# nsd1905_py02_day01

## 时间模块

### time模块

时间表示方法

- 时间戳：自1970-1-1 0:0:0到某一时间点之间的秒数
- UTC时间：世界协调时。以英国格林威治这个城市所在的经度点为0时区，向东向西，每隔15度角为1个时区，全球共分为24个时区。
- struct_time：它是一个9元组：(年、月、日、时、分、秒、一周中的第几天、一年中的第几天、是否使用夏季节约时间)

```
>>> import time
>>> time.time()   # 自1970-1-1 8:0:0到执行指令间的秒数
1570843812.3590305
>>> time.ctime()
'Sat Oct 12 09:31:20 2019'
>>> time.ctime(0)  # 参数是一个秒数
'Thu Jan  1 08:00:00 1970'
>>> time.localtime()  # 当前时间的struct_time
time.struct_time(tm_year=2019, tm_mon=10, tm_mday=12, tm_hour=9, tm_min=33, tm_sec=10,
tm_wday=5, tm_yday=285, tm_isdst=0)
>>> t1 = time.localtime()
>>> t1.tm_year
2019
>>> t1.tm_hour, t1.tm_min
(9, 33)
>>> time.sleep(3)   # 睡眠3秒
>>> time.strftime('%Y-%m-%d %H:%M:%S')  # 年-月-日 时:分:秒
'2019-10-12 09:49:27'
# 将时间字符串转换为9元组时间
>>> t2 = time.strptime('20191001 09:50:00', '%Y%m%d %H:%M:%S')
>>> t2.tm_mon
>>> t2.tm_mday
```

## datetime模块

```
>>> from datetime import datetime
>>> datetime.now()  # 返回年月日时分秒毫秒
datetime.datetime(2019, 10, 12, 10, 39, 47, 593038)
>>> t1 = datetime.now()
>>> t1.year, t1.month, t1.day, t1.hour, t1.minute, t1.second, t1.microsecond
(2019, 10, 12, 10, 40, 58, 342680)
>>> t1.strftime('%H:%M:%S')
'10:40:58'
```

```
# 将时间字符串转成datetime对象
>>> t2 = datetime.strptime('20191001 09:50:00', '%Y%m%d %H:%M:%S')
>>> t2
datetime.datetime(2019, 10, 1, 9, 50)

>>> from datetime import datetime, timedelta
>>> t = datetime.now()
>>> days = timedelta(days=50, hours=2)
>>> t - days  # 50天2小时之前
datetime.datetime(2019, 8, 23, 8, 54, 5, 724316)
>>> t + days  # 50Gd 2小时之后
datetime.datetime(2019, 12, 1, 12, 54, 5, 724316)
```

## 异常处理

```
try:
    有可能发生异常的语句块
except 异常1:
    解决代码1
except 异常2:
    解决代码2
else:
    不发生异常才执行的语句
finally:
    不管异常是否发生，都要执行的语句
```

### 主动触发异常

- 使用raise语句，异常名称自己决定
- 使用assert语句。一定触发AssertionError异常

## os模块

```
>>> import os
>>> os.getcwd()   # pwd
>>> os.listdir()  # ls
>>> os.listdir('/tmp')   # ls /tmp
>>> os.mkdir('/tmp/demo')   # mkdir /tmp/demo
>>> os.makedirs('/tmp/mydemo/aaa')   # mkdir -p /tmp/mydemo/aaa
>>> os.chdir('/tmp/mydemo/aaa')   # cd /tmp/mydemo/aaa
>>> os.getcwd()
'/tmp/mydemo/aaa'
>>> os.symlink('/etc/hosts', 'zhuji')   # ln -s /etc/hosts zhuji
>>> os.listdir()
['zhuji']

>>> os.mknod('hello.txt')   # touch hello.txt
>>> os.listdir()
['zhuji', 'abc', 'hello.txt']
>>> os.rmdir('abc')   # rmdir abc   # 只能删除空目录
>>> os.listdir()
['zhuji', 'hello.txt']
>>> os.remove('hello.txt')   # rm -f hello.txt
>>> os.rename('zhuji', 'zhuji.lnk')   # mv zhuji zhuji.lnk
>>> os.listdir()
['zhuji.lnk']

>>> os.listdir()
['zhuji.lnk']
>>> os.path.abspath('zhuji.lnk')   # 返回绝对路径
'/tmp/mydemo/aaa/zhuji.lnk'
>>> os.path.basename('/tmp/mydemo/aaa/zhuji.lnk')
'zhuji.lnk'
>>> os.path.dirname('/tmp/mydemo/aaa/zhuji.lnk')
'/tmp/mydemo/aaa'
>>> os.path.split('/tmp/mydemo/aaa/zhuji.lnk')
('/tmp/mydemo/aaa', 'zhuji.lnk')
>>> os.path.join('/tmp/mydemo/aaa', 'zhuji.lnk')
'/tmp/mydemo/aaa/zhuji.lnk'

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

os.walk的使用

os.walk可以递归遍历所有目录的内容

```
>>> import pprint
>>> import os
>>> result = list(os.walk('/etc/security'))
>>> pprint.pprint(result)
[('/etc/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']),
  ('/etc/security/console.apps',
   [],
   ['config-util', 'xserver', 'liveinst', 'setup']),
  ('/etc/security/console.perms.d', [], []),
  ('/etc/security/limits.d', [], ['20-nproc.conf']),
  ('/etc/security/namespace.d', [], [])]
# 分析
# 1. 判断有多少项
>>> len(result)
5
# 列表中有5项，每项的结构相同
# 2. 查看列表中的各项特点
>>> result[0]
('/etc/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'])
>>> result[1]
('/etc/security/console.apps', [], ['config-util', 'xserver', 'liveinst', 'setup'])
# 列表由元组构成
# 3. 分析元组
>>> f1 = result[0]
>>> len(f1)
3
>>> f2 = result[1]
>>> len(f2)
3
# 每个元组结构也完全一样，元组由3项内容构成:
# (路径字符串，路径下目录列表，路径下文件列表)

# 输出每个路径下的内容
>>> for path, folders, files in os.walk('/etc/security'):
...   print('%s:\n%s\n%s' % (path, folders, files))
...   print()

>>> for path, folders, files in os.walk('/etc/security'):
...   print('%s:' % path)
...   for d in folders:
...     print(d, end='\t')
...   for f in files:
...     print(f, end='\t')
...   print('\n')
```

以上内容简化：

```
>>> alist = [('aaa', [1, 2, 3], [4, 5, 6]), ('bbb', [10, 20, 30], [40,50])]
>>> for i in alist:
...   print(i)
...
('aaa', [1, 2, 3], [4, 5, 6])
```

```
('bbb', [10, 20, 30], [40, 50])

>>> for i in alist:
...   print(i[0], i[1], i[2])
...
aaa [1, 2, 3] [4, 5, 6]
bbb [10, 20, 30] [40, 50]

>>> for a, b, c in alist:
...   print(a, b, c)
...
aaa [1, 2, 3] [4, 5, 6]
bbb [10, 20, 30] [40, 50]


>>> for a, b, c in alist:
...   print('%s:' % a)
...   print(b)
...   print(c)
...
aaa:
[1, 2, 3]
[4, 5, 6]
bbb:
[10, 20, 30]
[40, 50]

>>> for a, b, c in alist:
...   print('%s:' % a)
...   for i in b:
...     print(i, end=', ')
...   print()
...   for j in c:
...     print(j, end=', ')
...   print()
...
aaa:
1, 2, 3,
4, 5, 6,
bbb:
10, 20, 30,
40, 50,
```

## pickle模块

- 普通文件只能写入str或bytes类型的数据
- pickle存储器可以把任意数据类型写入文件，还能无损地取出

```
>>> import pickle
>>> f = open('/tmp/data.txt', 'wb')
>>> shopping_list = ['apple', 'egg', 'banana']
>>> pickle.dump(shopping_list, f)
>>> f.close()

>>> import pickle
>>> f = open('/tmp/data.txt', 'rb')
>>> mylist = pickle.load(f)
>>> f.close()
>>> type(mylist)
<class 'list'>
>>> mylist
['apple', 'egg', 'banana']
```

## 练习：记账程序

记账：

| date | save | cost | balance | comment |
|---|---|---|---|---|
| 2019-10-12 | 0 | 0 | 10000 | init |
| 2019-10-12 | 0 | 60 | 9940 | eat |
| 2019-10-13 | 0 | 200 | 9740 | buy clothes |
| 2019-10-13 | 10000 | 0 | 19740 | salary |

数据结构的表示

```
>>> records = []
>>> record = ['2019-10-12', 0, 0, 10000, 'init']
>>> records.append(record)
>>> record = ['2019-10-12', 0, 100, 9900, 'eat']
>>> records.append(record)
>>> records
[['2019-10-12', 0, 0, 10000, 'init'], ['2019-10-12', 0, 100, 9900, 'eat']]
# 取出最新余额
>>> records[-1]
['2019-10-12', 0, 100, 9900, 'eat']
>>> records[-1][-2]
9900
```