3. Dates and Times

Introduction

```
#-----------------------------
#introduction
# There are three common ways of manipulating dates in Python
# mxDateTime - a popular third-party module (not discussed here)
# time - a fairly low-level standard library module
# datetime - a new library module for Python 2.3 and used for most of these samples
# (I will use full names to show which module they are in, but you can also use
# from datetime import datetime, timedelta and so on for convenience)

import time
import datetime

print "Today is day", time.localtime()[7], "of the current year"
# Today is day 218 of the current year

today = datetime.date.today()
print "Today is day", today.timetuple()[7], "of ", today.year
# Today is day 218 of 2003

print "Today is day", today.strftime("%j"), "of the current year"
# Today is day 218 of the current year
```

Finding Today's Date

```
#-----------------------------
# Finding todays date

today = datetime.date.today()
print "The date is", today
#=> The date is 2003-08-06

# the function strftime() (string-format time) produces nice formatting
# All codes are detailed at http://www.python.org/doc/current/lib/module-time.html
print t.strftime("four-digit year: %Y, two-digit year: %y, month: %m, day: %d")
#=> four-digit year: 2003, two-digit year: 03, month: 08, day: 06
```

Converting DMYHMS to Epoch Seconds

```
#-----------------------------
# Converting DMYHMS to Epoch Seconds
# To work with Epoch Seconds, you need to use the time module

# For the local timezone
t = datetime.datetime.now()
print "Epoch Seconds:", time.mktime(t.timetuple())
#=> Epoch Seconds: 1060199000.0

# For UTC
t = datetime.datetime.utcnow()
print "Epoch Seconds:", time.mktime(t.timetuple())
#=> Epoch Seconds: 1060195503.0
```

Converting Epoch Seconds to DMYHMS

```
#----------------------------
# Converting Epoch Seconds to DMYHMS

now = datetime.datetime.fromtimestamp(EpochSeconds)
#or use datetime.datetime.utcfromtimestamp()
print now
#=> datetime.datetime(2003, 8, 6, 20, 43, 20)
print now.ctime()
#=> Wed Aug  6 20:43:20 2003

# or with the time module
oldtimetuple = time.localtime(EpochSeconds)
# oldtimetuple contains (year, month, day, hour, minute, second, weekday, yearday, daylightSav
print oldtimetuple
#=> (2003, 8, 6, 20, 43, 20, 2, 218, 1)
```

Adding to or Subtracting from a Date

```
#----------------------------
# Adding to or Subtracting from a Date
# Use the rather nice datetime.timedelta objects

now = datetime.date(2003, 8, 6)
difference1 = datetime.timedelta(days=1)
difference2 = datetime.timedelta(weeks=-2)

print "One day in the future is:", now + difference1
#=> One day in the future is: 2003-08-07

print "Two weeks in the past is:", now + difference2
#=> Two weeks in the past is: 2003-07-23

print datetime.date(2003, 8, 6) - datetime.date(2000, 8, 6)
```

```
#=> 1095 days, 0:00:00

#-----------------------------
birthtime = datetime.datetime(1973, 01, 18, 3, 45, 50)   # 1973-01-18 03:45:50

interval = datetime.timedelta(seconds=5, minutes=17, hours=2, days=55)
then = birthtime + interval

print "Then is", then.ctime()
#=> Then is Wed Mar 14 06:02:55 1973

print "Then is", then.strftime("%A %B %d %I:%M:%S %p %Y")
#=> Then is Wednesday March 14 06:02:55 AM 1973

#-----------------------------
when = datetime.datetime(1973, 1, 18) + datetime.timedelta(days=55)
print "Nat was 55 days old on:", when.strftime("%m/%d/%Y").lstrip("0")
#=> Nat was 55 days old on: 3/14/1973
```

Difference of Two Dates

```
#-----------------------------
# Dates produce timedeltas when subtracted.

diff = date2 - date1
diff = datetime.date(year1, month1, day1) - datetime.date(year2, month2, day2)
#-----------------------------

bree = datetime.datetime(1981, 6, 16, 4, 35, 25)
nat  = datetime.datetime(1973, 1, 18, 3, 45, 50)

difference = bree - nat
print "There were", difference, "minutes between Nat and Bree"
#=> There were 3071 days, 0:49:35 between Nat and Bree
```

```
weeks, days = divmod(difference.days, 7)

minutes, seconds = divmod(difference.seconds, 60)
hours, minutes = divmod(minutes, 60)

print "%d weeks, %d days, %d:%d:%d" % (weeks, days, hours, minutes, seconds)
#=> 438 weeks, 5 days, 0:49:35

#------------------------------
print "There were", difference.days, "days between Bree and Nat."
#=> There were 3071 days between bree and nat
```

Day in a Week/Month/Year or Week Number

```
#------------------------------
# Day in a Week/Month/Year or Week Number

when = datetime.date(1981, 6, 16)

print "16/6/1981 was:"
print when.strftime("Day %w of the week (a %A). Day %d of the month (%B).")
print when.strftime("Day %j of the year (%Y), in week %W of the year.")

#=> 16/6/1981 was:
#=> Day 2 of the week (a Tuesday). Day 16 of the month (June).
#=> Day 167 of the year (1981), in week 24 of the year.
```

Parsing Dates and Times from Strings

```
#---------------------------
```

```
# Parsing Dates and Times from Strings

time.strptime("Tue Jun 16 20:18:03 1981")
# (1981, 6, 16, 20, 18, 3, 1, 167, -1)

time.strptime("16/6/1981", "%d/%m/%Y")
# (1981, 6, 16, 0, 0, 0, 1, 167, -1)
# strptime() can use any of the formatting codes from time.strftime()

# The easiest way to convert this to a datetime seems to be;
now = datetime.datetime(*time.strptime("16/6/1981", "%d/%m/%Y")[0:5])
# the '*' operator unpacks the tuple, producing the argument list.
```

Printing a Date

```
#-----------------------------
# Printing a Date
# Use datetime.strftime() - see helpfiles in distro or at python.org

print datetime.datetime.now().strftime("The date is %A (%a) %d/%m/%Y")
#=> The date is Friday (Fri) 08/08/2003
```

High-Resolution Timers

```
#-----------------------------
# High Resolution Timers

t1 = time.clock()
# Do Stuff Here
t2 = time.clock()
print t2 - t1
```

```
# 2.27236813618
# Accuracy will depend on platform and OS,
# but time.clock() uses the most accurate timer it can

time.clock(); time.clock()
# 174485.51365466841
# 174485.55702610247

#-----------------------------
# Also useful;
import timeit
code = '[x for x in range(10) if x % 2 == 0]'
eval(code)
# [0, 2, 4, 6, 8]

t = timeit.Timer(code)
print "10,000 repeats of that code takes:", t.timeit(10000), "seconds"
print "1,000,000 repeats of that code takes:", t.timeit(), "seconds"

# 10,000 repeats of that code takes: 0.128238644856 seconds
# 1,000,000 repeats of that code takes:  12.5396490336 seconds

#-----------------------------
import timeit
code = 'import random; l = random.sample(xrange(10000000), 1000); l.sort()'
t = timeit.Timer(code)

print "Create a list of a thousand random numbers. Sort the list. Repeated a thousand times."
print "Average Time:", t.timeit(1000) / 1000
# Time taken: 5.24391507859
```

Short Sleeps

```
#-----------------------------
# Short Sleeps

seconds = 3.1
time.sleep(seconds)
print "boo"
```

Program: hopdelta

```
#-----------------------------
# Program HopDelta
# Save a raw email to disk and run "python hopdelta.py FILE"
# and it will process the headers and show the time taken
# for each server hop (nb: if server times are wrong, negative dates
# might appear in the output).

import datetime, email, email.Utils
import os, sys, time

def extract_date(hop):
    # According to RFC822, the date will be prefixed with
    # a semi-colon, and is the last part of a received
    # header.
    date_string = hop[hop.find(';')+2:]
    date_string = date_string.strip()
    time_tuple = email.Utils.parsedate(date_string)

    # convert time_tuple to datetime
    EpochSeconds = time.mktime(time_tuple)
    dt = datetime.datetime.fromtimestamp(EpochSeconds)
    return dt

def process(filename):
```

```
    # Main email file processing
    # read the headers and process them
    f = file(filename, 'rb')
    msg = email.message_from_file(f)

    hops = msg.get_all('received')

    # in reverse order, get the server(s) and date/time involved
    hops.reverse()
    results = []
    for hop in hops:
        hop = hop.lower()

        if hop.startswith('by'):  # 'Received: by' line
            sender = "start"
            receiver = hop[3:hop.find(' ',3)]
            date = extract_date(hop)

        else:  # 'Received: from' line
            sender = hop[5:hop.find(' ',5)]
            by = hop.find('by ')+3
            receiver = hop[by:hop.find(' ', by)]
            date = extract_date(hop)

        results.append((sender, receiver, date))
    output(results)

def output(results):
    print "Sender, Recipient, Time, Delta"
    print
    previous_dt = delta = 0
    for (sender, receiver, date) in results:
        if previous_dt:
            delta = date - previous_dt

        print "%s, %s, %s, %s" % (sender,
```

```
                                receiver,
                                date.strftime("%Y/%d/%m %H:%M:%S"),
                                delta)
            print
            previous_dt = date

def main():
    # Perform some basic argument checking
    if len(sys.argv) != 2:
        print "Usage: mailhop.py FILENAME"

    else:
        filename = sys.argv[1]
        if os.path.isfile(filename):
            process(filename)
        else:
            print filename, "doesn't seem to be a valid file."

if __name__ == '__main__':
    main()
```

Prev Home Next
Numbers Arrays