Python 自学手册

老 A

目录

Number

```
# 数字类型转换

a = 1.2
print ( 'after transport 1.2 is ', int(a) )
b = 5
print ( 'after transport 5 is ', float(b) )
print ( 'after transport 1.2 and 5 to complex is ', complex(a,b) )


# python 中整数除法总是返回浮点型数据，Fortran 中整数相除返回整数

print ( '7 / 3 is ', 7 / 3 )


# 如果要使用数学函数以及 pi 与 e，需要先 import math

import math
print ( 'pi is ', math.pi )
print ( 'e is ', math.e )
print ( 'sin(pi/2) is ', math.sin( math.pi/2 ) )


# 这里还要指出一下，python 中的取绝对值函数为 fabs，与 Fortran 中的 abs

不同

print ( '|-3.1| is ', math.fabs(-3.1) )
```

Character

```
# 访问字符串中的值

var1 = 'hello world!'
var2 = 'luk'
print ( 'var1[0]: ', var1[0] )

# 这里要注意

# 1. python 中的元素下标从 0 开始，Fortran 中默认从 1 开始

# 2. 引用方式也不一样，python 是[]，fortran 是()

# 3. 引用单个元素时，比如，python 是 var[1]，fortran 是 var(1:1)


# 字符串更新（替换）

var = 'lkl'
print ( 'before var:', var )
var = var[0] + 'uk'
print ( 'after var', var )


# 字符串运算符

a = 'hello'; b = 'python'
print ( ' a is ', a )
print ( ' b is ', b )

print ( ' a + b is ', a + b ) # 字符串拼接，fortran 用//拼接字符串

print ( ' a*2 is ', a * 2 ) # 连续输出字符串 a 两次


if ( 'h' in a ):  # 判断字符'h'是否在字符串 a 中

    print ( ' h in a ' )
else:
    print ( ' h not in a ' )


# 字符串格式化

print ( ' I am %s, and I am %d years old! ' %( 'luk', 10 ) ) #
这里单引号中的内容与%后面内容之间无逗号
```

数字格式化

print ('%.15f' %(1.2)) # 格式化输出数字，python 为%.15f 输出 15 位小数，fortran 为 fn.15，此处的 n 一般要满足 n-15>=3

按规律输出字符

```
a = 'abcdefgh'
print ( a[::2] ) # 从第一个字符输出，步长为 2
```

使用负数从字符串右边末尾向左边反向索引，最右侧索引为-1，正向的话，最左侧为 0

```
str = 'luklukluk'
print ( str[-3] )
```

List

```
# 访问列表中的元素

list1 = [ 'Google', 'Runoob', 'luk', 1993, 2000 ]
list2 = [ 1, 2, 3, 4, 5, 6, 7, 8 ]
print ( ' list1[0]: ', list1[0] )

print ( ' list2[1:5]: ', list2[1:5] ) # 列表输出后，数据仍有[]，而

字符串没有''或""


# 改变列表

list1 = [ 'Google', 'luk', 1993, 1994 ]
print ( ' The third element is : ', list1[2] )

list1[2] = 2008  # 此处一定要注意，列表中的元素可以改变，字符串中的

单个元素不能用赋值的方法改变

print ( ' after update the third element: ', list1[2] )
print ( ' after update the list is: ', list1 )


# 删除列表中的元素

# 可以使用 del 语句来删除列表中的元素

list1 = list1
print ( ' before del the third element: ', list1 )
del list1[2]
print ( ' after del the third element: ', list1 )


# 列表脚本操作符

list1 = [ 1, 2, 3 ]
print ( ' the length of list1 = [ 1, 2, 3 ] is: ', len(list1) )
list2 = [ 4, 5, 6 ]
print ( ' [1,2,3] + [4,5,6] = ', list1 + list2 )
list1 = [ 'hi' ]
print ( ' list1 is: ', list1 )
print ( ' list1*4 is: ', list1*4 )

if ( 3 in [ 1,2,3 ] ):
  print ( ' 3 in [1,2,3] ' )
```

```
else:
  print ( ' 3 not in [1,2,3] ' )


# 列表截取与拼接

L = [ 'Google', 'luk', 'taobao' ]
print ( ' L[2] is: ', L[2] )

print ( ' L[-1] is: ', L[-1] ) # -1 表示最左边第一个

print ( ' L[1:] is: ', L[1:] )


# 嵌套列表

a = [ 'a', 'b', 'c' ]
n = [ 1, 2, 3 ]
x = [ a, n ]
print ( ' x is: ', x )
print ( ' x[0] is: ', x[0] )
print ( ' x[0][1] is: ', x[0][1] )


# 创建一维列表

L = [ i for i in range(0,15) ]  # 相当于 fortran 中的 L = [ ( i,

i = 1, 10 ) ]
print ( ' L is: ', L )
print ( ' L[::2] is: ', L[::2] ) # L[start:end:span]

# 创建二维列表

list_2d = [ [ 0 for col in range(5) ] for row in range(5) ]

list_2d[0].append(3); list_2d[0].append(5) # 替换第一行的数字，从

最后一个数字填

list_2d[2].append(7)
print ( ' list_2d is: ', list_2d )


# 列表的复制

a = [ 1, 2, 3 ]
print ( ' origin a is: ', a )
b = a
b[1] = 0
print ( ' change b[1] = 0, the a is: ', a )
```

```
print ( ' id(a) is: ', id(a), ' id(b) is: ', id(b) ) # 这样赋值，
```

id 相同，改变 b 中元素后，a 中相应位置元素也会改变，这点很是蛋疼

```
a = [ 1, 2, 3 ]
print ( ' origin a is: ', a )
b = a[:]
b[1] = 0
print ( ' change b[1] = 0, the a is: ', a )
print ( ' id(a) is: ', id(a), ' id(b) is: ', id(b) )
```

id 不同，改变 b 中元素后，a 中相应位置的元素不会改变。字符串无论是=还是[:]赋值，id 均相同，没有 list 这种情况

Tuple

python 的元组 tuple 与列表类似，不过元组的元素不能修改

元组使用小括号()，列表使用方括号[]

元组的创建：只需要在括号中添加元素，并使用逗号隔开

```
tup1 = ( 'Google', 'luk', 1993, 1994 )
tup2 = ( 1, 2, 3, 4, 5 )
tup3 = ( 'a', 'b', 'c', 'd' )
print ( ' tup1[0] = ', tup1[0] )
print ( ' tup2[1:5] = ', tup2[1:5] )
print ( ' tup3 = ', tup3 )
```

创建空元组

```
tup1 = ()
```

元组中只包含一个元素时，需要在元素的后面添加逗号，否则括号会被当作运

算符使用

```
tup1 = (50)
print ( ' tup1 = ', tup1 )
print ( ' type(tup1): ', type(tup1) ) # 不加逗号，类型为整型


tup1 = (50,)
print ( ' tup1 = ', tup1[:] )
print ( ' type(tup1): ', type(tup1) )  # 加逗号，类型为元组
```

修改元组

```
tup1 = ( 12, 34, 56 )
print ( ' tup1 = ', tup1 )
tup2 = ( 'abc', 'xyz' )
print ( ' tup2 = ', tup2 )
```

tup1[0] = 100, 这种修改对元组来说非法，对字符串也非法，但是对列表

可以

创建一个新元组

```
tup3 = tup1 + tup2
print ( ' tup1 + tup2 = ', tup3 )

# 删除元组，用 del 函数
del tup3

# 元组运算符
print ( ' len(tup1): ', len(tup1) )
print ( ' tup1*3: ', tup1*3 )
if ( 12 in tup1 ):
  print ( ' 12 in tup1 ' )
else:
  print ( ' 12 not in tup1 ' )

# 元组的索引与截取
L = ( 'Google', 'luk', 'Taobao' )
print ( ' L is ', L )
print ( ' L[2] is ', L[2] )
print ( ' L[-2] is ', L[-2] ) # -1 表示最左边第一个
print ( ' L[1:] ', L[1:] )

# 元组内置函数
print ( ' len(L) is ', len(L) )
print ( ' max(L) is ', max(L) )
print ( ' min(L) is ', min(L) )

# 将列表转为元组用函数 tuple
list1 = [ 'luk', 'fyn' ]
print ( ' list1 is ', list1 )
tup = tuple(list1)
print ( ' transpose list1 to tup is ', tup )

# 下面看一个"可变的"tuple
t = ( 'a', 'b', [ 'A', 'B' ] )
print ( ' t before is ', t )
t[2][0] = 'X'
t[2][1] = 'Y'
```

```
print ( ' t after is ', t )
```

Dict

```
# 字典可存储任意类型的对象

# 字典的每个键值（key=>value）对用冒号（:）分割

# 每个对之间用逗号（,）分割，整个字典写在花括号（{}）中

# d = { key1 : value1, key2 : value2 }

# python 中的字典有点类似 fortran 中的 type 数据类型


# 字典中的键必须是唯一的，但值则不必

# 值可以取任意类型


dict1 = { 'abc': 456 }
print ( " dict1 is: ", dict1 )
dict2 = { 'abc': 123, 98.6: 37 }
print ( ' dict2 is: ', dict2 )
print ( ' dict2[98.6] is ', dict2[98.6] )


# 访问字典里的值

dict1 = { 'Name': 'lkl', 'Age': 24, 'Class': 1 }
print ( " dict1['Name'] is ", dict1['Name'] )
print ( " dict1['Age'] is ", dict1['Age'] )


# 修改字典

dict1['Age'] = 25   # 更新 Age 的值

print ( " dict1['Age'] is ", dict1['Age'] )

dict1['School'] = "Chang'An University"  # 向 dict1 里面添加信息

print ( " dict1['School'] is ", dict1['School'] )


# 删除字典及其元素

del dict1['Name']

print ( " after del the dict1['Name'] is ", dict1 )  # 删除字典

中的底朝天['Name']
```

```
dict1.clear()  # 清空字典

del dict1  # 删除整个字典


# 字典键的特性

# 创建时如果同一个键出现两次，则后一个值被保留

dict1 = { 'Name': 'luk', 'Age': 4, 'Name': 'lkl' }
print ( " dict1 is ", dict1 )
print ( " dict1['Name'] ", dict1['Name'] )


# 字典中的键可以是数字，字符串，元组，不能是列表

dict1 = { 76: 12, 'Name': 'luk', ( 1, 2, 3 ): 1 }

print ( ' dict1 is ', dict1 )
print ( ' dcit1[76] is ', dict1[76] )
print ( ' dict1[(1,2,3)] is ', dict1[(1,2,3)] )
print ( " dict1['Name'] is ",dict1['Name'] )


print ( ' len(dict1) is ', len(dict1) )  # 计算字典元素个数

print ( ' str(dict1) is ', str(dict1) )  # 将字典输出成字符串

print ( ' type(dict1) is ', type(dict1) )  # 返回输入变量的类型
```

If

```
# if 语句的形式

# if condition_1:
#   statement_block1
# elif condition_2:
#   statement_block2
# else:
#   statement_block3

# 注意事项：

# 1. python 中用 elif 代替了 else if，所以 if 语句的关键字为：if--elif--[elif]--else

# 注意 fortran 中是：if--else if--[else if]--else--end if

#  2. 每个条件后面都要使用冒号(:),表示接下来是满足条件后要执行的语句块

# 3. 使用缩进划分语句块，一个语句块，缩进必须一致

# 4. python 中没有 switch-case 语句

var1 = 100
if var1:
  print ( ' 1 - if expression is ture ' )
  print ( ' var1 is ', var1 )

var2 = 0
if var2:
  print ( ' 2 - if expression is ture ' )
else:
  print ( ' 2 - if expression is false ' )

print ( ' Good bye! ' )


# 狗的年龄计算

age = int( input( " please input your dog's age: " ) )
print ( "" )
if age < 0:
  print ( " Are you kidding me ? " )
```

```
elif age == 1:
  print ( " eq to 14 years old human! " )
elif age == 2:
  print ( " eq to 22 years old human! " )
elif age > 2:
  humanAge = 22 + ( age - 2 ) * 5
  print ( " eq to human is %d old! " %( humanAge ) )

# if 中常用的操作符：>, >=, <, <=, ==, !=(fortran 中为/=)


# 数字比较运算

number = 7
guess = -1
print ( ' guess number game: ' )
while guess != number:
  guess = int( input( " please input your guess number: " ) )
  if guess == number:
    print ( ' you are right! ' )
  elif guess < number:
    print ( ' the guess number is less! ' )
  elif guess > number:
    print ( ' the guess number is more! ' )


# if 嵌套

#  在嵌套 if 语句中，可以把 if...elif...else 结构放在另一个

if...elif...else 结构中

# if 表达式 1:

#   语句

#   if 表达式 2:

#     语句

#   elif 表达式 3:

#     语句

#   else:
```

```
#     语句

# elif 表达式 4:

#   语句

# else:

#   语句

num = int( input( " please input a number: " ) )
if num % 2 == 0:
  if num % 3 == 0:
    print ( " the number of input can be devided by 2 and 3! " )
  else:
    print ( " the number of input can be devided by 2 and 3 is
not! " )
else:
  if num % 3 == 0:
    print ( " the number of input can be devided by 3 and 2 is
not! " )
  else:
    print ( " the number of input can not be devided by 2 and 3!
" )


# 例子

import random

x = random.choice( range(100) )  # 随机数 0-99

y = random.choice( range(200) )  # 随机数 0-199

if x > y:
  print ( ' x : ', x )
elif x == y:
  print ( ' x + y : ', x + y )
else:
  print ( ' y : ', y )

# 一个 if 对应一个 else，但是一个 if 中可以嵌套多个 if


# 数字猜谜游戏优化

print ( " guess number game: " )
```

```
a = 1
i = 0

while a != 20:  # 注意冒号

  a = int( input( " please input number: " ) )
  i += 1
  if a == 20:
    if i < 3:

      print ( " 真厉害，这么快就猜对了 " )

    else:

      print ( " 总算猜对了 " )

  elif a < 20:

    print ( " 你猜的数字太小了 " )

  else:

    print ( " 你猜的数字太大了 " )



###  退出提示

input ( " Press the enter! " )
```

TestIf

```
# -*- coding:utf-8 -*-
# 1. python1
height = float( input( " please input the heigth: " ) )
weight = float( input( " please input the weight: " ) )

# 输入数据时，一定要加 int，float 等，不然 type 是字符串

if weight > height - 100.0:
  print ( " too fat! " )
else:
  print ( " under control! " )

# 2. python2
rain = float ( input( " please input the rain: " ) )
wind = float ( input( " please input the wind: " ) )

if rain >= 500 or wind >= 10:   # 此处逻辑判断与 fortran 不同，

fortran 是.or.

  print ( " class is over! " )
else:
  print ( " as usual! " )

# 3. python3
source = float( input( " please input the source: " ) )
if source >= 90 and source <= 100:
  print ( " Grade = A " )
elif source >= 80:
  print ( " Grade = B " )
elif source >= 70:
  print ( " Grade = C " )
elif source >= 60:
  print ( " Grade = D " )
elif source >= 0:
  print ( " Grade = E " )
else:
  print ( " Grade = ? " )

# 4. python4
print ( " please input x, y: " )

x, y = map( float, input( '' ).split() ) # 这一行代码用来读取多个

数据，左边是一个变量时，存在列表里
```

```
print ( ' x, y is ', x, y )
if x > 0:
  if y > 0: # x > 0 and y > 0
    ans = 1
  elif y < 0: # x > 0 and y < 0
    ans = 4
  else:  # x > 0 and y = 0
    ans = 0
elif x < 0:
  if y > 0: # x < 0 and y > 0
    ans = 2
  elif y < 0: # x < 0 and y < 0
    ans = 4
  else:  # x < 0 and y = 0
    ans = 0
else:  # x = 0
  ans = 0

if ans != 0:
  print ( ' %d quadrant! ' %( ans ) )
else:
  print ( ' on the axis! ' )

# 5. python5
import math
eps = 1e-15
b = 3.0
a = math.sqrt(b) ** 2 - b
if math.fabs( a - 0.0 ) < eps:
  print ( ' a == 0.0 ' )
else:
  print ( ' a != 0.0 ' )
print ( ' a =  ', a )

# 6. python6
str1 = input( " please input str1: " )
str2 = input( " please input str2: " )
if str1 > str2:
  relation = '>'
elif str1 == str2:
  relation = '='
else:
  relation = '<'
```

```
print ( " %s %s %s " %( str1, relation, str2 ) )

# 总结

# 1. 读取多个数据时,语句为 x,y[z,...] = map( int, input().split() ),

其中的 int 也可以换为 float 等

# 2. input 读取数据，无论输入什么类型的数据，都是字符串

# 3. 如果想转换，则为 a = int( input() )

# 4. 使用数学函数，必须先 import math
```

Loop

```
# python 中的循环语句有 for 和 while

# 1. while 循环

# 格式为：

# while 判断条件:

#    语句

# 注意：冒号和缩进。这点与 if 语句一样

# python1: 计算 1 到 100 的总和

# python 中的 while 语句相当于 Fortran 中的 do 循环
n = 100
sum1 = 0
counter = 1
while counter <= n:
  sum1 = sum1 + counter
  counter += 1


print ( " 1 到 %d 之和为：%d " %( n, sum1 ) )
print ( " ------------------------------ " )
# python2: while 循环使用 else 语句

#  python 中的 while...else 语句相当于 fortran 中的 do...if...else 语
句
count = 0
while count < 5:
  print ( count, " 小于 5 " )
  count = count + 1
else:
  print ( count, " 大于或等于 5 " )


# python3: 简单语句组
```

```
flag = 1
# while ( flag ): print ( " welcome! " )
# 此语句为无限循环，相当于 fortran 中的 do end do 结构语句
print ( " Good bye! " )


# python4: for 语句
# for 循环可以遍历任何序列的项目，如一个列表或一个字符串
# for 循环的一般格式如下：
# for <variable> in <sequence>:
#   <statements>
# else:
#   <statements>
# 注意：冒号和缩进
languages = [ "c", "c++", "Perl", "fortran" ]
for x in languages:
  print ( x )


# python5: for 语句中使用 break,break 跳出整个循环体
# 判断循环与 if 语句，看缩进即可
sites = [ "Baidu", "Google", "Runoob", "Taobao" ]
for site in sites:
  if site == "Runoob":
    print ( " luk " )
    break # 跳出整个循环
  print ( " 元素： ", site )
else:
  print ( " 没有循环元素 " )


print ( " 循环完毕！ " )


# python6: range()函数
# 如果需要遍历数字序列，可以使用内置 range()函数,它可以生成数列
```

```
for i in range(5):  # 生成 5 个数，默认从 0 开始。相当于 fortran 中的
do i = 0, 4
    print ( i )

#  range()也可以指定区间， 还有步长（可以是负数）。注意指定区间时
range(n1,n2)输出的数据为 n1---n2-1

# 注意如果步长刚好“合适”，最后一个数不输出，如下：
for i in range( 5, 9, 2 ):
    print ( i )  # 输出为 5，7。9 不输出


# 结合 range()和 len()函数，这个例子要牢记!!!
a = [ "Google", "Baidu", "Taobao" ]
for i in range( len(a) ):  # len(a) = 3, range(3) = [ 0, 1, 2 ]
    print ( i, a[i] )

# 还可以使用 range()函数创建列表
# list(range(5))

#python7: break：相当于 Fortran 中的 exit
# break 语句跳出 for 和 while 的循环体。如果从 for 和 while 循环中终止，
任何对应的循环 else 块将不执行
for letter in 'Runoob':
    if letter == 'b':
        break
    print ( " 当前字母为： ", letter )

var = 10
while var > 0:
    print ( " 当前变量值为： ", var )
    var -= 1
    if var == 5:
        break
```

```
else:
    print ( " Good bye! " )


# python8: continue 相当于 fortran 中的 cycle

# continue 用来告诉 python 跳过当前循环块中的循环语句，然后执行下一轮

循环

for letter in 'Runoob':
    if letter == 'o':
        continue
    print ( " 当前字母为： ", letter )


var = 10
while var > 0:
    var -= 1
    if var == 5:
        continue
    print(" 当前变量值为： ", var)
else:
    print ( " Good bye! " )
# 循环语句中可以有 else 子句，它在穷尽列表（for 循环）或条件变为 false

（while 循环）

# 导致循环终止时被执行，但循环被 break 终止时不执行


# 查询质数

for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, '等于', x, '*', n//x)
            break
    else:
        # 循环中没有找到元素

        print(n, ' 是质数')
```

TestLoop

```
# 九九乘法表

# while 循环写九九乘法表

print ( " 九九乘法表： "  )

i = 1
while i <= 9:
  j = 1
  while j <= i:
    mult = i * j
    print ( " %d * %d = %d" %( j, i, mult ), end = " " )
    j = j + 1

  i = i + 1
  print ( " \r " )


# for 循环写九九乘法表

print ( " 九九乘法表： " )

numLoop = 10
for i in range( 1, numLoop ):
  for j in range( 1, i + 1 ):
    mult = i * j
    print(" %d * %d = %d" % (j, i, mult), end=" ")

  print ( " \r " )


# 使用 enumerate 遍历

sequence = [ 12, 34, 34, 23, 45, 76, 89 ]
for i, j in enumerate( sequence ):
  print ( i, j )


# 上面代码等同于下面代码

sequence = [ 12, 34, 34, 23, 45, 76, 89 ]

for i in range( len( sequence ) ):  # 要熟练使用 range()和 len()的

联合使用
```

```
  print ( i, sequence[i] )

# 求和
a = sum( range( 0,101 ) )
print ( " 0 - 100 之和为: ", a )


# pass 的作用：pass 只是为了防止语法错误
if a > 1:
  pass   # 如果没有内容，可以先写 pass，但是如果不写 pass，就会有语法
错误


# python1：显示 10 次"happy"
i = 1
while i <= 10:
  print ( i, " happy! " )
  i = i + 1

# python2: 计算 2+4+6+8+10
sum = 0
for i in range( 2, 11, 2 ):
  print ( " i = ", i )
  sum += i
print ( " 2 + 4 + 6 + 8 + 10 = ", sum )

# 用 while 循环写上面代码
i = 2
sum = 0
while i <= 10:
  sum = sum + i
  i = i + 2
print ( " 2 + 4 + 6 + 8 + 10 = ", sum )

# python3: 负数递增
for i in range( 10,0,-1 ):  # 从 10 输出到 1
```

```
  print ( i )


# python4: 内层循环输出

for i in range( 1, 4 ):
  for j in range( 1, 4 ):
    print ( " i, j = ", i, j )


# python5: 猜体重

import math
eps = 0.1
real_weight = 53.11

print ( " 请输入猜测值： " )

guess_weight =  float(input())
while math.fabs( real_weight - guess_weight ) > eps:

  print(" 请重新输入猜测值: ")

  guess_weight = float(input())
else:
  print(" you are rigth! ")


# for 循环与 while 循环中有 else 时，执行完 else 后面的代码，整个循环就

结束

sites = [ "Baidu", "Google", "Runoob", "Taobao" ]
for site in sites:
  if site == "Runoob":
    print ( " luk " )

    break # 跳出整个循环


  print ( " 元素： ", site )

else:

  print ( " 没有循环元素 " )


print ( " 循环完毕！ " )


# python6: 测试 continue，相当于 fortran 中的 cycle
```

```
nfloor = 9
i = 0
while i <= 9:
  i = i + 1
  if i == 4:
    continue
  print(" i = ", i)


# 测试 break，相当于 Fortran 中的 exit

import math
eps = 0.1
real_weight = 53.11
a = 1

print ( " 请输入猜测值： " )

guess_weight =  float(input())
while a == 1:
  if math.fabs( real_weight - guess_weight ) > eps:

    print(" 请重新输入猜测值: ")

    guess_weight = float(input())
  else:
    print(" you are rigth! ")
    break
```

IteratorGenerator

1. 迭代器

迭代器是一个可以记住遍历位置的对象

迭代器对象从集合的第一个元素开始访问，直到所有元素被访问完结束。迭代器只向前

迭代器有两个基本方法：iter()和 next()

字符串，列表或元组对象都可以用于创建迭代器

```
list = [1, 2, 3, 4]
it = iter( list ) # 创建迭代器
for x in it:
  print ( x, end = " " )
print ( "\r" )
```

使用 next 函数

```
import sys
list = [1, 2, 3, 4]
it = iter( list ) # 创建迭代器
while True:
  try:
    print ( next(it), end = " " )
  except StopIteration:
    sys.exit()
print ( "\r" )
```

Function

函数规则：

函数代码块以 def 关键词开头，后接函数标识符名称（函数名）和圆括号()

任何传入参数和自变量必须放在圆括号中间

函数内容以冒号起始，并且缩进

return [表达式]结束函数。不代表达式的 return 相当于返回 none

```
# python1
def hello():
  print ( " hello, world! " )


hello ( )  # 自定义函数用于输出信息


# python2
import math
def area( r ):
  s = math.pi * r * r
  return s

print ( " area = ", area(1.0) )

def print_luk( name ):
  return name
print ( " name = ", print_luk( "luk" ) ) # 这样调用，return 后面
```

必须跟返回值，否则返回 none

```
# print_luk( "luk" )  # 这样调用函数，因为函数 print_luk 中无执行语
```

句，就算有返回值，也无输出

```
# python3
# python 传不可变对象
def ChangeInt(a):
  print ( ' before a = ', a )
  a = 10
```

```
  print ( ' after a = ', a )
  return a
b = 2
print ( " func value: ", ChangeInt(b) )
print ( " b = ", b )
```

经过测试，建议写 print (函数名(参数))，即 print (ChangeInt(b))

如果写 ChangeInt(b)不会输出信息

python 传入可变参数

```
def Changme( list1 ):
  list1.append( 1 )  # append()函数里面的参数为 1 个整数，或列表（字
```

典等）

```
  print ( " in : ", list1 )
  return list1

list2 = [ 10, 20, 30, 40 ]
Changme( list2 )  # 如果写成 Change( list2[:] )，则两者的 id 不同
print ( " out : ", list2 )
```

特别注意

```
def Changeme( mylist ):
  mylist = [1,2,3,4];
  print ( " In function: ", mylist )
  return
mylist = [10,20,30,40];
Changeme( mylist );
print ( " Out function: ", mylist )
```

这样写也不会影响外部的 mylist，这是因为外面的 mylist 为全局变量，

Changeme 函数中的 mylist 为局部变量，两者 id 不同

python 传入时不指定参数顺序

```
def printinfo(name, age):
  print(" name: ", name);
  print(" age: ", age);
```

```
  return;
# 调用 printinfo 函数
printinfo( age = 50, name = "runoob" )  # 如果写成 printinfo( 50,
'runoob' )则要按顺序

#  python 传入默认参数，在调用函数时，如果没有传递参数(fortran 中的实
参)，则会使用默认参数。
def printinfo( name, age = 35 ):
  print ( " name: ", name )
  print ( " age: ", age )
  return
printinfo( age = 50, name = 'luk' )
print ( "--------------------" ) # 当函数中虚参有数值，并且在程序
内部过程有输出，只写 printinfo()也会输出 35
printinfo( name = 'luk' )

# python 传入可变长度变量
def printinfo( arg1, *vartuple ):
  print ( " arg1: ", arg1 )
  for var in vartuple:
    print ( " var: ", var )
  return

printinfo ( 10 )
printinfo ( 10, 20, 30 )

# python 匿名函数
# 1. python 使用 lambda 来创建匿名函数
# 2. lambda 只是一个表达式，函数体比 def 简单
#  3. lambda 函数拥有自己的命名空间，且不能访问自己参数列表之外或全局
命名空间里的参数
```

```
sum1 = lambda arg1, arg2: arg1 + arg2
print ( " 10 + 20 = ", sum1( 10, 20 ) )
print ( " 20 + 30 = ", sum1( 20, 30 ) )


# return 语句

def sum1( arg1, arg2 ):
  total = arg1 + arg2
  print ( " in function: ", total )
  return total

print ( " out function: ", sum1( 10, 20 ) )


# 变量作用域

# python 中，程序的变量并不是哪个位置都可以访问的，访问权限决定于这个

变量在哪里赋值

#  变量的作用域决定了在哪一部分程序可以访问哪个特定的变量名称。python

的作用域一共有四种

# 1. L(Local) 局部作用域

# 2. E(Enclosing) 闭包函数外的函数中

# 3. G(Global) 全局作用域

# 4. B(Bulit-in) 内建作用域


x = int(2.9) # 内建作用域

g_count = 0 # 全局作用域

def outer():

  o_count = 1 # 闭包函数外的函数中

  def inner():

    i_count = 2 # 局部作用域


# python 中只有模块(module)，类(class)，以及函数(def,lambda)，才会
```

引入新的作用域

其他代码块(如 if/elif/else/、try/expect、for/while 等)是不会引入新的作用域，也就是说这些语句内定义的变量，外部也可以访问

下面例子中，msg 变量定义在 if 语句块中，但外部还是可以访问的

```
if True:
  msg = ' I am lukailiang! '

print ( msg )
```

如果将 msg 定义在函数中，则它就是局部变量，外部不能访问

```
def test():
  msg1 = ' error! '
```

print (msg1) 这句报错，因为在全局中没定义变量 msg1

这里值得注意一下，将局部变量与全局变量的命名最好不一致，如果一致，有时会混淆

例如，上面如果在函数 test 中定义为 msg，然后再 print(msg)，如果全局中定义了 msg，就会输出全局中 msg 的值，而不是函数 test 中 msg 的值，这里注意一下

全局变量与局部变量

定义在函数内部的变量拥有一个局部作用域，定义在函数外的拥有全局作用域

局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问

```
total = 0; # 这是一个全局变量
```

```
def sum( arg1, arg2 ):
  total = arg1 + arg2; # total 在这里时局部变量
  print ( " In function total is ", total )
  return total;

# 调用函数 sum，输出函数执行语句结果
sum(10,20)
print ( " Out function total is ", total )

# global 和 nonlocal 关键字
num = 1
def fun1():
  global num  # 说明 num 是全局变量和局部变量，意思是局部变量 num 改变后，全局变量中的 num 也会改变
  print ( " before: num = ", num )
  num = 123 # 修改 num 的值
  print ( " after: num = ", num )

# 调用函数
fun1()

#  如果要修改嵌套作用域(enclosing 作用域，外层非全局作用域)中的变量则需要 nonlocal 关键字
def outer():
  num = 10
  def inner():
    nonlocal num  # nonlocal 关键字声明
    num = 100
    print ( " num =  ", num )
  # 调用函数 inner
  inner()
  print ( " num = ", num )
# 调用函数 outer
```

```
outer()

# lambda 匿名函数也是可以用“关键字参数”进行参数传递，为了不记混淆，建
议在调用函数的同时进行指定，例如 g(x=1,y=2)
g = lambda x, y: x**2 + y**2
print ( " g(2,3) = ", g(2,3) )  # 默认为 g(x=2, y=3)
print ( " g(y=3,x=2) =  ", g(y=3,x=2) )  # 不选择默认时，需要指定
# 传入一个参数
g = lambda x=0, y=0: x**2 + y**2
print ( " g(2) = ", g(2) )  # 默认为 g(x=2)，y 值为函数中 y 的值
print ( " g(y=3) = ", g(y=3) )  # 此时需要指定

#  下面这个例子证明全局变量在局部变量中仍然起作用（但是局部改变后并不
影响外部的值），反之则不行
# 如果想通过改变局部变量的值，而改变全局变量的值，需要使用 global
b = 1
def ss():
  a = 1 + b
  print ( " a = ", a )
# 第一次调用函数 ss()
ss()
# 该变 b 的值
b = 10
# 再次调用 ss()
ss()

# 严重注意 :函数内能访问全局变量，但不能更新（修改）其值，除非使用 global
# 例如
a = 10
```

```
def test():
  a = a + 1
  print ( " a = ", a )
# test()
```

这种情况报错,主要原因还是函数中局部变量 a 没有声明(fortran 为初始化)

或是非法修改全局变量 a 的值，记住，只能访问不能修改

```
a = 10
def sum(n):
  n = n + a  # 访问全局变量的值

  # 如果加下面一句会报错

  # a = 1，不能修改全局变量的值

  print ( " a = ", a, end = "," )
  print ( " n = ", n )
sum(3)


# 下面代码是变量作用域的例子

# 1. 局部作用域

x = int(3.3)

x = 0
def outer():
  x = 1
  def inner():
    x = 2

    print ( " x = ", x )  # 执行结果为 2，因为在函数 inner 内部找到
```

了变量 x

```
  inner()
outer()


# 2. 闭包函数外的函数中

x = int(3.3)

x = 0
```

```
def outer():
  x = 1
  def inner():
    i = 2
    print ( " x = ", x )  # 在局部变量中找不到，去局部外的局部寻找
  inner()
outer()


# 3. 全局作用域
x = int(3.3)
x = 0
def outer():
  o = 1
  def inner():
    i = 2
    print ( " x = ", x )  # 在局部(inner 函数),局部的局部(outer 函数)中都没找到，去全局找
  inner()
outer()


# 4. 内建作用域
x = int(3.3)
g = 0
def outer():
  o = 1
  def inner():
    i = 2
    print ( " x = ", x )
  inner()
outer()


# 寻找列表中绝对值最大的下标
myList = [-1,2,-3,4,6,-5]
absList = ( map(abs, myList) ) #对于 Python3.x 需要用 list 函数对 map 的返回值转换为列表
absList = list ( absList )
```

```
print (absList)
print ( absList.index( max( absList ) ) )
```

DataStructure

```
# 列表的使用方法

# 1. 添加元素

list1 = [1,2,3,4];

list1.append(5);  # list1.append(x)在列表后添加元素 x，这种方法每次只能添加一个

print ( " list1: ", list1 )

list2 = [1,2,3,4];

list2[len(list2):] = [5,6,7];  # 这种也是向列表后添加元素，这种方法可以一次添加多个元素

print ( " list2: ", list2 )

# 2. 列表扩充

list3 = list2[:];  # 这里要特别注意，如果写成 list3 = list2，则在后面改变两者中的任一个，另一个也会改变

list3.extend( list2 );  # 相当于 list3[len(list3):] = list2

print ( " list3: ", list3 )

# 3. 在指定位置插入一个元素

list3.insert(0,0); # 在 list3 第一个位置插入 0

list3.insert(len(list3),100)  # 在 list3 最后一个位置插入 100，相当于 list3.append(100)和 list3[len(list3):] = [100]

print ( " list3: ", list3 )

# 4. list.remove(x):删除列表中值为 x 的第一个元素。如果没有这样的元素，就会返回一个错误
```

```
list3.remove(1);
print ( " list3: ", list3 )

# 5. 从列表指定位置删除元素，并将其返回。如果没有指定索引，list.pop()
返回最后一个元素，元素随机从列表中删除
list3.pop(len(list3)-1);  # 删除最后一个元素
print ( " list3: ", list3 )

# 6. list.clear()移除列表中的所有项，等于 del list[:]，注意 del list
则是删除整个列表
list3.clear()
print ( " list3: ", list3 )
print ( " list3 has been cleared! " )

# 7. list.index(x):返回列表中第一个(如果有多个 x)值为 x 的索引。如果
没有该元素就会返回一个错误
print ( " list2: ", list2 )
print ( " list2.index(3): ", list2.index(3) )

# 8. list.count(x): 返回 x 在列表中出现的次数
num = list2.count(2)
print ( " list2.count(2) ", num )

# 9. list.reverse(): 倒排列表中的元素
list2.reverse() # 不带参数的语句要单独写一行
print ( " list2.reverse() ", list2 )

# 10. list.sort(): 排序
list2.sort()
print ( " list2.sort(): ", list2 )

# 11. list.copy(): 复制列表 list，相当于 list1 = list[:]
```

```
list3 = list2.copy();
print ( " list3: ", list3 )

# 将列表当堆栈使用

# 堆栈:最先进入的元素最后一个被释放（先进后出）

# 用 append()的方法可以把一个元素添加至堆栈顶，用不指定索引的 pop()方

法可以把一个元素从堆栈顶释放

stack = [3,4,5]

stack.append(6); # 尾部添加元素 6

stack.append(7); # 尾部添加元素 7

print ( " stack: ", stack )

stack.pop() # pop()不指定索引位置时，释放最后一个元素

print ( " stack: ", stack )

# 不建议将列表当作队列使用


# 列表推导式


vec = [1,2,3];

vec = [3*x for x in vec] # 类似于 fortran 的[ i, i = 1, 3 ]

print ( " vec = ", vec )

vec = [ [x,x**2] for x in vec ]
print ( " vec = ", vec )
print ( " vec[1][1] = ", vec[1][1] )

#  用 x.strip()函数去掉列表中字符串两端的空格，可用于后面文件的批量处

理

freshfruit = [ '  banana', ' loganberry ', 'passion fruit  ' ];
freshfruit = [ x.strip() for x in freshfruit ]
print ( " freshfruit: ", freshfruit )
```

```
# 用 if 子句作为过滤器，像 Fortran 中的 forall 语句
# vec = [2,3,4]
# 这里要注意，如果 vec 是一维向量，则下面语句写 if>某个数
# 如果 vec 是一个二维数组，2*x 作用于子列表，表示输出两次，并不是给每个
数字乘以 2，如果不满足，则清楚不满足条件的列表元素
vec = [ 2*x for x in vec if x > [3,10] ] # 2*list，重复输出 list
两次
print ( " vec = ", vec )

vec1 = [2,4,6]
vec2 = [4,3,-9]
vec = [x*y for x in vec1 for y in vec2]  # vec1 中的每个元素都和
vec2 中的每个元素相乘
print ( " vec = ", vec )
vec = [x+y for x in vec1 for y in vec2] # vec1 中的每个元素都和
vec2 中的每个元素相加
print ( " vec = ", vec )
# 对应元素相加
vec = [vec1[i]*vec2[i] for i in range(len(vec1))] # range()函数
真是好用
print ( " vec = ", vec )


# 重点：嵌套列表解析
matrix = [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ]
matrix1 = matrix
# 求其转置
trans = []
tmp = []
for i in range(4):
```

```
  for j in range(3):
    tmp.append( matrix[j][i] )
  trans.append(tmp)
  tmp =  []
print ( trans )

# 也可以用下面的代码

# trans = []
# for i in range(4):
#   tmp = []
#   for row in matrix:
#     tmp.append(row[i])
#   trans.append(tmp)

# 或者

# print ( [[row[i] for row in matrix1] for i in range(4)] )


# del 语句

# 使用 del 语句可以从列表中依据索引来删除一个元素。

a = [-1,1,66.25,333,333,1234.5]
del a[0]
print ( " a: ", a )
del a[2:4]
print ( " a: ", a )

del a[:] # 清空列表

print ( " a: ", a )

# 语句 del a 删除实体变量，引用报出错


# 元组和序列

# 元组在整体输出时总是有括号的，在输入时括号可有可无

# 输出元组中的单个元素时，没有括号

t = 12, 54, 'hello'
print ( " t[0]: ", t[0] )
print ( " t: ", t )
u = t, ( 1, 2, 3, 4, 5 )
print ( " u: ", u )
```

```
# 集合：是一个无序不重复元素的集合。基本功能包括关系测试和消除重复元素

# 可以使用大括号{}创建集合。注意：如果要创建一个空集合，必须使用 set()

而不是{}

# {}是创建一个空字典

basket = { 'apple', 'orange', 'pear', 'apple' }

print ( " basket: ", basket ) # 删除重复的元素

if 'orange' in basket:  # 检测成员

  print ( " orange in basket! " )
else:
  print ( " orange not in basket! " )


a = set( 'abcdaedf' ) # 使用 set()函数将其转化为集合，删除重复元素

b = set( 'alacazam' )
print ( " a: ", a )
print ( " b: ", b )

print ( " a - b ", a - b ) # 在 a 中的字母，但不在 b 中

print ( " a | b ", a | b ) # 在 a 或 b 中的字母，并集

print ( " a & b ", a & b ) # 在 a 和 b 中都有的字母

print ( " a ^ b ", a ^ b ) # 在 a 或 b 中的字母，但不同时在 a 和 b 中

# 集合也支持推导式

a = { x for x in 'abracada' if x not in 'abc' }


# 字典，添加、删除元素都是对键赋值或是删除

tel = { 'jack': 4098, 'sape': 4139 }

tel['guide'] = 4127 # 添加元素

print ( " tel: ", tel )

del tel['sape'] # 删除元素

tel['irv'] = 4127
print ( " tel: ", tel )
```

```
list1 = list( tel.keys() ) # 将字典的键值转化为列表
print ( " list1: ", list1 )
list1 = sorted( tel.keys() ) # 用 sorted()对字典的键值排序
print ( " list1: ", list1 )
list1 = sorted( tel.values() ) # 用 sorted()对字典的值排序
print ( " list1: ", list1 )

# 字典推导式
{ x: x**2 for x in (2, 4, 6) }

# 使用关键字参数指定键值对
dict( sape = 4123, guido = 1234, jack = 2435 )

int() # 整数
float() # 浮点数
str() # 字符串函数
tuple() # 元组函数
list() # 列表函数
dict() # 字典函数
set() # 集合函数

# 遍历技巧，重点
# 在字典中遍历时，关键字和对应的值可以使用 items()方法同时读出来
knigths = { 'luk': 24, 'xxx': 25 }
for k, v in knigths.items():
  print ( k, v )

# 在列表中遍历时，索引值和对应值可以使用 enumerate()函数
```

```
list1 = ['a','b','c','d']
for i, v in enumerate(list1):
  print ( ' i = ', i, ',', ' v = ', v )


# 要反向遍历一个 range()序列，首先指定这个序列，然后调用 reversed()函
数
for i in reversed( range(0,11) ):
  print ( " i = ", i )


# 同时遍历两个或多个列表，可以使用 zip()组合
que = [ 'name', 'quest', 'favorite color' ]
ans = [ 'luk', 'the holy grail', 'blue' ]
for q, a in zip( que, ans ):
  print ( "What is your {0}? It is {1}" .format(q,a) )
  # 也可写成
  print ( "What is your %s? It is %s" %(q,a) )
```

Module

把定义的函数和变量放在文件中，为一些脚本或者交互式的解释器实例使用，这个文件被称为模块

模块是一个包含了你定义的函数和变量的文件，后缀名为.py。模块可以被别的程序引入，以使用该模块中的函数等功能。

import 语句

想使用 python 源文件，只需在另一个源文件里执行 import 语句，语法如下：

import module1[,module2[,...moduleN]]

当解释器遇到 import 语句，如果模块在当前的搜索路径就会被导入

例子

导入模块

```
import support
# 调用模块里包含的函数
support.print_func( " luk " )

import fibo
print ( dir( fibo ) )   # 用 dir()函数可以找到模块内定义的所有名称


fibo.fib( 1000 )
print ( fibo.fib2( 100 ) )

print ( ' fiboName: ', fibo.__name__ )


# 如果打算经常使用一个函数，可以把它赋给一个本地名称
testfib = fibo.fib
testfib( 500 )


# from...import 语句
```

python 的 from 语句让你从模块中导入一个指定的部分到当前的命名空间中，

语法如下：

```
# from modname import name1[,name2[,...nameN]]
# 例如要导入模块 fibo 中的 fib 函数
# from fibo import fib, fib2
# fib ( 500 )
#  上面的声明不会把整个 fibo 模块导入到当前的命名空间中，它只会将 fibo 里的函数 fib 引入进来

# from ...import*语句
# 把一个模块的所有内容全都导入到当前的命名空间
# from modname import* 这种声明不该被过多使用

# 每一个模块都有一个__name__属性，当其值是'__main__'时，表示该模块自身在运行，否则是被引入

import using_name
```

Support 模块

```
def print_func( par ):
  print ( " hello: ", par )
  return
```

---

fibo 模块

```
# 斐波那契(fibonacci)数列模块
s = 1
def fib(n):
  a, b = 0, 1
  while b < n:
    print ( b, end = ' ' )
    a, b = b, a + b
  print ()
```

```
def fib2(n):
  result = []
  a, b = 0, 1
  while b < n:
    result.append(b)
    a, b = b, a + b
  return result
```

```
# __name__属性

# 一个模块被另一个程序第一次引入时，其主程序将运行。

#  如果我们想在模块引入时，模块中的某一程序块不执行，我们可以使用——

name__属性来使该程序块仅在该模块自身运行时执行


# Filename: using_name.py
if __name__ == '__main__':
  print ( ' 程序自身在运行 ' )
else:
  print ( ' 我来自另一模块 ' )
```

InputOutput

```
s = 'hello, luk!'
print ( str(s) ) # 用 str()函数输出的字符串不带''，用 repr()函数输出的字符串带''
print ( repr(s) ) # 用 str()函数和 repr()函数输出数字都不带''

x = 10 * 3.25
y = 200 * 200
s = 'x value is: ' + str(x) + ', y value is: ' + repr(y)
print ( s )

# repr()函数可以转义字符串中的特殊字符
hello = 'hello, luk\n'
print ( str(hello) ) # str()函数会将\r 认为是换行符，repr()函数可以原样输出
print ( repr(hello) )

# 输出平方与立方表
for x in range( 1, 11 ):
  # rjust()：将字符串靠右，并在左边填充空格
  # ljust()和 center()类似
  print ( repr(x).rjust(2), repr(x*x).rjust(3), end = ' ' )
  print ( repr(x*x*x).rjust(4) )

# 另一个方法是 zfill()，它会在数字的左边填充 0
s = '12'.zfill(5) # 用 5 个位宽输出，左边三位是 0
print (s)
s = '-3.14'.zfill(7) # 负号(-)与小数点(.)也占位宽
print (s)
s = '3.141592653'.zfill(5)  # 超过位宽，原样输出
```

```
print (s)


# str.format()的基本使用
print ( '{} {}!' .format( 'https://', 'www.baidu.com' ) )
# 大括号{}会被 format()中的参数替换掉，里面的字符会照样输出


# 在括号中的数字用于指向传入对象在 format 中的位置
print ( ' {0} and {1} ' .format( 'Google', 'baidu' ) )
print ( ' {1} and {0} ' .format( 'Google', 'baidu' ) )


# 如果在 format()中使用了关键字参数，那么他们的值会指向使用该名字的参
数
print ( ' {name}: {site} ' .format( name = 'baidu', site =
'www.baidu.com' ) )


# 位置及关键字参数可以任意组合，0，1，other 与输出参数对应，后面的数
字为输出的位宽
print     (     '     {0:10},     {1:10}     and     {other:10}.
' .format('Google','baidu',other = 'taobao') )

for x in range(1,11):
  print ( '{0:2d} {1:3d} {2:4d}' .format(x, x*x, x**3) )


import math
print ( " pi: {0:.3f} " .format( math.pi) ) # 0 对应的是 pi，.3f
是格式化输出


table = { 'Google': 1, 'baidu': 2, 'Taobao': 3 }
for name, number in table.items():
  print ( '{0:6} ==> {1:4d}' .format(name,number) )


# 读和写文件
# open ( filename, mode )
```

filename: 是一个包含了要访问的文件名称的字符串值

mode: 决定打开文件的模式：只读，写入，追加等。

mode 为 r 时，只读；r+ 读写。 指针都在文件开头

mode 为 w 时，写入；w+ 读写，如果该文件已存在则将其覆盖，如果不存在， 创建该文件

mode 为 a 时，打开一个文件用于追加。如果该文件存在，文件指针放在文件末尾，新的内容会被写在已有内容之后。如果没有文件，创建新文件写入

mode 为 a+ 时，打开一个文件用于读写。如果该文件已存在，文件指针会放在文件的结尾。如果该文件不存在，创建新文件用于读写

```
f = open( 'foo.txt', 'w' )
f.write( "python is a good language!\nbut fortra more better!" )
f.close()
```

检测 f.readline()

f.readline()会从文件中读取单独一行。换行符为'\n'。f.readline()如果返回一个空字符串，说明已经读取到最后一行

```
f = open( 'foo.txt', 'r' )
i = 0
while True:
  str1 = f.readline()
  if str1 == '':
    break
  print ( str1 )
  i = i + 1
f.close()
print ( i )
```

检测 f.read()

为了读取一个文件的内容，调用 f.read(size)：读取一定数目的内容返回

```
# size 是一个可选的数字类型参数。当 size 省略或者为负数时，读取文件所有内容
f = open( 'foo.txt', 'r' )
str1 = f.read(1) # 读取第一个字符
print ( str1 )
str1 = f.read() # 读取剩下的字符
print ( str1 )

# 检测 f.readlines()
# f.readlines()将返回该文件中所包含的所有行
# 如果设置参数 sizehint,则读取指定长度的字节，并且将这些字节按行分割
f = open( "foo.txt", 'r' )
str1 = f.readlines()
print ( str1 )
f.close()

f = open( "foo.txt", 'r' )
str1 = f.readlines(1) # 读取第一行
print ( str1 )
f.close()

# 读取每行
f = open( "foo.txt", 'r' )
for line in f:
  print ( line, end = '' ) # 加上 end = ''，输出行之间无空行
f.close()

# f.write()：f.write(string)将 string 写入到文件中，可以返回写入的字符数
f = open( "foo.txt", 'a' )
f.write( '\n' )
num = f.write( '123' ) # 只能写入字符串
```

```
print ( num )
f.close()


# 如果写入的不是字符串，需要先进行转换
f = open( "foo1.txt", 'w' )
value = ['123', 14]
for i in value:
  print (i)
  s = str(i)
  f.write( s ) # 默认不换行
  f.write( '\n' )
f.close()

f = open( "foo2.txt", 'w' )
str1 = [ 'luk', 2, 3, 4, 5, 6 ]
for i in range( len(str1) ):
  s = str1[i]
  s = str(s)
  f.write( s ) # 往文件中写，默认不换行
  f.write('\n')
f.close
```