Time

Time intervals are floating-point numbers in units of seconds. Particular instants in time are expressed in seconds since 12:00am, January 1, 1970.

The time module available in Python provides functions for working with times, and for converting between representations.

Example

```
import time;              # This is required to include time module.
ticks = time.time()
print "Number of ticks since 12:00am, January 1, 1970:", ticks
```

Date values can be stored from year 1970 to year 2038.

TimeTuple
Many of Python's time functions handle time as a tuple of 9 numbers, as shown below:

Index	Field	Values
0	4 digit year	2008
1	Month 1 to 12	1 to 12
2	Day 1 to 31	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 61 (60 and 61 are leap seconds)
6	Day of Week	0 to 6 (0 means Monday)
7	Day of Year	1 to 366
8	Daylight Savings	-1, 0, 1, -1

The above tuple is equivalent to struct_time structure which has following attributes.

Index	Field	Values
0	tm_year	2008
1	tm_mon	1 to 12
2	tm_mday	1 to 31
3	tm_hour	0 to 23
4	tm_min	0 to 59
5	tm_sec	0 to 61 (60 and 61 are leap seconds)
6	tm_wday	0 to 6 (0 means Monday)
7	tm_yday	1 to 366

8	tm_isdst	-1, 0, 1, -1

time.time() : The method time() returns the time as a floating point number expressed in seconds since Jan 1, 1970.
Example

```
time.time()
localtime = time.localtime(time.time())                 # current time

localtime = time.asctime( time.localtime(time.time()) )       # current time
```

time.clock() : returns current processor time in seconds

```
import time
t1 = time.clock()
time.sleep(5)
print(time.clock() - t1)
```

time.ctime() : convert a time expressed in seconds since Jan 1, 1970, 5.30am to a string representing local time

```
time.ctime(1)
time.ctime(100)
time.ctime(101)
time.ctime()              # current time
```

time.gmtime(sec) : The method gmtime() converts a time expressed in seconds since the epoch to a struct_time in UTC in which the dst flag is always zero
Example

```
time.gmtime(100)      # 100 sec from 1 Jan, 1970
time.asctime(time.gmtime(100))
time.gmtime()           # current time since no parameter is passed
time.asctime(time.gmtime())

print "time.gmtime() : %s" % time.gmtime()
```

time.localtime(sec) : Similar to gmtime() but it converts number of seconds to local time.
Example

```
time.localtime()
time.asctime(time.localtime())
time.localtime(100)
time.asctime(time.localtime(100))
```

time.mktime() : The argument to mktime is the struct_time or full 9-tuple and it returns a floating point number, for compatibility with time().

```
t = (2009, 2, 17, 17, 3, 38, 1, 48, 0)
time.mktime(t)                  # converts time tuple t to time (no of seconds)
time.asctime(time.localtime(time.mktime(t)))                  # converts time tuple t to time
```

time.sleep() : The method sleep() suspends execution for the given number of seconds.

```
time.sleep(100)
```

write following in a program file and execute

```
import time
print "Start : %s" % time.ctime()
time.sleep( 5 )
print "End : %s" % time.ctime()
```

time.strptime(str,fmt='%a %b %d %H:%M:%S %Y') : The method strptime() parses a string representing a time according to a format. The return value is a struct_time.
Syntax : time.strptime(string[, format])

Parameters

- string -- This is the time in string format which would be parsed based on the given format.
- format -- This is the directive which would be used to parse the given string.

The following directives can be embedded in the format string.
Directive

- %a - abbreviated weekday name
- %A - full weekday name
- %b - abbreviated month name
- %B - full month name
- %c - preferred date and time representation
- %C - century number (the year divided by 100, range 00 to 99)
- %d - day of the month (01 to 31)
- %D - same as %m/%d/%y
- %e - day of the month (1 to 31)
- %g - like %G, but without the century
- %G - 4-digit year corresponding to the ISO week number (see %V).
- %h - same as %b
- %H - hour, using a 24-hour clock (00 to 23)
- %I - hour, using a 12-hour clock (01 to 12)
- %j - day of the year (001 to 366)
- %m - month (01 to 12)
- %M - minute
- %n - newline character
- %p - either am or pm according to the given time value
- %r - time in a.m. and p.m. notation
- %R - time in 24 hour notation
- %S - second

- %t - tab character
- %T - current time, equal to %H:%M:%S
- %u - weekday as a number (1 to 7), Monday=1. Warning: In Sun Solaris Sunday=1
- %U - week number of the current year, starting with the first Sunday as the first day of the first week
- %V - The ISO 8601 week number of the current year (01 to 53), where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week
- %W - week number of the current year, starting with the first Monday as the first day of the first week
- %w - day of the week as a decimal, Sunday=0
- %x - preferred date representation without the time
- %X - preferred time representation without the date
- %y - year without a century (range 00 to 99)
- %Y - year including the century
- %Z or %z - time zone or name or abbreviation
- %% - a literal % character

Example
```
struct_time = time.strptime("30 Nov 00", "%d %b %y")
print(struct_time)
```

Calendar
```
import calendar
```

calendar.calendar() : calendar.calendar(year,w=2,l=1,c=6, m = 3)
Returns a multiline string with a calendar for year year formatted into three columns separated by c spaces. w is the width in characters of each date; each line has length 21*w+18+2*c. l is the number of lines for each week. M is no of months displayed horizontally.

Example
```
print(calendar.calendar(2014,w=2,l=1,c=6, m=2))
```

calendar.prcal(year,w=2,l=1,c=6) : Like print calendar.calendar(year,w,l,c).

calendar.firstweekday( ) : Returns the current setting for the weekday that starts each week. By default, when calendar is first imported, this is 0, meaning Monday.

```
print(calendar.firstweekday())
```

calendar.isleap(year) : Returns True if year is a leap year; otherwise, False.

```
print(calendar.isleap(2014))
```

calendar.leapdays(y1,y2) : Returns the total number of leap days in the years within range(y1,y2).

calendar.leapdays(2020, 2022)
calendar.leapdays(2020, 2024)
calendar.leapdays(2020, 2032)

calendar.month(year,month,w=2, l=1) : Returns a multiline string with a calendar for month month of year year, one line per week plus two header lines. w is the width in characters of each date; each line has length 7*w+6. l is the number of lines for each week.
Example
print(calendar.month(2014, 9))
print(calendar.month(2014, 9, w=2, l=1))
print(calendar.month(2014, 9, w=2, l=2))

calendar.prmonth(year,month,w=2,l=1) : Like print calendar.month(year,month,w,l).

calendar.monthcalendar(year,month) : Returns a list of lists of ints. Each sublist denotes a week. Days outside month month of year year are set to 0; days within the month are set to their day-of-month, 1 and up.

print(calendar.monthcalendar(2014, 9))
print(calendar.monthcalendar(2014, 10))

calendar.monthrange(year,month) : Returns two integers. The first one is the code of the weekday for the first day of the month month in year year; the second one is the number of days in the month. Weekday codes are 0 (Monday) to 6 (Sunday); month numbers are 1 to 12.

print(calendar.monthrange(2014, 9))
print(calendar.monthrange(2014, 10))

calendar.setfirstweekday(weekday) : Sets the first day of each week to weekday code weekday. Weekday codes are 0 (Monday) to 6 (Sunday).

calendar.timegm(tupletime) : The inverse of time.gmtime: accepts a time instant in time-tuple form and returns the same instant as a floating-point number of seconds since the epoch.
T=(2014, 9, 4, 11, 20, 40, 1, 245, 0)
time.gmtime(calendar.timegm(T))
time.asctime(time.gmtime(calendar.timegm(t)))

calendar.weekday(year,month,day) : Returns the weekday code for the given date. Weekday codes are 0 (Monday) to 6 (Sunday); month numbers are 1 (January) to 12 (December).
Example

```
calendar.weekday(2014,9, 5)
calendar.weekday(2014,9, 7)
```