Python Programming

- Presentation By Uplatz
- Contact us: https://training.uplatz.com
- Email: info@uplatz.com
- Phone: +44 7836 212635

Uplatz

DATE & TIME

Learning outcomes:

What is Tick?

What is TimeTuple?

Getting Current Time

Getting Formatted Time

Getting Calendar for a Month

Getting Calendar for a Year

DATE & TIME

A Python program can handle date and time in several ways. Converting between date formats is a common chore for computers. Python's time and calendar modules help track dates and times.

What is Tick?

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

There is a popular **time** module available in Python which provides functions for working with times and for converting between representations. The function ***time. time()*** returns the current system time in ticks since 12:00am, January 1, 1970(epoch).

What is Tick?

For 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 arithmetic is easy to do with ticks. However, dates before the epoch cannot be represented in this form. Dates in the far future also cannot be represented this way - the cutoff point is sometime in 2038 for UNIX and Windows.

What is 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
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 61 (60 or 61 are leap-seconds)
6	Day of Week	0 to 6 (0 is Monday)
7	Day of year	1 to 366 (Julian day)

What is TimeTuple?

The above tuple is equivalent to **struct_time** structure. This structure has following attributes:

Index	Attributes	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 or 61 are leap-seconds)
6	tm_wday	0 to 6 (0 is Monday)
7	tm_yday	1 to 366 (Julian day)

Getting Current Time

To translate a time instant from a *seconds since the epoch* floating-point value into a time-tuple, pass the floating-point value to a function (For example, localtime) that returns a time-tuple with all nine items valid.

For example:

```
import time;
localtime = time.localtime(time.time())
print ("Local current time :", localtime )
```

Getting Current Time

Example:

```
result = time.localtime()
print("result:", result)
print("\nyear:", result.tm_year)
print("tm_hour:", result.tm_hour)
print("tm_minute:", result.tm_min)
print("tm_second:", result.tm_sec)
print("tm_mday:", result.tm_mday)
print("tm_wday:", result.tm_wday)
```

Getting Formatted Time

You can format any time as per your requirement, but simple method to get time in readable format is **asctime().**

For example:

```
Formattedtime = time.asctime()
print ("Current time :", Formattedtime )
```

Getting Formatted Time

time.mktime():

The **mktime()** function takes struct_time (or a tuple containing 9 elements corresponding to struct_time) as an argument and returns the seconds passed since epoch in local time. **Basically, it's the inverse function of localtime().**

For Example:

```
seconds = 1596445145
t = time.localtime(seconds)
print("\nt1: ", t)
s = time.mktime(t)
print(s)
```

Getting Formatted Time

time.strftime():

The **strftime()** function takes **struct_time** (or tuple corresponding to it) as an argument and returns a string representing it based on the format code used. For example

```
import time
named_tuple = time.localtime()
time_string = time.strftime("%m/%d/%Y,
%H:%M:%S", named_tuple)
print(time_string)
```

Getting Calendar for a Month

The calendar module gives a wide range of methods to play with yearly and monthly calendars. Here, we print a calendar for a given month.

```
import calendar
cal = calendar.month(2020, 8)
print ("Here is the calendar:")
print (cal);
```

Getting Calendar for a Year

Example:

```
import calendar
calen = calendar.calendar(2020)
print ("Here is the calendar:")
print (calen);
```