THE UNIVERSITY OF NEWCASTLE AUSTRALIA School of Design, Communication, and Information Technology

Inft1004 Introduction to Programming – Assignment 2

Due by 3pm on Tuesday 12 May

Weighting 30%

Paired work Students are permitted and encouraged to work in pairs on this assignment

Your assessment task

Your task for this assignment is to write a program in JES (Python) that reads some stock market data from a file and draws some plots related to this data. The program will also save some of the plots as jpeg image files.

The program will contain a number of functions, but it *must* include two functions with these names:

1) `showVolumeChart(sizeOption)`
2) `drawCandlestickChart(inputDataFile, outputImageName, sizeOption)`

Be very careful with the spelling and definition of these two functions, because your code will be tested by calling these functions as they are specified here. Even if you make only a slight spelling mistake it may not be possible to test the function correctly and hence you will lose marks. In general you should be very careful to meet all requirements as specified in this document and carefully follow the process required for submitting deliverables. Once again failure to do this may mean you lose marks.

`showVolumeChart(sizeOption)`

Note that the function called `showVolumeChart(sizeOption)` takes one argument. This argument, called *sizeOption*, can only take the values, 1, 2, or 3, and will tell the function what size chart to produce. Option 1 will produce small charts with a width of 600 pixels. Option 2 will produce medium charts with a width of 700 pixels. Option 3 will produce large charts with a width of 800 pixels. The height of the chart will always be three quarters of the width. (Some examples of volume charts have been provided in the Examples/VolumeChart folder.)

This function will allow the user to pick a file that contains the stock market data and will then show on the screen a volume chart drawn from the data. The data file will be in comma separated values (csv) format, a format that is supported by Excel and by text-editors. If you open the file in Excel you will see that it contains text and numbers in rows and columns.

The first line (row) of the file will contain header information that identifies what data each column contains. In the assignment the header information will always look the same.

Header row of data:

```
"Date,Open Price,High Price,Low Price,Close Price,Volume (million)"
```

Every other line (row) of the file will contain a string consisting of values separated by commas. In the stock market data used in the assignment there will always be six values (columns).

Typical row of data:

```
"19/03/15,30.58,30.84,30.4,30.58,9626"
```

The example file *StockDataBKP.csv* is provided with this assignment and contains the data shown in the table below.

StockDataBKP.csv					
Date	**Open Price**	**High Price**	**Low Price**	**Close Price**	**Volume (million)**
6/03/15	32.5	32.78	32.35	32.64	6.6
9/03/15	32.14	32.22	31.77	31.9	6.2
10/03/15	32.03	32.91	32.23	32.78	8.3
11/03/15	32.5	32.56	32.13	32.33	12.5
12/03/15	32.1	32.66	32.01	32.2	7.9
13/03/15	32.75	32.94	32.67	32.86	7.7
16/03/15	32.4	32.55	32.01	32.38	6.3
17/03/15	32.5	32.89	32.35	32.77	6.1
18/03/15	32.16	32.3	32.07	32.2	7.7
19/03/15	32.58	32.84	32.4	32.58	9.6

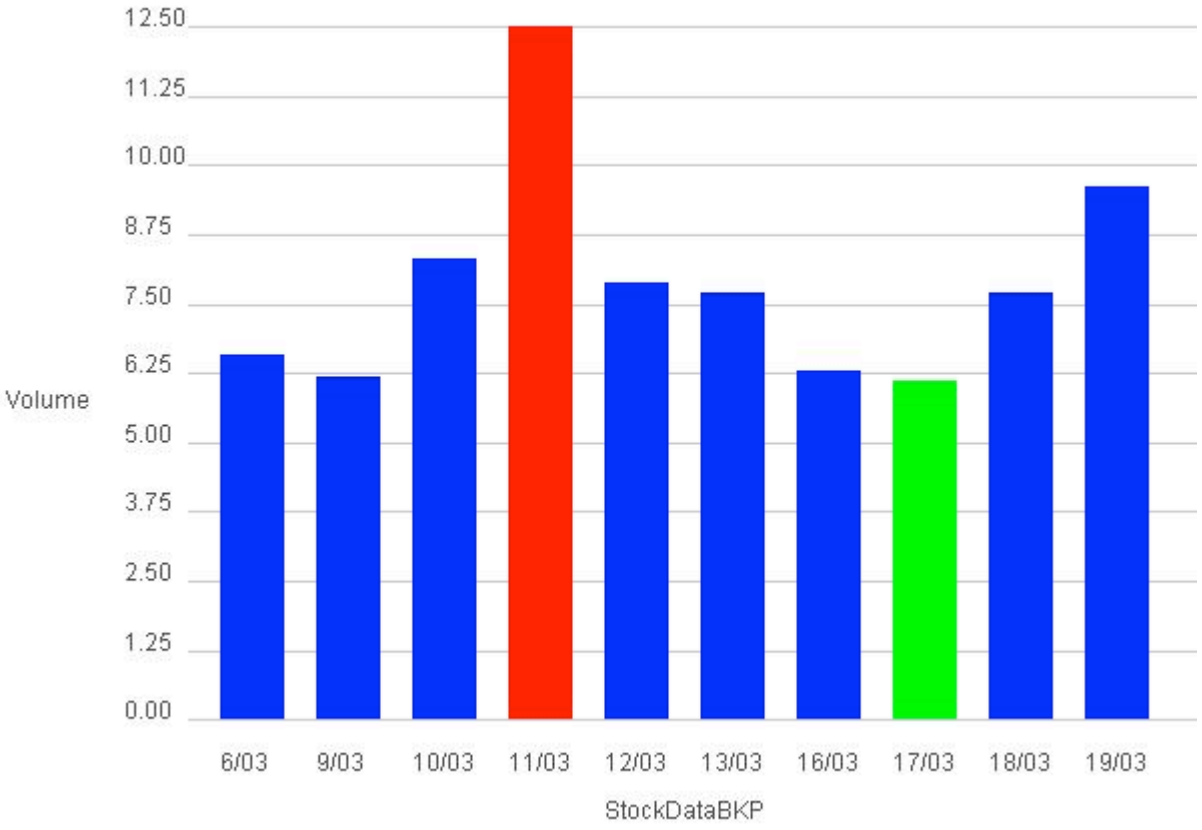
The example file *StockDataWKW.csv*, also provided with this assignment, contains the data shown in the table below.

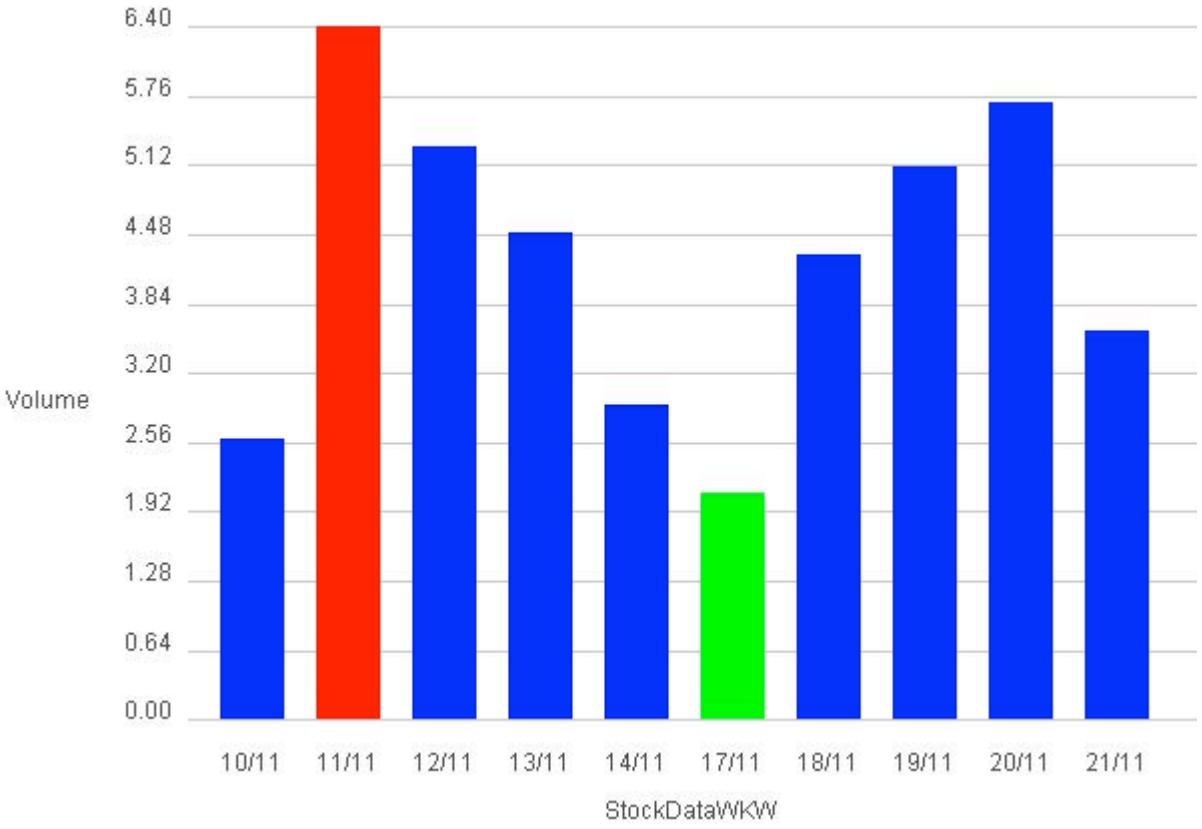
StockDataWKW.csv					
Date	**Open Price**	**High Price**	**Low Price**	**Close Price**	**Volume (million)**
10/11/14	5.8	5.89	5.5	5.64	2.6
11/11/14	5.7	5.92	5.2	5.34	6.4
12/11/14	5.2	5.4	5.07	5.17	5.3
13/11/14	5.4	5.6	5.4	5.55	4.5
14/11/14	5.3	5.7	5.2	5.64	2.9
17/11/14	5.4	5.89	5.5	5.64	2.1
18/11/14	5.32	5.69	5.15	5.54	4.3
19/11/14	5.8	5.8	5.44	5.8	5.1
20/11/14	5.68	5.79	5.23	5.44	5.7
21/11/14	5.58	5.69	5.45	5.64	3.6

You can assume that any data file will always contain 10 days of data and that it will be in basically the same format. Note: The file I will use to test your assignment will have slightly different data but it will be in the same format. (There is no need to do any sophisticated error checking with the data file – as long as your program works with the two files provided it should work with any other data I use to test your programs)

Your function should open the file and read in the data, storing it in a number of lists of floating point numbers – one for each column. Note: one list for each *column*, not one for each *row*. Hint: your work from Assignment 1 should be a good start for reading the data from this file. You will also find that the function `string.split(..)` is useful for breaking up a string with commas in it.

Once you have stored the data in a number of lists, one for each column, you should then process the data to produce a volume chart, which plots the volume of stock traded for each date in the table. Your function needs to be able to produce volume charts like the ones below. (Some more examples of volume charts have been provided in the Examples/VolumeChart folder.)


12.50
11.25
10.00
8.75
7.50
6.25
5.00
3.75
2.50
1.25
0.00
Volume
6/03
9/03
10/03
11/03
12/03
13/03
16/03
17/03
18/03
19/03
StockDataBKP


6.40
5.76
5.12
4.48
3.84
3.20
2.56
1.92
1.28
0.64
0.00
Volume
10/11
11/11
12/11
13/11
14/11
17/11
18/11
19/11
20/11
21/11
StockDataWKW

Note that the height scale of the chart depends on the data in the files. For example compare the values on the vertical axis for the example *StockDataBKP* and *StockDataWKW* charts. The scale on the vertical axis stops at the maximum value recorded for the 10 days of data.

Notice also that most bars are blue in colour. The exceptions are the maximum value bar, which is red, and the minimum value bar, which is green. You can choose whatever colours you like, but be sure to choose three colours that clearly distinguish the maximum and minimum bars from the other volume bars.

Hint: You may find that your maximum and minimum functions from Assignment1 are useful here.

Note that there are 10 grey gridlines on the volume chart and that each gridline is correctly labelled. The horizontal axis shows only the day and month for each bar, not the year. Underneath that axis is the name of the data file used, without the *.csv*. In these examples the lines are light grey and the text is dark grey, but you may any appropriate colours.

Your function should be able to produce these charts in the three possible sizes, showing the results on the screen using the *show()* function.

`drawCandlestickChart(inputDataFile, outputImageName, sizeOption)`

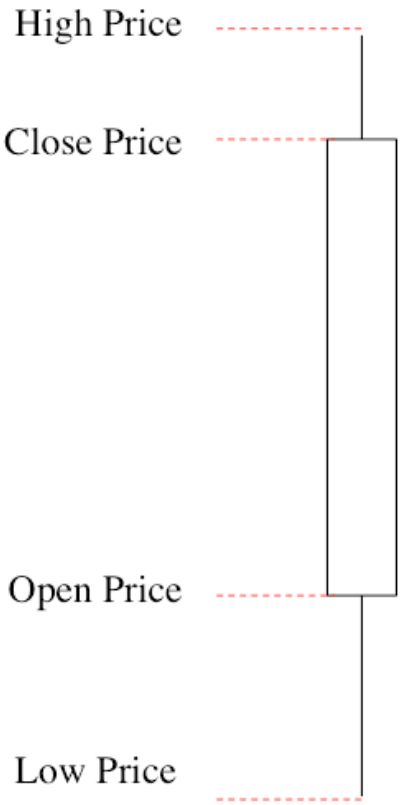
This function will work with the same data as the previous function. However, this function takes three arguments. The first argument, `inputDataFile`, will provide the name of the text file that contains the data your program will read in and analyse. As with Assignment 1, the file will be located in the media path, so before using this function you will need to set the media path correctly *in the command area* of JES.

The second argument, `outputImageName`, will provide the name and extension of an image file that will be used to save your chart from this question. It will be saved in the folder specified in the media path, that is, the same location as the data. Thus this function will both show the result on screen and also save the chart as an image file with the name specified by the string in `outputImageName`.

The third argument, `sizeOption`, works in the same way as in the first function. That is, it can only take the values, 1, 2, or 3, which will direct it to produce charts with widths of 600 pixels, 700 pixels, and 800 pixels. The height of the chart will always be three quarters of the width. (Some examples of candlestick charts have been provided in the Examples/CandleStick folder.)

This function will use the same data files as those used with the previous function, that is, csv files containing the Date, Open Price, High Price, Low Price, Close Price and Volume (million).

Candlestick charts are a specialised form of chart used in stock market trading. The figures (or candles) in this chart have a bar that is either white or black, which shows the difference between the open and close price of the stock. Candles are white if the open price is lower than the close price for the day, and black if the open price is higher than the close price for the day. The ‘wicks’ of the candle are lines drawn in the middle of the bars. The top wick extends up to the high price for the day, and the bottom wick extends down to the low price for the day. Examples of white and black candles are shown below.



white candles – open price is less than close price

black candles – close price is less than open price

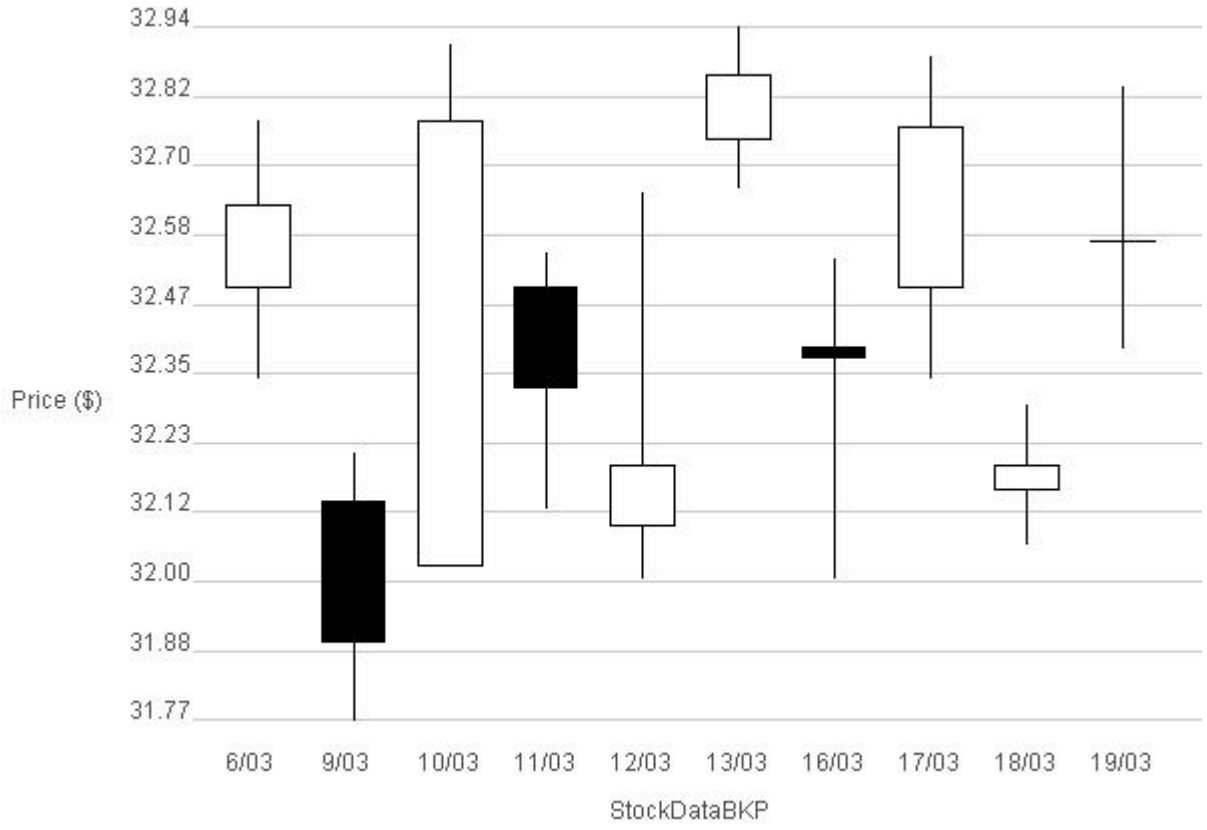
Your function should open the file and read in the data, storing it in lists of floating point numbers, one for each column in the file. Once the data is in the lists, the function will produce candlestick charts like the ones below in the specified size, either small, medium or large. (Some more examples of candlestick charts in different sizes have been provided in the Examples/Candlestick folder.)

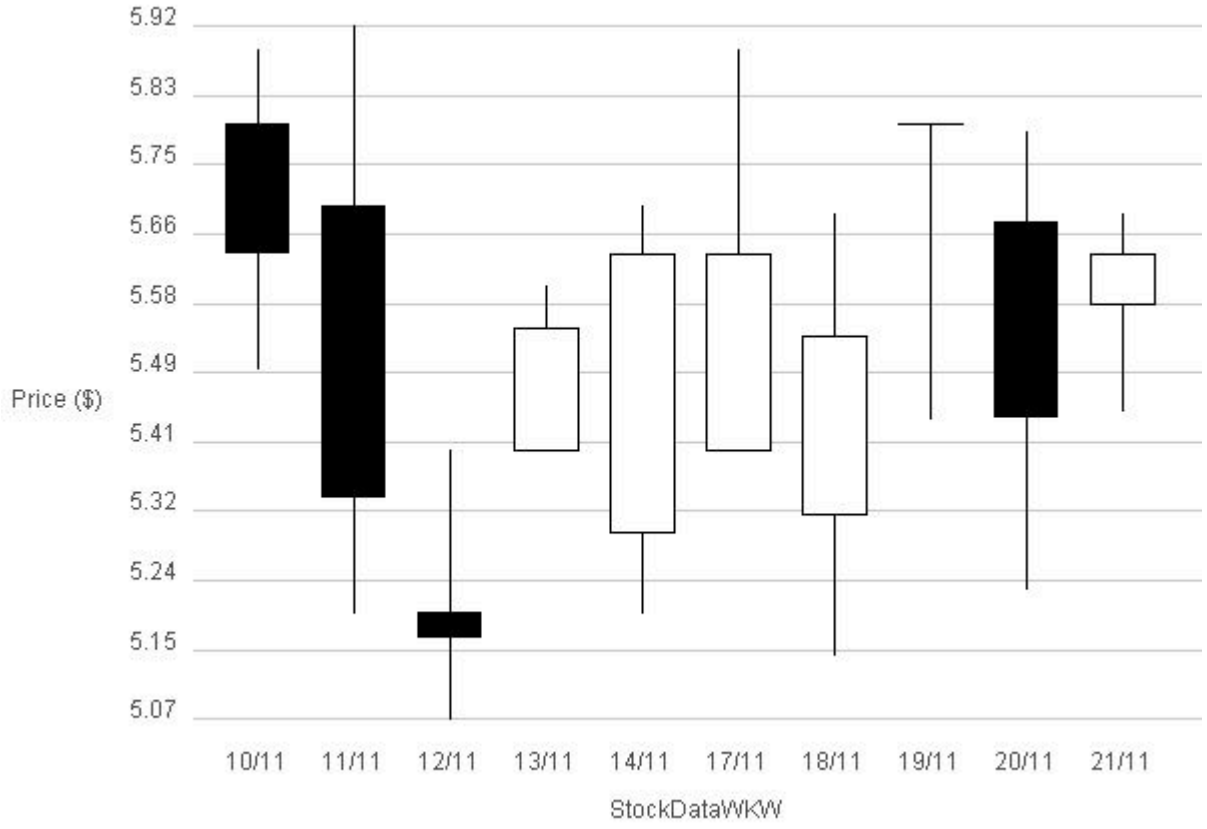
Again the scale of the charts will depend on the data in the files: the lowest gridline will represent the lowest low price over the 10 days, and the highest gridline will represent the highest high price over the 10 days. Confirm this by examining the candle charts for *StockDataBKP* and *StockDataWKW*, below.

Note that there are 10 grey gridlines on the candlestick chart and that each gridline is correctly labelled. The horizontal axis shows only the day and month for each candle, not the year. Underneath that axis is the name of the data file used, without the *.csv*.

Your function should be able to produce these charts in the three possible sizes, showing the results on the screen using the *show()* function; it should also save the chart to an image file, in the media path, using the name specified in the function argument called `outputImageName`.

> *Note: When you are working on your program, in the command area of JES (not the program area) use the JES function* `setMediaPath()` *to locate the folder where the data file is located. Then in your program, you can refer to* `getMediaPath(filename)`*, to both read the data and save the result image. This is essential for proper marking of the assignment.*


32.94
32.82
32.70
32.58
32.47
32.35
32.23
32.12
32.00
31.88
31.77
Price ($)
6/03
9/03
10/03
11/03
12/03
13/03
16/03
17/03
18/03
19/03
StockDataBKP


5.92
5.83
5.75
5.66
5.58
5.49
5.41
5.32
5.24
5.15
5.07
Price ($)
10/11
11/11
12/11
13/11
14/11
17/11
18/11
19/11
20/11
21/11
StockDataWKW

Journal

As programming is a complex task, you are required to maintain and submit a journal, a separate word-processed document in which you indicate when and for how long you work on which aspects of the assignment. You will also briefly list questions that arise, difficulties that you encounter, how you overcome them, and lessons that you learn. Note that if you work in a pair you should also record who is doing which tasks. By the time you've finished the program your journal will probably be at least a few pages long. The journal is intended to record your design thoughts, your planning, your research, your programming thoughts, and the time you spend on the task, so you must keep it up to date at all times. A 'journal' that is thrown together the day before the assignment is due is not a journal at all.

The journal needs to be in pdf format – please do not submit any other format as it will not be marked.

Files and folder

You will be submitting two electronic files, your Python program and your journal, on Blackboard.

Both files will be in a folder whose name is your name, without spaces, followed by the abbreviation *_Assgt2*. So Keith Nesbitt would have a folder called *KeithNesbitt_Assgt2*. If you work in a pair, the name should be the names of both students in the pair. For example, if Keith Nesbitt works with John Brown, the folder's name would be *KeithNesbitt_JohnBrown_Assgt2*.

Within that folder, your Python program will have the same name followed by *.py* (eg *KeithNesbitt_Assgt2.py* or *KeithNesbitt_JohnBrown_Assgt2.py.* If your assignment is not named correctly it may not be marked.

Your journal will have a similar name followed by *_Journal* and the pdf extension (eg *KeithNesbitt_Journal.pdf, KeithNesbitt_JohnBrown_Journal.pdf*). The journal must be in pdf format – please do not submit another format as it will not be marked.

Assessment criteria

Your work will be assessed according to the following criteria:

Your journal, as specified above, clearly showing the design and development process (make sure you include everything asked for).	10%
Programming style (eg functional decomposition, well-named variables, appropriate and useful comments in the code). This should also include a functional, helpful interface (eg error handling, well formatted messages, correct spelling).	15%
`showVolumeChart(sizeOption)` function works correctly, calculates and displays the results correctly.	45%
`drawCandlestickChart(inputDataFile, outputImageName, sizeOption)` function works correctly, calculates, saves, and displays the results correctly.	30%

Once these marks have been allocated, marks will be deducted for the following:

- failure to follow instructions, eg with file names
- syntax errors or runtime errors in your program
- failure to fully and clearly reference any material from external sources, such as function specifications, code written by other people, code with which other people have assisted you, and sources of any other information that you look up
- late submission – see below

If you use external sources to find any code or other information you must reference those sources, giving the details (including the URL for a web page) both in your journal and using a suitable comment

in your program. If you get code from other people or sources, or if other people help you with your own code, you must add comments making this clear.

If the marker uncovers evidence that you have cheated in any way, for example, by sharing your code with people other than your partner, by getting help from anyone other than your partner and not referencing it, or by using specifications without referencing their source, the matter will be reported to the Student Academic Conduct Officer.

Please note that the assignment will be marked on the Macintosh interface in the labs. You should test that your code runs correctly on these machines in the lab before submitting. For example if you develop the code on your own Windows machine you should make sure that it works correctly on the Macintosh interface in the labs. If your program does not run in this environment you may receive no marks.

Handing in your work

You are required to hand in three things. Two of these (your Python source code and the pdf of your journal) are to be submitted electronically using Blackboard's Assignment facility. Because you need to hand in a folder and its contents, you are required to zip together the folder containing the files. (Refer back to *Files and folder* for more details.)

When you zip your folder and its contents, in way that preserves their directory structure, be sure that you produce a *.zip* file, not some other format such as a *.rar* file. There are many zipping software packages, some commercially available, some free, and some provided with operating systems. Well before you submit your assignment, be sure that you have access to appropriate software and know how to use it. Once you have zipped your folder together, be sure to unzip it to a new location to check that it unzips correctly. Also be sure that the zip file has the same name as the folder, ie your name without spaces followed by *_Assgt*: examples are *KeithNesbitt_Assgt.zip*, *KeithNesbitt_JohnBrown_Assgt.zip*.

When you are ready to submit your zipped file, log in to Blackboard, go to the site for this course, and follow these steps . . .

- Select the *Assessment* folder.
- Click the *Assignment 2* link, which will take you to the appropriate upload page.
- In panel 2, *Assignment materials*, click *Browse my computer* (next to *Attach file*), and navigate to your zip file. There's no need to enter a link title. In the comments field put your name if you worked alone, or both of your names if you worked as a pair.
- In panel 3, *Submit*, click the *Submit* button.
- If you don't see a message saying the assignment is complete, go back and check that you've done all these steps. If there's still a problem, try Blackboard's help under *Course tools*, search for *assignment*, and select the help page on *submitting assignments*.
- If you want to submit an updated version of the assignment, go back to the Assignment link and click *Start new submission* on the assignment's *Review submission history* page. Make sure you're aware of the deadline: the final marking will be applied only to the most recent submission, and if it's submitted late it will be marked as late.

The third thing you need to hand it is a physical copy of the School's assignment cover sheet. If you are working as a pair you need to complete a group cover sheet and both people need to sign the cover sheet. The cover sheet must be filled in correctly then signed and handed in by the due date. Do NOT submit this electronically! You can hand the sheet to your lecturer on or before the deadline. (You can also lodge it under his office door – but it must be received by the deadline. Your assignment will be deemed late until this is received.)

You may be required to demonstrate your program, and perhaps to explain your approach, in a later tutorial class.

Paired work

For this assignment you are permitted and encouraged to work in pairs. Obviously, within a pair, collaboration is strongly encouraged. Outside the pair, it is not permitted. Different programs that are judged to have significant parts in common will normally be given marks of zero. For this reason, it would not be a good idea to base your work significantly on other people's work, including work found in books or on the Web. The goal is to see how well you can design and program, not how well you can adapt existing programs, either your own or somebody else's. If you do need to get help from others, for example in debugging code, that should be explained in both the journal and comments in the code.

When two students choose to work as a pair, the naming of folders and files (as described above) should use the name of both students, and the cover sheet, the journal, and the opening comments in the program should clearly indicate the names of both students in the pair. Only one copy of the work needs to be handed in, and both students in the pair will normally get the same mark for the assignment, regardless of who did how much of the work, unless it is clear that this would be a serious injustice.

Deadline and consequences for late submission

The assignment is due by 3pm on Tuesday 12 May. Work will be penalised 10% for every day or part day by which it is late.

For an assignment to meet the deadline, both the Blackboard submission and the signed hardcopy of the cover sheet must be on time. If either is late, the assignment is late. For example, if you hand in your code and journal on time but you don't hand in the signed hardcopy of the cover sheet for 5 days after the due date, you will lose 50% of the possible mark for this assignment.