Python 时间使用指南 - V1.0

出品：Python 数据之道

作者：Lemon

18 Oct, 2020

Contents

1 前言

在用 Python 进行数据处理，尤其是时间序列梳理的处理，经常会涉及处理时间或日期的地方，有些看似简单的问题，经常会混淆，甚至被困住。

虽然网上已有不少关于 Python 时间模块介绍的内容，但总体来说较为零散，因此 Lemon 萌生了整理一份《Python 时间使用指南》的念头。

《Python 时间使用指南》主要对 Python 内置的时间模块来进行了整理，Python 内置时间处理模块主要包括：

- Datetime
- Time
- Calendar

概括来说，时间和日期处理的模块和库，有几个主要的概念都是会涉及的，包括：

`日期`，`时间`，`时间戳`，`时间和字符串的转换`，`时区` 等。

`Datetime`、`Time`、`Calendar` 这三个模块中，我们用的较多的是 `Datetime`，在《指南》中我们介绍的也较为详细。

针对 `Datetime` 模块，《指南》从构建时间对象实例、时间转换、时间对象的运算三个方面进行了梳理，共涉及 13 个知识点，整理的大纲如下：

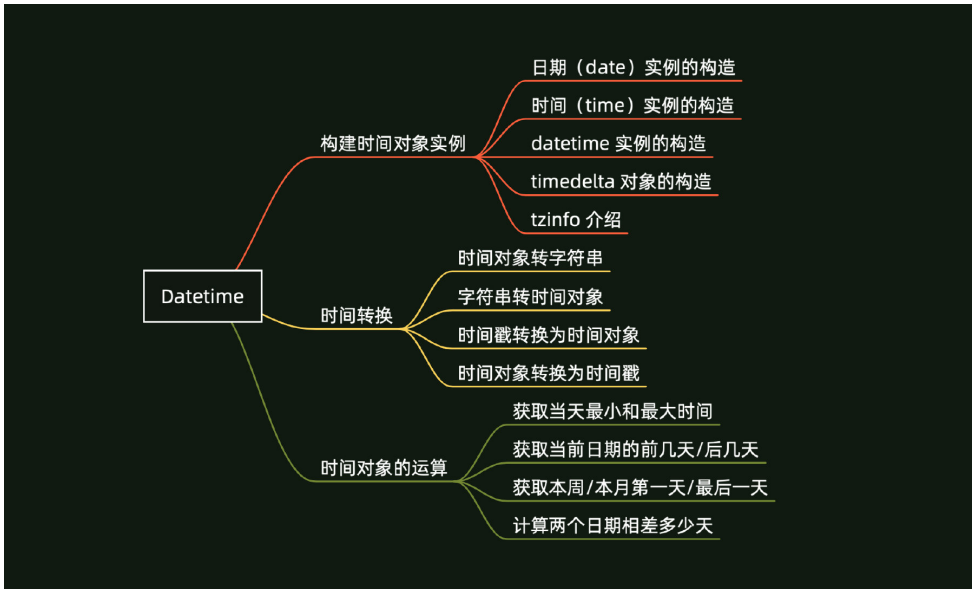

Figure 1.1: datetime

针对 `Time` 模块，《指南》从时间获取和时间转换两个方面进行了梳理，共涉及 6 个知识点，整理的大纲如下：

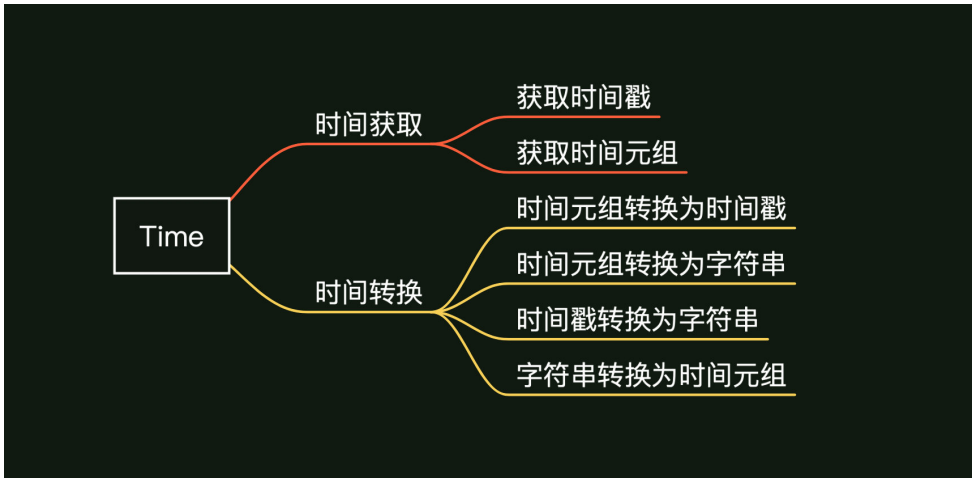

Figure 1.2: time

由于个人水平有限，《Python 时间使用指南》中如有不准确或不完善的地方，欢迎通过微信公众号「Python 数据之道」与我联系。

《指南》会不定期进行更新，最新版的《Python 时间使用指南》可以在公众号 **「Python 数据之道」**后台回复 **“time”** 来获取。

2 Datetime

首先要介绍的，是大家平时用的较多的 datetime 模块, 它属于 Python 的内置模块，功能相对较全。

2.1 构建时间对象实例

```
import datetime
# from datetime import date, datetime,time,timedelta,tzinfo
import time
```

时间实例的构造包括日期（如 2020 年 10 月 12 日），时间（如 20 点 10 分 01 秒），或者是包含 date 和 time 的 `datetime` （如 2020 年 10 月 12 日 20 点 10 分 01 秒），下面 Lemon 来跟大家介绍下具体的构造过程。

2.1.1 日期（date）实例的构造

date 是一个理想化的简单型日期，属性有 `year` , `month` , `day` 。

```
# 构造日期 date 实例
d1 = datetime.date(2020,10,12)
d1
```

```
datetime.date(2020, 10, 12)
```

```
# 构造日期 date 实例
d1 = datetime.date(2020,10,12)
print(d1)
```

```
2020-10-12
```

除了上面的构造方式，在 date 实例化的时候，还可以通过 `date.today()` 来构造当天的日期。

```
datetime.date.today()
```

```
datetime.date(2020, 10, 14)
```

`date` 类型的日期，可以通过 `.year` , `.month` , `.day` 来获取日期所属的年份，月份，和具体的日期号，这几个方法在数据分析中经常会用到。

```
# 获取日期所属的 年份，月份，和具体的日期号

```

```
print(f'日期所属的年份为：{d1.year}')
print(f'日期所属的月份为：{d1.month}')
print(f'日期具体的日期号为：{d1.day}')
```

```
日期所属的年份为：2020
日期所属的月份为：10
日期具体的日期号为：12
```

2.1.2 时间 time 实例的构造

time 是一个独立于任何特定日期的理想化时间，其属性有 hour，minute，second，microsecond 和 tzinfo 。

```
# 构造时间 time 实例
t1 = datetime.time(20,10,1)
t1
```

```
datetime.time(20, 10, 1)
```

```
# 获取日期所属的 年份，月份，和具体的日期号

print(f'time 所属的小时为：{t1.hour}')
print(f'time 所属的分钟为：{t1.minute}')
print(f'time 所属的秒为：{t1.second}')
```

```
time 所属的小时为：20
time 所属的分钟为：10
time 所属的秒为：1
```

2.1.3 datetime 实例的构造

datetime 是日期和时间的结合，其属性有 year，month，day，hour，minute，second，microsecond 和 tzinfo 。

```
# 构造时间 datetime 实例
dt1 = datetime.datetime(2020,10,11,20,10,1)
dt1
```

```
datetime.datetime(2020, 10, 11, 20, 10, 1)
```

除了上面的构造方式，在 datetime 实例化的时候，还有其他的一些方式，包括使用 datetime.now() 和 datetime.today()，以及在 date 的基础上使用 combine 方法等。

```
dt2 = datetime.datetime.now()
dt2
```

```
datetime.datetime(2020, 10, 14, 15, 12, 20, 303269)
```

```
dt3 = datetime.datetime.today()
dt3
```

```
datetime.datetime(2020, 10, 14, 15, 12, 20, 308733)
```

```
dt4 = datetime.datetime.combine(d1,t1)
dt4
```

```
datetime.datetime(2020, 10, 12, 20, 10, 1)
```

通过 datetime 的实例化，是我们使用时间是经常用到的方法，在日常使用过程中，我们可能只需要具体到某天，或者只需要具体到某天的某个时间点，这时候，也可以通过 datetime 的一些方法来实现。

```
# 从 datetime 来获取日期
dt4.date()
```

```
datetime.date(2020, 10, 12)
```

```
# 从 datetime 来获取具体的时间点
dt4.time()
```

```
datetime.time(20, 10, 1)
```

同样的 `datetime` 类型的时间，可以通过 `.year`，`.month`，`.day` 来获取日期所属的年份，月份，和具体的日期号。

```
# 获取日期所属的 年份，月份，和具体的日期号

print(f'日期所属的年份为：{dt4.year}')
print(f'日期所属的月份为：{dt4.month}')
print(f'日期具体的日期号为：{dt4.day}')
```

```
日期所属的年份为：2020
日期所属的月份为：10
日期具体的日期号为：12
```

还有一个可能涉及到的时间是获取某天属于星期几，可以通过 `weekday()` 和 `isoweekday()` 方法来实现。

```
# 从 datetime 来获取日期是星期几
# 使用 weekday 方法
# 数字从0开始，0代表星期一，1代表星期二，以此类推
dt4.weekday()
```

```
0
```

```
# 从 datetime 来获取日期是星期几
# 使用 isoweekday 方法
# 数字从1开始，1代表星期一，2代表星期二，以此类推
dt4.isoweekday()
```

```
1
```

datetime 还有一种方法，在这里也值得介绍下，如果 datetime 的值由于某些原因弄错了，我们也可以通过 `replace()` 方法来进行更正。这个方法在进行数据清洗的时候会有用。

replace 方法的参数如下：

```
datetime.replace(year=self.year, month=self.month, day=self.day, hour=self.hour, minute=self.minute, second=self.second, microsecond=self.microsecond, tzinfo=self.tzinfo, * fold=0)
```

```
dt5 = dt4.replace(year=2019)
dt5
```

```
datetime.datetime(2019, 10, 12, 20, 10, 1)
```

2.1.4 timedelta 对象的构造

timedelta 对象表示两个 date 或者 time 或者 datetime 的时间间隔。

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

所有参数都是可选的并且默认为 0。这些参数可以是整数或者浮点数，也可以是正数或者负数。

只有 days, seconds 和 microseconds 会存储在内部。参数单位的换算规则如下：

- 1 毫秒会转换成 1000 微秒。
- 1 分钟会转换成 60 秒。
- 1 小时会转换成 3600 秒。
- 1 星期会转换成 7 天。

```
delta = datetime.timedelta(weeks=3,
                           days=10,
                           hours=6,
                           minutes=50,
                           seconds=30,
                           microseconds=1000,
                           milliseconds=10000,
                          )

delta
```

```
datetime.timedelta(days=31, seconds=24640, microseconds=1000)
```

2.1.5 tzinfo 介绍

datetime.tzinfo 返回 datetime 对象的时区，前提是在创建 datetime 对象时需传入 tzinfo 参数，如果没有传入则返回值为 None 。

```
# 如果没有 pytz 库，则需要自行安装
import pytz

sh = pytz.timezone('Asia/Shanghai')
d_tz = datetime.datetime(2020,10,12,hour=8,tzinfo=sh)

d_tz.tzinfo
```

```
<DstTzInfo 'Asia/Shanghai' LMT+8:06:00 STD>
```

2.2 时间转换

时间的三种存在方式：时间对象，时间字符串，时间戳。

时间对象，比如前面介绍的 date 、datetime 、time 对象等；时间字符串，如：“2020-10-12”；时间戳，如 time.time() 返回的就是时间戳。

在数据处理过程中，经常会遇到需要将不同形式的时间进行转换。这里给大家介绍下常用的方法：

2.2.1 时间对象转字符串

时间对象转换为字符串，可以通过 isoformat 或 strftime 方法来实现。

strftime 的英文全称是 str format time ，根据给定的格式将时间对象转换为字符串

```
# 将 date 时间对象转换为字符串
d1 = datetime.date(2020,10,12)
d1.isoformat()
```

```
'2020-10-12'
```

```
# 用 srtftime 来转换
# YYYY-MM-DD 形式
d1.strftime('%Y-%m-%d')
```

```
'2020-10-12'
```

```
# MM DD, YYYY 形式
d1.strftime('%b %d, %Y')
```

```
'Oct 12, 2020'
```

```
# 将 time 时间对象转换为字符串
t1 = datetime.time(20,10,1)
t1.strftime('%H:%M:%S')
```

```
'20:10:01'
```

```
# 将 datetime 时间对象转换为字符串
dt2 = datetime.datetime.now()
dt2.strftime('%Y-%m-%d %H:%M:%S')
```

```
'2020-10-14 15:12:20'
```

```
dt2.isoformat()
```

```
'2020-10-14T15:12:20.403113'
```

python 中常见的时间日期格式化符号：

指令	意义	示例
%a	当地工作日的缩写。	Sun, Mon, …, Sat (en_US);So, Mo, …, Sa (de_DE)
%A	本地化的星期中每日的完整名称。	Sunday, Monday, …, Saturday (en_US);Sonntag, Montag, …, Samstag (de_DE)
%w	以十进制数显示的工作日，其中 0 表示星期日，6 表示星期六。	0, 1, …, 6
%d	补零后，以十进制数显示的月份中的一天。	01, 02, …, 31
%b	当地月份的缩写。	Jan, Feb, …, Dec (en_US);Jan, Feb, …, Dez (de_DE)
%B	本地化的月份全名。	January, February, …, December (en_US);Januar, Februar, …, Dezember (de_DE)
%m	补零后，以十进制数显示的月份。	01, 02, …, 12
%y	补零后，以十进制数表示的，不带世纪的年份。	00, 01, …, 99
%Y	十进制数表示的带世纪的年份。	0001, 0002, …, 2013, 2014, …, 9998, 9999
%H	以补零后的十进制数表示的小时（24 小时制）。	00, 01, …, 23
%I	以补零后的十进制数表示的小时（12 小时制）。	01, 02, …, 12
%p	本地化的 AM 或 PM 。	AM, PM (en_US);am, pm (de_DE)
%M	补零后，以十进制数显示的分钟。	00, 01, …, 59

指令	意义	示例
%S	补零后，以十进制数显示的秒。	00, 01, …, 59
%f	以十进制数表示的微秒，在左侧补零。	000000, 000001, …, 999999
%z	UTC 偏移量，格式为 ±HHMM[SS[.ffffff]]（如果是简单型对象则为空字符串）。	(空), +0000, -0400, +1030, +063415, -030712.345216
%Z	时区名称（如果对象为简单型则为空字符串）。	(空), UTC, GMT
%j	以补零后的十进制数表示的一年中的日序号。	001, 002, …, 366
%U	以补零后的十进制数表示的一年中的周序号（星期日作为每周的第一天）。在新的一年中第一个星期日之前的所有日子都被视为是在第 0 周。	00, 01, …, 53
%W	以十进制数表示的一年中的周序号（星期一作为每周的第一天）。在新的一年中第一个第期一之前的所有日子都被视为是在第 0 周。	00, 01, …, 53
%c	本地化的适当日期和时间表示。	Tue Aug 16 21:30:00 1988 (en_US);Di 16 Aug 21:30:00 1988 (de_DE)
%x	本地化的适当日期表示。	08/16/88 (None);08/16/1988 (en_US); 16.08.1988 (de_DE)
%X	本地化的适当时间表示。	21:30:00 (en_US);21:30:00 (de_DE)
%%	字面的 '%' 字符。	%

2.2.2 字符串转时间对象

字符串转时间对象，用的是 strptime 方法，与 strftime 方法刚好相反。

strptime 的英文全称是 str parse time ，将字符串解析为给定相应格式的时间对象。

```
s1 = '2020-10-09'
d = datetime.datetime.strptime(s1,'%Y-%m-%d')
d
```

```
datetime.datetime(2020, 10, 9, 0, 0)
```

下面提供了 strftime 方法与 strptime 方法的比较：

	strftime	strptime
用法	根据给定的格式将对象转换为字符串	将字符串解析为给定相应格式的 datetime对象
方法类型	实例方法	类方法

	strftime	strptime
方法	date; datetime; time	datetime
签名	strftime(format)	strptime(date_string, format)

需要注意的是，strftime 方法可供 date、time 和 datetime 对象使用，而 strptime 方法仅供 datetime 对象使用。

2.2.3 时间戳转换为时间对象

时间戳是指格林威治时间 1970 年 01 月 01 日 00 时 00 分 00 秒 (北京时间 1970 年 01 月 01 日 08 时 00 分 00 秒) 起至现在的总秒数。

```
# 获取当前的时间戳
ts_1 = time.time()
ts_1
```

```
1602660751.071785
```

```
# 获取当天00:00:00的时间戳
ts_2 = int(time.time()/86400)*86400
ts_2
```

```
1602633600
```

```
# 获取当天23:59:59的时间戳
# 一天有 24*60*60 = 86400 秒
ts_3 = int(time.time()/86400)*86400+86400-1
ts_3
```

```
1602719999
```

```
# 将时间戳转换为时间对象
datetime.datetime.fromtimestamp(ts_1)
```

```
datetime.datetime(2020, 10, 14, 15, 32, 31, 71785)
```

```
# 将时间戳转换为时间对象
datetime.date.fromtimestamp(ts_1)
```

```
datetime.date(2020, 10, 14)
```

2.2.4 将时间对象转换为时间戳

鉴于，时间戳是指格林威治时间 1970 年 01 月 01 日 00 时 00 分 00 秒 (北京时间 1970 年 01 月 01 日 08 时 00 分 00 秒) 起至现在的总秒数。

因此，将时间对象转换为时间戳时，直接计算两个时间对象的 timedelta ，并将 timedelta 以 “秒” 来表示就可以了。

```
dt1
```

```
datetime.datetime(2020, 10, 11, 20, 10, 1)
```

```
# 注意这里要用 北京时间
dt_s = datetime.datetime(1970,1,1,8)
dt_s
```

```
datetime.datetime(1970, 1, 1, 8, 0)
```

```
timedelta_1 = dt1 -dt_s
# 返回时间间隔包含了多少秒
timedelta_s = timedelta_1.total_seconds()
timedelta_s
```

```
1602418201.0
```

这里我们来反推下，看我们将时间对象转换为时间戳，是否正确。

```
# 将时间戳转换为时间对象
datetime.datetime.fromtimestamp(timedelta_s)
```

```
datetime.datetime(2020, 10, 11, 20, 10, 1)
```

2.3 时间对象的运算

2.3.1 获取当天最小时间和最大时间

```
# 获取当天最小时间
datetime.datetime.combine(datetime.date.today(),datetime.time.min)
```

```
datetime.datetime(2020, 10, 14, 0, 0)
```

```
# 获取当天最大时间
datetime.datetime.combine(datetime.date.today(),datetime.time.max)
```

```
datetime.datetime(2020, 10, 14, 23, 59, 59, 999999)
```

2.3.2 获取当前日期的前几天/后几天

```
# 获取明天
datetime.date.today() + datetime.timedelta(days=1)
```

```
datetime.date(2020, 10, 15)
```

```
# 获取明天
datetime.date.today() - datetime.timedelta(days=1)
```

```
datetime.date(2020, 10, 13)
```

2.3.3 获取本周或本月第一天及最后一天

```
d_today = datetime.date.today()
d_today
```

```
datetime.date(2020, 10, 14)
```

```
# 获取本周第一天
d_today - datetime.timedelta(d_today.weekday())
```

```
datetime.date(2020, 10, 12)
```

```
# 获取本周最后一天
d_today + datetime.timedelta(6-d_today.weekday())
```

```
datetime.date(2020, 10, 18)
```

2.3.4 计算两个日期相差多少天

```
# timedelta 对象的计算
td1 = dt2 - dt1
td1
```

```
datetime.timedelta(days=2, seconds=68539, microseconds=403113)
```

```
td1.days
```

```
2
```

注意下，如果需要计算两个日期之间总共相差多少秒，应该用 `total_seconds()` 方法。

```
td1.seconds
```

```
68539
```

```
td1.total_seconds()
```

```
241339.403113
```

参考文档：

- https://docs.python.org/zh-cn/3/library/datetime.html#strftime-strptime-behavior
- https://segmentfault.com/a/1190000012112097
- https://blog.csdn.net/qq_34493908/article/details/80888052
- https://zhuanlan.zhihu.com/p/96384066

3 Time

3.1 模块介绍

time 模块提供了各种时间相关的函数。

由于时间处理涉及到时区相关的知识，在开始之前，有一些术语和惯例需要阐述下：

- UTC 是协调世界时（以前称为格林威治标准时间，或 GMT）。缩写 UTC 不是错误，而是英语和法语之间的妥协。在中国为 UTC+8。
- DST 是夏令时，在一年中的一部分时间（通常）调整时区一小时。DST 规则很神奇（由当地法律确定），并且每年都会发生变化。C 库有一个包含本地规则的表（通常是从系统文件中读取以获得灵活性），并且在这方面是 True Wisdom 的唯一来源。
- 时间戳是指格林威治时间 1970 年 01 月 01 日 00 时 00 分 00 秒 (北京时间 1970 年 01 月 01 日 08 时 00 分 00 秒) 起至现在的总秒数。
- 时间元组 (struct_time) 方式：struct_time 元组共有 9 个元素，返回 struct_time 的函数主要有 gmtime()，localtime()，strptime()。下面列出这种方式元组中的几个元素：

索引（Index）	属性（Attribute）	值（Values）
0	tm_year（年）	比如 2020
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
6	tm_wday（weekday）	0 - 6（0 表示周日）
7	tm_yday（一年中的第几天）	1 - 366
8	tm_isdst（是否是夏令时）	默认为-1

3.2 时间获取

3.2.1 获取时间戳

time.time() 返回当前时间的时间戳

```
import time
```

```
# 返回当前时间的时间戳
t1 = time.time()
t1
```

```
1602728783.2113311
```

3.2.2 获取时间元组

```
# 返回当前时间的时间元组 （struct_time）
t2 = time.localtime()
t2
```

```
time.struct_time(tm_year=2020, tm_mon=10, tm_mday=15, tm_hour=10, tm_min=26,
    tm_sec=38, tm_wday=3, tm_yday=289, tm_isdst=0)
```

```
# 返回当前时间的 UTC 时区的时间元组 （struct_time）
t3 = time.gmtime()
t3
```

```
time.struct_time(tm_year=2020, tm_mon=10, tm_mday=15, tm_hour=2, tm_min=26,
    tm_sec=49, tm_wday=3, tm_yday=289, tm_isdst=0)
```

struct_time 元组共有 9 个元素，如果想自定义构造时间元组，则可以通过传入一个数组来实现，如下：

```
t3_1 = time.struct_time([2020,10,17,2,2,34,4,89,0])
t3_1
```

```
time.struct_time(tm_year=2020, tm_mon=10, tm_mday=17, tm_hour=2, tm_min=2,
    tm_sec=34, tm_wday=4, tm_yday=89, tm_isdst=0)
```

3.3 时间转换

3.3.1 时间元组转换为时间戳

```
# 将一个 struct_time 转化为时间戳
t4 = time.mktime(t2)
t4
```

```
1602728798.0
```

3.3.2 时间元组为字符串

time.asctime() 转换由 gmtime() 或 localtime() 所返回的表示时间的元组或 struct_time 为以下形式的字符串: “Sun Jun 20 23:21:05 1993”。

简单理解，转换时间元组为字符串形式。

```
# 转换时间元组为字符串形式
t5 = time.asctime(t2)
t5
```

```
'Thu Oct 15 10:26:38 2020'
```

如果想自定义字符串的格式，则可以通过 `strftime()` 来实现。`strftime()` 可以将时间元组转换为由 format 参数指定的字符串。

```
t5_1 = time.strftime('%Y-%m-%d',t2)
t5_1
```

```
'2020-10-15'
```

3.3.3 时间戳转换为字符串

time.ctime() 转换以距离初始纪元的秒数表示的时间为以下形式的字符串: “Sun Jun 20 23:21:05 1993” 代表本地时间。

简单理解，就是转换时间戳为字符串形式。

```
# 转换时间戳为字符串形式
t6 = time.ctime(t4)
t6
```

```
'Thu Oct 15 10:26:38 2020'
```

3.3.4 字符串转换为时间元组

`strptime()` 方式可以将特定格式的字符串解析为时间元组。字符串的默认匹配格式为 ”%a %b %d %H:%M:%S %Y” 。

```
time.strptime('Thu Oct 15 10:30:18 2020')
```

```
time.struct_time(tm_year=2020, tm_mon=10, tm_mday=15, tm_hour=10, tm_min=30,
    tm_sec=18, tm_wday=3, tm_yday=289, tm_isdst=-1)
```

指令表格

指令	意义
%a	本地化的缩写星期中每日的名称。
%A	本地化的星期中每日的完整名称。
%b	本地化的月缩写名称。
%B	本地化的月完整名称。
%c	本地化的适当日期和时间表示。
%d	十进制数 [01,31] 表示的月中日。
%H	十进制数 [00,23] 表示的小时（24 小时制）。
%I	十进制数 [01,12] 表示的小时（12 小时制）。
%j	十进制数 [001,366] 表示的年中日。
%m	十进制数 [01,12] 表示的月。
%M	十进制数 [00,59] 表示的分钟。
%p	本地化的 AM 或 PM 。
%S	十进制数 [00,61] 表示的秒。
%U	十进制数 [00,53] 表示的一年中的周数（星期日作为一周的第一天）作为。在第一个星期日之前的新年中的所有日子都被认为是在第 0 周。
%w	十进制数 [0(星期日),6] 表示的周中日。
%W	十进制数 [00,53] 表示的一年中的周数（星期一作为一周的第一天）作为。在第一个星期一之前的新年中的所有日子被认为是在第 0 周。
%x	本地化的适当日期表示。
%X	本地化的适当时间表示。
%y	十进制数 [00,99] 表示的没有世纪的年份。
%Y	十进制数表示的带世纪的年份。
%z	时区偏移以格式 +HHMM 或 -HHMM 形式的 UTC/GMT 的正或负时差指示，其中 H 表示十进制小时数字，M 表示小数分钟数字 [-23:59, +23:59] 。
%Z	时区名称（如果不存在时区，则不包含字符）。
%%	字面的 '%' 字符。

参考文档：

- https://docs.python.org/zh-cn/3/library/time.html#module-time

4 Calendar

Python 内置的日历（calendar）模块主要输出跟日历相关的内容，相对 datetime 和 time 模块来说，calendar 模块的功能要简单些。

最常用的两个功能是输出某年的日历，以及某个月的日历。

下面跟大家介绍下常见的情况：

4.1 获取某年的日历

```
import calendar
```

```
# 获取某年的日历并打印出来
calendar.prcal(2020)
```

```
                                  2020

      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
       1  2  3  4  5                      1  2                         1
 6  7  8  9 10 11 12       3  4  5  6  7  8  9       2  3  4  5  6  7  8
13 14 15 16 17 18 19      10 11 12 13 14 15 16       9 10 11 12 13 14 15
20 21 22 23 24 25 26      17 18 19 20 21 22 23      16 17 18 19 20 21 22
27 28 29 30 31            24 25 26 27 28 29         23 24 25 26 27 28 29
                                                    30 31

       April                     May                       June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
       1  2  3  4  5                   1  2  3       1  2  3  4  5  6  7
 6  7  8  9 10 11 12       4  5  6  7  8  9 10       8  9 10 11 12 13 14
13 14 15 16 17 18 19      11 12 13 14 15 16 17      15 16 17 18 19 20 21
20 21 22 23 24 25 26      18 19 20 21 22 23 24      22 23 24 25 26 27 28
27 28 29 30               25 26 27 28 29 30 31      29 30

        July                    August                  September
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
       1  2  3  4  5                      1  2          1  2  3  4  5  6
 6  7  8  9 10 11 12       3  4  5  6  7  8  9       7  8  9 10 11 12 13
13 14 15 16 17 18 19      10 11 12 13 14 15 16      14 15 16 17 18 19 20
20 21 22 23 24 25 26      17 18 19 20 21 22 23      21 22 23 24 25 26 27
```

```
27 28 29 30 31            24 25 26 27 28 29 30      28 29 30
                          31

      October                   November                  December
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
          1  2  3  4                         1          1  2  3  4  5  6
 5  6  7  8  9 10 11       2  3  4  5  6  7  8       7  8  9 10 11 12 13
12 13 14 15 16 17 18       9 10 11 12 13 14 15      14 15 16 17 18 19 20
19 20 21 22 23 24 25      16 17 18 19 20 21 22      21 22 23 24 25 26 27
26 27 28 29 30 31         23 24 25 26 27 28 29      28 29 30 31
                          30
```

4.2 获取某月的日历

```
calendar.month(2020,10)
```

```
'    October 2020\nMo Tu We Th Fr Sa Su\n          1  2  3  4\n 5  6  7  8  9
   10 11\n12 13 14 15 16 17 18\n19 20 21 22 23 24 25\n26 27 28 29 30 31\n'
```

这里需要注意下，在没有使用行数 `print()` 的情况下，输出的是原始的字符串形式。用 `print()` 输出后，显示如下：

```
print(calendar.month(2020,10))
```

```
    October 2020
Mo Tu We Th Fr Sa Su
          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
```

也可以用 `prmonth()` 函数将结果直接打印出来，效果也是一样的。

```
calendar.prmonth(2020,10)
```

```
    October 2020
Mo Tu We Th Fr Sa Su
          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
```

4.3 其他方法

calendar.monthcalendar()

返回表示一个月的日历的矩阵。每一行代表一周；此月份外的日子由零表示。每周从周一开始，除非使用 setfirstweekday() 改变设置。

```
print(calendar.monthcalendar(2020,10))
```

```
[[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]]
```

calendar.weekday()

返回某天是星期几，默认情况下 0-6 代表周一到周日。

```
# 获取日期是星期几 (0-6 代表周一到周日)
print(calendar.weekday(2020,10,15))
```

```
3
```

```
calendar.setfirstweekday(weekday)
```

设置每一周的开始 (0 表示星期一，6 表示星期天)。calendar 还提供了 MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY 和 SUNDAY 几个常量方便使用。例如，设置每周的第一天为星期天

```
import calendar
calendar.setfirstweekday(calendar.SUNDAY)
```

calendar.weekheader(n)

返回一个包含星期几的缩写名的头。n 指定星期几缩写的字符宽度。

```
print(calendar.weekheader(3))
```

```
Mon Tue Wed Thu Fri Sat Sun
```

5 延伸阅读

5.1 我是谁

大家好，我是 Lemon ，公众号「Python 数据之道」号主。

公众号「Python 数据之道」秉承“让数据更有价值”的理念，主要分享数据相关的内容，包括数据分析，挖掘，可视化，机器学习，深度学习等，希望能给大家分享有价值的内容。

若对我写的内容有兴趣，欢迎大家通过以下途径来关注。

5.1.1 微信公众号

Figure 5.1: Python 数据之道

「Python 数据之道」是我分享关于 Python 及数据分析相关内容的主阵地。

此外，给大家推荐下我的另外一个公众号 **「价值前瞻」（ID：ValueLab）**，主要分享读书、成长和投资思考，可以了解不一样的我，欢迎扫码关注。

Figure 5.2: 价值前瞻

5.1.2 个人网站

网址：http://liyangbit.com

Lemon 的个人网站中，包含更多的文章，并且在不断的进行更新。目前，网站中涉及了 Python 相关一系列内容，包括 Python 基础、Python 数据科学、项目实战等内容，欢迎访问。

5.2 推荐内容

5.2.1 《Python 知识手册》

「Python 数据之道」整理并出品了《Python 知识手册》，大家可以在公众号「Python 数据之道」后台回复数字**「600」**来获取高清 PDF 版。

Figure 5.3: Python 知识手册

5.2.2 精选文章

- 用 Python 可视化神器 Plotly 动态演示全球疫情变化趋势
- 用 Plotly 动态柱状图来演示全球疫情变化趋势

- 超火动态排序疫情变化图，这次我们用 Plotly 来绘制
- 用 Python 动态曲线图来对全球疫情进行演示
- 升级版，用 Python 来进行多条曲线动态演示全球疫情变化
- 深度好文｜ Matplotlib 可视化最有价值的 50 个图表（附完整 Python 源代码）
- 用 Python 读取巴菲特近期持仓数据
- 推荐一个牛逼的生物信息 Python 库 - Dash Bio
- 轻松用 Seaborn 进行数据可视化
- 用 Python 快速分析和预测股票价格
- 干货推荐: 轻松玩转 Bokeh 可视化（项目实战经验分享）
- 巧用 Matplotlib 动画, 让你的 Python 可视化大放异彩