

Python Date & Time Modules:

In Python, date, time and datetime classes provides a number of functions to deal with dates, times and time intervals.

Epoch

The Epoch is the point in time in Python from which time is measured. It is labelled 12:00AM, Jan 1, 1970. It is the beginning of an era.

The datetime classes in Python are categorized into main 5 classes.

date ==> Month, Day, Year

time ==> Hour, Minute, Second, Microsecond

datetime ==> Combination of Time & Date

timedelta==> A duration of time used for manipulating dates

tzinfo==> An abstract class for dealing with time zones

What is Tick?

The floating-point numbers in units of seconds for time interval are indicated by Tick in python.

Example

```
import time
ticks = time.time()
print("Number of ticks since 12:00am, January 1, 1970:", ticks)
```

Daylight Saving Time

US begins DST at 2:00 a.m. on the second Sunday in March and reverts to standard time on the first Sunday in November. DST-Benjamin Franklin.

In the European Union, Summer Time begins and ends at 1:00 a.m. Universal Time (Greenwich Mean Time). It begins the last Sunday in March and ends the last Sunday in October.

Struct_Time Structure

Index	Attributes	Values
0	tm_year	2020
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)
8	tm_isdst	-1, 0, 1, means library determines DST

Example:

```
#!/usr/bin/python
import time;
localtime = time.localtime(time.time())
print("Local current time :", localtime)
```

How to Use Date & DateTime Class

Before you run the code for datetime, it is important that you import the date time modules.

Example:

```
from datetime import date
td=date.today()
print("Today is: ",td)
day=td.day
print("Day is: ",day)
```

Example:

```
from datetime import date
td=date.today()
print(td.month)
print(td.year)
print(td.weekday())
print(date.weekday(td))
```

How to Format Time Output

1 We used the "strf time function" for formatting.
2 This function uses different control code to give an output.
3 Each control code resembles different parameters like year,month, weekday and date
[(%y/%Y - Year), (%a/%A- weekday), (%b/%B- month), (%d - day of month)] .

Example:

```
from datetime import datetime
x=datetime.now()
print(x.strftime("%y"))
print(x.strftime("%Y"))
print(x.strftime("%a"))
print(x.strftime("%A"))
print(x.strftime("%b"))
print(x.strftime("%B"))
print(x.strftime("%A %d %B,%y"))
print(x.strftime("%A %D %B,%Y"))
```

With the help of "Strftime" function we can also retrieve local system time, date or both.

%c- indicates the local date and time
%x- indicates the local date
%X- indicates the local time

Example:

```
from datetime import datetime
DFormat=datetime.now()
print(DFormat.strftime("%a"))
print(DFormat.strftime("%A"))
print(DFormat.strftime("%b"))
print(DFormat.strftime("%B"))
print(DFormat.strftime("%c"))
print(DFormat.strftime("%C"))
print(DFormat.strftime("%d"))
print(DFormat.strftime("%D"))
print(DFormat.strftime("%F"))
print(DFormat.strftime("%y"))
print(DFormat.strftime("%Y"))
print(DFormat.strftime("%A %d %B,%y"))
print(DFormat.strftime("%A %D %B,%Y"))
print(DFormat.strftime("%x"))
```

```
print(DFormat.strftime("%X"))
print(DFormat.strftime("%I:%M:%S %p"))
print(DFormat.strftime("%I:%M %p"))
```

Example:

```
from datetime import datetime
x=datetime.now()
print(x.strftime("%c"))
print(x.strftime("%x"))
print(x.strftime("%X"))
```

The "strftime function" allows you to call the time in any format 24 hours or 12 hours.

Example:

```
from datetime import datetime
x=datetime.now()
print(x.strftime("%I:%M:%S %p"))
print(x.strftime("%H:%M %p"))
```

How to use Timedelta Objects

With timedelta objects, you can estimate the time for both future and the past. In other words, it is a timespan to predict any special day, date or time.

Example:

```
from datetime import datetime
from datetime import timedelta
print(timedelta(days=365, hours=8, minutes=15))
print("ToDay is: ", datetime.now())
```

Example:

```
from datetime import timedelta
print(timedelta(days=345, hours=14, minutes=15))
#345 days, 14:15:00
print(timedelta(days=35, minutes=15, hours=14))
#35 days, 14:15:00
print(timedelta(days=35, minutes=15, hours=14, seconds=123))
#35 days, 14:17:03
```

The time Module

There is a popular time module available in Python which provides functions for working with times and for converting between representations. Here is the list of all available methods:

Getting formatted time

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

Example:

```
import time
lt=time.asctime(time.localtime(time.time()))
print(lt)
dst=time.asctime(time.localtime(time.daylight))
print(dst)
```

Python Calendar Module:

It allows you to output calendars like the Unix cal program. By

default, these calendars have Monday as the first day of the week, and Sunday as the last.

`iterweekdays()` method

It returns an iterator for the weekday numbers that will be used for one week. The first number from the iterator will be the same as the number returned by `firstweekday()`.

Syntax

```
iterweekdays()
```

Example:

```
import calendar
cal= calendar.Calendar(firstweekday=0)
for x in cal.iterweekdays():
    print(x)
```

`itermonthdays()` method

It returns an iterator of a specified month and a year. Days returned will simply be day numbers. The method is similar to `itermonthdates()`.

Syntax:

```
itermonthdays(year, month)
```

Example:

```
import calendar
cal= calendar.Calendar()
for x in cal.itermonthdays(2016, 5):
    print(x)
```

`itermonthdays2()` method

It is used to get an iterator for the month in the year similar to `itermonthdates()`. Days returned will be tuples consisting of a day number and a week day number.

Syntax

```
itermonthdays2(year, month)
```

Example:

```
import calendar
cal=calendar.Calendar(firstweekday=0)
for x in cal.itermonthdays2(2020,1):
    print(x)
```

NOTE: Do the task of the folloing calendar methods:

1. `itermonthdays3()`
2. `itermonthdays4()`

`itermonthdates()` method

It returns an iterator for the month (1-12) in the year. This iterator will return all days for the month and all days before the start of the month or after the end of the month that are required to get a complete week.

Syntax

```
itermonthdates(year, month)
```

Example:

```
import calendar
cal=calendar.Calendar(firstweekday=0)
for x in cal.itermonthdates(2020,1):
    print(x)
```

monthdatescalendar() method

It is used to get a list of the weeks in the month of the year as full weeks. Weeks are lists of seven datetime.date objects.

Syntax

```
monthdatescalendar(year, month)
```

Example:

```
import calendar
cal= calendar.Calendar()
print(cal.monthdatescalendar(2017, 5))
```

Python TextCalendar

formatyear() method

It is used to get a m-column calendar for an entire year as a multi-line string.

Syntax

```
formatyear(theyear, w=2, l=1, c=6, m=3)
```

year Year for which the calendar should be generated.

w The width between two columns. Default value is 2.

l Blank line between two rows. Default value is 1.

c Space between two months(Columnwise). Default value is 6

m Number of months in a row. Default value is 3.

Example:

```
import calendar
cal=calendar.TextCalendar(firstweekday=0)
print(cal.formatyear(2020,10))
```

Example:

```
import calendar
cal=calendar.TextCalendar(firstweekday=0)
print(cal.formatyear(2020,10,3,2,10,4))
```

Python HTMLCalendar Class:

formatmonth() method

It is used to get a month's calendar as an HTML table.

Syntax

```
formatmonth(theyear, themonth, withyear=True)
```

Example:

```
import calendar
cal=calendar.HTMLCalendar(firstweekday=0)
print(cal.formatmonth(2020,2))
```

More Calendar Methods:

isleap() method

It returns True if the year is a leap year, otherwise False.

Syntax

```
isleap(year)
```

Example:

```
import calendar
print(calendar.isleap(2016))
print(calendar.isleap(2020))
print(calendar.isleap(2019))
```

leapdays() method

It is used to get the number of leap years in a specified range of years.

Syntax

```
leapdays(y1, y2)
```

Example:

```
import calendar
print(calendar.leapdays(2015, 2018))
print(calendar.leapdays(2015, 2020))
print(calendar.leapdays(2000, 2020))
print(calendar.leapdays(2000, 2024))
```

weekheader() method

It is used to get a header containing abbreviated weekday names.

Syntax

```
weekheader(n)
```

Example:

```
import calendar
print(calendar.weekheader(3))
```

Display a specific Year Calendar

Example:

```
import calendar
Year=int(input("Enter Any Year: "))
Data=calendar.calendar(Year)
print(Data)
```

Example:Specific Month in a Year

```
import calendar
Year=2021
Month=1
Data=calendar.month(Year,Month)
print(Data)
```

calendar() method

It is used to get a 3-column calendar for an entire year as a multi-line string using the formatyear() of the TextCalendar class.

Syntax

```
calendar(year, w=2, l=1, c=6, m=3)
```

Example:

```
import calendar
print(calendar.calendar(2017))
```

month() method

It is used to get a month's calendar in a multi-line string using the formatmonth() of the TextCalendar class.

Syntax

```
month(theyear, themonth, w=0, l=0)
```

Example:

```
import calendar
print(calendar.month(2017,5))
```

Example:

```
while True:
    print("Options: ")
    print("Enter 'Yes' to Display Calendar: ")
    print("Enter Quit to End Program: ")
    user_input=input(":")
    if user_input=="Quit":
        break
    elif user_input=="Yes":
        import calendar
        Y=int(input("Enter Year: "))
        M=int(input("Enter Month: "))
        print(calendar.month(Y,M))
```

Q3 Measure the execution time of small bits of Python code with the "timeit" module

```
>>> import timeit
>>> timeit.timeit('"-".join(str(n) for n in range(100))',
                  number=10000)
0.3412662749997253
>>> timeit.timeit('"-".join([str(n) for n in range(100)])',
                  number=10000)
0.2996307989997149
>>> timeit.timeit('"-".join(map(str, range(100)))',
                  number=10000)
0.24581470699922647
```