Python常用内置函数用法精要

- 内置函数（BIF，built-in functions）是Python内置对象类型之一，不需要额外导入任何模块即可直接使用，这些内置对象都封装在内置模块__builtins__之中，用C语言实现并且进行了大量优化，具有非常快的运行速度，推荐优先使用。使用内置函数dir()可以查看所有内置函数和内置对象：

```
>>> dir(__builtins__)
```

- 使用help(函数名)可以查看某个函数的用法。

```
>>> help(sum)
Help on built-in function sum in module builtins:

sum(iterable, start=0, /)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

Python常用内置函数用法精要

函数	功能简要说明
abs(x)	返回数字x的绝对值或复数x的模
all(iterable)	如果对于可迭代对象中所有元素x都等价于True，也就是对于所有元素x都有bool(x)等于True，则返回True。对于空的可迭代对象也返回True
any(iterable)	只要可迭代对象iterable中存在元素x使得bool(x)为True，则返回True。对于空的可迭代对象，返回False
ascii(obj)	把对象转换为ASCII码表示形式，必要的时候使用转义字符来表示特定的字符
bin(x)	把整数x转换为二进制串表示形式
bool(x)	返回与x等价的布尔值True或False
bytes(x)	生成字节串，或把指定对象x转换为字节串表示形式
callable(obj)	测试对象obj是否可调用。类和函数是可调用的，包含__call__()方法的类的对象也是可调用的
compile()	用于把Python代码编译成可被exec()或eval()函数执行的代码对象
complex(real, [imag])	返回复数
chr(x)	返回Unicode编码为x的字符

Python常用内置函数用法精要

续表1

函数	功能简要说明
delattr(obj, name)	删除属性，等价于del obj.name
dir(obj)	返回指定对象或模块obj的成员列表，如果不带参数则返回当前作用域内所有标识符
divmod(x, y)	返回包含整商和余数的元组((x-x%y)/y, x%y)
enumerate(iterable[, start])	返回包含元素形式为(0, iterable[0]), (1, iterable[1]), (2, iterable[2]), ...的迭代器对象
eval(s[, globals[, locals]])	计算并返回字符串s中表达式的值
exec(x)	执行代码或代码对象x
exit()	退出当前解释器环境
filter(func, seq)	返回filter对象，其中包含序列seq中使得单参数函数func返回值为True的那些元素，如果函数func为None则返回包含seq中等价于True的元素的filter对象
float(x)	把整数或字符串x转换为浮点数并返回
frozenset([x])	创建不可变的集合对象
getattr(obj, name[, default])	获取对象中指定属性的值，等价于obj.name，如果不存在指定属性则返回default的值，如果要访问的属性不存在并且没有指定default则抛出异常

Python常用内置函数用法精要

续表2

函数	功能简要说明
globals()	返回包含当前作用域内全局变量及其值的字典
hasattr(obj, name)	测试对象obj是否具有名为name的成员
hash(x)	返回对象x的哈希值，如果x不可哈希则抛出异常
help(obj)	返回对象obj的帮助信息
hex(x)	把整数x转换为十六进制串
id(obj)	返回对象obj的标识（内存地址）
input([提示])	显示提示，接收键盘输入的内容，返回字符串
int(x[, d])	返回实数（float）、分数（Fraction）或高精度实数（Decimal）x的整数部分，或把d进制的字符串x转换为十进制并返回，d默认为十进制
isinstance(obj, class-or-type-or-tuple)	测试对象obj是否属于指定类型（如果有多个类型的话需要放到元组中）的实例
iter(...)	返回指定对象的可迭代对象
len(obj)	返回对象obj包含的元素个数，适用于列表、元组、集合、字典、字符串以及range对象和其他可迭代对象

Python常用内置函数用法精要

续表3

函数	功能简要说明
list([x])、set([x])、tuple([x])、dict([x])	把对象x转换为列表、集合、元组或字典并返回，或生成空列表、空集合、空元组、空字典
locals()	返回包含当前作用域内局部变量及其值的字典
map(func, *iterables)	返回包含若干函数值的map对象，函数func的参数分别来自于iterables指定的每个迭代对象，
max(x)、 min(x)	返回可迭代对象x中的最大值、最小值，要求x中的所有元素之间可比较大小，允许指定排序规则和x为空时返回的默认值
next(iterator[, default])	返回可迭代对象x中的下一个元素，允许指定迭代结束之后继续迭代时返回的默认值
oct(x)	把整数x转换为八进制串
open(name[, mode])	以指定模式mode打开文件name并返回文件对象
ord(x)	返回1个字符x的Unicode编码
pow(x, y, z=None)	返回x的y次方，等价于x ** y或(x ** y) % z

Python常用内置函数用法精要

续表4

函数	功能简要说明
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)	基本输出函数
quit()	退出当前解释器环境
range([start,] end [, step])	返回range对象，其中包含左闭右开区间[start,end)内以step为步长的整数
reduce(func, sequence[, initial])	将双参数的函数func以迭代的方式从左到右依次应用至序列seq中每个元素，最终返回单个值作为结果。在Python 2.x中该函数为内置函数，在Python 3.x中需要从functools中导入reduce函数再使用
repr(obj)	返回对象obj的规范化字符串表示形式，对于大多数对象有eval(repr(obj))==obj
reversed(seq)	返回seq（可以是列表、元组、字符串、range以及其他可迭代对象）中所有元素逆序后的迭代器对象

Python常用内置函数用法精要

续表5

函数	功能简要说明
round(x [, 小数位数])	对x进行四舍五入，若不指定小数位数，则返回整数
sorted(iterable, key=None, reverse=False)	返回排序后的列表，其中iterable表示要排序的序列或迭代对象key用来指定排序规则或依据，reverse用来指定升序或降序。该函数不改变iterable内任何元素的顺序
str(obj)	把对象obj直接转换为字符串
sum(x, start=0)	返回序列x中所有元素之和，返回start+sum(x)
type(obj)	返回对象obj的类型
zip(seq1 [, seq2 [...]])	返回zip对象，其中元素为(seq1[i], seq2[i], ...)形式的元组最终结果中包含的元素个数取决于所有参数序列或可迭代对象中最短的那个

类型转换与类型判断

- 内置函数bin()、oct()、hex()用来将整数转换为二进制、八进制和十六进制形式，这三个函数都要求参数必须为整数。

```
>>> bin(555)                    #把数字转换为二进制串
'0b1000101011'
>>> oct(555)                    #转换为八进制串
'0o1053'
>>> hex(555)                    #转换为十六进制串
'0x22b'
```

类型转换与类型判断

- 内置函数int()用来将其他形式的数字转换为整数，参数可以为整数、实数、分数或合法的数字字符串。当参数为数字字符串时，还允许指定第二个参数base用来说明数字字符串的进制，base的取值应为0或2-36之间的整数，其中0表示按数字字符串隐含的进制进行转换。

```
>>> int(-3.2)                                   #把实数转换为整数
-3
>>> from fractions import Fraction, Decimal
>>> x = Fraction(7, 3)
>>> x
Fraction(7, 3)
>>> int(x)                                      #把分数转换为整数
2
>>> x = Decimal(10/3)
>>> x
Decimal('3.333333333333333481363069950020872056484222412109375')
>>> int(x)                                      #把高精度实数转换为整数
3
```

类型转换与类型判断

```
>>> int('0x22b', 16)                    #把十六进制数转换为十进制数
555
>>> int('22b', 16)                      #与上一行代码等价
555
>>> int(bin(54321), 2)                  #二进制与十进制之间的转换
54321
>>> int('0b111')                        #非十进制字符串进，必须指定第二个参数
ValueError: invalid literal for int() with base 10: '0b111'
>>> int('0b111', 0)                     #第二个参数0表示使用字符串隐含的进制
7
>>> int('0b111', 6)                     #第二个参数应与隐含的进制一致
ValueError: invalid literal for int() with base 6: '0b111'
>>> int('0b111', 2)
7
>>> int('111', 6)                       #字符串没有隐含进制
                                        #第二个参数可以为2-36之间的数字
43
```

类型转换与类型判断

- 内置函数float()用来将其他类型数据转换为实数，complex()可以用来生成复数。

```
>>> float(3)                    #把整数转换为实数
3.0
>>> float('3.5')                #把数字字符串转换为实数
3.5
>>> float('inf')                #无穷大，其中inf不区分大小写
inf
>>> complex(3)                  #指定实部
(3+0j)
>>> complex(3, 5)               #指定实部和虚部
(3+5j)
>>> complex('inf')              #无穷大
(inf+0j)
```

类型转换与类型判断

- ord()和chr()是一对功能相反的函数，ord()用来返回单个字符的Unicode码，而chr()则用来返回Unicode编码对应的字符，str()则直接将其任意类型参数转换为字符串。

```
>>> ord('a')                #查看指定字符的Unicode编码
97
>>> chr(65)                 #返回数字65对应的字符
'A'
>>> chr(ord('A')+1)         #Python不允许字符串和数字之间的加法操作
'B'
>>> chr(ord('国')+1)        #支持中文
'图'
>>> ord('董')               #这个用法仅适用于Python 3.x
33891
>>> ord('付')
20184
>>> ord('国')
22269
```

类型转换与类型判断

```
>>> ''.join(map(chr, (33891, 20184, 22269)))
'董付国'
>>> str(1234)                                  #直接变成字符串
'1234'
>>> str([1,2,3])
'[1, 2, 3]'
>>> str((1,2,3))
'(1, 2, 3)'
>>> str({1,2,3})
'{1, 2, 3}'
```

类型转换与类型判断

- 内置类ascii可以把对象转换为ASCII码表示形式，必要的时候使用转义字符来表示特定的字符。

```
>>> ascii('a')
"'a'"
>>> ascii('董付国')
"'\\u8463\\u4ed8\\u56fd'"
>>> eval(_)                                    #对字符串进行求值
'董付国'
```

类型转换与类型判断

- 内置类bytes用来生成字节串，或者把指定对象转换为特定编码的字节串。

```
>>> bytes()                                    #生成空字节串
b''
>>> bytes(3)                                   #生成长度为3的字节串
b'\x00\x00\x00'
>>> bytes('董付国','utf8')                     #把字符串转换为字节串
b'\xe8\x91\xa3\xe4\xbb\x98\xe5\x9b\xbd'
>>> bytes('董付国','gbk')                      #可以指定不同的编码格式
b'\xb6\xad\xb8\xb6\xb9\xfa'
>>> str(_, 'gbk')                             #使用同样的编码格式进行解码
'董付国'
>>> '董付国'.encode('gbk')                     #等价于使用btyes()进行转换
b'\xb6\xad\xb8\xb6\xb9\xfa'
```

类型转换与类型判断

- list()、tuple()、dict()、set()、frozenset()用来把其他类型的数据转换成为列表、元组、字典、可变集合和不可变集合，或者创建空列表、空元组、空字典和空集合。

```
>>> list(range(5))                          #把range对象转换为列表
[0, 1, 2, 3, 4]
>>> tuple(_)                                #一个下划线表示上一次正确的输出结果
(0, 1, 2, 3, 4)
>>> dict(zip('1234', 'abcde'))              #创建字典
{'4': 'd', '2': 'b', '3': 'c', '1': 'a'}
>>> set('1112234')                          #创建可变集合，自动去除重复
{'4', '2', '3', '1'}
>>> _.add('5')
>>> _
{'2', '1', '3', '4', '5'}
>>> frozenset('1112234')                    #创建不可变集合，自动去除重复
frozenset({'2', '1', '3', '4'})
>>> _.add('5')                              #不可变集合frozenset不支持元素添加与删除
AttributeError: 'frozenset' object has no attribute 'add'
```

类型转换与类型判断

- 内置函数type()和isinstance()可以用来判断数据类型，常用来对函数参数进行检查，可以避免错误的参数类型导致函数崩溃或返回意料之外的结果。

```
>>> type(3)                                    #查看3的类型
<class 'int'>
>>> type([3])                                  #查看[3]的类型
<class 'list'>
>>> type({3}) in (list, tuple, dict)           #判断{3}是否为list,tuple或dict类型的实例
False
>>> type({3}) in (list, tuple, dict, set)      #判断{3}是否为list,tuple,dict或set的实例
True
>>> isinstance(3, int)                         #判断3是否为int类型的实例
True
>>> isinstance(3j, int)
False
>>> isinstance(3j, (int, float, complex))      #判断3是否为int,float或complex类型
True
```

最值与求和

- max()、min()、sum()这三个内置函数分别用于计算列表、元组或其他包含有限个元素的可迭代对象中所有元素最大值、最小值以及所有元素之和。
- sum()默认（可以通过start参数来改变）支持包含数值型元素的序列或可迭代对象，max()和min()则要求序列或可迭代对象中的元素之间可比较大小。

```
>>> from random import randint
>>> a = [randint(1,100) for i in range(10)]   #包含10个[1,100]之间随机数的列表
>>> print(max(a), min(a), sum(a))             #最大值、最小值、所有元素之和
>>> sum(a) / len(a)                           #平均值
```

最值与求和

- 函数max()和min()还支持default参数和key参数，其中default参数用来指定可迭代对象为空时默认返回的最大值或最小值，而key参数用来指定比较大小的依据或规则，可以是函数或lambda表达式。函数sum()还支持start参数，用来控制求和的初始值。

```
>>> max(['2', '111'])                    #不指定排序规则
'2'
>>> max(['2', '111'], key=len)           #返回最长的字符串
'111'
>>> print(max([], default=None))          #对空列表求最大值，返回空值None
None
```

基本输入输出

- input()和print()是Python的基本输入输出函数，前者用来接收用户的键盘输入，后者用来把数据以指定的格式输出到标准控制台或指定的文件对象。不论用户输入什么内容，input()一律返回字符串对待，必要的时候可以使用内置函数int()、float()或eval()对用户输入的内容进行类型转换。

基本输入输出

```
>>> x = input('Please input: ')
Please input: 345
>>> x
'345'
>>> type(x)                          #把用户的输入作为字符串对待
<class 'str'>
>>> int(x)                           #转换为整数
345
>>> eval(x)                          #对字符串求值，或类型转换
345
>>> x = input('Please input: ')
Please input: [1, 2, 3]
>>> x
'[1, 2, 3]'
>>> type(x)
<class 'str'>
>>> eval(x)
[1, 2, 3]
```

基本输入输出

- 内置函数print()用于输出信息到标准控制台或指定文件，语法格式为：

```
print(value1, value2, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

✓sep参数之前为需要输出的内容（可以有多个）；

✓sep参数用于指定数据之间的分隔符，默认为空格；

✓end参数用于指定输出完数据之后再输出什么字符；

✓file参数用于指定输出位置，默认为标准控制台，也可以重定向输出到文件。

```
>>> print(1, 3, 5, 7, sep='\t')          #修改默认分隔符
1       3       5       7
>>> for i in range(10):                  #修改end参数，每个输出之后不换行
    print(i, end=' ')
0 1 2 3 4 5 6 7 8 9
>>> with open('test.txt', 'a+') as fp:
    print('Hello world!', file=fp)       #重定向，将内容输出到文件中
```

排序与逆序

- sorted()对列表、元组、字典、集合或其他可迭代对象进行排序并返回新列表，reversed()对可迭代对象（生成器对象和具有惰性求值特性的zip、map、filter、enumerate等类似对象除外）进行翻转（首尾交换）并返回可迭代的reversed对象。

```
>>> x = list(range(11))
>>> import random
>>> random.shuffle(x)                                    #打乱顺序
>>> x
[2, 4, 0, 6, 10, 7, 8, 3, 9, 1, 5]
>>> sorted(x)                                            #以默认规则排序
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> sorted(x, key=lambda item:len(str(item)), reverse=True)
                                                         #按转换成字符串以后的长度降序排列
[10, 2, 4, 0, 6, 7, 8, 3, 9, 1, 5]
>>> sorted(x, key=str)                                   #按转换成字符串以后的大小升序排列
[0, 1, 10, 2, 3, 4, 5, 6, 7, 8, 9]
```

排序与逆序

```
>>> x = ['aaaa', 'bc', 'd', 'b', 'ba']
>>> sorted(x, key=lambda item: (len(item), item))
                                        #先按长度排序，长度一样的正常排序
['b', 'd', 'ba', 'bc', 'aaaa']
>>> reversed(x)                         #逆序，返回reversed对象
<list_reverseiterator object at 0x0000000003089E48>
>>> list(reversed(x))                   #reversed对象是可迭代的
[5, 1, 9, 3, 8, 7, 10, 6, 0, 4, 2]
```

枚举与迭代

- enumerate()函数用来枚举可迭代对象中的元素，返回可迭代的enumerate对象，其中每个元素都是包含索引和值的元组。

```
>>> list(enumerate('abcd'))                                  #枚举字符串中的元素
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
>>> list(enumerate(['Python', 'Greate']))                    #枚举列表中的元素
[(0, 'Python'), (1, 'Greate')]
>>> list(enumerate({'a':97, 'b':98, 'c':99}.items()))        #枚举字典中的元素
[(0, ('c', 99)), (1, ('a', 97)), (2, ('b', 98))]
>>> for index, value in enumerate(range(10, 15)):            #枚举range对象中的元素
    print((index, value), end=' ')
(0, 10) (1, 11) (2, 12) (3, 13) (4, 14)
```

map()、reduce()、filter()

- 内置函数map()把一个函数func依次映射到序列或迭代器对象的每个元素上，并返回一个可迭代的map对象作为结果，map对象中每个元素是原序列中元素经过函数func处理后的结果。

```
>>> list(map(str, range(5)))  #把列表中元素转换为字符串
['0', '1', '2', '3', '4']
>>> def add5(v):                    #单参数函数
    return v+5
>>> list(map(add5, range(10)))#把单参数函数映射到一个序列的所有元素
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
>>> def add(x, y):                  #可以接收2个参数的函数
    return x+y
>>> list(map(add, range(5), range(5,10)))
                                    #把双参数函数映射到两个序列上
[5, 7, 9, 11, 13]
```

map()、reduce()、filter()

```
>>> def myMap(iterable, op, value):   #自定义函数
    if op not in '+-*/':               #实现序列与数字的四则运算
        return 'Error operator'
    func = lambda i:eval(repr(i)+op+repr(value))
    return map(func, iterable)

>>> list(myMap(range(5), '+', 5))
[5, 6, 7, 8, 9]
>>> list(myMap(range(5), '-', 5))
[-5, -4, -3, -2, -1]
>>> list(myMap(range(5), '*', 5))
[0, 5, 10, 15, 20]
>>> list(myMap(range(5), '/', 5))
[0.0, 0.2, 0.4, 0.6, 0.8]
```

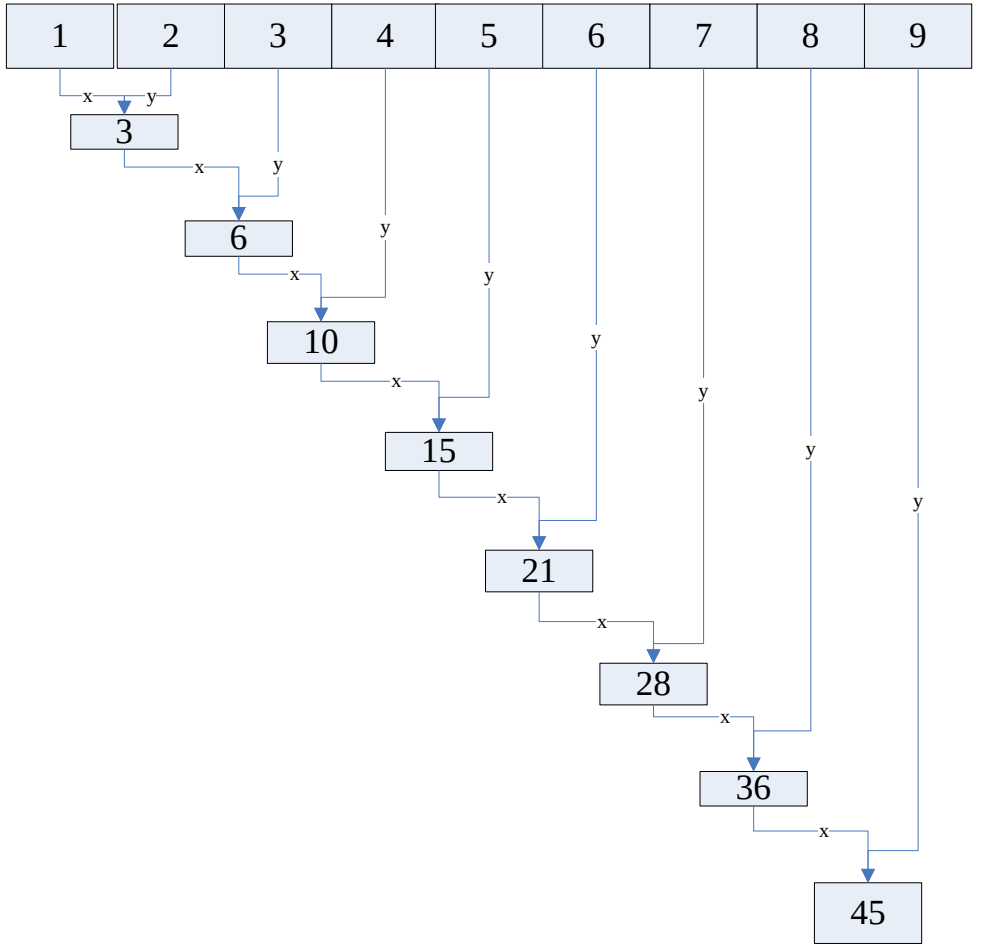
map()、reduce()、filter()

```
>>> import random
>>> x = random.randint(1, 1e30)      #生成指定范围内的随机整数
>>> x
839746558215897242220046223150
>>> list(map(int, str(x)))           #提取大整数每位上的数字
[8, 3, 9, 7, 4, 6, 5, 5, 8, 2, 1, 5, 8, 9, 7, 2, 4, 2, 2, 2, 0, 0, 4, 6,
2, 2, 3, 1, 5, 0]
```

map()、reduce()、filter()

- 标准库functools中的函数reduce()可以将一个接收2个参数的函数以迭代累积的方式从左到右依次作用到一个序列或迭代器对象的所有元素上，并且允许指定一个初始值。

```
>>> from functools import reduce
>>> seq = list(range(1, 10))
>>> reduce(lambda x, y: x+y, seq)
45
```



map()、reduce()、filter()

```
>>> import operator                                  #标准库operator提供了大量运算
>>> operator.add(3,5)                               #可以像普通函数一样直接调用
8
>>> reduce(operator.add, seq)                       #使用add运算
45
>>> reduce(operator.mul, seq)                       #乘法运算
362880
>>> reduce(operator.mul, range(1, 6))               #5的阶乘
120
>>> reduce(operator.add, map(str, seq))             #转换成字符串再累加
'123456789'
>>> reduce(operator.add, [[1, 2], [3]], []) #这个操作占用空间较大，慎用
[1, 2, 3]
```

map()、reduce()、filter()

- 内置函数filter()将一个单参数函数作用到一个序列上，返回该序列中使得该函数返回值为True的那些元素组成的filter对象，如果指定函数为None，则返回序列中等价于True的元素。

```
>>> seq = ['foo', 'x41', '?!', '***']
>>> def func(x):
    return x.isalnum()                    #测试是否为字母或数字

>>> filter(func, seq)                     #返回filter对象
<filter object at 0x000000000305D898>
>>> list(filter(func, seq))               #把filter对象转换为列表
['foo', 'x41']
```

range()

- range()是Python开发中非常常用的一个内置函数，语法格式为range([start,] end [, step])，有range(stop)、range(start, stop)和range(start, stop, step)三种用法。该函数返回具有惰性求值特点的range对象，其中包含左闭右开区间[start,end)内以step为步长的整数。参数start默认为0，step默认为1。

```
>>> range(5)                          #start默认为0，step默认为1
range(0, 5)
>>> list(_)
[0, 1, 2, 3, 4]
>>> list(range(1, 10, 2))             #指定起始值和步长
[1, 3, 5, 7, 9]
>>> list(range(9, 0, -2))             #步长为负数时，start应比end大
[9, 7, 5, 3, 1]
```

zip()

- zip()函数用来把多个可迭代对象中的元素压缩到一起，返回一个可迭代的zip对象，其中每个元素都是包含原来的多个可迭代对象对应位置上元素的元组，如同拉拉链一样。

```
>>> list(zip('abcd', [1, 2, 3]))                #压缩字符串和列表
[('a', 1), ('b', 2), ('c', 3)]
>>> list(zip('123', 'abc', ',.!'))              #压缩3个序列
[('1', 'a', ','), ('2', 'b', '.'), ('3', 'c', '!')]
>>> x = zip('abcd', '1234')
>>> list(x)
[('a', '1'), ('b', '2'), ('c', '3'), ('d', '4')]
```

精彩案例赏析

❖**例1：**用户输入一个三位自然数，计算并输出其佰位、十位和个位上的数字。

```
x = input('请输入一个三位数：')
x = int(x)
a = x // 100
b = x // 10 % 10
c = x % 10
print(a, b, c)
```

想一想，还有别的办法吗？

精彩案例赏析

- 还可以这样写

```
x = input('请输入一个三位数：')
x = int(x)
a, b = divmod(x, 100)
b, c = divmod(b, 10)
print(a, b, c)
```

还可以再简单些吗？

精彩案例赏析

- 居然可以这样？OMG

```
x = input('请输入一个三位数：')
a, b, c = map(int, x)
print(a, b, c)
```

精彩案例赏析

❖**例2：**已知三角形的两边长及其夹角，求第三边长。

```
import math

x = input('输入两边长及夹角（度）：')
a, b, theta = map(float, x.split())
c = math.sqrt(a**2 + b**2 - 2*a*b*math.cos(theta*math.pi/180))
print('c=', c)
```

精彩案例赏析

❖**例3：**任意输入三个英文单词，按字典顺序输出。

```
s = input('x,y,z=')
x, y, z = sorted(s.split(','))
print(x, y, z)
```