# 列表

## @@斐波那契数列

# fib = [0, 1]

# for i in range(8):

# fib.append(fib[-1] + fib[-2]) # 将列表中最后两项之和，追加到列表中

# print(fib)

###############################

fib = [0, 1]

n = int(input('长度: '))

for i in range(n - 2):

fib.append(fib[-1] + fib[-2])

print(fib)

## @@猜数字游戏

import random

all\_choice = ['石头', '剪刀', '布']

win\_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]

prompt = '''(0) 石头

(1) 剪刀

(2) 布

请选择(0/1/2): '''

pwin = 0

cwin = 0

while pwin < 2 and cwin < 2: # 人机都没赢够2次，继续循环，三局两胜

computer = random.choice(all\_choice)

ind = int(input(prompt)) # 将用户输入的字符转换为数字

player = all\_choice[ind] # 通过数字下标，取出列表中的字符串

print('你的选择是:%s, 计算机出拳: %s' % (player, computer))

if player == computer:

print('\033[32;1m平局\033[0m')

elif [player, computer] in win\_list:

pwin += 1

print('\033[31;1mYou WIN!!!\033[0m')

else:

cwin += 1

print('\033[31;1mYou LOSE!!!\033[0m')

## @@九九乘法表

# for i in range(3): # i 控制打印几行

# # 一行内打印3个hello后，打印回车

# for j in range(3): # 内层循环控制一行内打印几次hello

# print('hello', end=' ') # print默认在结尾打印\n，将结尾修改为空格

# print()

########################

# for i in range(3): # i 控制打印几行

# for j in range(i + 1): # 内层循环控制一行内打印几次hello

# print('hello', end=' ') # print默认在结尾打印\n，将结尾修改为空格

# print()

########################

for i in range(1, 10):

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

print('%sx%s=%s' % (j, i, i \* j), end=' ')

print()

~

# 遍历

chs = 'hello'

alist = ['bob', 'alice']

atuple = (10, 20)

adict = {'name': 'tom', 'age': 23}

for ch in chs:

print(ch)

for name in alist:

print(name)

for i in atuple:

print(i)

for key in adict:

print('%s: %s' % (key, adict[key]))

# while循环

# 列表解析

## 通过for循环

# print([10 + 5 for i in range(5)])

#输出：[15, 15, 15, 15, 15]，#10+5这个计算5次，for循环了5次

# print([10 + i for i in range(5)])

#输出：[10, 11, 12, 13, 14] #10+i,i来自于后面的for循环

# print([10 + i for i in range(2,5)])

#输出：[12, 13, 14] #10+i,i来自于后面的for循环

## 通过if过滤，

满足if条件的才参与10+i的运算

# print([10 + i for i in range(1,11) if i % 2 == 0]) #输出[12, 14, 16, 18, 20]

# print([10 + i for i in range(1,11) if i % 2 == 1]) #输出[11, 13, 15, 17, 19]

# print(['192.168.1.%s' % i for i in range(1,10) if i > 8]) #输出['192.168.1.9']

# a = ['192.168.1.%s' % i for i in range(1,10) if i > 8]

# print(a) #输出['192.168.1.9']

# 文件对象：

文件操作三个步骤：打开、读写、关闭

* r,以读方式打开(文件不存在则报错)
* w,以写方式打开(文件存在则清空,不存在则创建)
* a,以追加模式打开,(必要时创建新文件)
* b,以二进制模式打开,以bytes方式打开(直接显示为2进制的方式)
* r+/w+/a+,以读写模式打开

## 1.read,默认读取文件的全部内容

# f = open('/tmp/passwd1') #打开文件

# data =

# #读取文件

# print(data) #输出

# #读取文件时,文件指针会向后移动,读取全部内容后文件指针已经到结尾,再读就没有内容了

# # data = f.read()

# # print(data)

# data2 = f.read()

# print(data2) #data2是没有数据的,因为第一个data = f.read()全部读取完毕了

# data = f.close() #关闭

# f = open('/tmp/passwd2')

# data = f.read(4) #取前面四个字符

# print(data)

# data2 = f.read(4) #接着上面再读取四个字符

# print(data2)

# f.close() #关闭f 打开的文件

# print(f.read(4)) #报错,已经被关闭ValueError: I/O operation on closed file.

## 2.readline,读一行####

# f = open('/tmp/passwd1')

# a = f.readline() #读取第一行

# print(a)

# b = f.readline() #接着上面读取第二行

# print(b)

# f = open('/tmp/passwd1').readline() #不建议使用,无法读取第二行

# print(f)

## 3.readlines 读取所有行,放到列表中,

# f = open('/tmp/passwd2')

# a = f.readlines()

# print(f) #输出一个列表,每个value就是文件中的一行

## 4.通过for循环进行遍历,常用,需要记住

# f = open('/tmp/passwd2')

# for line in f:

# print(line,end='')

# f.close() #关闭

## 5.读取非文本文件

# f = open('/tmp/windows.jpg')

# a = f.read(4) #报错,默认情况下,文件被认为是文本文件,python试图将取出的4字节转换成文字,图片无法转换成文字

# f.close()

# f = open('/tmp/windows.jpg','rb') #以rb的方式打开,

# print(f.read(4)) #输出b'\xff\xd8\xff\xe0' 16进制数,python将2进制以16进制方式进行显示

# print(f.read(4)) #输出b'\x00\x10JF' #接着上面的显示

# f.close()

# f = open('/tmp/windows.jpg','rb') #以rb的方式打开,

# print(f.read(8)) #输出b'\xff\xd8\xff\xe0\x00\x10JF'

# f.close()

## 6,写入

# f = open('/tmp/passwd1','w') #以写的方式打开

# a = f.write('hello world!\n') #没有则创建.有的话就清空再写入,

# b = f.write('hello world!\n')

# # c = f.writelines('hello world!\n') #???????????

# print(a,b,)

#[root@room9pc01 tmp]# cat passwd1 #查看实际文件内容

#hello world!

#hello world!.

# f = open('/tmp/passwd1','w')

# a = f.writelines(['2nd line.', '3rd line.\n'])

# print(a)

# f.close() #关闭,也写入数据

###[root@room9pc01 tmp]# cat passwd1 #查看实际文件内容

###2nd line.3rd line.

## 7.with子句:语句结束,文件自动关闭

# with open('/tmp/passwd1') as f:

# print( f.readline() ) #显示文件的内容:2nd line.3rd line.

#

# # print(f.readline()) #继续执行.报错,file已closed

## 8.文件指针###以后用不着(了解)

# f = open('/tmp/passwd','rb') #以可读与二进制方式打开

# print(f.tell()) #输出0,

# f.read(5) #读取5个字节后

# print(f.tell()) #查看输出5,指针偏移5个字节

#####

###seek函数用与移动文件指针,有两个参数.第二个参数表示相对位置.

###第二个参数(0表示开头,1表示当前位置.2表示结尾),第一个参数是相对于第二个参数的偏移量

# f = open('/tmp/passwd1','rb')

# print(f.tell()) #当前指针位置0

# print(f.read(5)) #输出b'2nd l' #文件原文2nd line.3rd line.

# print(f.seek(3,1)) #输出8,,1表示当前位置.3表示偏移5个字节

# print(f.readline()) #输出b'.3rd line.\n' #文件原文2nd line.3rd line.

...

## @@cp功能命令

src\_fname = '/bin/ls' #定义源文件

dst\_fname = '/tmp/ls' #定义目标文件

src\_fobj = open(src\_fname,'rb') #以rb(读和二进制)方式打开，源文件

dst\_fobj = open(dst\_fname, 'wb') #=以写和二进制方式打开目标文件，没有则创建

while 'true': #无限制读取

data = src\_fobj.read(4096) #因为读取数据是放入内存,所以每次读取只读取4字节

# if data == b'': #判断方式1,data数据等于空,则表示数据取完,执行break

# if len(data) == 0: #判断方式2,统计data变量的数据长度,等于0,表示数据取完,执行break

if not data: #判断方式3,如果data变量是空字符串,则False.

break #终止所有循环

dst\_fobj.write(data) #将data变量值写入dst\_fobj,参考上面写入练习

src\_fobj.close()

dst\_fobj.close()

###验证,电脑上操作如下############

[root@room9pc01 tmp]# md5sum ls #哈希值验证两个文件是否一致

918cb545b3458e1bf18b712b36af304f ls

[root@room9pc01 tmp]# md5sum /bin/ls

918cb545b3458e1bf18b712b36af304f /bin/ls

## @@cp模块（函数方式）

import sys

def copy(src\_fname, dst\_fname): #定义函数,并定义两个参数

src\_fobj = open(src\_fname,'rb') #打开源文件,以rb方式

dst\_fobj = open(dst\_fname, 'wb')

while 'true': #无限制读取

data = src\_fobj.read(4096) #因为读取数据是放入内存,所以每次读取只读取4字节

if not data: #判断方式3,如果data变量是空字符串,则False.

break #终止所有循环

dst\_fobj.write(data) #将data变量值写入dst\_fobj,参考上面写入练习

src\_fobj.close()

dst\_fobj.close()

copy(sys.argv[1], sys.argv[2]) # argv是sys模块中的数列，将位置参数存到该列表中

##执行以及验证结果

[root@room9pc01 day02]# ls

day02.py day03.py home-day02.py test.py

[root@room9pc01 day02]# python3 day02.py home-day02.py test2.py

[root@room9pc01 day02]# ls #将home-day02.py复制到test2.py

day02.py day03.py home-day02.py test2.py test.py

# 函数

## 调用、默认值

####斐波那契数列

def gen\_fib(n=10): #定义名为gen\_fib函数，默认参数n=10

#number = int(input('请输入要计算的斐波那契数列范围:'))在这不再手动输入,在函数gen\_fib(n)中调用n参数

alist = [0,1] #定义最初始的列表,占用两个了

for i in range(n-2): #这里接受参数n.输入了10 ,实际运行要减去2个,

alist.append(alist[-1] + alist[-2]) #将列表中最后一个数加上倒数第二个的和,添加到列表中

# print(alist) #输入5 得到结果[0, 1, 1, 2, 3] 函数中不用这个输出,否者固定在函数中输出

return alist #将最后得到的alist 列表结果返回给函数gen\_fib(),函数结果需要return返回

mylist = gen\_fib() #调用函数执行,未传参,将会以默认参数n=10执行

print(mylist)

## 函数的位置参数

###cp功能模块

import sys

def copy(src\_fname, dst\_fname): #定义函数,并定义两个参数

src\_fobj = open(src\_fname,'rb') #打开源文件,以rb方式

dst\_fobj = open(dst\_fname, 'wb')

while 'true': #无限制读取

data = src\_fobj.read(4096) #因为读取数据是放入内存,所以每次读取只读取4字节

if not data: #判断方式3,如果data变量是空字符串,则False.

break #终止所有循环

dst\_fobj.write(data) #将data变量值写入dst\_fobj,参考上面写入练习

src\_fobj.close()

dst\_fobj.close()

copy(sys.argv[1], sys.argv[2]) # argv是sys模块中的数列，将位置参数存到该列表中

# import导入模块

1文件:是python从物理上组织代码的形式,

2模块:是python从逻辑上组织代码的形式,

3 以.py 我为结尾的python文件都是一个模块

4 模块文件名字去掉后面的扩展名（.py）即为模块

## import 导入模块1

hi = 'hello zdd'

def pstar(n=30):

print('$' \* n)

###实际执行如下#########

[root@room9pc01 tmp]# cd /root/PycharmProjects/day02/

[root@room9pc01 day02]# python3

>>> import day02 #调用名为day02的python文件全名为:day02.py

>>> day02.hi #调用day02文件中的hi变量

'heelo zdd'

>>> day02.pstar() #调用day02中的pstar函数

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

## import 导入模块2

#使用import导入就会执行一遍语句

hi = 'heelo zdd'

def pstar(n=30):

print('$' \* n)

pstar()

pstar(40)

[root@room9pc01 day02]# python3 day02.py #直接执行

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

[root@room9pc01 day02]# python3

Python 3.6.1 (default, Apr 27 2018, 04:53:30)

[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import day02 #调用这个模块也会执行一遍

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

>>>

## 模块导入的特性 \_\_name\_\_

当模块文件直接运行时,\_name\_的值是:'\_main\_'

当模块文件被导入时,\_name\_的值是:该模块的名字

#利用这个特性可控制模块的执行方式

hi = 'heelo zdd'

def pstar(n=30):

print('$' \* n)

if \_\_name\_\_ == '\_\_main\_\_': #判断\_\_name\_\_状态，如果是直接运行python文件就执行下面调用

pstar()

pstar(40)

[root@room9pc01 day02]# python3 day02.py

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

hi = 'heelo zdd'

def pstar(n=30):

print('$' \* n)

if \_\_name\_\_ == 'day02': #判断\_\_name\_\_状态，如果是被导入python文件就执行下面调用

pstar()

pstar(40)

##[root@room9pc01 day02]# python3

##>>> import day02 #如果是import调用才执行

##$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

##$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

## 随机密码

所有字母大小写+所有数字

import string # 导入string这个模块

print(string.digits) # 输出包含数字0~9的字符串

print(string.ascii\_letters) # 包含所有字母(大写或小写)的字符串

print(string.ascii\_lowercase) # 包含所有小写字母的字符串

print(string.ascii\_uppercase) # 包含所有大写字母的字符串

print(string.punctuation) # 所有特殊字符

### @@随机密码用户自己定义密码长度（自己写）

import string,random

all\_dx = string.ascii\_letters

all\_sx = string.digits

all = all\_dx + all\_sx

def suiji(n=8):

mm = ''

for i in range(n):

sj\_1 = random.choice(all)

mm += sj\_1

return mm

mm\_cd = int(input('请输入密码长度: '))

a = suiji(mm\_cd)

print(a)

### @@随机密码

from random import choice #调用random中的choice

from string import ascii\_letters,digits #/usr/local/lib/python3.6/string.py

all\_chs = ascii\_letters + digits #from方式调用， string可省略（string. ascii\_letters）

def gen\_pass(n=8):

result = ''

for i in range(n):

ch = choice(all\_chs)

result += ch

return (result)

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

print(gen\_pass())

print(gen\_pass(4))

# shutil模块

## shutil:shell工具

### 1,shutil.copyfileobj()

import shutil

file\_a = open('/etc/issue','rb')

file\_b = open('/tmp/issue1','wb')

shutil.copyfileobj(file\_a,file\_b)

file\_a.close()

file\_b.close()

####实际结果

[root@room9pc01 tmp]# cat /etc/issue

\S

Kernel \r on an \m

[root@room9pc01 tmp]# cat /tmp/issue1

\S

Kernel \r on an \m

### 2,shutil.copy

import shutil

shutil.copy('/etc/issue.net','/tmp') #拷贝文件, （常用，背下来）

[root@room9pc01 tmp]# ls /tmp/issue\*

/tmp/issue1 /tmp/issue2

### 3,shutil.copy2()

#相当于cp -p

### 4,shutil.move()

import shutil

shutil.move('/tmp/issue.net','/hahahahah') # -mv文件（常用，背下来）

### 5,shutil.remtree()

#相当于rm -rf 但是只能删除目录,不能删除文件.（常用，背下来）

import shutil

shutil.copytree('/etc/security', '/tmp/anquan') #拷贝目录用作测试

shutil.rmtree('/tmp/anquan') #删除目录

# 语法风格

## 变量复制,多重赋值

x = y = 10

print(x)

print(y) #输出：分两行输出两个10

a , b = 10 ,20

print(a , b) #输出：10 20

a, b = b, a # 互换a和b的值

print( a,b) #输出：20 10

aa , bb = (10,20)

print(aa,bb) #输出：10 20

aaa,bbb = [10,20]

print(aaa,bbb) #输出：10 20

## 合法标识符

合法标识符满足如下条件：

* 不能是关键字
* 第一个字符不能是数字，必须是字母或者下划线，剩下的可以是字母，数字，下划线
* 大小写敏感
* Python标识符字符串规则和其他大部分用C语言编写的高级语言相似

## 关键字

* 关键字应该保持稳定，但是pythong语言是一门不断成长和进化的语言，关键字偶尔会更新
* 关键字被保留，不能作为函数名，变量名等
* 加载关键字列表和iskeyword()函数都放入了keyword模块，以便于查阅

import keyword

print(keyword.kwlist) #输出所有关键字

print(keyword.iskeyword('pass')) #检测是否是关键字 输出:True,说明pass是关键字

print(keyword.iskeyword('adp')) #输出:False,说明adp不是关键字

## 内建

* 除了关键字之外，python还有可以在任何一级代码使用的‘内建’的名字集合，
* 这些名字由解释器设置或使用
* 虽然built-in不是关键字，但是应该把它当做“系统保留字”
* 保留的常量如：True、False 、None等
* 内建不应被当做变量来被覆盖

# 模块布局

## 模块布局模板

#!/usr/bin/env python #起始行

“this is a test module” #模块文档字符串

import sys #导入模块

import os

debug = True #全局变量声明

class FooClass(object): #类定义

'Foo class'

pass

def test(): #函数定义

"test funcDon"

foo = FooClass()

if \_\_name\_\_ == ‘\_\_main\_\_’: #程序主体

test()

## 编程思路：

1、弄清要求

2、发呆：思考程序是怎么运行的（交互的、非交互的？），如果是交互的脑补程序运行有什么

3、思考程序有哪些功能，把这些功能写成函数

4、编写主程序，调用各个功能函数

5、最后编写每个功能函数

## @@创建文件程序

#输入一个文件名，判断文件是否存在，存在则提示已存在，不存在则创建该文件，并可输入内容保存

! /usr/bin/env python

"this is a test module"

import sys,os #调用模块

def get\_fname(): #函数,y

while True: #循环

fname = input('请输入文件名: ') #此处的fname只在本循环中存在

if not os.path.exists(fname): #判断文件名是否已存在,如果不存在则执行break

break

print('该文件已存在,请重试!')

return fname #将fname的结果返回给get\_fname

def get\_content():

content = []

print('请输入内容,另起一行输入"end"结束!')

while True:

line = input('> ')

if line == 'end': #判断输入的如果是end 则执行break,结束所有循环

break

content.append(line) #判断输入的如果不是end 则将line的内容添加到content列表中

return content

def wfile(fname, content):

with open(fname,'w') as fobj: #以写方式打开文件fname，如果文件不存在则创建，with语句结束自动关闭文件

fobj.writelines(content) #将congtent收集的输入所有内容，写入到fname

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

fname = get\_fname() #fname得到get\_fname():中产生的不存在同名的文件名

content = get\_content() #content得到get\_content():中的产生的列表内容

content = [line + '\n' for line in content] #将content列表所有value添加一个\n

wfile(fname,content) #将content的内容写入到名为fname的文件中,没有则创建

# 字符串

## 字符串操作符

* 比较操作符：字符串大小按ASCII码值大小进行比较
* 切片操作符：[] 、 [:] 、 [ : : ]
* 成员关系操作符：in not in



## enumerate新列表

将列表重新组成含标识的列表

alist = ['tom','bob','jerry','alice']

for i\_1 in [0,1,2,3]:

print('%s : %s' % (i\_1,alist[i\_1]))

for i\_2 in range(4):

print('%s : %s' % (i\_2, alist[i\_2]))

for i\_3 in range(len(alist)):

print('%s : %s' % (i\_3, alist[i\_3]))

print(list(enumerate(alist))) #输出[(0, 'tom'), (1, 'bob'), (2, 'jerry'), (3, 'alice')]#enumerate将列表重新组成含标识的列表

for i\_4 in enumerate(alist):

print('%s :　%s' % (i\_4)) #必需两个%s

for i\_5, i\_6 in enumerate(alist):

print('%s:%s' % (i\_5,i\_6)) #与上一样，这种分两个变量更灵活，（常用）

####以上所有输出都一样如下

0:tom

1:bob

2:jerry

3:alice

## 列表翻转

alist = ['tom','bob','jerry','alice']

print(reversed(alist)) #输出一个文件:<list\_reverseiterator object at 0x7f2e1c9274e0>

print(list(reversed(alist))) #翻转，但是列表本身不变,输出['alice', 'jerry', 'bob', 'tom']

print(alist.reverse()) #翻转，改变列表本身

print(sorted(alist)) #排序，但是列表本身不变，输出['alice', 'bob', 'jerry', 'tom']

print(list(sorted(alist) #排序，但是列表本身不变，输出['alice', 'bob', 'jerry', 'tom']

print(alist.sort()) #排序，改变列表本身

## @@判断输入的标识符是否合法程序

import sys

import keyword,string

first\_chs = string.ascii\_letters + '\_' # first\_chs为所有大小写+下划线的集合

other\_chs = first\_chs + string.digits # first\_chs为特殊字符+下划线+数字的集合

def check\_idt(idt):

if keyword.iskeyword(idt): #判断是否是关键字，如果是则执行下面return

return '% 是关键字' % idt #一个函数只能return一次，return后函数结束，

if not idt[0] in first\_chs: #判断第一个字符如果不是大小写或者下划线，则执行return

return '第一个字符不合法'

for ind,ch in enumerate(idt[1:]): #把输入的字符串，从第2个开始重新列表，

if ch not in other\_chs: #判断上面获取的cd如果不是大小写或者下划线，则执行return

return '第%s 个字符不合法' % (ind + 2) #因为是从第2个开始，所以新标识跟实际相差2个

return '%s 是合法的标识符' % idt

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

print(check\_idt(sys.argv[1]))

####执行之后的结果##############

# [root@room9pc01 day02]# python3 day02.py asdf

# asdf 是合法的标识符

# [root@room9pc01 day02]# python3 day02.py 123sdfa

# 第一个字符不合法

# [root@room9pc01 day02]# python3 day02.py asdf@sdf

# 第5 个字符不合法

# 字符串格式化

## 1,基础样式

print('%s' % 'bob')

## 2括号可省略

print('%s' % 'bob') #当只有一个%s 占位符，后面的（）可省略

## 3字符串





print('%s is %s years old' % ('bob',20)) #bob is 20 years old

print('%s is %d years old' % ('bob',20)) #%d 只能做数字

print('%10s%8s' % ('name','age')) #name左对齐且占10个宽度,age左对齐且占8个宽度

print('%10s%8s' % ('zdd',20)) #

print('%-10s%-8s' % ('zdd',20)) #加-号，zdd右对齐且占10个宽度,20右对齐且占8个宽度

## 4了解（进制，浮点）

print('%#o' % 10) #8进制

print('%#x' % 10) #16进制

print('%f' % (5 / 3)) #f为浮点数

print('%5.2f' % (5 / 3))

print('%+d' % 10) #正数前加+号

print('%+d' % -10) #负数不需要加,就是-号

## 5 format函数，位置参数

print('{} is {} years old'.format('bob',20)) #输出bob is 20 years old

print('{0} is {1} years old'.format(20,'bob')) #输出20 is bob years old

print('{1} is {0} years old'.format(20,'bob')) #输出bob is 20 years old #位置1的bob左对齐点10个宽度

print( '{1:<10}:{0:>8}'.format(20, 'bob')) # 位置1的bob左对齐点10个宽度，位置0的20右对齐8个宽度

## 6原始字符串

# win\_path = 'c:\temp'

# print(win\_path) #输出c: emp，\t被当做特殊字符

#

# win\_path = 'c:\\temp'

# print(win\_path) #输出c:\temp 需要多个\来转义

#

# wpath = r'c:\temp' #原始字符串,字符串中的字符是本身含义 #字符串使用次数特别多（背下来）

# print(wpath) #输出c:\temp 直接原样输出

## 7字符串常用方法：

### 1、去除空白字符

s1 =' hello world\n'

print(s1)

print(s1.strip()) #去除两边的空白字符，包括换行,tab,\t

print(s1.lstrip()) # 去除左边空白字符

print(s1.rstrip()) #去除右边空白字符，包括换行,tab,\t

### 2、切割字符

s2 = 'hello world ni hao'

print(s2) #输出hello world ni hao

print(s2.split()) #输出['hello', 'world', 'ni', 'hao']

s3 = 'hello-world-ni-hao'

print(s3.split()) #输出['hello-world-ni-hao']

print(s3.split('-')) #输出['hello', 'world', 'ni', 'hao'] 指定分隔符‘-’

### 3、拼接字符

slist = ['hello','world','in','hao']

print(slist) #输出['hello', 'world', 'in', 'hao']

print(''.join(slist)) #输出helloworldinhao

print(' '.join(slist)) #输出hello world in hao

print('--'.join(slist)) #hello--world--in--hao

### 4、对齐

s2 = 'hello world ni hao'

print(s2.center(50)) #输出 hello world ni hao

print(s2.center(50),'-') #输出 hello world ni hao -

print(s2.center(50,'-')) #输出 ----------------hello world ni hao----------------

print(s2.rjust(50,'-')) #输出 --------------------------------hello world ni hao

print(s2.ljust(50,'-')) #输出 hello world ni hao--------------------------------

## 5、判断开头、结尾

s2 = 'hello world ni hao'

print(s2.startswith('h')) #输出 True

print(s2.startswith('he')) #输出 True

print(s2.startswith('hea')) #输出 False

print(s2.endswith('o')) #输出 True

print(s2.endswith('ao')) #输出 True

print(s2.endswith('sao')) #输出 False

# subprocess模块调用系统命令

python可以使用subprocess模块调用系统命令

>>> import subprocess

>>> rc = subprocess.run('id root', shell=True) #调用系统id命令查询root用户是否存在

>>> rc.returncode # retruncode就是shell中的$?，返回查询的结果

## @@创建用户程序

import sys

import subprocess

from randpass2 import gen\_pass #gen\_pass表示密码输入字符不显示，用于密码

def adduser(user, passwd, fname): #函数获取三个参数

info = '''用户信息

用户名:%s

密码:%s

''' % (user,passwd) #将按照此格式，将用户信息写入/tmp/users.txt文件中

subprocess.run('useradd %s' % user, shell=True) #调用系统命令创建用户

subprocess.run(

'echo %s | passwd --stdin %s' % (passwd,user), #调用系统命令给用户创建密码

shell=True

)

with open(fname,'a') as fobj: #用with方式，以a(追加)形式打开文件，

fobj.write(info) #最后将用户信息写入fname指定的文件中，with自动关闭文件

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

username = sys.argv[1] #username从位置函数获取

pwd = gen\_pass() #获取密码，输入时，隐藏不显示

filename = '/tmp/users.txt' #创建成功，将会把用户名和密码写入/tmp/users.txt文件中

adduser(username,pwd,filename) #执行函数，并传入username,pwd，filename三个参数

###20190415#####################

# 列表(list):下标

属于容器类型

# from random import randint

# alist1 = [randint(1,100) for i in range(10)]

# print(alist1) #[21, 88, 62, 5, 91, 85, 40, 45, 36, 7]

#

# alist = [21, 88, 62, 5, 91, 85,]

# print(alist.append(30)) #None

# print(alist) #[21, 88, 62, 5, 91, 85, 30]

# print(alist.count(21)) #输出1,统计21在列表中出现的次数

# print(alist.index(21)) #输出0 统计21在列表中的下标

#

# alist.insert(5,101) #在列表中下标为5的位置插入101

# print(alist) #[21, 88, 62, 5, 91, 101, 85, 30]

#

# alist.reverse()

# print(alist) #[30, 85, 101, 91, 5, 62, 88, 21]

#

# alist.sort() #列表排序

# print(alist) #[5, 21, 30, 62, 85, 88, 91, 101]

#

# alist.remove(30) #删除列表

# print(alist) #[5, 21, 62, 85, 88, 91, 101]

#

# alist.pop() #踢出元素,默认是踢出最后一个

# print(alist) #[5, 21, 62, 85, 88, 91]

# alist.pop(2) #踢出下标为2的元素

# print(alist) #[5, 21, 85, 88, 91]

#

# blist = alist.copy() #将alist拷贝给blist

# print(blist) #[5, 21, 85, 88, 91]

# clist = alist #clist 与 alist 使用相同的地址符

# print(clist) #[5, 21, 85, 88, 91]

#

# alist.extend(clist) #将clist的内容汇入到alist

# print(alist) #[5, 21, 85, 88, 91, 5, 21, 85, 88, 91]

# alist.clear() #清空列表

# print(alist) #[]

ages1 = (11,22,33,44,55) #定义元组

print(list(ages1)) #list将ages1元组转换列表 [11, 22, 33, 44, 55]

# 元组(tuple):下标,不可变

不可变

a = (10)

print(type(a)) #<class 'int'> 说明a是个int整数

print(a) #10

b = (10,)

print(type(b)) #<class 'tuple'> 说明b是个元组

print(b) #(10,)

ages1 = (11,22,33,44,55) #直接定义  
print(list(ages1)) #list将ages1元组转换列表  
ages = tuple(list(ages1)) #将列表转换成元组  
print(ages)  
  
ages2 = tuple(**'abcd'**) #将字符串转换成元组  
print(ages2)

# 字典(dict):无下标

## 1,工厂dict方法创建字典

adict = dict(['ab',['name','tom'],('age',22)])  
print(adict) #{'a': 'b', 'name': 'tom', 'age': 22}  
#dict的参数是个序列对象,序列中有三项,每一项又有两个项目,第一个项目是key,第二个是value

## 1创建具有相同值的默认字典bdict = {}.fromkeys(['tom','jerry','bob'],7) print(bdict) #{'tom': 7, 'jerry': 7, 'bob': 7}

## 2,访问字典

字典是映射类型,意味着没有下标.访问字典中的值需要使用相应的键

adict = dict(['ab',['name','tom'],('age',22)])

print(adict) for key in adict:

print('$s : $s' % (key, adict[key]))

#显示如下

{'a': 'b', 'name': 'tom', 'age': 22}

a : b

name : tom

age : 22

## 判断

使用 in 和 not in 判断键是否存在于字典中

print('tom' in adict) #False  
print('name' in adict) #True

## 3更新添加字典

字典的kye都是唯一的,不重复的,字典的key必须是不可变类型

adict = dict(['ab',['name','tom'],('age',22)])

adict['name'] = 'jerry' #有name key则更新tom为jerry

print(adict) #{'a': 'b', 'name': 'jerry', 'age': 22}

adict['zdd'] = '123456' #没有zdd key 则添加新的

print(adict) #添加字典{'a': 'b', 'name': 'jerry', 'age': 22, 'zdd': '123456'}

## 4字典-get-检验key\*\*\*

adict = dict(['ab',['name','tom'],('age',22)]) #创建字典

print(adict.get('name', 'not found')) #字典中有name的key,返回tom,没有则返回’not found’

a = adict.get('qq') #字典中没有key为qq项,返回值None给a

print(a) #None

adict.get('qq','not found') #字典中没有key为qq项,返回值None

print(adict.get('qq','not found')) #not found

## 5字典-keys-所有key

adict = dict(['ab',['name','tom'],('age',22)])

b = adict.keys() #返回adict字典中所有的key

print(b) #dict\_keys(['a', 'name', 'age'])

print(adict.keys()) #dict\_keys(['a', 'name', 'age'])

## 6字典-values-返回所有值

adict = dict(['ab',['name','tom'],('age',22)])

print(adict) #{'a': 'b', 'name': 'tom', 'age': 22}

print(adict.values()) #dict\_values(['b', 'tom', 22]) 返回所有的value

## 7字典-items-返回所有

adict = dict(['ab',['name','tom'],('age',22)])

print(adict) #{'a': 'b', 'name': 'tom', 'age': 22}

print(adict.items()) #dict\_items([('a', 'b'), ('name', 'tom'), ('age', 22)])返回所有的(key,value)组成的集合

## 8字典-pop-弹出

adict = dict(['ab',['name','tom'],('age',22)])

print(adict) #{'a': 'b', 'name': 'tom', 'age': 22}

print(adict.pop('a')) #b #弹出key是a的项

print(adict.pop()) #默认弹出最后一个

## 9 字典函数

len():返回字典中元素的数目

hash()本省不是为字典设计的,但是可以判断某个对象是否可以作为字典的键

## 10.字典内建方法---待补充

## @@@栈程序:方式一

#栈结构,类似于堆叠盘子,先放上去的堆在下方,先拿的都是最后放上去的

import random,sys

list\_a = []

def push\_it(): #进栈

print('压栈') #调试测试

itm = input('>').strip() #定义输入

if itm: #如果itm为非空则执行下面

list\_a.append(itm)

def pop\_it(): #出栈

print('出栈')

if list\_a: #如过list\_a列表为非空,

print('栈中弹出:%s' % list\_a.pop()) #输出的同时,列表中弹出最后一项

else:

print('空栈,无内容弹出')

def view\_it(): #查询

print('\033[31;1m%s\033[0m' % list\_a)

def show\_menu(): #选择菜单

prompt = '''0:压栈

1:出栈

2:查询

3:退出

请选择相应的操作(0/1/2/3):'''

while True:

xz\_1 = (input(prompt).strip()[0])

if xz\_1 not in ['0','1','2','3']:

print('无效,请重新输入')

continue

elif xz\_1=='0':

push\_it()

elif xz\_1=='1':

pop\_it()

elif xz\_1=='2':

view\_it()

else:

print('退出')

break

if \_\_name\_\_ == '\_\_main\_\_': #主程序

show\_menu()

## @@@栈程序,字典方式

栈结构,类似于堆叠盘子,先放上去的堆在下方,先拿的都是最后放上去的

import random,sys

list\_a = []

def push\_it(): #进栈

print('压栈') #调试测试

itm = input('>').strip() #定义输入

if itm: #如果itm为非空则执行下面

list\_a.append(itm)

def pop\_it(): #出栈

print('出栈')

if list\_a: #如果list\_a列表为非空,则执行下面

print('栈中弹出:%s' % list\_a.pop()) #输出的同时,列表中弹出最后一项

else:

print('空栈,无内容弹出')

def view\_it(): #查询

print('\033[31;1m%s\033[0m' % list\_a) #输出查询结果,并标红

def show\_menu(): #选择菜单

cmds = {'0':push\_it,'1':pop\_it,'2':view\_it} #定义字典

#采用字典方式:注意字典中函数名后不要加(),因为是把函数存入字典,

#并不是将函数结果存入字典,

prompt = '''0:压栈

1:出栈

2:查询

3:退出

请选择相应的操作(0/1/2/3):'''

while True:

# xz\_1 = (input(prompt).strip()[0]) #删除用户输入的两端空格,并取出第一个

# if xz\_1 not in ['0','1','2','3']:

# print('无效,请重新输入')

# continue

# elif xz\_1=='0':

# push\_it()

# elif xz\_1=='1':

# pop\_it()

# elif xz\_1=='2':

# view\_it()

# else:

# print('退出')

# break

choice = input(prompt).strip()[0] #删除用户输入的两端空格,并取出第一个

if choice not in ['0','1','2','3']:

print('无效,请重试:')

continue

if choice == '3':

print('退出')

break

cmds[choice]() #在字典中取出出入的数字对应的函数,加上后面的函数,即为调用函数

if \_\_name\_\_ == '\_\_main\_\_': #主程序

show\_menu()

## @@@模拟用户登录程序

#####模拟用用户登录系统

import getpass

userdb = {}

print(userdb)

def new():

print('正在注册')

user = input('请输入新用户名').strip()

if user and user not in userdb:

passwd = input('密码:')

userdb[user] = passwd

def old():

print('正在登录')

user = input('请输入用户名:').strip()

passwd = getpass.getpass('密码:') #定义输入的密码,且是不明文显示

if userdb.get(user) == passwd:

#如果输入的用户不在userdb字典中,返回值将是None不会等于刚刚输入的密码

#或者用户存在,返回的value值不等于,刚刚输入的passwd,都不执行下面

print('登录成功')

else:

print('登录失败')

def menu():

cmds = {'0':new,'1':old}

xz='''0:新用户注册

1:老用户登录

2:退出

请选择(0/1):'''

while True:

choice = input(xz).strip()

if choice not in ['0','1','2']:

print('请重新选择')

continue

if choice == '2':

print('退出')

break

cmds [choice]()

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

menu()

# linux与windows文件的转换

* windows文本文件的行结束标志是\r\n
* 类unix文本本间的行结束标志是\n

## @@@uninx2dos的转换程序

###uninx2dos的转换程序

import sys

def unix2dos(fname): #接收位置参数

dst\_fname = fname + '.txt' #重命名

with open(fname) as src\_fobj: #打开源文件

with open(dst\_fname,'w') as dst\_fobj: #以w方式打开目标文件,没有则创建

for line in src\_fobj: #遍历源文件

line = line.rstrip() + '\t\n' #重新获取line内容,去掉右边的空格,加上\t\n

dst\_fobj.write(line) #将line内容重新写入目标文件

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

unix2dos(sys.argv[1]) #位置参数

#转换前实际验证

# [root@room9pc01 python]# python3

# >>> f = open('/root/桌面/python/gen\_pass.py.txt')

# >>> f.readline()

# '#! /usr/bin/env python\n' #转换之前是\n结尾

#转换后实际验证

# [root@room9pc01 python]# python3 dayall.py gen\_pass.py

# [root@room9pc01 python]# ls #转换后多了gen\_pass.py.txt 文件

# dayall.py gen\_pass.py gen\_pass.py.txt home-100.py Python-3.7.3.tgz

# [root@room9pc01 python]# python3

# >>> f = open('/root/桌面/python/gen\_pass.py.txt')

# >>> f.readline()

# '#! /usr/bin/env python\t\n' #转换后是\t\n结尾了

# 进度条

## @@@类似进度条程序

**import** time  
n = 0  
print(**'#'** \* 20, end=**''**)  
**while True**:  
 print(**'\r%s@@@%s'** % (**'#'** \* n , **'#'** \* (19 - n)),end=**''**)  
 n +=1  
 **if** n == 20:  
 n = 0  
 time.sleep(0.3)

# 集合(set):不可变

* 由不同元素构成,元素必须是不可变对象
* 常用来去重和关系运算
* 用{}表示,像是一个无值的字典,只有key

from random import randint

alist = [randint(1, 20) for i in range (10)]

print(alist) #十个1到20之间的随机数[13, 19, 12, 4, 20, 5, 13, 8, 16, 16]

aset = set(alist)

print(aset) #按序排列,并去重{4, 5, 8, 12, 13, 16, 19, 20}

aset = set('abc')

print(aset) #{'b', 'a', 'c'}

bset = set('bcd')

print(bset) #{'c', 'd', 'b'}

print(aset|bset) #并集{'b', 'c', 'a', 'd'}

print(aset&bset) #交集{'b', 'c'}

print(aset-bset) #差集{'a'} aset中有,bset中没有

print(aset.add('abc')) #None 不能这么显示

aset.add('abc')

print(aset) #先添加在打印{'a', 'c', 'b', 'abc'}.

## 清空集合

a={3,"a",2.1,1}

print(a.clear()) # None

## 集合批量添加,删除

aset = set('abc')

print(aset) #{'b', 'a', 'c'}

aset.update('xyz') #批量添加元素

print(aset) #{'a', 'z', 'b', 'c', 'x', 'y'}

aset.update(['xxx','yyy','zzz'])

print(aset) #{'z', 'x', 'c', 'b', 'a', 'y', 'xxx', 'zzz', 'yyy'}

aset.remove('zzz')

print(aset) #{'xxx', 'c', 'yyy', 'x', 'b', 'z', 'y', 'a'}

## @@@比较两个a,b两个文件,

with open('/tmp/passwd1') as f1:

s1 = set(f1) #将第一个文件内容变成s1集合

with open('/tmp/passwd2') as f2:

s2 = set(f2) #将第二个文件内容变成s2集合

s3 = s2 - s1 #s3等于s2相对于s1的差集

with open('/tmp/result.txt','w') as f3: #打开一个新文件到f3

f3.writelines(s3) #将s3列表写入到f3中

###实际操作

#passwd1文件内容

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

#passswd2文件内容

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

ni hao a

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

root:x:0:0:root:/root:/bin/bash

zdd

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

roooasdft:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

#程序运行后产生的新文件内容

[root@room9pc01 tmp]# cat result.txt

roooasdft:x:0:0:root:/root:/bin/bash

ni hao a

zdd

# time模块时间表示方式

## 1,时间戳,距离1970-1-1 0:00:00 之间的秒数

# import time

# print(time.time()) #1555318981.2148066

## 2,UTC时间:世界协调时

# print(time.ctime()) #Mon Apr 15 17:03:01 2019

## 3,struct\_time九元组

# print(time.localtime()) #localtime将一个时间戳转换为当前时区的struct\_time

# #time.struct\_time(tm\_year=2019, tm\_mon=4, tm\_mday=15, tm\_hour=17, tm\_min=3, tm\_sec=1, tm\_wday=0, tm\_yday=105, tm\_isdst=0)

# t = time.localtime()

# a=(t.tm\_year)

# print(a) #2019

# print(t.tm\_mon) #4

# ##需要掌握的时间表示方式

# print(time.time()) #时间戳

# print(time.strftime('%Y-%m-%d %H:%M:%S')) #显示固定格式2019-04-15 16:56:07

# print(time.strptime('2019-04-15 16:56:07','%Y-%m-%d %H:%M:%S'))

# #将时间字符串转换成struct\_time样式(九元组)

## strptime 和 strftime

# t1 = time.strptime('2019-04-15 16:56:07','%Y-%m-%d %H:%M:%S')

#将时间字符串转换成struct\_time样式(九元组)

# t2 = time.localtime() #时间比较

# print(t1 > t2)

# print(t2 > t1)

#

# print(time.strftime('%a %A')) #显示固定格式2019-04-15 16:56:07

## 时间睡眠

# time.sleep(3) #时间睡眠

## datetime

# import datetime

# t1 = datetime.datetime.now() #取出当前时间,年月日时分秒毫秒

# t2 = datetime.datetime(2019, 4, 10) #创建一个时间

# print(t1) #2019-04-15 17:13:02.111975

# print(t2) #2019-04-10 00:00:00

# print(t1 > t2) #True

#

# days = datetime.timedelta(days=100,hours=10)

# print(t1 - days) #100天零10十小时之前的时间2019-01-05 07:13:02.111975

# print(t1 + days) #100天零10小时之后的时间2019-07-25 03:13:02.111975

# 异常处理

# 错误代码

# try:

# num = int(input('number: '))

# result = 100 / num

# except (ValueError,ZeroDivisionError): #字母无法转换成整数.多个错误情况放一起必须加括号

# print('滚蛋,无效的数字1')

# # except ZeroDivisionError: #0无法作为被除数, 与上面可以一起

# # print('滚犊子,无效的数字2')

# except KeyboardInterrupt: #ctrl+c

# print('\n拜拜了您勒')

# exit() #程序遇到exit就会退出,后续代码不执行

# except EOFError: #ctrl+D

# print('\n会不会用?不会滚蛋')

# exit()

# else:

# print(result) #输出结果

# finally: #不管程序是否发生异常,都会执行

# print('Done')

######触发异常#####

def set\_age(name, age):

if not 0 < age < 120:

raise ValueError('年龄超出范围')

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

def set\_age2(name, age):

assert 0 < age < 120, '年龄超出范围' # 如果age不在此范围，一定发生AssertionError

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

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

set\_age2('杨晨', 24)

## 断言异常 \*\*待补充

断言，判断某个表达式的真假，真则程序继续执行，否则抛出 AssertionError 异常。

# OS模块

* import os #调用模块
* 对文件系统的访问大多通过os模块实现
* 该模块是python访问操作系统功能的主要接口
* 有些方法,os并没有提供,如copy等,可以使用shutil模块作为补充



>>> os.getcwd() #查看当前目录

'/gitall/zdd-repository/zddpython2-day02'

>>> os.mkdir (test) #创建目录

>>> os.mkdir('/tmp/demo') #

>>> os.makedirs('/tmp/aaa/bbb/ccc') #相当于mkedir -P

>>> os.chdir('/tmp/demo') #相当于 cd

>>> os.getcwd()

'/tmp/demo'

>>> os.listdir() #相当于ls

[]

>>> os.makedirs('aaa/bbb/ccc')

>>> os.listdir()

['aaa']

>>> os.mknod('mytest') #相当于touch--mytest

>>> os.listdir() #ls

['aaa', 'mytest']

>>> os.symlink('/etc/hosts','zhuji') #相当于ln -s 创建软连接

>>> os.listdir()

['zhuji', 'aaa', 'mytest']

>>> os.unlink('zhuji') #删除软连接

>>> os.remove('mytest') #相当于rm -f

>>> os.path.isfile('/etc/hosts') #判断是否是文件

True

>>> os.path.isdir('/etc/ho') #判断是否是目录

False

>>> os.path.islink('/etc/hosts') #判断是否是软连接

False

>>> os.path.exists('/etc') #判断文件是否存在

True

>>> os.path.split('.etc.sysconfig/network') #将目录名和文件名分开

('.etc.sysconfig', 'network')

>>> os.path.join('/etc/sysconfig', 'network') #将目录名和文件名合并

'/etc/sysconfig/network'

>>> os.path.getsize('/etc/passwd') #获取文件大小

2460

# pickle模块

* 把数据写入文件时,常规的文件方法只能把字符串对象写入,其他数据需要先转换成字符串再写入
* pickle模块,可以将任意内容存入文件,又可以把它完整无缺地取出来
* 之前的方法无法写入字典,列表,,,pickle就可以

pickle模块dump()存储 和 load()读取方法

>>> import pickle

>>> f = open('/tmp/myfile','wb')

>>> shop\_list = ['apple','egg']

>>> pickle.dump(shop\_list,f) #dump存储写入，将shop\_list列表写入f打开的文件目录中，

>>> f.close()

>>>

>>> with open('/tmp/myfile','rb') as fobj:

... mylist = pickle.load(fobj) #load读取

... #连续回车

>>>

>>> type(mylist)

<class 'list'>

>>>

>>> mylist #mylist列表用load读取的结果

['apple', 'egg']

>>>

[root@room9pc01 tmp]# cat myfile #myfile文件中有内容了,存入的是二进制

�XaqXbq�q.

## @@@记帐本程序

#! /usr/bin/env python

import os,pickle,time

def ruzhang(fname):

print('jz')

date = time.strftime('%Y-%m-%d')

rz\_je = int(input('入账金额：'))

rz\_sm = input('入账说明：')

with open(fname,'rb') as fobj:

data = pickle.load(fobj)

balance = data[-1][-2]+rz\_je

line = [date,rz\_je,0,balance,rz\_sm]

data.append(line)

with open(fname,'wb') as fobj:

pickle.dump(data,fobj)

def chuzhang(fname):

print('cz')

date = time.strftime('%Y-%m-%d')

cz\_je = int(input('出账金额：'))

cz\_sm = input('出账说明：')

with open(fname,'rb') as fobj:

data = pickle.load(fobj)

balance = data[-1][-2]-cz\_je

line = [date,0,cz\_je,balance,cz\_sm]

data.append(line)

with open(fname,'wb') as fobj:

pickle.dump(data,fobj)

def chaxun(fname):

print('cx')

print('%-10s%-8s%-8s%-10s%-20s' % ('登记时间','入账金额','出账金额','余额','说明'))

with open(fname,'rb') as fobj:

data = pickle.load(fobj)

for line in data:

print('%-14s%-13s%-11s%-11s%-20s' % tuple(line))

def view():

xz\_1 = '''0:入账

1:出账

2:查询

3:退出

请选择（0/1/2）：'''

cmds = {'0':ruzhang,'1':chuzhang,'2':chaxun}

fname = 'jzb.txt'

if not os.path.exists(fname):

date\_a = time.strftime('%Y-%m-%d')

data\_a = [

[date\_a,0,0,10000,'初始金额']

]

with open(fname,'wb') as fobj:

pickle.dump(data\_a,fobj)

while True:

xz\_2 = input(xz\_1).strip()

if xz\_2 == '3':

print('选择退出')

exit()

if xz\_2 not in ['0','1','2']:

print('无效输入')

continue

cmds[xz\_2](fname)

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

view()

# 函数

## 函数传参:

**def** myfunc(name, age=23):  
 **pass #pass 不做任何事情，一般用做占位语句。为了保持程序结构完整性***#函数中的name就被称作位置参数，age被称作关键字参数***def** get\_info(name, age):  
 print(**'%s is %s years old'** % (name, age))  
*#get\_info() # error，参数个数不足  
#get\_info('tom', 20, 30) # error, 参数个数太多*get\_info(**'tom'**, 20) *# OK #tom is 20 years old*get\_info(20, **'tom'**) *# OK，但是语义不对 20 is tom years old*get\_info(age=20, name=**'tom'**) *# OK #tom is 20 years old  
#get\_info(age=20, 'tom') # error, 位置参数必须在关键字参数前  
#get\_info(20, name='tom') # error, name得到了多个值*get\_info(**'tom'**, age=20) *# OK #tom is 20 years old*

## 函数参数组

通过把元组(非关键字参数)或者字典(关键字参数)作为参数组传递给函数

def func1(\*args): #args前面的一个\*号,表明args是元组,传参会把参数放入元组

print(args) #调用

def func2(\*\*kwargs): #kwargs前面两个\*\*号,表名kwargs是字典

print(kwargs) #调用

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

func1()

func1('hao')

func1('zdd',123)

func2()

func2(name='tom',age=20)

显示结果:

()

('hao',)

('zdd', 123)

{}

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

## 参数前面加\*号拆开

print('abc')

print(\*'abc')

##print(\*\*'abc') #错误

print(\*[10,20,30])

print(\*(10,20,30))

显示结果:

abc

a b c

10 20 30

10 20 30

## 字典前面加两个\*\*号再传参

表示把字典拆成关键字参数

**def** info(name,age):  
 print(**'%s:%s'** % (name,age))  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 info(\*\*{**'name'**:**'tom'**,**'age'**:20})

显示结果:

tom:20

info(name=**'tom'**,age=20)

显示结果:

tom:20

## @@@简单的加减法数学游戏--有错误搜集

*###简单的加减法数学游戏:增加了错误收集 try***import** random  
**def** add(x,y):  
 **return** x+y  
**def** sub(x,y):  
 **return** x-y  
  
**def** exam():  
 cmds = {**'+'**:add,**'-'**:sub} *#定义函数选择字典* nums = [random.randint(1,100) **for** i **in** range(2)] *#随机生成两个数* nums.sort(reverse=**True**) *#重新排列,默认升序,这是降序* op = random.choice(**'+-'**) *#随机选择是+法还是-法* result = cmds[op](\*nums) *#将列表拆开,传参到add函数或者sub函数,进行运算* prompt = **'%s %s %s = '** % (nums[0],op,nums[1]) *#固定格式,第一个数 +/- 第二个数 =* counter = 0 *#用作循环次数计数* **while** counter < 3:  
 **try**:  
 answer = int(input(prompt)) *#用户输入答案时捕获错误* **except**: *#捕获所有错误操作,不按程序答对,或者答错三次不结束,不推荐* print(**'非法操作,请继续'**)  
 **continue  
  
 if** answer == result: *#如果输入的等于运算返回的结果* print(**'答对了'**)  
 **break** print(**'不对哦'**) *#如果输入的不等于算返回的结果,则执行* counter +=1  
 **else**:  
 print(**'%s%s'** % (prompt,result)) *#输出上面的固定格式和结果***def** main():  
 **while True**:  
 print(**'开始测智商了'**)  
 exam() *#调用函数* **try**:  
 y\_n = input(**'请选择是否继续(y/n)?'**).strip()[0] *#输入是否继续时捕获错误* **except** IndexError:  
 **continue  
 except** (KeyboardInterrupt,EOFError): *#ctrl+c 与 ctrl+d 的错误* y\_n = **'n'  
  
 if** y\_n **in 'nN'**:  
 print(**'\n测试结束'**)  
 **break  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

## 匿名函数(lambda)

* lambda关键自创造匿名函数
* 匿名是因为不需要以标准的def方式来声明,简化语句
* 一个完整的lambda “语句”代表了一个表达式,这个表达式的定义体必须和声明放在同一行

## @@@简单的加减法--lambda匿名函数方式

lambda 参数1,参数2 :表达式　　　 lambda x,y:x+y

import random

def exam():

cmds = {'+':lambda x,y:x+y,'-':lambda x,y:x-y,'\*':lambda x,y:x\*y} #用匿名函数代替运算

nums = [random.randint(1,100) for i in range(2)] #随机生成两个数

nums.sort(reverse=True) #重新排列,默认升序,这是降序

op = random.choice('+-\*') #随机选择是+法还是-法

result = cmds[op](\*nums) #将列表拆开,传参到add函数或者sub函数,进行运算

prompt = '%s %s %s = ' % (nums[0],op,nums[1]) #固定格式,第一个数 +/- 第二个数 =

counter = 0 #用作循环次数计数

while counter < 3:

try:

answer = int(input(prompt)) #用户输入答案时捕获错误

except: #捕获所有错误操作,不按程序答对,或者答错三次不结束,不推荐

print('非法操作,请继续')

continue

if answer == result: #如果输入的等于运算返回的结果

print('答对了')

break

print('不对哦') #如果输入的不等于算返回的结果,则执行

counter +=1

else:

print('%s%s' % (prompt,result)) #输出上面的固定格式和结果

def main():

while True:

print('开始测智商了')

exam() #调用函数

try:

y\_n = input('请选择是否继续(y/n)?').strip()[0] #输入是否继续时捕获错误

except IndexError:

continue

except (KeyboardInterrupt,EOFError): #ctrl+c 与 ctrl+d 的错误

y\_n = 'n'

if y\_n in 'nN':

print('\n测试结束')

break

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

main()

## filter函数过滤(高阶函数)

调用一个布尔函数func来迭代遍历每个序列中的元素,返回一个使func返回值为trure的元素序列

from random import randint

def func1(x): #布尔函数,x%2要么是0 要么是1,

return x % 2 #x的值除以2取余,结果为0的是flase, 结果为1的是ture

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

nums = [randint(1,100) for i in range(10)] #定义10个100以内随机数列表,赋值给nums

print(nums)

result = filter(func1,nums) #将nums列表传参给程序

print(list(result))

result2 = filter(lambda x: x % 2,nums) #以lambda匿名函数方式,执行结果与上面一样

print(list(result2))

result3 = filter(lambda x: not x % 2,nums) #不在奇数列表中.求随机数中的偶数,

print(list(result3))

显示结果:

[51, 36, 78, 24, 77, 60, 71, 29, 21, 78]

[51, 77, 71, 29, 21] #

[51, 77, 71, 29, 21]

[36, 78, 24, 60, 78]

## map函数

from random import randint

def func1(x):

return x + 2

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

nums = [randint(1,100) for i in range(10)]

print(nums)

result = map(func1,nums) #将nums中的每个数字交给func1函数进行加工,返回结果

print(list(result))

result2 = map(lambda x: x + 2, nums) #将nums中每个数字交给lambda函数进行加工,返回结果

print(list(result2))

# #输出结果:

# [80, 1, 61, 44, 28, 37, 47, 39, 61, 83] #nums值

# [82, 3, 63, 46, 30, 39, 49, 41, 63, 85] #map给func1函数加工后

# [82, 3, 63, 46, 30, 39, 49, 41, 63, 85] #直接用lambda加工

# 高级函数应用

## 全局变量,局部变量

x = 10

def foo():

print(x)

foo()

def foo2():

x = 'hello' #局部变量也有变量x,它将会把全局变量x在此函数中遮盖住

print(x) #输出hello

foo2()

print(x) #输出10

def foo3():

global x #声明x为全局变量,将会覆盖之前的全局变量

x = 10000

print(x) #输出:10000

foo3()

print(x) #输出:10000

## 名字空间

i = 1

def f():

i = 2

def g():

print(i)

return g

func = f()

func()

print(i)

**分类**

* local，局部命名空间，每个函数所拥有的命名空间，记录了函数中定义的所有变量，包括函数的入参、内部定义的局部变量。
* global，全局命名空间，每个模块加载执行时创建的，记录了模块中定义的变量，包括模块中定义的函数、类、其他导入的模块、模块级的变量与常量。
* Built-in，python自带的内建命名空间，任何模块均可以访问，放着内置的函数和异常。

**搜索顺序依次是**:局部>全局>内建

**生命周期**

* local（局部命名空间）在函数被调用时才被创建，但函数返回结果或抛出异常时被删除。（每一个递归函数都拥有自己的命名空间）。
* global（全局命名空间）在模块被加载时创建，通常一直保留直到python解释器退出。
* Built-in（内建命名空间）在python解释器启动时创建，一直保留直到解释器退出。

## 偏函数(partial):

改造现有函数,生成新函数,

是将函数式编程的概念和默认参数以及可变参数结合在一起

一个带有多个参数的函数,如果其中某些参数基本上固定的,那么就可以通过偏函数为这些参数赋默认值

from functools import partial

def add(a,b,c,d,e):

return a + b + c + d + e

a = add(10,20,30,40,5) #执行add函数,结果赋值给a

print(a)

myadd = partial(add,10,20,30,40) #将前面四个参数给固定为默认值,将函数add生成新函数myadd

b = myadd(5) #执行 新函数myadd并新传参5,结果赋值给b,

print(b)

## @@@简单GUI程序,,简单测试

import tkinter

from functools import partial

window = tkinter.Tk()

lb = tkinter.Label(window, text="Hello World!", font="Arial 20")

# b1 = tkinter.Button(window, fg='white', bg='blue', text='Button 1')

MyButton = partial(tkinter.Button, window, fg='white', bg='blue')

b1 = MyButton(text='Button 1')

b2 = MyButton(text='Button 1')

b3 = MyButton(text='Button 1')

qb = MyButton(text='QUIT', command=window.quit)

lb.pack()

b1.pack()

b2.pack()

qb.pack()

# 递归函数

如果函数包含了对其自身的调用,该函数就是递归的.

## 递归算法:

5!=5x4x3x2x1

5!=5x4!

5!=5x4x3!

5!=5x4x3x2!

5!=5x4x3x2x1!

## @@@递归函数案例

def func1(x): #定义

if x == 1:

return 1

return x \* func1(x - 1)

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

print(func1(5))

## @@@快速排序函数

from random import randint

def paixu(item):

if len(item) < 2:

return item

px\_zj = item[0]

px\_zx = [] #比列表第一个数小的放到这个集合

px\_zd = [] #比列表第一个数打的放到这个集合

for a in item[1:]: #从列表第二个数开始遍历

if a <= px\_zj:

px\_zx.append(a) #把比第一个数小的添加到px\_zx列表

else:

px\_zd.append(a) #把比第一个数大的添加到px\_zd列表

return paixu(px\_zx) + [px\_zj] + paixu(px\_zd)

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

nums = [randint(1,100) for i in range(10)]

print(nums)

print(paixu(nums)) #调用函数,并将nums列表传参给item

# 生成器-yield

* 生成器本身是一个带yield语句函数
* 与迭代器相似,生成器以另外的方式来运作
* 一个函数或者子程序,只返回一次,当达到一个真正的返回或者函数结束没有更多的值返回,会报StopIteration异常

def my\_gen():

yield 'hello world'

num = 10 + 5

yield num

yield 100

yield 1

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

my\_gen()

a = my\_gen()

print(a.\_\_next\_\_())

print(a.\_\_next\_\_())

print(a.\_\_next\_\_())

print(a.\_\_next\_\_())

# print(a.\_\_next\_\_()) #报错,只定义了4个生成器

print() #空格

mg = my\_gen()

for i in mg: #实际使用中使用for循环

print(i)

#输出结果

hello world

15

100

1

hello world

15

100

1

# 内部函数

## 闭包

一个函数套着另外的函数

def deco(func):

def red():

return '\033[31;1m%s\033[0m' % func()

return red

def hello():

return 'hello world!'

def greet():

return '你好'

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

print(hello()) #普通调用

a = deco(hello) #将hello函数作为参数传递给deco ,deco的返回值是red函数

print(a()) #调用 a 函数,实际上是调用red函数

## 装饰器

def deco(func):

def red():

return '\033[31;1m%s\033[0m' % func()

return red

def hello():

return 'hello world!'

@deco #装饰器

def greet():

return '你好'

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

# print(hello()) #普通调用

a = deco(hello) #将hello函数作为参数传递给deco ,deco的返回值是red函数

print(a()) #调用 a 函数,实际上是调用red函数

print(greet())

# 模块和文件

import sys

print(sys.path) #模块导入的时候,将会在sys.path定义的位置下搜索模块

## 模块导入

import time

import os,sys #不推荐使用

from random import random,choice,randint

import pickle as p #导入的同时,给模块起别名

## 包

def pstar():

print('\*' \* 30)

import mypace.hi

## 相对导入模块

## 绝对导入模块

# 模块

模块支持从逻辑上组织python代码

当代码量变得相当大的时候,最好把代码分成一些有组织的代码段

代码段相互间有一定练习,可能是一个包含数据成员和方法的类,也可鞥是一组相关批次独立的操作,

## 搜索路径

1,模块导入的时候,将会在sys.path定义的位置下搜索模块

import sys

print(sys.path)

路径具体如下:

['/gitall/zdd-repository', '/root/PycharmProjects/day02', '/usr/local/lib/python36.zip', '/usr/local/lib/python3.6', '/usr/local/lib/python3.6/lib-dynload', '/usr/local/lib/python3.6/site-packages', '/root/bin/pycharm2017/pycharm-2017.3/helpers/pycharm\_matplotlib\_backend']

2如果希望自己编写的模块也能在任意位置被导入的两种方法:

方法1,把模块文件拷贝到site-packages/目录下

方法2,使用PYTHONPATH环境变量定义

[root@room9pc01 ~]# export PYTHOPATH=/var/ftp/nsd.... (目录路径为模块所在路径)

# 模块导入方法

方法1,import time

方法2,import os,sys #不推荐使用

方法3,from random import random,choice,randint

方法4,import pickle as p #导入的同时,给模块起别名

## 导入和加载

当导入模块时,模块的顶层代码会被执行

一个模块不管被导入(import)多少次,只会被加载(load)一次



## 从zip文件中导入



# 包

## 目录结构

包是一个有层次的文件目录结构,为平坦的名称空间加入有层次的组织结构

允许程序员把有联系的模块组合到一起

[root@room8pc16 day03]# mkdir mypack

[root@room8pc16 day03]# vim mypack/hi.py

hello = 'hello world'

def pstar():

print('\*' \* 30)

>>> import mypack.hi

>>> mypack.hi.hello

'hello world'

>>> mypack.hi.pstar()

如果使用的是python2，需要在目录下创建\_\_init\_\_.py，否则不能成为包

## 绝对导入模块



## 相对导入模块



# 内置模块

## hashlib模块(MD5哈希算法)

hashlib用来替换md5和sha模块,并使他们的API一致,专门提供hash算法

hash(哈希):是一种加密算法,可以理解为将原始数据,生成一串固定长度的乱码,只要原始数据相同,生成的乱码一定是相同,,,即使原始数据有微小的变化,生成的乱码一定不同.,且无法通过乱码推算出原始数据

用途：

(1) 存储加密的密码 # grep root /etc/shadow

(2) 校验文件的完整性

m = hashlib.md5() #创建文件,

m.update(data) #更新

m.hexdigest() #输出

[root@room8pc16 day03]# cp /etc/passwd /tmp/mima

[root@room8pc16 day03]# md5sum /etc/passwd /tmp/mima

574e11b1a648bc9669b07015ead96f47 /etc/passwd

574e11b1a648bc9669b07015ead96f47 /tmp/mima

包括的算法有：md5/sha

>>> import hashlib

>>> m = hashlib.md5(b'123456') #创建文件

>>> m.hexdigest() #输出

'e10adc3949ba59abbe56e057f20f883e'

>>> m1 = hashlib.md5()

>>> m1.update(b'12')

>>> m1.update(b'34')

>>> m1.update(b'56')

>>> m1.hexdigest()

'e10adc3949ba59abbe56e057f20f883e'

## @@@创建一个将文件生成哈希值的程序

import hashlib

import sys,pickle

def ahash(fname):

m = hashlib.md5() #创建文件

with open(fname,'rb') as fobj:

while True:

data = fobj.read(4096)

if not data:

break

m.update(data) #更新数据,将文件内容分4字节一次加入到m文件

return m.hexdigest() #输出完整文件的哈希值

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

try:

f = sys.argv[1]

print(ahash(f))

except IndexError:

print('错误,请输入需要检测的文件名')

except (FileNotFoundError,NameError):

print('请输入正确的文件名')

## tarfile模块(压缩包)

tarfile模块允许创建/访问tar文件, 同时支持gzip,bzip2格式

tar包、压缩、解压缩

打包解压过程

import os

import tarfile

os.chdir('/etc') #进入目录

os.mkdir('/pythonbackup') #创建目录

tar = tarfile.open('/pythonbackup/mytest.tar.gz','w:gz')

#开始使用gzip压缩.以w方式,会创建一个mytest.tar.ge文件,pythontest目录需提前创建

tar.add('hosts') #添加入要压缩的文件绝对路径

tar.add('/etc/hosts') #绝对路径,会目录结果一起打包

tar.add('security') #继续添加(这是目录)

tar.close()

#以上是压缩文件的全部步骤,以下是解压缩的全部过程

tar = tarfile.open('/pythonbackup/mytest.tar.gz','r:gz') #打开之前创建tar包,以r的方式

tar.extractall('/pythonbackup/mydemo') #解压缩到指定目录,目录没有则会创建

tar.close()

# os.walk 原理分析

>>> import os

>>> result = list(os.walk('/tmp/mydemo/security'))

>>> len(result) #统计列表有五项组成

5

>>> r1 = result[0] #第一项

>>> len(r1) #第一项中有3项组成

3

>>> r2 = result[1] #第二项

>>> len(r2) #第二项也是三项组成

3

>>> r1 #查看第一项如下(是个元组,那么os.walk获得的是五项元组组成的内容)

('/tmp/mydemo/security', ['console.apps', 'console.perms.d', 'limits.d', 'namespace.d'], ['access.conf', 'chroot.conf', 'console.handlers', 'console.perms', 'group.conf', 'limits.conf', 'namespace.conf', 'namespace.init', 'opasswd', 'pam\_env.conf', 'sepermit.conf', 'time.conf', 'pwquality.conf'])

#共有5个元组,第一个元组r1(就是result[0] )的三项分别是:

'/tmp/mydemo/security', #第一项文件路径字符串

['console.apps', 'console.perms.d', 'limits.d', 'namespace.d'], #第二项为第一项的路径中的目录列表

['access.conf', 'chroot.conf', 'console.handlers', 'console.perms', 'group.conf', 'limits.conf', 'namespace.conf', 'namespace.init', 'opasswd', 'pam\_env.conf', 'sepermit.conf', 'time.conf', 'pwquality.conf'] #第三项为第一项路径中的文件列表

>>> r2 #查看第二项如下

('/tmp/mydemo/security/console.apps', [], ['config-util', 'xserver', 'liveinst', 'setup'])

共有5个元组,第二个元组r2(就是result[1])的三项分别是:

'/tmp/mydemo/security/console.apps', #第一项文件路径字符串,该路经在r1的目录列表第一个

[], #第二项为第一项的路径中的目录列表

['config-util', 'xserver', 'liveinst', 'setup'] #第三项为第一项路径中的文件列表

**进行查看**

>>> r1[0]

'/tmp/mydemo/security'

>>> r2[0]

'/tmp/mydemo/security/console.apps'

>>> r1[1]

['console.apps', 'console.perms.d', 'limits.d', 'namespace.d']

>>> r2[1]

[]

>>> r1[2]

['access.conf', 'chroot.conf', 'console.handlers', 'console.perms', 'group.conf', 'limits.conf', 'namespace.conf', 'namespace.init', 'opasswd', 'pam\_env.conf', 'sepermit.conf', 'time.conf', 'pwquality.conf']

>>> r2[2]

['config-util', 'xserver', 'liveinst', 'setup']

os.walk()得到的每一项都是元组，每个元组由三项构成，这三项分别是：路径字符串、

该路径下的目录列表、该目录下文件的列表

>>> for path, folders, files in os.walk('/tmp/mydemo/security'): #遍历walk形成的所有元组

... for file in files: #再遍历所有文件名

... os.path.join(path, file) #合并路径与文件名

path:得到所有的路径名

folders:得到所有的路径中的所有目录名

files:得到所有的文件名

# @@@备份程序

###具有完全备份和增量备份的程序  
**from** time **import** strftime  
**import** os  
**import** tarfile  
**from** check\_md5 **import** check\_md5  
**import** pickle  
  
#定义全量备份函数  
**def full\_backup(**src,dst,md5file**):** #给完全备份的tar包命名  
 # fname = os.path.basename(src) #只获取目标的文件名,可结合到下面一条语句了  
 fname **= '%s\_full\_%s.tar.gz' % (**os.path.basename**(**src**)**, strftime**('%Y%m%d'))** #取名  
 fname **=** os.path.join**(**dst,fname**)** #将备份的tar包名拼接绝对路径  
  
 #打包文件  
 tar **=** tarfile.open**(**fname,**'w:gz')** #打包开始  
 tar.add**(**src**)** #添加要打包的文件  
 tar.close**()** #计算每个文件的md5值  
 md5dict **= {}** #准备用来存放哈希值的列表  
 **for** path, folders, files **in** os.walk**(**src**):** #os.walk方式来获取文件路径和文件名,遍历元组,详情请看walk的详解  
 **for** file **in** files**:** #再遍历得到的所有文件名  
 key **=** os.path.join**(**path, file**)** #拼接所有文件的路径/文件名  
 md5dict**[**key**] =** check\_md5**(**key**)** #经过check\_md5程序产生的哈希值赋值给字典md5dict  
 #自编写的check\_md5模块必须与本程序文件放同一目录,该模块作用是生成哈希值  
 #将字典写入文件  
 **with** open**(**md5file,**'wb') as** fobj**:** #以wb方式打开md5file指定的文件,没有则创建  
 pickle.dump**(**md5dict,fobj**)** #pickle是能将任何内容存入文件中,将字典mdict的内容写入md5file中  
  
**def incr\_backup(**src,dst,md5file**):** #给备份的tar包命名  
 # fname = os.path.basename(src)  
 fname **= '%s\_incr\_%s.tar.gz' % (**os.path.basename**(**src**)**,strftime**('%Y%m%d'))** fname **=** os.path.join**(**dst,fname**)** #计算每个文件的md5值  
 md5dict **= {}** #将用来装哈希值  
 **for** path , folders, files **in** os.walk**(**src**):  
 for** file **in** files**:** key **=** os.path.join**(**path,file**)** md5dict**[**key**] =** check\_md5**(**key**)  
  
 with** open**(**md5file,**'rb') as** fobj**:** old\_md5 **=** pickle.load**(**fobj**)  
  
 with** open**(**md5file,**'wb') as** fobj**:** pickle.dump**(**md5dict,fobj**)** tar **=** tarfile.open**(**fname,**'w:gz')  
 for** key **in** md5dict**:** #遍历md5dict字典,返回key值  
 **if** md5dict**[**key**] !=** old\_md5.get**(**key**):** tar.add**(**key**)** tar.close**()  
  
if** \_\_name\_\_ **== '\_\_main\_\_':** src **= '/pythonbackup/mydemo/security'** #定义要备份的源目录  
 dst **= '/pythonbackup/demo'** #定义放备份文件的目录  
  
 md5file **= '/pythonbackup/demo/md5.data'** #定义校验目录  
 **if** strftime**('%a') == 'Mon':** #判断是不是星期三  
 full\_backup**(**src,dst,md5file**)** #调用全量备份函数  
 **else:** incr\_backup**(**src,dst,md5file**)** #调用增量备份函数

# OOP简介