Given a sequence of values $[v_0, v_1, \ldots, v_n]$ where $0, 1, \ldots, n$ represents time, a candlestick is the most popular visualization that is used to both summarize and inform financial analysts. The colors, red and green, indicate whether the valuation has negatively or positively changed, respectively. The line gives the range of valuations. The rectangle's height indicates the starting and ending value for that time period.

For a given sequence, a box and lines are drawn as shown in Fig. 4 (right)

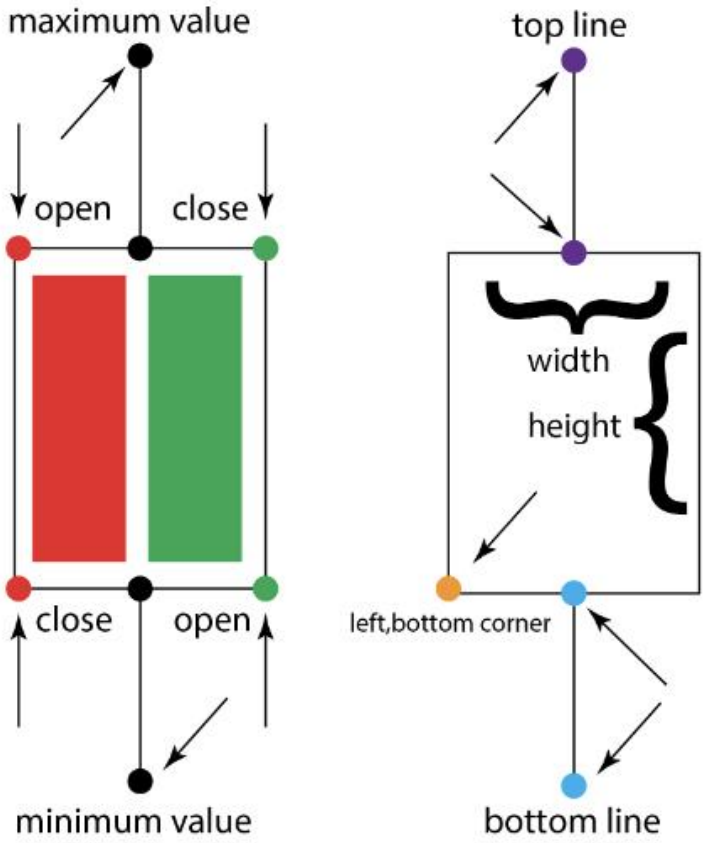

Figure 4: A candlestick image (right). The box is created by the opening and closing values. The top line is from the maximal value to the top middle of the box. The bottom line is from the minimal value to the bottom middle of the box. The color is determined by the open, close values. If open is greater than close, then the color is red; otherwise it's green. We use matplotlib patches. The candlestick image is determined by the rectangle and two lines. The rectangle is determined by the lower-left point (x, y), width, and height. The two lines are determined by two points computed from max, min, open, and close.

Let's look at the code:

```
1  data = [[20,15,32,10],[10,14,15,9],
2          [22,23,27,9],[15,16,16,15],
3          [26,12,30,2],[5,30,40,4]]
4
5  # open,close,max_p,min_p = 20, 15, 32, 10
6
7  # INPUTS ith candle, starting value of x, default width, and the four ↩
       critical values: open, close, max\_p, min\_p.
8  # RETURN three tuples: (point, width, height, color), topline, ↩
       bottomline
9  # point is the coordinates of the lower-left point x, y, width and ↩
       height are numeric values and color will be a string of color ex. '↩
       red' or 'green'
10 # topline ((xt0,yt0),(xt1,yt1)) line from max to top middle of box
11 # bottomline ((xb0,yb0),(xb1,yb1)) line from min to bottom middle of ↩
       box
12
13 # When you see the code for testing Problem5 under __main__, you will ↩
       see that the first three values of the first tuple i.e., point, ↩
       width and height are passed as the first arguement of matplotlib.↩
       patches.Rectangle() function and the last value i.e. color is ↩
       passed as the second arguement. Feel free to play around with the ↩
       test code to get a feeling of how it is working. You will ↩
       understand it much better with a bit of experimentation.
14
15 def make(i, start, width_default, d):
```

```
16      pass
17
18  fig = plt.figure()
19  ax = fig.add_subplot(111)
20  start = 0
21  default_width = 10
22  for i in range(len(data)):
23
24      candle_box, top_line, bottom_line = make(i,start,default_width, ↩
            data[i])
25      print(candle_box)
26      ax.add_patch(matplotlib.patches.Rectangle(*candle_box[0:3],color =↩
            candle_box[3]))
27      plt.plot([x for x,_ in top_line],[y for _,y in top_line],'black')
28      plt.plot([x for x,_ in bottom_line],[y for _,y in bottom_line],'↩
            black')
29      start += default_width
30
31  plt.xlabel("time (hour)")
32  plt.ylabel("Stock X price")
33  plt.title("Candlestick for Stock X mm/dd/yyyy")
34  plt.xlim([0, 60])
35  plt.ylim([0, 35])
36  plt.show()
```

will produce

```
1  ((0, 15), 10, 5, 'red')
2  ((10, 10), 10, 4, 'green')
3  ((20, 22), 10, 1, 'green')
4  ((30, 15), 10, 1, 'green')
5  ((40, 12), 10, 14, 'red')
6  ((50, 5), 10, 25, 'green')
```

and the plot in Figure 5. We can look at some of the Walmart data. Specifically,

```
1       wms = pd.read_csv('walmart_stock.csv', index_col='Date',↩
            parse_dates=['Date'])
2       # print(wms.head())
3       wms_c = wms[['Open','Close','High','Low']]
4       print(wms_c.iloc[0].values.flatten().tolist())
5       N_stocks = 50
6       for i in range(N_stocks):
7           candle_box,top_line,bottom_line = make(i,start,default_width,↩
                wms_c.iloc[i].values.flatten().tolist())
8
```

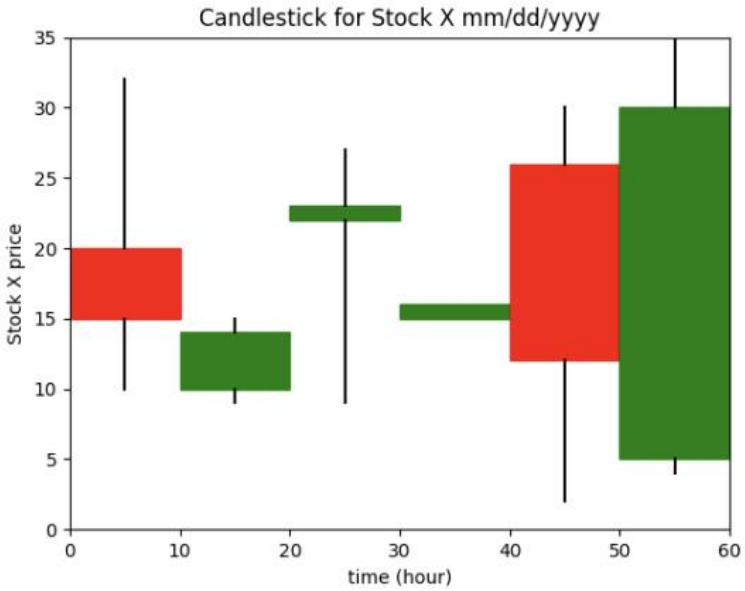

Figure 5: Six candlesticks

```
9           # print(candle_box)
10          ax.add_patch(matplotlib.patches.Rectangle(*candle_box[0:3],↩
                color = candle_box[3]))
11          plt.plot([x for x,_ in top_line],[y for _,y in top_line],'↩
                black')
12          plt.plot([x for x,_ in bottom_line],[y for _,y in bottom_line↩
                ],'black')
13          start += default_width
14
15      plt.xlabel("time day")
16      plt.ylabel("Walmart price $")
17      plt.title("Candlestick for Walmart")
18      plt.xlim([0, N_stocks*default_width]) #depends on number
19      plt.ylim([58, 63]) #depends on price
20      plt.show()
```