# Simple example of functions

def wait():
    x = input()

def Birthday_Engelbert():
    # no parameter(s)
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday, dear Engelbert.")
    print("Happy Birthday to you!")

def Birthday(someone):
    # parameter --- now you can wish anyone
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday, dear", someone + ".")
    print("Happy Birthday to you!")

def main():
    wait()
    Birthday_Engelbert()
    wait()
    Birthday("Engelbert")
    wait()
    Birthday("Robert")

#(Here is the data for the following code, in 2.py. Save it in "wishfile.txt")

Happy, Hannukah, Igor
Merry, Xmas, Murray  
Glorious, Everyday, Mom  
Groovy, Valentine's day, Valerie

#2.py  
# Intro to functions

# Say we want to sing the "Happy Birthday" song to 3 people. The song  
# has 4 lines.

# Option 1: Write a main() program with 3*4 = 12 lines of "Happy Birthday"

# Option 2: Write a function for each line, call the function 12 times

# Advantages? It makes the code look cleaner, and more importantly,  
# if you are in a good/creative mood and want to change the "Happy" to  
# "Joyous", you only need to change it in one place!

# Now what if want the same tune to work for Merry Xmas and Happy Hannukah?

# Let's first wish a couple of people and then read names  
# as strings from a file and wish each name a given wish

def wait():
    x = input(" ")

def Line(wish, event):
    print (wish, event, "to you!")

def Nline(wish, event, name):
    print(wish, event, "dear", name+".")  # want no space between name and period

def Sing(wish, event, name):
    Line(wish, event)
    Line(wish, event)
    Nline(wish, event, name)
    Line(wish, event)

def main():

    print(" ")
    Sing("Happy", "Birthday", "Igor")

    print(" ")
    Sing("Happy", "Hannukah", "Murray")
Sing("Merry","Xmas","Xerxes")

wait()

wfile = open("wishfile.txt","r")
for line in wfile:
    print(" ")
data = line[:-1].split(",")
#remove \n or "." will go on next line
Sing(data[0],data[1],data[2]) #when you print

#(Here is the data for the following code in 3.py. Save it in "googdatafile.txt"

Date Open High Low Close Volume Adj Close*
Feb 6, 2015 527.64 537.20 526.41 531.00 1,744,600 531.00
Feb 5, 2015 523.79 528.50 522.09 527.58 1,840,300 527.58
Feb 4, 2015 529.24 532.67 521.27 522.76 1,656,800 522.76
Feb 3, 2015 528.00 533.40 523.26 529.24 2,029,200 529.24
Feb 2, 2015 531.73 533.00 531.20 518.55 2,826,300 528.48
Jan 30, 2015 515.86 515.87 515.52 534.52 5,581,100 534.52
Jan 29, 2015 511.00 510.99 501.20 510.00 3,834,700 510.00
Jan 28, 2015 522.78 522.99 510.00 510.00 1,674,100 510.00
Jan 27, 2015 529.97 530.70 518.19 518.63 1,897,300 518.63
Jan 26, 2015 538.53 539.00 529.67 535.21 1,532,400 535.21
Jan 23, 2015 535.59 542.17 533.00 539.95 2,266,800 539.95
Jan 22, 2015 521.48 536.33 519.70 534.39 2,653,600 534.39
Jan 21, 2015 507.25 519.28 506.20 518.04 2,244,300 518.04
Jan 20, 2015 511.00 512.50 506.02 506.90 2,221,800 506.90

#3.py

# Graphing stock price using candlesticks. In a previous lecture we wrote
# a program to only graph the X and axes. No we will us that same program,
# read in some GOOG (Google) price data from Yahoo history and draw a
# simple candlestick chart. We will only use what are called closed
# candlesticks.

def wait():
    dummy = input(" ")
def getwindow(text, sizex, sizey):
    # this is an example of a "function"
    # think of it as a helper you call to do some work for you
    # You'll see that getwindow does not have anything to work with unless
    # whoever calls it also gives it something to work with. These are
    # "parameters" of the function, or variables that act as placeholders.
    win = GraphWin(text, sizex, sizey)
    # win is "local" to this function getwindow(). Whoever called getwindow()
    # does not know this variable and is asking for its value, so it can
    # reach
    # the window that getwindow() creates
    # So getwindow() gives this value to the caller via the "return"
    # statement
    # It finishes doing its job and RETURNS THE VARIABLE's VALUE
    return (win)

def label_yaxis(w, x, start, stop, step):
    # Inside this function is code very much like what we did for the x-axis
    # We have to give the function parameters to work with
    # We'll give it the window w, the x-axis, and the y-axis values at which
    # we want little label lines and label values
    # Remember that (xl+3,yl+25) is the (0,0) of our axes
    # If you look inside the code for this function you will see what it does
    # Put little lines and labels next to those lines on the y-axis
    for i in range(start, stop, step):
        if (i > start):
            lin = Line(Point(x - 0.10, i), Point(x + 0.10, i))
            lin.draw(w)

            t = Text(Point(x - 0.25, i), str(i))  # convert number i to tex
            # by asking str() for help
            t.draw(w)

    # between two labeled points, add a little mid-point line without a label
    # this if-statement prevents a little mid-point line from printing
#above the blue y-axis line

    if (i < (stop-25)):
        midpt = Line(Point(x-0.05, i+step/2),Point(x+0.05,i+step/2))
        midpt.draw(w)

#--------------------------- END Y-label function----------------------

def label_xaxis(w, xl, yl, xh, yh):

    # now draw both axes
    # the bottom left corner, i.e., (0,0) will be at (xl+3,yl+25)

    horz = Line(Point(xl+3,yl+25),Point(xl+17,yl+25))
    horz.setOutline("blue3")
    horz.setWidth(3)
    vert = Line(Point(xl+3,yl+25),Point(xl+3,yh-25))
    vert.setOutline("blue3")
    vert.setWidth(3)

    vert.draw(w)
    horz.draw(w)

    #remember that (xl+3,yl+25) is the (0,0) on the axes, and
    #at the top right of the axes system is the high value (xl+14,yh-25)

    #you don't see the high value, but knowing it's there makes you
    #think about the square in which you will draw your graph

    #----------------------------------------------------------------------
    #now put labels for 11 days on the x-axis. We'll print a tiny vertical
    #line at every unit on the x-axis, but use an "if-statement" to skip the
    #origin because we really don't need such a line there

    for i in range(0,14,+1):

        if(i > 0):
            lin = Line(Point(xl+3+i,yl+25+1),Point(xl+3+i,yl+25-1))
            lin.draw(w)

        t = Text(Point(xl+3+i-0.05,yl+25-6),str(i)) #convert number i to
text
        t.draw(w) #by asking str() for help

    #--------------------------- END X-Label function----------------------
from graphics import *

def main():
    #get window()
    w = getwindow("GOOG price-chart", 1600, 800)

    #inside this window our chart has to fit someplace
    #let the bottom left corner of the window be at (xl,yl) and the
    #top right corner be at (xh,yh)
    #xl(i.e.,xlow), xh, yl and yh will b our new coordinate system
    xl = -3
    xh = 15
    yl = 450
    yh = 600

    #inside this window our chart has to fit someplace, and we need
    #an x-axis and a y-axis
    #let the bottom left corner of the x-y graph be at (0,475) and the
    #top right corner be at (11,575). So we will have space for 10 data
    #points on the GOOG graph. We can easily make this much larger later.
    w.setCoords(xl,yl,xh,yh)    #remember (x,y) low point, (x,y) high point

    # now draw both axes
    # the bottom left corner, i.e., (0,0) will be at (xl+3,yl+25)
    label_xaxis(w,xl,yl,xh,yh)

    label_yaxis(w,xl+3,yl+25,yh,25)    # yl+25 because we start label at 475

    #Notice how, by calling some function to do a clear piece of work, you
    #can farm out such tasks cleanly, and reduce clutter in the main program
    #
    get_data(w,xl,yl,"googdatafile.txt")

def get_data(w,xl,yl,dfile):

```python
d = []
sfile = open(dfile,"r")
for line in sfile:
    data = line[:-1].split(" ")  # split using space char
    d.append(data[2].split("\t"))  # remove tab characters

d.reverse()

for j in range(0,len(d)-1,1):
    prices = d[j]
    print(prices)
    o = eval(prices[1])  # open, high, low and close
    h = eval(prices[2])
    l = eval(prices[3])
    c = eval(prices[4])

    # Now we will skip the first data point and draw each candle in the loop.
    # We will use pairs of consecutive "closes" (i.e., c) to tell if the second
    # candle of each pair is red or green

    # We are ignoring the cases of "open" green/red candles and showing all
    # candles
    # as closed

    if (j == 0):
        cprev = c  # saving first closing price
    else:
        #print(o,h,l,c
        line = Line(Point(xl+3+j,l),Point(xl+3+j,h))
        line.draw(w)
        rect = Rectangle(Point(xl+3+j-0.2,o),Point(xl+3+j+0.2, c))
        if (cprev < c):  # green if today's close is
            rect.setFill("green")
            rect.setOutline("green")
        else:
            rect.setFill("red")  # otherwise candle is painted
            rect.setOutline("red")
    cprev = c
rect.draw(w)
```
