DATE AND TIME

The epoch

'epoch' is the point where the time starts. This point is taken as the 0.0 hours of January 1st of the current year. For Unix, the epoch is 0.0 hours of January 1st of 1970. It is possible to measure the time in seconds since the epoch using time() function of 'time' module.

Epoch time can be converted into date and time with the help of ctime() function.

PROGRAMS
41. To know the time since the epoch.

Date and time now

The current date and time as shown in our computer system can be known using the following:

- ctime() function of 'time' module. Just call ctime() without passing anything.
  Ex: dt = time.ctime()
- today() method of 'datetime' class of 'datetime' module.
  Ex: dt = datetime.today()
- now() method of 'datetime' class of 'datetime' module.
  Ex: dt = datetime.now()
  We can retrieve individual values as: dt.day, dt.month, dt.year, dt.hour, dt.minute, dt.second.

PROGRAMS
42. To know the current date and time.

Combining date and time

We can create 'datetime' class object by combining date class object and time class objects using combine() method. Please remember that the date and time classes belong to 'datetime' module. For example, we can create a date class object with some date, as:

d = date(2016, 4, 29)

Similarly, we can create time class object and store some time, as:

t = datetime.time(15, 30)

Now, the combine() method is a class method in the class 'datetime' that can combine the previously created objects, as:

dt = datetime.combine(d, t)

PROGRAMS
43. Enter date and time from keyboard and combine them.

Formatting dates and times

The contents of the ‘datetime’,’ date’ and ‘time’ classes objects can be formatted using strftime() method. ‘strftime’ represents ‘string format of time’. This method converts the objects into a specified format and returns the formatted string.
dt = strftime(‘Formatted string’)

Format code	**Meaning**	**Example**
%a	Weekday as an abbreviated name.	Sun, Mon, … Sat
%A	Weekday as full name.	Sunday, Monday, … Saturday
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, … 6
%d	Day of the month as a zero-padded decimal number.	01, 02, … 31
%b	Month as an abbreviated name.	Jan, Feb, … Dec
%B	Month as full name.	January, February, … December
%m	Month as zero-padded decimal number.	01, 02, … 12
%y	Year without century as a zero-padded decimal number.	00, 01, … 99
%Y	Year with century as decimal number.	0001, 0002, … 2016, … 9999
%H	Hour (24-hour clock) as zero-padded decimal number.	00, 01, … 23
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, … 12
%p	Either AM or PM.	AM, PM
%M	Minute as a zero-padded decimal number.	00, 01, … 59
%S	Second as a zero-padded decimal number.	00, 01, … 59
%f	Microsecond as a decimal number, zero-padded on the left.	000000, 000001, ... 999999
%Z	Time zone name.	(empty), UTC, EST, CST
%j	Day of the year as a zero-padded decimal number.	001, 002, … 366
%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.	00, 01, … 53
%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.	00, 01, … 53
%c	Appropriate date and time representation.	Tue Aug 16 21:30:00 1988
%x	Appropriate date representation.	08/16/88 (None); 08/16/1988 (en_US)
%X	Appropriate time representation.	21:30:00 (en_US)
%%	A single % character.	%

Example 1: To display date, month name and year.

```
from datetime import *
dt = date.today()
str = dt.strftime('Date = %d, %B, %Y')
print(str)

Output
Date = 05, September, 2017
```

Example 2: To find current year's day number and week day name.

```
from datetime import *
dt = date.today()
str = dt.strftime('Today is the %j th day of current year')
print(str)
str = dt.strftime('And it is %A')
print(str)

Output
Today is the 248 th day of current year
And it is Tuesday
```

Example 3: To format the current time and show it as HH: MM: SS.

```
from datetime import *
dt = datetime.now()
str = dt.strftime('Current time= %H:%M:%S')
print(str)

Output
Current time= 18:49:03
```

Finding durations using timedelta

The 'timedelta' class of 'datetime' module is useful to find the durations like difference between two dates or finding the date after adding a period to an existing date. It is possible to know the future dates or previous dates using 'timedelta'. The 'timedelta' class object is available in the following format:

timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

All arguments passed to the 'timedelta' object are optional and default to 0. Arguments may be integers or floats, and may be positive or negative.

Example 4: Find the future date if we add 10 days, 12 hours, 30 minutes, 10 seconds to the existing date.

```
from datetime import *
d1 = datetime(2017, 9, 5, 6, 50, 30)
period = timedelta(days=10, hours=12, minutes=30, seconds=10)
print('New date and time = ', d1+period)
Output
New date and time =  2017-09-15 19:20:40
```

Comparing two dates

It is possible to compare two 'date' class objects or 'datetime' class objects just like comparing two numbers. For example, d1 and d2 are to be compared, we can write:

```
d1 == d2
d1> d2
d1<d2
```

These expressions return either True or False.
Ex:

```
b1 = date(y1, m1, d1)
b2 = date(y2, m2, d2)
if b1==b2: print('Both persons are of same age')
```

Sorting dates

One of the best ways to sort a group of dates is to store them into a list and then apply list's sort() method. Dates can be stored in date class object as: date(y,m,d).

PROGRAMS
44. Sort a given group of dates.

Stopping execution temporarily

To stop execution of a program temporarily for a given amount of time, we can use sleep() function of 'time' module. This function is used in the format:

```
time.sleep(2)  # sleep for 2 seconds
time.sleep(2.5)   # sleep for 2.5 seconds
```

Knowing the time taken by a program

Python provides two functions: perf_counter() and process_time() of 'time' module to measure the time difference between two points in a program. perf_counter() is useful to know the time taken by a program. process_time() is useful to know the time taken by a program + CPU's time in executing the program.

PROGRAMS
45. A Python program to find the execution time of a program.

Working with calendar module

The 'calendar' module is useful to create calendar for any month or year and to test whether year is leap or not.
Ex:

```
calendar.isleap(y)   # returns True if y is leap year

str = calendar.month(2017, 9)
print(str)   # displays calendar for Sep 2017
```

str = calendar.calendar(2017)
print(str)   # displays calendar for the year 2017

SOLUTIONS FOR PROGRAMS IN THIS CHAPTER

```
# 41. knowing the time since the epoch
import time
epoch = time.time()   # call time() function of time module
print('epoch seconds= ', epoch)

# convert epoch time into date and time
t = time.ctime(epoch)
print('date and time as per epoch= ', t)

# 42. knowing the current date and time
import time, datetime
dt = time.ctime()
print('Current date and time= ', dt)

dt = datetime.datetime.today()
print('Current date and time= ', dt)

dt = datetime.datetime.now()
print('Current date: {}/{}/{}'.format(dt.day, dt.month, dt.year))
print('Current time: {}:{}:{}'.format(dt.hour, dt.minute, dt.second))

# 43. combining date and time
from datetime import *
d,m,y = [int(x) for x in input('Enter date, month, year:
').split(',')]
hr,min = [int(x) for x in input('Enter hour, minute: ').split(',')]

d = date(y,m,d)
t = time(hr, min)
dt = datetime.combine(d, t)
print('Created object= ', dt)

# 44. sorting dates
from datetime import *

# take an empty list
lst = []

# create date class objects with year, month and date
n = int(input('How many dates? '))
for i in range(n):
    d,m,y = [int(x) for x in input('Enter date, month, year:
').split()]
    dt = date(y,m,d)
    # append it to lst
    lst.append(dt)

# sort the list
lst.sort()
```

```
# display sorted dates
for i in lst:
    print(i)

# 45. to measure the time taken by the program
from time import *

# note the starting time of the program
t1 = perf_counter()

# do some processing
i, sum = 0, 0
while(i<1000000):
    sum+=i
    i+=1

# make the PVM sleep for 3 seconds
# this is also measured by the perf_counter()
sleep(3)

# note the ending time of the program
t2 = perf_counter()

# find time for the program in seconds
print('Execution time = %f seconds' %(t2- t1))
```