Python 系统管理

文件系统

Python的`os`模块实现了操作文件系统的接口。这些操作包括遍历目录树，删除/重命名文件等。此外`os.path`模块可以实现一些针对路径名的操作。

`os`模块的函数

文件处理

- `remove()` / `unlink()` 删除文件
- `rename()` / `renames()` 重命名文件
- `stat()` 返回文件信息
- `symlink()` 创建符号链接
- `utime()` 更新时间戳
- `walk()` 生成一个目录树下的所有文件名

目录/文件夹

- `mkdir()` / `mkdirs()` 创建目录/创建多层目录
- `rmdir()` / `removedirs()` 删除目录/删除多层目录
- `listdir()` 列出指定目录的文件
- `chdir()` / `fcdir()` 改变当前工作目录 / 通过一个文件描述符改变当前目录
- `chroot()` 改变当前进程的根目录
- `getcwd()` / `getcwdu()` 返回当前工作目录 / 功能相同，但返回Unicode对象

访问/权限

- `access()` 检验权限模式
- `chmod()` 改变权限模式
- `chown()` / `lchown()` 改变owner和group ID / 功能相同，但不会跟踪链接
- `umask()` 设置默认权限模式

`os.path`模块中的路径名访问函数

分隔

- `basename()` 去掉目录路径，返回文件名
- `dirname()` 去掉文件名，返回目录路径
- `join()` 将分隔的部分组合成路径
- `split()` 返回`(dirname(), basename())`元组
- `splitdrive()` 返回`(drivename, pathname)`元组
- `splitext()` 返回`(filename, extension)`元组

信息

- `getatime()` 返回最近访问时间
- `getctime()` 返回文件创建时间
- `getmtime()` 返回最近文件修改时间
- `getsize()` 返回文件大小

查询

- `exists()` 指定路径（文件或者目录）是否存在
- `isabs()` 指定路径是否为绝对路径
- `isdir()` 指定路径是否存在且是一个目录
- `isfile()` 指定路径是否存在且是一个文件
- `islink()` 指定路径是否存在且是一个符号链接
- `ismount()` 指定路径是否存在且是一个挂载点

- `samefile()` 两个路径名是否指向同一个文件

```
import os
os.path.exists('10-system-management.ipynb')  # True
os.path.isfile('10-system-management.ipynb')  # True
os.path.isdir('10-system-management.ipynb')  # False
os.path.isabs('10-system-management.ipynb')  # False
```

```
list(os.walk('.'))
```

使用`glob`模块列出匹配文件

`glob.glob()`函数会使用Unix shell的规则来匹配文件或者目录：

- `*` 匹配任意名称（`re`中是`.*`）
- `?` 匹配一个字符
- `[abc]` 匹配字符`a`、`b`和`c`
- `[!abc]` 匹配出了`a`、`b`和`c`之外所有字符

```
import glob

glob.glob('*.ipynb')
```

日期和时间

`datetime`模块

其定义了4个主要的对象，每个对象处理的内容：

- `date` 处理年、月、日
- `time` 处理时、分、秒和微秒
- `datetime` 处理日期和时间同时出现的情况
- `timedelta` 处理日期和（或）时间间隔

```
from datetime import date


halloween = date(2017, 4, 21)
halloween
print(halloween.day, halloween.month, halloween.year)
halloween.isoformat()
```

iso是指ISO 8601，一种表示日期和时间的国际标准。这个标准的显示顺序是从一般（年）到特殊（日）。其可用来对日期进行正确的排序：先按照年，然后是月，最后是日。

```
now = date.today()
now
```

```
from datetime import timedelta

one_day = timedelta(days=1)
tomorrow = now + one_day
print(tomorrow)
print(now + 17 * one_day)
yesterday = now - one_day
print(yesterday)

from datetime import datetime
print(repr(datetime.resolution))
```

date的范围是`date.min`到`date.max`。

```
print(date.min)
print(date.max)
```

`datetime`模块中的`time`对象用来表示一天中的时间：

```
from datetime import time

noon = time(12, 0, 0)
print(noon)
print(noon.hour, noon.minute, noon.second, sep=':')
print(noon.microsecond)
```

参数的顺序按照时间单位从大到小排列（时、分、秒、微秒）。没有参数的话，`time`会默认使用0。

注意，时间不一定时精确的，对于**微秒**和**秒**。

```
from datetime import datetime

def print_repr(obj):
    print(repr(obj))

some_day = datetime(2017, 4, 21, 2, 43, 50, 7)
print_repr(some_day.isoformat())

right_now = datetime.now()
print_repr(right_now)

from datetime import time, date
noon = time(12)
this_day = date.today()
noon_today = datetime.combine(this_day, noon)
print_repr(noon_today)

print_repr(noon_today.date())
print_repr(noon_today.time())
```

下面的代码展示计算一个月份的开始日到结束日中间的日期范围：

```
from datetime import datetime, date, timedelta
import calendar


def get_month_range(start_date=None):
    if start_date is None:
        start_date = date.today().replace(day=1)
    _, days_in_month = calendar.monthrange(start_date.year, start_date.month)
    end_date = start_date + timedelta(days=days_in_month)
    return (start_date, end_date)


a_day = timedelta(days=1)
first_day, last_day = get_month_range()
while first_day < last_day:
    print(first_day)
    first_day += a_day
```

上面的`get_month_range()`函数接受一个`datetime`对象并返回一个由当前月份开始日和下个月开始日组成的元组对象。

计算出一个对应月份第一天的日期，一种快速的方法就是使用`date`或`datetime`对象的`replace()`方法简单地将`days`属性设置成`1`即可。

使用`calendar.monthrange()`来获得该月的总天数。任何时候只要你想获得日历信息，可以使用`calendar`模块。

`time`模块

一种表示绝对时间的方法时计算从某个起始点开始的秒数。Unix时间使用从1970年1月1日0点开始的秒数。这个值通常被成为**纪元**（epoch），它是不同系统之间最简单的交换日期时间的方法。

```
import time

# time() 返回当前时间的纪元值
now = time.time()
print_repr(now)

# ctime() 将纪元值转换成一个字符串
print_repr(time.ctime(now))

# localtime() 返回当前系统时区下的时间
print_repr(time.localtime(now))

# gmtime() 返回UTC时间
print_repr(time.gmtime(now))

print_repr(time.localtime())
print_repr(time.gmtime())

# mktime() 将 struct_time 对象转换回纪元值
print_repr(time.mktime(time.localtime()))
```

`localtime()`和`gmttime()`返回的是一个`struct_time`对象（命名元组）。其结构如下：

Index	Attribute	Values
0	tm_year	(for example, 1993)
1	tm_mon	range [1, 12]
2	tm_mday	range [1, 31]
3	tm_hour	range [0, 23]
4	tm_min	range [0, 59]
5	tm_sec	range [0, 61];
6	tm_wday	range [0, 6], Monday is 0
7	tm_yday	range [1, 366]
8	tm_isdst	0, 1 or -1;
N/A	tm_zone	abbreviation of timezone name
N/A	tm_gmtoff	offset east of UTC in seconds

建议：

- 尽量多使用UTC来代替时区，特别是将服务器设置为UTC时间，不要使用本地时间。
- 有可能的话绝对不使用夏时制时间。

读写日期和时间

使用`strftime()`将日期和时间转换成字符串，`datetime`、`date`、`time`对象和`time`模块中都包含此方法。`strftime()`使用格式化字符串来指定输出，见下表：

格式化字符串	日期/时间单元	范围
Y	年	1900-...
m	月	01-12
B	月名	January,...
b	月名简写	Jan,...
d	日	01-31
A	星期	Sunday,...
a	星期缩写	Sun,...
H	时（24小时制）	00-23
I	时（12小时制）	01-12
p	上午/下午	AM,PM
M	分	00-59
S	秒	00-59

数字左侧都是补零。更多内容请参考官方文档。

```
import time

fmt = "It's %A, %B %d, %Y, local time %I:%M:%S%p"
t = time.localtime()
print_repr(t)
print(time.strftime(fmt, t))


from datetime import date

some_day = date(2017, 4, 21)
print(some_day.strftime(fmt))  # 只能获取日期部分，时间默认是午夜


from datetime import time

some_time = time(10, 35)
print(some_time.strftime(fmt))  # 只会转换时间部分
```

使用`strptime()`可以将格式化的字符串转换为日期或时间。不能使用正则表达式，字符串的非格式化部分必须完全匹配。

```
import time

fmt = '%Y-%m-%d'
print_repr(time.strptime('2017-04-21', fmt))
print_repr(time.strptime('2017-04-31', fmt))  # ValueError
```

名称可以通过操作系统中的`locale`进行设置。如果要打印不同的月和日名称，可通过`setlocale()`来设置，其第一个参数是`locale.LC_TIME`，表示设置的是日期和时间，第二个参数是一个结合了**语言**和**国家名称**的缩写字符串。

```
import locale
help(locale.setlocale)
```

```
import locale
from datetime import date

halloween = date(2014, 10, 31)
for lang_country in ['en_us', 'fr_fr', 'de_de', 'zh_cn']:
    locale.setlocale(locale.LC_TIME, lang_country)
    print(halloween.strftime('%A, %B %d'))
```

```
import locale
names = locale.locale_alias.keys()
good_names = [name for name in names
              if len(name) == 5 and name[2] == '_']
for name in list(good_names)[-5:]:
    print(name)

zh = [name for name in good_names if name.startswith('zh')]
print_repr(zh)
```

其他操作日期和时间的类库

- arrow：更好的 Python 日期时间操作类库。
- maya：Timestamps for humans.
- Chronyk：Python 3 的类库，用于解析手写格式的时间和日期。
- dateutil：Python datetime 模块的扩展。
- delorean：解决 Python 中有关日期处理的棘手问题的库。
- moment：一个用来处理时间和日期的Python库。灵感来自于Moment.js。
- PyTime：一个简单易用的Python模块，用于通过字符串来操作日期/时间。
- pytz：现代以及历史版本的世界时区定义。将时区数据库引入Python。
- when.py：提供用户友好的函数来帮助用户进行常用的日期和时间操作。