

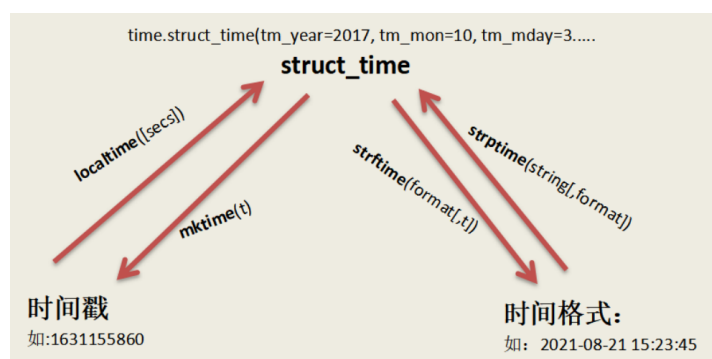
Python 中提供了对时间日期的多种多样的处理方式，主要是在 `time` 和 `datetime` 这两个模块里

一、time模块

不牵扯时区的问题,便于计算

- a、timestamp时间戳，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
- b、struct_time时间元组，共有九个元素组。
- c、format time 格式化时间，已格式化的结构使时间更具可读性。包括自定义格式和固定格式。

1、时间格式转换图：



主要time生成方法和time格式转换方法实例：

```
1 # 导入time模块
2 import time
```

```
1 # 生成timestamp
2 time.time()
```

```
1 1650263361.5318103
```

```
1 int(time.time())
```

```
1 1650263381
```

```
1 # 程序开始时间
2 start_time = time.time()
3
4 # 程序
5 s = ""
6 for i in range(10000):
7     s += str(i)
8
9 end_time = time.time()
10
11 print("程序消耗时间=",end_time-start_time)
```

```
1 程序消耗时间= 0.012003898620605469
```

```
1 #生成struct_time
2 # timestamp to struct_time 本地时间
3 my_time = time.localtime()
4 print(my_time)
5
6 print(my_time.tm_year)
7 print(my_time.tm_mon)
8 print(my_time.tm_mday)
```

```
1 time.struct_time(tm_year=2022, tm_mon=4, tm_mday=18, tm_hour=14,
tm_min=36, tm_sec=47, tm_wday=0, tm_yday=108, tm_isdst=0)
```

```
1 2022
```

```
1 # 将timestamp转化为 struct_time
2 time.localtime(1650177058)
```

```
1 time.struct_time(tm_year=2022, tm_mon=4, tm_mday=17, tm_hour=14,
tm_min=30, tm_sec=58, tm_wday=6, tm_yday=107, tm_isdst=0)
```

```
1 #格式化字符串到 struct_time
2 time.strptime('2011-05-05 16:37:06', '%Y-%m-%d %X')
```

```
1 time.struct_time(tm_year=2011, tm_mon=5, tm_mday=5, tm_hour=16,
tm_min=37, tm_sec=6, tm_wday=3, tm_yday=125, tm_isdst=-1)
```

```
1 time.mktime(time.strptime('2011-05-05 16:37:06', '%Y-%m-%d %X'))
```

```
1 1304584626.0
```

```
1 #生成format_time
2 #struct_time to format_time
3 time.strftime("%Y-%m-%d %X")
4 #time.strftime("%Y-%m-%d %X",time.localtime())
```

```
1 '2022-04-18 14:41:56'
```

```
1 #生成format_time
2 #struct_time to format_time
3 time.strftime("%Y-%m-%d")
4 #time.strftime("%Y-%m-%d %X",time.localtime())
```

```
1 '2022-04-18'
```

```
1 #生成format_time
2 #struct_time to format_time
3 time.strftime("%m-%d-%Y")
4 #time.strftime("%Y-%m-%d %X",time.localtime())
```

```
1 '04-18-2022'
```

✧ struct_time元组元素结构

属性	值
○ tm_year (年)	比如2011
○ tm_mon (月)	1 - 12
○ tm_mday (日)	1 - 31
○ tm_hour (时)	0 - 23
○ tm_min (分)	0 - 59
○ tm_sec (秒)	0 - 61
○ tm_wday (weekday)	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

作用:

- 取得 时间戳/时间格式的字符串 中对应的 年/月/日等信息
- 作为时间戳和字符串时间之间的桥梁

```
1 time_struct = time.strptime('2011-05-07 16:37:06', '%Y-%m-%d %X')
2 print(time_struct.tm_year)
3 print(time_struct.tm_mon)
4 print(time_struct.tm_mday)
5 print(time_struct.tm_hour)
6 print(time_struct.tm_min)
7 my = 'aaa'
8 '%s'% my
9 my_int = 1
10 '%d'% my_int
11
12 "我们在{}工作".format('家里')
13 addr = "家里"
14
15 f"我们在{addr}工作"
```

1	2011
2	5
3	7
4	16
5	37

* format time结构化表示

格式 含义

- %Y -年[0001, ..., 2018, 2019, ..., 9999]
- %m -月[01, 02, ..., 11, 12]
- %d -天[01, 02, ..., 30, 31]
- %H -小时[00, 01, ..., 22, 23]
- %M -分钟[00, 01, ..., 58, 59]
- %S -秒[00, 01, ..., 58, 61]
- %X 本地相应时间
- %y 去掉世纪的年份 (00 - 99)

常见结构化时间组合

```
1 time.strftime("%Y-%m-%d %X")
```

```
1 '2022-04-18 14:50:46'
```

```
1 time.strftime("%Y-%m-%d")
```

```
1 '2022-04-18'
```

```
1 time.strftime("%m")
```

```
1 '04'
```

time运算

```
1 #获取明天的这个时间点
2 import time
3 t1 = time.time()
4 #timestamp加减单位以秒为单位
5 t2=t1+24*60*60
6
7 time.strftime("%Y-%m-%d %X",time.localtime(t2))
```

```
1 '2022-04-19 14:52:00'
```

```
1 # 倒计时
2 for i in range(5):
3     print('\r', ' %s 秒! ' % (5-i), end='')
4     # 暂停1s后运行
5     time.sleep(1)
6     print('\r',"发射!!!!")
```

```
1 发射!!!!
```

二、datetime模块

datetime模块重新封装了time模块，提供更多接口，提供的类有：

date,time,datetime,timedelta,tzinfo

1.date类

```
datetime.date(year, month, day)
```

静态方法和字段

- date.today(): 返回一个表示当前本地日期的date对象；

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

```
1 from datetime import *
2 import time
3 print('date.today():', date.today())
4 print('date.fromtimestamp():', date.fromtimestamp(time.time()))
```

方法和属性

```
d1 = date(2011,06,03) #date对象
```

- `d1.year`、`date.month`、`date.day`: 年、月、日;
- `d1.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)
- `d1.timetuple()`: 返回日期对应的`time.struct_time`对象;
- `d1.weekday()`: 返回`weekday`, 如果是星期一, 返回0; 如果是星期二, 返回1, 以此类推;
- `d1.isoweekday()`: 返回`weekday`, 如果是星期一, 返回1; 如果是星期二, 返回2, 以此类推;
- `d1.isoformat()`: 返回格式如'YYYY-MM-DD'的字符串;
- `d1.strftime(fmt)`: 和`time`模块`format`相同。

```
1 now = date(2021, 10, 26)
2 print(now.year, now.month, now.day)
3 tomorrow = now.replace(day = 27)
4 print('now:', now, ', tomorrow:', tomorrow)
5 print('timetuple():', now.timetuple())
6 print('weekday():', now.weekday())
7 print('isoweekday():', now.isoweekday())
8 print('isoformat():', now.isoformat())
9 print('strftime():', now.strftime("%Y-%m-%d"))
```

datetime类

`datetime`相当于`date`和`time`结合起来。

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

静态方法

- `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;
- `datetime.fromtimestamp(timestamp[, tz])`: 根据时间戳创建一个`datetime`对象, 参数`tz`指定时区信息;
- `datetime.strptime(date_string, format)`: 将格式字符串转换为`datetime`对象;

```
1 from datetime import *
2 import time
3 now = datetime.now()
4 print('today():', datetime.today())
5 print('now():', datetime.now())
6 print('fromtimestamp(tmstamp):',
7       datetime.fromtimestamp(time.time()))
8 print('datetime.strptime(date_string,
9       format):', datetime.strptime('2022-03-21', "%Y-%m-%d"))
```

* timedelta类, 时间加减

使用`timedelta`可以很方便的在日期上做天`days`, 小时`hour`, 分钟, 秒, 毫秒, 微妙的时间计算, 如果要计算月份则需要另外的办法

```
1 from datetime import *
2
3 dt = datetime.now()
4 #日期减一天
5 dt_1 = dt + timedelta(days=-1)#昨天
6 dt_11 = dt - timedelta(days=1)#昨天
7 dt3 = dt + timedelta(days=1)#明天
8
9 delta_obj = dt3-dt
10 print(type(delta_obj), delta_obj)#<type 'datetime.timedelta'> 1
11   day, 0:00:00
12 # total_seconds():返回在该时间实例的给定持续时间内覆盖的总秒数
13 print(delta_obj.days , delta_obj.total_seconds())#1 86400.0
```