THE UNIVERSITY OF NEWCASTLE AUSTRALIA School of Electrical Engineering and Computer Science

Inft1004 Introduction to Programming – Assignment

Due Sunday 10th Jan by 11.55pm (Week 11)

Weighting 25%

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 Python program in JES 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.

Journal

As programming is a complex task, you are required to maintain and submit a journal, a separate word-processed (word) document in which contains:

- Title Page:
  - Who is in group, names, student numbers lab session
- Introduction
  - Overview of what is to be completed
- Each Coding Day: Day, Time and Date eg Day 1: Thursday 7pm 10/12/2020
  - record when and for how long you work on which aspects of the assignment, and
  - which bits are done by which member of the pair;
  - briefly list questions that arise,
  - difficulties that you encounter, and
    - eg line 50 still cannot do the drawCircle() function
    - the issue is loop is not right, not sure yet
  - how you overcome them;
    - With line 50 we had the variable in the wrong place, we read some examples on the web (Brown and Bar 2019) and this pointed us to a possible error
  - summarise lessons that you learn.
  - Reference eg sources of code (APA)
- Conclusion

This will be an overview of what you learn and how you would do it differently next time
Include your name/s in a footer and also page numbers

By the time you've finished the program your journal will probably be many pages long. The journal is intended to record your design thoughts, your programming thoughts, and the time you spend on the task, so you must keep it up to date at all times. The journal will need to provide sufficient evidence (documentation) of your development process (design, problem-solving, testing, etc) to be marked. A quickly thrown together assembly of thoughts may be marked accordingly.

When you hand in your files, your journal must be a pdf file.

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 below) 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.

Program Comment

At the beginning of your program you will have the following details and format, as discussed in the lecture and lab classes.

```
####
# Author: Both Authors if pair (or just yours)
# Date Begun: This is the date you started to develop the program
# Date Completed: Date finished (eg day of upload)
# Task: Assignment Stock Data
####
```

Assignment Part 1

The program required for Part 1 of the assignment will read stock market data from csv files and use the data to produce images called volume charts. The program should include the following top-level function.

`showVolumeChart(sizeOption)`

The function called *showVolumeChart()* 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, 600 pixels wide; option 2 will produce medium charts, 700 pixels wide; option 3 will produce large charts, 800 pixels wide. The height of the chart will always be three quarters of the width.

Be very careful with the spelling and parameter of the function, because your code will be tested by calling it by exactly this name, and with exactly the argument specified. Even a slight spelling mistake will mean that the function we are trying to test cannot be found, and 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, or you are likely to lose marks.

This function will allow the user to pick a file that contains the stock market data and will then display 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 date and number of floating-point values separated by commas. In the stock market data used in the assignment there will always be six values (columns).

Typical row of data:

```
19/03/15,32.58,32.84,32.4,32.58,9.6
```

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)**
6/03/20	32.5	32.78	32.35	32.64	6.6
9/03/20	32.14	32.22	31.77	31.9	6.2
10/03/20	32.03	32.91	32.23	32.78	8.3
11/03/20	32.5	32.56	32.13	32.33	12.5
12/03/20	32.1	32.66	32.01	32.2	7.9
13/03/20	32.75	32.94	32.67	32.86	7.7
16/03/20	32.4	32.55	32.01	32.38	6.3
17/03/20	32.5	32.89	32.35	32.77	6.1
18/03/20	32.16	32.3	32.07	32.2	7.7
19/03/20	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.

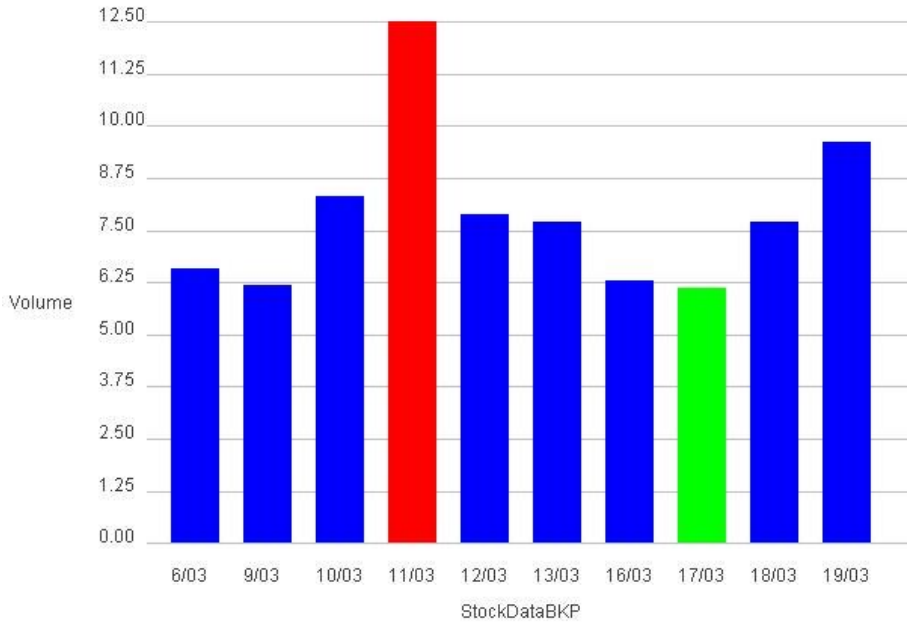
StockDataWKW.csv					
Date	**Open Price**	**High Price**	**Low Price**	**Close Price**	**Volume (million)**
10/11/20	5.8	5.89	5.5	5.64	2.6
11/11/20	5.7	5.92	5.2	5.34	6.4
12/11/20	5.2	5.4	5.07	5.17	5.3
13/11/20	5.4	5.6	5.4	5.55	4.5
14/11/20	5.3	5.7	5.2	5.64	2.9
17/11/20	5.4	5.89	5.5	5.64	2.1
18/11/20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

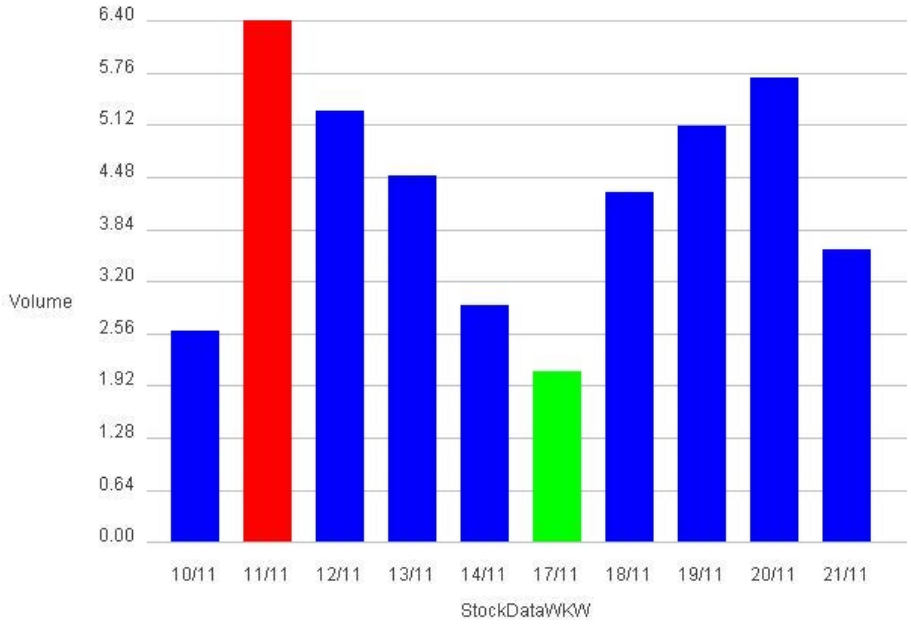
You can assume that any data file will always contain 10 days of data and that it will be in basically the same format. The files that will be used to test your assignment will have different data, but in the same format. There is no need to do any sophisticated error checking with the data file – so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

Your function should open the file (unless the user presses Cancel in the file picker) 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*.

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 on each date in the table.

Your function needs to be able to produce volume charts like the ones below.





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 *StockDataBKP* and *StockDataWKW* charts. The scale on the vertical axis goes from 0 to 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.

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 use any appropriate colours.

We have not specified the exact positions or sizes of the many elements on the chart; you will need to do your own problem solving and design to come up with suitable values. For this reason, we do not require your charts to look absolutely identical to ours.

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

As you write your function, be aware of functional decomposition: if it needs to do several distinct tasks, consider writing a separate function for each of the tasks. In particular, look ahead to the next function, *drawCandlestickChart()*. If there are tasks that both top-level functions need to do, they should definitely be written as separate functions, so that each top-level function can call them.

Assignment Part 2

The program required for Part 2 of the assignment is a continuation of the program for Part 1. It will read stock market data from csv files and use the data to produce volume charts and a new type of chart called a candlestick chart.

`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 and extension of the text file that contains the data your program will read in and analyse. This time 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.

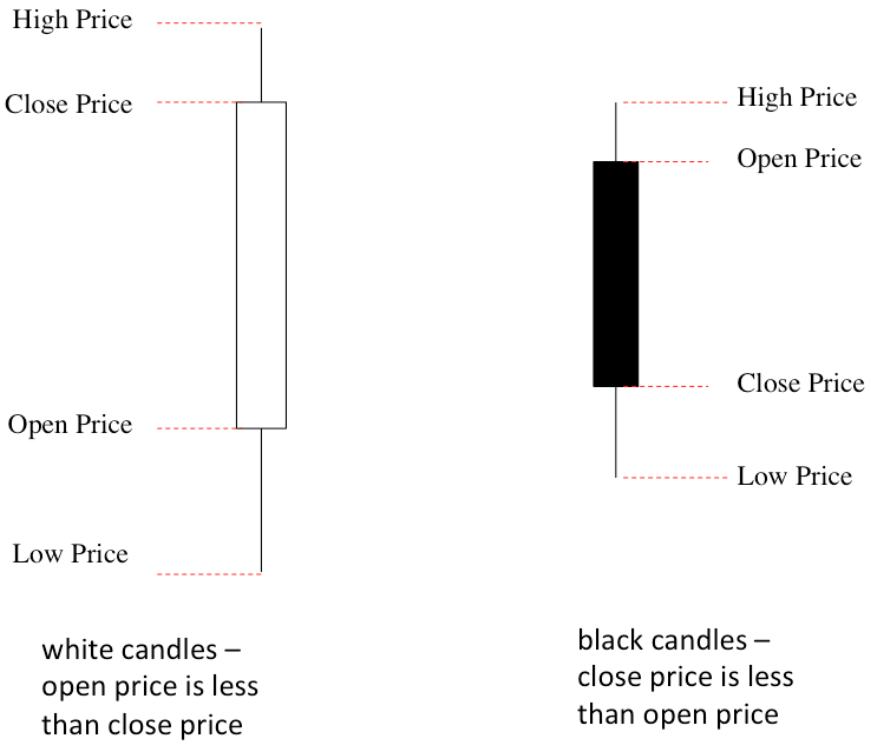
The second argument, *outputImageName*, will provide the name and extension of an image file that will be used to save your chart from this question. It will be saved in the folder specified in the media path, that is, the same location as the data. This function will both show the result on screen and also save the chart as an image file with the name and extension 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, or 800 pixels. The height of the chart will always be three quarters of the width.

Be very careful with the spelling and parameters of the function, because your code will be tested by calling it by exactly this name, and with exactly the arguments specified. Even a slight spelling mistake will mean that the function we are trying to test cannot be found, and you will lose marks.

The 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 vertical 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, and more have been provided for you on Blackboard.



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.

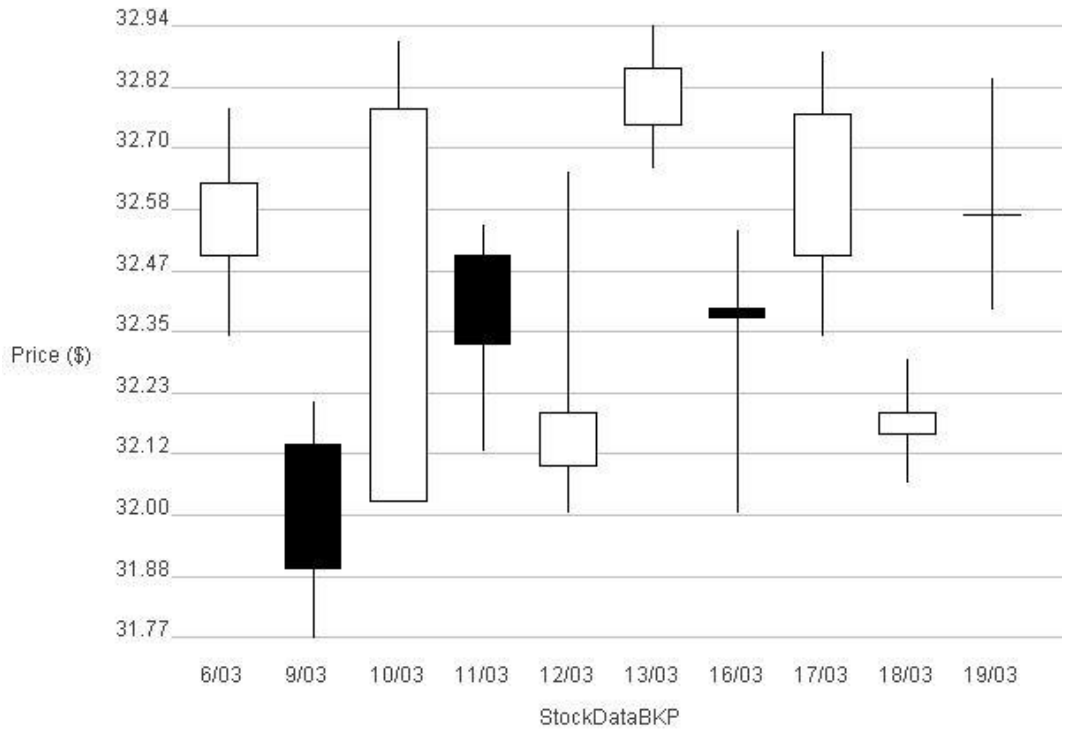
Again the scale of the charts will depend on the data in the files: the lowest gridline will represent the lowest low price over the 10 days, and the highest gridline will represent the highest high price over the 10 days. Confirm this by examining the candlestick charts for *StockDataBKP* and *StockDataWKW*,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

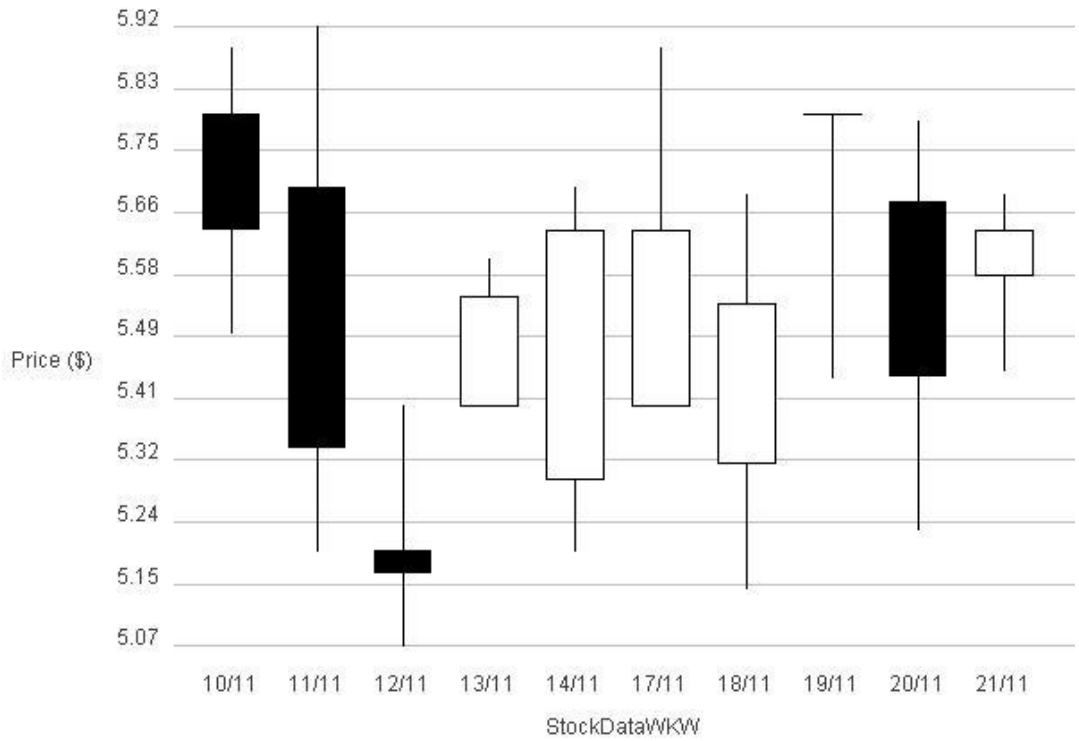
Note that there are 10 grey gridlines on the candlestick chart and that each gridline is correctly labelled. The horizontal axis shows only the day and month for each candle, not the year. Underneath that axis is the name of the data file used, without the *.csv*.

We have not specified the exact positions or sizes of the many elements on the chart; you will need to do your own problem solving and design to come up with suitable values. For this reason, we do not require your charts to look absolutely identical to ours. The biggest aspect of the problem solving will be working out how your program will compute the vertical coordinates of the top and bottom of each candle and each wick. You will probably need to spend substantial time with pen and paper working out how to calculate these.

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





Files and folder

You will be submitting three electronic files, assignment cover sheet, your Python program and your journal, on Moodle.

Both files will be in a folder whose name is your name, without spaces, followed by the abbreviation *Ass1*. So Eugene Lutton would have a folder called *EugeneLuttonAss1*. If you work in a pair, the name should be the names of both students in the pair. For example, if Eugene Lutton works with Simon, the folder's name would be *EugeneLuttonSimonAss1*.

Within that folder, your Python program will have the same name followed by *.py* (eg *EugeneLuttonAss1.py* or *EugeneLuttonSimonAss1.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 *EugeneLuttonJournal.pdf, EugeneLuttonSimonJournal.pdf*). The journal must be in pdf format – please do not submit another format as it will not be marked.

These two files are to be the only files in the folder. Do not submit additional files.

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).	20%
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).	20%
`drawCandlestickChart(inputDataFile, outputImageName, sizeOption)` function works correctly, calculates, saves, and displays the results correctly.	30%
`showVolumeChart(sizeOption)` function works correctly, calculates 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

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 a Windows computer. If you develop the code on a Macintosh or Linux machine you should make sure that it will work correctly on a Windows computer. The principal difference is in file paths, which use a slash on Macs and a backslash on Windows: the string containing a full filename might be "C:/Documents/Uni/Inft1004/Assignment2/StockDataBKP.csv" on the Mac and "C:\\Documents\\Uni\\Inft1004\\Assignment2\\StockDataBKP.csv" on Windows (where each backslash is preceded by another backslash to tell Python to treat it as a literal backslash).

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 and cover sheet) are to be submitted electronically using Moodle 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, 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, 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 *Ass1*: examples are *EugeneLuttonAss1.zip*, *EugeneLuttonSimonAss1.zip*.

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

- Select the Assessment 1 folder.
- Click the Add Submission link, which takes you to the upload page.
- In panel 2, click Add, and browse to your compressed file . Be absolutely sure to choose the right file. There is no need to type a link title. In the comments field enter your name.
- click the Upload this file button.
- Then click Save changes button.

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