**Python datetime模块详解、示例**

**一、datetime模块介绍**

**（一）、datetime模块中包含如下类：**

| **类名** | **功能说明** |
| --- | --- |
| date | 日期对象,常用的属性有year, month, day |
| time | 时间对象 |
| datetime | 日期时间对象,常用的属性有hour, minute, second, microsecond |
| datetime\_CAPI | 日期时间对象C语言接口 |
| timedelta | 时间间隔，即两个时间点之间的长度 |
| tzinfo | 时区信息对象 |

**（二）、datetime模块中包含的常量**

| **常量** | **功能说明** | **用法** | **返回值** |
| --- | --- | --- | --- |
| MAXYEAR | 返回能表示的最大年份 | datetime.MAXYEAR | 9999 |
| MINYEAR | 返回能表示的最小年份 | datetime.MINYEAR | 1 |

**二、date类**

**（一）、date对象构成**

**1、date对象由year年份、month月份及day日期三部分构成：**

date（year，month，day)

**2、 通过year, month, day三个数据描述符可以进行访问：**

>>> a = datetime.date.today()

>>> a

datetime.date(2017, 3, 22)

>>> a.year

2017

>>> a.month

3

>>> a.day

22

**3、当然，你也可以用\_\_getattribute\_\_(...)方法获得上述值：**

>>> a.\_\_getattribute\_\_('year')

2017

>>> a.\_\_getattribute\_\_('month')

3

>>> a.\_\_getattribute\_\_('day')

22

**（二）、date对象中包含的方法与属性**

**1、用于日期比较大小的方法**

| **方法名** | **方法说明** | **用法** |
| --- | --- | --- |
| \_\_eq\_\_(…) | 等于(x==y) | x.\_\_eq\_\_(y) |
| \_\_ge\_\_(…) | 大于等于(x>=y) | x.\_\_ge\_\_(y) |
| \_\_gt\_\_(…) | 大于(x>y) | x.\_\_gt\_\_(y) |
| \_\_le\_\_(…) | 小于等于(x<=y) | x.\_\_le\_\_(y) |
| \_\_lt\_\_(…) | 小于(x | x.\_\_lt\_\_(y) |
| \_\_ne\_\_(…) | 不等于(x!=y) | x.\_\_ne\_\_(y) |

以上方法的返回值为True\False  
**示例如下：**

>>> a=datetime.date(2017,3,1)

>>> b=datetime.date(2017,3,15)

>>> a.\_\_eq\_\_(b)

False

>>> a.\_\_ge\_\_(b)

False

>>> a.\_\_gt\_\_(b)

False

>>> a.\_\_le\_\_(b)

True

>>> a.\_\_lt\_\_(b)

True

>>> a.\_\_ne\_\_(b)

True

**2、获得二个日期相差多少天**

使用**\_\_sub\_\_(...)**和**\_\_rsub\_\_(...)**方法，其实二个方法差不太多，一个是正向操作，一个是反向操作：

| **方法名** | **方法说明** | **用法** |
| --- | --- | --- |
| \_\_sub\_\_(…) | x - y | x.\_\_sub\_\_(y) |
| \_\_rsub\_\_(…) | y - x | x.\_\_rsub\_\_(y) |

**示例如下:**

>>> a

datetime.date(2017, 3, 22)

>>> b

datetime.date(2017, 3, 15)

>>> a.\_\_sub\_\_(b)

datetime.timedelta(7)

>>> a.\_\_rsub\_\_(b)

datetime.timedelta(-7)

计算结果的返回值类型为datetime.timedelta, 如果获得整数类型的结果则按下面的方法操作：

>>> a.**\_\_sub\_\_**(b).days

7

>>> a.**\_\_rsub\_\_**(b).days

-7

**3、ISO标准化日期**

如果想要让所使用的日期符合ISO标准，那么使用如下三个方法:  
1).*\* isocalendar(...)\**:返回一个包含三个值的元组，三个值依次为：year年份，week number周数，weekday星期数（周一为1…周日为7)：  
**示例如下**

>>> a = datetime.date(2017,3,22)

>>> a.isocalendar()

(2017, 12, 3)

>>> a.isocalendar()[0]

2017

>>> a.isocalendar()[1]

12

>>> a.isocalendar()[2]

3

2). **isoformat(...)**: 返回符合ISO 8601标准 (YYYY-MM-DD) 的日期字符串；  
**示例如下**

>>> a = datetime.date(2017,3,22)

>>> a.isoformat()

'2017-03-22'

3). **isoweekday(...)**: 返回符合ISO标准的指定日期所在的星期数（周一为1…周日为7)  
**示例如下：**

>>> a = datetime.date(2017,3,22)

>>> a.isoweekday()

3

4).与**isoweekday(...)**相似的还有一个**weekday(...)**方法，只不过是weekday(...)方法返回的周一为 0, 周日为 6  
**示例如下：**

>>> a = datetime.date(2017,3,22)

>>> a.weekday()

2

**4、其他方法与属性**

1). **timetuple(...)**:该方法为了兼容time.localtime(...)返回一个类型为time.struct\_time的数组，但有关时间的部分元素值为0：

>>> a = datetime.date(2017,3,22)

>>> a.timetuple()

time.struct\_time(tm\_year=2017, tm\_mon=3, tm\_mday=22, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=2, tm\_yday=81, tm\_isdst=-1)

>>> a.timetuple().tm\_year

2017

>>> a.timetuple().tm\_mon

3

>>> a.timetuple().tm\_mday

22

2).**toordinal(...)**： 返回公元公历开始到现在的天数。公元1年1月1日为1

>>> a = datetime.date(2017,3,22)

>>> a.toordinal()

736410

3). **replace(...)**：返回一个替换指定日期字段的新date对象。参数3个可选参数，分别为year,month,day。注意替换是产生新对象，不影响原date对象。

>>> a = datetime.date(2017,3,22)

>>> b = a.replace(2017,2,28)

>>> a

datetime.date(2017, 3, 22)

>>> b

datetime.date(2017, 2, 28)

4).**resolution**：date对象表示日期的最小单位。这里是天。

>>> datetime.date.resolution

datetime.timedelta(1)

5).**fromordinal(...)**：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。

>>> a = datetime.date(2017,3,22)

>>> b = a.toordinal()

>>> datetime.date.fromordinal(b)

datetime.date(2017, 3, 22)

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

>>> time.time()

1490165087.2242179

>>> datetime.date.fromtimestamp(time.time())

datetime.date(2017, 3, 22)

7).**today(...)**：返回当前日期

>>> datetime.date.today()

datetime.date(2017, 3, 22)

8).**max**： date类能表示的最大的年、月、日的数值

>>> datetime.date.max

datetime.date(9999, 12, 31)

9).**min**： date类能表示的最小的年、月、日的数值

>>> datetime.date.min

datetime.date(1, 1, 1)

**（三）、日期的字符串输出**

**1、如果你想将日期对象转化为字符串对象的话，可以用到\_\_format\_\_(...)方法以指定格式进行日期输出：**

>>> a = datetime.date(2017,3,22)

>>> a.\_\_format\_\_('%Y-%m-%d')

'2017-03-22'

>>> a.\_\_format\_\_('%Y/%m/%d')

'2017/03/22'

>>> a.\_\_format\_\_('%y/%m/%d')

'17/03/22'

>>> a.\_\_format\_\_('%D')

'03/22/17'

与此方法等价的方法为strftime(...)

>>> a.strftime("%Y%m%d")

'20170322'

关于格式化字符串的相关内容，请查阅本文最后的：**附录：python中时间日期格式化符号**  
**2、如果只是相简单的获得日期的字符串，则使用\_\_str\_\_(...)**

>>> a.\_\_str\_\_()

'2017-03-22'

**3、如果想要获得ctime样式的格式请使用ctime(...):**

>>> a.ctime()

'Wed Mar 22 00:00:00 2017'

**三、time类**

**(一)、time类的数据构成**

time类由hour小时、minute分钟、second秒、microsecond毫秒和tzinfo五部分组成

time([hour[, minute[, second[, microsecond[, tzinfo]]]]])

相应的，time类中就有上述五个变量来存储应该的值：

>>> a = datetime.time(12,20,59,899)

>>> a

datetime.time(12, 20, 59, 899)

>>> a.hour

12

>>> a.minute

20

>>> a.second

59

>>> a.microsecond

899

>>> a.tzinfo

与date类一样，time类也包含\_\_getattribute\_\_(...)方法可以读取相关属性：

>>> a.\_\_getattribute\_\_('hour')

12

>>> a.\_\_getattribute\_\_('minute')

20

>>> a.\_\_getattribute\_\_('second')

59

**（二）、time类中的方法和属性**

**1、比较时间大小**

相关方法包括：\_\_eq\_\_(...), \_\_ge\_\_(...), \_\_gt\_\_(...), \_\_le\_\_(...), \_\_lt\_\_(...)， \_\_ne\_\_(...)  
这里的方法与date类中定义的方法大同小异，使用方法与一样，这里就不过多介绍了，示例如下：

>>> a = datetime.time(12,20,59,899)

>>> b = datetime.time(11,20,59,889)

>>> a.\_\_eq\_\_(b)

False

>>> a.\_\_ne\_\_(b)

True

>>> a.\_\_ge\_\_(b)

True

>>> a.\_\_gt\_\_(b)

True

>>> a.\_\_le\_\_(b)

False

>>> a.\_\_lt\_\_(b)

False

**2、其他属性**

1）、**max**：最大的时间表示数值：

>>> datetime.time.max

datetime.time(23, 59, 59, 999999)

2）、**min**：最小的时间表示数值

>>> datetime.time.min

datetime.time(0, 0)

3）、**resolution**：时间间隔单位为分钟

>>> datetime.time.resolution

datetime.timedelta(0, 0, 1)

**（三）、时间的字符串输出**

**1、如果你想将时间对象转化为字符串对象的话，可以用到\_\_format\_\_(...)方法以指定格式进行时间输出：**

>>> a = datetime.time(12,20,59,899)

>>> a.\_\_format\_\_('%H:%M:%S')

'12:20:59'

与此方法等价的方法为strftime(...)

>>> a = datetime.time(12,20,59,899)

>>> a.strftime('%H:%M:%S')

'12:20:59'

关于格式化字符串的相关内容，请查阅本文最后的：**附录：python中时间日期格式化符号**  
**2、ISO标准输出**  
如果要使输出的时间字符符合ISO标准，请使用isoformat(...):

>>> a = datetime.time(12,20,59,899)

>>> a.isoformat()

'12:20:59.000899'

**3、如果只是相简单的获得时间的字符串，则使用\_\_str\_\_(...)**

>>> a = datetime.time(12,20,59,899)

>>> a.\_\_str\_\_()

'12:20:59.000899'

**四、datetime类**

**(一)、datetime类的数据构成**

datetime类其实是可以看做是date类和time类的合体，其大部分的方法和属性都继承于这二个类，相关的操作方法请参阅，本文上面关于二个类的介绍。其数据构成也是由这二个类所有的属性所组成的。

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

**（二）、专属于datetime的方法和属性**

**1、 date(…)：返回**[datetime](https://so.csdn.net/so/search?q=datetime&spm=1001.2101.3001.7020)**对象的日期部分：**

>>> a = datetime.datetime.now()

>>> a

datetime.datetime(2017, 3, 22, 16, 9, 33, 494248)

>>> a.date()

datetime.date(2017, 3, 22)

**2、time(…)：返回datetime对象的时间部分：**

>>> a = datetime.datetime.now()

>>> a

datetime.datetime(2017, 3, 22, 16, 9, 33, 494248)

>>> a.time()

datetime.time(16, 9, 33, 494248)

**3、utctimetuple(…)：返回UTC时间元组：**

>>> a = datetime.datetime.now()

>>> a

datetime.datetime(2017, 3, 22, 16, 9, 33, 494248)

>>> a.utctimetuple()

time.struct\_time(tm\_year=2017, tm\_mon=3, tm\_mday=22, tm\_hour=16, tm\_min=9, tm\_sec=33, tm\_wday=2, tm\_yday=81, tm\_isdst=0)

**4、combine(…)：将一个date对象和一个time对象合并生成一个datetime对象：**

>>> a = datetime.datetime.now()

>>> a

datetime.datetime(2017, 3, 22, 16, 9, 33, 494248)

>>>datetime.datetime.combine(a.date(),a.time())

datetime.datetime(2017, 3, 22, 16, 9, 33, 494248)

**5、now(…)：返回当前日期时间的datetime对象：**

>>> a = datetime.datetime.now()

>>> a

datetime.datetime(2017, 3, 22, 16, 9, 33,

**6、utcnow(…):返回当前日期时间的UTC datetime对象：**

>>> a = datetime.datetime.utcnow()

>>> a

datetime.datetime(2017, 3, 22, 8, 26, 54, 935242)

**7、strptime(…)：根据string, format 2个参数，返回一个对应的datetime对象：**

>>> datetime.datetime.strptime('2017-3-22 15:25','%Y-%m-%d %H:%M')

datetime.datetime(2017, 3, 22, 15, 25)

**8、utcfromtimestamp(…):UTC时间戳的datetime对象，时间戳值为time.time()：**

>>> datetime.datetime.utcfromtimestamp(time.time())

datetime.datetime(2017, 3, 22, 8, 29, 7, 654272)

**五、timedelta类**

timedelta类是用来计算二个datetime对象的差值的。  
此类中包含如下属性：  
**1、days**:天数  
**2、microseconds**：微秒数(>=0 并且 <1秒）  
**3、seconds**：秒数(>=0 并且 <1天）

**六、日期计算实操**

**1.获取当前日期时间：**

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

>>> now

datetime.datetime(2017, 3, 22, 16, 55, 49, 148233)

>>> today = datetime.date.today()

>>> today

datetime.date(2017, 3, 22)

>>> now.date()

datetime.date(2017, 3, 22)

>>> now.time()

datetime.time(16, 55, 49, 148233)

**2.获取上个月第一天和最后一天的日期：**

>>> today = datetime.date.today()

>>> today

datetime.date(2017, 3, 22)

>>> mlast\_day = datetime.date(today.year, today.month, 1) - datetime.timedelta(1)

>>> mlast\_day

datetime.date(2017, 2, 28)

>>> mfirst\_day = datetime.date(mlast\_day.year, mlast\_day.month, 1)

>>> mfirst\_day

datetime.date(2017, 2, 1)

**3.获取时间差**

时间差单位为秒

>>> start\_time = datetime.datetime.now()

>>> end\_time = datetime.datetime.now()

>>> (end\_time - start\_time).seconds

7

差值不只是可以查看相差多少秒，还可以查看天(days), 秒(seconds), 微秒(microseconds).

**4.计算当前时间向后8个小时的时间**

>>> d1 = datetime.datetime.now()

>>> d2 = d1 + datetime.timedelta(hours = 8)

>>> d2

datetime.datetime(2017, 3, 23, 1, 10, 37, 182240)

可以计算: 天(days), 小时(hours), 分钟(minutes), 秒(seconds), 微秒(microseconds).

**5.计算上周一和周日的日期**

today = datetime.date.today()

>>> today

datetime.date(2017, 3, 23)

>>> today\_weekday = today.isoweekday()

>>> last\_sunday = today - datetime.timedelta(days=today\_weekday)

>>> last\_monday = last\_sunday - datetime.timedelta(days=6)

>>> last\_sunday

datetime.date(2017, 3, 19)

>>> last\_monday

datetime.date(2017, 3, 13)

**6.计算指定日期当月最后一天的日期和本月天数**

>>> date = datetime.date(2017,12,20)

>>> def eomonth(date\_object):

... if date\_object.month == 12:

... next\_month\_first\_date = datetime.date(date\_object.year+1,1,1)

... else:

... next\_month\_first\_date = datetime.date(date\_object.year, date\_object.month+1, 1)

... return next\_month\_first\_date - datetime.timedelta(1)

...

>>> eomonth(date)

datetime.date(2017, 12, 31)

>>> eomonth(date).day

31

**7.计算指定日期下个月当天的日期**

这里要调用上一项中的函数eomonth(...)

>>> date = datetime.date(2017,12,20)

>>> def edate(date\_object):

... if date\_object.month == 12:

... next\_month\_date = datetime.date(date\_object.year+1, 1,date\_object.day)

... else:

... next\_month\_first\_day = datetime.date(date\_object.year,date\_object.month+1,1)

... if date\_object.day > eomonth(last\_month\_first\_day).day:

... next\_month\_date = datetime.date(date\_object.year,date\_object.month+1,eomonth(last\_month\_first\_day).day)

... else:

... next\_month\_date = datetime.date(date\_object.year, date\_object.month+1, date\_object.day)

... return next\_month\_date

...

>>> edate(date)

datetime.date(2018, 1, 20)

**8.获得本周一至今天的时间段并获得上周对应同一时间段**

>>> today = datetime.date.today()

>>> this\_monday = today - datetime.timedelta(today.isoweekday()-1)

>>> last\_monday = this\_monday - datetime.timedelta(7)

>>> last\_weekday = today -datetime.timedelta(7)

>>> this\_monday

datetime.date(2017, 3, 20)

>>> today

datetime.date(2017, 3, 23)

>>> last\_monday

datetime.date(2017, 3, 13)

>>> last\_weekday

datetime.date(2017, 3, 16)

**附录：python中时间日期格式化符号：**

| **符号** | **说明** |
| --- | --- |
| %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 |