Read: Chapters 5 and 6 from Python Programming in Context

Read: Read about TkInter at https://wiki.python.org/moin/TkInter.

Lab03: Visualizing Stocks

In this lab we will develop a way to visualize the stock prices for stocks in a 2D graph. Interactively, from the console, your final application will prompt the user for a stock trading symbol (such as "AAPL") and the start and ending dates strings of interest (for example, from "2012-01-01" to "2014-01-01"). Then your application will fetch the stock data from the ichart.finance.yahoo.com website (see lecture slides).

Your application will read the stock data, ignoring any stocks whose date is out of range.

Finally, your application will generate and display an x-y plot of the data. The minimum requirements are shown below (title, grids, labeled axis, dots).

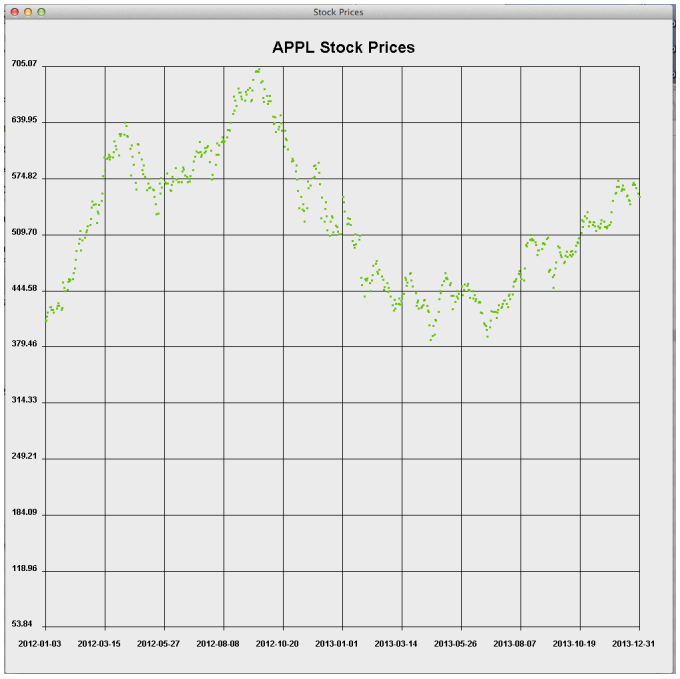


Figure 1: Minimum Visual Output

What objects do we need to achieve our goal? What does our computational space look like? Who is responsible for what activities?

MainFrame: presents our human interface in an TkInter image window. Your mainframe's __init__ method should customize the window

ChartingCanvas: paints the x-y plot (graph) based on the data. Define getters and setters for configure:

- vertical grid spacing
- horizontal grid spacing
- vertical axis title
- horizontal axis title
- chart title
- data – a list of x,y tuples.

Define private helper methods:

- drawing vertical line from (x1,y1) given length in pixels
- drawing horizontal line from (x1,y1) given length in pixels

The plot method should:

- draw and label the horizontal grid
- draw and label the vertical grid
- draw a bounding box and draw the plot title
- draw the data points

The main frame will create, initialize, and hold onto an instance of a StockReader object. The mainframes __init__ method should receive the trading *symbol* and *start* and *end* dates in order to pass these on to the StockReader helper object.

StockReader: fetches the data and creates/holds a collection of stock instances. Since an instance of this class owns the filtered list of stocks, it can answer questions about the stocks (smallest price, largest price, earliest date, latest date), can also return a collection of date, price tuples for the closing stock.

Hints:

- Your **__init__** should receive the stock trading symbol and start and end dates as parameters.

- Your **__init__** method should fetch the data from the following URL: http://ichart.finance.yahoo.com/table.csv?s=AAPL

- Your **__init__** method should prepare a collection of Stock instances for all stocks whose date is between the *start* and *end* dates.

- Define a method for deriving the **smallest low price** from the resulting list of stocks. This method will eventually be used by the Canvas' paint algorithms.

- Define a method for deriving the **largest high price** from the resulting list of stocks. This method will eventually be used by the Canvas' paint algorithms.

- Define a method for deriving the **earliest date** from the resulting list of stocks. This method will eventually be used by the Canvas' paint algorithms.

- Define a method for deriving the **latest date** from the resulting list of stocks. This method will eventually be used by the Canvas' paint algorithms.

- Test your object using unit testing framework.

ViewPort: a convenient helper object (we design) that helps convert world data (dates, prices) can be converted to pixel coordinates. After your canvas gets the StockReader object to acquire the data, your canvas should create, configure, and hold onto a viewport object. You should design the viewport object hold self variables (vl, vr, vt, vb … wl, wr, wt, wb).

- Define a convenient __init__ method.

- Provide methods for setting the viewing extent (vl,vr,vt,vb) and the world extent (wl,wr,wt,wb). After your canvas has created/configured the StockReader helper, you should use create/configure the viewport helper. Hardcode the viewing extent and set the world extent from the min,max method results of the StockReader.

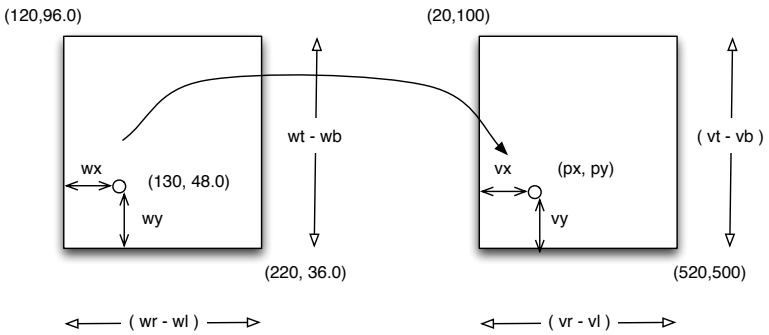
- Define a method for converting from wx, wy (date,price) to vx, vy (pixel, pixel)….. **toPixels(wx,wy) : (int, int)**

- Use your viewport helper object whenever to you paint. It is best to paint based on "world" coordinates.

- Test your object using the unit testing framework.

Stock: encapsulates information about a single stock on a single day. This includes the current *data* (see python date module) as well as the *opening* price, *closing* price, *high* price for the day, *low* price for the day, and the *volume* (ie, the number of shares that was bought and sold).

Mapping from world coordinates (wx, wy) to screen pixels (vx, vy) uses the rule of proper proportions.



vx = vl + (vr – vl) * (wx – wl) / (wr – wl)
vy = vb + (vt – vb) * (wy – wb) / (wt – wb)

vl: viewport left (in pixels) – the horizontal position of the left side of the drawing area

vr: viewport right (in pixels) – the horizontal position of the right side of the drawing area

vt: viewport top (in pixels) – the vertical position of the top side of the drawing area

vb: viewport bottom (in pixels) – the vertical position of the bottom side of the drawing area.

wl: world left (date) – the horizontal minimum date

wr: world right (date) – the vertical maximum date

wt: world top (price) – the largest vertical value

wb: world bottom (price) – the smallest vertical value