**Python 日期和时间**

Python 程序能用很多方式处理日期和时间，转换日期格式是一个常见的功能。

Python 提供了一个 time 和 calendar 模块可以用于格式化日期和时间。

时间间隔是以秒为单位的浮点小数。

每个时间戳都以自从1970年1月1日午夜（历元）经过了多长时间来表示。

Python 的 time 模块下有很多函数可以转换常见日期格式。如函数time.time()用于获取当前时间戳, 如下实例:

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

import time; # 引入time模块

ticks = time.time()

print "当前时间戳为:", ticks

以上实例输出结果：

当前时间戳为: 1459994552.51

时间戳单位最适于做日期运算。但是1970年之前的日期就无法以此表示了。太遥远的日期也不行，UNIX和Windows只支持到2038年。

**什么是时间元组？**

很多Python函数用一个元组装起来的9组数字处理时间:

|  |  |  |
| --- | --- | --- |
| **序号** | **字段** | **值** |
| 0 | 4位数年 | 2008 |
| 1 | 月 | 1 到 12 |
| 2 | 日 | 1到31 |
| 3 | 小时 | 0到23 |
| 4 | 分钟 | 0到59 |
| 5 | 秒 | 0到61 (60或61 是闰秒) |
| 6 | 一周的第几日 | 0到6 (0是周一) |
| 7 | 一年的第几日 | 1到366 (儒略历) |
| 8 | 夏令时 | -1, 0, 1, -1是决定是否为夏令时的旗帜 |

上述也就是struct\_time元组。这种结构具有如下属性：

|  |  |  |
| --- | --- | --- |
| **序号** | **属性** | **值** |
| 0 | tm\_year | 2008 |
| 1 | tm\_mon | 1 到 12 |
| 2 | tm\_mday | 1 到 31 |
| 3 | tm\_hour | 0 到 23 |
| 4 | tm\_min | 0 到 59 |
| 5 | tm\_sec | 0 到 61 (60或61 是闰秒) |
| 6 | tm\_wday | 0到6 (0是周一) |
| 7 | tm\_yday | 1 到 366(儒略历) |
| 8 | tm\_isdst | -1, 0, 1, -1是决定是否为夏令时的旗帜 |

**获取当前时间**

从返回浮点数的时间辍方式向时间元组转换，只要将浮点数传递给如localtime之类的函数。

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

import time

localtime = time.localtime(time.time())

print "本地时间为 :", localtime

以上实例输出结果：

本地时间为 : time.struct\_time(tm\_year=2016, tm\_mon=4, tm\_mday=7, tm\_hour=10, tm\_min=3, tm\_sec=27, tm\_wday=3, tm\_yday=98, tm\_isdst=0)

**获取格式化的时间**

你可以根据需求选取各种格式，但是最简单的获取可读的时间模式的函数是asctime():

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

import time

localtime = time.asctime( time.localtime(time.time()) )

print "本地时间为 :", localtime

以上实例输出结果：

本地时间为 : Thu Apr 7 10:05:21 2016

**格式化日期**

我们可以使用 time 模块的 strftime 方法来格式化日期，：

time.strftime(format[, t])

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

import time

# 格式化成2016-03-20 11:45:39形式

print time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())

# 格式化成Sat Mar 28 22:24:24 2016形式

print time.strftime("%a %b %d %H:%M:%S %Y", time.localtime())

# 将格式字符串转换为时间戳

a = "Sat Mar 28 22:24:24 2016"

print time.mktime(time.strptime(a,"%a %b %d %H:%M:%S %Y"))

以上实例输出结果：

2016-04-07 10:25:09

Thu Apr 07 10:25:09 2016

1459175064.0

python中时间日期格式化符号：

* %y 两位数的年份表示（00-99）
* %Y 四位数的年份表示（000-9999）
* %m 月份（01-12）
* %d 月内中的一天（0-31）
* %H 24小时制小时数（0-23）
* %I 12小时制小时数（01-12）
* %M 分钟数（00=59）
* %S 秒（00-59）
* %a 本地简化星期名称
* %A 本地完整星期名称
* %b 本地简化的月份名称
* %B 本地完整的月份名称
* %c 本地相应的日期表示和时间表示
* %j 年内的一天（001-366）
* %p 本地A.M.或P.M.的等价符
* %U 一年中的星期数（00-53）星期天为星期的开始
* %w 星期（0-6），星期天为星期的开始
* %W 一年中的星期数（00-53）星期一为星期的开始
* %x 本地相应的日期表示
* %X 本地相应的时间表示
* %Z 当前时区的名称
* %% %号本身

**获取某月日历**

Calendar模块有很广泛的方法用来处理年历和月历，例如打印某月的月历：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

import calendar

cal = calendar.month(2016, 1)

print "以下输出2016年1月份的日历:"

print cal;

以上实例输出结果：

以下输出2016年1月份的日历:

January 2016

Mo Tu We Th Fr Sa Su

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30 31

**Time 模块**

Time 模块包含了以下内置函数，既有时间处理相的，也有转换时间格式的：

|  |  |
| --- | --- |
| **序号** | **函数及描述** |
| 1 | [time.altzone](http://www.runoob.com/python/att-time-altzone.html) 返回格林威治西部的夏令时地区的偏移秒数。如果该地区在格林威治东部会返回负值（如西欧，包括英国）。对夏令时启用地区才能使用。 |
| 2 | [time.asctime([tupletime])](http://www.runoob.com/python/att-time-asctime.html) 接受时间元组并返回一个可读的形式为"Tue Dec 11 18:07:14 2008"（2008年12月11日 周二18时07分14秒）的24个字符的字符串。 |
| 3 | [time.clock( )](http://www.runoob.com/python/att-time-clock.html) 用以浮点数计算的秒数返回当前的CPU时间。用来衡量不同程序的耗时，比time.time()更有用。 |
| 4 | [time.ctime([secs])](http://www.runoob.com/python/att-time-ctime.html) 作用相当于asctime(localtime(secs))，未给参数相当于asctime() |
| 5 | [time.gmtime([secs])](http://www.runoob.com/python/att-time-gmtime.html) 接收时间辍（1970纪元后经过的浮点秒数）并返回格林威治天文时间下的时间元组t。注：t.tm\_isdst始终为0 |
| 6 | [time.localtime([secs])](http://www.runoob.com/python/att-time-localtime.html) 接收时间辍（1970纪元后经过的浮点秒数）并返回当地时间下的时间元组t（t.tm\_isdst可取0或1，取决于当地当时是不是夏令时）。 |
| 7 | [time.mktime(tupletime)](http://www.runoob.com/python/att-time-mktime.html) 接受时间元组并返回时间辍（1970纪元后经过的浮点秒数）。 |
| 8 | [time.sleep(secs)](http://www.runoob.com/python/att-time-sleep.html) 推迟调用线程的运行，secs指秒数。 |
| 9 | [time.strftime(fmt[,tupletime])](http://www.runoob.com/python/att-time-strftime.html) 接收以时间元组，并返回以可读字符串表示的当地时间，格式由fmt决定。 |
| 10 | [time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')](http://www.runoob.com/python/att-time-strptime.html) 根据fmt的格式把一个时间字符串解析为时间元组。 |
| 11 | [time.time( )](http://www.runoob.com/python/att-time-time.html) 返回当前时间的时间戳（1970纪元后经过的浮点秒数）。 |
| 12 | [time.tzset()](http://www.runoob.com/python/att-time-tzset.html) 根据环境变量TZ重新初始化时间相关设置。 |

Time模块包含了以下2个非常重要的属性：

|  |  |
| --- | --- |
| **序号** | **属性及描述** |
| 1 | **time.timezone** 属性time.timezone是当地时区（未启动夏令时）距离格林威治的偏移秒数（>0，美洲;<=0大部分欧洲，亚洲，非洲）。 |
| 2 | **time.tzname** 属性time.tzname包含一对根据情况的不同而不同的字符串，分别是带夏令时的本地时区名称，和不带的。 |

**日历（Calendar）模块**

此模块的函数都是日历相关的，例如打印某月的字符月历。

星期一是默认的每周第一天，星期天是默认的最后一天。更改设置需调用calendar.setfirstweekday()函数。模块包含了以下内置函数：

|  |  |
| --- | --- |
| **序号** | **函数及描述** |
| 1 | **calendar.calendar(year,w=2,l=1,c=6)** 返回一个多行字符串格式的year年年历，3个月一行，间隔距离为c。 每日宽度间隔为w字符。每行长度为21\* W+18+2\* C。l是每星期行数。 |
| 2 | **calendar.firstweekday( )** 返回当前每周起始日期的设置。默认情况下，首次载入caendar模块时返回0，即星期一。 |
| 3 | **calendar.isleap(year)** 是闰年返回True，否则为false。 |
| 4 | **calendar.leapdays(y1,y2)** 返回在Y1，Y2两年之间的闰年总数。 |
| 5 | **calendar.month(year,month,w=2,l=1)** 返回一个多行字符串格式的year年month月日历，两行标题，一周一行。每日宽度间隔为w字符。每行的长度为7\* w+6。l是每星期的行数。 |
| 6 | **calendar.monthcalendar(year,month)** 返回一个整数的单层嵌套列表。每个子列表装载代表一个星期的整数。Year年month月外的日期都设为0;范围内的日子都由该月第几日表示，从1开始。 |
| 7 | **calendar.monthrange(year,month)** 返回两个整数。第一个是该月的星期几的日期码，第二个是该月的日期码。日从0（星期一）到6（星期日）;月从1到12。 |
| 8 | **calendar.prcal(year,w=2,l=1,c=6)** 相当于 print calendar.calendar(year,w,l,c). |
| 9 | **calendar.prmonth(year,month,w=2,l=1)** 相当于 print calendar.calendar（year，w，l，c）。 |
| 10 | **calendar.setfirstweekday(weekday)** 设置每周的起始日期码。0（星期一）到6（星期日）。 |
| 11 | **calendar.timegm(tupletime)** 和time.gmtime相反：接受一个时间元组形式，返回该时刻的时间辍（1970纪元后经过的浮点秒数）。 |
| 12 | **calendar.weekday(year,month,day)** 返回给定日期的日期码。0（星期一）到6（星期日）。月份为 1（一月） 到 12（12月）。 |