列中的元素以指定的字符连接生成一个新的字符串。

语法:

```
字符串名字.join(列表名字)
```

例子:

```
linked = '-'
```

```
data_list = ['Python', 'is', 'NO.1']
```

```
print(linked.join(data_list))
```

```
print(type(linked.join(data_list)))
```

结果为:

```
Python-is-NO.1
```

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

3. index 方法: 返回元素所在的索引位置:

语法:

```
列表名.index('元素名')
```

例子：

```
L1=['a','a','b','c','d','e','f','g']
print('B 所在的位置',L1.index('b'))#反为 b 的索引位子
B 所在的位置 2
```

4.len()函数计算列表的长度

语法：

```
len(列表名)
```

例子：

```
print('L1 长度为:',len(L1))#确保列表的长度
```

元组

元组是 python 中不能被修改元素值的数据结构。元组是使用圆括号()表示，而列表是使用方括号[]。请注意两者的区别

语法：

```
元组名 = (元素 1 , 元素 2,.....)
```

例子：

```
t1=1,2,3
t2="jeffreyzhao","cnblogs"
t3=(1,2,3,4)
t4=()
t5=(1,)
print t1,t2,t3,t4,t5
```

结果为：

```
(1, 2, 3) ('jeffreyzhao', 'cnblogs') (1, 2, 3, 4) () (1,)
```

从上面我们可以分析得出：

- ◆ 逗号分隔一些值，元组自动创建完成；
- ◆ 元组大部分时候是通过圆括号括起来的；
- ◆ 空元组可以用没有包含内容的圆括号来表示；

◆ 只含一个值的元组，必须加个逗号（,）；

元组的值虽然不能被修改，但是可以给存储元组的变量赋不同的值。

例子：

```
range = (30,40,50)
print('old range is:\n')
print(range)
range = (60,70,80)
print('new range is:\n')
print(range)
```

结果为：

```
old range is:
(30, 40, 50)
new range is:
(60, 70, 80)
```



项目实战：《延禧攻略》之魏璎珞请客之道

这次的项目选择的是最近热播电视剧《延禧攻略》，根据里面的人物创作而成。本项目没有具体的答案，自己根据理解写代码即可，用到的知识点都是前面的内容。

项目描述：

春节到临之际，魏璎珞计划宴请所有妃嫔吃饭，包括，太后、皇后、纯妃、小嘉嫔、舒妃、以及皇上，并且唱了一出鸿门宴。请创建一个列表存储所有妃嫔的列表，然后打出每一个人的名字，并且告诉大家“春节将至，请大家过来延禧宫小聚。”。但是小嘉嫔得知后，由于在争宠失败不想参加她的宴会，就让宫女拒绝了，请打印出谁不能参加此次宴会，于是魏璎珞想请尔晴参加，请重新修改列表，打印出邀请的名单。

皇上收到邀请后，感觉魏璎珞的点子特别好，于是特许她在御花园宴请大家。于是魏璎珞可以邀请更多的人了，请使用 insert 方法把‘哥哥’放在邀请名单的开头；由于傅恒是自己的小情人，所以请用 append 方法把‘傅恒’放着名单最后。请重新打印所有人的名单，并且使用 len 方法打印出，一共邀请了多少人，并且复制一个新的列表备份。明玉

看到最后的邀请名单后，先是打印了前三个名字，然后又打印查看了后三个人的名字，最后感觉顺序不对。于是她颠倒了一下顺序，看着舒服多了。

马上到了宴会开始的时候，皇上得知傅恒和魏璎珞的关系，特别生气。于是收回了魏璎珞的御花园宴请宾客的命令。魏璎珞不得不把宴请重新搬回到延禧宫，并且为了避险，只能宴请两位妃嫔：皇后和尔晴，请用 `pop` 方法把多余的名单删除，并且告诉他们特别遗憾不能邀请大家吃饭。然后告诉皇后和尔晴依然在受邀之列。

宴会开始之后，请使用 `del` 语句删除邀请名单。

项目要求：

1. 代码要有注释，
2. 变量名字要清楚描述代表的内容，不能使用 `a b c` 等
3. 提交代码到知识星球。

提示：

第一步：先不要着急写代码，首先把邀请名单列出来，

第二步：然后记录每一步的变化。

第三步：最后联想上面学到的知识点。

什么是字典

字典是另外一个可变的数据结构，且可存储任意类型对象，比如字符串、数字、列表等。字典是由关键字和值两部分组成，也就是 `key` 和 `value`，中间用冒号分隔。这种结构类似于新华字典，字典中每一个字都有一个对应的解释，具体语法如下：

语法：

字典名 = { 关键字 1: 值, 关键字 2: 值, 关键字 3: 值 }

注意： 每个键与值用冒号隔开 (`:`)，每对用逗号分割，整体放在花括号中 (`{}`)。

键必须独一无二，但值则不必。

可以有任意多个键值对

值可以取任何数据类型，但键必须是不可变的，如字符串，数或元组。

例子：

#构建一个字典，记录各宫嫔妃的年薪银子

```
name_dictionary = {'魏璎珞':300,'皇后':1000,'皇贵妃':800,'贵妃':600,'斌':200}
```

```
print(name_dictionary)
```

```
print('字典的数据类型表示是:', type(name_dictionary))
```

结果是：

```
{'魏璎珞': 300, '皇后': 1000, '皇贵妃': 800, '贵妃': 600, '斌': 200}
```

字典的数据类型表示是: <class 'dict'>

字典的特性

字典值可以没有限制地取任何 python 对象，既可以是标准的对象，也可以是用户定义的，但键不行。两个重要的点需要记住：

- 1) 不允许同一个键出现两次。创建时如果同一个键被赋值两次，后一个值会被记住

例子：

```
#定义两个同样的关键字 Name
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'};
print ("dict['Name']: ", dict['Name'])
```

结果为：

```
#dict['Name']: Manni
```

- 2) 键必须不可变，所以可以用数，字符串或元组充当，所以用列表就不行，如下实例：

```
#关键字 Name 为列表
dict = {'Name': 'Zara', 'Age': 7};
print ("dict['Name']: ", dict['Name'])
```

结果为：

Traceback (most recent call last):

```
File "/Applications/PyCharm.app/Contents/helpers/pydev/pydev_run_in_console.py", line 52, in run_file
```

```
pydev_imports.execfile(file, globals, locals) # execute the script
```

```
File "/Applications/PyCharm.app/Contents/helpers/pydev/_pydev_imps/_pydev_execfile.py", line 18, in execfile
```

```
exec(compile(contents+"\n", file, 'exec'), glob, loc)
```

```
File "/Users/yoni.ma/PycharmProjects/seven_days_python/Forth_day_strcure/dict.py", line 47, in <module>
```



```
dict = {'Name': 'Zara', 'Age': 7};  
TypeError: unhashable type: 'list'
```

字典的基本操作

字典在 python 中的类型表示是 `<class 'dict'>`。当查看到变量的类型是 `dict`,则可以对其进行字典的操作。常见的字典操作是访问字典、遍历字典等。这些操作在实际项目中经常被使用到,比如 excel 文件读入内存以后,按照字典的方法存放。然后对其增加或删除值。

访问字典

访问字典也就是获取关键字对应的值,方法是指定字典名和放在方括号内的关键字,具体如下。获取后的值可以赋值给变量。

语法:

变量名 = 字典名[关键字]

例子:

```
#访问字典  
weiyingluo = name_dictionary['魏璎珞']  
print(f'魏璎珞的年薪是: {weiyingluo}两')
```

结果为:

魏璎珞的年薪是: 300 两

添加键值对

字典是一种可变的数据结构,可以随时添加或者删除其中的键值对。其中添加键值的方法是,指定字典名、用方括号括起的键和相关的值,具体如下。

语法:

字典名[关键字名] = 值

例子:

```
#增加贵人和常在的年薪银子  
print(f'原来的后宫年薪字典是:{name_dictionary}')  
name_dictionary['贵人'] = 100  
name_dictionary['常在'] = 50  
print(f'增加键值后的后宫年薪字典变成: {name_dictionary}')
```

结果为：

原来的后宫年薪字典是：{'魏璎珞': 300, '皇后': 1000, '皇贵妃': 800, '贵妃': 600, '斌': 200}

增加键值后的后宫年薪字典变成：{'魏璎珞': 300, '皇后': 1000, '皇贵妃': 800, '贵妃': 600, '斌': 200, '贵人': 100, '常在': 50}

修改键值对

如果字典中的值不是我们想要的，可以通过修改的方法达到。以此指定字典名、用方括号括起的键以及与该键相对应的新值。

语法：

字典名[关键字名] = 新的值

例子：

```
#修改字典的值，比如修改常在的年薪为 70 两
print('常在原来的年薪是 {} 两'.format(name_dictionary['常在']))
name_dictionary['常在'] = 70
change_changzai = name_dictionary['常在']
print(f'常在修改后的年薪是 {change_changzai} 两')
```

结果为：

常在原来的年薪是 50 两

常在修改后的年薪是 70 两

删除键值对

如果字典中的键值对不在需要，我们可以彻底删除。python 使用的是 del 语句，必须要指定要删除的字典名和关键字。注意是永久删除

语法：

del 字典名[关键字]

例子：

```
#删除字典中的键值对，比如删除常在
del name_dictionary['常在']
print(f'删除常在后的后宫嫔妃年薪字典变成: {name_dictionary}')
```

结果为：

删除常在后的后宫嫔妃年薪字典变成: {'魏璎珞': 300, '皇后': 1000, '皇贵妃': 800, '贵妃': 600, '斌': 200, '贵人': 100}

创建空字典

在实际项目中，我们可能不知道字典中存放的内容是什么。这时，我们可以采用从空的字典开始动态创建，也就是在程序运行的时候添加具体的内容。

常见的使用场景是：第一个：需要用户输入数据存储为字典；第二个是自动生成大量的键值对，比如爬虫，爬取豆瓣电影的排名信息。我们可以把排名放入空的字典中，然后每次爬取一个电影，添加一个对应的键值对。

例子：

```
#从空的字典开始创建
douban_movies = {} #定义空的字典
douban_movies['排名'] = 1
douban_movies['片名'] = '霸王别姬'
douban_movies['主演'] = '张国荣、张丰毅、巩俐'
douban_movies['导演'] = '陈凯歌'
print('从空的列表中构建字典:',douban_movies)
```

结果为：

从空的列表中构建字典: {'排名': 1, '片名': '霸王别姬', '主演': '张国荣、张丰毅、巩俐', '导演': '陈凯歌'}

字典内置函数

Python 字典包含了以下内置函数和方法。我们就不在给大家一一举例子了，自己可以试一下。这些函数或者方法不需要死记硬背，用到的时候去查就看了。

比如

内置函数

cmp(dict1, dict2) #比较两个字典元素。

len(dict) #计算字典元素个数，即键的总数。

str(dict) #输出字典可打印的字符串表示。

内置方法

字典名.clear() #删除字典内所有元素

字典名.copy() #返回一个字典的浅复制

字典名.fromkeys() #创建一个新字典，以序列 seq 中元素做字典的键，val 为字典所有键对应的初始值

字典名.get(key, default=None) #返回指定键的值，如果值不在字典中返回 default 值

字典名.has_key(key) #如果键在字典 dict 里返回 true，否则返回 false

字典名.items() #以列表返回可遍历的(键, 值) 元组数组

字典名.keys() #以列表返回一个字典所有的键

字典名.setdefault(key, default=None) #和 get()类似，但如果键不已经存在于字典中，将会添加键并将值设为 default

字典名.update(dict2) #把字典 dict2 的键/值对更新到 dict 里

字典名.values() #以列表返回字典中的所有值

例子：

#内置函数和方法

```
print('计算字典的个数: ',len(name_dictionary))
```

```
print('输出字典可以打印的字符串',str(name_dictionary))
```

#内置函数

```
print('返回指定的贵妃的年薪',name_dictionary.get('贵妃'))
```

```
print('以列表的形式返回字典中的所有关键字',name_dictionary.keys()) #经常被用到
```

```
print('以元组形式返回所有的键值对',name_dictionary.items()) #经常被用到
```

```
print('返回键值中的所有值',name_dictionary.values())#经常被用到
```

结果为：

计算字典的个数： 6

输出字典可以打印的字符串 {'魏璎珞': 300, '皇后': 1000, '皇贵妃': 800, '贵妃'

```
' ': 600, '斌': 200, '贵人': 100}
```

返回指定的贵妃的年薪 600

以列表的形式返回字典中的所有关键字, `dict_keys(['魏璎珞', '皇后', '皇贵妃', '贵妃', '斌', '贵人'])`

以元组形式返回所有的键值对 `dict_items([('魏璎珞', 300), ('皇后', 1000), ('皇贵妃', 800), ('贵妃', 600), ('斌', 200), ('贵人', 100)])`

返回键值中的所有值 `dict_values([300, 1000, 800, 600, 200, 100])`

字典与列表结合

字典和列表是 python 中经常用到的两个数据结构，并且都是可变的。有时候，我们需要把两者结合起来使用。把一系列字典存储在列表中，或将列表作为值放在字典中，这称为嵌套。你可以在列表中嵌套字典、在字典中嵌套列表甚至在字典中嵌套字典。这在项目中经常用到。

什么时候用列表什么时候用字典呢？面对这个问题我的想法是，当你存取的数据类型都是一样的时候，使用列表，当你存取的数据类型不一样时就用字典。这里说明一下数据类型不一样不是指整形或者字符型。举个例子：如果你需要存很多人的姓名，仅仅这一个属性，就用列表来进行处理，当你要存取不仅仅是人名，包括年龄，性别，国籍等等这些信息时，这时候用字典是最合适的。

字典列表

列表中的元素都是字典为字典列表。一般用在列表的元素信息比较复杂，单一的字符串不能满足。

例子：

```
#两个列表合并为一个字典
la = {'name':'charles','age':18}
lb = {'name':'mol', 'age':'unknown'}

name = [la,lb]

print(name)
```

结果为：

```
[{'name': 'charles', 'age': 18}, {'name': 'mol', 'age': 'unknown'}]
```

在字典中存储列表

字典中的值有时候不是一个，而是多个。这时需要把字典中的值变成一个列表，而不是单个的数字。

例子：

#字典中存储列表

```
favorite_actor = {  
    '魏璎珞': ['傅恒', '皇上', '富察皇后'],  
    '皇上': ['魏璎珞', '富察皇后', '纯妃', '高贵妃'],  
    '高贵妃': '皇上'  
}  
  
print(favorite_actor)
```

结果为：

```
{'魏璎珞': ['傅恒', '皇上', '富察皇后'], '皇上': ['魏璎珞', '富察皇后', '纯妃', '高贵妃'],  
 '高贵妃': '皇上'}
```

在字典中存储字典

字典的值也可以是字典，称为字典中存储字典。一般用在键对应的值是二维的信息，比如登录某一个网站的用户信息，用户名是键，用户名对应的值比较部分，既包括用户的地址、职业、收入等信息。

例子：

#字典中存储字典

```
users = {  
    '爱上不该爱的人': {  
        '姓名': '魏璎珞',  
        '职位': '妃子',  
        '年薪': '300 两'},  
    '只爱皇上': {  
        '姓名': '高贵妃',
```

```

        '职位': '贵妃',
        '年薪': '800 两',
    }

print(users)

```

结果为:

```

{'爱上不该爱的人': {'姓名': '魏璎珞', '职位': '妃子', '年薪': '300 两'}, '只爱皇上': {'姓名': '高贵妃', '职位': '贵妃', '年薪': '800 两'}}

```

项目练习:《扶摇》之演员简介

根据下面的图片, 构建一个字典, 叫 `Fuyao_Actor_Profile`. 包括演员名字, 饰演角色, 配音演员。然后打印出杨幂扮演的角色是谁?。创建一个备份字典 `Copy_Fuyao`, 防止后面有所变化。

假如由于阮经天有事情不能参加本次拍摄, 请在演员表中去除他的信息。然后更替为陈晓。并且增加新的角色, 如第二张图所示



杨幂 饰 扶摇
配音 王潇倩



阮经天 饰 长孙无极
简介 天权国太子
配音 马正阳



刘奕君 饰 齐震
简介 太渊国国公
配音 刘奕君



高伟光 饰 战北野
简介 天煞烈王
配音 赵成晨



王劲松 饰 长孙迺
简介 天权国皇帝
配音 王劲松



黄宥明 饰 燕惊尘
简介 玄元剑派大师兄
配音 文森



高瀚宇 饰 江枫
简介 长孙无极的贴身密卫
配音 袁聪宇



顾又铭 饰 战北恒
简介 天煞国恒王
配音 林强



秦焰 饰 周叔
配音 宣晓鸣



蒋龙 饰 小七
配音 苏尚卿



张雅钦 饰 雅兰珠

简介 邛叶公主

配音 吟良犬



王鹤润 饰 凤净梵

简介 璇玑国二王女

配音 蔡娜



周俐蒧 饰 时岚^[4]

简介 扶摇假扮宇文紫时的婢女

配音 张晗



魏晖倪 饰 简雪

简介 太渊国德夫人

配音 曹一茜

打印出阮经天所在的演员字典中的演员名以及角色名，并统计一个有多少个角色。

接下来重点描述一下杨幂主演的角色信息，扶摇。重新创建一个新的字典存放以下信息：扶摇的名字，喜欢她的男角色有（长孙无极、战北野、小七），去过的国家（太渊、天权、天煞、璇玑）

扫码加入 Python 实战圈





Python 实战圈

学python，来实战圈