列表、元组、字典、集合

列表 list

列表是Python内置可变序列之一，是包含若干元素的有序连续内存空间

列表定义

命令	语法	输出	备注
直接定义	list1=[1,2]	[1,2]	
字符串转列表	str1='CDA' list(str1)	['C','D','A']	
创建副本	list2=list1.copy()	[1,2]	内容一样，ID不一样

列表定义在一对中括号中，每个元素用逗号隔开，每个元素类型可以不同，没有长度限制

当列表元素增加或删除时，列表对象自动进行扩展或收缩内存，保证元素之间没有缝隙（自动内存管理）

列表切片

语法: mylist[start:stop:step]

切片包含start,不包含stop，与字符切片类似，左闭右开

表示方法	第1个元素	第2个元素	第3个元素	……	第n-1个元素	第n个元素
正数下标	0	1	2	……	n-2	n-1
负数下标	-n	-(n-1)	-(n-2)	……	-2	-1

列表元素的修改

通过元素的索引位置来修改元素

位置不变，元素内容变化　　list[0]=99

切出来几个元素，就需要赋值几个元素　　list[1:3]=[77,88]

多层切片　　list[1][-1]=88

列表可被遍历　　列表中每个元素乘以2

```
for i in list1:
    print(i*2)
```

列表间的同步计算

```
list7 = [1,2,3,4]
list8= [11,12,13,14]
list9=[0,0,0,0]
for i in range(4):
    list9[i]=list7[i]+list8[i]
list9
```

列表间嵌套的同步计算

```
list10 = [[1,2,3],[4,5,6]]
list11 =[[11,12,13],[11,12,13]]
list12=[0,0,0],[0,0,0]]
for i in range(2):
    for j in range(3):
        list12[i][j]=list10[i][j]+list11[i][j]
list12
```

#外层括号内取值数
#内层括号内取值数

列表对象的常用方法

列表操作	命令	语法	参数	备注
增	append	list1.append(S)	list1：需要操作的列表 S：增加的内容	• 将对象作为元素追加到列表的末尾 • ID不变，内容改变
	extend	list1.extend(list2)	list1：需要操作的列表 list2：需增加的对象	• 将对象列表中的元素在末尾逐一添加 • ID不变，内容改变 • list2必须是可迭代可遍历的内容
	insert	insert(index,object)	index：需要添加的位置索引 object：需要添加的内容	在指定位置index前插入元素object
删	del+切片	del list1[x]	list1：需要操作的表 x：需要删除的位置索引	将需要删除的内容用切片索引出来
	pop	list1.pop(x)	list1：需要操作的表 x：需要删除的位置索引	• 按位置删，默认为-1，删除最后一个 • 执行后会显示删除了的内容
	remove	list1.remove(S)	list1：需要操作的表 S：需要删除的内容	按元素删，并且只删除一次
	clear	list1.clear()	list1：需要清空的表	清空列表，但保留列表结构，内容为空
	del	del list1	list1：需要删除的表	彻底删除变量
查	in	x in list1	list1：需要查找的表 x y：需要查找的内容	为真返回TURE，否返回FALSE 一次只能判断一个元素，最后in的那个y
	not in	x,y not in list1		
排序	sort	list1.sort() list1.sort(reverse=Ture)	list1：需要排序的表 reverse：排序参数	• 对列表中的元素进行排序，默认升序 • 直接修改原数据 • reverse为TURE，降序排列
	sorted函数	sorted(list1)	list1：需要排序的表	• 对列表中的元素进行排序，默认升序 • 不修改原数据，需要赋值变量
	reverse	list1.reverse()	list1：需要排序的表	• 对原列表元素进行逆置 • 效果等同于：list1[::-1] • 只是逆向排列，不涉及大小判断
元素统计	count	list1.count(x)	list1：需要统计的表 x：需要统计的元素	• 统计指定列表中某个元素出现的总次数 • 不能指定范围
元素查找	index	list1.index(x,start,stop)	list1：需要统计的表 x：需要统计的元素 start:开始查找的位置索引 stop：停止查找的位置索引	• 查找指定列表中的某个元素的索引位置 • 可以指定范围，若范围内没有则报错
复制	=	list2=list1	list1：需要拷贝的表 list2：拷贝的表	只用一个等号赋值的，2个列表相互不独立，会跟随改变（ID一致）
浅拷贝	copy	list2=list1.copy()	list1：需要拷贝的表 list2：拷贝的表	• 拷贝的表ID不同 • 浅拷贝只复制单元素一层，嵌套的情况下嵌套内容会被修改
深拷贝	deepcopy	import copy list2=list1.deepcopy()	list1：需要拷贝的表 list2：拷贝的表	• 拷贝的表ID不同 • 嵌套列表不受影响
列表推导式	[ ]	[i for i in range(10)]	创建0-10的列表	必须要写列表中括号
		[list1[i]+list2[i] for i in range(4)]	两个列表元素相加	range（4）等同于从0到3，不包含4
	range函数	range(start, stop[, step])	• start: 计数从 start 开始。默认从 0 开始 • end: 计数到 end 结束，但不包括 end • step：步长，默认为1	range() 函数返回的结果是一个整数序列的对象，而不是列表，需要进行一定的转换才能以列表的形式显示。

元组 tuple

定义

一种不可变的有序容器

元素有位置上的先后顺序

标志性符号是圆括号和逗号（,），括号可省略

适用场景

元组比列表操作速度要快，适合遍历

不需要被修改的数据可以用元组“保护起来”

由于不可变属性，在很多方法和操作中只能用元组作为结构的一部分，比如没有嵌套列表的元组可以作为字典中的key

支持序列基本操作

没有增、删操作

可查找/、切片、运算等

命令	语法	输出	备注
直接定义	tp1=(1,2)	(1,2)	括号可省略
单个元素的元组	tp1=('a',)	'a'	创建只包含一个元素的元组时，不要忘记写逗号
强制转换	tp1=tuple(list1)	(1,2,3)	转换后需要赋值
元组切片	tp1[1:-2]	(2,3)	取多个时，切出来的左闭无闭 切出来的元素不可修改 若元组中有嵌套列表，切出来后列表内的元素可修改（各个元素ID不改变）

字典 dict

定义

字典是一个可变、无序的容器

字典内的元素都是键值对，每一个值（value）都对应一个键（key）

标志性符号是花括号{}和冒号

字典可遍历

默认的是键

```
for i in dict1:
    print(i)
```

需要取值时用

```
for i in dict1:
    print(dict1[i])
```

设置两个变量获取键和值

```
for i,j in dict1.items():
    print(i,j)
```

命令	语法	输出	备注
直接定义，左键右值	dict1={'a':1,'b':'c'3}	{'a':1,'b':'c'3}	字典的键不可以出现重复值，值可以重复 若键有重复，则输出最后一个键值对
函数创建字典	dict1=dict(a=1,b=2)	{'a':1,'b':2}	用函数时，需要赋值
函数强制转换	dict1(list1)	{'a':1,'b':2}	列表嵌套列表，列表嵌套元组，元组嵌套元组均可强制转换为字典
创建值相同的字典	dict1.fromkeys(['a','b'],9)	{'a':9,'b':9}	只有一个值参数，只能创建的值相同时使用 值可以是一个列表 输出结果是自动显示的，不能手动设置
访问字典的值	dict1['a']	{'a':1,'b':2}	通过键访问字典的值，没有索引
修改字典的值	dict1['a']=2	{'a':2,'b':2}	通过键修改字典的值
增加键值对	dict1['c']=2	{'a':9,'b':9,'c':2}	键必须在原字典中不存在 若键已存在，则直接改变键的值
	dict1.setdefault('a',3)	9	若键已存在，不添加，并返回原键的值 若键不存在，未赋值，则添加后键值对的值为none
删	del dict1['a']	{'b':9,'c':2}	通过键进行删除
	dict1.pop('a')	9	必须指定删除的键，并返回删除键对应的值 删除后的键的值值可赋值给变量
	dict1.popitem()	('c',2)	随机删除，没有参数，默认删除最后一个看到的键值对 删除后的键的值值可赋值给变量
清空	dict1.clear()	{}	清空字典
	del dict1		彻底删除变量
查	'a' in dict1	true	查找的是键，为真返回TRUE，否则返回FALSE
	'd' not in dict1	false	
get函数查找	dict1.get('a')	9	查找键并返回找到的值 若没有找到，则无输出（若设定了default参数，则返回该参数的内容）
获取字典所有的key	dict1.keys()	dict_keys(['a','b'])	输出的结果是对象，可以整循环，也可用list函数转成列表
获取字典所有的值	dict1.values()	dict_values([9,2])	
字典和其他数据类型的转换	dict1.items()	dict_items([('a',9),('b',9)])	
更新字典	dict1.update(dic2)	二者合并，同键更新	用dic1更新dict1 二者都存在的键，保留dic2的键值对 dic1有，dic2没有的保留 dic1没有，dic2有的添加进dic1

集合 set

定义

集合是一种无序的可变的容器

对应数学中的集合,标志性符号是花括号{}

集合中的元素被看作是字典中的键(里面的元素需要是不可变的数据类型)，

集合里面没有重复值

适用场景　　集合常用来对其他数据类型进行去重

命令	语法	输出	备注
直接定义	set1={1,2,3,'a'}	{1,2,3,'a'}	值不重复
函数转换集合	set(tp1)	{1,2,3,4}	用函数时，需要赋值 转换后的内容为去重后的集合
增	set1.add('b')	{1,2,3,'a','b'}	增加的内容由系统自动排序，不能指定位置 不能使用切片
随机删	set1.pop()	1	随机删除一个元素，并输出删除的内容
按元素的值删	set1.pop('a')	{1,2,3,'b'}	删除指定的元素，并输出删除后的集合
清空	set1.clear()	set()	清空集合
改	set1.update(set2)	二者合并	二者合并的去重后集合
查	1 in set1	true	为真返回TRUE，否则返回FALSE 查找的内容为元组时，只能作为一个元素判断（子集）
	(2,3) not in set1	true	

专属于集合的操作

求并集　|	set1 | set2		
求交集　&	set1 & set2		
求差集　-	set1 - set2	set1有，set2没有的内容	
对称差分	(set1 | set2)-(set1 & set2)	并集-交集的部分	
集合的判断	set1 = = set2	判断的是集合中的元素是否相等	
	set1 is set2	判断ID是否相同	

基本运算	运算符号
包含	in
不包含	not in
等于	==
不等于	!=
子集	<
父集	>
并集	|
交集	&
差集	-
对称差分	^
删除集合元素	pop/remove
增加集合元素	add

序列

可变对象

该对象所指向的内存中的值可以被改变（原地改变）

列表、字典、集合这三类是可变数据类型

不可变对象

该对象所指向的内存中的值不能被改变

当改变某个变量时候，把原来的值复制一份，变量再指向这个新地址

字符串、整数、浮点数是不可变对象

列表：[ ]
元组：( )
字典：{ : }
集合：{ }