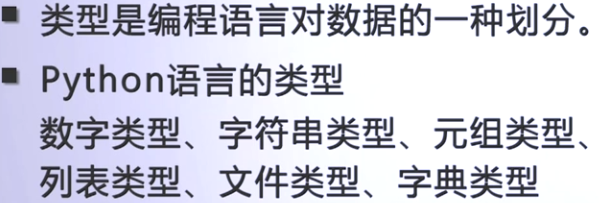
Python中的数据类型

# Python中的数据类型



**布尔类型bool**

**Python中没有数组的定义。**

# 布尔类型

只有两个：**True 和 False。（注意：首字母大写，否则不是布尔值）**

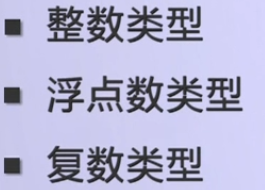
**True可以1表示，0表示False**。非零数字，代表True。

print(True+True)#2  
print(True+False)#1

print(type(True))#<class 'bool'>  
print(type(False))#<class 'bool'>

# 数字类型

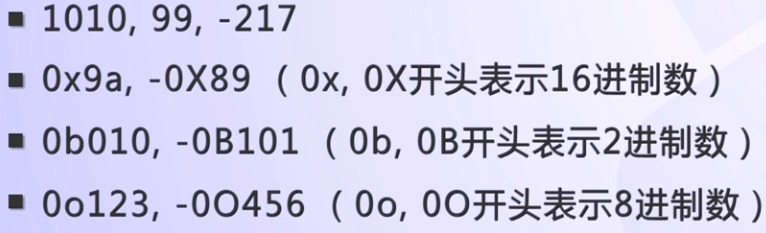
Python中数据类型主要**有三种**：



## 整数类型

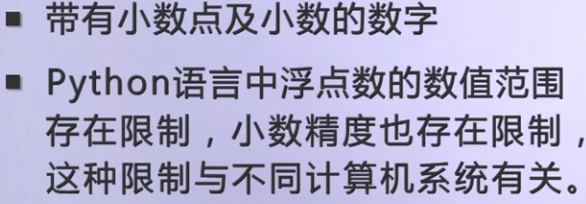
没有位数限制，无论多大的数都可以算出来，前提是计算能力允许。

**16进制以0x开头，2进制以0b或0B开头，8进制以0o或0O开头。**



a = pow(2,10)  
print(a)#1024  
print(pow(2,pow(2,8)))  
#115792089237316195423570985008687907853269984665640564039457584007913129639936

## 浮点数类型



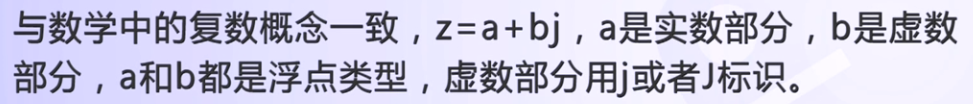
print(type(3))#<class 'int'>  
print(type(3.1415))#<class 'float'>  
print(type(3.13+8.9j))#<class 'complex'>

一般精度也是相当够，所以也可以认为浮点数类型也是没有位数限制的。

import sys  
print(sys.float\_info)

sys.float\_info(max=1.7976931348623157e+308, max\_exp=1024, **max\_10\_exp=308, min=2.2250738585072014e-308**, min\_exp=-1021, **min\_10\_exp=-307**, dig=15, mant\_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)

## 复数类型 : z = a+bj



**利用复数的属性real和imag获取其实部和虚部**。

z = 4 + 5j  
print(z)#(4+5j)  
print(z.real)#4.0  
print(z.imag)#5.0

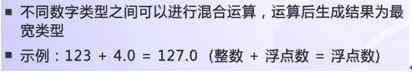
## 三种数字类型之间的关系



## 三种数字类型之间的转换

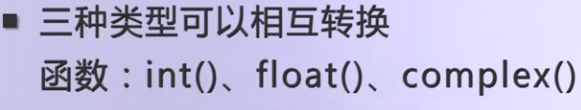
### 自动数据类型转换

不同的数字类型运算，自动转换成级别高的类型。

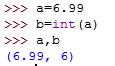


### 强制数字类型转化

**利用int()、float()、complex()三个函数完成。**



**注意：浮点型转成整型时，不是四舍五入，而是直接把小数去掉。**



a = 10.54  
b = int(a)  
print(b) #10  
print(complex(b)) #(10+0j)

注意不能将复数类型转化为整型或浮点型。

z = 3 + 9j  
print(int(z))#TypeError: can't convert complex to int  
print(float(z))#TypeError: can't convert complex to float

### 数字类型与字符串类型之间的转换

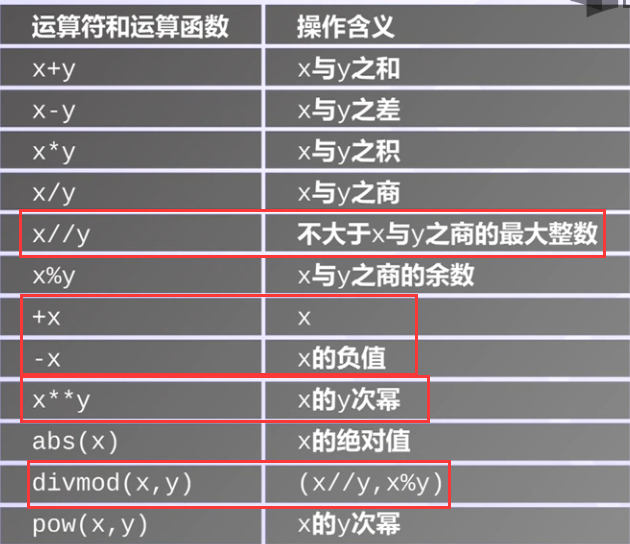
**整型或浮点型**都可以转成字符串，但是**非数值含义的字符串**不可以转成整型或浮点型。

str1 = '100'  
a = int(str1)  
print(a)#100  
print(type(a))#<class 'int'>  
str2 = '99.909'  
b = float(str2)  
print(b)#99.909  
print(type(b))#<class 'float'>  
str3 = '4+5j'  
c = complex(str3)  
print(c)#(4+5j)  
print(type(c))#<class 'complex'>

也可以利用eval()函数自动转换成相应的类型：

str1 = '100'  
str2 = '99.909'  
str3 = '4+5j'  
print(type(eval(str1)))#<class 'int'>  
print(type(eval(str2)))#<class 'float'>  
print(type(eval(str3)))#<class 'complex'>

## 数字类型之间的运算符



除此之外，利用**math库**，可以使用更多的函数，完成相应的计算。

# Python中索引的使用

## 具有正负索引，-n表示倒数第n个元素。

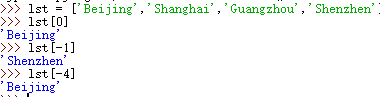
正索引： 0、1......N-1；

负索引：-N、-（N-1）......、-1 。

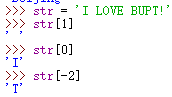
**正索引与负索引完全等价的**。

**负索引 = 正索引 – N；N = 正索引+|负索引|**

例如：列表



**字符串也可以利用索引获取字符或字符串**：



## 索引范围：满足左闭右开原则。

range(N): 为 0-N-1；

range(N,M):为N-M-1 ； M>N ，否则无数据。

字符串中取子字符串：

### 从索引n开始到结束：str[n:] 包括索引n的元素（左闭），

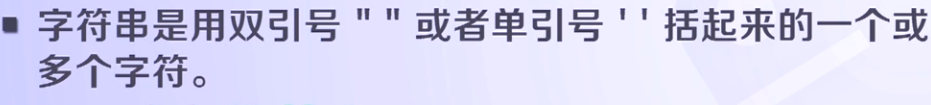
### 从开头到索引n： str[:n] 不包括索引n的元素（右开）。

### 从索引n到索引m：str[n,m] 包括索引n，不包括索引m。

# 字符串类型

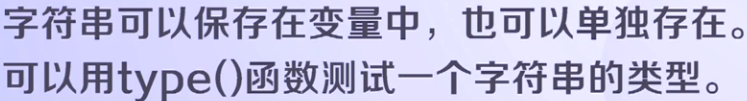
## 字符串定义





三引号：可以是**三个单引号**，也可以是**三个双引号**。

str1 = """  
北京  
邮电  
大学  
"""  
str2 = '''  
BUPT  
in   
Beijing  
'''



print('aaa')  
print("aaa")  
a = 'BUPT'  
print(a)  
print(type(a))#<class 'str'>

## Python中的字符与字符串

Python中没有**字符数据类型**，认为**字符就是长度为1的字符串**。

## 如何输出单引号或双引号或反斜杠

单引号中可以包含双引号，双引号中也可以包含单引号；

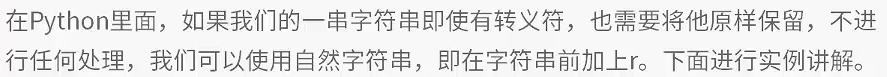
但是，单引号中的单引号需要利用反斜杠转义，同理，双引号中的双引号也需要利用反斜杠进行转义。

**需要输出反斜杠本身，需要自己对自己进行转义**。

print('"aaa"')#"aaa"  
print("'aaa'")#'aaa'  
print('\'aaa\'')#'aaa'  
print("\"aaa\"")#"aaa"  
print('\\aaa\\')#\aaa\

print('E://new/code.txt')#E://new/code.txt  
print('\home\new\code.txt')#这时候\n表示换行，需要进行转义  
print('\home\\new\code.txt')#\home\new\code.txt

## 自然字符串（原生字符串） r 表示raw



在**定义的字符串前面加上个字符r**，表示此字符串是**原始字符串**，就会以原始的字符串输出。**但是字符串不能以反斜杠结尾**。

print('I am a \n student!')  
#I am a   
# student!  
print(r'I am a \n student!')#I am a \n student!

print('I am a \\n student!')#I am a \n student!

print(r'\home\new\code.txt')#\home\new\code.txt  
print(r'\home\\new\code.txt')#\home\\new\code.txt  
print(r'\home\\new\code.txt\') #错误，不允许以反斜杠结尾

## 三重引号的使用

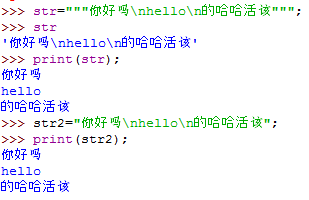
**三重引号两种功能**：

### 第一种：多行注释；

### 第二种：可以用来定义多行字符串

要得到一个**跨越多行**的字符串，需要利用**三重引号**。即三重引号字符串。

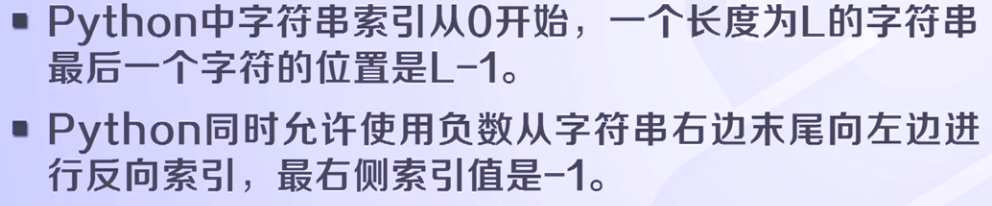
#三重引号两种功能：第一种：多行注释；第二种：可以用来定义多行字符串  
*"""  
这是一个多行注释  
第一行  
第二行  
"""*str1 = """中国  
美国  
英国  
"""  
print(str1)



**没有区别啊?不知道哪里不对？**

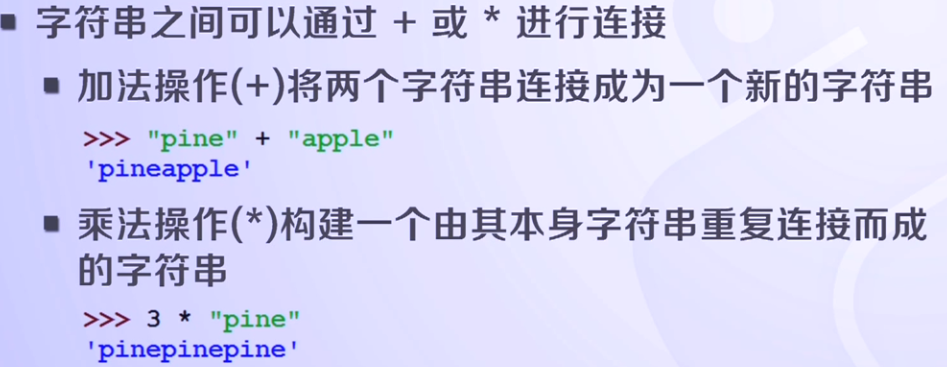
## 字符串的索引

符合Python中索引的一般使用方法。



str1 = 'I love BUPT very much'  
print(str1[0])#I  
print(str1[-1])#h  
print(str1[7:])#BUPT very much  
print(str1[10:-1])#T very muc

## 字符串的加法与运算：



print('aaa'+'bbb')#aaabbb  
print('aaa'\*3)#aaaaaaaaa

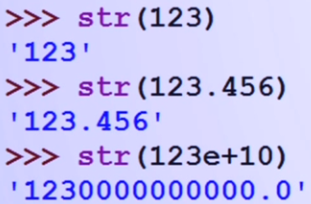
## 字符串的重复输出即字符串的乘法

print('BUPT'\*3)#BUPTBUPTBUPT

## len(str)获取字符串的长度

str1 = 'I love BUPT very much'  
print(len(str1))#21

## str()将其他类型数据转换成字符串类型



print(type(str({'name':'zhaohong'})))#<class 'str'>

## 遍历字符串中的每个字符

利用 **for ch in str1:**

str1 = 'BUPT'  
for ch in str1:  
 print(ch)

与下面方法等价：

str1 = 'BUPT'  
for i in range(len(str1)):  
 print(str1[i])

B

U

P

T

# 字符串对应的函数



## upper()与lower()方法

**将字符串都转换成大写或小写字母**。

str1 = 'University In Beijing'  
print(str1.upper())#UNIVERSITY IN BEIJING  
print(str1.lower())#university in beijing

## capitalize()函数：使字符串的首字母大写

str2 = 'university in beijing'  
print(str2.capitalize())#University in beijing

## isdigit()函数：判断字符串是否全是数字

str1 = '43534'  
str2 = '43534re'  
print(str1.isdigit())#True  
print(str2.isdigit())#False

## find()函数与index()函数：返回索引





两个方法类似，都具有三个参数，都是寻找出字符串中**sub子字符串**出现的位置。后两个参数可以限定寻找的位置。

str1 = 'kkkkddddedeeeder'  
print(str1.find('de'))#7  
print(str1.find('de',8))#9  
print(str1.find('de',10,15))#13  
print(str1.index('de'))#7  
print(str1.index('de',8))#9  
print(str1.index('de',10,15))#13

**两个方法的区别**：

若字符串中不存在sub子字符串，find函数返回-1，而index函数则会报错。

str1 = 'kkkkddddedee'  
print(str1.find('deeee'))#-1  
print(str1.index('deeee'))#ValueError: substring not found

## replace()函数



三个参数：

**old：需要替换的子字符串；**

**new：新的字符串；**

**count:表示最多替换的个数。**

str1 = 'kkkkdeddedeeeder'  
print(str1.replace('de','RR'))  
print(str1.replace('de','RR',1))

## strip()函数：去除字符串两端的空格或指定字符

**方法**：

“\*\*\*”**.lstrip(s[, chars])**

“\*\*\*”**.rstrip(s[, chars])**

“\*\*\*”**.strip(s[, chars])**

**string.lstrip(s[, chars]) ：只删除开头一端**  
Return **a copy of the string with leading characters removed**. If chars is omitted or None, whitespace characters are removed. If given and not None, chars must be a string; the characters in the string will be stripped from the beginning of the string this method is called on.

**string.rstrip(s[, chars])：只删除结尾一端。**  
**Return a copy of the string with trailing characters removed**. If chars is omitted or None, whitespace characters are removed. If given and not None, chars must be a string; the characters in the string will be stripped from the end of the string this method is called on.

**string.strip(s[, chars])：删除开头和结尾两端。**  
**Return a copy of the string with leading and trailing characters removed**. If chars is omitted or None, whitespace characters are removed. If given and not None, chars must be a string; the characters in the string will be stripped from the both ends of the string this method is called on.

**这三种方法，并没有改变字符串本身，而是产生了一个新的字符串。**

其中提供的**参数chars为一个字符串，**用来指示需要删除的**特定字符**。

当没有参数时，**默认删除字符串两端的空格**，获得一个**首尾不含多余空格**的字符串。**“\*\*\*”.strip(' ') 与 “\*\*\*”.strip() 等价。**

str1 = ' Beijing University '  
print(str1.strip(' '))  
print(str1.strip())

**strip**这三个方法要么就是**无参数**，要么就是**字符串**，这个字符串作用就是指示出删除该字符串两端所包含的字符，直至出现其他字符。

示例: 删除字符串两端的**x和y字符**。

str2='xyyyxxzxxyyBUPTxxyyzdxyyx'  
print(str2.strip('xy'))#zxxyyBUPTxxyyzd  
print(str2)#xyyyxxzxxyyBUPTxxyyzdxyyx

当指定特定字符的时候，此时就不会自动删除空格。需要特意指定一个空格字符。

str3 = ' BUPT '  
print(str3.strip('BT'))# BUPT   
print(str3.strip('B '))#UPT

**strip方法会自动拷贝一份，然后删除两端的某些字符，原来的字符不变。**

## join函数

**join函数**的参数为一个列表等，字符串可以插入到这些元素之间，返回一个大字符串。

示例1：

strs = ['I','Love','You','!']  
newStr = "...".join(strs)  
print(newStr)

示例2：

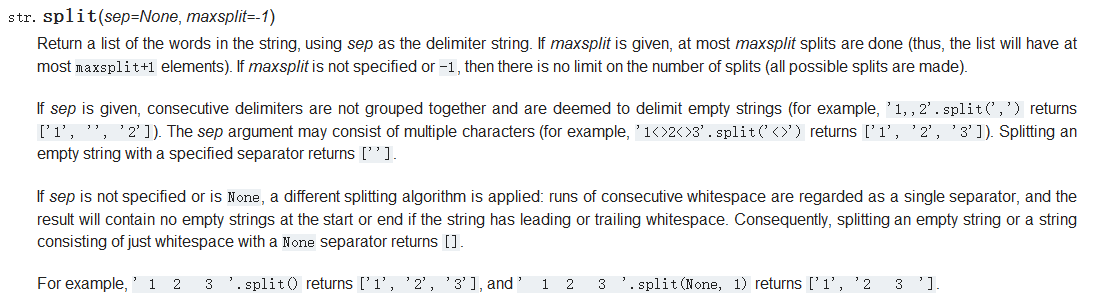
headList = ['姓名','年龄','学历','籍贯']  
userList = ['zhaohong','24','研究生','山东菏泽']  
print('\t'.join(headList))  
print('\t'.join(userList))

姓名 年龄 学历 籍贯

zhaohong 24 研究生 山东菏泽

## split方法：

**str.split(sep,maxsplit): sep为分割字符串，maxsplit是分割次数，分割后为maxsplit+1个部分。**

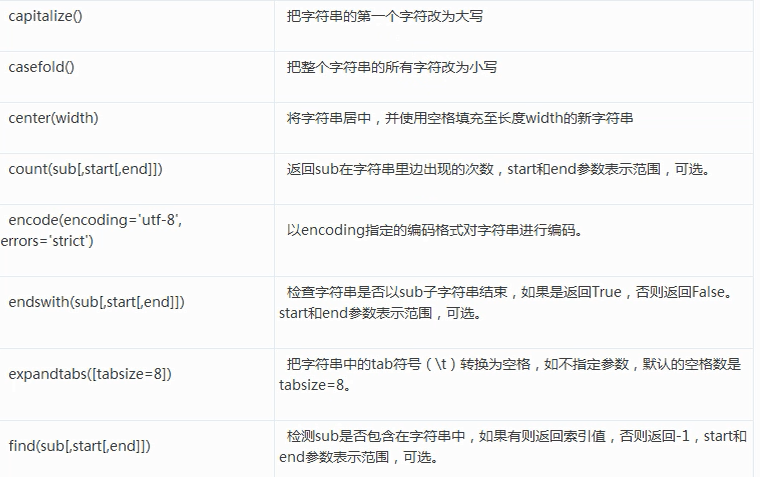


## zfill(width)方法：

**用0填充，使字符数目达到指定的width，0在原字符串的前面填充**。



## Python中的字符串的其他方法：

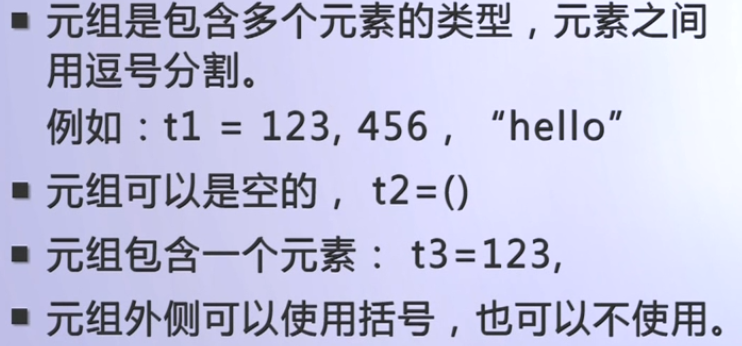






# 元组类型tuple

## 元组概念

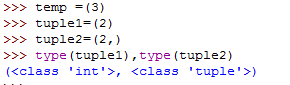


**元祖tuple：是不可以修改的，相当于加上枷锁的列表。**

**元祖和字符串差不错，一旦创建，就不可以修改了。**

**区别是：列表用中括号，而元祖用小括号。**

**但是元祖定义一个只有一个元素的元祖时，必须加上逗号。**

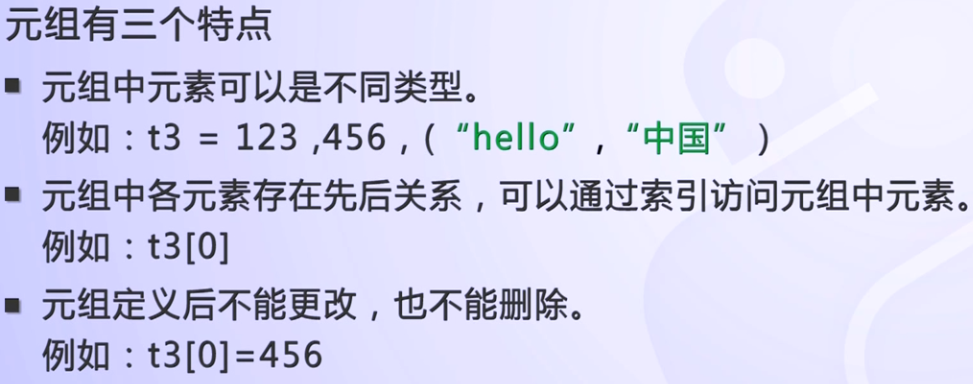


元组中可以包含任何数据类型，元组可以包括元组，元组中也可以包括字典。

t2 = 'ABC',123,'BUPT',78.90,{'name':'zh'},(3,'4')  
print(type(t2))#<class 'tuple'>

t1 = 4,5,6  
print(t1)#(4, 5, 6)  
print(type(t1))#<class 'tuple'>  
print(len(t1))#3

## 元组的三个特点



**元组一旦定义不可改变，提高了程序的安全性。**

## 元组支持索引

**和字符串类似，元组tuple**也支持索引功能。同样有**正索引和负索引，及范围提取**。

t2 = 'ABC',123,'BUPT',78.90,{'name':'zh'},(3,'4')  
print(t2[2])#BUPT  
print(t2[-1])#(3, '4')  
print(t2[0:])#('ABC', 123, 'BUPT', 78.9, {'name': 'zh'}, (3, '4'))  
print(t2[0:-1])#('ABC', 123, 'BUPT', 78.9, {'name': 'zh'})

## 元组之间可以利用+号和\*号进行运算

和字符串类似，元组之间也可以利用+号和\*号进行运算。

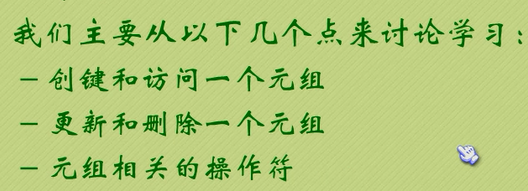
t1 = 1,2,'AA',('A','B')  
t2 = 'BUPT','University',66  
t3 = t1+t2  
print(t1)#(1, 2, 'AA', ('A', 'B'))  
print(t2)#('BUPT', 'University', 66)  
print(t3)#(1, 2, 'AA', ('A', 'B'), 'BUPT', 'University', 66)  
print(t1\*2)#(1, 2, 'AA', ('A', 'B'), 1, 2, 'AA', ('A', 'B'))

## 判断一个元素是否在元组中in或not in

print(1 in t1)#True

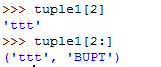
## 元组的主要方法介绍

**元祖和列表的操作很相近，学习元祖可以用类比的方法学习。**



1. **虽然定义时，元祖用小括号，但是调用某个元素时仍然用中括号。**





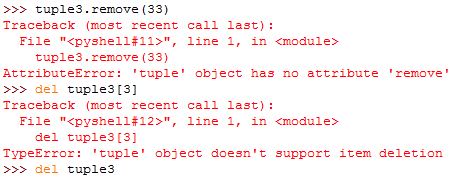
1. **元组的复制，和list列表一样**



1. **元祖之间的拼接**



1. **元祖没有删除某个元素的功能，没有remove、pop方法，del也只能删除整个元祖，不能删除元祖中的某个元素。**



# 列表类型

## 列表概念





C语言或Java中的数组只可以存放同种类型的数据，但是

**Python的列表中可以同时存放多种类型的数据。**



## 列表支持索引功能

list1 = ['34',4,'qw',(3,4),{'name':'zh'}]  
print(list1[-1])#{'name': 'zh'}  
print(list1[0:])#['34', 4, 'qw', (3, 4), {'name': 'zh'}]  
print(list1[0:-1])#['34', 4, 'qw', (3, 4)]

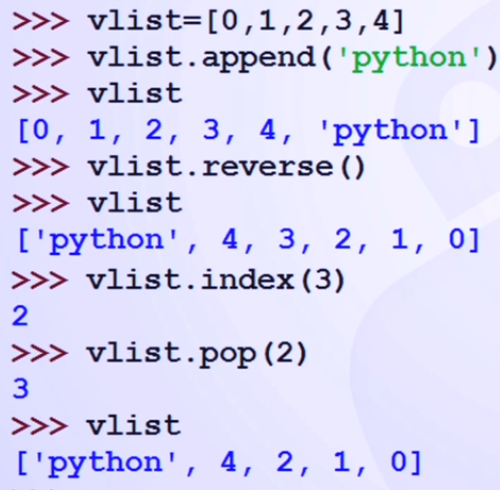
## 可以利用+号和\*号对列表进行操作

list2 = [3,'A']  
list3 = ['BUPT',43]  
list4 = list2+list3  
print(list2)#[3, 'A']  
print(list3)#['BUPT', 43]  
print(list4)#[3, 'A', 'BUPT', 43]  
print(list3\*3)#['BUPT', 43, 'BUPT', 43, 'BUPT', 43]

## 对列表的操作：常用的方法



示例：

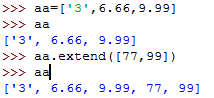


### append(value…)方法：





### extend(列表)方法：将一个列表追加到另外一个列表末尾。

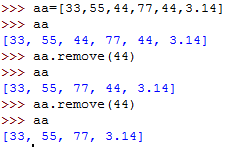


### insert(index,value)方法：



### remove、pop、del：删除列表中的某个元素：三个方法

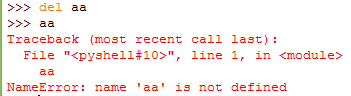
**（1）根据元素值删除某个元素：remove（value）**



**（2）del语句：del不是列表的一个方法，del是一个语句。**

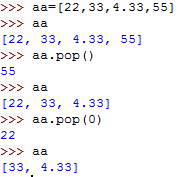


**也可以删除整个列表。**



**（3）pop方法：**

**取出索引对应的元素值，并在列表中删除这个元素。**



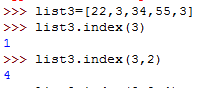
**不指定索引时，默认是最后一个元素。**

### count方法：



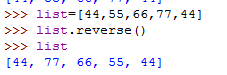
### index方法：根据value查找索引。

**可以直接从索引0处开始查询，也可以指定从某个索引处查询，也可以设置在某个范围内进行查询。**





### reverse()方法：倒置。



### sort()方法：

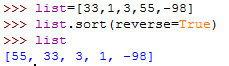


**默认从小到大进行排序。**

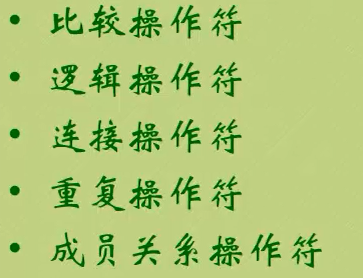
**如何进行从大到小进行 排序呢？**

**可以先利用sort()方法，再调用reverse()方法。**

**也可以直接进行从到小的排序：**



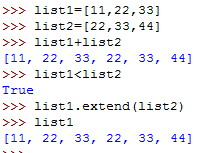
## 列表的操作符：



**（1）可以比较大小：**

**（2）可以进行相加：即两个列表的拼接，不过最好用extend方法。**

**但是，列表之间不能相减或相除或相乘。**

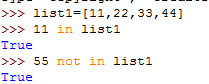


**（3）列表可以乘以大于0的数字，功能就是进行复制拓展。称为重复操作符。**

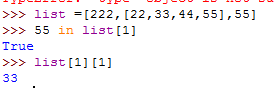




**（4）判断某个元素是否存在于列表中：利用in 或 not in 判断。**

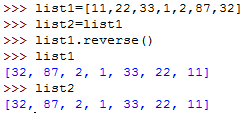


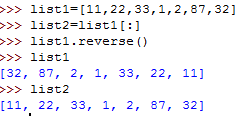
## 列表中可以存放列表：构成了二维列表。



**二维列表：如同C语言中的二维数组一样。**

## 注意列表的拷贝问题：



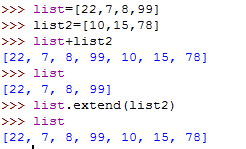


**list2=list1 只是变量指向了同一个位置，但是实际列表还是一个。**

**而list2=list[:] 是创建了一个新的列表。**

**同时，注意 + 和extend的区别**

**利用extend确实是加到了列表中，而用+号并没有加到列表中。**



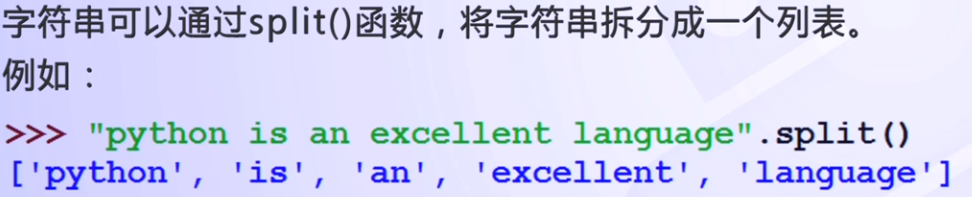
## 对列表的遍历

for value in list3:  
 print(value)

## 判断一个元素是否在列表中：in

print('BUPT' in list3)#True

## 字符串通过split()函数拆分成一个列表。



# 元组tuple与列表list的区别

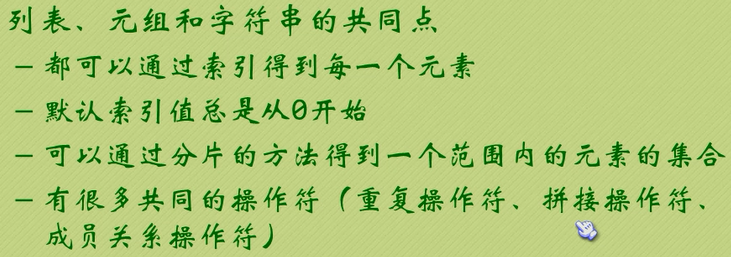
## 元组利用()定义，而列表利用[]定义；

## 元组中的元素的值不可以修改，而列表中的元素的值可以修改。

# 序列：列表、元组、字符串

## 序列的概念：

包括 列表、元组、字符串等。



**了解关于list、tuple、str等的方法，可以利用help(xxxxx)。**

## 序列一般都是可以迭代的。

## 序列之间的转化：

list()

tuple()



## 序列的通用方法：

### len方法

### max方法





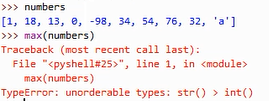
也可以传入字符串：



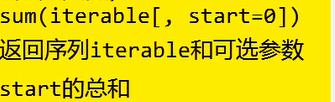


**注意：在一个序列中的数据类型必须一致时，才可以使用此方法，否则报错。**

如：

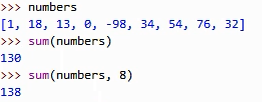


### sum方法：



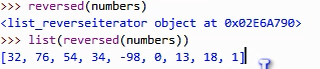
可选参数就是另外需要相加的参数。



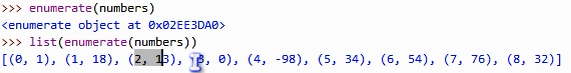


### sorted()方法

### reversed()方法：



### enumerate



### zip



# 字典类型dict

单独文档介绍。

# 文件类型（单独介绍）

单独文档介绍。