Python 3.5入门学习

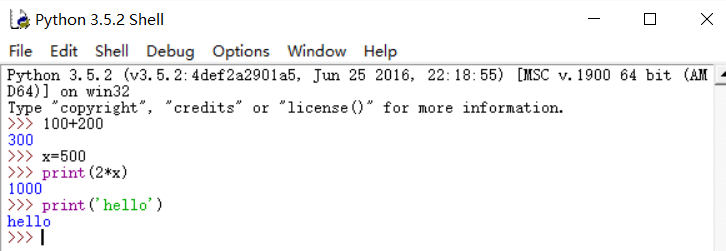
2017-10-23

# 1、基础

Python 语言1991年发布，是一种解释型/动态语言，源代码以.py为后缀名。创建者是Guido van Rossum，其因为喜欢Monty Python(1970年代英国一个著名六人喜剧团体)而将语言命名为Python。官网：<http://www.python.org> ,在下载时应选择和本机操作系统配套的版本。例如win7/10 32位版可下载安装python-3.5.2.exe， win7/10 64位版可下载安装python-3.5.2-amd64.exe。

python目前主要有2.7 和 3.x两个独立发展的版本，3.x不能完全兼容2.x，两者的很多语句/函数都不一样。**现在一般使用3.x版本。**

安装后运行python程序组中的 IDLE 可启动Python开发环境，如下图。在">>>"提示符后输入python指令，按回车键即可运行。



IDLE(Interactive DeveLopment Environment交互式开发环境)

练习时直接在提示符>>>后输入语句即可执行，这是交互模式，该模式下的语句都没有存盘。

x=500 #注意 **=** 是赋值，不是数学上的相等运算,表示将=右边的式子计算出结果，然后赋值给=左边的变量.

x= x + 500 # 运算后x是1000

python特点：面向对象，简单，内置丰富的数据结构（元组、列表、字典、集合等），健壮性（提供异常处理），跨平台性（模块可以先编译为与平台无关的二进制码，然后再解释执行），可扩展性（用C开发，可使用C扩展Python），动态性，应用广泛（数据库、网络、图形、数学计算、WEB等，有很多第三方库）。

例子:输入一个正整数N，求从1开始到N的累加和

num=input('请输入一个正整数:') #输入的数据是字符串

num=int(num) #调用int 函数，将字符串转为整数

s=0 #变量s存储和，定义时无需数据类型说明

k=1

while k<=num**:** #此处按语法要求，必须有冒号:

s=s+k #此行必须缩进（即行首应插入4个空格）

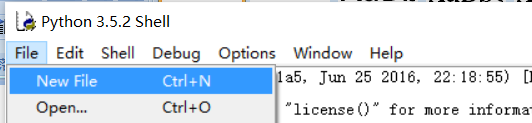
k=k+1 #此行必须缩进，且和上一行的缩进一致

print('1到'+str(num)+'的累加和=' , s) #两个输出内容中间用逗号隔开

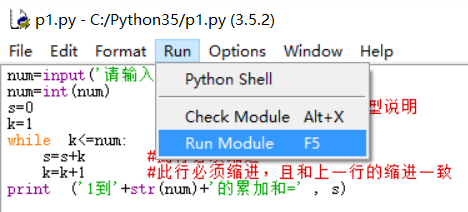
练习时将上述代码都输入在一个.py文件中，然后再执行。#后面的文字是注释，可以不用输入。

创建一个.py源文件的步骤如下：

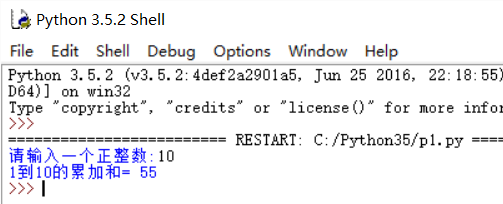
1. 在IDLE中选择菜单项“File/New File", 新建一个源代码录入窗口；



2. 在新窗口中录入上例的全部代码，以.py文件名(后缀名应为.py)保存，例如p1.py。录入代码时，注意while后面有冒号，其下的两行语句要缩进,在录入时可按Tab键自动产生4个空格的缩进。



3. 保存源程序后，选择Run菜单下的"Run Module F5"（如上图）或直接按F5键，即可运行程序。结果如下：



还可以直接双击程序名来运行。参考本文档【14杂项 /1运行.py程序】。

注意：**对于python，语句块缩进（即在该行最左侧插入若干空格）是必须的语法**。Python没有用花括号{ }或begin..end之类的定界符来包含语句块，而是用冒号和代码缩进来区分语句块。python中if, else, elif, while, for , def, class,try等语句后应有冒号，该冒号的下级语句必须要右缩进，语句块中同级的语句缩进应保持一致，如缩进量不一致将导致语法错。缩进结束或不再缩进就意味着上一语句块结束。python利用缩进迫使程序员写出统一、整齐并具有良好可读性的程序。

python语句**开头处**多一个或少一个空格都可能导致语法错误。语句的中间可以插入任意空格。例如:

>>> print(10,20) #错误，因为提示符>>> 和 print之间有空格

>>> print(10, 20) #正确，10 和 20之间插入任意个空格都可以

>>>if 10>5:

print('大于') # 错误,缺少空格缩进。因为print从属于if, 必须向右缩进

标准python风格是每个缩进级别使用4个空格。如果你用记事本这类编辑软件输入源程序，注意不要混用空格和TAB符来产生缩进。在python的IDLE中按TAB键将被自动替换为4个空格符，推荐用此方法以产生统一的缩进。

python没有类似C语言的main函数，python从第一条非函数的语句开始执行。程序中可以插入空行，python将忽视空行。空行有助于将大块代码分割，便于阅读。Python执行语句中所有的标点符号应采用英文标点符号，在输入程序时最好将输入法切换到英文，避免输入中文标点符号导致程序运行错误。在字符串和注释中可以使用中文标点。

Python行末可以使用分号;结尾，但不是必须的。如果在一行中有多条语句，则必须用分号分割。例如同一行写三条语句 x=100; y=200; z=10

行末可使用 \ 作为多行连接符,将一条语句s=x+y+z分为两行可写为：

s= x+y \

+z

注释符号有两种 # 和 ''' ：

1. #单行注释。
2. 如要多行注释，可用三个单引号''' 或双引号"""做前后定界符。例如

**‘’’**

多行注释行1

多行注释行2

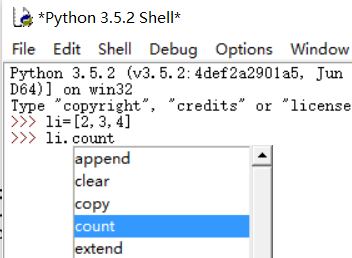
**‘’’**

python语句区分大小写，一般语句/函数都是小写。变量x和变量X是不同的。

在IDLE中可按 ALT+p / ALT+ n键前后翻滚调出已执行过的指令。

终止正在运行的程序: ctrl + c

对象名后输入“.”或按Tab键将出现成员提示，按空格键则选中提示的成员。如下图：



python内置的帮助help 和 dir：

help(para) : 返回有关para的帮助信息。如果para是一个字符串，则会自动搜索以para命名的模块或方法。如果para是一个对象，则会显示这个对象的类型帮助信息。例如：

help('print') #返回有关print 语句的帮助信息

help('time') #返回有关time 模块的帮助信息

help(str) #返回str数据类型的帮助信息

x=10

help(x) #返回整数的相关信息

import math # 引入 math 模块

dir(math) #列出math模块的所有函数

dir(\_\_builtins\_\_) #列出python内置函数清单

s='abc'

dir(s) # 列出字符串可调用的方法

li=[1,2,3]

dir(li) # 列出列表可调用的方法

print(abs.\_\_doc\_\_) #显示abs这个内置函数的简短说明

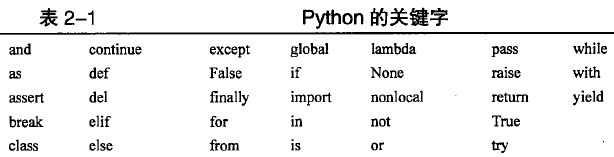
# 2、常用内置数据对象

x=123 ; 姓名='张三' ; flag=True

此处的x , 姓名 , flag 是变量名（也称为标识符名）。python3中变量名必须以英文字母或中文或下划线\_开头，不能数字开头，后面再跟上字符或数字。变量名中不能含有？- + 空格等特殊字符。 3x, ab?都不是合法变量名。

变量名区分大小写，X和x 是两个不同的变量。此处的 123 ， '张三' ， True 是不同类型的数据。

变量名字不能和python中已有的关键字名相同，即变量名不能使用下表中的保留关键字。if, and, True 都不是合法的变量名。



python中的数据类型分为如下几种：

（1）数值: python支持整数，浮点数（即小数），复数。

x=12 #整数

x=11.23 #浮点数

x=1.23e+12 #科学计数法浮点数。即1.23 X 1012

python中整数没有大小限制，只要内存允许，整数可以无限大。500\*\*10000不会溢出.

python的浮点型存在上限和下限，例如 500.0\*\*10000 (即500.0的10000次方, 此处500.0是浮点数)将溢出。

python定义变量时不需要说明数据类型，python会根据所赋的值自动确定变量的数据类型。

x=10+5j #复数

复数由实部和虚部组成，例如 3+2j (后缀j 表示虚部)。例如 -1开平方为1j

import cmath #引入复数模块

cmath.sqrt(-1) # -1开平方， 结果为 1j

(2)字符串: 'abc', "1122", '''334455'''用单引号/双引号/三个单（双）引号都可以做定界符，可以在单（双）引号的字符串中包含双（单）引号，以便能够输出引号。例如 "ab'c" ， 'ab"c' ，

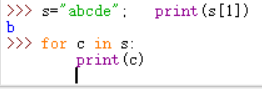
字符串可通过下标引用其包含的单个字符。

s="abcde"

print(s[1]) #输出s的第1个字符 b , 注意序号是从0开始

for c in s:

print(c) #依次输出每个字符



注：以上语句都可以在IDLE shell >>>中测试。在>>>提示符输入 for c in s:按回车，光标跳到下一行且自动缩进，再输入 print(c), 再回车后，光标又跳到下一行，如果还有语句可继续输入，如果不输入直接按回车，系统就执行刚输入的语句，如上图所示。

要让字符串跨越多个行，可用反斜线\作为上一行的最后一个字符。如：

s=" a very long string \

that spans two lines"

(3)列表 li= [1,2,3] 用中括号，列表元素可增/删，元素的值可修改

(4)元组 yz= (4,5,6) 用小括号，元组不能增/删，其元素不可修改

(5)字典 d= {'a':'apple', 'b':'banana','p':'pear'} 用花括号, 键-值对

(6)布尔型bool : 只有真假两个值 True, False (注意大小写)

x=5>3 #True

type(True) #显示<class 'bool'>

在python中, 0/False/None/空序列(如空字符串、空列表)等都表示假，非0和True表示真.

True==1 False==0

True + 4 #等于5, 因为True可视为整数1

(7) 集合 x=set([1,2,3]) ，集合中的元素不能重复

(8) NoneType类型 ：该类型中只有一个值None ，表示一个空对象(null), 空对象不能理解为0。

x=None

x is None #返回True

python中的变量无须事先显示声明，变量在第一次赋值时自动声明，变量无须指定数据类型。一个变量可以先后被赋予不同类型的值，其数据类型自动改变。程序员不用关心内存管理，系统自动回收垃圾内存。

# 3、操作符

+, - , \* , / #加减乘除

注意：在python2.x中整数/整数将只返回商的整数部分,例如 23/5 结果为4；浮点数相除可返回浮点商 , 23.0/5 结果4.6。在python3.x中整数/整数返回浮点商。

// #整除(只含商的整数部分) 23.6//5 为4.0 , 23//5 为 4

% #求余数 23%5 为 3

\*\* #幂运算 2\*\*4 （即2的4次方）为 16. 幂运算也可用函数pow完成，例如pow(2,4)

<, <= , >, >= #比较运算符

大小比较如5>=3 结果为True

1<=x and x<=10 python 也允许这样写： 1<=x<=10

比较时数据类型应相同

"3">4 #将报错， 字符串和整数不能比较

x==y 相等比较(注意相等比较是两个= , 只写1个=是赋值)

x!=y 不等比较

x or y 逻辑或 5>3 or 3>4

x and y 逻辑与 5>3 and 3>4

not x 逻辑非 not 5>3

成员测试

x in y 测试成员x是否在序列y中。例：2 in [1,2] 结果为True

x not in y 测试成员x不在序列y中。 2 not in [1,2] 结果为False

位运算（运算按二进制位）

x | y 按位或， 例如 2|4 即010|100=110 结果为6

x & y 按位与 ,例如 2&4即010&100=000 结果为0

x^y 按位异或，例如 2^4即010^100=110 结果为6

~ x 按位求反 例如 ~2 即 ~00000010 = 11111101, 此为-3的补码，所以屏幕上显示 -3

x<<y , x>>y x左移或右移y位

例： 2<<3 结果为16，相当于2乘以2的3次方

例： 8>>1 结果为4， 相当于8除以2

注：**上述位运算时都应按二进制位来运算**

Python 没有其它编程语言中常量的概念。

type(变量名) 可测试变量的数据类型。例如:

k=100

type(k) #<type 'int'>整型

s="abc"

type(s) #<type 'str'>字符串型

x=False

type(x) #<type 'bool'>布尔型

也可用isinstance测试某个值是否属于某种类型

isinstance(100, int) #True 100是整型

s="abc"

isinstance(s, str) #True s是字符串

isinstance(s, int) #False s不是整数

Python没有C语言中的字符类型, 只有字符串str类型，用单引号、双引号、三引号做定界符均可。'abc', "abc", '''abc'''

不同进制表示：

0xAF 以0x开头的数是十六进制 x=0xab #即十进制的 171

0o10 以0o开头的数是八进制 x=0o34 # 十进制的28

0b101 以0b开头的数是二进制 x=0b1100 # 十进制的12

python 支持x+=1的写法，表示x=x+1, 但不支持类似C++中的 x++

# 4、输入/输出数据

python3.x 输入数据使用input( )函数，函数后面一定要有括号，不能省略括号。所有的输入都视为字符串，输入后的数据应根据需要转换为其它数据类型。

x=input("please input x=") #假设输入3，则x="3"

print(x+5) #会报错, 因为x 是字符串 "3",不能和数值5相加

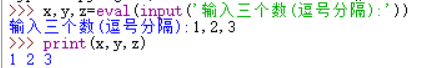
print(int(x)+5) #用int函数将字符串转换为整数后再相加

print(float(x)+5) #用float函数将字符串转换为浮点数再相加

如果用户不输入任何文本，只按回车键，则x 为空字符串 x=''

一次性输入多个变量值的方法

x,y,z=eval(input('输入三个数(逗号分隔):'))



执行后x=1, y=2, z=3

解释：输入的1,2,3 是一个字符串"1,2,3", eval函数将此字符串转换为一个有3个数的元组(1,2,3)， 再将此元组中的各个值赋值给x,y,z 变量。

python使用print( )输出信息。

print('a','b','c') # 输出a b c 每个数据默认间隔一个空格

print('a','b','c', sep='.') # 指定.为分隔符， 所以输出a.b.c

print('a','b','c', sep='') # 指定''空字符串为分隔符, 输出abc ,中间无空格

默认情况，每次print后将换行。 如果不想换行，可加end=''的参数

在一个.py文件中包含如下两行，则输出abc

print('a', end='') # end='' 的意思是不换行

print('bc') #将输出abc . 前后两行的print没有换行

或

>>>print('a', end=''); print('bc')

abc

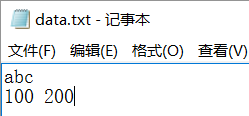
#将内容直接输出保存到文件data.txt中

f=open('data.txt', 'a') #打开文件， 'a'表示按添加模式打开

print('abc', file=f ) #指定将内容输出到 f 文件中

print(100, 200 , file=f ) #将100,200 输出到f 中

f.close( ) #关闭文件 此时data.txt 中含有上两行print的输出内容, 如下图所示：



在>>>交互模式下不用print语句，只写变量名也可显示出变量内容

>>>x=10

>>>x

10 #输出10

但在.py程序中，必须用print(x)才可以输出内容，不能只写x

**python的赋值语句**比较灵活

x=100 #注意=是赋值，不是数学上的相等运算

x=y=z=200 #x, y, z都是200

x,y,z=1 ,2,3 #x, y, z分别为1,2,3

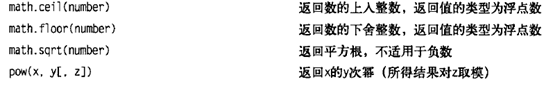
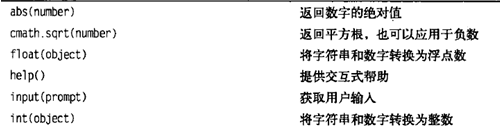
x+=2 #增量赋值等同于x=x+2

x='ab'\*4 # x='abababab' 将字符串ab重复4次

x,y=y,x # x, y交换值

# 5、常用函数

函数可视为完成特定功能的语句集合。编程语言通常会提供很多系统函数供用户调用。用户也可以编写自定义函数，供以后编程中调用,以实现代码复用。下表是几个常用的函数：



一些内置函数（如abs, round, pow, int, hex, bin等）可以直接调用。

abs(-10) # 求绝对值 ,结果为 10

pow(2,5) # 2的5次方 ， 结果为 32

help(abs) #返回abs函数的帮助信息

x, y=divmod(23 , 5) #同时返回商4和余数3 ， x=4 y=3

很多函数不是内置的，而是包含在其它模块中，必须先用import引入函数所在的模块才能调用函数。例如上表中math.ceil包含在math数学模块中, cmath.sqrt 包含在cmath复数模块中，使用前需先引入这两个模块。

import math, cmath #先引入两个模块, 才能使用sqrt函数

help(math.sqrt) #返回math.sqrt函数的帮助信息

math.sqrt(3) # 3开平方

cmath.sqrt(-1) # -1 开平方, 得到复数 1j (尾部字母j表示复数)

(1)字母和ASCII码之间的转换

ord('a') #返回'a'字符的ASCII码 97

chr(97) #根据ASCII码返回字符 'a'

for x in range(97,123):

print(x, chr(x)) #依次输出各个小写字母'a','b',… 'z'

for x in range(0x2600, 0x2700):

print(x, chr(x)) #输出0x2600 到0x2700的unicode编码对应的符号

(2)进制转换

x=100

hex(x) #将十进制100转16进制串 0x64

oct(x) #将十进制100转八进制串0o144

bin(x) #将十进制100转二进制串0b1100100

int('12' , 16) #字符串"12"按十六进制看待, 转换得到十进制数18

int('12') #字符串"12"默认按十进制看待, 转换得到十进制数12

y=100

format(y,'x') # 将变量y按 'x'格式( 16进制)输出，结果为 64

format(y,'b') # 按 'b' 格式(二进制) , 结果为 1100100

(3)四舍五入

round(8.5) # 返回8 。 python按银行家圆整法处理，小数部分正好为 .5时，如前一位为奇数则入，前一位为偶数则舍。此处前一位为8，舍弃

round(9.5) # 返回10 此处前一位为奇数，所以入

round(8.51) # 返回9

round(3.1415 , 2) #四舍五入保留2位小数，值为3.14

round(3.1415 , 3) #四舍五入保留3位小数，值为3.142

round(23456, -2) # -2表示四舍五入到小数点前2位，即百位， 结果23500

int(5.6) #结果5， int函数是直接丢弃小数部分,不四舍五入

math.trunc(5.6) #结果5， 丢弃小数部分

math.ceil(5.6) #结果6， 取>=5.6的最小整数

math.floor(5.6) #结果5， 取<=5.6的最大整数

(4)数据类型转换。用 int , float 将字符串转数值，用str将数值转字符串

x=int('56') #字符串'56'转十进制整数,得到 56

x=int('ab', 16) #字符串'ab'按十六进制转换得到十进制数171

x=int('100', 2) #字符串'100'按二进制转换得到十进制数4

f=float('56.2') #字符串转浮点数， 56.2

s=str(100) #数值转字符串，得到"100"

(5)针对序列的统计函数

序列是指元组、列表、字符串这样的数据类型，可以按下标存取其中的元素。

s=[1,2,3,4,5] # s 是列表

len(s) #返回对象s包含的元素个数, 长度为5

sum(s) #求序列s的和 15

max(s) #返回序列s的最大值 5

min(s) #返回序列s的最小值 1

(6) range函数

range(0,10000) :在python3中产生一个range对象，该对象含0到9999的整数（不含结尾的10000），但在内存中并没有真正产生这些数，只在以后实际需要时才逐一生成这些数。这样做的目的是节省内存。

for x in range(0,100):

print(x) #每次从range中自动取得1个数, 打印0,1，…99

li=list(range(0,10000)) #在内存中实际生成 0,1,2,…9999的列表

>>>range(1,101)

range(1, 101) # 产生一个range对象，但还没生成实际包含的那些数

>>>li=list(range(1,101)) #可以用list(range对象), 生成 1,2,…100的列表

list(range(start,end,step)) 返回1个等差数列，不含终值end。 step是步长

注：start, end , step 必须为整数，不能为小数

例如list(range(1,10,3)) 生成[1,4,7]列表

list(range(0,10,2)) # 偶数列表 [0, 2, 4, 6, 8]

list(range(10, 0, -2)) # 偶数列表[10, 8, 6, 4, 2]

注：步长可以为负数，此时要求 start>end

list(range(end)) #初值start为0时可省略,只写end, step默认为1

list(range(5)) #生成[0,1,2,3,4] , 不含终值 5

list(range(start,end)) #省略step时，步长默认为1

list(range(2,5)) #生成[2,3,4]

(7) id函数

id(变量名) 将返回变量的内存地址，类似C中的指针

>>>x=10

>>>id(x)

1989739248 #返回变量x的类似内存地址的一个值

>>>x=20

>>>id(x) # 比较前后两次id(x) 的返回值是不同的，说明x重新分配了不同的内存空间

# 6、流程控制语句

（1）if条件语句

单分支语句

if 条件表达式: #必须要有冒号:

语句块 #语句块中各条语句必须保持一致的缩进

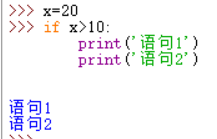
例:

x=20

if x>10:

print ('语句1') #语句1和2的缩进要一致

print ('语句2')



如果if后面只有一条语句，也可写在同一行。如下所示：

if x>10: print('ok')

双分支语句

if 条件表达式: #必须要有冒号:

语句块1 #此行必须缩进

else: #必须有冒号: , 不能缩进. else / if 要对齐

语句块2 #此行必须缩进

if x>=100:

print('大于等于100')

else:

print('小于100')

多分支语句

if 条件表达式: #必须要有冒号:

语句块1 #缩进

elif 条件表达式2: #有冒号 ,不能缩进

语句块2 #缩进

else: #有冒号，不能缩进。 if /elif/else要对齐

语句块3 #缩进

score=85

if score>=90:

print('优秀')

elif score>=80:

print('良好')

elif score>=60:

print('及格')

else :

print('不及格')

（2）循环语句 while 和 for

while 条件表达式：#必须有冒号:

语句块1 #缩进

k=1; s=0; #计算1到100的累加和

while k<=100: #有冒号:

s=s+k #缩进

k=k+1 #缩进, while循环中包含的语句缩进应一致

print(s) #此行不缩进，则整个while循环结束后再输出一次 s

#如果此行也缩进，则print从属于while, 将输出100次

for循环适合处理序列类（列表、元组、字符串、文件等）数据，依次取出各元素进行处理。

for 变量 in 序列**:**

语句块

li=[2,4,6,8]

for x in li:

print (x) #依次输出列表中的各个数, x会自动逐次取li中的每个值

#此处的x不需要事先定义，用任何变量名x,y,z均可

（3）break

break ：直接跳出循环，执行循环体后面的语句, break只用于while和for循环体中内，不能用在其它语句中。

li=[ ] #先定义一个空列表

while True: # True 定义了一个死循环

x=int(input('请输入一个数据 (0则结束): '))

if x==0:

break #当输入0时跳出循环

else:

li=li+[x] #非0，则将新输入的数据加入到列表中

print(li)

（4）continue

continue:结束本次循环（即本次循环continue后续的语句不再执行），重新开始一次新循环。该语句用于while和for循环体中。

k=0

while k<=100:

k=k+1

if k%5!=0:

continue #如不是5的倍数则continue, 这样就不会执行后面的print

print(k) #将只输出1-100中 5的倍数

Python 不支持 C语言中 for(表达式1; 表达式2; 表达式3)的形式

Python 没有C语言中的switch 语句

（5）pass 空语句，不进行任何操作，充当占位符。例如：

if x>5:

pass #不做任何操作

（6）断言 assert

有时程序需要确保满足某个条件才继续执行，如不满足则应直接退出程序，此时可用断言 assert。

x=int(input('x='))

# x必须满足下面的条件，如不满足则弹出异常AssertionError

assert 0<x<100 , ' x should be 0<x<100 '

print(x) #只有当 0<x<100，此句才会执行

# 7、内置数据结构：元组/列表/字典/集合

序列是指有顺序的一批数据。对序列可以按索引取值，切片。python中的序列主要包括：元组、列表、字符串。

元组和列表可容纳多个数据，类似C中的数组，但更灵活，其中包含的元素可以是不同的数据类型。

(1)、元组(tuple): x=(元素1，元素2，…) , 用小括号。元组不能添加/删除元素（即元组没有append, remove等方法），不能修改元素。通过[n]下标访问。下标从0开始编号，如下标为负数 -1，则指倒数第一个元素。

例子：

x=(1,) # 定义只含一个元素的元组x , 注意只有1个元素时，后面要有逗号

type(x) #返回 <class 'tuple'> 表示是元组类型

x=(1,3,5,7,8) # 定义元组x , 用小扩号

x=1,3,5,7,8 # 也可只用逗号分隔, 不用小扩号也行

print(x[0], x[2], x[-1], x[-2])

x[0]返回1， x[2]返回5, x[-1]返回8， x[-2]返回7

print(x[5]) #将报错 x的下标只有0到4， 下标5越界了

切片:返回序列中的某一段数据

x[start:end:step] : start:起始 end:结束下标(不含) step:步长(为1可省略)

x[m:n] 切片，从第m个开始到第n个（不含第n个）

x[m:] 从第m个一直到结束

x[m:-2] 从第m个到倒数第2个（不含倒数第2个）

x[:m] 从第0个到第m个（不含第m个）

x=(0,1,2,3,4,5)

x[2:4] 返回(2,3)

x[2:] 返回(2,3,4,5)

x[0:5:2] 返回(0,2,4) 步长为2

x[:-2] 返回(0,1,2,3)

x[: :-1] 返回(5, 4, 3, 2, 1, 0) , 步长为-1 就是倒数切片

x[1:4 :-1] 返回() . 注意：步长为负数时，start应大于end， 所以应写为

x[4:1 :-1] 返回(4,3,2) . 注意：步长为负数时，start应大于end

Python中将创建元组称为"打包"，将元组直接赋值给多个变量称为"解包"。

yz=("aaa","bbb","ccc") #打包

a,b,c=yz #解包后 a b c的值分别为'aaa',' bbb',' ccc'

#遍历元组方法1

yz=('aaa','bbb','ccc')

for x in yz: # x依次代表每个元素

print (x)

#遍历元组方法2

for x in range(len(yz)): # x代表每个元素的下标

print (yz[x]) #通过下标来引用元素

注：元组的元素不能增加/删除，所以元组没有类似列表的append(),insert(),remove()等方法。元组的各个元素也不能修改。

元组只提供了两种方法count和index：

yz.count(x) 返回对象x在元组中出现的次数

yz.index(x) 返回x 在元组中出现的位置，如元组不包含x,则产生异常

x=(1,2,3, 2,5,2,8)

x.count(2) # 返回3，表示元组中有3个2

x.index(3) # 返回2，表示数据3 在元组的第2个位置

x.index(9) #将报错， 因为x中不包含9

x[1]=100 # 将报错，元组的元素不能修改

id(x) # 返回 x所指向的内存地址

x=x+(4,5) #此处并不表示元组x可增加元素，而是产生了一个新元组x

id(x) #对比前后两次的id(x) , 可发现是指向不同地址的两个不同元组

(2)列表(list): li=[元素1,元素2,…] 用中括号。**列表是python中最常用的数据结构**。列表可以添加/删除元素，修改元素。下标从0开始。

li=[ ] #空列表

for x in range(10):

li=li+[x] #将元素x 加入列表, 要写为[x], 不能只写 x

print(li) #将输出 [0,1,2,3,4,5,6,7,8,9]

li=[1,2]\*4 #则li为[1,2,1,2,1,2,1,2]

列表的常用方法：



li=[1,3,5]

li[0]=2 #修改列表0号元素的值为2 ，列表变为 [2,3,5]

li.append(value) #添加元素, 添加的元素在尾部

li.append(8) # [2,3,5,8]

或 li=li+[8] # 注意不能写 li=li + 8

li.insert(pos,value) #将新值value插入到指定位置pos

li.insert(1,7) # [2,7,3,5,8]

li.remove(value) #删除value元素. 如value不存在，则抛出异常错误

li.remove(5) # li 为[2,7,3,8]

li.remove(9) # 将报错, 因为列表中没有9

del li[1] #删除第1个元素 [2,3,8]

del li[1:3] #删除第1-3(不含第3)元素。即第1, 第2个元素

li=[0,1,2,3]

li[1:3]=[7, 8,9,10] #将原第1,2元素删除 ，然后插入7, 8,9,10

# li 变为 [0, 7, 8, 9, 10, 3]

li[2:2]=[5,6] #向第2个位置插入5,6

# li 变为 [0, 7, 5, 6, 8, 9, 10, 3]

li.pop() #将弹出最后一个元素，弹出后该元素将被删除

li.pop(0) #将弹出第0个元素，弹出后该元素将被删除

使用列表的append(), pop()可模拟堆栈后进先出，pop()将末尾元素弹出

使用列表的append(), pop(0)可模拟队列先进先出,pop(0)将第0个元素弹出

li.pop(2) #将弹出并删除列表的第2个元素

li2=[9,10]

li.extend(li2) 或 li=li+li2 #列表的连接操作

li.index(value) #返回value的下标位置. 如value不存在将抛出异常

li=[0,2,4]

li.index(2) # 返回1， 表示数据2在列表的第1个位置

li.index(3) # 将报错， 因为列表中没有数据3

value in li #测试li中是否含有value, 如有返回True, 否则返回False

2 in li # 返回 True

3 in li # 返回False

len(list) #返回列表中元素个数，即列表的长度

li.reverse() #反转, 将列表元素前后颠倒, li被改变

另一种方法是调用reversed函数来反转,如下：

li2=list(reversed(li)) #将列表li前后颠倒得到新列表li2, li本身不变

列表排序方法1：

li.sort() #默认从小到大排，排序后li被改变

li.sort(reverse=True) #指定从大到小排，排序后li被改变

不能写为 li=li.sort() #因为li.sort()没有返回值，视为None, 赋值后, li就变为None了

列表排序方法2, 调用sorted()函数：

li2=sorted(li) # li排序生成一个从小到大的新列表li2。 li本身不变

li2=sorted(li,reverse=True) #生成从大到小的新列表li2. li本身不变

li=['ab', 'ddd', 'xyzf', 'a', '345678']

li.sort() # ['345678', 'a', 'ab', 'ddd', 'xyzf'] 按字母的ascii码顺序排

li.sort(key=len) # ['a', 'ab', 'ddd', 'xyzf', '345678'] 按字符串长度排. 本例key=len表示按len函数来排，每个元素用len测试，得到长度，然后按长度排

li.count(value) #返回value在列表中出现的次数。未出现则返回0

注意列表的赋值 list2=list1 ， 两个列表指向同一内存位置，修改任一个列表，另一个也会受影响

list1=[2,3,4]

list2=list1 #两个列表均指向同一个内存位置，实际上是一个列表。如果修改了list2, 则list1也随之改变。

list2[0]=100

print(list1[0]) # 输出100，表明list1和list2是一样的，修改了list2,所以list1也变了

列表复制方法1：

list3=list1[**:**] #中括号内只写一个冒号: , 表示复制list1全部元素生成新列表list3，新旧列表指向不同位置，list1和list3互不影响。

列表复制方法2：引入copy 模块，实现列表的克隆复制

import copy

list4=copy.copy(list1) # 浅复制,list4 是新列表，新旧列表指向不同位置

list5=copy.deepcopy(list1) #深度复制 list5 是新列表

列表和元组的转换：

yz=tuple(列表) 将一个列表转换为一个元组

li=list(元组) 将一个元组转换为一个列表

li=[3,4,5]

yz=tuple(li) #得到元组

li2=list(yz) #得到列表

存储3x3的二维矩阵, 可以用双重列表实现（即列表的元素也是一个列表）

1 2 3

4 5 6

7 8 9

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

print(x[0][0], x[2][1]) # 输出1 8

for i, k in [(1,2),(3,4),(5,6)]:

print (i, k)

元组与列表对比：

元组元素不可增删，不能修改，但操作速度快。列表可增删，元素可修改，速度稍慢。python某些函数的参数或返回值是元组,所以应掌握对元组的操作。

列表生成式, 在生成时使用for , if进行条件判断

li=[ x for x in range(5)] # [0,1,2,3,4]

li=[x \*\*2 for x in range(1,6) ] #[1, 4, 9, 16, 25]

li=[ x for x in range(100) if x %3==0 ] #只包含0-100中3的倍数

s=[chr(x) for x in range(65,91)] # 生成大写字母列表，因为大写字母A-Z的ASCII码编号为65至90

li=[m + n for m in 'ABC' for n in 'XYZ']

['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']

nums=[-1, 0, 6, -4, -2,3]

result=[n for n in nums if n>0] #只保留>0的数 [6,3]

li=['aa','bb','cc','dd','aa']

li2= [x for x in li if x not in ['aa']] #从li列表中剔除'aa'

将只含['bb', 'cc', 'dd']

利用zip 将两个列表配合生成元组, 两个列表的长度可以不同,最后生成的列表长度以最短列表为准.

name=['a1','a2','a3','a4']

age=[20,21,22]

stu=list(zip(name,age)) #则 stu=[('a1', 20), ('a2', 21), ('a3', 22)]





filter/ lambda /map/reduce: 函数式编程（较难，初次看不懂可先跳过）

(1) filter 过滤函数

li=list(range(10))

def even(x):

if x%2==0:

return True # 偶数则返回True

else:

return False

#将li的值依次交给even函数判断，如返回True则选中该值

li2=list(filter(even, li)) # 将filter的返回用list()再处理一次，得到列表

print(li2) # 输出[0, 2, 4, 6, 8]

li=[12, 'ab', 'de', 6, 10] # 列表的元素包含整数和字符串

def isint(x):

if isinstance(x,int): #判断是否为整型

return True

else:

return False

li2=list(filter(isint, li)) # 利用isint判断， 只保留整数数据

print(li2) # [12, 6, 10]

(2) lambda

lambda 是匿名函数, 即定义一个没有名称的函数。函数体一般很简单, 执行有限的操作,不能包含if ,else, 变量定义这样的语句.

上例中专门定义了一个even函数，这个函数可以用lambda来代替。如下

li2=list(filter(lambda x : x%2==0, li)) # x依次取li中的值,筛选出偶数

[0, 2, 4, 6, 8]

解释：lambda x: x%2==0 定义了一个匿名函数，参数为x, 函数体只有x%2==0 这一句（即x为偶数返回True）。 filter 从 li中依次取出每个值，传给匿名函数，如返回True就选中该值。

例，有学生成绩表如下，4个学生，每个学生两门课

>>> stu=[('01',100,98),('02',97,96),('03',89,100),('04',95,98)]

>>> stu

[('01', 100, 98), ('02', 97, 96), ('03', 89, 100), ('04', 95, 98)]

>>> max(stu) #找出最大学号的学生

('04', 95, 98)

>>> max(stu, key=lambda x:x[1]) # 第一门课的最高分

('01', 100, 98)

>>> max(stu, key=lambda x:x[2]) # 第二门课的最高分

('03', 89, 100)

>>> min(stu, key=lambda x:x[2]) # 第二门课的最低分

('02', 97, 96)

(3) map 映射函数

map 用于将列表交给一个函数计算，得到计算后的结果列表.

import math

li=range(5)

li2=list(map( math.sin, li)) # map会将li交给sin计算，得到li 各数的sin值

li3=list(map(lambda k : k\*\*2, li)) # 计算li 各数的平方

[0, 1, 4, 9, 16]

解释：lambda k: k\*\*2 定义了一个匿名函数，参数为k, 函数体只有k\*\*2 这一句, 返回k的平方。 map 从 li中取出每个值，传给匿名函数，得到返回的平方值。

filter和map的第1个参数是函数, 第2个参数是列表, 表示将列表中的元素逐一交给函数处理。lambda的一个主要作用就是让我们方便的定义一个匿名函数，这个函数可用在map, filter内部。

再看一个map的例子

li1=[1,2,3]

li2=[20, 30, 25]

li3=[4,5,6]

li4=list(map(lambda x,y,z : x+y+z, li1, li2, li3))

print ( li4) # li4=[25, 37,34] 将三个列表的值按索引号加起来

再看一个filter的例子

def f(s): #定义了函数f, 当len(s)>4 时返回True

if len(s)>4:

return s

li=['abc', 'xyzab', 'sssss', 'ab']

li2=list(filter(f, li)) #过滤li, 只包含长度>4的字符串

print(li2) #只输出 ['xyzab', 'sssss']

(4) reduce

reduce 提供了一种累积式计算,即将前次的计算结果自动用于后续的计算。

from functools import reduce #必须先引入reduce函数

>>>reduce(lambda x, y : x+y, range(1,101))

5050

注：上例定义了匿名函数lambda x, y : x+y , 即两个数相加， 相加以后的结果继续做下一次相加，所以完成了 1+2+3+…+100=5050 的累加

>>>reduce(lambda x, y : x\*y, range(1,5))

24

注：上例定义了匿名函数lambda x, y:x\*y , 即两个数相乘， 相乘后的积继续做下一次相乘，所以完成了 1\*2\*3\*4=24 的累乘 （4！）

练习:要求将列表li中的元素都取绝对值

li=[1, -2, -3, 4]

方法1

li=[abs(x) for x in li]

方法2

li=list(map(abs, li))

练习:要求将列表li中的每个元素都加2

li=list(range(5))

方法1

li=[x+2 for x in li]

方法2

for x in range(len(li)):

li[x]=li[x]+2

方法3

x=0

while x<len(li):

li[x]=li[x]+2

x=x+1

方法4

li=list(map(lambda x:x+2, li))

(3)字典 dic={"key1":"value1", "key2":"value2"…}, 花括号。可以添加和修改元素。key1,key2是键（区分大小写），通过键查找value。字典中键必须唯一，值可以不唯一。字典和集合不是序列数据，不能按索引取值，不能切片。

di= {'a':'apple', 'b':'banana', 'p':'pear'} #构造字典方法1

di['a'] #返回'apple'

di['p'] #返回'pear'

di.get('a') #返回'apple'

di['c'] #将报错KeyError: 'c'，因为字典中无'c'键

di[1] #将报错, 字典不能用索引下标存取

注：python2.x的has\_key方法在python3.x中不可用

x=di.get('c') #用get方法时即使无'c'键也不会报错，将返回None

x=di.get('c', 'no') #用get方法时也可指定第2个参数，有'c'键时就返回其对应值，无'c'键时就将返回第2个参数设定的值'no'

di.pop('a') #得到'a'对应的'apple', 同时从字典中删除'a'键

di.pop('c') #如字典中无'c'键将报错

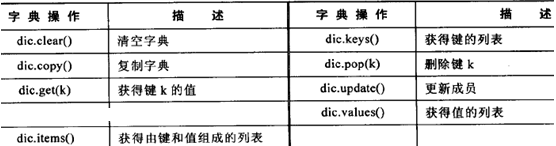
name=[ '张三' , '李四', '王五']

tel=['13802345893', '13700001234', '18000098765']

di= dict(zip(name,tel)) #构造字典方法2， 得到如下字典

di={'王五': '18000098765', '张三': '13802345893', '李四': '13700001234'}

di['张三'] # 返回 '13802345893'



d=dict() #得到空字典对象 d

items=[('name', 'john'), ('age',42)]

d=dict(items) # 构造字典方法3 , d={'age': 42, 'name': 'john'}

dic[key] 通过key访问字典,返回对应的value。如无此key，则报错。字典不能用下标访问。

dic.keys() # 返回所有key构成的列表

dic.values() # 返回所有value构成的列表

dic.items() 返回一个由全体键/值对元组构成的列表

'key' in dic #返回布尔值, 表明是否含有此key

di2=dic.copy() 字典复制

dic["x"]="value" #如果字典中已有x键，则是修改其对应的value；如没有x键，则是添加新项( 例如di['l']='lemon' # 添加一个新的项'l':'lemon')

字典没有remove(), 可调用内建函数del删除字典项

del dic['a'] #删除字典中某个键

del dic['c'] #如无'c'键将报错

del dic #删除整个字典, dic对象消失

dic.clear() #清空字典, 但dic对象还在

len(dic) #字典长度

>>>di={x:ord(x) for x in 'abcd'} #构造字典方法4

>>> di #构建的字典不保证一定按 a b c d 键的顺序存储

{'b': 98, 'c': 99, 'a': 97, 'd': 100}

>>>di=dict(name='smith',age=20) #构造字典方法5 ， 小括号

>>>di

{'age': 20, 'name': 'smith'}

dic1.update(dic2) #字典更新. 比较dic1和dic2, 将dic1缺少但dic2中有的元素添加到dic1, 然后将dic2中匹配的元素的值更新到dic1中。

dic1={'name':'john', 'age':20}

dic2={'name':'smith', 'height':1.8}

dic1.update(dic2) #dic1修改为 dic1={'name':'smith','age':20,'height':1.8}

#遍历字典方法1

dic= {'a':'apple', 'b':'banana', 'p':'pear'}

for key in dic:

print (dic[key]) # key 代表键, dic[key]代表对应的值

#遍历字典方法2

for key in dic.keys():

print (key ,dic[key])

#遍历字典方法3

for (k,v) in dic.items(): #items()取出所有的键/值对

print (k, v) # k 代表键, v代表对应的值

#字典内嵌套字典

stu={'1':{'name':'张三','age':20}, '2':{'name':'李四','age':22}}

print (stu['1']['name']) #输出1号同学的姓名

对字典进行排序，排序后得到一个列表，原字典不变

dic= {'a1': 25, 'a3': 28, 'a2':21}

1）按值顺序排 ， x[1]是值

li=sorted(dic.items(), key=lambda x:x[1])

# [('a2', 21), ('a1', 25), ('a3', 28)]

解释: dic.items()返回1个列表, 这个列表的每个项是1个元组x, x[0]是键,x[1]是值, 将每个项传给lambda匿名函数, 返回x[1]作为排序依据

（2）按值逆序排

li=sorted(dic.items(), key=lambda x:x[1], reverse=True)

#[('a3', 28), ('a1', 25), ('a2', 21)]

reverse=True 表示逆序，从大到小

（3）按键逆序排 , x[0]是键

li=sorted(dic.items(), key=lambda x:x[0], reverse=True)

#[('a3', 28), ('a2', 21), ('a1', 25)]

和list比较，dict有以下几个特点：

1. 查找和插入的速度极快，耗费时间不会随着key的增加而增加；

2. 字典需要占用大量的内存，内存浪费多。

而list相反：

1. 查找和插入的时间随着元素的增加而增加；

2. 占用空间比字典小，浪费内存少。

(4)集合set

集合由一组无序排列的元素组成，主要用于检查成员资格，集合中不允许重复元素。集合中的元素必须是不可变元素（如整数/字符串/元组），不能是可变元素（如列表/字典）。集合可以分为可变集合（set）和不可变集合（frozenset）。可变集合创建后可以添加元素，修改元素和删除元素。不可变集合创建后不能改变。

li=[1,1,2,3,3,5]

s=set(li) # 由列表产生可变集合s，创建时自动剔除重复元素

print(s) # s 为可变集合，输出{1,2,3,5}

for x in s:

print(x)

print(s[0]) # 错误，集合不是序列类数据，不能用下标操作

s2=frozenset('abcde') #s2是不可变集合 , {'c', 'd', 'a', 'e', 'b'}

type(s2) #<type 'frozenset'>

a=set(range(10)) #可变集合 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

type(a) #<type 'set'>集合类型

使用add()和update()可以在集合中添加新项

a.add(8) #添加一项

a.update([10,11,12]) #向a中添加多项

a.remove(8) #删除8

a.clear() #清空集合元素, 执行后 a为空集 set()

s=set([0,1,2,0,1]) #得到集合{0,1,2}

s={0,1,2} #集合也可以直接用{ }来定义，产生和上句同样的集合

0 in s #检查0是否在集合s中, 结果为 True

3 in s # False

集合可以使用并、交、差等集合运算

a=set([1,2,3])

b=set([2,3,4])

c=a.union(b) #a并b, {1,2,3,4}

c=a.intersection(b) #a交b, {2,3}

c.issubset(a) #判断c是a的子集 True

c=a.difference(b) #差(a-b) {1}

集合并、交、差运算的另一种表示方法

a=set([1,2,3])

b={2,3,4}

c=a | b #并集 {1, 2, 3, 4}

c=a &b #交集 {2, 3}

c=a - b #差集（在a中，但不在b中） {1}

c=a ^ b #对称差集（在a或b中，但不会同时出现在a和b中）{1, 4}

a=set([1,2,3])

b=set([1,2])

1 in a # True

a - b # 集合{3}

b < a # True ， 表示b 包含于a

b> a # False

a+b #报错，不支持集合相加

集合应用举例:

s={1,2,3,4,5} #假定有全集s, 包含5个同学

s1={1,2} # s1 代表选第一门课的同学

s2={2,3,4} # s2 代表选第二门课的同学

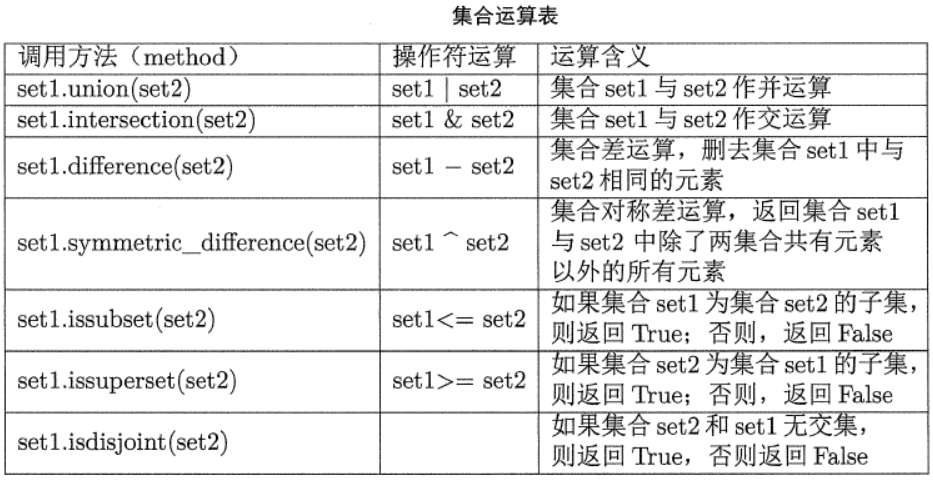
(1) 同时选了两门课的同学 s1 & s2

(2) 选了任意一门课的同学 s1 | s2

(3) 只选了第一门课，没有选第二门课的同学 s1 - s2

(4) 一门课都没有选的同学 s - s1 - s2

(5) 只选了一门课，没有选二门课的同学 s1 ^ s2



# 8、函数与模块

**一、函数**

函数：将完成某种功能的语句编写在一个函数内，然后可反复调用。

abs(-3) # abs是求绝对值函数， 返回-3的绝对值3

help(abs) #显示abs 的帮助信息

print(abs.\_\_doc\_\_) #显示该函数简短说明，doc的前后都是2根下划线

用户可调用系统已有的函数（如sqrt, abs等），也可自定义函数。

自定义函数的语法如下：

def 函数名(参数1，参数2…)**: #有冒号**

函数语句体 #函数体内的所有语句都要缩进

[return 表达式]

函数可以没有参数，但在定义和调用时即便没有参数，也必须加上括号()。return是返回值，可以没有。如果没有return语句, 则返回None.

例：编写函数，比较两个数的大小，输出并返回较大的数。

def printMax(x, y):

''' return max of x, y ''' #函数的简短说明,写在函数名的下面

if x>=y:

print (x, y, ' the max is:', x)

return x

else:

print (x, y, ' the max is:', y)

return y

编写好后，可用下面的语句测试：

a=printMax(1,100)

b=printMax(20,3)

print (a, b)

print(printMax.\_\_doc\_\_) #显示函数的简短说明

1.函数如需要返回多个值,可返回元组。如return x, y, z 将返回一个元组

2.python中任何变量都是对象，所以参数只支持引用传递。还支持默认值传递，支持按名传递。字符串/数值/元组作为实际参数时不会被函数内的修改所改变，列表作为实际参数时，会被函数内的修改改变。(请参考其它更详细的资料)

3.函数定义时可以在函数名冒号的下一行写一个字符串，这个字符串可视为函数功能的简短说明，可以用 函数名.\_\_doc\_\_的形式查看，帮助其他程序员了解你所编写的函数的作用。

4.调用函数时一定要有括号，如果漏了括号，系统并不报错，只在命令行上显示该函数的信息， 但并没有执行函数。如下例：

>>> printMax #调用时漏了小括号

<function printMax at 0x00000269BDC8E158>

python函数的参数非常灵活，可以有以下特殊形式:

(1) 默认值参数

def hi(x='Hello'): # x是默认值参数，定义时提供了参数x的默认值

print (x)

hi('你好') #调用时，如提供了参数值，则输出"你好"

hi() #如不提供参数值，则输出默认值"Hello"

默认值参数应定义在所有参数的最后面

def hi2(x='Hello', name) #错误. x默认值参数被定义在普通参数name的前面

def hi2(name , x='Hello') #正确

def hi2(name, x='Hello'):

print (name, x)

hi2('Jack', '你好') # 输出 Jack 你好

hi2('Tom' ) # 输出 Tom Hello

(2) 参数按名称传递(也称为关键字参数)

#支持参数按名传递,这样调用时不用一定按照声明时的参数顺序进行赋值.如下：

hi2(x='aaa…', name='Jack')

也可

hi2( name='Jack', x='aaa…')

(3) 参数可变个数

def func(\*args): # \*传递可变长度的参数, 以元组形式

例：编写函数, 求若干个数的和。

def calcsum(**\***data): #注意有 \* 号

s=0

for k in data:

s=s+k

return s

print ('2+5=', calcsum(2,5)) #传递了2个参数

print ('3+5+8+9=', calcsum(3,5,8,9)) #传递了 4个参数

(4) 字典参数

def func(\*\*kw): # \*\*可传递字典参数, kw被视为字典

for x in kw:

print (x, kw[x])

func(x=1,y=2,z=3) #传过去 kw={'x':1, 'y':2, 'z':3}

变量的作用域分为：局部作用域（局限于函数内），全局作用域（作用于整个模块文件）。在函数内部定义的变量/参数属于局部变量，在所有函数体以外定义的变量属于全局变量。示例如下：

#此处的a, x, y 定义在模块级（不是在函数内部），都是全局变量

a=3 ; x=5 ; y=10

def fun(x):

global a # global说明要使用全局变量a

a=a+10 #此行两边的两个a都是指全局变量a，执行后修改了全局变量a

x=x+20 #此行两边的x都是局部变量，执行后不影响全局变量x

y=1 #这里的赋值产生了一个局部变量y, 只在函数内有效

print (a, x, y) #先输出初始值 3 5 10

fun(x) #调用函数

print (a, x, y ) #输出13 5 10, 表明调用函数fun后，a 被修改，x,y不变

注：函数内部的赋值语句创建的变量（如上例的y=1），如果没有经过global声明，都是局部变量，只在函数内有效，当函数执行结束，该变量即消失。

**二、模块**

Python程序由包、模块、函数构成。

(1)包其实就是文件夹, 该文件夹中必须含有一个\_\_init\_\_.py文件(文件可为空)，包可含有若干模块文件。 python自带的工具包在安装目录的lib子目录中。

(2) 一个python文件就是一个模块。模块由代码、函数或类构成。除了在模块中声明函数外，还可以定义变量。模块中的变量可以在其他脚本中使用。python标准库包含了数以百计的模块，这些模块可以直接引入，所以说python是“自带电池”。

help('modules') #将列出本机已安装模块(包括当前目录下的.py程序)

import os

help(os) #查看os模块的信息

模块就是命名空间，两个模块中可含有同名的函数，由于模块名不同，这两个函数就可以区分开。例如:

import my1, my2

my1.fun1() #调用my1模块中的fun1函数

my2.fun1() #调用my2模块中的fun1函数

使用模块前，需要先用import语句导入模块，类似C语言的include。常用的模块有sys, math, os, os.path, datetime, time等。导入模块的几种语法如下：

1 import 模块名

import math

print (math.pi) # 数学常量 pi

print (math.e) # 数学常量 e

print (math.exp(9)) # e 的 9次幂 模块名.函数名

print (math.sqrt(9)) #开平方

print (sqrt(9)) #会出错，前面应加math.模块名

dir(math) # 列出math中的所有函数

python的import语句可以出现在程序的任何位置。

2 from 模块名 import 函数名

from math import sqrt

# 直接引入math 模块中的sqrt 函数，这样在使用sqrt时，前面就不用写math.

print (sqrt(9)) #直接用函数名调用, 前面不用再写math.

3 from 模块名 import \* #导入整个模块。导入后可以直接用函数名调用

from math import \*

print (sqrt(9)) #可直接调用math中的任何函数, 不用写math.

print (sin(2)) #计算sin值， 前面不用写math.

以上三种方法，推荐用第一种。这样模块中即使有同名的函数也不会相互影响。

自定义模块: 一个.py源文件就是一个模块。如下例：

# 该文件名一定要为 mymodule.py

info='aaaa' # 定义模块内的一个变量

def mysum(y):

'求1+2+….+y 的累加和' #此处的字符串即函数的\_\_doc\_\_属性

result=0

for x in range(1, y+1):

result=result+x

return result #返回计算结果

if \_\_name\_\_=='\_\_main\_\_':

#下面是测试代码，仅在直接运行模块时执行

print (mysum(10))

#输出mysum的\_\_doc\_\_属性 doc前后都是两个下划线

print (mysum.\_\_doc\_\_)

print ('以主文件形式运行')

解释：\_\_name\_\_是python预定义的变量，其值由python根据运行环境自动赋值。如果其值为"\_\_main\_\_"就表示该模块正在独立运行。当模块独立运行时，就执行if 后面的测试代码，便于我们测试。当模块被其它程序调用时，此时\_\_name\_\_的值不等于"\_\_main\_\_", 就不执行if后面的测试代码，这样就不会对主调程序造成干扰。

注：name,main前后均为两条下划线。

调用模块

import mymodule #引入mymodule.py , 只写主文件名，不能加.py扩展名

print (mymodule.mysum(20)) #调用模块内的mysum函数

print (mymodule.info) #输出模块内info变量

dir(mymodule) #可列出模块中定义的名字、函数名等. 内容如下：

['\_\_builtins\_\_', '\_\_cached\_\_', '\_\_doc\_\_', '\_\_file\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', 'info', 'mysum']

调用此模块后，python3.5会自动产生mymodule.cpython-35.pyc文件，这是.py编译后的字节码文件。对于模块而言，python总是在第一次调用后将其编译成字节码的形式，以提高脚本的启动速度。不作为模块的脚本，python不会在运行时将其编译为字节码.pyc。

有时模块名本身比较长，可以用as 重命名。例如下面将mymodule 命名为my, 后续就可用 my 来引用模块

>>>import mymodule as my

>>>print (my.info)

在import导入模块时，先在当前目录找，然后在sys模块中的sys.path变量指定的目录找，如没有，则报错。

假定你编写的模块m1.py放在c:\abc目录下, 可用如下语句调用：

import sys

sys.path.append(**r**'c:\abc') # 在搜索目录中添加 c:\abc目录

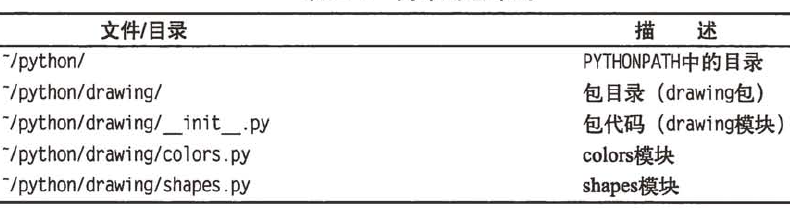
import m1 #导入时不能写m1.py , 只写主文件名m1

这样系统就可在c:\abc中找到你的m1模块。

也可将自己编写的模块放在python安装目录的site-packages目录中，例如C:\Python35\lib\site-packages, 这个目录默认包含在sys.path中。

包的练习（请参考更详细的资料）：

创建drawing文件夹（文件夹中必须含有一个\_\_init\_\_.py文件, 内容可为空），其中包含shapes.py和colors.py模块，此时就产生了一个名为drawing的包。包布局如下：



针对上图，下列语句都是合法的：

import drawing #引入drawing包

import drawing.colors #引入drawing包中的colors模块

import drawing.shapes

from drawing import shapes

drawing.colors.fun1() #调用colors中的函数fun1

包提供了一种分级组织模块的方法。

# 9、字符串

以单/双/三引号作定界符均可。字符串是最常处理的数据，应重点掌握。

'abc' , "abc" , '''abc''' 均可

"I'm a student" # 在双引号中可包含单引号

'I"m a student' # 在单引号中可包含双引号

''' # 三个单引号用于表达多行文字构成的字符串

多行文本行1

多行文本行2

'''

s='abcde'

print(s[0]) #输出第0个字符 a

for x in s:

print(x) #逐个字符输出

s[0]='x' #将报错，字符串中的单个字符不能修改. 字符串是不可变量

print("abc"\*3) # 将输出 abcabcabc

(1) 字符串可用%格式化输出。例如：

age=30

name='Bill'

print ('%s is %d years old ' % (name,age) )

输出 Bill is 30 years old

import math

print ('pi is %.2f %.5f %.8f ' % (math.pi, math.pi, math.pi))

输出 pi is 3.14 3.14159 3.14159265

常用的格式控制符：

%s :格式化为字符串

%8s 字符串8个字符宽，右对齐，不足8位则左边补空格

%**-**8s 字符串8个字符宽，前面的负号表示左对齐，右边补空格

%d 整型

%8d 整型，表示转换为8位长整数，右对齐, 不足8位则前面补空格

%08d 整型，表示转换为8位长整数，不足则前面补0

%**+**8d 整数8个字符宽，前面的+号表示要添加一个+

%u无符号整数 %o 无符号八进制

%x 十六进制小写 %X 十六进制大写

%e 科学计数法 %.2f 2位小数精度

%6.2f 总宽度6位，2位小数精度

print ('%d, %o, %x, %e' % (100,100,100,100) )

输出100, 144, 64, 1.000000e+02

(2) 字符串也可用.format方法格式化

s='my name is {0} and I am from {1}'

s1=s .format ('al','houston')

print(s1) #显示 my name is al and I am from houston

#用{0}、{1}代表第0个、1个参数

'my name is {0:s} and my age is {1:4d}' .format ('john',21)

{0:s}表示第0个参数用字符串格式，{1:4d}表示第1个参数用4d整数格式

s='my name is {0:8s} and my age is {1:4d} salary:{2:.2f}' .format ('smith', 21, 820.3)

print(s)

{0:8s}表示第0个参数字符串格式，8位宽

{1:4d}表示第1个参数整数格式， 4位宽

{2:.2f}表示第2个参数小数格式(注意有个小数点)， 小数部分2位

也可简写为下面的格式,省略序号 ({}中间不要有空格)

s='name: {} , age: {} , salary:{}' .format ('smith', 21, 820.3)

输出时字符串默认靠左对齐， 数值靠右对齐。可添加额外的控制字符调整对齐方式：

s1='abcd' ; s2='ekfj' ; s3='yey'; s4='xyzq'

print('s1={0:^10} ,s2={1:<10} ,s3={2:>10}, s4={3:.2}'.format(s1,s2,s3,s4))

输出内容如下：

's1= abcd ,s2=ekfj ,s3= yey, s4=xy'

^居中对齐

<左对齐

>右对齐

:.2 注意有个.号， 表示该字符串只占2个字符宽，超出的部分截断

注： 推荐使用.format方式格式化字符串。 %的格式化方式是旧模式，以后的python版本可能放弃%模式。

原始字符串

>>>path='c:\nowhere'

>>>print (path)

c: #由于\n被理解为换行转义符, 所以输出时换行了

owhere

>>>p=**r**'c:\nowhere' #串前加r或R , 表示原始raw字符串，将忽略转义符

>>>print (p)

c:\nowhere #输出正常

>>>p='c:\\nowhere' #本例也可写成\\ , \\就不是转义，代表\自身

raw原始字符串可以方便地表示目录路径。注：原始字符串不能以 \ 结尾。

import string #导入字符串模块

print (string.digits) #数字0-9

print(string. ascii\_letters) #所有字母

print(string.ascii\_lowercase) #小写字母

print(string.ascii\_uppercase) #大写字母

字符串的方法：

str.find(sub) #返回子串在主串中的位置，找不到则返回 -1

'abcdefcdp'.find('cd') #返回2。表示'cd'在'abcdefcdp'的第2个位置

'abcdefcdp'.find('cd',4) #返回6 . 从第4个位置开始找, 找到的cd在第6个位置

'abcd'.find('xy') #返回 -1 , 表示没有找到

str.split() 分离字符串，返回一个列表

s1='aa,bb,cc,dd'

li=s1.split(',') #将s1以逗号分离，得到一个列表['aa','bb','cc','dd']

s1='11 22 33 44 55 '

li=s1.split() #默认以空格或制表符\t分隔, 得到['11','22','33','44','55']

>>> 'a b c'.split(' ') #此处指定以单个空格为分隔符

['a', 'b', '', '', 'c']

>>> 'aa bb dd'.split() #此处没有指定，则所有空格都视为分隔符

['aa', 'bb', 'dd']

str.join(list) 将一个列表以str为间隔联接为字符串. 例如：

sep=','

s=sep.join(['a','b','c','d']) #得到字符串'a,b,c,d'

x=['A', 'B', 'C', 'D', 'E', 'F']

''.join(x) #得到'ABCDEF'

s=str.replace(old,new) 替换子串. 用new替换old

s='abccaa'.replace('a','x') # 'xbccxx'

s=' abc \n'

s1=s.strip() # strip删除字符串两端的空格和转义字符 'abc'

s2=s.lstrip() # lstrip删除字符串左端的空格和转义字符 'abc \n'

s3=s.rstrip() # rstrip删除字符串右端的空格和转义字符 ' abc'

s=str(90.6) 转换为字符串 "90.6"

s='abcde'

for c in s:

print (c) #逐个字符输出

切片： string[start:end:step]返回子串

"abcde"[1:3] #返回"bc"

python使用==, !=比较两个字符串的内容

"ab"=="ab" #结果True

'ab'=='AB' # False

'ab'.lower()=='AB'.lower() #True , 都转为小写字母再比较

"ab"!="AB" #结果True

s="abXYZ"

s1=s.lower() #转小写字母 abxyz

s2=s.upper() #转大写字母 ABXYZ

s.isalnum() 如s每个字符都是字母或数字且非空，则返回True

s.isalpha() 如s每个字符都是字母且非空，则返回True

s.isdigit() 如s每个字符都是数字且非空，则返回True

s.islower() 如s每个字符都是小写字母且非空，则返回True

s.index(子串) 返回子串在s中的位置。如不存在则异常出错

'abcd'.index('bc') #返回1， 表示 bc 在 abcd的第1个位置

'abcd'.index('cc') #将报错， 因为abcd中不包含cc

''.join(reversed('abcd')) 将字符串反转, 'abcd' 变为 'dcba'

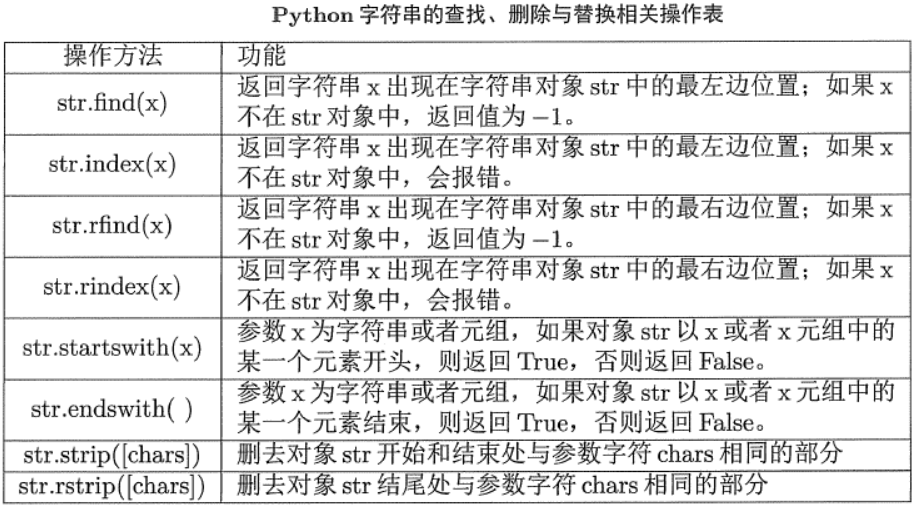
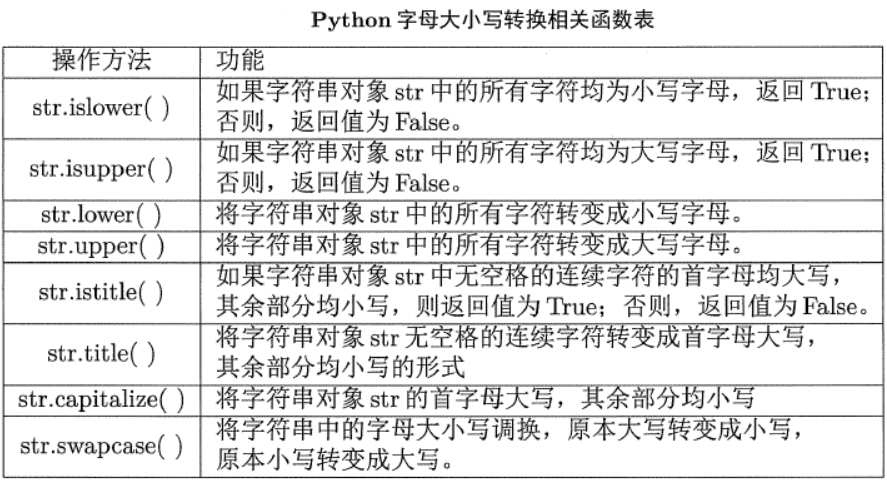
'abcd'[::-1] # 也是反转 'abcd' 变为 'dcba'

s.startswith(substring) 判断开头 'abcd'.startswith('ab') True

s.endswith(substring) 判断结尾 'abcd'.endswith('cd') True

fname='a.jpg'

fname.endswith(('.bmp','.jpg','.gif')) # True, 判断fname是否是图片后缀名



# 10、日期和时间

python提供了datetime , time模块处理日期和时间.

import datetime #先导入datetime

today=datetime.date.today() #今天，只含日期，不含小时

one=datetime.timedelta(days=1) #获得1天的日期差距类型

yesterday=today-one #昨天

tomorrow=today + one #明天

print (today, yesterday, tomorrow)

1.获取当前时间的方法

import datetime, time

a=datetime.datetime(2015,5,1,10,30,20) #构造一个时间: 年 月 日 时 分 秒

print(a)

now2 = datetime.datetime.now() #得到当前的日期和时间

type(now2) #now2的类型 <class 'datetime.datetime'>

print (now2) #类似 2014-10-12 13:50:36.003000

print(now2.strftime("%Y-%m-%d %H:%M:%S")) #将now2转为字符串输出

#输出类似2014-10-12 13:50:36

daystr = now2.strftime("%Y-%m-%d") #将日期转为字符串

type(daystr) # <class 'str'>

print (daystr)

t=time.strptime(daystr,"%Y-%m-%d") #将特定格式的字符串转日期

python中时间日期格式化符号（要区分大小写）：

%y 两位数的年份表示（00-99）

%Y 四位数的年份表示（0000-9999）

%m 月份（01-12）

%d 月内中的一天（1-31）

%H 24小时制小时数（0-23）

%I 12小时制小时数（01-12）

%M 分钟数（00-59）

%S 秒（00-59）

%a 本地简化星期名称例如 Fri

%A 本地完整星期名称例如 Friday

%b 本地简化的月份名称Oct

%B 本地完整的月份名称October

%c 本地相应的日期表示和时间表示

%j 年内的第几天（001-366）

%p 本地A.M.或P.M.的等价符。一般返回AM PM

%U 一年中的星期数（00-53星期天为星期的开始）

%w 星期（0-6），星期天为星期的开始. 星期1至6返回 1-6，星期天返回0

%W 一年中的星期数（00-53星期一为星期的开始）

%x 本地相应的日期表示

%X 本地相应的时间表示

%Z 当前时区的名称

%% %号本身

from datetime import date, time, datetime, timedelta

today = date.today()

print("today: {0!s}".format(today))

print("year: {0!s}".format(today.year))

print("month: {0!s}".format(today.month))

print("day: {0!s}".format(today.day))

current\_datetime = datetime.today()

print("now: {0!s}".format(current\_datetime))

注： {0!s} 是强制按字符串格式输出，无论后面的实际参数是什么类型。

2. 构造日期和时间

import datetime,time

d1=datetime.date(2014,5,31) #构造一个日期<class 'datetime.date'>

d2=datetime.date(2014,10,31) #构造一个日期

print(d1.day) #输出d1中的日期 31

print(d2.year) #输出d2中的年份 2014

print(d2.weekday()) #输出星期几, 星期一到六分别对应 0-6

x=d2 - d1 #x是时间差值 <class 'datetime.timedelta'>

print (x.days) #d2, d1差距的天数 153

d3=d2+x #d2后推x天，得到一个新日期d3

t1=datetime.time(23,11,30) #构造一个时间23:11:30<type 'datetime.time'>

3.获取上个月最后一天的日期(用本月的第一天减去1天)

x=datetime.date.today() #得到今天

#先构造本月的第一天datetime.date(x.year,x.month,1)

first=datetime.date(x.year, x.month,1)

last=first-datetime.timedelta(1) #本月第一天再减1天,得到上月最后一天

print (last)

4.获取时间差(时间差单位为秒，常用于计算程序运行的时间)

import time, datetime

starttime = datetime.datetime.now()

#long running

endtime = datetime.datetime.now()

print ('程序共耗时毫秒数:', (endtime - starttime). microseconds)

#或

t1=time.time() #得到一个秒数。从1970-1-1日 0时 0分 0秒起到现在经过的秒数

time.sleep(7) #暂停7秒

t2=time.time()

print (t1, t2, t2-t1) # t2- t1 即是运行花费时间, 单位秒

#或

t1=time.clock() # 返回从上一次调用clock函数后所经过的秒数

time.sleep(7) #暂停7秒

t2=time.clock()

print (t2-t1) # t2- t1 即是运行花费时间, 单位秒

上面两个函数可查看帮助了解函数功能。

help(time.time)

help(time.clock)

t=time.time() # 得到时间戳， 即从1970-1-1 日 0时开始的秒数

time.localtime(t) # 将时间戳化为时间元组格式，有9个元素，分别代表（年，月，日，时，分，秒，一周中的第几天，一年中的第几天，夏令时）

time.ctime(t) # 将时间戳化为 字符串表示的时间格式

5.计算当前时间向后10个小时的时间

d1 = datetime.datetime.now()

d3 = d1 + datetime.timedelta(hours=10)

print(d3.ctime())

常用的类有：datetime和timedelta两个。它们之间可以相互加减。每个类都有一些方法和属性可以查看具体的值，如 datetime可以查看：天数(d.day)，小时数(d.hour)，星期几(d.weekday() 返回0-6 ,依次代表星期一/二/..日), 日期( d.date() )等;timedelta时间差可以查看：间隔天数(dt.days)，间隔秒数 (dt.seconds)等。

例：一个倒计时10S的小程序，倒计时结束时播放指定的音乐文件

import time, subprocess

timeLeft = 10

while timeLeft > 0:

print(timeLeft,' ', end='')

time.sleep(1) # 每次延时 1秒

timeLeft = timeLeft - 1

subprocess.Popen(['start', '音乐.mp3'], shell=True)

注：subprocess.Popen 将把 [音乐.mp3] 文件交给windows系统， 系统会调用合适的播放器播放。

# 11、文件

Python提供了os ， os.path等模块处理文件、目录、路径。

读写文件时使用open函数。

open(filename , mode)

filename: 要打开的文件名

mode: 文件打开模式。如省略mode, 则默认值为r, 读文件

文本文件写入的例子：

#注 w是写文件，将覆盖旧文件

logfile=open(r'd:\log.txt','w') #以 w写文件方式创建文件

logfile.write("第一行 aaa\n") # 写入文件, \n是换行

logfile.write( "第二行 bbb\n")

logfile.write( "第三行 ccc\n")

logfile.close() #一定要关闭文件

注：（1）当我们写文件时，操作系统往往不会立刻把数据写入磁盘，而是放到内存缓存起来，空闲的时候再慢慢写入。只有调用close()方法时，操作系统才保证把没有写入的数据全部写入磁盘。忘记调用close()的后果是数据可能只写了一部分到磁盘，剩下的丢失了。

（2）文本文件读写时要注意自行添加或删除换行符\n。

文本文件读的常用方法：

f=open(r'd:\log.txt', 'r') #以 r 读模式打开文件 ,该文件必须存在

s1=f.readline() #读入第一行到字符串s1中，含换行符. s1="第一行 aaa\n"

s2=f.read(2) #再读2个字符到s2中, s2="第二"

s3=f.read() #将余下的部分都读入字符串s3中

print(s1)

print(s2) #输出第二

print(s3)

f.seek(0) #将文件指针重新移动到位置0 ，即文件头部

li=f.readlines() #将文件按行全部读入到一个列表中, 每行对应一个列表项

print(li) # li中包含文件全部内容

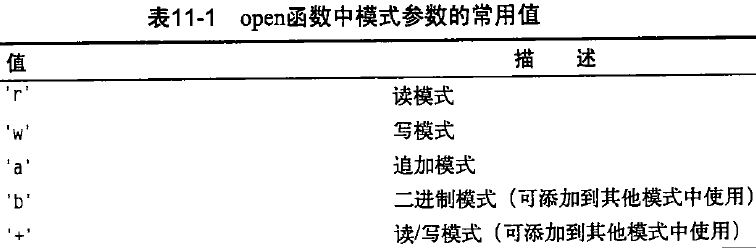
f.close()

文件写的常用方法：

file.write(s) #向文件写入一个字符串s， s必须是字符串

file.writeline**s**(list) #向文件写入一个列表list, 列表的元素均应为字符串

file.close() #关闭文件



文件模式说明：

r :只读。如文件不存在，则出错

r+ :读写。如文件不存在，则出错

w :写入, 先删除原内容，重新写入新内容。如文件不存在，则创建

w+ :读写

a :添加写入在文件末尾。如文件不存在，则创建

a+ :读写

b :以二进制模式打开可与 r w a + 结合使用

对于图片、视频等文件必须用b模式读写。

以r/w打开文件时，文件指针指向开始处，随着读写的进行而移动。以 a 打开时，文件指针在文件末尾。以r / a 打开是安全模式，不会删除原内容；以w打开会删除原内容。

文件读写时可假想有一个指针指向读写的位置。可用seek()函数移动文件指针:

f.seek(offset[, whence])

offset:是偏移量。可选的参数whence=0表示文件头；whence=1表示当前位置；whence=2表示文件尾。当whence=0时可省略此参数。

f.seek(0) 文件指针移动到文件头

f.seek(n) 文件指针移动到位置n

f.seek(m,1) 1表示从当前位置出发，再移动m( m>0向尾部移，m<0向头部移)

f.seek(0,2) 移动到文件末尾。 0,2组合表示距文件尾的偏移为0，即文件尾。

f.seek(-3,2) 移动到文件末尾前面第3个位置

f.tell() 返回文件指针位置

文件读写例子1

f=open('data.txt','w') # w新建文件data.txt。如data.txt已存在将被覆盖

f.write('aaaaa\n')

f.write('bbbbb\n')

f.close() #关闭文件

f=open('data.txt','a') # a 添加模式

f.write('ccccc\n')

f.write('ddddd\n')

f.close()

f=open('data.txt','r') #以读方式r打开文件data.txt

while True:

line=f.readline() #每次读一行

if line=='':

break #如line为空，表示文件已读完，跳出循环

print (line)

f.close()

文件读写例子2:

f=open('data.txt', 'r')

lines=f.readline**s**() #readlines一次读完文件，返回一个列表

f.close()

for line in lines: #依次输出每一行

print (line)

或

f=open('data.txt', 'r')

for line in f:

print (line)

f.close()

文件读写例子3 (使用with语句后可省略 close语句, 系统会自动关闭文件)

with open('data.txt', 'r') as filereader**:**

for line in filereader**:**

print(line)

注：上面的例子使用with语句打开文件，得到一个操作对象filereader, 然后使用该对象读取文件，读取结束后，系统会自动关闭文件, 无需再调用f.close。

#删除文件. 要先引入os, os.path

import os, os.path

if os.path.exists("hello.txt"): #先判断文件是否存在

os.remove("hello.txt") #如存在则删除。如删除一个不存在的文件将报错

#文件复制方法1

src=open(r"d:\hello.txt","r") # 以读方式打开源文件

#dst目标文件 r表示原始字符串，这样可将\视为路径，而不是转义字符

dst=open(r"d:\new.txt","w") # 以写方式创建目标文件

dst.write( src.read() ) #读源文件全部内容并写入目标文件

src.close()

dst.close()

#文件复制方法2

import shutil #使用shutil模块实现文件复制

shutil.copyfile('hello.txt', 'new.txt') #将hello.txt文件复制得到new.txt

shutil.move('new.txt', 'ok.txt') # 将new.txt改名为 ok.txt

shutil.rmtree("c:\photos") #删除整个目录

os.remove('ok.txt') #删除文件 ok.txt

二进制写入例子：按二进制写入数值数据

import struct #先要引入struct 模块

f=open('2.bin', 'wb') # b 二进制格式

f.write(struct.pack('i',2)) #写入时必须用struct.pack按整数i格式转换为字节串

f.write(struct.pack('i',-3)) #写入-3的补码（4字节长）

f.write(struct.pack('i',5))

f.close()

f=open('2.bin', 'rb') # 读取二进制格式文件

for i in range(3):

xx=f.read(4)

print(struct.unpack('i',xx)) #将输出 2, -3 , 5

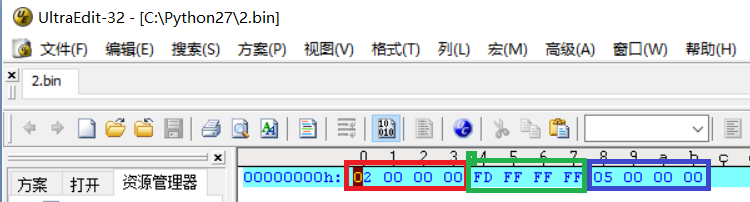
f.close()

执行上面程序后将得到2.bin的数据文件, 将此文件在ultraedit中打开 ,用十六进制形式观察, 结果如下:

02 00 00 00 : 是 2的 4字节补码

FD FF FF FF: 是 -3的 4字节补码

05 00 00 00 : 是5的4字节补码



struct.pack的格式字符如下：

c: 1个字节字符 b:1字节整数 B:无符号1字节整数 ?:1字节bool

H:2字节整数 i:4字节整数 L:4字节长整数

Q:8字节长整数 d:8字节浮点

目录的操作: os模块提供了针对目录的操作

import os

os.mkdir(dir) #创建目录 os.mkdir(r'd:\abc')

os.mkdir(r'd:\abc\d1\d2') # 如d1不存在将出错, 该命令只能一级一级的建立，不能同时建立多级目录

os.makedirs(dir) #允许一次创建多级目录

os.makedirs (r'd:\abc\d1\d2') #d1\d2同时创建

os.rmdir(dir) #删除空目录

os.rmdir(r'd:\abc\d1\d2') #删除d2子目录. 如目录不为空，先要删除其下的文件和子目录

os.listdir(dirpath) :以列表返回指定目录下的文件名和一级子目录名

os.listdir('d:\\') : 列出D盘根目录

os.listdir() : 列出当前目录的内容

os.getcwd() : 返回当前工作目录名

os.chdir(dir) :切换当前目录 os.chdir('c:\\')

例子：将当前工作目录修改为“c:\”，并验证，最后将当前工作目录恢复为原来的目录。

>>>import os

>>>os.getcwd()

'C:\\Python35'

>>>os.chdir(r'c:\\')

>>>os.getcwd()

'c:\\'

>>>os.chdir(r'c:\Python35')

>>>os.getcwd()

'c:\\Python35'

os.walk(dir) 返回一个遍历dir目录树的序列, 序列中的每一项是个三元组，其中第0项dirpath是完整的目录路径，第1项dirnames是dirpath中子目录列表，第2项filename是dirpath中每个文件。例子如下：

import os

li=list(os.walk(r'c:\abc')) # 列表c:\abc 文件夹

for dirpath,dirnames, filename in li:

print(dirpath, dirnames,filename)

[('c:\\abc', ['a', 'b'], ['notepad.exe']), ('c:\\abc\\a', [ ], ['f1.txt', 'f2.txt']), ('c:\\abc\\b', ['bb'], ['b1.txt', 'b2.txt']), ('c:\\abc\\b\\bb', [], [])]

import os.path

os.path.isfile(pathname) 判断是否是文件。返回True则是文件; 如是目录或不存在则返回False

os.path.isfile(r'c:\abc\data.txt')

os.path.exists(dir[filename]) #判断目录或文件是否存在, 返回True或False

os.path.isdir(pathname) #如果是目录返回True, 否则返回False

os.path.isfile("c:\windows") # 返回False

os.path.isdir("c:\windows") #返回True

os.path.splitext(filename) #分解文件名， 返回1个列表。列表第0个元素是主文件名，第1个元素是后缀名

li=os.path.splitext(r'data.txt') li[0] 为 data li[1]为.txt

os.path.split(filename) #返回1个列表。第0个元素是目录路径，第1个元素是文件名

li=os.path.split(r'c:\abc\data.txt') li[0] 为 c:\abc li[1]为data.txt

文件名通配符: 使用glob 模块

import glob

glob.glob("c:\python35\h\*") #返回该目录下所有 h开头的文件名构成的列表

glob.glob("c:\python35\\*.py") #返回该目录下所有的 .py文件

for path in glob.glob("c:\source\\*.bak"):

os.remove(path) #删除source目录下的所有.bak文件

把当前目录中后缀名为".html"的文件修改为以".htm"为后缀。

import os, os.path

files=os.listdir(".") # 返回当前目录下的文件名列表

for filename in files:

pos=filename.find(".") #判断文件名中 . 的位置

if filename[pos+1:]=="html": #如果后缀名为 .html

newname=filename[:pos+1]+"htm"

os.rename(filename, newname) #改后缀名为 .htm

遍历目录,输出所有的文件名:

用自定义递归函数visitdir实现

import os

def visitdir(path):

li=os.listdir(path) #列表包含当前目录下的文件名和一级子目录名

for p in li:

pathname=os.path.join(path,p)

if os.path.isdir(pathname):

visitdir(pathname) #如果是目录，则递归访问

else :

print(pathname) #如果是文件，则输出文件名

编写好后，可用下面语句访问c:\abc目录，列出所有文件名和子目录

visitdir(r'c:\abc')

>>> path=r'c:\python35\a1\p1.py'

>>> import os

>>> os.path.basename(path) #取得文件路径中的文件名

'p1.py'

>>> os.path.dirname(path) #取得文件路径中的目录路径

'c:\\python35\\a1'

# 12、面向对象编程OOP

程序设计主要有传统的结构化编程和面向对象编程两种模式。我们前面的例子都是采用结构化编程的方法编写的，下面介绍面向对象的模式。面向对象是指将要处理的数据定义为一个类(class), 类中包含了数据和施加在数据上的处理方法。OOP将数据和方法捆绑在类中， 传统的结构化编程中数据和处理方法是分离的。

下面我们来定义一个书籍类 Book. 书籍有书名，作者，价格等属性数据，还有修改价格，展示书籍等方法。

类定义时类名的首字符按惯例一般应大写。注意下面语句中的\_\_都是两条下划线。python中类的方法必须有一个self参数, 位于参数列表的开头, 代表对象本身，但调用方法时不用传递self这个参数。

class Book:

nums=0 #类属性, 全体实例都可访问

def \_\_init\_\_(self,bookname,author,price): #构造函数, 前后都是两根下划线

self.bookname=bookname #实例的公共属性

self.author=author #实例的公共属性

self.\_\_price=price #以两根下划线\_\_开头，实例的私有属性

def show(self) : #类的方法必须包含参数self,且必须为第一个参数

print ('书名:%-20s 作者:%-10s 价格:%8.2f ' % (self.bookname, self.author, self.\_\_price))

def modiprice(self , newprice) : # 修改价格的方法

self.\_\_price=newprice

def getprice(self) : #返回价格

return self.\_\_price

def \_\_check(self): #\_\_开头的是私有方法,不能在对象以外访问

print ('\_\_private check')

b1=Book('计算机导论','瞿中',34) # 创建对象b1, 会自动调用\_\_init\_\_

b1.show() #调用对象的show方法

b1.modiprice(50) #修改价格

b1.show() #再次show， 可看到价格已改变

print (b1.getprice())

b2=Book('Python','John Smith', 54) # 创建对象b2

b2.show()

\_\_init\_\_ 这个函数有特定含义，是构造函数。

如果要在类内部的方法中使用私有属性，则用 self.\_\_price 的形式

如果直接调用b1.\_\_check()将出错，因为\_\_check是私有方法，不能在对象外部调用。其实 Python并没有真正的私有方法，允许以b1.\_Book\_\_check()的形式(即对象名.\_类名.\_\_私有方法)访问私有方法,但这种访问方式不提倡。

dir(b1) #显示出上面的b1对象的相关内容

==========================

class Novel(Book) # 类的继承， Novel继续了Book类

class c3(c1, c2) # python允许多重继承 ， c3 继续了c1, c2 类

类的继承

class A(object): #默认object是基类

def \_\_init\_\_(self,name):

self.name=name

def hello(self):

print ("In A, I'm ", self.name)

class B(A): # 类B 继承了类 A

def \_\_init\_\_(self,name,age):

super().\_\_init\_\_(name) #super调用父类的构造方法

self.age=age

def hello(self): #重写同名方法

super().hello() #super 调用父类同名方法

print ("In B, I'm {0} age:{1} ".format(self.name, self.age))

b1=B('John', 35)

b1.hello()

# 13、异常处理

python具有异常处理语句try…except ，用于处理程序运行时发生的错误。使用异常处理可以允许程序员自行处理错误，避免程序异常退出或直接将复杂的异常信息显示给最终用户。

try:

x=float(input('x='))

y=float(input('y='))

print(x/y)

except ZeroDivisionError: #如y=0将引发此异常

print('除以0错误')

except (ValueError, TypeError): #如输入的非数值将引发此异常

print('值或类型错误')

except Exception as e: #捕获所有其它的异常

print('错误: ', e) # e是异常对象，可输出异常信息

else:

print('else') #else是可选子句。如不发生异常，else子句将执行

finally:

print('final') # finally是可选子句。不管有无异常，finally子句都要执行

当程序执行没有错误时，不会执行except子句。当有异常发生时才会执行except子句。

如果异常在函数内引发而不被处理，它就会传播至函数调用的地方，直至传播到主程序。

例子：使用try语句保证输入0-100的整数年龄

while True:

try:

age=int(input('请输入你的年龄0-100:'))

if 0<=age<=100: break

except ValueError:

print('要输入0-100的整数，请重试...')

print('你的年龄:' , age)

例子： 变量未定义时的异常

try:

print(xx)

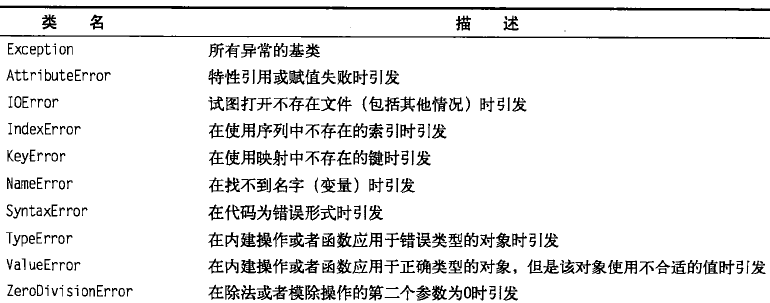
except Exception as e:

print(e) # 将输出 name 'xx' is not defined

print(type(e)) # 将输出 <class 'NameError'>

raise NameError('变量xx未定义') #可以用raise再抛出异常

一些内建异常类如下表：



# 14、杂项

1.运行.py程序有如下几种方法：

(1)在idle中打开.py源文件，然后选择 Run/Run Module

(2)在windows的命令行窗口中输入 python 程序文件名 执行, 例如：

c:\python35> python p1.py

(3)鼠标双击.py程序文件名执行。双击执行时，如窗口一闪就关闭了（这是因为windows在程序运行结束后立即自动关闭了控制台窗口），无法看清程序输出，可在程序最后加一行 input() 语句。加了input语句后窗口会一直保留在屏幕上，直到你按下回车键。

(4) 也可只输入程序文件名, 假定p1.py保存在c:\python35目录中

c:\python35>p1.py

(5) 在>>>交互状态下运行p1.py文件

>>>exec( open('p1.py').read() )

注意：使用第4/ 5种方法运行时，如程序中包含中文输出或注释，应在程序首行添加如下一行

#coding=cp936

这一行是指定采用windows目前的cp936编码集保存源程序。如果没有这一行运行时将因为中文编码的缘故报错。

2.python没有提供清空IDLE shell窗口的命令。如果IDLE窗口已满屏，新开一个IDLE窗口的方法如下：

(1)先"File/new file"，新建一个空白窗口；

(2)关闭原来满屏的IDLE窗口；

(3)在刚才新建的空白窗口中执行菜单" Run /Python Shell"即可新开一个IDLE窗口

3.调用windows中的程序：

import os

os.system('notepad.exe') #调用记事本

调用后控制权交给记事本, 需要关闭记事本才能继续运行原程序。

4. 在主程序执行到中途时退出程序（即结束程序运行）

import sys

#…………………

sys.exit() #退出

5. 命令行参数用sys.argv表示

#p1.py

import sys # 先引入sys模块

print (sys.argv[0], sys.argv[1]) # [0]程序名, [1]第1个命令行参数

执行c:\python35>python p1.py 123 时sys.argv[0]为'p1.py', sys.argv[1]为'123'

6.python2.x python3.x 两个版本的一些细节区分

以前python2.x使用较多，目前已转为python3.x。python2.x 和 python3.x 大部分内容兼容，有一些方面不同：

(1) 2.x可以写print x也可以写print(x) ; 3.x必须写print(x)

2.x 不换行输出可写为print x , #后面多加一个逗号

3.x 不换行输出可写为print(x , end='')

(2) 2.x中整数除法自动取整 23/5 结果为4。3.x中则保留小数， 23/5结果为4.6

(3) 2.x中input()语句根据输入内容自动决定数据类型，在输入字符串时必须有单引号定界符。3.x的input()将所有输入都按字符串处理，不需要定界符，且没有2.x中的raw\_input()函数了。

(4) 2.x x=range(4) 则x=[0,1,2,3]

3.x x=list(range(4)) 则x=[0,1,2,3]，必须用list再处理一次才能得到列表

(5) 在一些语句和函数库的调用上不同。 2.x中的一些调用在3.x中有变动。

7.python之禅

执行>>>import this 将显示下面的内容

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

…..

翻译为中文：

Python之禅 by Tim Peters

优美胜于丑陋（Python 以编写优美的代码为目标）

明了胜于晦涩（优美的代码应当是明了的，命名规范，风格相似）

简洁胜于复杂（优美的代码应当是简洁的，不要有复杂的内部实现）

复杂胜于凌乱（如果复杂不可避免，那代码间也不能有难懂的关系，要保持接口简洁）

扁平胜于嵌套（优美的代码应当是扁平的，不能有太多的嵌套）

间隔胜于紧凑（优美的代码有适当的间隔，不要奢望一行代码解决问题）

可读性很重要（优美的代码是可读的）

即便假借特例的实用性之名，也不可违背这些规则（这些规则至高无上）

不要包容所有错误，除非你确定需要这样做（精准地捕获异常，不写 except:pass 风格的代码）

当存在多种可能，不要尝试去猜测

而是尽量找一种，最好是唯一一种明显的解决方案（如果不确定，就用穷举法）

虽然这并不容易，因为你不是 Python 之父（这里的 Dutch 是指 Guido ）

做也许好过不做，但不假思索就动手还不如不做（动手之前要细思量）

如果你无法向人描述你的方案，那肯定不是一个好方案；反之亦然（方案测评标准）

命名空间是一种绝妙的理念，我们应当多加利用（倡导与号召）

# 15.常用模块

(1) sys模块

import sys

sys.platform #显示操作系统平台名字例如win32

sys.path #是一个目录名列表，在import语句执行时，解释器将从这些目录中查找模块

sys.path.append(r'e:\mypy') #添加一个搜索目录

sys.argv #程序运行时的命令行参数 argv[0] 是程序名, argv[1]第0个参数

sys.exit() #退出当前程序

sys.version #返回python的版本

sys.getdefaultencoding()#返回utf-8 的默认编码字符集

（2） os , os.path, shutil 模块

import os, os.path

os.system('notepad.exe') #调用外部程序

os.getcwd() #返回当前目录名

os.chdir('e:\\abc') #切换当前目录到 e:\abc

os.listdir(r'c:\abc') #列出指定目录下的所有文件名和子目录名

os.listdir() #列出当前目录下的所有文件名和子目录名

os.rename('test.txt', 'test.py') # 将test.txt重命名为 test.py

os.remove('test.py') # 删掉文件

os.path.abspath('data.txt') # 返回文件的绝对路径 C:\\Python35\\data.txt

os.path.exists(r'c:\windows') # 返回True 判断路径是否真正存在

os.path.isfile("c:\windows") # 返回False , 判断是否是文件

os.path.isdir("c:\windows") #返回True，, 判断是否是目录

os.path.split(r'c:\mydata\abc\data.txt')

('c:\\mydata\\abc', 'data.txt') #分隔文件路径，分别得到目录路径和文件名

os.path.splitext('data.txt')

('data', '.txt') #分隔文件名，得到主文件名和扩展名

import shutil #引入shutil模块

shutil.copyfile('hello.txt', 'new.txt') #复制hello.txt得到new.txt

shutil.move('new.txt', 'ok.txt') # 将new.txt改名为 ok.txt

（3） platform模块

import platform

platform.platform() #返回操作系统名称和版本

platform.system() #返回操作系统名称windows

platform. machine() #cpu硬件类型 AMD64

platform.node() #计算机的网络名称

platform.processor() # cpu 信息

platform.uname() #返回一个综合信息

platform.python\_version() # python 版本信息

（4）随机数random模块

该模块的常用方法:

choice :接受一个列表，随机返回其中的一个元素。

sample :接收两个参数，第一个表示一个列表，第二个是一个整数，表示从该列表中随机取出的元素个数， 该取样为不重复取样。

shuffle :接收一个列表参数，但是它无返回值，直接对该列表进行随机重排

random :返回一个从0到1之间的实数。

uniform :接收两个数，返回这两个数之间的一个实数。

randrange(a,b) :返回 [a, b) 之间的一个整数(不含b)。

randint(a,b) :返回[a，b]之间的一个整数值(含b)。

举例如下：

import random

li=[3,4,5,8,9,10]

print(random.choice(li)) #从li中随机选一个

print(random.sample(li,3)) #从li中随机选三个，该取样为不重复取样

print('重排前的序列',li)

random.shuffle(li) #对li 随机重新排列

print('随机重排后的序列',li)

print(random.random()) #返回 0- 1 之间的随机小数

print(random.uniform(1.5, 3.5)) #返回 1.5 - 3.5 之间的随机小数

print(random.randrange(20)) #返回 [0, 20) (含0，不含20)之间的随机整数

print(random.randint(2,8)) #返回 [2, 8] （可含2或8）之间的随机整数

(5)统计模块collections

import collections #先引入该模块

s=['a', 'b', 'a','c','d','b','a']

c=collections.Counter (s) #统计各数据项的出现的次数

print('统计结果',c)

#统计结果 Counter({'a': 3, 'b': 2, 'c': 1, 'd': 1})

print('最多的2个元素是：' , c.most\_common(2))

#最多的2个元素是： [('a', 3), ('b', 2)]

print('d的个数是：',c['d'])

(6) glob模块

提供了一个函数用于从目录通配符搜索中生成文件列表:

**>>> import glob**

**>>>** glob.glob("\*.py") #列出当前目录下所有的.py文件

[’p1.py’, ’p2.py’, ’quote.py’]

# 16.常见错误信息

1.缺少缩进。 下级语句应缩进。

while k<=num**:**

s=s+k

k=k+1 #这两行是while的下级语句，应缩进

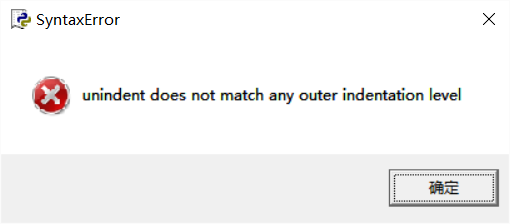


2.同一级别的语句缩进不匹配

while k<=num:

s=s+k #

k=k+1 #前后两行的缩进不一致，将报错



注：当你从网上复制粘贴python代码时，一定要检查缩进。网上的很多代码由于网页显示的问题无法保留正确的空格数。

3.变量未定义

y=x+100

如果x变量事先不存在，执行时将报错

Traceback (most recent call last):

File "C:\Python35\p1.py", line 1, in <module>

y=x+100

NameError: name 'x' is not defined

4. 数据类型不正确

num=input('请输入一个正整数:') # input输入的数据都是字符串

x=num+100

Traceback (most recent call last):

File "C:\Python35\p1.py", line 2, in <module>

x=num+100

TypeError: Can't convert 'int' object to str implicitly

input 输入的数据是字符串，例如输入50， 则num='50', 字符串不能和整数100直接相加。应改为:

num=int(input('请输入一个正整数:') ) #用int将字符串转换为整数

5.索引下标越界

li=[0,1,2]

print(li[3])

IndexError: list index out of range

li 列表只有3个元素，下标为0,1,2. li[3]下标3越界了

6. 函数参数个数/类型不正确

>>> abs(1,2)

Traceback (most recent call last):

File "<pyshell#159>", line 1, in <module>

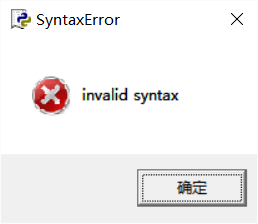
abs(1,2)

TypeError: abs() takes exactly one argument (2 given)

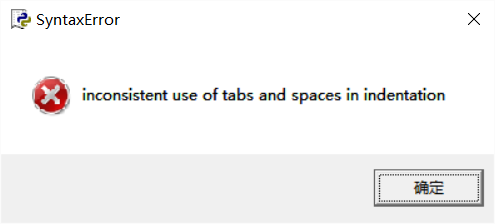
abs绝对值函数只需要1个参数， 上例提供了2个参数

7. 非法的语法, 缺少冒号

while k<=num # if while for 等语句后应有冒号:



8.混用了tab和空格进行缩进： 有时从网上和别的编辑软件中将代码复制过来，其中可能混用了tab和空格来产生缩进，就报下面的错误。此时可以将前句、本句、后句的前后空白缩进全部删除（因为这些空白混合了TAB，python将报错），然后在IDLE中重新按TAB键产生正确的缩进。



# 附录：编程练习题

**实践是提高编程能力和问题解决能力的最好方法！**

学习编程的首要条件是对编程本身的热爱，对于探索横亘于人脑和机器之间的神秘空间、如何使机器满足人类的需求抱有强烈的探索需求。第二个条件是允许失败。编程是一门设计算法的艺术，同时是一门调试错误程序的手艺。

1. 编写程序p1.py, 用户输入一个三位自然数，输出百位、十位、个位。

#coding=cp936

x=int(input('请输入一个三位数:')) # 用int 将输入的字符串转为整数

a=x//100

b=(x-100\*a)//10

c=x-100\*a-10\*b

print("百位、十位、个位依次为:", a, b,c)

print("百位、十位、个位依次为:%d %d %d" % (a, b,c) )

print("百位、十位、个位依次为:{0} {1} {2}".format(a, b,c) )

解释：（1）#coding=cp936是指定中文编码字符集。如果你用记事本（而不是用python的IDLE）来编辑程序，当程序中含有中文时应包含该行。因为记事本默认以cp936编码保存信息，当python在命令行下执行该文件时, 例如 :

c:\python35>python p1.py

python就知道以cp936编码来处理中文， 这样可正确执行程序。如果没有cp936编码指定，python将按默认的utf-8编码处理，此时命令行上 c:\python35>python p1.py 就会报错。

如果是用IDLE来编辑程序，则可以不用包含cp936这一行。 因为python默认用utf-8保存中文, 其在命令行下执行时也以utf-8来处理中文。

（2）程序最后特意演示了三种字符串格式化输出方法, 大家以后做输出时可模仿这些语法。

另一种方法：

s=input('请输入一个三位数:')

print ("百位、十位、个位依次为:", s[0], s[1],s[2])

2. 输入一元二次方程ax2+bx+c=0的系数 a, b, c , 解方程。

import math #要导入math模块，才能使用sqrt

a,b,c=eval(input('请输入一元二次方程的a,b,c系数，以逗号分隔:'))

p=b\*b-4\*a\*c

if p<0:

print ('系数不合理，方程无实数解')

else:

x1=(-b+math.sqrt(p))/(2\*a)

x2=(-b-math.sqrt(p))/(2\*a)

print('方程的两个解是:', x1, x2)

解释：例如输入 1, 2, -3 时，eval将其转换为元组(1,2,-3)，再赋值给a,b,c三个变量

注：输入a,b, c 也可用下面的方法

s=input('请输入一元二次方程的a,b,c系数，以逗号分隔:')

a,b,c=[float(x) for x in s.split(',')]

输入1, 2, -3, s.split(',') 分解得到['1', '2', '-3'], 然后再逐一转为浮点数，赋值给a,b,c

3.输入若干个成绩,当输入-1时表示结束输入。然后输出最高、最低、平均成绩，并按成绩由高到低输出。

li=[] #定义空列表，用于容纳成绩

x=0

while x!=-1:

x=float(input('请再输入一个成绩(输入-1结束):'))

if x!=-1:

li=li+[x] #添加到成绩列表中

else:

break #如输入-1,则跳出while循环

if len(li)==0: #len 是返回列表长度的函数 / 相等比较符是两个等号

print ('成绩列表为空，没有输入成绩')

else:

print ('输入数据个数:',len(li))

print ('最高成绩:',max(li))

print ('最低成绩:',min(li))

print ('平均成绩:',round(sum(li)/len(li),1)) #保留1位小数

print ('由高到低为：',sorted( li,reverse=True ) ) #逆序排

注:此处的max,min ,sum, len, sorted 等都是python提供的函数

4. 输入一个十进制正整数，用除2取余法输出其二进制串。注：不能使用python自带的bin函数。

x=int(input('请输入一个正整数:'))

y=x

if x==0:

s='0'

else:

s=''

while y>0:

z=y%2 #除以2，取余数

y=y//2 #商

s=str(z)+s #将余数加到转换结果的高位

print (x,'转二进制:',s)

5.编制一个将十进制整数转换为十六进制的函数,用除16取余法。

def tohex(y): #定义一个十进制转16进制的函数

s=''

while y>0:

z=y%16 #余数

y=y//16 #整除，商

s= '%X' % z + s # 将余数z用%X格式化为十六进制数

return '0x'+s

print(12,tohex(12))

print(100,tohex(100))

print(1000,tohex(1000))

6.输入若干个成绩，输入负数表示结束。请排序并统计优（100-90），良（89-80）、中（79-60）、差（59-0）4个等级的人数。

a,b,c,d=0,0,0,0 #定义4个变量，对应每个等级的人数

cj=[]

while True:

x=float(input('请输入一个成绩(负数结束):'))

if x<0:

break #如为负，则跳出循环

cj=cj+[x] #否则就添加到成绩列表中

if x>=90:

a=a+1

elif x>=80:

b=b+1

elif x>=60:

c=c+1

else:

d=d+1

print('输入的原始成绩为:', cj)

print('排序后为:', sorted(cj,reverse=True))

print('优（100-90）:', a)

print('良（89-80）:', b)

print('中（79-60）:', c)

print('差（59-0）:', d)

7. 如果年份能被400整除，则为闰年；如果年份能被4整除但不能被100整除也为闰年。编程判断输入的年份是否为闰年。

x=int(input('请输入一个年份:'))

if (x%400==0) or (( x%4==0) and (x%100!=0)):

print(x, '是闰年')

else:

print(x, '不是闰年')

8.编程输出斐波拉契数列的前20项，已知该数列第一、二项分别是0,1，从第三项开始，每一项都是前两项之和。例如: 0 ,1, 1, 2, 3, 5, 8,…

li=[0,1]

k=2

while k<20:

x=li[-2]+li[-1] # 新项=已有数列的倒数第2项 + 倒数第1项

li=li+[x]

k+=1

print (li)

9.编写程序，将输入的一行英文字符串颠倒顺序后保存在文件data.txt中。

例如输入 abc123 , 则在文件中保存为321cba

s1=input('please input string :')

s2=s1[ : : -1] #用-1 的步长即可实现颠倒取字符

print ('s1=',s1)

print ('s2=',s2)

f=open('data.txt','w') #创建一个data.txt文件

f.write(s2) #将s2字符串写入文件

f.close() #关闭文件

颠倒顺序也可用下面的代码实现

k=len(s1) - 1

s2=''

while k>=0:

s2=s2+s1[k]

k=k-1

print ('s2=',s2)

10. 输入一个任意的正整数，输出各个位上的数字。

s=input('请输入一个数:')

for k in s:

print(k) #依次输出各个位

或者

s=input('请输入一个数:')

for k in range(len(s)):

print (s[k]) #依次输出各个位

11. 猜数游戏：计算机随机产生1-20间的一个整数，用户来猜测。计算机根据用户输入的数提示猜测值偏大还是偏小，最多允许猜5次。

#coding=cp936

import random #要先导入random随机数模块

num=random.randint(1,20) #产生 [1-20] 间的一个随机整数. 可含1或20

k=1 #猜测次数

while k<=5 :

x=int(input('请输入一个[1-20]间的猜测数:'))

if num>x:

print( '偏小')

elif num<x:

print ('偏大')

else:

break # 如猜中了则跳出循环

k=k+1

if k<=5:

print ('你猜对了。次数:%d' % k)

else:

print ('你没有猜对。随机数为:%d' % num)

12.输出9\*9口诀。

for k in range(1,10):

for j in range(1,k+1):

print (j,'\*',k,'=',j\*k, ' ', end='') #end='' 是为了输出时不换行

print('') #用于换行

注：本例运行时，将IDLE窗口调宽一些，以便能看到9\*9 完整输出。

13.判断101-200之间有多少个素数，并输出所有素数。

import math

li=[]

for k in range(101,201):

bz=0

for j in range(2, int(math.sqrt(k))+1):

if k % j==0:

bz=1 #设为1，表示找到一个可整除的数，则k不是素数

break

if bz==0: #如结束上面的for循环后, bz还为0，表明k是素数

li=li+[k]

print (li)

print ('101到200，素数个数：' , len(li))

14. 求1!+2!+3!+...+N!的和. N为输入值。

n=int(input('n='))

s=0

k=1

j=1

while k<=n:

j=j\*k # j 依次为 1 , 2! , 3!, 4!, … n!

s=s+j

k+=1

print ('阶乘累加和 1!+2!+...%d! =%d' % (n,s) )

本例也可直接调用python的factorial阶乘函数, 写为

import math

n=int(input('n='))

s=0

for x in range(1,n+1):

s=s+math.factorial(x)

print ('阶乘累加和 1!+2!+...%d! =%d' % (n,s) )

15. 有1、2、3、4个数字，能组成多少个互不相同且无重复数字的三位数（输出各个数）？

li=[]

for i in range(1,5):

for j in range(1,5):

for k in range(1,5):

if i!=j and i!=k and j!=k:

li=li+[i\*100+j\*10+k]

print ('个数为：', len(li))

print ('数据依次为:', li)

另一种做法，直接调用python已有的排列函数permutations

import itertools # 先引入该模块，以调用其中的排列函数permutations

digits=[1,2,3,4]

li=list(itertools.permutations(digits,3)) # 从digits中任取3个数，算排列

print ('个数为：', len(li))

print ('数据依次为:', li)

注：itertools.combinations(digits,3) # 从digits中任取3个数，算组合

16.输入一行字符，分别统计出其中英文字母、空格、数字和其它字符的个数

import string

s = input('input a string:')

letter = 0

space = 0

digit = 0

other = 0

for c in s:

if c.isalpha(): #判断是否是字母

letter+=1

elif c.isspace(): #判断是否是空格

space+=1

elif c.isdigit(): #判断是否是数字

digit+=1

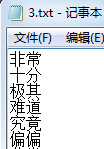
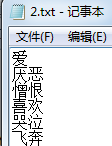
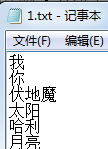
else:

other+=1

print ('字母:{0}, 空格:{1}, 数字:{2}, 其它:{3}'.format(letter,space,digit,other))

17.计算机随机创作诗。本例练习文件读写、随机选择等概念。

假定在文件1.txt中存放名词/代词，2.txt中存放动词,3.txt中存放副词/形容词(每行存一个词)。计算机先从这3个文件中读入词，然后随机拼凑成若干行诗歌。注意这三个文件应先自行创建，并存在当前目录c:\python35中。



import random

def readfile(filename): #定义读文件的函数

f=open(filename, 'r');

li=f.readlines() #将文件内容全部读入列表

f.close( )

s=' '.join(li) #将各行连接为一个长字符串

li=s.split() #以空格为分割符，分割得到词语列表

return li #读出的词语以列表形式返回

c1=readfile('1.txt') #读入名词

c2=readfile('2.txt') #读入动词

c3=readfile('3.txt') #读入副词

f=open('poem.txt','a') #poem.txt用于保存随机产生的诗

f.write('\n') #先插入一个空行

for k in range(8):

# 按名词+副词+动词+名词随机组合

# random.choice(列表) 从列表中随机选一个词

s=random.choice(c1)+ random.choice(c3)+random.choice(c2)+random.choice(c1)

s=s+'\n' # 最后加一个换行符

print (s)

f.write(s) #将随机产生的诗保存在文件poem.txt

f.close() #关闭poem.txt文件

18. 21点纸牌游戏。规则：一副牌（不含王牌）， 10，J, Q, K都算10点，其余牌按牌面点数算。你和电脑先各发一张暗牌，然后轮流决定是否再要牌，后发的牌是明牌。最后计算两方的牌点数之和，接近21点者算赢，超过21点则算爆仓。如果平分，算电脑赢。如果都爆仓，则平局。

import random

def getcard(): #产生一副牌

li=list(range(1,11))\*4 + 'J Q K'.split()\*4

random.shuffle(li) #将li列表随机打乱，相当于洗牌

return li

def calc(li): #计算点数

s=0

for x in li:

if x in ['J','Q','K']:

s=s+10

else:

s=s+x

return s

c1=c2=0 #记录各自赢的次数

count=0 #总局数

while True:

count+=1

card=getcard()

card1=[card[0]] #card1代表你的牌，先发一张

card2=[card[1]] #card2代表电脑的牌，先发一张

k=2 # k 是待发的牌的序号

need1=need2=True #True 表示还要要牌

while need1 or need2:

print ( '你的牌:',card1, ' 电脑的牌: ? ',card2[1:]) #电脑的第0张牌是暗牌，用?表示

if need1:

yn=input('你还要牌吗？ (按Enter or N)')

if yn.upper()=='N':

need1=False

else:

card1.append(card[k]) #你再要一张牌

k=k+1

ds1=calc(card1) #计算点数

ds2=calc(card2)

if ds1>21:need1=False #如爆仓，则你不能再要牌

if ds1>21 or ds2>16: #电脑判断，如你已爆仓，或电脑点数>16, 则电脑不再要牌

need2=False

else:

card2.append(card[k]) #电脑再要一张牌

k=k+1

print()

ds1=calc(card1)

ds2=calc(card2)

print ('你的牌点数:',ds1,card1, ' 电脑的牌点数:',ds2,card2)

if ds1>21 and ds2>21 :

print ('你们都爆仓了')

elif (ds1>ds2 and ds1<=21) or ds2>21:

print ('你赢了')

c1+=1

else:

print ('电脑赢了')

c2+=1

yn=input('再玩一局吗？ (按Enter or N)')

if yn.upper()=='N': break

print ('总局数:',count, ' 你赢:',c1, ' 电脑赢:',c2)

19.统计一篇英文文件中出现次数最多的10个单词。

import string

def countfile(filename): #定义统计函数 入口参数：文件名

f=open(filename,'r')

data=f.read() #读取文件所有的字符

f.close()

print('文件名:',filename)

print('总字符数:', len(data))

print('总行数:', data.count('\n')) #统计换行符，得到行数

print('单词数:', len(data.split())) #分隔，得到单词总数

#下面将字符处理一下，只保留字母和空格，且都转小写

char=string.ascii\_lowercase #所有小写字母

char=char+ ' ' +'\'' #再加一个空格 和 一个单引号 I'd

s=''

for x in data:

if x.lower() in char: #判断每个字符， 不属于char的就丢弃

s=s+x.lower() # 属于char 的就加入s , 相当于所有字母转小写

print(' ')

print('清除其它字符，只保留 字母 空格 \' 后:')

print('总字符数:', len(s))

words=s.split() #再次分隔，这次得到的单词都是小写形式

#统计各单词次数 方法1

di={ } #空字典

for x in words:

if not x in di: #如果该单词还不在字典中

di[x]=1 #则加入字典， 初始次数1

else:

di[x]=di[x]+1 #该单词已在字典中，则次数加1

print('共有不同单词个数:', len(di))

num\_words=[ (num,k) for k, num in di.items()] #将字典变为元组列表

num\_words.sort(reverse=True) #按次数排序, 每个元组的[0]是次数，

print('出现次数最多的10个单词是:')

for x in range(10):

print(num\_words[x][0] , num\_words[x][1]) #[0]是次数 [1]是单词

print(' ')

#统计各单词次数 方法2

import collections #用collections 模块 统计更快捷方便

c=collections.Counter (words) #统计各数据项的出现的次数

print('出现次数最多的10个单词是：' , c.most\_common(10) )

if \_\_name\_\_=='\_\_main\_\_':

countfile('傲慢与偏见英文版.txt') #测试

20. 命令行方式的网络群聊室。（仅供参考，不要求掌握）

本系统分3个子程序：

(1) chatserver.py ：群聊服务器,负责接收并转发聊天信息

(2) chatshow.py :负责接收服务器转发过来的信息，在屏幕上显示

(3) chat.py :负责将信息发送给服务器

chatserver

chatshow

chat

chatshow

chat

假定这三个程序都放在 c:\python35目录中

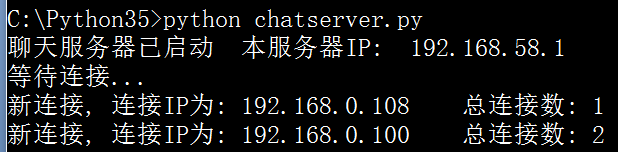
进入命令行窗口 , 执行 cd \python35 , 将当前目录切换到 c:\python35目录中

c:\python35>

（1）.聊天时，先仅在一台电脑的命令行窗口中运行

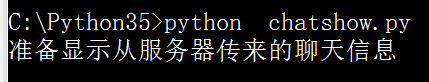
c:\python35>python chatserver.py

运行后，将显示出服务器的ip. 类似下图：



（2）然后再开一个命令行窗口，执行下面命令

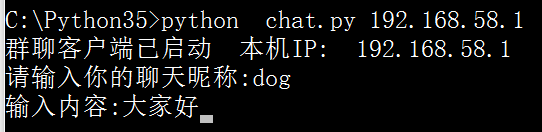
c:\python35>python chatshow.py



（3）再开一个命令行窗口，执行下面命令

c:\python35>python chat.py 服务器IP

注意： 运行时要改成服务器的IP， 例如



**chatserver.py 如下：**

# coding=cp936

# 运行方法【python chatserver.py 】

import socket, sys

from threading import \*

MAX\_BYTES=65535 #定义数据包最大字节数

def handlechild(clientsock): #定义接收信息的函数

try:

while True:

data=clientsock.recv(MAX\_BYTES) #收到信息

info=data.decode('utf-8') #收到的是bytes, 按utf-8解码为字符串

if info!='':

ip=clientsock.getpeername()[0] #得到发送方的ip地址

for client in links:

try: #将收到的信息转发给所有已连接的客户端

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

data=str(ip) +':'+info

data=data.encode('utf-8') #变为 bytes 才能发送

sock.sendto(data, (str(client), 4900)) #发往客户机4900端口

except Exception as e:

links.remove(client) #如发送失败，则去掉该客户连接

except Exception as e:

print ('线程出错:',e)

try:

links=[ ] #用于包含所有连接的客户端IP地址

localIP = socket.gethostbyname(socket.gethostname()) #得到本地ip

print ('聊天服务器已启动 本服务器IP: ',localIP)

s=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # TCP

s.bind(('',5000)) # 服务器在端口5000 处等待客户机连接

s.listen(5) #启动监听

print ('等待连接...')

while True:

clientsock, clientaddr=s.accept() # 有客户端来连接

if clientaddr[0] not in links: # 如果ip还不在表中

links.append(clientaddr[0]) #将新连接的 ip 加入列表

print ('新连接, 连接IP为:', clientaddr[0], ' 总连接数:',len(links))

t=Thread(target=handlechild,args=[clientsock]) #以线程方式处理客户端连接

t.setDaemon(1) #线程需要的语句

t.start() #运行线程

except Exception as e:

print ('server 错误：', e)

**chatshow.py 如下：**

# coding=cp936

# 运行方法【python chatshow.py 】

import socket,sys

MAX\_BYTES=65535 #定义数据包最大字节数

print('准备显示从服务器传来的聊天信息')

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) # udp

sock.bind(('', 4900)) #在4900端口等待服务器传来的信息

while True:

data, address=sock.recvfrom(MAX\_BYTES) #接收信息

print(data.decode('utf-8')) # 收到的是bytes, 解码为字符串再显示

s.close()

**chat.py 如下**

#coding=cp936

# 运行方法【python chat.py 服务器IP 】

import socket, sys

try:

server='127.0.0.1' #默认连接本机

server=sys.argv[1] #如有输入服务器IP，则连接服务器

except:

pass

localIP = socket.gethostbyname(socket.gethostname()) #得到本地ip

print ('群聊客户端已启动 本机IP: ', localIP)

name=input('请输入你的聊天昵称:')

name='('+name+')'

s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) # TCP

s.connect((server,5000)) #连接聊天服务器

while True:

data=input('输入内容:')

if data!='':

s.send((name+data).encode('utf-8')) #将信息转bytes才能发送到服务器

s.close()

21. 刑侦队正在调查一桩疑案，案件涉及六个嫌疑人，已掌握的证据表明：

* A 、B至少有一人作案；
* A、E、F三人中至少有两人参与作案；
* A 、D不可能是同案犯；
* B、C或同时作案，或与本案无关；
* C、D中有且仅有一人作案；
* 如果D没有参与作案，则E也不可能参与作案。

试编一程序，将作案人找出来。

# 1表示罪犯， 0表示不是罪犯, x 表示不明确. 上述6条证据表示如下：

r1=['10xxxx','01xxxx','11xxxx']

r2=['1xxx1x','1xxxx1','xxxx11','1xxx11']

r3=['0xx0xx','0xx1xx','1xx0xx']

r4=['x11xxx','x00xxx']

r5=['xx10xx','xx01xx']

r6=['xxx00x','xxx1xx']

def hebin(s1,s2): # 将两个约束条件合并

li=[]

for c1,c2 in zip(s1,s2):

if c1=='x': # c1为x,则取c2

li.append(c2)

elif c2=='x': # c2为x,则取c1

li.append(c1)

elif c1==c2: # c1 /c2 同为0 或 1

li.append(c1)

else: # c1 / c2 一个为0，一个为1，有冲突

li.append('n') # 用 n 表示此项目的合并有冲突

return ''.join(li) # 将合并结果连接为字符串返回

rule=['xxxxxx'] # 初始状态

k=0 # 用于记录比较次数

for r in [r1,r2,r3,r4,r5,r6]:

if len(r)==0:continue # 当该列表中无约束条件时则跳过

li=rule[:] # 上一轮的约束条件

rule=[ ]

for s1 in li:

for s2 in r:

s= hebin(s1,s2) # 将两个约束条件合并

k+=1

if s.find('n')== -1: #合并后如没有冲突，

rule.append(s) # 则加入到本轮的规则列表中

print('推导结果为:',rule)

print('比较次数:',k)

输出结果为：

推导结果为: ['111001'] # 表明罪犯是 ABCF

比较次数: 92