Python基础 列表和元组

序列

序列是一块用于存放多个指的连续内存空间，并且按照一定顺序排列，每一个值(元素)都分配一个数字，称为索引或位置。

1. 索引

   索引是从0开始的。Python的索引是可以为负数的，当为负数时，就是从右边数。

```
names = ['路飞', '娜美', '乔巴']
print(names[1]) #这里的1就是索引
print(names[-1]) #-1表示从右边数第一个
```

2. 切片

   切片操作是访问序列中元素的另一种方法。语法：

> sname[start : end : step]
>
> sname序列名
>
> start：开始位置，包括该位置的元素
>
> end：结束位置，不包括该元素
>
> step：步长，即每次增加多少。默认为1

```
names = ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克']
print(names[1:5]) #获取第二个元素到第5个元素的内容
print(names[0:5:2]) #获取第1，  3，  5个元素
```

```
['娜美', '乔巴', '索隆', '山治']
['路飞', '乔巴', '山治']
```

3. 序列相加

   python中支持两种类型相同的序列相加操作。用运算符 + 即可。

```
names1 = ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克']
names2 = ['鸣人', '小樱']
print("names1 + names2:",names1 + names2) #将连个序列相加
```

```
names1 + names2: ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克', '鸣人', '小樱']
```

   这里的类型相同指的是那个同为列表，元组，或集合等，而不是元素的类型

```
nums = [1, 2, 3, 4]
print("nums + names2:",nums + names2) #将连个序列相加
```

```
nums + names2: [1, 2, 3, 4, '鸣人', '小樱']
```

4. 乘法

   在python中使用数字n成一个序列会产生一个新的序列。新序列的内容为元序列被重复的n次的结果。

```
phone = ['锤子', '华为', 'VIVO']
print("phone * 3:",phone *3) #将会重复输出3次phone的内容，并生产一个新的序列
```

```
phone * 3: ['锤子', '华为', 'VIVO', '锤子', '华为', 'VIVO', '锤子', '华为', 'VIVO']
```

5. 检查某个元素是否是序列的成员（元素）

   语法：

   > value in 序列名

```
names = ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克']
print("路飞在names里吗：", '路飞' in names)
```

```
路飞在names里吗： True
```

   也可以用 not in

```
print("路飞不在names里吗：", '路飞' not in names)
```

```
路飞不在names里吗： False
```

6. 计算序列的长度，最大值，最小值

```
nums = [1, 2, 3, 4, 78, 21, 12, 43, -32]
print("nums的长度为：",len(nums)) #获取nums的长度，即包含几个元素

print("nums中最大的是:",max(nums)) #获取nums的最大值
print("nums中最小的是:", min(nums)) #获取nums的最小值
```

```
nums的长度为： 9
nums中最大的是: 78
nums中最小的是: -32
```

列表

1. 列表的创建和删除
   - 使用赋值运算符直接创建列表

     > listname = [element1, element2, element3]

```
#使用赋值运算符创建列表
numList= [7, 3, 2, 3]
print(numList)
```

- 创建空列表

```
emptyList = [] #创建空的列表
```

- 创建数值列表

```
#创建数值列表
numList2 = list(range(10, 20, 2)) #range(10, 20, 2)表示从10~20，每次增加2个
print(numList2
```

```
[10, 12, 14, 16, 18]
```

2. 删除列表

   语法:

   > del listname

   del在实际开发中并不经常使用，因为Python自带垃圾回收机制，如果一个对象不用了，他会自动删除

```
#del 删除对想
print("-----删除前-----")
print(numList)
print("-----删除后-----")
del numList
print(numList)
```

```
NameError: name 'numList' is not defined
```

   这里会报错，因为已经删除了。

3. 访问列表元素

   和序列的操作是一样的。

4. 遍历列表

   - 直接使用for循环遍历，语法：

     > for item in listname:
     >
     > do something

```
#用for循环遍历列表
namelist = ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克']
for item in namelist:
    print(item)
```

```
路飞
娜美
乔巴
索隆
山治
布鲁克
```

- 使用for循环和enumerate()函数实现,语法:

> for index,item in enumerate(listname):
>
> do something
>
> index用于保存元素的索引
>
> item用于保存渠道的值
>
> listname：列表名

```
print("2018年俄罗斯世界杯四强:")
team = ['法国', '比利时', '英格兰', '克罗地亚']
for index,item in enumerate(team):
    print(index + 1,item)
```

```
1 法国
2 比利时
3 英格兰
4 克罗地亚
```

5. 添加，修改和删除列表元素
   - 添加元素

> listname.append(obj)
>
> obj:要添加的元素

```
# 添加元素
print("添加前的namelist: ", namelist)
namelist.append('弗兰奇')
print("添加后的namelist: ", namelist)
```

```
添加前的namelist:  ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克']
添加后的namelist:  ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇']
```

还可以使用extend()函数添加

```
print("添加前的namelist: ", namelist)
newList= ['女帝']
namelist.extend(newList)
print("添加后的namelist: ", namelist)
```

```
添加前的namelist:  ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇']
添加后的namelist:  ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇', '女帝']
```

   - 删除元素

```
# 使用del删除元素
print("删除前的namelist: ", namelist)
del namelist[-1]
print("删除后的namelist: ", namelist)
```

```
删除前的namelist:  ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇', '女帝']
删除后的namelist:  ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇']
```

```
#根据元素值删除
print("删除前的namelist: ", namelist)
namelist.remove('路飞') #删除路飞
print("删除后的namelist: ", namelist)
```

```
删除前的namelist:  ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇']
删除后的namelist:  ['娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇']
```

在使用remove时最好先判断一下是否存在，否则会报错

```
remVal = '女帝' #这个是不存在的
namelist.remove(remVal) #没有进行判断
```

```
ValueError: list.remove(x): x not in list
```

```
remVal = '女帝' #这个是不存在的
if namelist.count(remVal) > 0:
    namelist.remove(remVal) #没有进行判断
print(namelist)
```

```
['娜美', '乔巴', '索隆', '山治', '布鲁克', '弗兰奇']
```

加了判断就不会报错了。

6. 对列表进行统计计算
   - 统计元素出现的次数

```
# 获取指定元素出现的次数
namelist = ['路飞', '娜美', '乔巴', '索隆', '山治', '布鲁克']
namelist2 = ['路飞',  '索隆', '山治', '布鲁克']
namelist.extend(namelist2)

print("索隆出现了: ",namelist.count('索隆'),"次") #统计索隆出现的次数
```

```
索隆出现了:  2 次
```

   - 获取指定元素第一次出现的坐标

```
print("索隆第一次出现的下标:",namelist.index("索隆"))
```

```
索隆第一次出现的下标： 3
```

- 统计数值列表的元素和

```
numlist = [1, 2, 3, 44, 5, 34,23]
print("numlist所有元素的和：", sum(numlist))
```

```
numlist所有元素的和： 112
```

7. 对列表进行排序
   - 使用sort方法实现

```
# 对numlist排序
print("排序前：", numlist)
numlist.sort()
print("排序后：", numlist)
```

```
排序前： [1, 2, 3, 44, 5, 34, 23]
排序后： [1, 2, 3, 5, 23, 34, 44]
```

   - 使用sorted方法实现

```
numlist = [1, 2, 3, 44, 5, 34,23]

numlist_as = sorted(numlist) # 升序，默认
print("numlist升序排列:", numlist_as)
numlist_des = sorted(numlist, reverse = True)
print("numlist降序排列:", numlist_des)
```

```
numlist升序排列: [1, 2, 3, 5, 23, 34, 44]
numlist降序排列: [44, 34, 23, 5, 3, 2, 1]
```

   sort和sorted作用基本一样。不同的地方是sort会改变原来列表的顺序，而sorted这是创建一个副本，不会改变原的排序。

8. 列表推导式

```
# 列表推导式
newnums = [int(x * 2) for x in numlist] #这个就是推导式
print(newnums) #numlist的元素乘以2 生成性的列表
```

```
[2, 4, 6, 88, 10, 68, 46]
```

元组

1. 元组的创建和删除

```
# 使用赋值运算符创建
num = (1, 2, 3, 4)
print(num)
print(type(num)) #查看num的类型
```

```
(1, 2, 3, 4)
<class 'tuple'>
```

```
del num # 同样使用del
```

2. 访问元组元素

   也是通过下标访问，或者使用切片

```
# 访问元组
num = (1, 2, 3, 4, '路飞')
print(num[0]) #访问第一个元素
print(num[:3]) #获取前三个
```

```
1
(1, 2, 3)
```

3. 修改元组元素

   元组是不可变得序列，如果要修改，就需要重新赋值

4. 元组和列表的区别

   - 列表属于可变序列，而元组则是不可变得
   - 列表是用append(),extend(),insert(), remove(), pop()等方法实现添加和修改。
   - 元组比列表访问和处理速度快
   - 列表不能作为字典的键，而元组则可以。