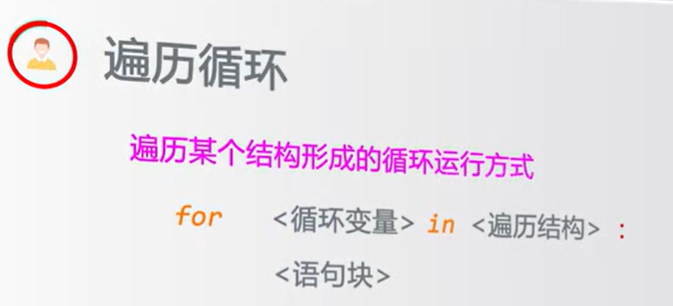
遍历结构



应用：

1计数循环

For I in range（M,N,K）

<语句块>

遍历由range（）函数产生的数字序列，产生循环

举例：

>>>for I in range(1,6,2):

Print(“hello:”,i)

2.字符串遍历循环

For c in s

<语句块>

举例：

>>>For c in “python123”

Print(c,end=”,”)

3.列表遍历循环

For item in ls

<语句块>

Ls是一个列表，遍历其每个元素，产生循环

举例：

>>>for item in [123,”PY”,456]:

Print(item,end=”,”)

4.文件遍历循环

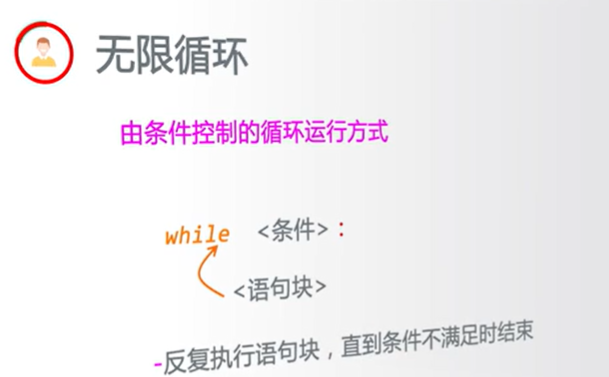
For line in fi：

<语句块>

Fi是一个文件标识符，遍历其每行，产生循环

5……

无限循环



应用：

无限循环的条件

>>>a=3

>>>while a>0:

a=a-1

print(a)

ctrl+c跳出循环

循环控制保留字

Break和continue

Break跳出并结束当前整个循环，执行循环后的语句

Continue结束档次循环，继续执行后续次数循环

Break和continue可以与for和while循环搭配使用

举个例子：

>>>for c in “PYTHON”：

If c ==“T”：

Continue

Print（c，end=“”）

执行结果：PYHON

(没有T了)

>>>for c in “PYTHON”：

If c ==“T”：

Break

Print（c，end=“”）

执行结果：PY

(没有THON了)

再举个例子

>>>s=”PYTHON”

>>>while s !=””:

For c in s:

Print(c,end=””)

s=s[:-1]

输出结果是

PYTHONPYTHOPYTHPYTPYP

会发现，每次都少一个字母

>>>s=”PYTHON”

>>>while s !=””:

For c in s:

If c ==“T”

BREAK

Print(c,end=””)

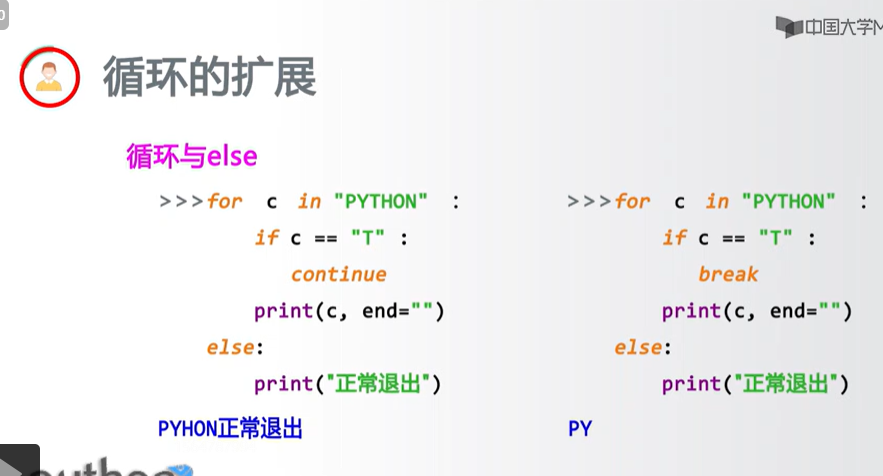
s=s[:-1]

PYPYPYPYPYP

会发现，BREAK每次都跳出当次循环

**循环的高级用法**

****



**Radom库使用**

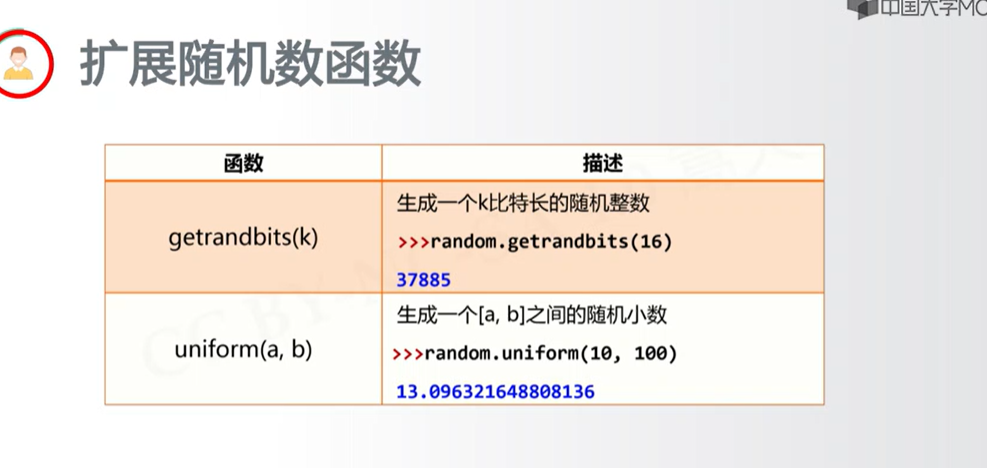
基本随机函数：seed（），random（）

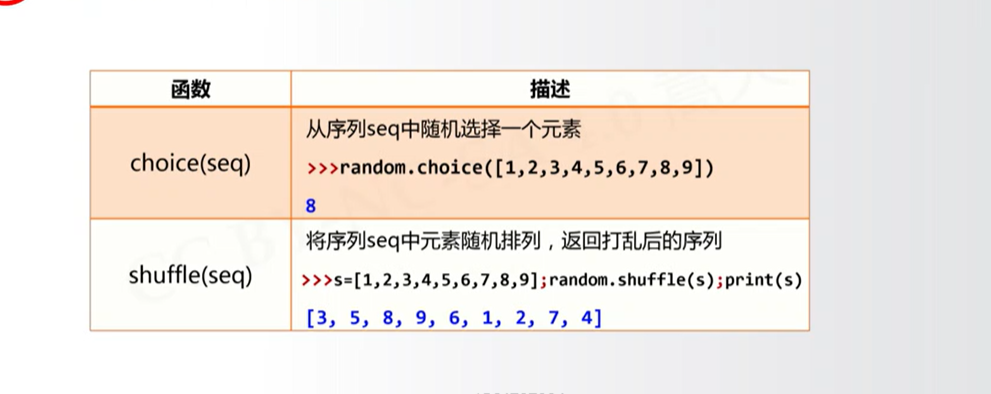
扩展随机函数：randint（）getrandbits（），uniform（），randrange（）

Choice（）shuffle（）









**圆周率计算**

见练习题的附件

**函数和代码复用**



5.1函数的定义与使用

函数的理解与定义

函数的使用及调用过程

函数的参数传递

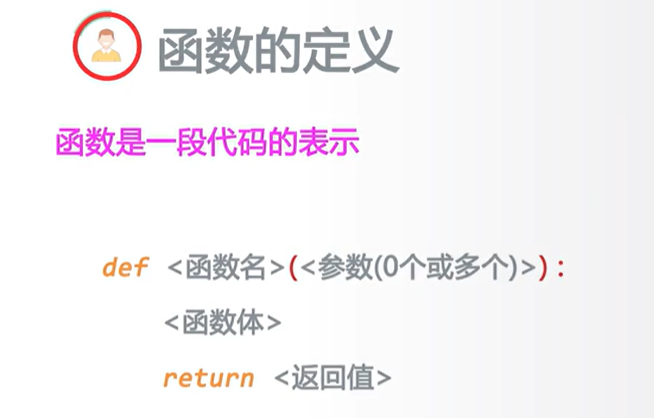
函数的返回值

局部变量和全局变量

Lambda函数

函数的理解与定义

函数就是有功能的东西，可以调用



举个例子，计算n！（n的阶乘）

Def fact（n）

S = 1

For I in range（1，n+1）：

S \*= i

Return s

函数的使用及调用过程

调用就是运行函数的方式

函数的参数传递

参数，有，无，不重要，一定要有括号！！！

函数的返回值

return可以有，可以无

局部变量和全局变量

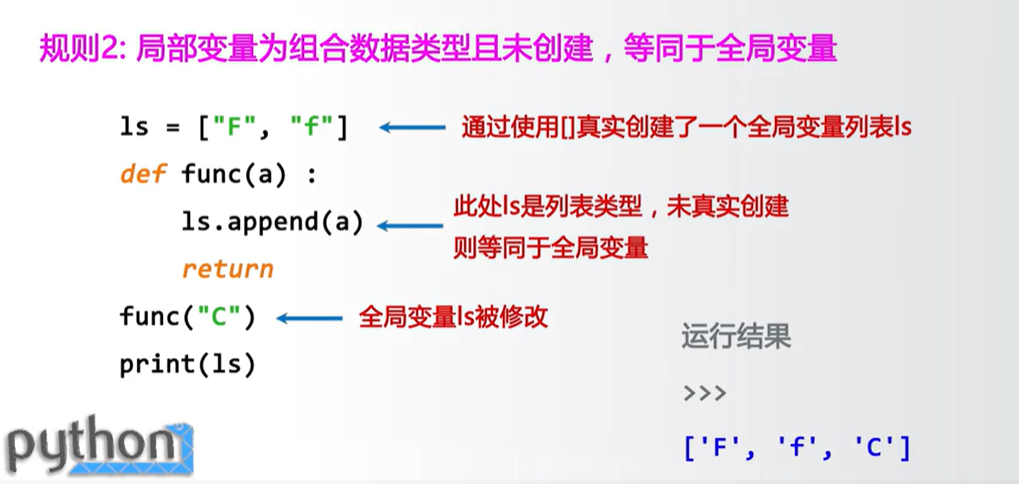
1. 局部变量就是，函数里面的一个参数和外面的一个参数重名了，

即使名字相同不是同一个变量

局部变量只在函数里面有效

要是想变成全局变量，加global，声明为全局变量。

2.局部变量如果是组合数据类型，未在函数中创建，它，就是全局变量



Lambda函数



举个例子

>>>f=lambda x,y : x+y

>>>f(10,15)

25

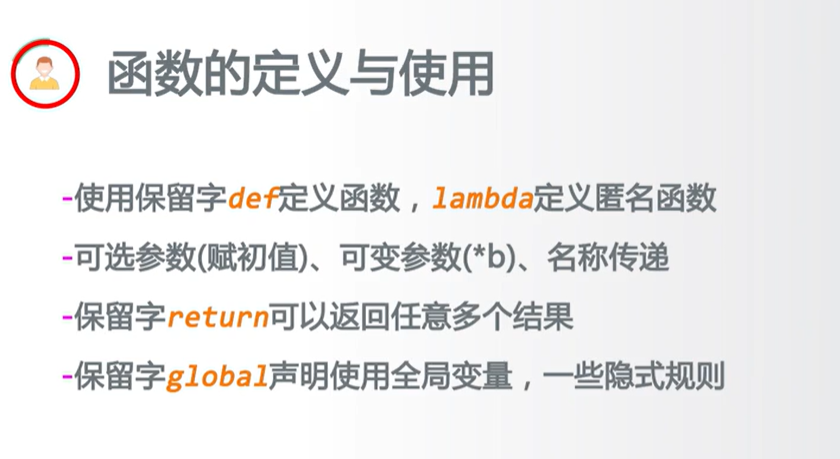
谨慎使用lambda函数，

主要用作特定函数or方法，

有一些固定的使用方式，建议逐步掌握

一般用def定义普通函数

函数小结



代码复用与函数递归

1.代码复用与模块化设计

2.函数递归的理解

3.函数递归的调用过程

4.函数递归实例解析

1.代码复用与模块化设计

把代码当成资源进行抽象

同一份代码在需要时可以被重复使用

函数，代码命名抽象化、

对象，有属性和方法

通过封装，进一步抽象，功能设计，达到模块化设计

紧耦合（部分间交流密切，无法独立）＆松耦合（部分间交流松散，可独立）

2.递归

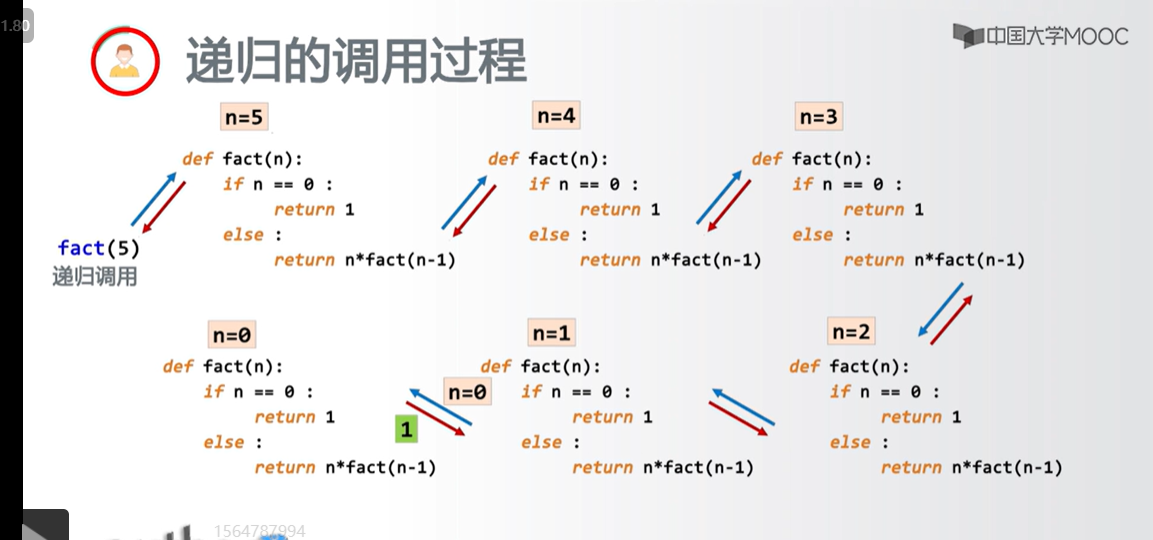
函数调用函数自身

链条，计算过程存在递归链条、

基例，存在一个或多个不需要再次递归的基例

3.函数递归的调用过程

函数+分支语句



4.函数递归实例解析

字符串反转[::-1]

用函数封装的形式，

Def rvs（s）

If s==“”：

Return s

Else：

Return rvs（s[1:]+s[0]）

斐波那契数列

F(n)=F（n-1）+F（n-2）

Def f（n）

If n==1 or n==2：

Return 1

Else：

Return f（n-1）+f（n-2）

汉诺塔问题

Count = 0

Def Hanoi（n，src，dst，mid）：

Global count

If n==1 ：

Print（“{}：{}->{}”.format（1，src，dst））

Count +=1

Else：

Hanoi（n-1，src，mid，dst）

Print（“{}：{}->{}”.format(n,src,dst)）

Count +=1

Hanoi(n-1,mid,dst,src)

Pyinstaller库的使用说明



Pyinstaller –F<文件名.py>

**第6章，组合数据类型**

**6.1集合类型及操作**

集合就是集合，数学里的那个集合，多个元素无序组合

无序，唯一，不存在相同元素

集合不可更改，不是可变数据类型

非可变数据类型：整数，浮点数，复数，字符串类型，元组类型··········~

建立集合，用大括号{}，集合里面的元素用，（逗号）隔开

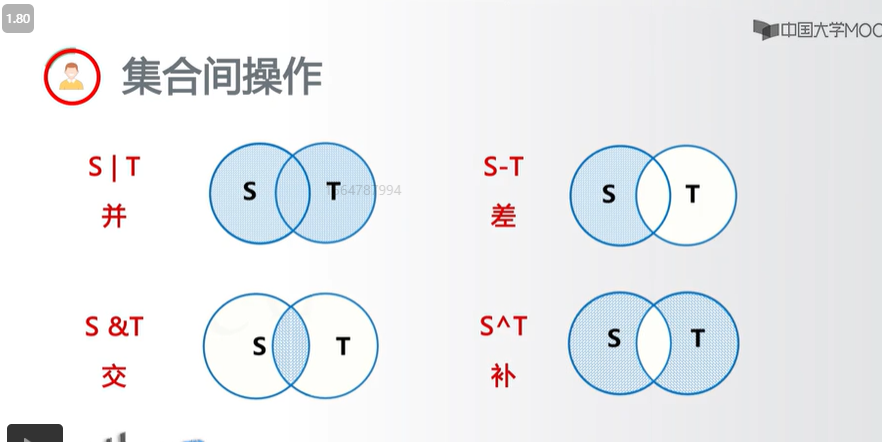
或者，set{}建立集合。

空集必须是set{}

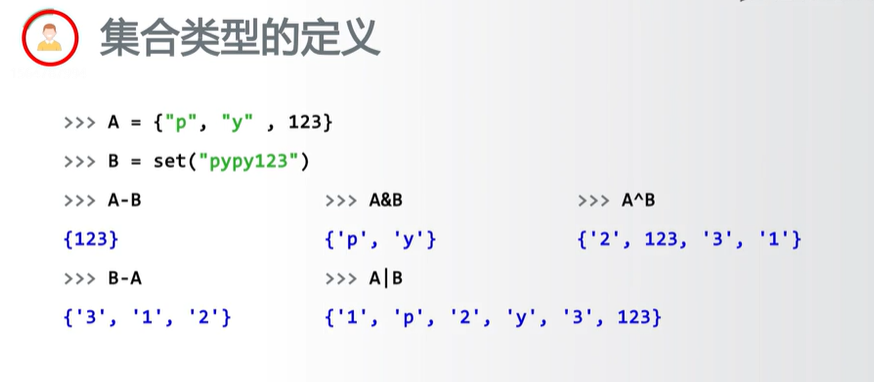
小括号（）里面的是元组

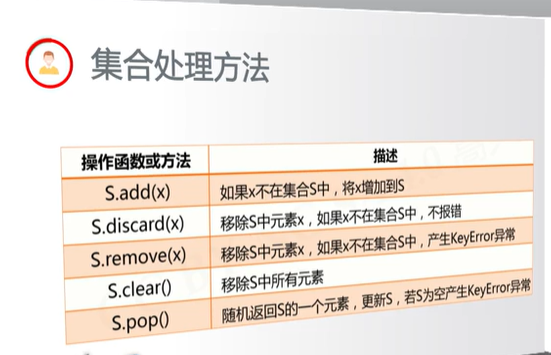
集合间操作

交叉并补





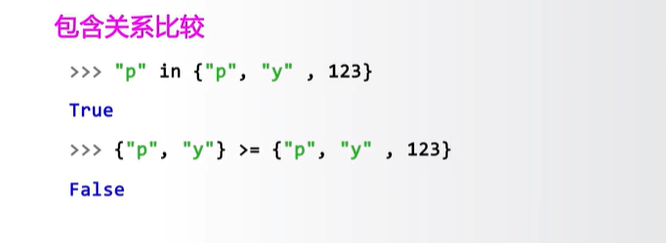




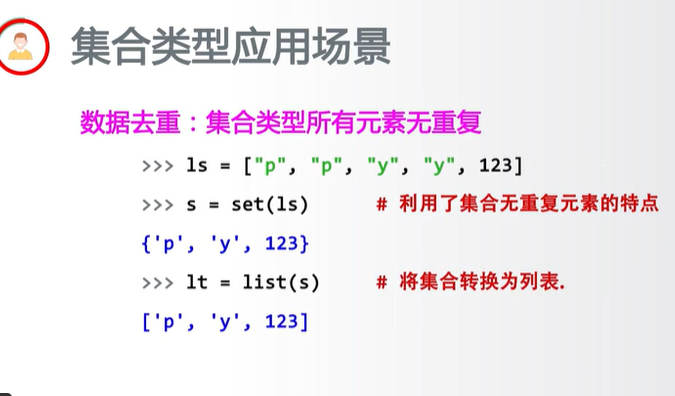


集合运用场景

包含关系比较



**数据去重**

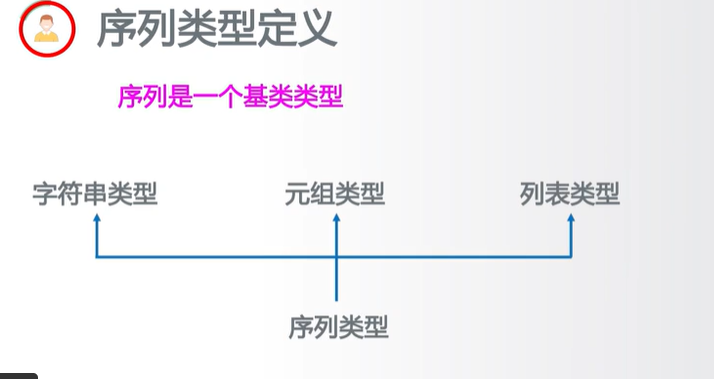


**序列类型操作**

****

序列类型定义

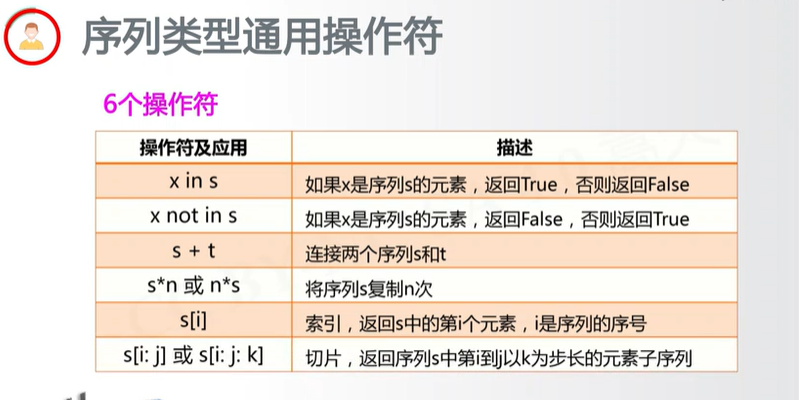
序列，具有先后关系的一组元素，一维元素的向量

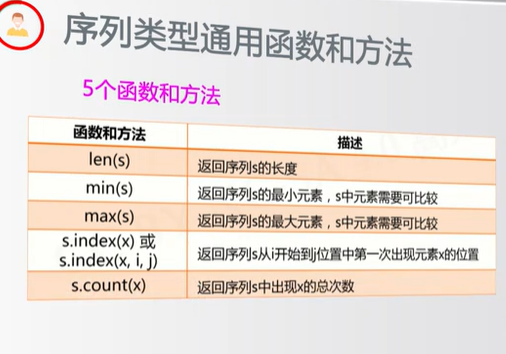


正向递增

反向递减

序列处理函数及方法



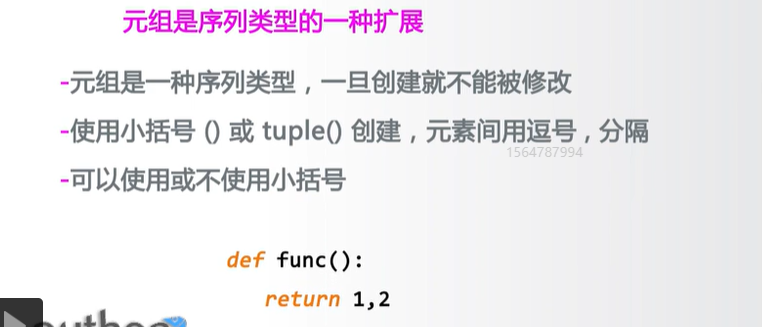


元组类型及操作

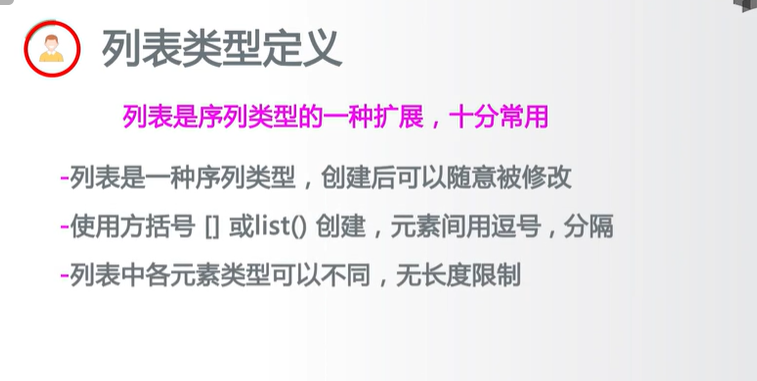
元组，序列类型的一种拓展

元组类型继承序列类型的全部操作

元组创建后不能修改，没有特殊操作

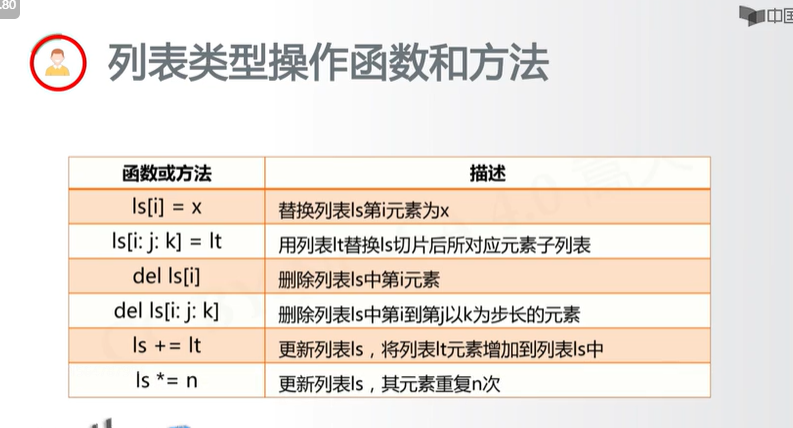


列表类型及操作

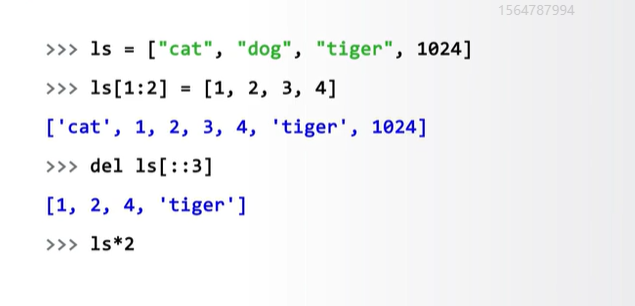


用赋值的话，没有创建新列表，还是原来那个列表。相当于重新命名一次

列表操作函数及方法

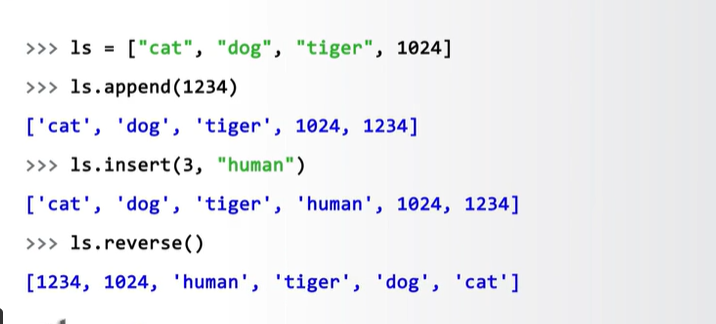


举例

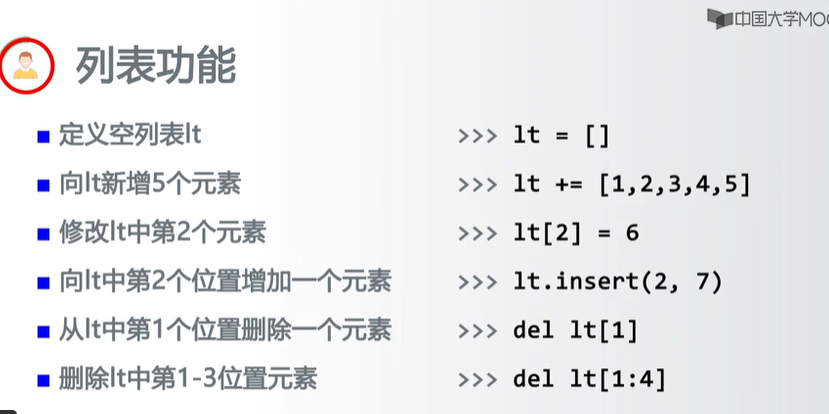


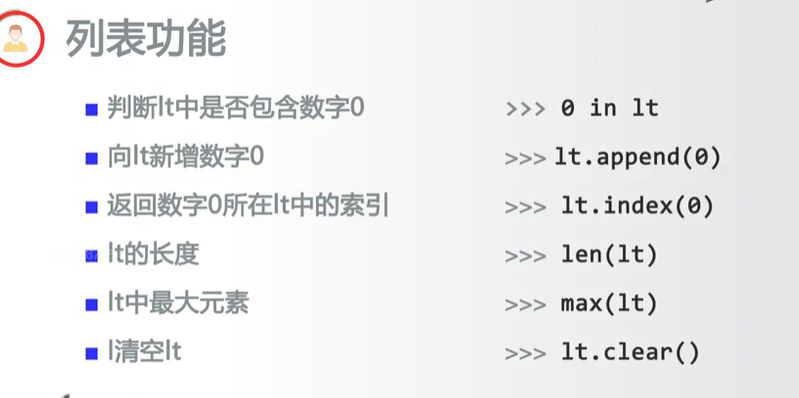


举个例子

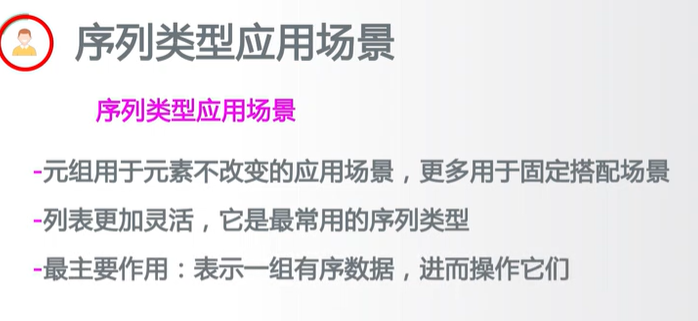


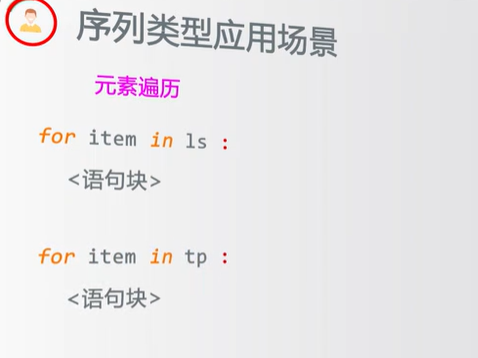
举个例子





序列类型应用场景





Ls是列表类型，遍历

Tp是元组类型，遍历

元组是不可改的，so元组，用来，数据保护

Tuple（）列表变为元组



应用，基本统计值

