nsd1903_py02_day01

time模块

时间的表示方式：

- 时间戳：1970-1-1 00:00:00到某一时间点之间的秒数
- UTC时间：世界协调时
- 9元组：

```
>>> import time
# 时间戳
>>> time.time()
1565141637.870625
# UTC时间
>>> time.ctime()   # 返回当前时间
'Wed Aug  7 09:34:59 2019'
# 9元组
>>> time.localtime()  # 当前时间的9元组，括号中是时间对象拥有的属性
time.struct_time(tm_year=2019, tm_mon=8, tm_mday=7, tm_hour=9, tm_min=35, tm_sec=39,
tm_wday=2, tm_yday=219, tm_isdst=0)
>>> t1 = time.localtime()
>>> t1.tm_year
2019
>>> t1.tm_yday
219

# 睡眠3秒
time.sleep(3)

# 将时间转换成指定的字符串样式
>>> time.strftime('%Y-%m-%d %H:%M:%S')
'2019-08-07 09:51:28'
>>> time.strftime('%a')
'Wed'

# 将时间字符串转换为9元组时间对象
>>> time.strptime('2019-08-07 10:17:23', '%Y-%m-%d %H:%M:%S')
time.struct_time(tm_year=2019, tm_mon=8, tm_mday=7, tm_hour=10, tm_min=17, tm_sec=23,
tm_wday=2, tm_yday=219, tm_isdst=-1)
```

datetime模块

- datetime.datetime模块用于表示时间
- dateteme.timedelta常用于计算时间差额

```
>>> from datetime import datetime
```

```
>>> from datetime import datetime
>>> t = datetime.now()
>>> t   # 括号中是datetime对象的属性
datetime.datetime(2019, 8, 7, 10, 56, 27, 91802)
>>> t.year, t.month, t.day, t.hour, t.minute, t.second, t.microsecond
(2019, 8, 7, 10, 56, 27, 91802)

# 获取datetime对象相应的时间字符串
>>> t.strftime('%Y-%m-%d %H:%M:%S')
'2019-08-07 10:56:27'

# 将时间字符串转换成datetime对象
>>> datetime.strptime('2019-8-7', '%Y-%m-%d')
datetime.datetime(2019, 8, 7, 0, 0)

# 100天3小时之前、之后的时间
>>> from datetime import datetime, timedelta
>>> days = timedelta(days=100, hours=3)
>>> t = datetime.now()
>>> t - days
datetime.datetime(2019, 4, 29, 8, 24, 9, 967534)
>>> t + days
datetime.datetime(2019, 11, 15, 14, 24, 9, 967534)
```

异常处理

程序报错了怎么办？

没有异常处理，程序遇到错误就崩溃终止执行了。异常处理需要发现问题，并给出解决问题的编码方案，使得程序再遇到错误时，不会崩溃，仍然能够向下继续执行。

异常处理的完整代码：

```
try:
    有可能发生异常的语句
except 异常1:
    处理代码
except (异常2, 异常3):
    处理代码
... ...
except 异常n:
    处理代码
else:
    不发生异常才执行的代码
finally:
    不管是否发生异常都执行的代码
```

os模块

对文件系统的访问大多通过python的os模块实现。

```
>>> import os
>>> os.getcwd()   # pwd
'/var/ftp/nsd2019/nsd1903/python02/day01'
>>> os.listdir()  # ls
>>> os.listdir('/tmp')   # ls /tmp
>>> os.mkdir('/tmp/demo')
>>> os.makedirs('/tmp/aaaa/bbb/ccc')   # mkdir -p
>>> os.chdir('/tmp/demo')   # cd /tmp/demo
>>> os.listdir()   # ls
>>> import shutil
>>> shutil.copy('/etc/passwd', '.')
>>> os.listdir()

>>> os.symlink('/etc/hosts', 'zhuji')   # ln -s /etc/hosts zhuji
>>> os.mkdir('abc')
>>> os.listdir()
['passwd', 'zhuji', 'abc']
>>> os.rmdir('abc')   # rmdir abc   只能删除空目录
>>> os.remove('zhuji')   # rm -rf zhuji

>>> os.path.basename('/tmp/demo/abc.txt')   # 返回文件名部分
'abc.txt'
>>> os.path.dirname('/tmp/demo/abc.txt')   # 返回路径部分
'/tmp/demo'
>>> os.path.split('/tmp/demo/abc.txt')  # 切割
('/tmp/demo', 'abc.txt')
>>> os.path.abspath('.')   # 返回当前路径的绝对路径
'/tmp/demo'
>>> os.path.abspath('passwd')  # 返回当前目录下文件的绝对路径
'/tmp/demo/passwd'
>>> os.path.join('/tmp/demo', 'abc.txt')  # 拼接路径
'/tmp/demo/abc.txt'

>>> os.path.isabs('passwd')  # 是绝对路径吗？
False
>>> os.path.isfile('/etc/hosts')   # 存在并且是文件吗？
True
>>> os.path.ismount('/')   # 是挂载点吗？
True
>>> os.path.isdir('/etc/abcd')   # 存在并且是目录吗？
False
>>> os.path.islink('/etc/grub2.cfg')  # 存在并且是链接吗？
True
>>> os.path.exists('/etc/')   # 存在吗？
True
```

pickle模块

它可以将任意数据类型写入到文件，并且可以无损地取出。

```
>>> f = open('/tmp/myfil', 'w')
>>> f.write('Hello World.\n')
```

```
13
# 常规则的文件操作，只能把字符写入文件，不能写其他数据类型
>>> f.write(100)   # 报错
>>> f.write({'name': 'tom', 'age': 20})   # 报错
>>> f.close()

# 使用pickle模块操作
>>> import pickle
>>> user = {'name': 'tom', 'age': 20}
# 将字典写入文件
>>> with open('/tmp/myfile', 'wb') as f:
...   pickle.dump(user, f)

# 在文件中将字典取出
>>> with open('/tmp/myfile', 'rb') as f:
...   mydict = pickle.load(f)

>>> mydict
{'name': 'tom', 'age': 20}
```

练习：记账程序

- 假设在记账时,有一万元钱
- 无论是开销还是收入都要进行记账
- 记账内容包括时间、金额和说明等
- 记账数据要求永久存储

1. 程序的运作方式

```
(0) 收入
(1) 支出
(2) 查询
(3) 退出
请选择(0/1/2/3): 2
日期          收入      支出      余额        说明
2019-8-7      0         0         10000      初始化
(0) 收入
(1) 支出
(2) 查询
(3) 退出
请选择(0/1/2/3): 1
金额： 100
备注： 七夕买花
(0) 收入
(1) 支出
(2) 查询
(3) 退出
请选择(0/1/2/3): 2
日期          收入      支出      余额        说明
2019-8-7      0         0         10000      初始化
2019-8-7      0         100       9900       七夕买花
(0) 收入
(1) 支出
```

```
(2) 查询
(3) 退出
请选择(0/1/2/3): 0
金额： 10000
备注： 工资
(0) 收入
(1) 支出
(2) 查询
(3) 退出
请选择(0/1/2/3): 2
日期          收入      支出      余额        说明
2019-8-7      0         0         10000       初始化
2019-8-7      0         100       9900        七夕买花
2019-8-7      10000     0         19900       工资
(0) 收入
(1) 支出
(2) 查询
(3) 退出
请选择(0/1/2/3): 3
Bye-bye
```

2. 思考有哪些功能，将功能写成函数

```
def save():


def cost():


def query():


def show_menu():


if __name__ == '__main__':
    show_menu()
```

3. 分析存储的数据结构

```
# 将每一行数据保存成一个小列表，所有行存入一个大列表
records = [
    ['2019-8-7', 0, 0, 10000, 'init'],
    ['2019-8-7', 0, 100, 9900, 'buy flowers'],
    ['2019-8-7', 10000, 0, 19900, 'salary'],
]
# 取出最新余额
>>> records[-1][-2]
19900
# 取出时间
>>> import time
>>> time.strftime('%Y-%m-%d')
'2019-08-07'
```

1. 填写函数代码