'''

有时候我们需要将一些互相之间有关联的数据保存在一起，很多接触过编程的朋友都会想起一个概念叫做数组

数组允许吧相同类型的数据挨个摆在一起，然后通过下表进行索引

Python中也有类似与数组的东西,不过更为强大。

由于python的变量没有数据类型,所以python的"数组"可以同时存放不同类型的变量,在python中被我们称作列表

'''

创建数组:只需要使用中括号将数据包裹起来（数据之间用逗号分隔）

```
list1 = [1, 2, 3, 4, 5, 6]

print(type(list1))

# range(0,20)--> [0, 1,2,3,4,5, ... , 19]

for i in list1:

        print(i)
```

'''

没有哪项规定Python的列表保存同一个数据类型,因此它支持各种不同的数据存放在一起

'''

```
list2 = [1, 222, 'niu', 3.14, [1,2,3]]

for i in list2:

    print(i)
```

如果是在不知道要在列表里面放什么类型的话，那我们就创建一个空列表

```
empty = []
```

'''

向列表添加元素：列表不是一成不变的，可以随意地往里面添加新元素

可以使用append()方法

' {}'.format()

'''

```
num = [1, 2, 3]

num.append(6)

print(num)
```

'''

介么方便,那我是不是可以把东西全部用append()塞入num的后备箱呢

'''

```
# num.append(1,3)

# print(num)
```

'''

当我们想要将多个元素塞入列表的时候,这个时候我们就需要使用到extend()

extend():使用一个列表来扩充另一个列表，所以他的参数是列表

'''

```
num.extend([5, 6])

print(num)
```

'''

无论是使用append()还是使用insert()方法，都是往列表的末尾添加数据，是否可以将数据插入特定的位置呢?

当然是可以的，这里我们就要使用一个新的方法,那就是insert()

insert():他有两个参数，第一个参数指定待插入的位置(索引值),第二个参数是待插入的元素值

在insert()中支持第一个参数为负数，表示与列表末尾的相对距离

tips: 在编程语言中,我们都是从0开始数数的,所以我们的第一个索引值是0,而不是1

'''

```
num.insert(0, 110)

print(num)



num.insert(-1, '-1')

print(num)
```

'''

那我们有了添加，我们想要获取列表里面的元素该怎么做呢?

当然可以直接通过索引值获取列表中的某个元素

'''

```
eggs = ['鸡蛋', '鸭蛋', '鹅蛋', '翟沁妤']

print(eggs[3])
```

'''

如果我向直接访问最后一个元素，怎么办?

这里我们介绍一个新的内置函数len()

len()获取列表的长度,利用列表的长度-1进行索引就可以啦

'''

```
print(eggs[len(eggs) - 1])
```

'''

当然了len()也可以直接用负数索引

'''

```
print(eggs[-1])
```

'''

那我们想要'鸭蛋'和'蠢蛋'调换位置我们应该怎么做呢?

'''

```
temp = eggs[1]

eggs[1] = eggs[2]

eggs[2] = temp

print(eggs)
```

'''

temp在这里是一个临时变量,避免互相覆盖

在python中有一种偷懒的方式

'''

```
eggs[1], eggs[2] = eggs[2], eggs[1]

print(eggs)
```

'''

列表里面还可以嵌套列表,并且我们可以对其进行索引

'''

```
eggss = ['鸡蛋', '鸭蛋', ['天鹅蛋', '企鹅蛋', '加拿大鹅蛋'], '鹅蛋', '蠢蛋']

print(eggss[2][2])
```

'''

添有了，找有了，删我们该怎么做呢?

在列表中删除元素，有三种方法

1. pop():将列表中的指定元素'弹'出来,也就是取出并删除的意思，他的参数是一个索引值(如果不带参数，他就删除最后一个)
2. remove():需要指定待删除的元素，不需要知道元素再列表中的具体位置(列表没有的元素不可以删除哟)
3. del: del在python的应用非常丰富,不经可以用来删除某个元素，还可以用来删除整个变量

'''

```
eggss1 = ['鸡蛋', '鸭蛋', ['天鹅蛋', '企鹅蛋', '加拿大鹅蛋'], '鹅蛋', '蠢蛋']



print('_' * 25)

eggss1.remove('鸭蛋')
```

```
print(eggss1)

print('_' * 25)

eggss1.pop(1)

print(eggss1)

print('_' * 25)

del eggss1[0]

print(eggss1)
```

'''

列表的切片（slice）语法的引入，是的python真正走向了高端

'''

要求将列表1的三个元素取出来，放到列表2中

```
list1 = ['aaa', 'bbb', 'ccc', 'ddd', 'eee']

list2 =[list1[0], list1[2], list1[-1]]

print(list2)
```

'''

像这样，从列表取出元素是非常常见的操作，但是这里是取出三个元素

如果我们要求取出最后200个元素，那时不是很辛酸

'''

```
list2 = []

for i in range(-200, 0):

    list2.append(list1[i])
```

'''

虽然可以这样，但是每次都要嵌套一个循环太麻烦了，切片的引入很好的解决了这一个问题

'''

```
list2 = list1[2:5]

print(list2)
```

'''

需要注意的是，结尾的位置是不会被包含的

当然切片列表还有偷懒的写法

'''

```
list2 = list1[:2]

print(list2)

list3 = list1[-2:]

print(list3)
```

'''

切片的进阶玩法：列表的切片其实还有第三个参数（步长：默认值为1）

'''

```


list4 = list(range(10))

print(list4)

print(list4[0:4:2])

# 如果步长为-1会实现什么结果呢?

print(list4[::-1])
```

'''

我们之前学的一些常见的操作符同样也可以用在列表中

'''

```
list11 = [123, 456]

list22 = [156, 123]

print(list11 > list22)
```

'''

为什么列表可以判断大小了？是根据什么来判断的呢?

其实列表或者字符串都是对第一个元素进行比较

这就需要知道ASCII这个概念了

'''

'''

我们之前在拼接两个列表的是时候用的是extend，但其实我们还有一种更方便的方法

列表的拼接,就和我们之前字符串的拼接一样

'''

```
list5 = list11 + list22

print(list5)
```

'''

(*)乘号也叫做重复操作符，重复操作同样可以使用列表中

'''

```
listC = ['nihao']

print(listC * 3)
```

'''

我们之前在for循环的时候经常可以见到一个 in 他是一个成员关系操作符

in 和 not in

'''

```
list_in = ['pig', 'dog', 'fish']

print("people" in list_in)

print("people" not in list_in)
```

'''

那我们能不能用 in and

not in 来测试元素在不再列表内呢?

'''

```
print('大聪明' in list_in)

print('大聪明' not in list_in)
```

'''

in 和 not in 只能判断一个层次的成员关系，这个和break和continue语句只能跳出一个层次的循环是一个道理

在开发中有时候我们需要去除重复的数据，只好利用in 和 not in

'''

```
old_list = ['西班牙','老美','中国','俄罗斯','老美']

new_list = []

for i in old_list:

•    if i not in new_list:

•        new_list.append(i)

print(new_list)
```

'''

到如今我们学了这么多list的用法

那么list到底还有多少用法呢

我们可以使用一个内置函数dir来查看

'''

```
print(dir(list))
```

'''

['**add**', '**class**', '**class_getitem**', '**contains**', '**delattr**',

'**delitem**', '**dir**', '**doc**', '**eq**', '**format**', '**ge**',

'**getattribute**', '**getitem**', '**gt**', '**hash**', '**iadd**',

'**imul**', '**init**', '**init_subclass**', '**iter**', '**le**', '**len**',

'**lt**', '**mul**', '**ne**', '**new**', '**reduce**', '**reduce_ex**',

'**repr**', '**reversed**', '**rmul**', '**setattr**', '**setitem**',

'**sizeof**', '**str**', '**subclasshook**',

'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',

'pop', 'remove', 'reverse', 'sort']

哇塞一共有这么多东西，那我们继续介绍以下吧

'''

'''

count():作用是统计某个元素在列表中出现的次数

'''

```
list_con = [1, 2, 3, 4, 5, 6, 7, 4, 5]

print(list_con.count(4))
```

'''

index():作用是返回某个元素在列表第一次出现的索引值

'''

```
print(list_con.index(4))
```

'''

reverse():方法可以将整个列表原地翻转

'''

```
list_con.reverse()

print(list_con)
```

'''

sort():方法的作用是对列表的元素进行排序，默认从小到大

'''

```
list_sort = [3, 4, 7, 2, 5, 9]

list_sort.sort()

print(list_sort)
```

'''

那我们如果想从大到小排序该怎么做呢?

'''

```
list_sort.reverse()

print(list_sort)
```

'''

以前我们说过python的精髓是什么？偷懒

那我们肯定有方法不需要这么麻烦的拉

sort(func, key, reverse)

sort的reverse默认参数为False，表示不用颠倒排序

现在你会了吗?

'''

```
list_last = [9, 2, 3, 5, 6, 7, 10, 11, 20, 25]

list_last.sort(reverse=True)

print(list_last)
```