2020

Free Python Course

O. Elijah
onlinepythonclass@gmail.com

1 Time and date 2
1.1 Time module 2
1.2 Datetime module 6
1.3 Calendar module 7
1.4 Date Mathematics 10
1.5 Application 11
1.6 References 13
1.7 Exercise 13
1.8 Appendix 14

1 TIME AND DATE

Welcome to chapter five of the free python course. In this chapter we shall look at how to use the time and date built-in modules in python. Note there are different built-in modules that can be used for date and time functions. We shall explore some of them in this chapter.

What we intend to achieve

1. Learn how to use the **datetime** and **time** modules in python
2. display time and date in python
3. Learn how to work with calendar and dates

1.1 Time module

time module in python is a popular module that provides functions for working with times and for converting between representations. To use the time module in Python you type the following line of code

```
import time;  # This is required to include time module.
```

There are many methods in module **time** that can be used to format or display your time and date as shown

Getting current time

methods	remarks
time.time() 1519401078.2639325	# returns a time frame that contains all the data of time i..e year, month, day, hour, min, sec, day of week, day of year and day light savings
time.localtime(1519401078.2639325) **Results** time.struct_time(tm_year=2018, tm_mon=2, tm_mday=23, tm_hour=23, tm_min=51, tm_sec=18, tm_wday=4, tm_yday=54, tm_isdst=0)	# returns the time structure that contains all the data of time as a tuple of 9 numbers. (See Table 4.1)

or

```
localtime = time.localtime(time.time())

print ("Local current time :", localtime)
```

Many of python's time functions handle time as a tuple of 9 numbers as shown below (See Section 1.4 to know more about **tuple**)

Table 4.1 Time as a tuple of 9 numbers

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, 6 is Sunday)
7	Day of year	1 to 366 (Julian day)
8	Daylight savings	-1, 0, 1, -1 means library determines DST

Getting formatted time

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

Methods	
time.asctime([tupletime]) `import time;` `time.asctime( time.localtime(time.time()) )` 'Sat Feb 24 00:04:43 2018'	returns a readable 24-character string
time.ctime([secs]) `import time` `print ("time.ctime() : %s" % time.ctime())` Results `time.ctime() : Tue Feb 17 10:00:18 2009`	The method ctime() converts a time expressed in seconds since the epoch to a string representing local time. If secs is not provided or None, the current time as returned by time() is used. This function is equivalent to asctime(localtime(secs)).
time.sleep(secs) import time print (time.ctime()) time.sleep(5) print (time.ctime()) Results Fri Mar 2 17:32:42 2018 Fri Mar 2 17:32:47 2018	The method sleep() suspends execution for the given number of seconds. See difference of 5 Seconds.
time.localtime([secs]) `import time` `print ("time.localtime() : %s" % time.localtime())` Results `time.localtime() : (2009, 2, 17, 17, 3, 38, 1, 48, 0)`	Converts number of seconds to local time. If secs is not provided or None, the current time as returned by time() is used.
time.mktime(tupletime)	The method mktime() is the inverse function of localtime(). Its argument is the struct_time or full

`` import time t = (2009, 2, 17, 17, 3, 38, 1, 48, 0) secs = time.mktime(t) print ("time.mktime(t) : %f" % secs) print ("asctime(localtime(secs)): %s" % time.asctime(time.localtime(secs))) results time.mktime(t) : 1234915418.000000 asctime(localtime(secs)): Tue Feb 17 17:03:38 2009 ```	9-tuple and it returns a floating point number, for compatibility with time().
time.strftime(fmt[,t]) t is the time in number of seconds to be formatted. fmt − This is the directive which would be used to format given time. The following directives can be embedded in the format string – See Appendix for all the directives. ``` import time t = (2009, 2, 17, 17, 3, 38, 1, 48, 0) t = time.mktime(t) print time.strftime("%b %d %Y %H:%M:%S", time.gmtime(t)) ``` **results** ``` Feb 18 2009 00:03:38 ```	Accepts an instant expressed as a time-tuple in local time and returns a string representing the instant as specified by string fmt. See Appendix for all the types of formats.

For more time methods see the link
https://www.tutorialspoint.com/python/python_date_time.htm

1.2 Datetime module

The **datetime** module provides a few types to deal with dates, times, and time intervals. This module replaces the integer/tuple-based time mechanisms in the **time** module with a more object-oriented interface.

The module contains the following types:

- The **datetime** type represents a date and a time during that day.
- The **date** type represents just a date, between year 1 and 9999 (see below for more about the calendar used by the datetime module)
- The **time** type represents a time, independent of the date.
- The **timedelta** type represents the difference between two time or date objects.
- The **tzinfo** type is used to implement timezone support for time and datetime objects; more about that below.

Creating a datetime object

```
import datetime

now = datetime.datetime(2003, 8, 4, 12, 30, 45)

print(now)
print('\n')
print('year is:', now.year)
print('\n')
print('month is:', now.month)
print('\n')
print('day is:', now.day)
print('\n')
print('hour is:', now.hour)
print('\n')
print('min is:', now.minute)
print('\n')
print('second is:', now.second)
```

Getting the current date and time using **strftime** method

```
import time
import datetime
print("Current date and time: " , datetime.datetime.now())
print("Current year: ", datetime.date.today().strftime("%Y"))
print("Month of year: ", datetime.date.today().strftime("%B"))
```

```
print("Week number of the year: ",
datetime.date.today().strftime("%W"))
print("Weekday of the week: ", datetime.date.today().strftime("%w"))
print("Day of year: ", datetime.date.today().strftime("%j"))
print("Day of the month : ", datetime.date.today().strftime("%d"))
print("Day of week: ", datetime.date.today().strftime("%A"))
```

Results

```
Current date and time:  2017-05-05 17:21:19.106836
Current year:  2017
Month of year:  May
Week number of the year:  18
Weekday of the week:  5
Day of year:  125
Day of the month :  05
Day of week:  Friday
```

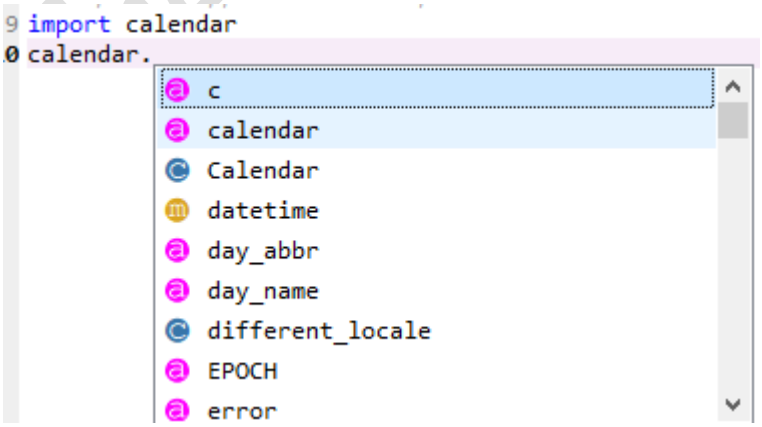
1.3 Calendar module

The calendar module supplies calendar-related functions, including functions to print a text calendar for a given month or year.

To use the calendar module in Python you type the following line of code

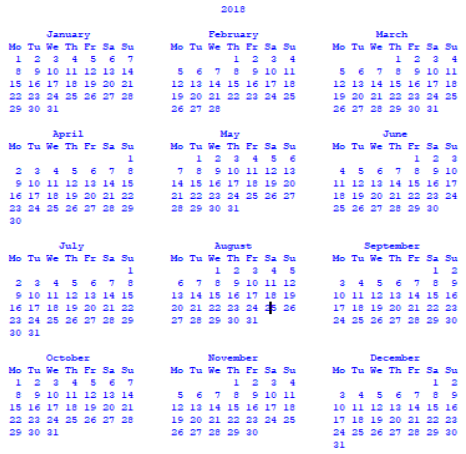
```
import calendar; # This is required to include calender module.
```

There are many methods in module **calendar** that can be used to format or display your calendar as shown below . To check the methods, you can try the word calendar.



Methods	

`` import calendar cal = calendar.month(2018, 3) print ("Here is the calendar:") print (cal) Results Here is the calendar: March 2018 Mo Tu We Th Fr Sa Su 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 ```	This prints a calendar for a given month (2018, March).
calendar.isleap(year) Example calendar.isleap(2018) **Result** False	Returns True if year is a leap year; otherwise, False
calendar.leapdays(y1,y2) Example print(calendar.leapdays(2001,2018)) **Result** 4	Returns the total number of leap days in the years within range(y1,y2).
calendar.month(year,month,w=2,l=1) Example print(calendar.month(2018,3,w=2,l=1)) **Results** March 2018 Mo Tu We Th Fr Sa Su 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 Try to change the w and l as follows and observe what happens print(calendar.month(2018,3,w=5,l=2))	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.

calendar.weekday(year,month,day) Example print(calendar.weekday(2018,3,4)) **Result** 6	Returns the weekday code for the given date. Weekday codes are 0 (Monday) to 6 (Sunday); month numbers are 1 (January) to 12 (December).
calendar.calendar(year,w=2,l=1,c=6) Example print(calendar.calendar(2018,w=2,l=1,c=6)) 	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.
calendar.firstweekday( ) Example calendar.firstweekday( ) **Result** 0	Returns the current setting for the weekday that starts each week. By default, when calendar is first imported, this is 0, meaning Monday.
calendar.setfirstweekday(weekday) Example calendar.setfirstweekday(3) To check the new setting try calendar.firstweekday( ) **Result** 3	Sets the first day of each week to weekday code weekday. Weekday codes are 0 (Monday) to 6 (Sunday).
calendar.weekday(year,month,day) Example calendar.weekday(2018,3,5) 0 It returns a code 0 meaning (Monday) for the date 2018 March 5th. Try for a different date 2018 March 23rd calendar.weekday(2018,3,23) **Result** 4	Returns the weekday code for the given date. Weekday codes are 0 (Monday) to 6 (Sunday); month numbers are 1 (January) to 12 (December). Returns a code 4 meaning (Friday)

1.4 Date Mathematics

You can use the timedelta object to estimate the time for both future and the past.

To run timedelta Objects, you need to declare the import statement first and then execute the code

Retrieve date a year from now through delta objects

```
from datetime import date, timedelta
dt = date.today() + timedelta(365)
print('Current Date :',date.today())
print(' one year from Current Date :',dt)
```

Determine how many days past the new year

```
from datetime import date, timedelta
dt = date.today()  # get today's date

ny = date(dt.year,1,1)  # get new year date

if ny < dt:   # check if new year date is less than today's date

print('todays date is: ',dt)

print('New year date is: ', ny)

print('days after new year is: ',(dt-ny).days)
```

Results

todays date is: 2018-03-04

New year date is: 2018-01-01

days after new year is: 62

Other date and time modules

There other modules and functions to play with date and time in Python. Examples are

- pytz module
- dateutil module

1.5 Application

Example 1

Write a python program to display calendar of any month in any year. The user should enter the year and month between 1 – 12 . Example 2020 and 2

```
import calendar

# Example 1 Get the year and month from a user and print out the  month - e.g year 2020 and month 02

y = int(input("Input the year : "))

m = int(input("Input the month (1-12 : "))



my_months = ['Jan', 'Feb', 'March', 'April', 'May', 'June', 'July', 'August', 'Sept', 'Oct', 'Nov', 'Dec']



print("    Month of {} {}".format(my_months[m-1],y))

print("======================")

print(calendar.month(y, m))
```

Result

```
Input the year : 2020

Input the month (1-12) : 2
    Month of Feb 2020
======================
   February 2020
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29
```

Example 2

Write a python program to calculate the age of a person when the year of birth is given

```
import datetime

y = int(input("Input the year of birth : "))

current_year  = datetime.date.today().strftime("%Y") # get current year

age = int(current_year) -  y

print("Current year is {}".format(current_year))

print("Your year of birth is {}".format(y))

print("Your age is {}".format(age))
```

Result

```
Input the year of birth : 1990
Current year is 2020
Your year of birth is 1990
Your age is 30
```

Example 3

Write a program to check if an applicant is eligible for employment. The applicant must be under 30 years old

```
import datetime

y = int(input("Input the year of birth : "))

current_year  = datetime.date.today().strftime("%Y") # get current year

age = int(current_year) -  y

if age < 30:

    print ("eligible  for employment" )

else:

    print("Not eligible")
```

Result

```
Input the year of birth : 1999
eligible  for employment
```

1.6 References

[1] http://strftime.org/

[2] https://www.guru99.com/date-time-and-datetime-classes-in-python.html

[3] https://www.w3resource.com/python-exercises/date-time-exercise/

1.7 Exercise

1. Write a Python script to display following using time or datetime module

a) Current date and time

b) Current year

c) Month of year as a locale's full name example September (refer to the strftime format in appendix)

d) Week number of the year

e) Weekday of the week as a locale's full name example Monday (refer to the strftime format in appendix)

f) Day of year

g) Day of the month

h) Day of week

1. Write a script to check if the following years are leap years
    a. 1900
    b. 2016
    c. 1816
    d. 2020
2. Write a script to write the today's date and the date a year from today
   current Date : 2020-03-04

one year from Current Date : 2021-03-04

3. Write a Python program to print next 7 days starting from today.
   Hint: https://www.w3resource.com/python-exercises/date-time-exercise/python-date-time-exercise-9.php
4. Write python program to display the July and august, 2020 calendar

1.8 Appendix

`strftime(format)` method, to create a string representing the time under the control of an explicit format string.

Code	Meaning	Example
%a	Weekday as locale's abbreviated name.	Mon
%A	Weekday as locale's full name.	Monday
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	1
%d	Day of the month as a zero-padded decimal number.	30
%-d	Day of the month as a decimal number. (Platform specific)	30
%b	Month as locale's abbreviated name.	Sep
%B	Month as locale's full name.	September
%m	Month as a zero-padded decimal number.	09
%-m	Month as a decimal number. (Platform specific)	9
%y	Year without century as a zero-padded decimal number.	13
%Y	Year with century as a decimal number.	2013
%H	Hour (24-hour clock) as a zero-padded decimal number.	07
%-H	Hour (24-hour clock) as a decimal number. (Platform specific)	7
%I	Hour (12-hour clock) as a zero-padded decimal number.	07
%-I	Hour (12-hour clock) as a decimal number. (Platform specific)	7
%p	Locale's equivalent of either AM or PM.	AM
%M	Minute as a zero-padded decimal number.	06
%-M	Minute as a decimal number. (Platform specific)	6
%S	Second as a zero-padded decimal number.	05

Code	Meaning	Example
%-S	Second as a decimal number. (Platform specific)	5
%f	Microsecond as a decimal number, zero-padded on the left.	000000
%z	UTC offset in the form +HHMM or -HHMM (empty string if the the object is naive).	
%Z	Time zone name (empty string if the object is naive).	
%j	Day of the year as a zero-padded decimal number.	273
%-j	Day of the year as a decimal number. (Platform specific)	273
%U	Week number of the year (Sunday as the first day of the week) as a zero padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	39
%W	Week number of the year (Monday as the first day of the week) as a decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	39
%c	Locale's appropriate date and time representation.	Mon Sep 30 07:06:05 2013
%x	Locale's appropriate date representation.	09/30/13
%X	Locale's appropriate time representation.	07:06:05
%%	A literal '%' character.	%