```
# TheDaylightClass.py
""" This module supports the reading of text files
that
contain daily sunrise/sunset information.
"""

from numpy import *
from
GetData import fileToStringList

# These help functions help decipher the data in the
sunrise/sunset
# text files.

def ConvertTime(s):
    """ Returns a float that
is the equivalent (in hours)
    of the time encoded by s.

    '21:45' means 9:45 pm.


    PredC: s a 5-character string of the form hh:mm that
    specifies time.

"""

    return float(s[0:2])+float(s[2:])/60

def LatLong(s):

""" Returns a tuple (Long,Lat) of floats that are the
    equivalent (in
degrees) of the longitude and latitude
    encoded by s.

    PredC: s an 11-character string
of the form 'cdddmmCDDMM'
    where cdddmm specifies longitude in degrees and minutes with

c =  'W' or 'E' and CDDMM species latitude in degrees and minutes
    with C = 'N' or 'S'.

"""
    Long = float(s[1:4])+float(s[4:6])/60
    if s[0]=='E':
       Long =
-Long
    Lat = float(s[7:9])+float(s[9:11])/60
    if s[6]=='S':
        Lat = -Lat
    return
(Lat,Long)

def CityToFileName(CityName):
    """ Returns a string that encodes
the name of the file that houses
    the sunrise/sunset data for the the city named by cName.

  An error results if no such file exists.

    PreC: CityName is a string that names a
city. The directory/folder
    RisSetData is in the current working directory.

"""

    # There is Rise/Set data for these cities..
    Cities =
(['Anaheim','Anchorage','Arlington','Athens','Atlanta',

'Baltimore','Bangkok','Beijing','Berlin','Bogata',

'Boston','BuenosAires','Cairo','Chicago','Cincinnati',
'Cleveland','Denver','Detroit','Honolulu','Houston',

'Ithaca','Johannesburg','KansasCity','Lagos','London',

'LosAngeles','MexicoCity','Miami','Milwaukee','Minneapolis',

'Moscow','NewDelhi','NewYork','Oakland','Paris',

'Philadelphia','Phoenix','Pittsburgh','RiodeJaneiro','Rome',

'SanFrancisco','Seattle','Seoul','Sydney','Tampa',

'Teheran','Tokyo','Toronto','Washington','Wellington'])

    assert CityName in Cities,
('No sunrise/sunset data for ' + cName)
    FileName = 'RiseSetData/' + CityName + '.dat'

return FileName


class Daylight(object):
    """
    This class support
manipulations of sunrise/sunset data.

    Attributes:
       City:     name of the city
[str]
       Lat:      latitude in degrees [float]
       Long:     longitude in degrees
[float]
       RiseTime: length-365 numpy array of sunrise times
       SetTime:  length-365
numpy array of sunset times

    For k in range(365), RiseTime[k] is the sunrise time on the
k-th day of the
    year and RiseTime[k] is the sunset time on the kth day of the year,

"""

    def __init__(self,CityName):
        """ Returns a
reference to a Daylight object.

        PreC: CityName is the name of a city [str]

    """

        self.RiseTime = zeros(365)
        self.SetTime  =
zeros(365)
        FileName = CityToFileName(CityName)
        L = fileToStringList(FileName)

      lineNum = 0
        for line in L:
           # Isolate the columns of data

parts = line.split()
           lineNum+=1
           if lineNum==1:
                # First
line has the city name
                self.City = parts[0]
           elif lineNum==2:

        # Second line has lat/long information
                (self.Lat,self.Long) =
LatLong(parts[0])
           else:
                # Remaining lines have rise/set pairs
```

```
            day = int(parts[0])
                  # Get all the rise and set times

RiseTimeList = map(ConvertTime,parts[1:len(parts):2])
                  SetTimeList  =
map(ConvertTime,parts[2:len(parts):2])
                  p = len(RiseTimeList)

for k in range(p):
                      if day<=28:
                          # All months have
at least 28 days
                          starts  =
[0,31,59,90,120,151,181,212,243,273,304,334]
                          dayIndex = day + starts[k]
- 1
                      elif day==29 or day==30:
                          # All months except
February have a day 29 and a day 30
                          starts = [0,
59,90,120,151,181,212,243,273,304,334]
                          dayIndex = day + starts[k] - 1

                    else:
                          # Only January, March, May, July, August,
October, and December have
                          # a day 31.
                          starts =
[0,59,120,181,212,273,334]
                          dayIndex = day + starts[k] - 1

      self.RiseTime[dayIndex] = RiseTimeList[k]
                      self.SetTime[dayIndex] =
SetTimeList[k]
       # f.close()

    def SunUp(self):

""" Returns a length-365 numpy array D with the
        property that D[k] is
the sun-up time on day k.

        Note: "sun-up" time is the time (in hours
from sunrise to sunset
        """
        return self.SetTime - self.RiseTime


    def MonthAves(self):
        """ Returns a length-12 numpy array M with
the
        property that M[k] is the average sun-up time
        during month k.

"""
        M = zeros(13)
        D = self.SunUp()
        start  = [0, 31, 59,
90, 120, 151, 181, 212, 243, 273, 304, 334,365]
        for k in range(12):
            z =
D[start[k]:start[k+1]]
            M[k+1] = sum(z)/len(z)
        return M
```