Python提供了多个内置模块用于操作日期时间，像calendar，time，datetime。

time模块提供的接口与C标准库time.h基本一致。

相比于time模块，datetime模块的接口则更直观、更容易调用。

## datetime模块

datetime模块定义了两个常量：datetime.MINYEAR和datetime.MAXYEAR，分别表示datetime所能表示的最小、最大年份。其中，MINYEAR = 1，MAXYEAR = 9999。（对于偶等玩家，这个范围已经足够用了）

datetime模块定义了下面这几个类

datetime.date：表示日期的类。常用的属性有year, month, day；

datetime.time：表示时间的类。常用的属性有hour, minute, second, microsecond；

datetime.datetime：表示日期时间。

datetime.timedelta：表示时间间隔，即两个时间点之间的长度。

datetime.tzinfo：与时区有关的相关信息。

Note:上面这些类型的对象都是不可变（immutable）的。

## date类

date类表示一个日期。日期由年、月、日组成。date类的构造函数如下：

class datetime.date(year, month, day)：参数的意义就不多作解释了，只是有几点要注意一下：

year的范围是[MINYEAR, MAXYEAR]，即[1, 9999]；

month的范围是[1, 12]。（月份是从1开始）；

day的最大值根据给定的year, month参数来决定。例如闰年2月份有29天；

date类定义了一些常用的类方法与类属性，方便我们操作：

date.max、date.min：date对象所能表示的最大、最小日期；

date.resolution：date对象表示日期的最小单位。这里是天。

date.today()：返回一个表示当前本地日期的date对象；

date.fromtimestamp(timestamp)：根据给定的时间戮，返回一个date对象；

datetime.fromordinal(ordinal)：将Gregorian日历时间转换为date对象；（Gregorian Calendar：一种日历表示方法，类似于我国的农历，西方国家使用比较多。）

例子：

datetime.date(2015, 7, 6) 或者datetime(1949,1,1)

from datetime import \*

import time

print 'date.max:', date.max

print 'date.min:', date.min

print 'date.today():', date.today()

print 'date.fromtimestamp():', date.fromtimestamp(time.time())

# # ---- 结果 ----

# date.max: 9999-12-31

# date.min: 0001-01-01

# date.today(): 2010-04-06

# date.fromtimestamp(): 2010-04-06

date提供的实例方法和属性

date.year、date.month、date.day：年、月、日；

date.replace(year, month, day)：生成一个新的日期对象，用参数指定的年，月，日代替原有对象中的属性。（原有对象仍保持不变）

date.timetuple()：返回日期对应的time.struct\_time对象；

date.toordinal()：返回日期对应的Gregorian Calendar日期；

date.weekday()：返回weekday，如果是星期一，返回0；如果是星期2，返回1，以此类推；

data.isoweekday()：返回weekday，如果是星期一，返回1；如果是星期2，返回2，以此类推；

date.isocalendar()：返回格式如(year，month，day)的元组；

date.isoformat()：返回格式如'YYYY-MM-DD’的字符串；

date.strftime(fmt)：自定义格式化字符串。在下面详细讲解。

datetime 对象与它所保存的字符串格式时间戳之间可以互相转换。str() 函数是可用的，但更推荐 datetime.strptime() 方法。这个方法可以实现双向转换。

例子：

>>> str(now)

'2014-06-17 15:56:19.313193'

>>> now.strftime('%Y-%m-%d')

'2014-06-17'

>>> datetime.strptime('2010-01-01','%Y-%m-%d')

datetime.datetime(2010, 1, 1, 0, 0)

使用例子

now = date(2010, 04, 06)

tomorrow = now.replace(day = 07)

print 'now:', now, ', tomorrow:', tomorrow

print 'timetuple():', now.timetuple()

print 'weekday():', now.weekday()

print 'isoweekday():', now.isoweekday()

print 'isocalendar():', now.isocalendar()

print 'isoformat():', now.isoformat()

# # ---- 结果 ----

# now: 2010-04-06 , tomorrow: 2010-04-07

# timetuple(): (2010, 4, 6, 0, 0, 0, 1, 96, -1)

# weekday(): 1

# isoweekday(): 2

# isocalendar(): (2010, 14, 2)

# isoformat(): 2010-04-06

date还对某些操作进行了重载，它允许我们对日期进行如下一些操作：

date2 = date1 + timedelta # 日期加上一个间隔，返回一个新的日期对象（timedelta将在下面介绍，表示时间间隔）

date2 = date1 - timedelta # 日期隔去间隔，返回一个新的日期对象

timedelta = date1 - date2 # 两个日期相减，返回一个时间间隔对象

date1 < date2 # 两个日期进行比较

Note:对日期进行操作时，要防止日期超出它所能表示的范围。

## Time类

time类表示时间，由时、分、秒以及微秒组成。time类的构造函数如下：

class datetime.time(hour[, minute[, second[, microsecond[, tzinfo]]]]) ：各参数的意义不作解释，这里留意一下参数tzinfo，它表示时区信息。注意一下各参数的取值范围：hour的范围为[0, 24)，minute的范围为[0, 60)，second的范围为[0, 60)，microsecond的范围为[0, 1000000)。

time类定义的类属性：

time.min、time.max：time类所能表示的最小、最大时间。其中，time.min = time(0, 0, 0, 0)， time.max = time(23, 59, 59, 999999)；

time.resolution：时间的最小单位，这里是1微秒；

time类提供的实例方法和属性：

time.hour、time.minute、time.second、time.microsecond：时、分、秒、微秒；

time.tzinfo：时区信息；

time.replace([hour[, minute[, second[, microsecond[, tzinfo]]]]])：创建一个新的时间对象，用参数指定的时、分、秒、微秒代替原有对象中的属性（原有对象仍保持不变）；

time.isoformat()：返回型如"HH:MM:SS"格式的字符串表示；

time.strftime(fmt)：返回自定义格式化字符串。在下面详细介绍；

使用例子：

from datetime import \*

tm = time(23, 46, 10)

print 'tm:', tm

print 'hour: %d, minute: %d, second: %d, microsecond: %d' /

% (tm.hour, tm.minute, tm.second, tm.microsecond)

tm1 = tm.replace(hour = 20)

print 'tm1:', tm1

print 'isoformat():', tm.isoformat()

# # ---- 结果 ----

# tm: 23:46:10

# hour: 23, minute: 46, second: 10, microsecond: 0

# tm1: 20:46:10

# isoformat(): 23:46:10

像date一样，也可以对两个time对象进行比较，或者相减返回一个时间间隔对象。

## datetime类

datetime是date与time的结合体，包括date与time的所有信息。它的构造函数如下：datetime.datetime(year, month, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]])，各参数的含义与date、time的构造函数中的一样，要注意参数值的范围。

datetime类定义的类属性与方法：

datetime.min、datetime.max：datetime所能表示的最小值与最大值；

datetime.resolution：datetime最小单位；

datetime.today()：返回一个表示当前本地时间的datetime对象；

datetime.now([tz])：返回一个表示当前本地时间的datetime对象，如果提供了参数tz，则获取tz参数所指时区的本地时间；# ctime()) #Accurate to microseconds

datetime.utcnow()：返回一个当前utc时间的datetime对象；

datetime.fromtimestamp(timestamp[, tz])：根据时间戮创建一个datetime对象，参数tz指定时区信息；

datetime.utcfromtimestamp(timestamp)：根据时间戮创建一个datetime对象；

datetime.combine(date, time)：根据date和time，创建一个datetime对象；

datetime.strptime(date\_string, format)：将格式字符串转换为datetime对象；

使用例子：

from datetime import \*

import time

print 'datetime.max:', datetime.max

print 'datetime.min:', datetime.min

print 'datetime.resolution:', datetime.resolution

print 'today():', datetime.today()

print 'now():', datetime.now()

print 'utcnow():', datetime.utcnow()

print 'fromtimestamp(tmstmp):', datetime.fromtimestamp(time.time())

print 'utcfromtimestamp(tmstmp):', datetime.utcfromtimestamp(time.time())

# ---- 结果 ----

# datetime.max: 9999-12-31 23:59:59.999999

# datetime.min: 0001-01-01 00:00:00

# datetime.resolution: 0:00:00.000001

# today(): 2010-04-07 09:48:16.234000

# now(): 2010-04-07 09:48:16.234000

# utcnow(): 2010-04-07 01:48:16.234000 # 中国位于+8时间，与本地时间相差8

# fromtimestamp(tmstmp): 2010-04-07 09:48:16.234000

# utcfromtimestamp(tmstmp): 2010-04-07 01:48:16.234000

datetime类提供的实例方法与属性（很多属性或方法在date和time中已经出现过，在此有类似的意义，这里只罗列这些方法名，具体含义不再逐个展开介绍，可以参考上文对date与time类的讲解。）：

datetime.year、month、day、hour、minute、second、microsecond、tzinfo：

datetime.date()：获取date对象；

datetime.time()：获取time对象；

datetime.replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]]])：

datetime.timetuple()

datetime.utctimetuple()

datetime.toordinal()

datetime.weekday()

datetime.isocalendar()

datetime.isoformat([sep])

datetime.ctime()：返回一个日期时间的C格式字符串，等效于time.ctime(time.mktime(dt.timetuple()))；

datetime.strftime(format)

像date一样，也可以对两个datetime对象进行比较，或者相减返回一个时间间隔对象，或者日期时间加上一个间隔返回一个新的日期时间对象。

datetime.timedelta类和对象

直接创建或者两个datetime对象相减就可以得到timedelta对象。

timedelta对象属性和方法

timedelta.min: The most negative timedelta object, timedelta(-999999999).

timedelta.max: The most positive timedelta object, timedelta(days=999999999,hours=23, minutes=59, seconds=59, microseconds=999999).

timedelta.resolution: The smallest possible difference between non-equal timedelta objects,timedelta(microseconds=1).

Note: 要将datetime时间差换算成hour不能使用seconds转换（这样只包含秒的差额），而应使用total\_seconds()函数。

示例

from dateutil.parser import parse

import datetime

a = 'Sat Jul 31 20:15:24 +0000 2011'

b = 'Sat Jul 30 20:07:44 +0000 2011'

# print(parse(a))

aa = datetime.datetime.timestamp(parse(a))

bb = datetime.datetime.timestamp(parse(b))

print(datetime.datetime.fromtimestamp(aa) - datetime.datetime.fromtimestamp(bb))

print((datetime.datetime.fromtimestamp(aa) - datetime.datetime.fromtimestamp(bb)).total\_seconds() / 3600)

## 格式字符串

datetime、date、time都提供了strftime()方法，该方法接收一个格式字符串，输出日期时间的字符串表示。下表是从python手册中拉过来的.

格式字符 意义

%a星期的简写。如 星期三为Web

%A星期的全写。如 星期三为Wednesday

%b月份的简写。如4月份为Apr

%B月份的全写。如4月份为April

%c: 日期时间的字符串表示。（如： 04/07/10 10:43:39）

%d: 日在这个月中的天数（是这个月的第几天）

%f: 微秒（范围[0,999999]）

%H: 小时（24小时制，[0, 23]）

%I: 小时（12小时制，[0, 11]）

%j: 日在年中的天数 [001,366]（是当年的第几天）

%m: 月份（[01,12]）

%M: 分钟（[00,59]）

%p: AM或者PM

%S: 秒（范围为[00,61]，为什么不是[00, 59]，参考python手册~\_~）

%U: 周在当年的周数当年的第几周），星期天作为周的第一天

%w: 今天在这周的天数，范围为[0, 6]，6表示星期天

%W: 周在当年的周数（是当年的第几周），星期一作为周的第一天

%x: 日期字符串（如：04/07/10）

%X: 时间字符串（如：10:43:39）

%y: 2个数字表示的年份

%Y: 4个数字表示的年份

%z: 与utc时间的间隔 （如果是本地时间，返回空字符串）

%Z: 时区名称（如果是本地时间，返回空字符串）

%%: %% => %

例子：

dt = datetime.now()

print '(%Y-%m-%d %H:%M:%S %f): ', dt.strftime('%Y-%m-%d %H:%M:%S %f')

print '(%Y-%m-%d %H:%M:%S %p): ', dt.strftime('%y-%m-%d %I:%M:%S %p')

print '%%a: %s ' % dt.strftime('%a')

print '%%A: %s ' % dt.strftime('%A')

print '%%b: %s ' % dt.strftime('%b')

print '%%B: %s ' % dt.strftime('%B')

print '日期时间%%c: %s ' % dt.strftime('%c')

print '日期%%x：%s ' % dt.strftime('%x')

print '时间%%X：%s ' % dt.strftime('%X')

print '今天是这周的第%s天 ' % dt.strftime('%w')

print '今天是今年的第%s天 ' % dt.strftime('%j')

print '今周是今年的第%s周 ' % dt.strftime('%U')

# # ---- 结果 ----

# (%Y-%m-%d %H:%M:%S %f): 2010-04-07 10:52:18 937000

# (%Y-%m-%d %H:%M:%S %p): 10-04-07 10:52:18 AM

# %a: Wed

# %A: Wednesday

# %b: Apr

# %B: April

# 日期时间%c: 04/07/10 10:52:18

# 日期%x：04/07/10

# 时间%X：10:52:18

# 今天是这周的第3天

# 今天是今年的第097天

# 今周是今年的第14周

## time模块

time模块提供各种操作时间的函数

说明：一般有两种表示时间的方式:

第一种是时间戳的方式(相对于1970.1.1 00:00:00以秒计算的偏移量),时间戳是惟一的

第二种以数组的形式表示即(struct\_time),共有九个元素，分别表示，同一个时间戳的struct\_time会因为时区不同而不同

year (four digits, e.g. 1998)

month (1-12)

day (1-31)

hours (0-23)

minutes (0-59)

seconds (0-59)

weekday (0-6, Monday is 0)

Julian day (day in the year, 1-366)

DST (Daylight Savings Time) flag (-1, 0 or 1) 是否是夏令时

If the DST flag is 0, the time is given in the regular time zone;

if it is 1, the time is given in the DST time zone;

if it is -1, mktime() should guess based on the date and time.

python获取当前时间

1.asctime()

asctime([tuple]) -> string

将一个struct\_time(默认为当时时间)，转换成字符串

Convert a time tuple to a string, e.g. 'Sat Jun 06 16:26:11 1998'.

When the time tuple is not present, current time as returned by localtime()

is used.

2. clock()

clock() -> floating point number

该函数有两个功能，

在第一次调用的时候，返回的是程序运行的实际时间；

以第二次之后的调用，返回的是自第一次调用后,到这次调用的时间间隔

python3中clock()已经弃用的，使用perf\_counter()或者process\_time()代替。

Deprecated since version 3.3: The behaviour of this function depends on the platform: use perf\_counter() or process\_time() instead, depending on yourrequirements, to have a well defined behaviour.

示例：

import time

if \_\_name\_\_ == '\_\_main\_\_':

time.sleep(1)

print "clock1:%s" % time.clock()

time.sleep(1)

print "clock2:%s" % time.clock()

time.sleep(1)

print "clock3:%s" % time.clock()

输出：

clock1:3.35238137808e-006

clock2:1.00004944763

clock3:2.00012040636

其中第一个clock输出的是程序运行时间

第二、三个clock输出的都是与第一个clock的时间间隔

3.sleep(...)

sleep(seconds)

线程推迟指定的时间运行，经过测试，单位为秒，但是在帮助文档中有以下这样一句话，这关是看不懂

“The argument may be a floating point number for subsecond precision.”

4.ctime(...)

ctime(seconds) -> string

将一个时间戳(默认为当前时间)转换成一个时间字符串

例如：

time.ctime()

输出为：'Sat Mar 28 22:24:24 2009'

5.gmtime(...)

gmtime([seconds]) -> (tm\_year, tm\_mon, tm\_day, tm\_hour, tm\_min,tm\_sec, tm\_wday, tm\_yday, tm\_isdst)

将一个时间戳转换成一个UTC时区(0时区)的struct\_time，如果seconds参数未输入，则以当前时间为转换标准

6.localtime(...)

localtime([seconds]) -> (tm\_year,tm\_mon,tm\_day,tm\_hour,tm\_min,tm\_sec,tm\_wday,tm\_yday,tm\_isdst)

将一个时间戳转换成一个当前时区的struct\_time，如果seconds参数未输入，则以当前时间为转换标准

利用time模块的格式化时间的方法来处理：

time.localtime(time.time())

用time.localtime()方法，作用是格式化时间戳为本地的时间。

输出的结果是：

time.struct\_time(tm\_year=2010, tm\_mon=7, tm\_mday=19, tm\_hour=22, tm\_min=33, tm\_sec=39, tm\_wday=0, tm\_yday=200, tm\_isdst=0)

现在看起来更有希望格式成我们想要的时间了。

7.mktime(...)

mktime(tuple) -> floating point number

将一个以struct\_time转换为时间戳

8.time.strftime()方法

把一大串信息格式化成我们想要的东西

time.strftime('%Y-%m-%d',time.localtime(time.time()))

time.strftime的参数：

strftime(format[, tuple]) -> string

将指定的struct\_time(默认为当前时间)，根据指定的格式化字符串输出

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

9.strptime(...)

strptime(string, format) -> struct\_time

将时间字符串根据指定的格式化符转换成数组形式的时间

例如：

2009-03-20 11:45:39 对应的格式化字符串为：%Y-%m-%d %H:%M:%S

Sat Mar 28 22:24:24 2009 对应的格式化字符串为：%a %b %d %H:%M:%S %Y

10.time(...)

time() -> floating point number

返回当前时间的时间戳

## 小应用

1.python获取当前时间

time.time() 获取当前时间戳

time.localtime() 当前时间的struct\_time形式

time.ctime() 当前时间的字符串形式

2.python格式化字符串

格式化成2009-03-20 11:45:39形式

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

格式化成Sat Mar 28 22:24:24 2009形式

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

3.将格式字符串转换为时间戳

a = "Sat Mar 28 22:24:24 2009"

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