U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Python library – time (*)

Provides several time-related functions. Some conventions and terminology.

- The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).
- The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding leap seconds.
- The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.
- Function strptime() can parse 2-digit years when given %y format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.
- UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT).
- DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year.
- The time value as returned by gmtime(), localtime(), and strptime(), and accepted by asctime(), mktime() and strftime(), is a sequence of 9 integers. The return values of gmtime(), localtime(), and strptime() also offer attribute names for individual fields.

(*) https://docs.python.org/3/library/time.html

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Python library – time

Use the following functions to convert between time representations:

From	To	Use
seconds since the epoch	struct_time in UTC	gmtime()
seconds since the epoch	struct_time in local time	localtime()
struct_time in UTC	seconds since the epoch	calendar.timegm()
struct_time in local time	seconds since the epoch	mktime()

Class struct_time

Index	Attribute	Values
0	tm_year	(for example, 1993)
1	tm_mon	range [1, 12]
2	tm_mday	range [1, 31]
3	tm_hour	range [0, 23]
4	tm_min	range [0, 59]
5	tm_sec	range [0, 61]; see **(2)** in strftime() description

Index	Attribute	Values
6	tm_wday	range [0, 6], Monday is 0
7	tm_yday	range [1, 366]
8	tm_isdst	0, 1 or -1; see below
N/A	tm_zone	abbreviation of timezone name
N/A	tm_gmtoff	offset east of UTC in seconds

Python library – time

Methods and attributes

```
In [1]: import time
        import calendar

In [2]: # Converts time since the epoch to a struct_time
        time.gmtime(0)

Out[2]: time.struct_time(tm_year=1970, tm_mon=1, tm_mday=1, tm_hour=0, t
        m_min=0, tm_sec=0, tm_wday=3, tm_yday=1, tm_isdst=0)

In [3]: # Returns the current time in UTC
        time_UTC = time.gmtime()
        time_UTC

Out[3]: time.struct_time(tm_year=2021, tm_mon=12, tm_mday=26, tm_hour=1
        2, tm_min=58, tm_sec=44, tm_wday=6, tm_yday=360, tm_isdst=0)

In [4]: # Returns the number of seconds since the epoch
        # Inverse of gmtime()
        calendar.timegm(time_UTC)

Out[4]: 1640523524
```

struct_time attributes

```
In [5]: time_UTC.tm_year, time_UTC.tm_mon, time_UTC.tm_mday

Out[5]: (2021, 12, 26)

In [6]: time_UTC.tm_hour, time_UTC.tm_min, time_UTC.tm_sec

Out[6]: (12, 58, 44)

In [7]: time_UTC.tm_wday, time_UTC.tm_yday

Out[7]: (6, 360)

In [8]: time_UTC.tm_isdst, time_UTC.tm_zone, time_UTC.tm_gmtoff

Out[8]: (0, 'UTC', 0)
```

Python library – time

Methods and attributes

```
In [9]: # Returns the number of seconds since the epoch
        time.time()
```

```
Out[9]: 1640523524.5299852
```

```
In [10]: # Returns the number of nanoseconds since the epoch
         time.time_ns()
```

```
Out[10]: 1640523524556971600
```

```
In [11]: # Returns the string representation of the time
         # expressed in floating number of seconds
         # Default argument is current time
         time.ctime()
```

```
Out[11]: 'Sun Dec 26 12:58:44 2021'
```

```
In [12]: # Returns the current local time - struct_time
         local_time = time.localtime()
         local_time
```

```
Out[12]: time.struct_time(tm_year=2021, tm_mon=12, tm_mday=26, tm_hour=1
         2, tm_min=58, tm_sec=44, tm_wday=6, tm_yday=360, tm_isdst=0)
```

```
In [13]: # struct_time time zone attribute
         local_time.tm_zone
```

```
Out[13]: 'GMT Standard Time'
```

```
In [14]: # struct_time UTC offset attribue
         local_time.tm_gmtoff
```

```
Out[14]: 0
```

```
In [15]: # struct_time DST attribute
         local_time.tm_isdst
```

```
Out[15]: 0
```

Python library – time

Methods and attributes

```
In [16]: # Convert local_time to seconds
         # inverse function of localtime()
         time.mktime(local_time)
Out[16]: 1640523524.0

In [17]: # Convert a struct_time to a time string
         # Default argument is current local time
         time.asctime()
Out[17]: 'Sun Dec 26 12:58:44 2021'

In [18]: # String format time given a struct_time
         # if time is not provided localtime() is used
         time.strftime('%Y-%m-%d', time.localtime())
Out[18]: '2021-12-26'

In [19]: # Convert a given time string in a struct_time
         time.strptime("Sat Dec 25 21:45:38 2021", '%a %b %d %H:%M:%S %Y')
Out[19]: time.struct_time(tm_year=2021, tm_mon=12, tm_mday=25, tm_hour=2
         1, tm_min=45, tm_sec=38, tm_wday=5, tm_yday=359, tm_isdst=-1)

In [20]: # Suspends the program execution by an amount of time
         time.sleep(10)
```

```
In [21]: # Computes the execution time of the code
         t1 = time.perf_counter()
         time.sleep(10)
         t2 = time.perf_counter()
         t2-t1
Out[21]: 10.0019793

In [22]: # Computes the execution time of the code
         # using nanoseconds
         t1 = time.perf_counter_ns()
         time.sleep(10)
         t2 = time.perf_counter_ns()
         t2-t1
Out[22]: 10008809300
```

Timezone constants

```
In [23]: time.altzone
Out[23]: -3600

In [24]: time.daylight
Out[24]: 1

In [25]: time.timezone
Out[25]: 0

In [26]: time.tzname
Out[26]: ('GMT Standard Time', 'GMT Summer Time')
```