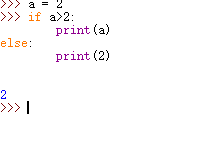
1. python shell里的缩进问题：

if 语句前’>>>’不算缩进，else前要不留空格

1. python 没有自加、自减之说,’i++’会报错
2. python random.randint(a,b)返回在[a,b]之间的数。
3. python怎么把输入的字符串转成列表呢

>>> x=input()

1,2,3,4

>>>print(x)

‘1,2,3,4’

>>> xlist=x.split(",")

>>> print(xlist)

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

>>>xlist = [ int(xlist[i]) for i in range(len(xlist))]

>>> print(xlist)

[1, 2, 3, 4]

5. Python split()方法

描述:

Python split() 通过指定分隔符对字符串进行切片，如果参数 num 有指定值，则仅分隔 num 个子字符串

语法:

str.split(str="", num=string.count(str)).

参数:

str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。

num -- 分割次数。(可缺省)

返回值:

返回分割后的字符串列表。

实例:

str = "Line1-abcdef \nLine2-abc \nLine4-abcd";

print(str.split( ));

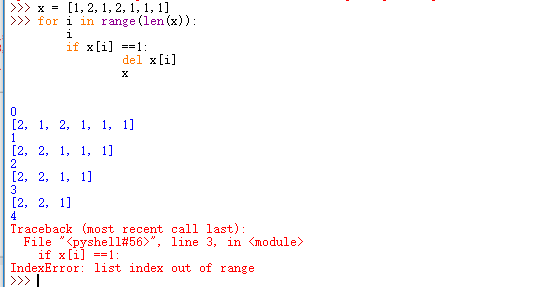
print(str.split(' ', 1 ));

以上实例输出结果如下：

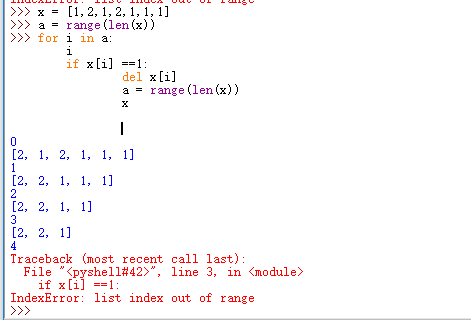
['Line1-abcdef', 'Line2-abc', 'Line4-abcd']

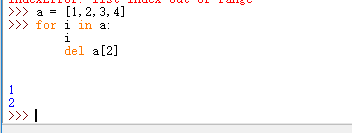
['Line1-abcdef', '\nLine2-abc \nLine4-abcd']

1. Python中对于无需关注其实际含义的变量可以用\_代替，for \_ in range(5)就和for i in range(5)一样，因为这里我们对i并不关心，所以用\_代替仅获取值而已。
2. Python 中for \_ in range(len(x))问题 # x为列表

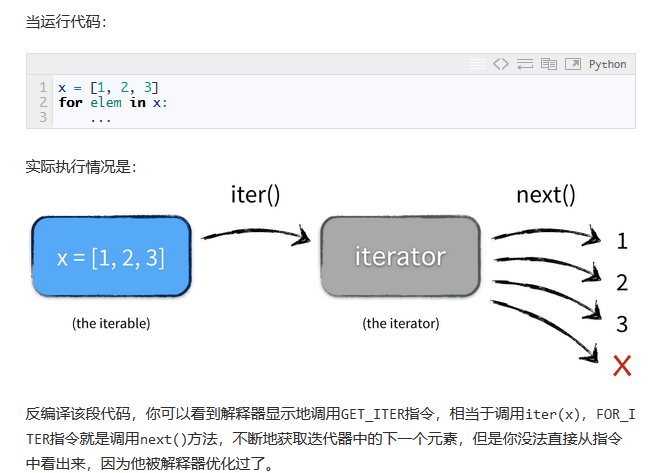


按照我们以前学过的编程逻辑，在x=[2,2,1]后，range(len(x))最多只能迭代到3，作何解释？？？





深层次的原因，



Range()返回一个可迭代对象,但是在图中这个对象没有指向它的引用，虽然这个列表x在变化,但是这个可迭代对象没有变；每执行一次range()就返回一个迭代对象,i要先迭代完第一个对象

a就是一个可迭代对象

8. eval(str)函数很强大，官方解释为：将字符串str当成有效的表达式来求值并返回计算结果。所以，结合math当成一个计算器很好用。

eval()函数常见作用有：   
1、计算字符串中有效的表达式，并返回结果

>>> eval('pow(2,2)')

4

>>> eval('2 + 2')

4

>>> eval("n + 4")

85

2、将字符串转成相应的对象（如list、tuple、dict和string之间的转换）

>>> a = "[[1,2], [3,4], [5,6], [7,8], [9,0]]"

>>> b = eval(a)

>>> b

[[1, 2], [3, 4], [5, 6], [7, 8], [9, 0]]

>>> a = "{1:'xx',2:'yy'}"

>>> c = eval(a)

>>> c

{1: 'xx', 2: 'yy'}

>>> a = "(1,2,3,4)"

>>> d = eval(a)

>>> d

(1, 2, 3, 4)

3、将利用反引号转换的字符串再反转回对象

>>> list1 = [1,2,3,4,5]

>>> `list1`

'[1, 2, 3, 4, 5]'

>>> type(`list1`)

<type 'str'>

>>> type(eval(`list1`))

<type 'list'>

>>> a = eval(`list1`)

>>> a

[1, 2, 3, 4, 5]

1. Python序列解包,

python中的解包可以这样理解：一个list是一个整体，想把list中每个元素当成一个个个体剥离出来，这个过程就是解包.

假如一个字符串'ABCDEFGH'，要输出下列格式:

A ['B', 'C', 'D', 'E', 'F', 'G', 'H']

B ['C', 'D', 'E', 'F', 'G', 'H']

C ['D', 'E', 'F', 'G', 'H']

D ['E', 'F', 'G', 'H']

E ['F', 'G', 'H']

F ['G', 'H']

G ['H']

H []

即: 每次取出第一个作为首，然后的字符串拆成列表，放置在后面，最后成上面的输出:

一般的处理是:

>>> s = 'ABCDEFGH'

>>> while s:

front, s = s[0], list(s[1:])

print(front, s)

A ['B', 'C', 'D', 'E', 'F', 'G', 'H']

B ['C', 'D', 'E', 'F', 'G', 'H']

C ['D', 'E', 'F', 'G', 'H']

D ['E', 'F', 'G', 'H']

E ['F', 'G', 'H']

F ['G', 'H']

G ['H']

H []

备注:

1.将切片中索引为0的字符赋值给front

2.将切片中索引为1之后字符再赋值给s

3.用list函数将字符串转变为列表

4.用while循环来s来判断，为空，则退出循环

上面的处理，可以用序列解包的方法会来处理，并好理解。 序列解包是Python 3.0之后出现，之前的版本一般赋值的时候，一定要对等，才能正常赋值，比如说:

>>> a, b, c = (1, 2, 3)

>>> a, c

(1, 3)

>>> [a, b, c] = (1, 2, 3)

>>> a, c

(1, 3)

>>> a, b, c = 'SON' # 刚好三个字符的字符串

>>> a, c

('S', 'N')

如果是不对等的情况下，是会报错的: too many values to unpack

>>> a, b = 'SON'

Traceback (most recent call last):

File "<pyshell#42>", line 1, in <module>

a, b = 'SON'

ValueError: too many values to unpack (expected 2)

[python] view plain copy

3.0出现了解包，一切就简单多了，如果我不之后字符有多少个时候，都可以如此:

>>> a, \*b = 'BOOK'

>>> a, b

('B', ['O', 'O', 'K'])

>>> \*a, b = 'BOOK'

>>> a, b

(['B', 'O', 'O'], 'K')

上面就是序列解包，在赋值时无疑更方便，适用性更强！运用序列解包的功能重写上面的代码:

从上面可以看出: 代码更简洁，也更好理解了！

More about Python解包：[Python解包压包](http://www.baidu.com/link?url=2IHOk5ja-TU8P3EO6c2FPTHJqfGL09eJpwDaG3oJVuOUlcHAGaeKRe-nLcJ45o5lr6AxGZIhN-F6go9_MoxFtsup12G0ispsnCU5VbUYcTO&ck=2400.7.120.242.251.335.251.368&shh=www.baidu.com&sht=monline_dg&wd=&eqid=9b325c630002e620000000065aed40a2)

10.python格式化输出：[python格式化输出](http://www.baidu.com/link?url=07tKQzRCbUv1F9uI2U2ChJ9in3SVGATlXqjFwPrFHChoonwdClV7SygagRB3JgtUoboq6GPwgtg1-rC3ifd99q&ck=1268.26.84.301.149.498.139.534&shh=www.baidu.com&sht=monline_3_dg&wd=&eqid=b95def81000061ba000000065aed66d4)