nsd1906_py02_day01

时间相关模块

time模块

time模块对时间的表示方式：

- 时间戳：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。以英国格林威治所在的纵切面作为标准时区，每隔15度角是一个时区，共24个时区。
- 9元组：(年，月，日，时，分，秒，一周中的第几天，一年中的第几天，是否为夏时制)

```
# 当前时间的时间戳
>>> import time
>>> time.time()
1573176067.5228834
# 字符串表示的UTC时间
>>> time.ctime()
'Fri Nov  8 09:32:08 2019'
>>> time.ctime(0)   # 0是时间戳
'Thu Jan  1 08:00:00 1970'
# 时间格式化
>>> time.strftime('%Y-%m-%d %H:%M:%S')
'2019-11-08 09:35:09'
>>> time.strftime('%Y-%m-%d %a %H:%M:%S')
'2019-11-08 Fri 09:35:37'
# 9元组时间
>>> t1 = time.localtime()
>>> t1.tm_year
2019
>>> t1.tm_yday
312
>>> time.strptime('2019-11-11 00:00:00', '%Y-%m-%d %H:%M:%S')
time.struct_time(tm_year=2019, tm_mon=11, tm_mday=11, tm_hour=0, tm_min=0, tm_sec=0,
tm_wday=0, tm_yday=315, tm_isdst=-1)
# 睡眠3秒
>>> time.sleep(3)
```

datetime模块

```
# 取出当前时间，分别为年、月、日、时、分、秒、毫秒
>>> t1 = datetime.now()
>>> t1.year, t1.month, t1.day
(2019, 11, 8)
>>> t1.hour, t1.minute, t1.second, t1.microsecond
(10, 41, 34, 707563)
# 创建指定时间，没有指定的，默认为0
```

```
>>> t2 = datetime(2019, 11, 11)
# 返回指定格式字符串
>>> t1.strftime('%Y-%m-%d %H:%M:%S')
'2019-11-08 10:41:34'
# 将指定字符串转为datetime对象
>>> t3 = datetime.strptime('2019-11-11 08:00:00', '%Y-%m-%d %H:%M:%S')
>>> t3
datetime.datetime(2019, 11, 11, 8, 0)

# 计算时间差
>>> from datetime import datetime, timedelta
>>> t1 = datetime.now()
>>> days = timedelta(days=50, hours=1)
>>> t1 - days  # 50天1小时之前
datetime.datetime(2019, 9, 19, 9, 55, 48, 298941)
>>> t1 + days  # 50天1小时之后
datetime.datetime(2019, 12, 28, 11, 55, 48, 298941)
```

异常处理

- 如果没有异常处理，当程序运行遇到错误时，程序将崩溃、终止执行。
- 异常处理就是检测到发生的错误，并给出解决方案，使程序可以继续运行。

```
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
```

主动触发异常

- 使用raise，触发指定类型的异常
- 使用assert，触发断言异常AssertionError

os模块

```
>>> import os
>>> os.getcwd()   # pwd
'/var/ftp/nsd2019/nsd1906/py02/day01'
>>> os.mkdir('/tmp/demo')   # mkdir /tmp/demo
>>> os.makedirs('/tmp/nsd1906/demo')   # mkdir -p /tmp/nsd1906/demo
>>> os.chdir('/tmp/nsd1906/demo')  # cd /tmp/nsd1906/demo
>>> os.getcwd()
'/tmp/nsd1906/demo'
>>> os.listdir()   # ls
[]
```

```
>>> os.listdir('/etc/security')   # ls /etc/security
>>> os.mknod('myfile.txt')   # touch myfile.txt
>>> os.listdir()
['myfile.txt']
>>> os.symlink('/etc/hosts', 'zhuji')  # ln -s /etc/hosts zhuji
>>> hosts = os.stat('/etc/hosts')  # stat /etc/hosts
>>> hosts.st_size   # 文件大小
9806

# 改权限，注意，linux的权限是8进制数
>>> os.chmod('myfile.txt', 0o644)   # chmod 644 myfile.txt
>>> 0o755
493
>>> os.chmod('myfile.txt', 493)   # chmod 755 myfile.txt
>>> os.remove('zhuji')   # rm zhuji

>>> os.listdir()
['myfile.txt']
>>> os.path.abspath('myfile.txt')  # 取出绝对路径
'/tmp/nsd1906/demo/myfile.txt'
>>> os.path.basename('/tmp/nsd1906/demo/myfile.txt')
'myfile.txt'
>>> os.path.dirname('/tmp/nsd1906/demo/myfile.txt')
'/tmp/nsd1906/demo'
>>> os.path.split('/tmp/nsd1906/demo/myfile.txt')
('/tmp/nsd1906/demo', 'myfile.txt')
>>> os.path.join('/tmp/nsd1906/demo', 'myfile.txt')  # 路径拼接
'/tmp/nsd1906/demo/myfile.txt'

>>> os.path.isfile('/etc/hosts')  # 存在并且是文件吗？
True
>>> os.path.ismount('/etc')   # 是挂载点吗？
False
>>> os.path.isdir('/etc/host')   # 存在并且是目录吗？
False
>>> os.path.islink('/etc/grub2.cfg')   # 存在并且是软链接吗？
True
>>> os.path.exists('/etc')   # 存在吗？
True
```

os.walk的使用

```
>>> list(os.walk('/etc/security'))
>>> result = list(os.walk('/etc/security'))
>>> len(result)
5
>>> 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'])
>>> len(result[0])
3
```

```
>>> result[1]
('/etc/security/console.apps', [], ['config-util', 'xserver', 'liveinst', 'setup'])
# 分析，result列表共有5项，每项的内容其结构完全一样。
# 如result[0]的结构：(字符串, 列表, 列表)
# 字符串：路径
# 第一个列表：路径下的目录
# 第二个列表：路径下的文件


# 循环5遍，a是元组，元组长度为3
>>> for a in os.walk('/etc/security'):
...   print(a)
...   print()

# 既然元组有3项，可以把这三项分开赋值
# a: 路径字符串
# b: 路径下的目录列表
# c: 路径下的文件列表
>>> for a, b, c in os.walk('/etc/security'):
...   print(a, b, c)
...   print()

# 既然b和c仍然是列表，还可以对它们继续进行遍历
>>> for a, b, c in os.walk('/etc/security'):
...   print(a)
...   for d in b:
...     print(d, end='\t')
...   print()
...   for e in c:
...     print(e, end='\t')
...   print()
```

pickle模块

- 通过文件的write方法，只能把字符形式的数据写入文件
- pickle可以将任意类型的数据写入文件，还能无损地取出

```
>>> import pickle
>>> adict = {'name': 'bob', 'age': 20}
>>> f = open('/tmp/a.data', 'wb')
>>> pickle.dump(adict, f)   # 将字典写入文件
>>> f.close()

>>> import pickle
>>> f = open('/tmp/a.data', 'rb')
>>> bdict = pickle.load(f)
>>> f.close()
>>> bdict
{'name': 'bob', 'age': 20}
```

记账练习

- 功能：记收入、记支出、查账
- 记账内容：

date	save	cost	balance	comment
2019-11-8	0	0	10000	init
2019-11-8	0	200	9800	eat
2019-11-10	20000	0	29800	salary

- 数据结构

```
# 记账时，可以将每一笔记账写成小列表(字典)。
# 再将每一行放到大列表中
>>> records = []  # 用于存放全部记录
>>> init_data = ['2019-11-08', 0, 0, 10000, 'init data']  # 一行记录
>>> records.append(init_data)
>>> records
[['2019-11-08', 0, 0, 10000, 'init data']]
>>> records[-1]   # 大列表的最后一项，仍然是列表
['2019-11-08', 0, 0, 10000, 'init data']
>>> records[-1][-2]  # 从大列表中取出最后的小列表，再把小列表倒数第二项取出
10000
>>> record = ['2019-11-9', 20000, 0, records[-1][-2] + 20000, 'salary']
>>> record
['2019-11-9', 20000, 0, 30000, 'salary']
>>> records.append(record)
>>> records
[['2019-11-08', 0, 0, 10000, 'init data'], ['2019-11-9', 20000, 0, 30000, 'salary']]
```