**17Python\_06时间方法+异常处理+文件系统相关**

**一 时间方法-time模块: time模块,对时间的表示方式**

**1.1 struct\_time元组**

• 时间戳timestamp:表示的是从1970年1月1日00:00:00开始按秒计算的偏移量.

import time

time.time() #当前时间的时间戳

1573176108.572319

• UTC(Coordinated[kəʊˈɔːdɪneɪtɪd] Universal Time,世界协调时)亦即格林威治天文时间,世界标准时间。在中国为UTC+8。DST(Daylight Saving Time)即夏令时

• 9元组(struct\_time):由9个元素组成



**1.2 time模块方法**

• time.localtime([secs]):将一个时间戳转换为当前时区的struct\_time。secs参数未提供,则以当前时间为准

>>> import time

>>> time.localtime() #将本地当前时间转换为9元组

time.struct\_time(tm\_year=2019, tm\_mon=11, tm\_mday=8, tm\_hour=9, tm\_min=22, tm\_sec=41, tm\_wday=4, tm\_yday=312, tm\_isdst=0)

• time.gmtime([secs]):和localtime()方法类似,gmtime()方法是将一个时间戳转换为UTC时区(0时区)的struct\_time

>>> import time

>>> time.gmtime() #将UTC0当前时间转换为9元组

time.struct\_time(tm\_year=2019, tm\_mon=11, tm\_mday=8, tm\_hour=1, tm\_min=40, tm\_sec=39, tm\_wday=4, tm\_yday=312, tm\_isdst=0)

• time.time():返回当前时间的时间戳

>>> import time

>>> time.time() # 将本地当前时间转换为时间戳

1573177455.8614495

• time.mktime(t):将一个struct\_time转化为时间戳

>>> import time

>>> time.localtime() #输出为9元组

time.struct\_time(tm\_year=2019, tm\_mon=11, tm\_mday=8, tm\_hour=9, tm\_min=45, tm\_sec=50, tm\_wday=4, tm\_yday=312, tm\_isdst=0)

>>> time.mktime(time.localtime()) #将9元组转换为时间戳

1573177579.0

• time.sleep(secs):线程推迟指定的时间运行。单位为秒

• time.asctime([t]):把一个表示时间的元组或者struct\_time表示为这种形式:‘Sun Jun 2023:21:05 1993’。如果没有参数,将会将time.localtime()作为参数传入

import time

time.localtime()

time.struct\_time(tm\_year=2019, tm\_mon=11, tm\_mday=8, tm\_hour=9, tm\_min=50, tm\_sec=0, tm\_wday=4, tm\_yday=312, tm\_isdst=0)

time.asctime(time.localtime()) #将本地当前9元组时间转换为固定格式时间

'Fri Nov 8 09:50:21 2019'

time.asctime(time.gmtime()) #将UTC0当前9元组时间转换为固定格式时间

'Fri Nov 8 01:50:38 2019'

• time.ctime([secs]):把一个时间戳(按秒计算的浮点数)转化为time.asctime()的形式;secs(时间戳)未指定,默认本地当前的时间戳

import time

time.ctime(0) #将时间戳0转换为UTC0时间

'Thu Jan 1 08:00:00 1970'

time.ctime() #将本地当前时间戳转换为当前固定格式时间

'Fri Nov 8 09:32:59 2019'

• time.strftime(format[, t]):把一个代表时间的元组或者struct\_time(如由time.localtime()和time.gmtime()返回)转化为格式化的时间字符串。如果t未指定,将传入time.localtime()

>>> import time

>>> time.strftime('%Y-%y-%d %H:%M:%S')

# t未指定,时间默认本地当前9元组时间

'2019-19-08 09:36:39'

>>> time.strftime('%Y-%y-%d %H:%M:%S',time.gmtime())

# t指定时间为UTC0的9元组时间

'2019-19-08 01:59:08'

• time.strptime(string[, format]):把一个格式化时间字符串转化为struct\_time。实际上它和strftime()是逆操作

format省略,默认格式为: '%a %b %d %H:%M:%S %Y'

>>> time.ctime()

'Fri Nov 8 10:06:47 2019'

>>>　time.strptime('Fri Nov 8 10:06:47 2019')

time.struct\_time(tm\_year=2019, tm\_mon=11, tm\_mday=8, tm\_hour=10, tm\_min=6, tm\_sec=47, tm\_wday=4, tm\_yday=312, tm\_isdst=-1)

案例: 取出log.log中9点-12点之间的日志

[root@room9pc01 day06]# vim log.log

2019-10-10 09:00:00 aaaa

2019-10-10 09:09:09 bbbb

2019-10-10 10:10:10 cccc

2019-10-10 11:11:11 dddd

2019-10-10 12:12:12 eeee

import time

t9 = time.strptime('2019-10-10 09:00:00', '%Y-%m-%d %H:%M:%S')

t12 = time.strptime('2019-10-10 12:00:00', '%Y-%m-%d %H:%M:%S')

with open('log.log', 'r') as fobj:

for line in fobj:

t = time.strptime(line[0:19], '%Y-%m-%d %H:%M:%S')

if t > t12:

break

if t >= t9:

print(line, end='')

实现睡眠3秒

>>> import time

>>> time.sleep(3)

**1.3 时间样式**



**计算电脑从1数到9999用了多少时间**

import time

result = 0

start = time.time()

print(start)

for i in range(1,10000):

result += 1

end = time.time()

print(end)

print(result)

print(end-start)

**二 时间方法-datetime模块**

**2.1 datetime模块方法**

• datetime.today():返回一个表示当前本地时间的datetime对象

• datetime.now([tz]):返回一个表示当前本地时间的datetime对象,如果提供了参数tz,则获取tz参数所指时区的本地时间

• datetime.strptime(date\_string, format):将格式字符串转换为datetime对象

• datetime.ctime(datetime对象):返回时间格式字符串

• datetime.strftime(format):返回指定格式字符串

取出当前时间,分别为年 月 日 时 分 秒

>>> import datetime

>>> t1=datetime.datetime.now()

>>> t1.year,t1.month,t1.day

(2019, 11, 8)

创建指定时间,没有指定的,默认为0

>>> t2=datetime(2019,11,11)

返回指定格式的时间字符串

>>> t1.strftime('%Y-%m-%d %H:%M:%S')

'2019-11-08 10:49:58'

**2.2 时间计算**

• 使用timedelta可以很方便的在日期上做天days,小时hour,分钟,秒,毫秒,微妙的时间计算

计算时间差

>>> from datetime import datetime,timedelta

>>> t1 = datetime.now()

>>> days = timedelta(days=50,hours=1)

>>> t1-days #50天之前1小时

datetime.datetime(2019, 9, 19, 9, 58, 18, 105159)

>>> t1+days #50提案之后1小时

datetime.datetime(2019, 12, 28, 11, 58, 18, 105159)

按条件提提取日志

from datetime import datetime

t9 = datetime.strptime('2019-10-10 09:00:00', '%Y-%m-%d %H:%M:%S')

t12 = datetime.strptime('2019-10-10 12:00:00', '%Y-%m-%d %H:%M:%S')

with open('log.log', 'r') as fobj:

for line in fobj:

t = datetime.strptime(line[0:19], '%Y-%m-%d %H:%M:%S')

if t > t12:

break

if t >= t9:

print(line, end='')

**三 异常处理-异常**

**3.1 什么是异常**

• 当python检测到一个错误时,解释器就会指出当前流已经无法继续执行下去,这时候就出现了异常

• 异常是因为程序出现了错误而在正常控制流以外采取的行为

• 这个行为又分为两个阶段:

首先是引起异常发生的错误

然后是检测(和采取可能的措施)阶段

try:

有可能发生异常的语句

except (异常1, 异常2):

发生异常时执行的代码

except (异常3, 异常4):

发生异常时执行的代码

else:

不发生异常才执行的代码

finally:

发不发生异常都要执行的代码

**3.2 Python中的异常**

• 当程序运行时,因为遇到未解的错误而导致中止运行,便会出现traceback消息,打印异常



**3.3 try-except语句**

• 如果没有异常处理,当程序运行遇到错误时,

try:

try\_suite #监控有可能发生异常的代码组

except (错误名称)

**案例:**

try:

n=int(input('number:'))

result = 100/n

print(result)

print('Done')

except ValueError:

print('invalid input')

except ZeroDivisionError:

print('invalid input')

except KeyboardInterrupt:

print('\nBye')

except EOFError:

print('\nBye')

(nsd1906) [root@room9pc01 day06]# python3 try1.py

number:^C # ctrl+c

Bye

(nsd1906) [root@room9pc01 day06]# python3 try1.py

number: # ctrl+d

Bye

(nsd1906) [root@room9pc01 day06]# python3 try1.py

number:abc #输入abc

invalid input

(nsd1906) [root@room9pc01 day06]# python3 try1.py

number:0 #输入0

invalid input

(nsd1906) [root@room9pc01 day06]# python3 try1.py

number:12 #正常输入数字

8.333333333333334

Done

**改进:**

try:

n=int(input('number:'))

result = 100/n

# 将异常保存到变量e中

except (ValueError,ZeroDivisionError,NameError) as e:

print('invalid input:',e)

except (KeyboardInterrupt,EOFError):

print('\nBye')

exit()

else:

print(result)

finally:

print('异常处理完成')

print('Done')

**3.4 带有多个except的try语句**

• 可以把多个except语句连接在一起,处理一个try块中可能发生的多种异常

**3.6 异常参数**

• 异常也可以有参数,异常引发后它会被传递给异常处理器

• 当异常被引发后参数是作为附加帮助信息传递给异常处理器的

>>> try:

... 10 / 0

... except ZeroDivisionError as e:

... print('error', e)

...

error division by zero

**3.7 else字句**

• 在try范围中没有异常被检测到时,执行else子句

• 在else范围中的任何代码运行前,try范围中的所有代码必须完全成功

>>> try:

... result = 100 / int(input("number: "))

... except Exception as e:

... print('Error:', e)

... else:

... print(result)

...

number: 10

10.0

**3.8 finally字句**

• finally子句是无论异常是否发生,是否捕捉都会执行的一段代码

• 如果打开文件后,因为发生异常导致文件没有关闭,可能会发生数据损坏。使用finally可以保证文件总是能正常的关闭

**四 异常处理-主动触发异常**

**4.1 使用raise语句主动触发异常**

• 要想引发异常,最简单的形式就是输入关键字raise,后面跟要引发的异常的名称

• 执行raise语句时,Python会创建指定的异常类的一个对象

• raise语句还可指定对异常对象进行初始化的参数

**4.2 使用assert断言语句触发异常**

• 断言是一句必须等价于布尔值为真的判定

• 此外,发生异常也意味着表达式为假

>>> assert 10 > 100, "Wrong"

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AssertionError: Wrong

def set\_age(name, age):

if not 0 < age < 120:

raise ValueError('年龄超过范围')

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

def set\_age2(name, age):

assert 0 < age < 120, '年龄超过范围'

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

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

set\_age('nb', 200)

set\_age2('nb', 200)

**五 OS相关模块-OS模块**

**5.1 OS模块简介**

• 对文件系统的访问大多通过python的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,如copy等,并没有提供,可以使用shutil模块作为补充

**5.2 OS模块方法**



>>> import os

>>> os.getcwd() # pwd

'/root/xyy/day06'

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

>>> os.makedirs('/tmp/nsd1906/demo') # mkdir -p

>>> os.chdir('/tmp/nsd1906/demo') # cd

>>> os.getcwd() # pwd

'/tmp/nsd1906/demo'

>>> os.listdir() # ls

[]

>>> os.listdir('/etc/security') # ls /etc/security

>>> os.mknod('myfile.txt') # touch

>>> os.symlink('/etc/hosts','zhuji') # ln -s /etc/host zhuji

>>> os.stat('/etc/hosts') 相当于

(nsd1906) [root@room9pc01 day06]# stat /etc/hosts

hosts =os.stat('/etc/hosts') 相当于

hosts.st\_size

**477**

(nsd1906) [root@room9pc01 day06]# ll /etc/hosts

-rw-r--r-- 1 root root **477** 1月 24 2019 /etc/hosts

**案例:代码实现CP功能,带进度条**

# Python3 虚拟环境下 pip install tqdm

import os

import sys

from tqdm import tqdm

def copy(src\_fname, dst\_fname, length=4096):

size = os.stat(src\_fname).st\_size

times, extra = divmod(size, length)

if extra:

times += 1

with open(src\_fname, 'rb') as src\_fobj:

with open(dst\_fname, 'wb') as dst\_fobj:

for i in tqdm(range(times)):

data = src\_fobj.read(length)

dst\_fobj.write(data)

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

copy('/etc/hosts','/tmp/hosts',4096)

**修改文件权限,注意:Linux的权限是8进制数**

[root@room9pc01 demo]# ls -ld /tmp/nsd1906/demo/myfile.txt

-rw------- 1 root root 0 11月 8 14:29 /tmp/nsd1906/demo/myfile.txt

>>> 0o755

493

>>> os.chmod('/tmp/nsd1906/demo/myfile.txt',493)

(nsd1906) [root@room9pc01 demo]# ls -ld /tmp/nsd1906/demo/myfile.txt

-rwxr-xr-x 1 root root 0 11月 8 14:29 /tmp/nsd1906/demo/myfile.txt

**删除文件**

os.remove('/tmp/nsd1906/demo/myfile.txt')

**获取文件绝对路径**

>>> os.listdir()

['zhuji']

>>> os.path.abspath('zhuji') #获取文件的绝对路径

'/tmp/nsd1906/demo/zhuji'

>>> os.path.dirname('/tmp/nsd1906/demo/zhuji') #父目录路径

'/tmp/nsd1906/demo'

>>> os.path.basename('/tmp/nsd1906/demo/zhuji')

'zhuji' #获取路径中的文件名或路径名

(nsd1906) [root@room9pc01 day06]# dirname /tmp/nsd1906/demo/zhuji

/tmp/nsd1906/demo

(nsd1906) [root@room9pc01 day06]# basename /tmp/nsd1906/demo/zhuji

zhuji

>>> os.path.isfile('/etc/hosts') #存在且为文件?

True

>>> os.path.ismount('/') #存在且为挂载点?

True

>>> os.path.isdir('/etc/hosts') #存在且为目录?

False

>>> os.path.islink('/etc/grub2.cfg') #存在且为软连接?

True

>>> os.path.exists('/etc') #存在?

True

**文件路径拆分与拼接**

>>>>>> os.path.ismount('/')

True os.path.split('/tmp/nsd1906/demo/zhuji')

('/tmp/nsd1906/demo', 'zhuji')

>>> os.path.join('/tmp/nsd1906/demo', 'zhuji')

'/tmp/nsd1906/demo/zhuji'

**os.walk的使用**

(nsd1906) [root@room9pc01 day06]# ls -R /etc/security

os.walk('/etc/security')

<generator object walk at 0x7f4fb4205c80>

list(os.walk('/etc/security')) #为了便于识别,进行了排版,原文为列表

[

('/etc/security', # 元组1

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

['limits.conf', 'chroot.conf', 'console.perms', 'group.conf',

'namespace.conf', 'sepermit.conf', 'pam\_env.conf', 'console.handlers',

'access.conf', 'namespace.init', 'opasswd', 'time.conf',

'pwquality.conf']),

('/etc/security/limits.d', # 元组2

[],

['20-nproc.conf']),

('/etc/security/console.apps', # 元组3

[],

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

('/etc/security/console.perms.d', [], []), # 元组4

('/etc/security/namespace.d', [], []) # 元组5

]

result = list(os.walk('/etc/security'))

len(result)

5 #列表里有5个元组

result[0]

('/etc/security',

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

['limits.conf', 'chroot.conf', 'console.perms', 'group.conf',

'namespace.conf', 'sepermit.conf', 'pam\_env.conf', 'console.handlers',

'access.conf', 'namespace.init', 'opasswd', 'time.conf',

'pwquality.conf']

)

>>> result[1]

('/etc/security/limits.d', [], ['20-nproc.conf'])

分析,result列表共有5项,每项的内容的结构完全一样

result[0]的结构:(字符串,列表,列表) #元组里3个列表

字符串:路径 #对应代码中的path

第一个列表:路径下的目录 #对应代码中的folder

第二个列表:路径下的文件 #对应代码中的files

result[1]的结构:(字符串,列表,列表) #元组里3个列表

字符串: /etc/security下第一个目录的路径

第一个列表:路径下的目录,为空

第二个列表:路径下的文件

代码:

import os

import sys

def lsdir(directory):

for path,folder,files in os.walk(directory):

print('%s:' % path)

for d in folder:

print('\033[31;1m%s\033[0m\t' % d,end='')

for f in files:

print('%s\t' % f,end='')

print('\n')

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

lsdir(sys.argv[1])

**六 OS相关模块-pickle模块**

**6.1 pickle模块简介**

• 通过文件的write方法,只能把字符形式的数据写入文件。

• pickle:python的一个标准模块;可以将任意类型的数据(对象)写入文件,并且无损的提取出来.

**6.2 pickle模块方法**

• 分别调用dump()和load()可以存储、写入

>>>　import pickle

>>> adict={'name':'bob','age':20}

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

>>> **pickle.dump**(adict,f) #将字典写入文件

f.close()

>>> f=open('/tmp/a','rb')

>>> **bdict=pickle.load**(f)

>>> f.close()

>>> bdict

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

案例:记账程序

1. 开始,1000元; 2. 不论支出收入都要记账; 3. 记账的内容:时间 金额 说明; 4. 记账数据要求永久存储; 5. 支持查账

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| date | save | cost | balance | comment |
| 2019-11-8 | 0 | 0 | 10000 | init |
| 2019-11-8 | 0 | 200 | 9800 | eat |
| 2019-11-10 | 2000 | 0 | 2980 | salary |

数据结构

记账时,可以将每1笔记账写成一行小列表,再将每一行小列表写成大列表

>>> records=[]

>>> init\_data=['2019-11-08',0,0,10000,'init']

>>> records.append(init\_data)

>>> records

[['2019-11-08', 0, 0, 10000, 'init']]

>>> records[-1][-2] #从大列表中取出小列表,再从小列表中取出元素

10000

>>> record=['2019-11-10',2000,0,records[-1][-2]+2000,'salary']

>>> records.append(record)

>>> records

[['2019-11-08', 0, 0, 10000, 'init'], ['2019-11-10', 2000, 0, 12000, 'salary']]

import os

import pickle

from time import strftime

def save(fname):

amount=int(input('金额:'))

comment=input('备注:')

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

#从文件中取出全波记录

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

records=pickle.load(fobj)

#计算最新余额

balance=records[-1][-2]+amount

record=[date,amount,0,balance,comment]

records.append(record)

#把更新后的列表再次存入文件

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

pickle.dump(records,fobj)

def cost(fname):

amount=int(input('金额:'))

comment=input('备注:')

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

#从文件中取出全波记录

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

records=pickle.load(fobj)

#计算最新余额

balance=records[-1][-2]-amount

record=[date,0,amount,balance,comment]

records.append(record)

#把更新后的列表再次存入文件

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

pickle.dump(records,fobj)

def query(fname):

#取出全部记录

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

records=pickle.load(fobj)

#打印表头

print(

'%-12s%-8s%-8s%-12s%-20s' % ('date','save','cost','balance','commit')

)

#打印记录

for record in records:

print(

'%-12s%-8s%-8s%-12s%-20s' % tuple(record)

)

def show\_menu():

cmds={'0':save,'1':cost,'2':query}

prompt="""(0) save

(1)cost

(2)query

(3)quit

Please input your choice(0/1/2/3)"""

fname='account.data'

init\_data=[

[strftime('%Y-%m-%d'),0,0,10000,'init data']

]

#若文件不存在,把初始化数据写进去

if not os.path.exists(fname):

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

pickle.dump(init\_data,fobj)

while 1:

choice=input(prompt).strip()

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

print('invalid input,please try again.')

continue

if choice=='3':

print('Bye!')

break

cmds[choice](fname)

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

show\_menu()