## #os模块

os.sep 可以取代操作系统特定的路径分隔符。windows下为 '\\'

os.name 字符串指示你正在使用的平台。比如对于Windows，它是'nt'，而对于Linux/Unix用户，它是 'posix'

os.getcwd() 函数得到当前工作目录，即当前Python脚本工作的目录路径

os.getenv() 获取一个环境变量，如果没有返回none

os.putenv(key, value) 设置一个环境变量值

os.listdir(path) 返回指定目录下的所有文件和目录名

os.remove(path) 函数用来删除一个文件

os.system(command) 函数用来运行shell命令

os.linesep 字符串给出当前平台使用的行终止符。例如，Windows使用 '\r\n'，Linux使用 '\n' 而Mac使用 '\r'

os.path.split(path) 函数返回一个路径的目录名和文件名

os.path.isfile() 和os.path.isdir()函数分别检验给出的路径是一个文件还是目录

os.path.exists() 函数用来检验给出的路径是否真地存在

os.curdir 返回当前目录 ('.')

os.mkdir(path) 创建一个目录

os.makedirs(path) 递归的创建目录

os.chdir(dirname) 改变工作目录到dirname

os.path.getsize(name) 获得文件大小，如果name是目录返回0L

os.path.abspath(name) 获得绝对路径

os.path.normpath(path) 规范path字符串形式

os.path.splitext() 分离文件名与扩展名

os.path.join(path,name) 连接目录与文件名或目录

os.path.basename(path) 返回文件名

os.path.dirname(path) 返回文件路径

os.walk(top,topdown=True,onerror=None) 遍历迭代目录

os.rename(src, dst) 重命名file或者directory src到dst 如果dst是一个存在的directory, 将抛出OSError. 在Unix, 如果dst在存且是一个file, 如果用户有权限的话，它将被安静的替换. 操作将会失败在某些Unix 中如果src和dst在不同的文件系统中. 如果成功, 这命名操作将会是一个原子操作 (这是POSIX 需要). 在 Windows上, 如果dst已经存在, 将抛出OSError，即使它是一个文件. 在unix，Windows中有效。

os.renames(old, new) 递归重命名文件夹或者文件。像rename()

## # shutil 模块

shutil.copyfile( src, dst) 从源src复制到dst中去。当然前提是目标地址是具备可写权限。抛出的异常信息为IOException. 如果当前的dst已存在的话就会被覆盖掉。src和dst均是文件名。

shutil.move( src, dst) 移动文件或重命名

shutil.copymode( src, dst) 只是会复制其权限其他的东西是不会被复制的

shutil.copystat( src, dst) 复制权限、最后访问时间、最后修改时间

shutil.copy( src, dst) 复制一个文件到一个文件或一个目录

shutil.copy2( src, dst) 在copy上的基础上再复制文件最后访问时间与修改时间也复制过来了，类似于cp –p的东西

shutil.copy2( src, dst) 如果两个位置的文件系统是一样的话相当于是rename操作，只是改名；如果是不在相同的文件系统的话就是做move操作

shutil.copytree( olddir, newdir, True/Flase)

把olddir拷贝一份newdir，如果第3个参数是True，则复制目录时将保持文件夹下的符号连接，如果第3个参数是False，则将在复制的目录下生成物理副本来替代符号连接

shutil.rmtree( src ) 递归删除一个目录以及目录内的所有内容

## # time 模块

https://www.cnblogs.com/tkqasn/p/6001134.html

### 一、time模块

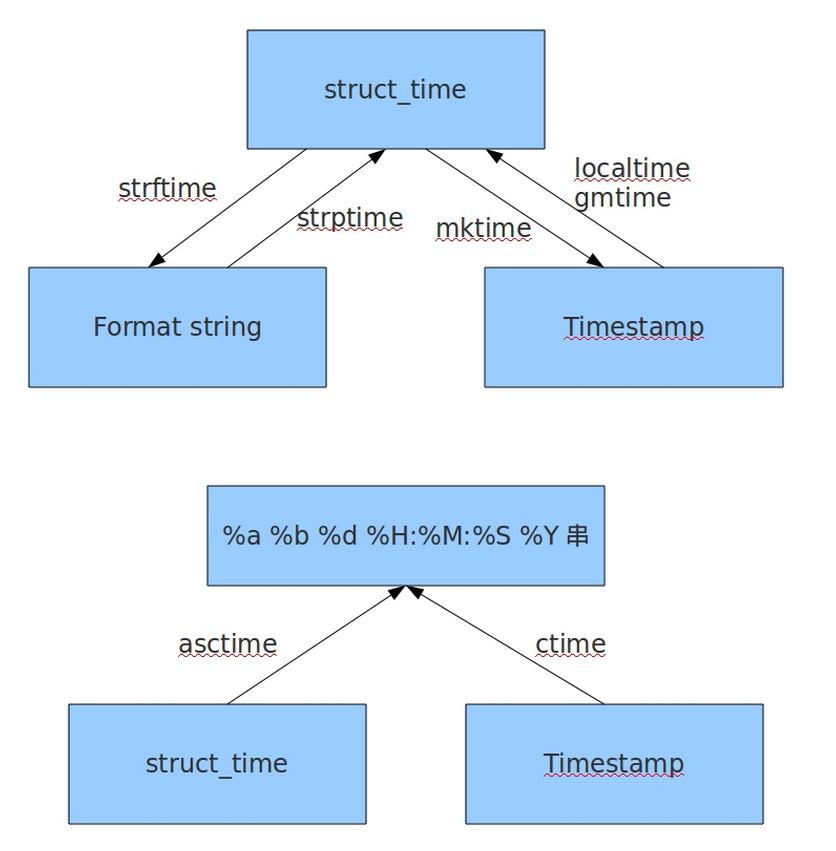
 time模块中时间表现的格式主要有三种：

　　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：**



**2、主要time生成方法和time格式转换方法实例：**

[复制代码](javascript:void(0);)

#! /usr/bin/env python

# -\*- coding:utf-8 -\*-

# \_\_author\_\_ = "TKQ"

import time

**# 生成timestamp**

time.time()

# 1477471508.05  
#struct\_time to timestamp  
time.mktime(time.localtime())

**#生成struct\_time**

# timestamp to struct\_time 本地时间

time.localtime()

time.localtime(time.time())

# time.struct\_time(tm\_year=2016, tm\_mon=10, tm\_mday=26, tm\_hour=16, tm\_min=45, tm\_sec=8, tm\_wday=2, tm\_yday=300, tm\_isdst=0)

# timestamp to struct\_time 格林威治时间

time.gmtime()

time.gmtime(time.time())

# time.struct\_time(tm\_year=2016, tm\_mon=10, tm\_mday=26, tm\_hour=8, tm\_min=45, tm\_sec=8, tm\_wday=2, tm\_yday=300, tm\_isdst=0)

#format\_time to struct\_time

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

# 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)

**#生成format\_time**

#struct\_time to format\_time

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

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

# 2016-10-26 16:48:41

**#生成固定格式的时间表示格式**

time.asctime(time.localtime())

time.ctime(time.time())

# Wed Oct 26 16:45:08 2016

[复制代码](javascript:void(0);)

**struct\_time元组元素结构**

[复制代码](javascript:void(0);)

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

[复制代码](javascript:void(0);)

**format time结构化表示**

|  |  |
| --- | --- |
| **格式** | **含义** |
| %a | 本地（locale）简化星期名称 |
| %A | 本地完整星期名称 |
| %b | 本地简化月份名称 |
| %B | 本地完整月份名称 |
| %c | 本地相应的日期和时间表示 |
| %d | 一个月中的第几天（01 - 31） |
| %H | 一天中的第几个小时（24小时制，00 - 23） |
| %I | 第几个小时（12小时制，01 - 12） |
| %j | 一年中的第几天（001 - 366） |
| %m | 月份（01 - 12） |
| %M | 分钟数（00 - 59） |
| %p | 本地am或者pm的相应符 |
| %S | 秒（01 - 61） |
| %U | 一年中的星期数。（00 - 53星期天是一个星期的开始。）第一个星期天之前的所有天数都放在第0周。 |
| %w | 一个星期中的第几天（0 - 6，0是星期天） |
| %W | 和%U基本相同，不同的是%W以星期一为一个星期的开始。 |
| %x | 本地相应日期 |
| %X | 本地相应时间 |
| %y | 去掉世纪的年份（00 - 99） |
| %Y | 完整的年份 |
| %Z | 时区的名字（如果不存在为空字符） |
| %% | ‘%’字符 |

常见结构化时间组合：

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

#2016-10-26 20:50:13

 3、time加减

[复制代码](javascript:void(0);)

#timestamp加减单位以秒为单位

import time

t1 = time.time()

t2=t1+10

print time.ctime(t1)#Wed Oct 26 21:15:30 2016

print time.ctime(t2)#Wed Oct 26 21:15:40 2016

[复制代码](javascript:void(0);)

### 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.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；

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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())

#Output======================

# date.max: 9999-12-31

# date.min: 0001-01-01

# date.today(): 2016-10-26

# date.fromtimestamp(): 2016-10-26

[复制代码](javascript:void(0);)

方法和属性

[复制代码](javascript:void(0);)

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

d1.isoformat()：返回格式如'YYYY-MM-DD’的字符串；

d1.strftime(fmt)：和time模块format相同。

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

from datetime import \*

now = date(2016, 10, 26)

tomorrow = now.replace(day = 27)

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()

print 'strftime():', now.strftime("%Y-%m-%d")

#Output========================

# now: 2016-10-26 , tomorrow: 2016-10-27

# timetuple(): time.struct\_time(tm\_year=2016, tm\_mon=10, tm\_mday=26, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=2, tm\_yday=300, tm\_isdst=-1)

# weekday(): 2

# isoweekday(): 3

# isocalendar(): (2016, 43, 3)

# isoformat(): 2016-10-26

# strftime(): 2016-10-26

[复制代码](javascript:void(0);)

**2、time类**

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

静态方法和字段

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；

方法和属性

[复制代码](javascript:void(0);)

t1 = datetime.time(10,23,15)#time对象  
t1.hour、t1.minute、t1.second、t1.microsecond：时、分、秒、微秒；

t1.tzinfo：时区信息；

t1.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）；

t1.isoformat()：返回型如"HH:MM:SS"格式的字符串表示；

t1.strftime(fmt)：同time模块中的format；

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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()

print 'strftime()', tm.strftime("%X")

#Output==============================================

# tm: 23:46:10

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

# tm1: 20:46:10

# isoformat(): 23:46:10

# strftime() 23:46:10

[复制代码](javascript:void(0);)

**3、datetime类**

datetime相当于date和time结合起来。  
datetime.datetime (year, month, day[ , hour[ , minute[ , second[ , microsecond[ , tzinfo] ] ] ] ] )

静态方法和字段

[复制代码](javascript:void(0);)

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.utcnow()：返回一个当前utc时间的datetime对象；#格林威治时间

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；

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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())

#output======================

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

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

# datetime.resolution: 0:00:00.000001

# today(): 2016-10-26 23:12:51.307000

# now(): 2016-10-26 23:12:51.307000

# utcnow(): 2016-10-26 15:12:51.307000

# fromtimestamp(tmstmp): 2016-10-26 23:12:51.307000

# utcfromtimestamp(tmstmp): 2016-10-26 15:12:51.307000

[复制代码](javascript:void(0);)

方法和属性

[复制代码](javascript:void(0);)

dt=datetime.now()#datetime对象

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

dt.date()：获取date对象；

dt.time()：获取time对象；

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

dt. timetuple ()

dt. utctimetuple ()

dt. toordinal ()

dt. weekday ()

dt. isocalendar ()

dt. isoformat ([ sep] )

dt. 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()))；

dt. strftime (format)

[复制代码](javascript:void(0);)

**4.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。

[复制代码](javascript:void(0);)

#coding:utf-8

from datetime import \*

dt = datetime.now()

#日期减一天

dt1 = dt + timedelta(days=-1)#昨天

dt2 = 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

print delta\_obj.days ,delta\_obj.total\_seconds()#1 86400.0

[复制代码](javascript:void(0);)

**5、tzinfo时区类**

[复制代码](javascript:void(0);)

#! /usr/bin/python

# coding=utf-8

from datetime import datetime, tzinfo,timedelta

"""

tzinfo是关于时区信息的类

tzinfo是一个抽象类，所以不能直接被实例化

"""

class UTC(tzinfo):

"""UTC"""

def \_\_init\_\_(self,offset = 0):

self.\_offset = offset

def utcoffset(self, dt):

return timedelta(hours=self.\_offset)

def tzname(self, dt):

return "UTC +%s" % self.\_offset

def dst(self, dt):

return timedelta(hours=self.\_offset)

#北京时间

beijing = datetime(2011,11,11,0,0,0,tzinfo = UTC(8))

print "beijing time:",beijing

#曼谷时间

bangkok = datetime(2011,11,11,0,0,0,tzinfo = UTC(7))

print "bangkok time",bangkok

#北京时间转成曼谷时间

print "beijing-time to bangkok-time:",beijing.astimezone(UTC(7))

#计算时间差时也会考虑时区的问题

timespan = beijing - bangkok

print "时差:",timespan

#Output==================

# beijing time: 2011-11-11 00:00:00+08:00

# bangkok time 2011-11-11 00:00:00+07:00

# beijing-time to bangkok-time: 2011-11-10 23:00:00+07:00

# 时差: -1 day, 23:00:00

[复制代码](javascript:void(0);)