目录：

**一、查看第三方库的版本**

**二、几种取整方式：**int round **math**.ceil

**三、几个特殊函数的应用：**zip enumerate

**四、生成随机数**

**五、**[**Python中\*args 和\*\*kwargs的用法**](http://blog.csdn.net/chenjinyu_tang/article/details/8136841)

**六、Python3 中出现的错误**

**七、Python3 用户输入**

**八、Python3 中np.loadtxt**

**九、**[**Python中is和==的区别**](http://blog.csdn.net/imzoer/article/details/8637408)

**一、查看第三方库的版本**

# Python version
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.\_\_version\_\_))
# numpy
import numpy
print('numpy: {}'.format(numpy.\_\_version\_\_))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.\_\_version\_\_))
# pandas
import pandas
print('pandas: {}'.format(pandas.\_\_version\_\_))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.\_\_version\_\_))

**二、几种取整方式**

原文  [http://www.csuldw.com/2015/12/04/2015-12-04-Python-Round/](http://www.csuldw.com/2015/12/04/2015-12-04-Python-Round/?utm_source=tuicool&utm_medium=referral)

**背景**

在处理数据的时候，碰到了一个问题，就是取整方式！比如给定一个数值型列表，我需要分别获取它位置为长度的0%,25%,50%,75%,100%处的几个数字。但Python自带的 int 是向下取整，如果数字长度是5，理论上这五个数字分别对应0%,25%,50%,75%,100%的位置，但使用 int ，结果却并不是入次。比如当 5\*0.75 时,如果加上 int(5\*0.75) ，就等于 3 ，而我想要的应该是4，显然不是我想要的，所以这里需要用到向上取整方式。因此，顺便总结了一下Python的几种取整方式。

**取整方式**

下面介绍几种常用的取整方法，包括向下取整、四舍五入、向上取整。

**（1）向下取整**

向下取整很简单，直接使用int()函数即可，如下代码(Python 2.7.5 IDLE)

>>> a = 3.75

>>> int(a)

3

**（2）四舍五入**

第二种就是对数字进行四舍五入，具体的看下面的代码：

>>> a=3.25;b=3.75

>>> round(a);round(b)

3.0

4.0

**（3)向上取整**

但三种，就是向上取整，也就是我这次数据处理中需要的，由于之前没在Python中用到过，所以不太熟悉，其实Python的math中就带了向上取整的函数，即 ceil 方法，专门用于向上取整，实例如下：

>>> **import** **math**

>>> **math**.ceil(3.25)

4.0

>>> **math**.ceil(3.75)

4.0

**三、几个特殊函数的应用**

**1 zip**

与序列有关的内建函数有：sorted()、reversed()、enumerate()、zip()),

其中sorted()和zip()返回一个序列(列表)对象，

reversed()、enumerate()返回一个迭代器(类似序列)

zip([seql, ...])接受一系列可迭代对象作为参数，将对象中对应的元素打包成一个个tuple（元组），然后返回由这些tuples组成的list（列表）。若传入参数的长度不等，则返回list的长度和参数中长度最短的对象相同。

1 >>> name=('jack','beginman','sony','pcky')
2 >>> age=(2001,2003,2005,2000)
3 >>> for a,n in zip(name,age):
4 print a,n
5
6 输出：
7 jack 2001
8 beginman 2003
9 sony 2005
10 pcky 2000

1 >>> z1=[1,2,3]
2 >>> z2=[4,5,6]
3 >>> result=zip(z1,z2)
4 >>> result
5 [(1, 4), (2, 5), (3, 6)]
6 >>> z3=[4,5,6,7]
7 >>> result=zip(z1,z3)
8 >>> result
9 [(1, 4), (2, 5), (3, 6)]
10 >>>

zip()配合\*号操作符,可以将已经zip过的列表对象解压

1 >>> zip(\*result)
2 [(1, 2, 3), (4, 5, 6)]

\* 二维矩阵变换（矩阵的行列互换）
比如我们有一个由列表描述的二维矩阵
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
通过python列表推导的方法，我们也能轻易完成这个任务
print [ [row[col] for row in a] for col in range(len(a[0]))]
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
另外一种让人困惑的方法就是利用zip函数：
>>> a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> zip(\*a)
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
>>> map(list,zip(\*a))
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
zip函数接受任意多个序列作为参数，将所有序列按相同的索引组合成一个元素是各个序列合并成的tuple的新序列，新的序列的长度以参数中最短的序列为准。另外(\*)操作符与zip函数配合可以实现与zip相反的功能，即将合并的序列拆成多个tuple。
①tuple的新序列
>>>>x=[1,2,3],y=['a','b','c']
>>>zip(x,y)
[(1,'a'),(2,'b'),(3,'c')]
②新的序列的长度以参数中最短的序列为准.
>>>>x=[1,2],y=['a','b','c']
>>>zip(x,y)
[(1,'a'),(2,'b')]
③(\*)操作符与zip函数配合可以实现与zip相反的功能,即将合并的序列拆成多个tuple。
>>>>x=[1,2,3],y=['a','b','c']
>>>>zip(\*zip(x,y))
[(1,2,3),('a','b','c')]

1.zip打包解包列表和倍数
>>> a = [1, 2, 3]
>>> b = ['a', 'b', 'c']
>>> z = zip(a, b)
>>> z
[(1, 'a'), (2, 'b'), (3, 'c')]
>>> zip(\*z)
[(1, 2, 3), ('a', 'b', 'c')]
2. 使用zip合并相邻的列表项
>>> a = [1, 2, 3, 4, 5, 6]
>>> zip(\*([iter(a)] \* 2))
[(1, 2), (3, 4), (5, 6)]
>>> group\_adjacent = lambda a, k: zip(\*([iter(a)] \* k))
>>> group\_adjacent(a, 3)
[(1, 2, 3), (4, 5, 6)]
>>> group\_adjacent(a, 2)
[(1, 2), (3, 4), (5, 6)]
>>> group\_adjacent(a, 1)
[(1,), (2,), (3,), (4,), (5,), (6,)]
>>> zip(a[::2], a[1::2])
[(1, 2), (3, 4), (5, 6)]
>>> zip(a[::3], a[1::3], a[2::3])
[(1, 2, 3), (4, 5, 6)]
>>> group\_adjacent = lambda a, k: zip(\*(a[i::k] for i in range(k)))
>>> group\_adjacent(a, 3)
[(1, 2, 3), (4, 5, 6)]
>>> group\_adjacent(a, 2)
[(1, 2), (3, 4), (5, 6)]
>>> group\_adjacent(a, 1)
[(1,), (2,), (3,), (4,), (5,), (6,)]
3.使用zip和iterators生成滑动窗口 (n -grams)
>>> from itertools import islice
>>> def n\_grams(a, n):
... z = (islice(a, i, None) for i in range(n))
... return zip(\*z)
...
>>> a = [1, 2, 3, 4, 5, 6]
>>> n\_grams(a, 3)
[(1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 6)]
>>> n\_grams(a, 2)
[(1, 2), (2, 3), (3, 4), (4, 5), (5, 6)]
>>> n\_grams(a, 4)
[(1, 2, 3, 4), (2, 3, 4, 5), (3, 4, 5, 6)]
4.使用zip反转字典
>>> m = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
>>> m.items()
[('a', 1), ('c', 3), ('b', 2), ('d', 4)]
>>> zip(m.values(), m.keys())
[(1, 'a'), (3, 'c'), (2, 'b'), (4, 'd')]
>>> mi = dict(zip(m.values(), m.keys()))
>>> mi
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}

**2 enumerate**

我们知道Python里的字典对象的每一个字典项都包含一个关键字和一个值，在使用iteritems()函数进行迭代的时候，会返回(key,value)键值对；但是，我们迭代一个列表的时候，返回的却是value1,value2,vaule3……，我们没有办法知道每一个值对应的位置，尤其是在我们对不同位置的值要做不同处理的时候就更加明显，这个时候我们就要额外引入一个i变量来充当这个位置，看看如下代码：

**for** i **in** range(len(L)):
   item = L[i]

*# ...  ...*
   L[i] = result

是不是很恶心，美国人觉得很odd，于是就发明了这个enumerate函数，于是就变成如下代码：

**for** i, item **in** enumerate(L):

*# ...  ...*
   L[i] = result

可以看到，简洁了很多，优美了许多，enumerate函数返回的是(0,L[0]),(1,L[1]),(2,L[2])……

**四、生成随机数**

numpy.random.randn: 从标准正态分布中返回一个样本值

numpy.random.randint：从在在半开半闭区间[low,high)上的离散均匀分布中返回随机的整数

        值，如果high is None， 那么结果从[0,low)中得到。

numpy.random.normal:  从一个高斯（正态）分布中得到数据

random.random()用于生成一个0到1的随机符点数: 0 <= n < 1.0

random.uniform(a, b)，用于生成一个指定范围内的随机符点数。如果a > b，则生成的随机数n: b <= n <= a。如果 a <b， 则 a <= n <= b。

random.randrange([start], stop[, step])，从指定范围内，按指定基数递增的集合中 获取一个随机数。

random.choice(sequence)。参数sequence表示一个有序类型。这里要说明 一下：sequence在[Python](http://lib.csdn.net/base/python)不是一种特定的类型，而是泛指一系列的类型。list, tuple, 字符串都属于sequence。

random.shuffle(x[, random])，用于将一个列表中的元素打乱。

random.sample(sequence, k)，从指定序列中随机获取指定长度的片断。

rand(d0, d1, ..., dn)    Random values in a given shape.

random.seed(123456789) # 种子不同，产生的随机数序列也不同，随机数种子都是全局种子

random.RandomState(123456789) # 定义局部种子

linspace(start, end, num): 如linspace(0,1,11)结果为[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1];

arange(n): 产生一个从0到n-1的向量，如arange(4)结果为[0,1,2,3]

**五、**[**Python中\*args 和\*\*kwargs的用法**](http://blog.csdn.net/chenjinyu_tang/article/details/8136841)

当函数的参数不确定时，可以使用\*args 和\*\*kwargs，\*args 没有key值，\*\*kwargs有key值。

\*args表示任何多个无名参数，它是一个tuple

\*\*kwargs表示关键字参数，它是一个dict

测试代码如下：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | def foo(\*args,\*\*kwargs): print 'args=',args print 'kwargs=',kwargs print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*' if \_\_name\_\_=='\_\_main\_\_': foo(1,2,3) foo(a=1,b=2,c=3) foo(1,2,3,a=1,b=2,c=3) foo(1,'b','c',a=1,b='b',c='c') |

执行结果如下：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | args= (1, 2, 3) kwargs= {} \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* args= () kwargs= {'a': 1, 'c': 3, 'b': 2} \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* args= (1, 2, 3) kwargs= {'a': 1, 'c': 3, 'b': 2} \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* args= (1, 'b', 'c') kwargs= {'a': 1, 'c': 'c', 'b': 'b'} \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

**六、Python3 中出现的错误**

with open('C:/Users/Administrator/Desktop/data/titanic.csv', 'rb') as csvfile:

解决方案：

For version 3.2 and above

Change: csv\_file\_object.next()

To: next(csv\_file\_object)

then I get another error:

\_csv.Error: iterator should return strings, not bytes (did you open the file in text mode?)

Edit: Figured it out needed to change rb to rt

Finally, it works.

混淆了 python2 里边的 str 和 unicode 数据类型。

1.

对需要 str->unicode 的代码，可以在前边写上

import sys

reload(sys)

sys.setdefaultencoding('utf8')

把 str 编码由 ascii 改为 utf8 (或 gb18030)

2.

python3 区分了 unicode str 和 byte arrary，并且默认编码不再是 ascii

**七、Python3 用户输入**

x = input('What is your name?: ')
print('Hello',x)

**八、Python3 中np.loadtxt**

date,bid,ask = np.loadtxt('py35\_data\sentdex\GBPUSD1d.txt', unpack=True,

delimiter=',',

converters={0:mdates.strpdate2num('%Y%m%d%H%M%S')})

TypeError: strptime() argument 0 must be str, not <class 'bytes'>

Python2 中不报错

Python3 报错

处理方法：

def bytespdate2num(fmt, encoding='utf-8'):

strconverter = mdates.strpdate2num(fmt)

def bytesconverter(b):

s = b.decode(encoding)

return strconverter(s)

return bytesconverter

date,bid,ask = np.loadtxt('py35\_data\sentdex\GBPUSD1d.txt', unpack=True,delimiter=',',

converters={0:bytespdate2num('%Y%m%d%H%M%S')})

**原因**：<https://github.com/matplotlib/matplotlib/issues/4126>

The problem is that numpy passes a byte string to the converter instead of a string. In python 2, sincebytes and str are equivalent it does not matter. For python 3, however, this results in an error as strpdate2num passes it to the standard library's strptime which only expects a string.

We could add another converter for byte input to the dates module. Not sure if it really would get enough use to make sense in adding, however.

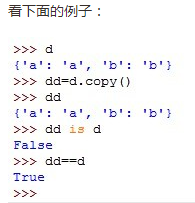
**九、**[**Python中is和==的区别**](http://blog.csdn.net/imzoer/article/details/8637408)

[**Python**](http://lib.csdn.net/base/python)中的对象包含三要素：id、type、value。

其中**id用来唯一标识一个对象，type标识对象的类型，value是对象的值**。

**is**  判断的是a对象是否就是b对象，是**通过id来判断的**。

**==** 判断的是a对象的值是否和b对象的值相等，是**通过value来判断的**。



**十、**[**列表排序方法reverse、sort、sorted详解**](http://www.cnblogs.com/qytang/p/5534452.html)

reverse反转/倒序排序、sort正序排序、sorted可以获取排序后的列表。在更高级列表排序中，后两中方法还可以加入条件参数进行排序。

**reverse()方法**

将列表中元素反转排序，比如下面这样

|  |  |
| --- | --- |
| 1 2 3 4 | >>> x = [1,5,2,3,4] >>> x.reverse() >>> x [4, 3, 2, 5, 1] |

reverse列表反转排序：是把原列表中的元素顺序从左至右的重新存放，而不会对列表中的参数进行排序整理。如果需要对列表中的参数进行整理，就需要用到列表的另一种排序方式sort正序排序。

**sort()排序方法**

此函数方法对列表内容进行正向排序，排序后的新列表会覆盖原列表（id不变），也就是sort排序方法是直接修改原列表list排序方法。

|  |  |
| --- | --- |
| 1 2 3 4 | >>> a = [5,7,6,3,4,1,2] >>> a.sort() >>> a [1, 2, 3, 4, 5, 6, 7] |

在玩蛇网许多python初学者，对sort()方法比较糊涂。有的时候会需要一个排序好的列表，而又想保存原有未排序列表，他们会这么操作：

|  |  |
| --- | --- |
| 1 2 3 4 | >>> a = [5,7,6,3,4,1,2] >>> b = a.sort() >>> print b None |

这个时候问题出现了，变量b得到的是一个空值。那么想要得到排序好的列表，又想保留原列表怎么办呢？列表sorted()方法可以帮你实现。

**sorted()方法**

即可以保留原列表，又能得到已经排序好的列表sorted()操作方法如下：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | >>> a = [5,7,6,3,4,1,2] >>> b = sorted(a) >>> a [5, 7, 6, 3, 4, 1, 2] >>> b [1, 2, 3, 4, 5, 6, 7] |

sorted()方法可以用在任何数据类型的序列中，返回的总是一个列表形式：

|  |  |
| --- | --- |
| 1 2 | >>> sorted('iplaypython.com') ['.', 'a', 'c', 'h', 'i', 'l', 'm', 'n', 'o', 'o', 'p', 'p', 't', 'y', 'y'] |

**三者的区别**

sort()是可变对象(字典、列表)的方法，无参数，无返回值，sort()会改变可变对象，因此无需返回值。sort()方法是可变对象独有的方法或者属性，而作为不可变对象如元组、字符串是不具有这些方法的，如果调用将会返回一个异常。

|  |  |
| --- | --- |
| 1 2 3 4 5 | >>> a=[5,4,3,2,1] >>> a.sort() >>> >>> a [1, 2, 3, 4, 5] |

sorted()是python的内置函数，并不是可变对象(列表、字典)的特有方法，sorted()函数需要一个参数(参数可以是列表、字典、元组、字符串)，无论传递什么参数，都将返回一个以列表为容器的返回值，如果是字典将返回键的列表。

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | >>> mystring="54321" >>> mytuple=(5,4,3,2,1) >>> mylist=[5,4,3,2,1] >>> sorted(mystring) ['1', '2', '3', '4', '5'] >>> sorted(mytuple) [1, 2, 3, 4, 5] >>> sorted(mylist) [1, 2, 3, 4, 5] |

reverse()与sort的使用方式一样，而reversed()与sorted()的使用方式相同

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | >>> mylist=[5,4,3,2,1] >>> mylist.reverse() >>> mylist [1, 2, 3, 4, 5] >>> mylist=[5,4,3,2,1] >>> for i in reversed(mylist): ... print i, ... 1 2 3 4 5 |

通过序列的切片也可以达到“逆转”的效果

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | >>> mystring="54321" >>> mytuple=(5,4,3,2,1) >>> mylist=[5,4,3,2,1] >>> mystring[::-1] '12345' >>> mytuple[::-1] (1, 2, 3, 4, 5) >>> mylist[::-1] [1, 2, 3, 4, 5] |