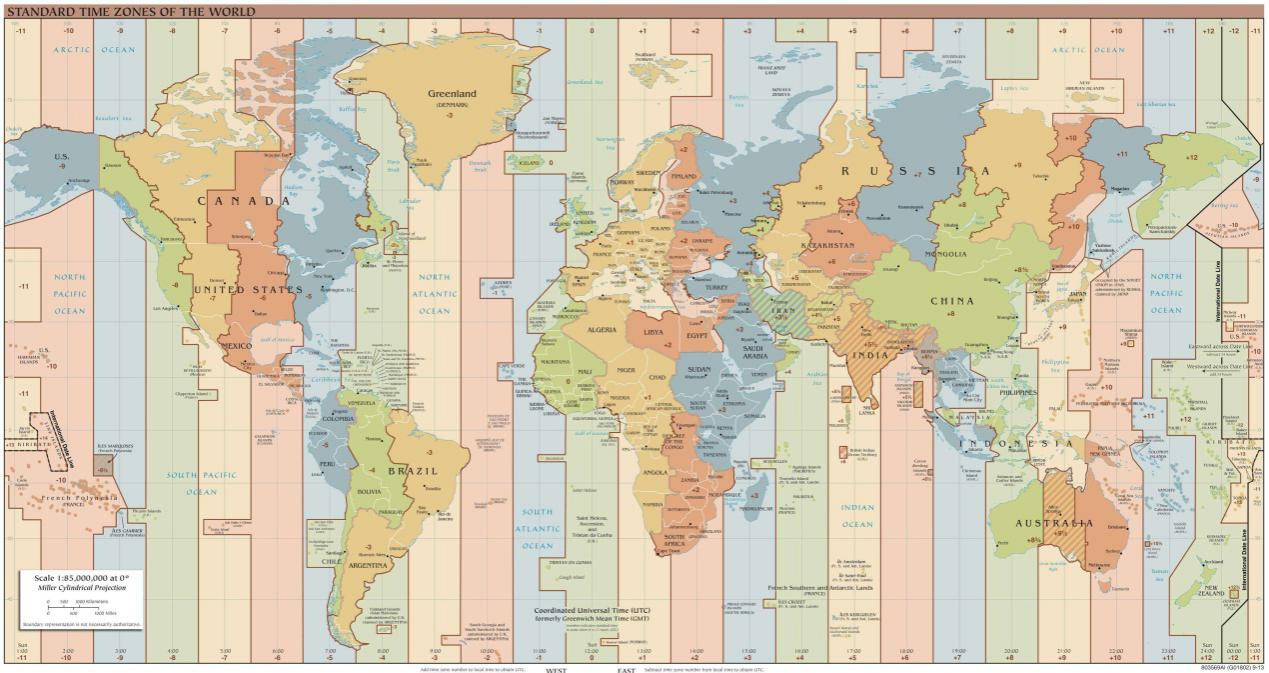
# 日期与时间处理

Python中提供了多个用于对日期和时间进行操作的内置模块：time模块、datetime模块和calendar模块。time模块相比，datetime模块提供的接口更直观、易用，功能也更加强大。

## 时间的相关术语

**UTC time** Coordinated Universal Time，世界协调时，又称**格林尼治天文时间**、世界标准时间。与UTC time对应的是各个时区的local time，东N区的时间比UTC时间早N个小时，因此UTC time + N小时 即为东N区的本地时间；而西N区时间比UTC时间晚N个小时，即 UTC time - N小时 即为西N区的本地时间； 中国在东8区，因此比UTC时间早8小时，可以以UTC+8进行表示。



<http://zh.thetimenow.com/gmt/greenwich_mean_time>

**timestamp（时间戳）** 也称为Unix时间 或 POSIX时间；它是一种时间表示方式，表示从格林尼治时间1970年1月1日0时0分0秒开始到现在所经过的毫秒数，其值为float类型。时间戳是个差值，其值与时区无关。

## 2、time模块

time模块是通过调用C库实现不变的，所以有些方法在某些平台上可能无法调用，但是其提供的大部分接口与C标h准库time.h基本一致。

>>> time.time()

1528444335.2044096

小数点前面的是总的秒数，后面是毫秒信息。

获取现在计算机的当地(本地)时间和世界标准时间。

>>>time.localtime()

time.struct\_time(tm\_year=2017, tm\_mon=2, tm\_mday=4, tm\_hour=14, tm\_min=2, tm\_sec=34, tm\_wday=5, tm\_yday=35, tm\_isdst=0)

>>>time.gmtime()

time.struct\_time(tm\_year=2017, tm\_mon=2, tm\_mday=4, tm\_hour=6, tm\_min=2, tm\_sec=56, tm\_wday=5, tm\_yday=35, tm\_isdst=0)

>>>type(time.localtime())

<class 'time.struct\_time'>

字符串转化为时间戳：

>>>time.strftime('%Y-%m-%d %H:%M', time.localtime())

'2017-02-04 14:19'

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

1486189282.0

## 2、datetime模块

datetime模块定义了以下几个类：

| **类名称** | **描述** |
| --- | --- |
| datetime.date | 表示日期，常用的属性有：year, month和day |
| datetime.time | 表示时间，常用属性有：hour, minute, second, microsecond |
| datetime.datetime | 表示日期时间 |
| datetime.timedelta | 表示两个date、time、datetime实例之间的时间间隔，分辨率（最小单位）可达到微秒 |
| datetime.tzinfo | 时区相关信息对象的抽象基类。它们由datetime和time类使用，以提供自定义时间的而调整。 |
| datetime.timezone | Python 3.2中新增的功能，实现tzinfo抽象基类的类，表示与UTC的固定偏移量 |

类之间的关系：

object

date

datetime

time

timedelta

tzinfo

timezone

### datetime.datetime类

class datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None)

各参数的取值范围为：

| **参数名称** | **取值范围** |
| --- | --- |
| Year | [0, 9999] |
| Month | [1, 12] |
| Day | [1, 指定年份的月份中的天数] |
| Hour | [0, 23] |
| Minute | [0, 59] |
| Second | [0, 59] |
| microsecond | [0, 1000000] |
| Tzinfo | tzinfo的子类对象，如timezone类的实例 |

**类方法和属性**

| **类方法/属性名称** | **描述** |
| --- | --- |
| datetime.today() | 返回一个表示当前本期日期时间的datetime对象 |
| datetime.now([tz]) | 返回指定时区日期时间的datetime对象，如果不指定tz参数则结果同上 |
| datetime.utcnow() | 返回当前utc日期时间的datetime对象 |
| datetime.fromtimestamp(timestamp[, tz]) | 根据指定的时间戳创建一个datetime对象 |
| datetime.utcfromtimestamp(timestamp) | 根据指定的时间戳创建一个datetime对象 |
| datetime.combine(date, time) | 把指定的date和time对象整合成一个datetime对象 |
| datetime.strptime(date\_str, format) | 将时间字符串转换为datetime对象 |

**对象方法和属性**

| **对象方法/属性名称** | **描述** |
| --- | --- |
| dt.year, dt.month, dt.day | 年、月、日 |
| dt.hour, dt.minute, dt.second | 时、分、秒 |
| dt.microsecond, dt.tzinfo | 微秒、时区信息 |
| dt.date() | 获取datetime对象对应的date对象 |
| dt.time() | 获取datetime对象对应的time对象， tzinfo 为None |
| dt.timetz() | 获取datetime对象对应的time对象，tzinfo与datetime对象的tzinfo相同 |
| dt.replace([year[, month[, day[, hour[, minute[, second[, microsecond[,bh tzinfo]]]]]]]]) | 生成并返回一个新的datetime对象，如果所有参数都没有指定，则返回一个与原datetime对象相同的对象 |
| dt.timetuple() | 返回datetime对象对应的tuple（不包括tzinfo） |
| dt.utctimetuple() | 返回datetime对象对应的utc时间的tuple（不包括tzinfo） |
| dt.toordinal() | 返回日期是是自 0001-01-01 开始的第多少天 |
| dt.weekday() | 返回日期是星期几，[0, 6]，0表示星期一 |
| dt.isocalendar() | 返回日期是星期几，[1, 7], 1表示星期一 |
| dt.isoformat([sep]) | 返回一个‘%Y-%m-%d |
| dt.ctime() | 等价于time模块的time.ctime(time.mktime(d.timetuple())) |
| dt.strftime(format) | 返回指定格式的时间字符串 |

实例

>>> from datetime import datetime, timezone

>>>

>>> datetime.today()

datetime.datetime(2017, 2, 4, 20, 44, 40, 556318)

>>> datetime.now()

datetime.datetime(2017, 2, 4, 20, 44, 56, 572615)

>>> datetime.now(timezone.utc)

datetime.datetime(2017, 2, 4, 12, 45, 22, 881694, tzinfo=datetime.timezone.utc)

>>> datetime.strptime('2017/02/04 20:49', '%Y/%m/%d %H:%M')

datetime.datetime(2017, 2, 4, 20, 49)

>>> dt = datetime.now()

>>> dt

datetime.datetime(2017, 2, 4, 20, 57, 0, 621378)

>>> dt.year

2017

>>> dt.month

2

>>> dt.day

4

>>> dt.hour

20

>>> dt.minute

57

>>> dt.second

0

>>> dt.microsecond

621378

>>> dt.tzinfo

>>> dt.timestamp()

1486213020.621378

>>> dt.date()

datetime.date(2017, 2, 4)

>>> dt.time()

datetime.time(20, 57, 0, 621378)

>>> dt.replace()

datetime.datetime(2017, 2, 4, 20, 57, 0, 621378)

>>> dt.replace(2016)

datetime.datetime(2016, 2, 4, 20, 57, 0, 621378)

>>> dt.weekday()

5

>>> dt.strftime('%Y%m%d %H:%M:%S.%f')

'20170204 20:57:00.621378'

### datetime.timedelta类

timedelta对象表示连个不同时间之间的差值。可以很方便的对时间进行运算，比如10天以后是什么日期。

class datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, hours=0, weeks=0)

实例：

>>> import datetime

>>>

>>> datetime.timedelta(365).total\_seconds() # 一年包含的总秒数

31536000.0

>>> dt = datetime.datetime.now()

>>> dt + datetime.timedelta(3) # 3天后

datetime.datetime(2017, 2, 8, 9, 39, 40, 102821)

>>> dt + datetime.timedelta(-3) # 3天前

datetime.datetime(2017, 2, 2, 9, 39, 40, 102821)

>>> dt + datetime.timedelta(hours=3) # 3小时后

datetime.datetime(2017, 2, 5, 12, 39, 40, 102821)

>>> dt + datetime.timedelta(hours=-3) # 3小时前

datetime.datetime(2017, 2, 5, 6, 39, 40, 102821)

>>> dt + datetime.timedelta(hours=3, seconds=30) # 3小时30秒后

datetime.datetime(2017, 2, 5, 12, 40, 10, 102821)

### 时间格式码

|  |  |  |
| --- | --- | --- |
| **标识** | **含义** | **举例** |
| **%a** | 星期简写 | **Mon** |
| **%A** | 星期全称 | **Monday** |
| **%b** | 月份简写 | **Mar** |
| **%B** | 月份全称 | **March** |
| **%c** | 适合语言下的时间表示 | **May Mon May 20 16:00:02 2013** |
| **%d** | 一个月的第一天，取值范围： [01,31]. | **20** |
| **%H** | 24小时制的小时，取值范围[00,23]. | **17** |
| **%I** | 12小时制的小时，取值范围 [01,12]. | **10** |
| **%j** | 一年中的第几天，取值范围 [001,366]. | **120** |
| **%m** | 十进制月份，取值范围[01,12]. | **50** |
| **%M** | 分钟，取值范围 [00,59]. | **50** |
| **%p** | 上、下午，AM 或 PM. | **PM** |
| **%S** | 秒，取值范围 [00,61]. | **30** |
| **%U** | 这一年的星期数（星期天为一个星期的第一天，开年的第一个星期天之前的天记到第0个星期）趋势范围[00,53]. | **20** |
| **%w** | 星期的十进制表示，取值范围 [0(星期天),6]. | **1** |
| **%W** | 这一年的星期数（星一为一个星期的第一天，开年的第一个星期一之前的天记到第0个星期）趋势范围[00,53]. | **20** |
| **%x** | 特定自然语言下的日期表示 | **05/20/13** |
| **%X** | 特定自然语言下的时间表示 | **16:00:02** |
| **%y** | 年的后两位数，取值范围[00,99]. | **13** |
| **%Y** | 完整的年 | **2013** |
| **%Z** | 时区名 | **CST（China Standard Time）** |
| **%%** | %字符 | **%** |

## 3、calendar模块

calendar（日历）模块，默认每周第一天是星期一，最后一天是星期天。

函数及描述

**（1）、 calendar.calendar(year, w=2, l=1, c=6, m=3)**

返回一个多行**字符串**格式的year年年历。m: 3个月1行. C是月与月之间的间隔 w是每个日期的宽度。l是每星期行数。

print(calendar.calendar(2018))

**（2）、 calendar.firstweekday()**

返回当前每周起始日期的设置。默认返回0，即星期一。

**（3）、 calendar.isleap(year)**

闰年返回True，否则False。

**（4）、 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。

print(calendar.month(2017, 1))

**（5）、 calendar.monthcalendar(year,month)**

返回一个整数的单层嵌套列表。每个子列表装载一个星期。该月之外的日期都为0，该月之内的日期设为该日的日期，从1开始。

lst = list(calendar.monthcalendar(2017,1))  
print(lst)

[[0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0, 0]]

**（6）、 calendar.monthrange(year, month)**

返回两个整数组成的元组，第一个是该月的第一天是星期几，第二个是该月的天数。

>>>calendar.monthrange(2017, 2)

(2, 28)

**（7）、 calendar.setfirstweekday(weekday)**

设置每周起始日期码。0（星期一）到6（星期日）

**（8）、 calendar.weekday(year, month, day)**

返回给定日期的星期码，0（星期一）到6（星期日）。