[python中的time模块](https://www.cnblogs.com/renpingsheng/p/6965044.html)

time模块－－时间获取和转换

time模块提供各种时间相关的功能

与时间相关的模块有：time,datetime,calendar

**必要说明：**

这个模块的功能不是适用于所有的平台

这个模块中定义的大部分函数是调用C平台上的同名函数实现

**一些术语和约定的解释：**

1.时间戳(timestamp)的方式：通常来说，时间戳表示的是从1970年1月１日开始按秒计算的偏移量(time.gmtime(0))此模块中的函数无法处理1970纪元年以前的时间或太遥远的未来(处理极限取决于C函数库，对于32位系统而言，是2038年)

2.UTC(Coordinated Universal Time,世界协调时)也叫格林威治天文时间，是世界标准时间．在我国为UTC+8

3.DST(Daylight Saving Time)即夏令时

4.一些实时函数的计算精度可能不同

**时间元祖(time.struct\_time)**

gmtime(),localtime()和strptime()以时间元祖(struct\_time)的形式返回

| **索引值(index)** | **属性(Attribute)** | **值(Values)** |
| --- | --- | --- |
| 0 | tm\_year(年) | (例如:2015) |
| 1 | tm\_mon(月) | 1-12 |
| 2 | tm\_mday(日) | 1-31 |
| 3 | tm\_hour(时) | 0-23 |
| 4 | tm\_min(分) | 0-59 |
| 5 | tm\_sec(秒) | 0-61(60代表闰秒,61是基于历史原因保留) |
| 6 | tm\_wday(星期几) | 0-6(0表示星期一) |
| 7 | tm\_yday(一年中的第几天) | 1-366 |
| 8 | tm\_isdst(是否为夏令时) | 0,1,-1(-1代表夏令时) |

**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

**time.asctime([t])**

接受时间元组并返回一个可读的形式"Tue May 30 17:17:30 2017"(2017年5月30日周二17时17分30秒)的24个字符的字符串

**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

python3.3以后不被推荐使用，该方法依赖操作系统，建议使用per\_counter(返回系统运行时间)或process\_time(返回进程运行时间)代替

**time.ctime([secs])**

作用相当于asctime(localtime(secs)),未给参数相当于asctime()

**time.gmtime([secs])**

接收时间辍(1970纪元年后经过的浮点秒数)并返回格林威治天文时间下的时间元组t(t.tm\_isdst始终为０)

**time.daylight**

如果夏令时被定义，则该值为非零

**time.localtime([secs])**

接收时间辍(1970纪元年后经过的浮点秒数)并返回当地时间下的时间元组t(t.tm\_isdst可取为０或１，取决于当地当时是不是夏令时)

**time.mktime(t)**

接受时间元组并返回时间辍(1970纪元年后经过的浮点秒数)

**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

**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.sleep(secs)**

推迟调用线程的运行，secs的单位是秒

**time.strftime(format[,t])**

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．如果t未指定，将传入time.localtime()，如果元组中任命一个元素越界，将会抛出ValueError异常

format格式如下：

%a 本地(local)简化星期名称

%A 本地完整星期名称

%b 本地简化月份名称

%B 本地完整月份名称

%c 本地相应的日期和时间表示

%d 一个月中的第几天(01-31)

%H 一天中的第几个小时(24小时制，00-23)

%l 一天中的第几个小时(12小时制，01-12)

%j 一年中的第几天(01-366)

%m 月份(01-12)

%M 分钟数(00-59)

%p 本地am或者pm的相应符

%S 秒(01-61)

%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

%X 本地相应时间

%y 去掉世纪的年份(00-99)

%Y 完整的年份

%z 用+HHMM或者-HHMM表示距离格林威治的时区偏移(H代表十进制的小时数，M代表十进制的分钟数)

%Z 时区的名字(如果不存在为空字符)

%% %号本身

%p只有与%I配合使用才有效果

当使用strptime()函数时，只有当在这年中的周数和天数被确定的时候%U和%W才会被计算

**time.strptime(string[,format])**

把一个格式化时间字符串转化为struct\_time,实际上它和strftie()是逆操作

**time.time()**

返回当前时间的时间戳(1970元年后的浮点秒数)

**time.timezone()**

是当地时区(未启动夏令时)距离格林威治的偏移秒数(美洲＞０，欧洲大部分，亚洲，非洲<＝０)

**time.tzname**

包含两个字符串的元组，第一是当地夏令时区的名称，第二是当地的DST时区的名称