**eval()**

**格式：**

eval (source, global s = None, locals = None, /)

**功能：**

将source当作一个python表达式进行解析和计算，返回计算 结果。

**参数说明：**

source是一个字符串，这个字符串能表示成Python表达式，或者是能够通 过编译的代码；globals是可选的参数，默认为None,如果设置属性不为None的话,就必须 是dictionary对象；locals也是可选的参数，默认为None,如果设置属性不为None的话，可 以是任何map对象。

>> x= 3

>> eval( \*x+ 1')

>> eval( \*3 + 5')

8

>> eval( \*[1,2,3] \*)

[1, 2, 3]

**Iterator()**

注：isinstance( 对象1, Iterator)，可用来判断对象1是否能迭代

(有\_\_iter\_\_方法的对象)（from collections.abc import Iterable）

>>vehicle = [ 'train', 'bus', 'car', 'ship']

>> v = iter (vehicle)#构造vehicle对象的迭代器

>>type( v)

< class ' 1ist\_iterator '>

>>is instance (v, Iterator)

True

>>v.\_\_next\_\_()

'train'

>> v.\_\_next\_\_()

'bus'

**相关内置函数**

**enumerate()**

**格式：**

enumerate(iterable, start = 0)

**功能：**

返回下标和值的enumerate对象。第1个参数表示可迭代(Iterable)对象，第2 个参数表示下标的开始值，默认从0开始。(这里还要补充说明的是list()、tuple()、dict()、set()函数能将可迭代(Iterable)对象转 换为列表、元组、字典、集合。)**迭代**

>>vehicle = [ 'train','bus', 'car', 'ship']

>>vvl = enumerate(vehicle,1)

>>vv1.\_\_next\_\_()

(1, 'train')

>>for i in enumerate(vehicle) : # 遍历 enumerate 对象中 的元素

print(i,end= '')

(0, 'train')(1, 'bus')(2, 'car')(3, 'ship')

>>for i,x in enumerate(vehicle) : # 遍历enumerate对象元素的下标和值

print(i,x,end= ' ')

0 train 1 bus 2 car 3 ship

**zip()**

**格式：**

zip( iterl [, iter2 [•••]])

**功能：**

将多个迭代器(Iterator)对象(或者可迭代(Iterable)对象)中的元素压缩到一起, 返回一个zip对象。

同样通过帮助我们得知zip对象既是一个可迭代(Iterable)对象也是一个迭代器 (Iterator)对象。(不同长短，匹配完短的结束)**压缩**

>>vehicle = [ 'train','bus', 'car', 'ship']

>>wl = zip( 'abcd', vehicle)

>>list(wl)

[('a', 'train'), ('b', 'bus'), ('e', 'car'), ('d', 'ship')]

**map()**

**格式：**

map(func, \* iterables)

**功能：**

把一个函数func依次映射到可迭代(Iterable)对象的每个元素上，返回一个map 对象。同样通过帮助我们得知map对象既是一个可迭代(Iterable)对象也是一个迭代器 (Iteralor)对象。**计算**

>>aa= [ '1', '5.6','7.8', '9']

>>bbl = map(float, aa)

>>bbl

<map object at 0x0000026970C9AC40>

>>list (bbl)

[1.0, 5.6, 7.8, 9.0]

>>list (map (str, range(5)))

['0', '1', '2', '3', '4']

>>def cc(x, y):

return x\*\* 2 + y \*\* 2

>>list(map(cc, [ 1, 2], (2,3)))

[5, 13]

>>list (map (lambda x,y:x\*\* 2 + y \*\* 2, [1,2], (2,3)))

[5, 13]

**filter()**

**格式:**

filter(function or None, iterable)

**功能：**

把一个带有一个参数的函数function作用到一个可迭代(Iterable)对象上，返回 一个filter对象,filter对象中的元素由可迭代(Iterable)对象中使得函数function返回值为 True的那些元素组成；如果指定函数为None,则返回可迭代(Iterable)对象中等价于True 的元素。**过滤**

>>aa= [5,6, - 9, - 56, - 309,206]

>>def func(x):

return x% 2!= 0

>>bb = filter( func, aa)

>>type(bb)

<class 'filter'>

>>list(bb)

[5, -9, -309]

>>cc = filter(lambda x:x % 2!= 0,aa)

>>list(cc)

[5, -9, -309]

>>[x for x in aa if x% 2!= 0]

[5, -9, -309]

>>dd= [6, True, 1,0, False]

>>ee = filter(None, dd)

>>list(ee)

[6, True, 1]