# Python中time模块详解

在我们平常的代码中，经常需要和时间打交道。在Python中，与时间处理相关的模块有：**time、datetime以及calendar**。

1. 在Python中，通常有这几种方式来表示时间：**1）时间戳 2）格式化的时间字符串 3）元组（struct\_time）共九个元素**。
2. UTC（Coordinated Universal Time，世界协调时）亦即格林威治天文时间，世界标准时间。在中国为UTC+8。DST（Daylight Saving Time）即**夏令时**。
3. 时间戳（timestamp）的方式：通常来说，**时间戳表示的是从1970年1月1日00:00:00开始按秒计算的偏移量**。我们运行“type(time.time())”，返回的是float类型。返回时间戳方式的函数主要有time()，clock()等。
4. 元组（struct\_time）方式：struct\_time元组共有9个元素，返回struct\_time的函数主要有**gmtime()，localtime()，strptime()。**下面列出这种方式元组中的几个元素：

在开始前，先说明几点：

## struct\_time 共九种元素

**struct\_time 共九种元素**

>>> import time

>>> ls = time.localtime()

>>> ls

time.struct\_time(**tm\_year=2015, tm\_mon=8, tm\_mday=24, tm\_hour=9, tm\_min=39, tm\_sec=38, tm\_wday=0, tm\_yday=236, tm\_isdst=0**)

**直接使用元组索引**获取对应项的值：

>>> ls[0]

2015

>>> ls[1]

8

>>> ls[-1]

0

或者是**使用成员符号调用**：

>>> ls.tm\_year

2015

既然是一个元组，那么就遵循元组的所有特性，比如索引（都从0开始），切片等。下面元组中各元素的解释：

**tm\_year ：年**

**tm\_mon ：月（1-12）**

**tm\_mday ：日（1-31）**

**tm\_hour ：时（0-23）**

**tm\_min ：分（0-59）**

**tm\_sec ：秒（0-59）**

**tm\_wday ：星期几（0-6,0表示周日）**

**tm\_yday ：一年中的第几天（1-366）**

**tm\_isdst ：是否是夏令时（默认为-1）**

## time模块中常用的函数

使用该模块中的函数时，必须先引入该模块（import time）

### \*\* time.time()：返回当前时间的时间戳

>>> time.time()

1440337405.85

#对时间戳取整

>>> int(time.time())

1440746424

### \*\* time.localtime( [secs] ) 取当前时区的struct\_time

将一个时间戳转换为当前时区的struct\_time，即时间数组格式的时间

参数：

sec – 转换为time.struct\_time类型的对象的秒数

如果secs参数未提供，则以当前时间为准（即会默认调用time.time()）。

未给定参数

>>> time.localtime()

time.struct\_time(tm\_year=2015, tm\_mon=8, tm\_mday=23, tm\_hour=21, tm\_min=49, tm\_sec=27, tm\_wday=6, tm\_yday=235, tm\_isdst=0)

给定参数

>>> time.localtime(1440337405.85)

time.struct\_time(tm\_year=2015, tm\_mon=8, tm\_mday=23, tm\_hour=21, tm\_min=43, tm\_sec=25, tm\_wday=6, tm\_yday=235, tm\_isdst=0)

### 3、time.gmtime([secs]) 将一个时间戳转换为UTC时区的struct\_time

将一个时间戳转换为UTC时区的struct\_time

time.gmtime() 函数将一个时间戳转换为UTC时区（0时区）的struct\_time，可选的参数sec表示从1970-1-1 00:00:00以来的秒数。其默认值为time.time()，函数返回time.struct\_time类型的对象。（struct\_time是在time模块中定义的表示时间的对象）。

如果secs参数未提供，则以当前时间为准。

参数：

sec – 转换为time.struct\_time类型的对象的秒数

>>> time.gmtime()

time.struct\_time(tm\_year=2015, tm\_mon=8, tm\_mday=23, tm\_hour=13, tm\_min=55, tm\_sec=43, tm\_wday=6, tm\_yday=235, tm\_isdst=0)

>>> time.gmtime(1440337405.85)

time.struct\_time(tm\_year=2015, tm\_mon=8, tm\_mday=23, tm\_hour=13, tm\_min=43, tm\_sec=25, tm\_wday=6, tm\_yday=235, tm\_isdst=0)

### 4、time.mktime(t)：将一个struct\_time转化为时间戳

time.mktime() 函数执行与gmtime(), localtime()相反的操作，它接收struct\_time对象作为参数,返回用秒数表示时间的浮点数。

如果输入的值不是一个合法的时间，将触发 OverflowError 或 ValueError。

参数：

t – 结构化的时间或者完整的9位元组元素

>>> time.mktime(time.localtime())

1440338541.0

### 5、\*\* time.sleep(secs)：线程推迟指定的时间运行

线程睡眠指定时间，单位为妙。

睡眠10秒

>>> time.sleep(10)

### 6、\*\* time.clock() 返回当前的CPU时间

这个函数， 函数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。在不同的系统上含义不同。在NUix系统上，它返回的是“进程时间”，它是用妙表示的浮点数（时间戳）。而在Windows中，第一次调用，返回的是进程运行时实际时间。而第二次之后的调用是自第一次调用以后到现在的运行时间。

返回值

**该函数有两个功能：**

（1）在第一次调用的时候，返回的是程序运行的实际时间；

（2）第二次之后的调用，返回的是自第一次调用后,到这次调用的时间间隔在win32系统下，这个函数返回的是真实时间（wall time），而在Unix/Linux下返回的是CPU时间。

实例：

#!/usr/bin/python

#coding=utf-8

import time

def procedure() :

time.sleep(3)

time1 = time.clock()

procedure()

print time.clock() - time1, "seconds process time!"

time2 = time.time()

procedure()

print time.time() - time2, "seconds wall time!"

执行结果：

2.99947838799 seconds process time!

3.0 seconds wall time!

### 7、time.asctime( [t] )

把一个表示时间的元组或者struct\_time表示为 ‘Sun Aug 23 14:31:59 2015’ 这种形式。如果没有给参数，会将time.localtime()作为参数传入。

参数：

t – 9个元素的元组或者通过函数 gmtime() 或 localtime() 返回的时间值

>>> time.asctime(time.gmtime())

'Sun Aug 23 14:31:59 2015'

### 8、time.ctime([secs])

把一个时间戳（按秒计算的浮点数）转化为time.asctime()的形式。如果为指定参数，将会默认使用time.time()作为参数。它的作用相当于time.asctime(time.localtime(secs))

参数：

sec – 要转换为字符串时间的秒数

>>> time.ctime(1440338541.0)

'Sun Aug 23 22:02:21 2015'

>>> time.ctime()

'Sun Aug 23 22:44:14 2015'

### 9、\*\* time.strftime( format [, t] )

返回字符串表示的当地时间。

把一个代表时间的元组或者struct\_time（如由time.localtime()和time.gmtime()返回）转化为格式化的时间字符串，格式由参数format决定。如果未指定，将传入time.localtime()。如果元组中任何一个元素越界，就会抛出ValueError的异常。函数返回的是一个可读表示的本地时间的字符串。

参数：

**format：格式化字符串**

t ：可选的参数是一个struct\_time对象

时间字符串支持的格式符号：（区分大小写）

%a 本地星期名称的简写（如星期四为Thu）

%A 本地星期名称的全称（如星期四为Thursday）

%b 本地月份名称的简写（如八月份为agu）

%B 本地月份名称的全称（如八月份为august）

%c 本地相应的日期和时间的字符串表示（如：15/08/27 10:20:06）

%d 一个月中的第几天（01 - 31）

%f 微妙（范围0.999999）

%H 一天中的第几个小时（24小时制，00 - 23）

%I 第几个小时（12小时制，0 - 11）

%j 一年中的第几天（001 - 366）

%m 月份（01 - 12）

%M 分钟数（00 - 59）

%p 本地am或者pm的相应符

%S 秒（00 - 61）

%U 一年中的星期数。（00 - 53星期天是一个星期的开始。）第一个星期天之 前的所有天数都放在第0周。

%w 一个星期中的第几天（0 - 6，0是星期天）

%W 和%U基本相同，不同的是%W以星期一为一个星期的开始。

%x 本地相应日期字符串（如15/08/01）

%X 本地相应时间字符串（如08:08:10）

%y 去掉世纪的年份（00 - 99）两个数字表示的年份

%Y 完整的年份（4个数字表示年份）

%z 与UTC时间的间隔（如果是本地时间，返回空字符串）

%Z 时区的名字（如果是本地时间，返回空字符串）

%% ‘%’字符

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

实例：

>>> formattime

time.struct\_time(tm\_year=2015, tm\_mon=8, tm\_mday=24, tm\_hour=13, tm\_min=1, tm\_sec=30, tm\_wday=0, tm\_yday=236, tm\_isdst=-1)

1

2

通过函数将struct\_time转成格式字符串

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

'2015-08-24 13:01:30'

1

2

也可以直接使用字符串拼接成格式时间字符串

>>> print str(formattime.tm\_year) + "年" + str(formattime.tm\_mon) + "月" + str(formattime.tm\_mday) + "日"

2015年8月24日

1

2

将当前时间的时间戳转换成想要的时间格式字符串

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

2015/08/28 12:25:25

1

2

3

获取当前时间的时分秒

>>> print time.strftime('%H:%M:%S')

18:46:42

1

2

获取当前时间的年月日

>>> print time.strftime('%y-%m-%d')

15-08-27

1

2

格式符号实例：

>>> t = datetime.datetime.now()

>>> t

datetime.datetime(2015, 8, 27, 18, 46, 42, 605000)

1

2

3

获取当前时间的格式化时间

>>> print t.strftime('%Y-%m-%d %H:%M:%S %f')

2015-08-27 18:46:42 605000

>>> print t.strftime('%y-%m-%d %I:%M:%S %p')

15-08-27 06:46:42 PM

1

2

3

4

星期的简写

>>> print t.strftime('%a')

Thu

1

2

星期的全写

>>> print t.strftime('%A')

Thursday

1

2

月份的简写

>>> print t.strftime('%b')

Aug

1

2

月份的全写

>>> print t.strftime('%B')

August

1

2

得到日期时间的字符串

>>> print t.strftime('%c')

08/27/15 18:46:42

1

2

日期字符串

>>> print t.strftime('%x')

08/27/15

1

2

时间字符串

>>> print t.strftime('%X')

18:46:42

1

2

今天在这周是星期几

>>> print t.strftime('%w')

4

1

2

今天是今年的第几天

>>> print t.strftime('%j')

239

1

2

这周是今年的第几周

>>> print t.strftime('%U')

34

1

2

### 10、time.strptime(string[,format])

将格式字符串转化成struct\_time.

该函数是time.strftime()函数的逆操作。time strptime() 函数根据指定的格式把一个时间字符串解析为时间元组。所以函数返回的是struct\_time对象。

参数：

string ：时间字符串

format：格式化字符串

创建一个时间字符串变量stime

>>> stime = "2015-08-24 13:01:30"

1

通过strptime()函数将stime转化成strcut\_time形式

>>> formattime = time.strptime(stime,"%Y-%m-%d %H:%M:%S")

>>> print formattime

time.struct\_time(tm\_year=2015, tm\_mon=8, tm\_mday=24, tm\_hour=13, tm\_min=1, tm\_sec=30, tm\_wday=0, tm\_yday=236, tm\_isdst=-1)

1

2

3

遍历返回的时间元组序列

>>> for i in formattime :

... print i,

...

2015 8 24 13 1 30 0 236 -1

1

2

3

4

注意在使用strptime()函数将一个指定格式的时间字符串转化成元组时，参数format的格式必须和string的格式保持一致，如果string中日期间使用“-”分隔，format中也必须使用“-”分隔，时间中使用冒号“:”分隔，后面也必须使用冒号分隔，否则会报格式不匹配的错误。

**date类**

date类包含三个参数，分别为year，month，day，返回格式为year-month-day。

**构造函数**

1. \_\_new\_\_(year,month,day)：默认的构造函数，创建date类的对象时直接传入year，month，day三个参数即可返回对应的日期。
2. fromtimestamp(t)：使用时间戳构造对象，使用方法为：datetime.date.fromtimestamp(t)，传入参数t为一个时间戳，返回时间戳t对应的日期。
3. today()：使用今天的日期构造对象，使用方法为：datetime.date.today()，无参数，返回今天的日期。
4. fromordinal(n)：使用日期序数构造对象，使用方法为：datetime.date.fromordinal(n)，传入参数为一个整数序数，代表从公元1年1月1日开始的序数，序数每增加1代表增加1天，返回最终计算出的日期。

**方法**

1. timetuple()：返回日期对应的time.struct\_time对象，格式为time.struct\_time(tm\_year=1, tm\_mon=1, tm\_mday=2, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=1, tm\_yday=2, tm\_isdst=-1) 。
2. toordinal()：相当于fromordinal(n)的逆过程，返回值即为fromordinal(n)中的日期序数n。
3. weekday()：返回该日期对应星期几，用[0,6]代表星期一到星期日。
4. isoweekday()：作用同weekday()，用[1,7]代表星期一到星期日。
5. isocalendar()：返回一个三元组，格式为(year,week\_number,weekday)，分别代表年，第几周，星期几。
6. isoformat()：返回标准日期格式：YYYY-MM-DD。
7. ctime()：返回格式为：Sat Sep 8 00:00:00 2018
8. strftime(format)：把日期按照format指定的格式进行格式化，具体的格式化符号如下。
9. replace(year,month,day)：传入参数为year，month，day，返回对应的新日期。

# calendar模块用法

## calendar模块用法

## calendar.calendar(year,w=2,l=1,c=6) 返回一个多行字符串格式

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。

## 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。

## calendar.isleap(year)

是闰年返回True，否则为false。

## calendar.leapdays(y1,y2)

返回在Y1，Y2两年之间的闰年总数。

## 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

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。

## 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。

## 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。

calendar.prcal(year,w=2,l=1,c=6)

相当于 print calendar.calendar(year,w,l,c).

calendar.prmonth(year,month,w=2,l=1)

相当于 print calendar.calendar（year，w，l，c）。

calendar.setfirstweekday(weekday)

设置每周的起始日期码。0（星期一）到6（星期日）。

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）。

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）。

下面是一个关于calendar模块用法的例子：

#python中的calendar

import calendar

#返回指定年的某月

def get\_month(year, month):

    return calendar.month(year, month)

#返回指定年的日历

def get\_calendar(year):

    return calendar.calendar(year)

#判断某一年是否为闰年，如果是，返回True，如果不是，则返回False

def is\_leap(year):

    return calendar.isleap(year)

#返回某个月的weekday的第一天和这个月的所有天数

def get\_month\_range(year, month):

    return calendar.monthrange(year, month)

#返回某个月以每一周为元素的序列

def get\_month\_calendar(year, month):

    return calendar.monthcalendar(year, month)

def main():

    year = 2016

    month = 11

    test\_month = get\_month(year, month)

    print(test\_month)

    print('=' \* 50)

    test\_year = get\_calendar(year)

    print(test\_year)

    print('#' \* 50)

    #print(get\_calendar(year))

    print('{0}这一年是否为闰年？：{1}'.format(year, is\_leap(year)))

    print(get\_month\_range(year, month))

    print(get\_month\_calendar(year, month))

if \_\_name\_\_ == '\_\_main\_\_':

    main()

运行结果如下：

November 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

==================================================

2016

January February March

Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su

1 2 3 1 2 3 4 5 6 7 1 2 3 4 5 6

4 5 6 7 8 9 10 8 9 10 11 12 13 14 7 8 9 10 11 12 13

11 12 13 14 15 16 17 15 16 17 18 19 20 21 14 15 16 17 18 19 20

18 19 20 21 22 23 24 22 23 24 25 26 27 28 21 22 23 24 25 26 27

25 26 27 28 29 30 31 29 28 29 30 31

April May June

Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su

1 2 3 1 1 2 3 4 5

4 5 6 7 8 9 10 2 3 4 5 6 7 8 6 7 8 9 10 11 12

11 12 13 14 15 16 17 9 10 11 12 13 14 15 13 14 15 16 17 18 19

18 19 20 21 22 23 24 16 17 18 19 20 21 22 20 21 22 23 24 25 26

25 26 27 28 29 30 23 24 25 26 27 28 29 27 28 29 30

30 31

July August September

Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su

1 2 3 1 2 3 4 5 6 7 1 2 3 4

4 5 6 7 8 9 10 8 9 10 11 12 13 14 5 6 7 8 9 10 11

11 12 13 14 15 16 17 15 16 17 18 19 20 21 12 13 14 15 16 17 18

18 19 20 21 22 23 24 22 23 24 25 26 27 28 19 20 21 22 23 24 25

25 26 27 28 29 30 31 29 30 31 26 27 28 29 30

October November December

Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su

1 2 1 2 3 4 5 6 1 2 3 4

3 4 5 6 7 8 9 7 8 9 10 11 12 13 5 6 7 8 9 10 11

10 11 12 13 14 15 16 14 15 16 17 18 19 20 12 13 14 15 16 17 18

17 18 19 20 21 22 23 21 22 23 24 25 26 27 19 20 21 22 23 24 25

24 25 26 27 28 29 30 28 29 30 26 27 28 29 30 31

31

##################################################

2016年是否为闰年？True

(1, 30)