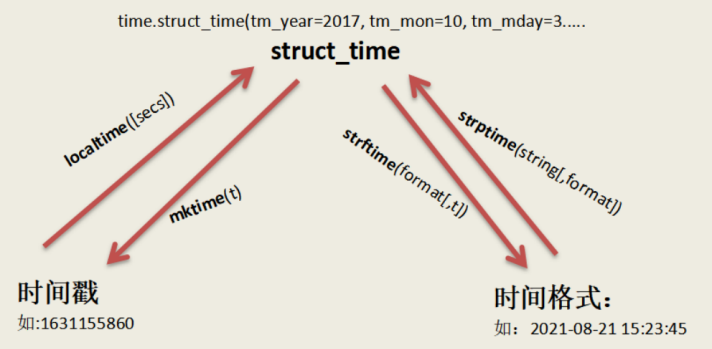
Python 中提供了对时间日期的多种多样的处理方式，主要是在 time 和 datetime 这两个模块里

一、time模块

- 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：

```
# 导入time模块
import time
```

```
# 生成timestamp
time.time()
```

```
#生成struct_time
# timestamp to struct_time 本地时间
print(time.localtime())
time.localtime(time.time())
```

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

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

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

```
time_stuct = time.strptime('2011-05-07 16:37:06', '%Y-%m-%d %X')
print(time_stuct.tm_year)
print(time_stuct.tm_mon)
print(time_stuct.tm_mday)
print(time_stuct.tm_hour)
print(time_stuct.tm_min)
```

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

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

time运算

```
#获取明天的这个时间点
import time
t1 = time.time()
#timestamp加减单位以秒为单位
t2=t1+24*60*60

time.strftime("%Y-%m-%d %X",time.localtime(t2))
```

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

二、datetime模块

datatime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；

```
from datetime import *
import time
print('date.today():', date.today())
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2，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2，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。

```
now = date(2021, 10, 26)
print(now.year,now.month,now.day)
tomorrow = now.replace(day = 27)
print('now:', now, ', tomorrow:', tomorrow)
print('timetuple():', now.timetuple())
print('weekday():', now.weekday())
print('isoweekday():', now.isoweekday())
print('isoformat():', now.isoformat())
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；

```
from  datetime import *
import time
now = datetime.now()
print('today():', datetime.today())
print('now():', datetime.now())
print('fromtimestamp(tmstmp):',
datetime.fromtimestamp(time.time()))
print('datetime.strptime(date_string,
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

```
from  datetime import *

dt = datetime.now()
#日期减一天
dt_1 = dt + timedelta(days=-1)#昨天
dt_11 = dt - timedelta(days=1)#昨天
dt3 = dt + timedelta(days=1)#明天

delta_obj = dt3-dt
print(type(delta_obj),delta_obj)#<type 'datetime.timedelta'> 1
day, 0:00:00
# total_seconds():返回在该时间实例的给定持续时间内覆盖的总秒数
print(delta_obj.days ,delta_obj.total_seconds())#1 86400.0
```

```

```