zip() 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。

如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同，利用 \* 号操作符，可以将元组解压为列表。

>>> a =[1,2,3]

>>> b =[4,5,6]

>>> c =[4,5,6,7,8]

>>> zipped =zip(a,b)# 打包为元组的列表

[(1,4),(2,5),(3,6)]

>>>zip(a,c)# 元素个数与最短的列表一致

[(1,4),(2,5),(3,6)]

>>>zip(\*zipped)# 与 zip 相反，\*zipped 可理解为解压，返回二维矩阵式

[(1,2,3),(4,5,6)]

>>> a = [1,2,3]

>>> b = [4,5,6]

>>> c = [4,5,6,7,8]

>>> zipped = zip(a,b)     # 返回一个对象

>>> zipped

<zip object at 0x103abc288>

>>> list(zipped)  # list() 转换为列表

[(1, 4), (2, 5), (3, 6)]

>>> list(zip(a,c))              # 元素个数与最短的列表一致

[(1, 4), (2, 5), (3, 6)]

>>> a1, a2 = zip(\*zip(a,b))          # 与 zip 相反，zip(\*) 可理解为解压，返回二维矩阵式

>>> list(a1)

[1, 2, 3]

>>> list(a2)

[4, 5, 6]

>>>

map() 会根据提供的函数对指定序列做映射。

第一个参数 function 以参数序列中的每一个元素调用 function 函数，返回包含每次 function 函数返回值的新列表。

>>> def square(x) :            # 计算平方数

...     return x \*\* 2

...

>>> map(square, [1,2,3,4,5])   # 计算列表各个元素的平方

[1, 4, 9, 16, 25]

>>> map(lambda x: x \*\* 2, [1, 2, 3, 4, 5])  # 使用 lambda 匿名函数

[1, 4, 9, 16, 25]

# 提供了两个列表，对相同位置的列表数据进行相加

>>> map(lambda x, y: x + y, [1, 3, 5, 7, 9], [2, 4, 6, 8, 10])

[3, 7, 11, 15, 19]

>>> def square(x) :         # 计算平方数

...     return x \*\* 2

...

>>> map(square, [1,2,3,4,5])    # 计算列表各个元素的平方

<map object at 0x100d3d550>     # 返回迭代器

>>> list(map(square, [1,2,3,4,5]))   # 使用 list() 转换为列表

[1, 4, 9, 16, 25]

>>> list(map(lambda x: x \*\* 2, [1, 2, 3, 4, 5]))   # 使用 lambda 匿名函数

[1, 4, 9, 16, 25]

>>>