nsd1902_py02_day01

时间表示方式

- 时间戳：1970-1-1 0:00:00 到某一时间点之间的秒数

```
>>> import time
>>> time.time()
```

- UTC：世界协调时

```
>>> time.ctime()
'Mon Jul  8 09:47:54 2019'
```

- 九元组struct_time

```
>>> time.localtime()
time.struct_time(tm_year=2019, tm_mon=7, tm_mday=8, tm_hour=9, tm_min=49, tm_sec=5,
tm_wday=0, tm_yday=189, tm_isdst=0)
>>> t = time.localtime()
>>> t.tm_year
2019
>>> t.tm_hour
9
```

time模块

```
>>> time.sleep(3)    # 睡眠3秒
>>> time.strftime('%Y-%m-%d %H:%M:%S')   # 转成指定格式
'2019-07-08 09:59:59'
# 字符串时间转成9元组格式
>>> time.strptime('2019-07-08 09:59:59', '%Y-%m-%d %H:%M:%S')
time.struct_time(tm_year=2019, tm_mon=7, tm_mday=8, tm_hour=9, tm_min=59, tm_sec=59,
tm_wday=0, tm_yday=189, tm_isdst=-1)
```

datetime模块

```
>>> import datetime
>>> t1 = datetime.datetime.now()
>>> t1
datetime.datetime(2019, 7, 8, 10, 54, 25, 922956)
# 因为上面的写法太长了，可以改为以下方式
>>> from datetime import datetime
>>> t1 = datetime.now()
```

```
>>> t1   # 年月日时分秒毫秒
datetime.datetime(2019, 7, 8, 10, 55, 17, 81886)
>>> t1.year, t1.month, t1.day, t1.hour, t1.minute, t1.second, t1.microsecond
(2019, 7, 8, 10, 55, 17, 81886)
>>> t1.year
2019
# 将datetime对象转成时间字符串
>>> datetime.strftime(t1, '%Y-%m-%d %H:%M:%S')
'2019-07-08 10:55:17'
# 将时间字符串转换成datetime对象
>>> datetime.strptime('2019-07-08 10:55:17', '%Y-%m-%d %H:%M:%S')
datetime.datetime(2019, 7, 8, 10, 55, 17)
# 创建指定时间的datetime对象
>>> t = datetime(2019, 7, 8)
>>> t
datetime.datetime(2019, 7, 8, 0, 0)
```

计算时间差额，如100天零4小时前、100天零4小时后是什么时候

```
>>> from datetime import datetime, timedelta
>>> dt = timedelta(days=100, hours=4)
>>> t = datetime.now()
>>> t
datetime.datetime(2019, 7, 8, 11, 38, 13, 922027)
>>> t - dt
datetime.datetime(2019, 3, 30, 7, 38, 13, 922027)
>>> t + dt
datetime.datetime(2019, 10, 16, 15, 38, 13, 922027)
```

异常处理

程序遇到错误时，如果没有相应的处理代码，将会崩溃、终止执行。

异常处理就是把有可能发生异常的语句发到try中去执行，用except捕获发生的异常。完整的语法是：

```
try:
    有可能发生异常的语句
except 异常名字:
    处理异常的代码
else:
    不发生异常才执行的代码
finally:
    不管异常是否发生，都要执行的代码
```

os模块

```
>>> import os
>>> os.getcwd()   # pwd
>>> os.listdir()  # ls
>>> os.listdir('/tmp')  # ls /tmp
```

```
>>> os.makedirs('/tmp/mydemo/mydir')   # mkdir -p
>>> os.mkdir('/tmp/abcde')   # mkdir
>>> os.chdir('/tmp/mydemo/mydir')  # cd /tmp/mydemo/mydir
>>> os.listdir()
[]
>>> os.mknod('hello')   # touch hello
>>> os.listdir()
['hello']
>>> os.symlink('/etc/hosts', 'zhuji')   # ln -s /etc/hosts zhuji
>>> os.chmod('hello', 0o644)   # chmod 644 hello
>>> os.rename('hello', 'welcome')   # mv hello welcome
>>> os.rmdir('/tmp/abcde')    # rmdir /tmp/abcde   只能删空目录
>>> os.unlink('zhuji')   # unlink zhuji   # 删除软链接
>>> os.remove('welcome')   # rm -f welcome
```

os.path子模块

```
>>> os.path.abspath('.')   # 当前路径的绝对路径
'/tmp/mydemo/mydir'
>>> os.path.split('/tmp/demo/abc.txt')   # 路径切割
('/tmp/demo', 'abc.txt')
>>> os.path.dirname('/tmp/demo/abc.txt')
'/tmp/demo'
>>> os.path.basename('/tmp/demo/abc.txt')
'abc.txt'
>>> os.path.join('/tmp/demo', 'abc.txt')   # 路径拼接
'/tmp/demo/abc.txt'
>>> os.path.isdir('/etc')  # 存在并且是目录吗？
True
>>> os.path.isfile('/etc/hosts')  # 存在并且是文件吗？
True
>>> os.path.islink('/etc/passwd')  # 存在并且是链接吗？
False
>>> os.path.ismount('/')  # 存在并且是挂载点吗？
True
>>> os.path.exists('/abcd')   # 存在吗？
False
```

pickle模块

当写入文件时，只能写入字符串，写入其他类型的数据，就会报错：

```
>>> f = open('/tmp/abc.txt', 'w')
>>> f.write('hello world!\n')
13
>>> f.write(100)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: write() argument must be str, not int
```

pickle模块可以把任意的数据类型写到文件中，还能无损地取出。

```
>>> import pickle
>>> shopping_list = ['apple', 'banana', 'orage']
>>> with open('/tmp/shop.data', 'wb') as fobj:
...     pickle.dump(shopping_list, fobj)

# 取出数据，还是列表的形式
>>> import pickle
>>> with open('/tmp/shop.data', 'rb') as fobj:
...     mylist = pickle.load(fobj)
...
>>> type(mylist)
<class 'list'>
>>> mylist
['apple', 'banana', 'orage']
```

记账程序练习

日期	收入	支出	余额	说明
2019-07-08	0	0	10000	init
2019-07-09	10000	0	20000	salary
2019-07-09	0	200	19800	eat

```
# 数据保存的形式
# 用列表存储数据，第一行是列表中的一项。列表中的每一项又是列表
# 小列表中的每一项就是各个字段
# 初始化时列表的样式：
[
    ['2019-07-08', 0, 0, 10000, 'init'],
]
# 记录一次收入时列表的样式：
[
    ['2019-07-08', 0, 0, 10000, 'init'],
    ['2019-07-09', 10000, 0, 20000, 'salary'],
]
```

取出小列表中的余额：

```
>>> data = [
...     ['2019-07-08', 0, 0, 10000, 'init'],
... ]
>>> data[-1]
['2019-07-08', 0, 0, 10000, 'init']
>>> data[-1][-2]
10000
```