Python3 time tzset()方法 Python3 time clock()方法

Python3 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/python3

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: 1459996086.7115328
```

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	一年中的第几天，1 到 366
8	tm_isdst	是否为夏令时，值有：1(夏令时)、0(不是夏令时)、-1(未知)，默认 -1

获取当前时间

从返回浮点数的时间戳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/python3

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=28, tm_sec=49, tm_wd
ay=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/python3

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 10:29:13 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/python3

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 10:29:46
Thu Apr 07 10:29:46 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/python3

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，既有时间处理的，也有转换时间格式的：

序号	函数及描述	实例

1	time.altzone
	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。
		以下实例展示了 altzone()函数的使用方法：
		`>>> import time`
		`>>> print ("time.altzone %d " % time.altzone)`
		`time.altzone -28800`

2	time.asctime([tupletime])
	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。
		以下实例展示了 asctime()函数的使用方法：
		`>>> import time`
		`>>> t = time.localtime()`
		`>>> print ("time.asctime(t): %s " % time.asctime(t))`
		`time.asctime(t): Thu Apr  7 10:36:20 2016`

3	time.clock()
	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。
		实例
		由于该方法依赖操作系统，在 Python 3.3 以后不被推荐，而在 3.8 版本中被移除，需使用下列两个函数替代。
		`time.perf_counter()  # 返回系统运行时间`
		`time.process_time()  # 返回进程运行时间`

4	time.ctime([secs])
	作用相当于asctime(localtime(secs))，未给参数相当于asctime()
		以下实例展示了 ctime()函数的使用方法：
		`>>> import time`
		`>>> print ("time.ctime() : %s" % time.ctime())`
		`time.ctime() : Thu Apr  7 10:51:58 2016`

5	time.gmtime([secs])
	接收时间戳（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
		以下实例展示了 gmtime()函数的使用方法：
		`>>> import time`
		`>>> print ("gmtime :", time.gmtime(1455508609.34375))`
		`gmtime : time.struct_time(tm_year=2016, tm_mon=2, tm_mday=15, tm_hour=3, tm_min=56, tm_sec=49, tm_wday=0, tm_yday=46, tm_isdst=0)`

6	time.localtime([secs]
	接收时间戳（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）。
		以下实例展示了 localtime()函数的使用方法：
		`>>> import time`
		`>>> print ("localtime(): ", time.localtime(1455508609.34375))`
		`localtime():  time.struct_time(tm_year=2016, tm_mon=2, tm_mday=15, tm_hour=11, tm_min=56, tm_sec=49, tm_wday=0, tm_yday=46, tm_isdst=0)`

7	time.mktime(tupletime) 接受时间元组并返回时间戳（1970纪元后经过的浮点秒数）。	实例

8	time.sleep(secs)
	推迟调用线程的运行，secs指秒数。
		以下实例展示了 sleep()函数的使用方法：
		`#!/usr/bin/python3`
		`import time`
		
		`print ("Start : %s" % time.ctime())`
		`time.sleep( 5 )`
		`print ("End : %s" % time.ctime())`

9	time.strftime(fmt[,tupletime])
	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。
		以下实例展示了 strftime()函数的使用方法：
		`>>> import time`
		`>>> print (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))`
		`2016-04-07 11:18:05`

10	time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')	以下实例展示了 strptime()函数的使用方法：

	根据fmt的格式把一个时间字符串解析为时间元组。	`>>> import time`
		`>>> struct_time = time.strptime("30 Nov 00", "%d %b %y")`
		`>>> print ("返回元组: ", struct_time)`
		`返回元组: time.struct_time(tm_year=2000, tm_mon=11, tm_mda`
		`y=30, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=3, tm_yday=33`
		`5, tm_isdst=-1)`

11	time.time( )
	返回当前时间的时间戳（1970纪元后经过的浮点秒数）。
		以下实例展示了 time()函数的使用方法：
		`>>> import time`
		`>>> print(time.time())`
		`1459999336.1963577`

12	time.tzset() 根据环境变量TZ重新初始化时间相关设置。	实例
13	**time.perf_counter()** 返回计时器的精准时间（系统的运行时间），包含整个系统的睡眠时间。由于返回值的基准点是未定义的，所以，只有连续调用的结果之间的差才是有效的。	实例
14	**time.process_time()** 返回当前进程执行 CPU 的时间总和，不包含睡眠时间。由于返回值的基准点是未定义的，所以，只有连续调用的结果之间的差才是有效的。	

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。
	`>>> import calendar`
	`>>> print(calendar.isleap(2000))`
	`True`
	`>>> print(calendar.isleap(1900))`
	`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，第二个是该月有几天。星期几是从0（星期一）到 6（星期日）。
	`>>> import calendar`
	`>>> calendar.monthrange(2014, 11)`
	`(5, 30)`
	(5, 30)解释：5 表示 2014 年 11 月份的第一天是周六，30 表示 2014 年 11 月份总共有 30 天。

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，返回该时刻的时间戳（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）。

其他相关模块和函数

在Python中，其他处理日期和时间的模块还有：

- time 模块
- datetime模块

← Python3 time tzset()方法 Python3 time clock()方法 →


```
import time

scale = 50

print("执行开始".center(scale//2,"-"))  # .center() 控制输出的样式，宽度为 25//2，即 22，汉字居中，两侧填充 -

start = time.perf_counter() # 调用一次 perf_counter()，从计算机系统里随机选一个时间点A，计算其距离当前时间点B1有多少秒。当第二次调用该函数时，默认从第一次调用的时间点A算起，距离当前时间点B2有多少秒。两个函数取差，即实现从时间点B1到B2的计时功能。
for i in range(scale+1):
    a = '*' * i             # i 个长度的 * 符号
    b = '.' * (scale-i)  # scale-i)  个长度的 . 符号。符号 * 和 . 总长度为50
    c = (i/scale)*100  # 显示当前进度，百分之多少
```

```
    dur = time.perf_counter() - start     # 计时，计算进度条走到某一百分比的用时
    print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c,a,b,dur),end='')  # \r用来在每次输出完成后，将光标移至行首，这样保证进度条始终在同一行输出，即在一行不断刷新的效果；{:^3.0f}，输出格式为居中，占3位，小数点后0位，浮点型数，对应输出的数为c；{}，对应输出的数为a；{}，对应输出的数为b；{:.2f}，输出有两位小数的浮点数，对应输出的数为dur；end=''，用来保证不换行，不加这句默认换行。
    time.sleep(0.1)     # 在输出下一个百分之几的进度前，停止0.1秒
print("\n"+"执行结果".center(scale//2,'-'))
```

测试输出：

```
-----------执行开始----------
24 %[************->.....................................]1.24s
```

GaiFan 6个月前 (09-29)

```
#!/usr/bin/python

import time
import calendar


"""
    时间元组（年、月、日、时、分、秒、一周的第几日、一年的第几日、夏令时）
        一周的第几日：0-6
        一年的第几日：1-366
        夏令时：-1, 0, 1
"""

"""
    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  # 乱码
    %% %号本身
"""


# （1）当前时间戳
# 1538271871.226226
time.time()


# （2）时间戳 → 时间元组，默认为当前时间
# time.struct_time(tm_year=2018, tm_mon=9, tm_mday=3, tm_hour=9, tm_min=4, tm_sec=1, tm_w
day=6, tm_yday=246, tm_isdst=0)
time.localtime()
time.localtime(1538271871.226226)


# （3）时间戳 → 可视化时间
# time.ctime(时间戳)，默认为当前时间
time.ctime(1538271871.226226)


# （4）时间元组 → 时间戳
# 1538271871
time.mktime((2018, 9, 30, 9, 44, 31, 6, 273, 0))


# （5）时间元组 → 可视化时间
# time.asctime(时间元组)，默认为当前时间
time.asctime()
time.asctime((2018, 9, 30, 9, 44, 31, 6, 273, 0))
time.asctime(time.localtime(1538271871.226226))


# （6）时间元组 → 可视化时间（定制）
# time.strftime(要转换成的格式，时间元组)
time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())


# （7）可视化时间（定制） → 时间元祖
# time.strptime(时间字符串，时间格式)
print(time.strptime('2018-9-30 11:32:23', '%Y-%m-%d %H:%M:%S'))


# （8）浮点数秒数，用于衡量不同程序的耗时，前后两次调用的时间差
time.clock()
```

爱在旧城窄巷 6个月前 (09-30)