**Analysis**

**Introduction**

Numerical Methods is one of the main topics taught in Core 3 A-Level Mathematics. It is split up into different sections:

* Locating Roots
* Staircase and Cobweb Diagrams
* The Mid-ordinate Rule
* Simpson’s Rule

This project will focus on staircase and cobweb diagrams.

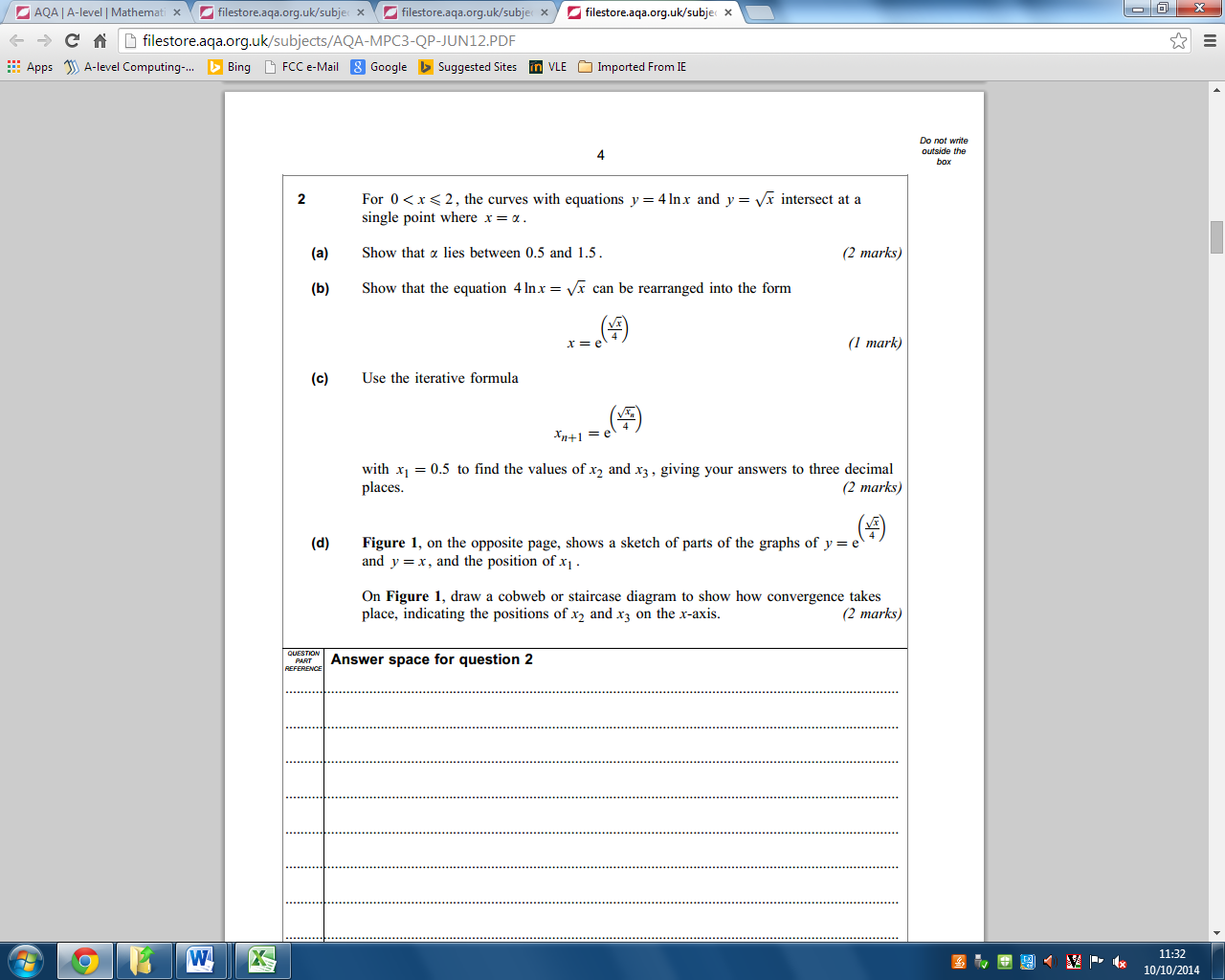
The purpose of using staircase and cobweb diagrams is to solve an equation by expressing it as a numerical relation (iterative formula) that can be solved numerically. In most questions an equation or sequence is given to the student. If an equation is given it usually needs to be rearranged to make x the subject, the result is then equated into a separate y = graph. The diagrams are started by first drawing the graph of the sequence and the graph y = x. Below is a screenshot of a past exam question taken from the AQA website. This is an example of what students are expected to know about numerical methods. Figure 1.2 is a photocopy of an example from the Core 3 for AQA Maths book. This is the method students would use to solve part (d) of the exam question.

Figure 1.1 – Question 2(c) and 2(d) from the June 2012 Core 3 Maths exam paper.

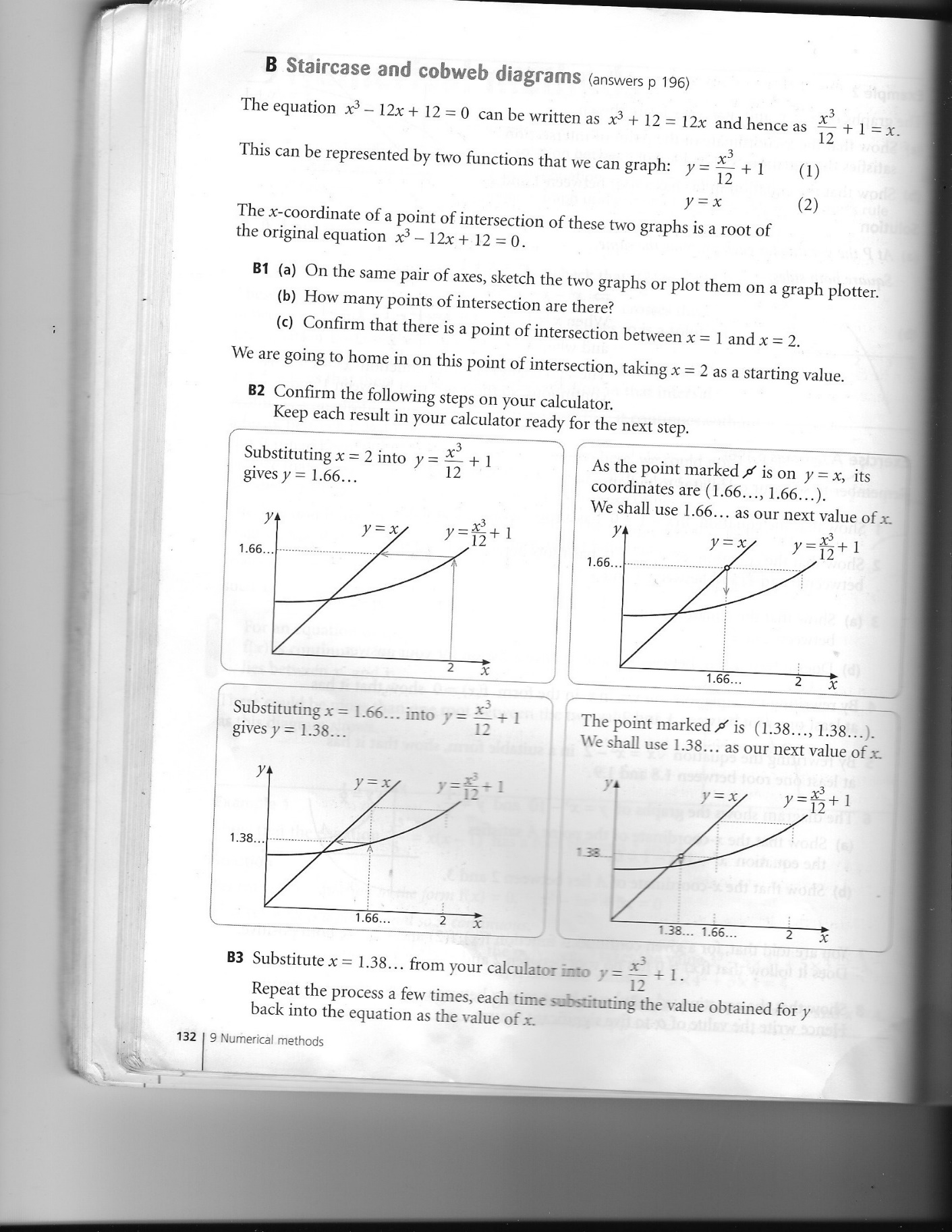
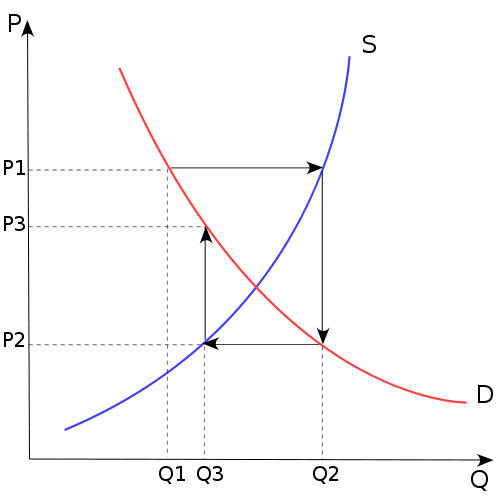


Figure 1.2 – Example of how a staircase diagram is drawn.

The other method of working out the value of xn or the limit is to plot a cobweb diagram. The equations of the graphs dictate whether the student will use a staircase diagram or a cobweb diagram to answer the question.

**Background/Identification of the Problem**

The A-Level Maths students at Felpham Community College sometimes find it difficult to understand the theory behind staircase and cobweb diagrams. As a result when it comes to plotting and drawing the diagrams they find themselves struggling and needing help.

The customer has asked me to create a solution that helps him and the other Maths teachers demonstrate the theory for staircase/cobweb diagrams, as well as allowing students to revise the topics. In this school there are approximately 20 students who are currently taught A-Level Maths and about 5 teachers teaching with it.

**The Current System**

The current method of solving problems that involve staircase diagrams or cobweb diagrams is to solve by drawing the diagram with a pencil and paper. The teacher starts by teaching his students how to solve with staircase diagrams, then moves on to cobweb diagrams (as these are considered more difficult). For both types of diagram the teacher uses a range of examples (from the textbook) to aid the students and gives them multiple exercises that involve drawing out the graphs to test their knowledge. An example that has been made in Excel that can solve the questions straight away might be shown to the students, but the answer is only displayed in numerical form (i.e. not on a graph). Extended learning is done as homework or revision; however it is constrained to using the textbook. This does not provide the best learning experience because there is no immediate feedback (work is handed in and marked), meaning the students might not know why they got a question wrong. The current method is useful for students because in their exams they will have to use a pencil and ruler to draw out their answers.

**Problems with the Current System**

Drawing lots of diagrams during practise takes a long time and small errors when drawing means the student must rub out and correctly redraw the graph, or start over. The time taken to draw each graph and correct errors restricts how effectively the students learn this part of the Maths course; time constraints for the teacher mean he may have to move on when some of his students have yet to fully grasp the theory. It is also difficult for the teacher to convey the theory without having to draw graphs himself as there are no readily available graphical alternatives. This makes it harder to embed the theory into the method, which is a problem, considering the complexity of the topic. Another issue with the current system is repeatability. The textbook that is used has a limited number of questions and examples for each topic. This is disadvantageous because it means the teacher and students have no reason to return to the book after they have finished using its examples.

**Identification of Prospective Users**

The Maths Department at Felpham Community College will use the system to teach numerical methods to A-Level Maths students. The students will also have access to the system so they can use it to revise staircase/cobweb diagrams. The system should be easy to use for all of the users, especially the teachers as they will use it to demonstrate examples to their class.

**Specific Objectives**

I have spoken with the customer to put together a list of specific objectives that the system aims to complete. Here is what the customer would like from the system:

1. The system shall display cobweb diagrams or staircase diagrams on a screen.
2. The system shall display each step taken to create the diagram via a trail.
3. The system shall display the trail for the staircase/cobweb diagram in a different, distinguishable colour. The two line graphs will be black, cobweb trails will be red and staircase trails will be blue.
4. The system should have the option to undo the most recent step it has taken. This will allow the user to work backwards, helping them see how to reach the answer.
5. The system should be able to work for different types of polynomial graph, to a reasonable extent (e.g. the teacher is unlikely to use powers of x that are above 5)
6. The system shall allow the user to input values for the graphs.
7. The system shall be able to run on the school’s computers.
8. The system shall have a ‘new graph’ / a ‘clear’ button so the user can delete the current graphs and workings and start over.
9. The system shall display the x coordinate and the y coordinate of the point where the two graphs intersect. This is the limit (L) the two graphs converge towards. Recursion will be used to achieve this objective.
10. The system shall give the user the capabilities to supply an answer before solving an input problem.
11. The system will then check the input answer against the actual one and display theory if it is incorrect.
12. The system shall have a theory button that can be pressed at any time to display a short demo of the theory.
13. The system should allow the user control over how many iterations are done at any time. This can be implemented with something like a ‘complete 1/5/10 iterations’ button.

**User Needs**

The users of the solution will have varying knowledge of how to use computer applications, the system should therefore be easy to navigate and easy to operate. A help button can be supplied to aid unfamiliar users.

Student users who do not understand the known theory will need access to it within the system. I will implement a short demo that includes the known theory for this.

The users will need instant feedback if they decide to answer one of the questions on the system. I will use a recursive algorithm that can solve the answer to the given question and compare the answer against the user’s answer. The system will then feedback to the user whether they were correct or incorrect.

**Limitations**

Because of the timescale of the project, the system will only be able to help students with staircase and cobweb diagrams from the numerical methods topic.

The system will only be able to work with polynomial graphs and exponential functions. This is so that the system can be implemented within the allotted timescale.

The x and y axes will have a size and scale limit because the functions of the graphs will be within a certain range. The multiplier range will be -15 to 15 or -20 to 20. This is so that unrealistic multiples of x (that can’t be drawn) aren’t inputted into the system.

The system will only be available on the school’s network to begin with. If students wish to take it home they will have to copy the program onto a portable memory storage device from the network.

**Potential Solutions**

**Excel Spreadsheet**

One of the potential solutions to the problem is to expand on the textbook’s example of how to complete an iterative sequence in Excel. The solution would adapt the example’s algorithm and produce a graph for the user to see, as well as displaying answers.

**Advantages**:

Making graphs in Excel is relatively easy to do – many of the functions used for plotting graphs are readily available for me to use.

Most of the users will be familiar with Excel so they will be able to use the system easier.

**Disadvantages:**

Displaying the graph dynamically will be difficult for me to implement. The graph will have to be completed instantly, rather than via the step-by-step method that my customer has asked for. This will also make the undo functionality harder to make.

**Mobile Application**

An application will be able to be downloaded onto mobile devices owned by the users. The app will display the graphs and theory on the mobile’s screen. The users can use the mobile’s keyboard to input parameters.

**Advantages:**

The system will be able to be accessed and used almost anywhere by the users. Students can revise on the go.

There is an added element of interactivity because the users will each be on their own device. The teacher can ask his students to input the parameters during demonstrations and be able to get their individual answers.

**Disadvantages:**

The school has a ‘no mobile devices during lessons’ policy, meaning the students can’t use the app in class.

The application will be used on a small screen. The graphs might be difficult to see on smaller displays and navigation will be more difficult.

It is possible that not all of the students have a mobile device that can run applications, so it is not definite whether all of the users will be able to access the system. Also different mobiles use different operating systems so it would be far too time consuming to create an application that can run on all of them.

Mobile applications are more difficult to make compared to Excel Spreadsheets due to the coding complexity required to create them. I also have no previous experience with creating apps for mobiles, so this solution might be too complicated to implement.

**Python Computer Program (Object-Oriented)**

An application can be created to be used on the school’s computers. Parameters are input into the program and it plots the desired graphs.

**Advantages:**

The system will be accessible to all of the main users – Python is already installed on the school’s computer systems.

Python is a flexible programming language and unlike Excel I can use it to create the requested ‘n steps’ method.

I have learned programming with Python throughout the Computing course so I will have a better knowledge of how to implement the code in it.

**Disadvantages:**

The use of classes and methods when implementing the system will increase the complexity of the project. I will need to decide whether it is necessary to go down this route or if a simpler alternative is more viable.

**Python Computer Program (Procedural)**

An application with a similar design to the OO program will be used on the school’s network. This program will use functions with iterative procedures to gather the user’s inputs instead of classes and objects.

**Advantages:**

The nature of solving problems in Maths is often linear, so creating a procedural program might be the most logical approach for the system’s design. A procedural approach might be quicker to implement.

I have done procedural programming more than OO so I am more accustomed to its style.

I can easily define all of the functions I will need in Python.

**Disadvantages:**

It might be more difficult to make sure the functions work with each other properly compared to creating objects that interact.

Procedural programming is not as flexible as object oriented programming, so making adjustments to the program will be more difficult.

**Justification of Chosen Solution**

I have chosen to create a Python program for the solution. I have decided to use an object oriented approach to the design of the program. I have chosen to do this in Python because it is a flexible programming language and it is the language I have learnt the most of in the Computing course. Python is also already installed on the school’s computers so the solution will be available to all of the prospective users.

I didn’t choose to make a mobile application because of the rules at my school that don’t allow mobiles to be on in classes. Also the displays would be too small for the students to clearly see what’s happening in the application.

The Excel spreadsheet was a good option because it would have been easy to make the graph plotter and perform the calculations. However, the way the solution would have needed to be implemented is not what the customer has asked for. I am also not very familiar with graph drawing functions in Excel, so it was less feasible that the solution would be finished in the set timescale.

I will use the systems development life cycle to help me complete the solution like so:

**Analysis** – Interviews with the main customer will be conducted and I will hand out questionnaires for both the teacher and students to answer about the topic.

**Design** – My design will include class diagrams and interaction diagrams to help me with the implementation stage. I will also create a GUI design so that I have an idea of what the interface should look like. This design will be shown and edited at the customer’s request.

**Implementation** – The program will be created in Python because it is the programming language that I am most familiar with. I will be able to use these skills and my designs to make an effective user-interface for the customer.

**Testing** – The Maths teachers and students will be helpful testers for the program, being able to feedback if something doesn’t work how they’d like it to. I will use black box testing in this stage because it is simple and effective.

**Evaluation** – I will use results collecting during development and a final customer feedback to decide whether the solution has fulfilled its original objectives. If it hasn’t I will return to the Analysis stage to start the cycle again.

**Evidence of Analysis**

Throughout the Analysis stage of development I have had contact with my customer. I discussed and noted a list of objectives for my solution and made them into a draft list of SMART objectives. I returned to the customer with the list, where edited and finalised it according to what he told me.

I have set up a time with my main customer when, once a fortnight, I see him and show the state of the current solution. I made a prototype that could create graphs from certain entered equations (polynomials and exponentials) and wrote feedback. We decided that it would be easier if the user first rearranged the given equation manually before entering the coefficients as parameters into the program.

**Design**

**Overall Design**

|  |  |  |
| --- | --- | --- |
| Inputs | Processes | Outputs |
| Equations.  Coefficients.  Type of diagram to be drawn. | Calculate the root of an equation.  Draw the diagram. | Graph drawn on x and y axes.  The root of the equation.  Steps taken to reach the root. |

**Context Diagram**

This is a small context diagram showing the users of the system.

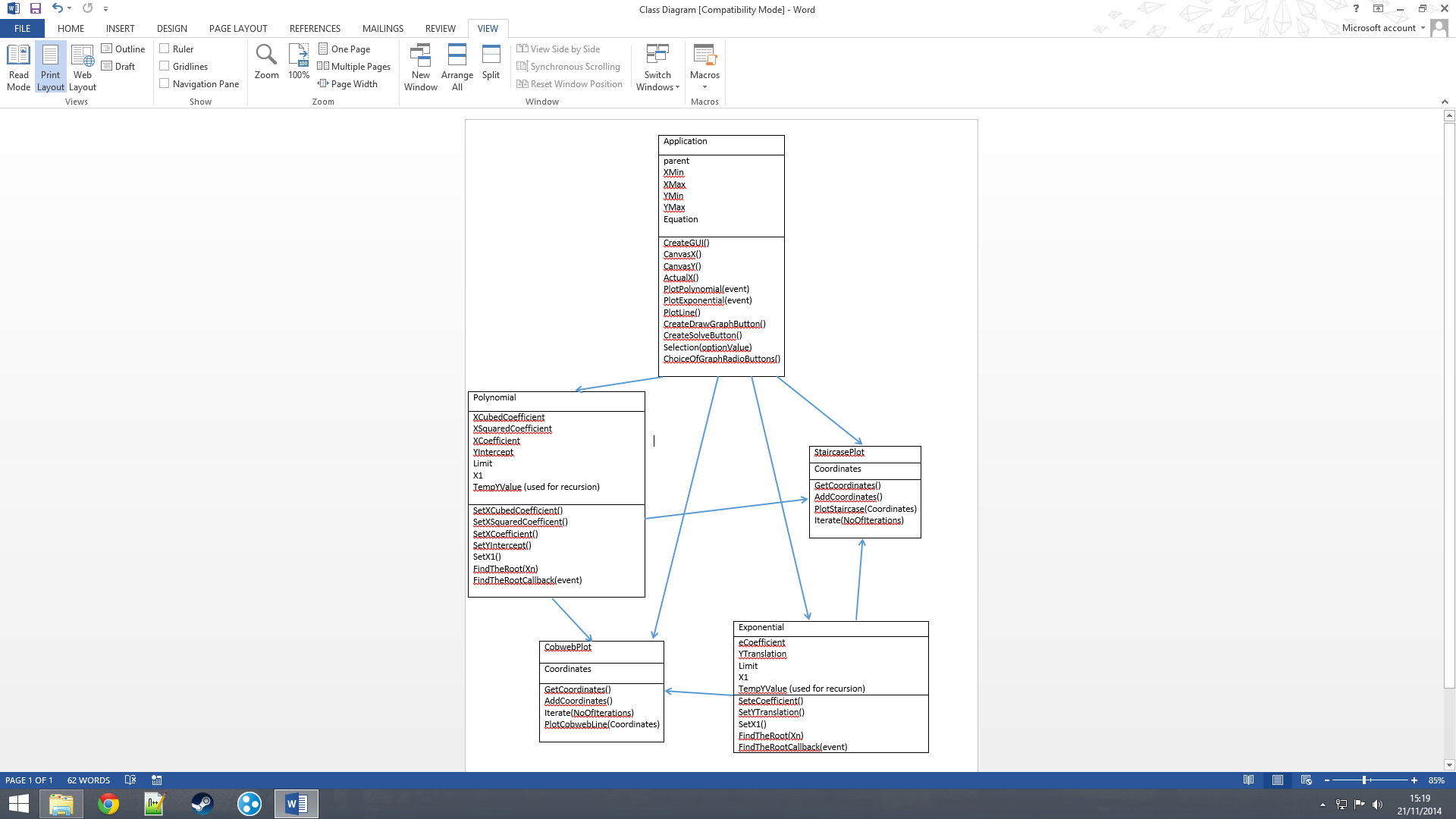
**Validation Required**

To ensure there are no program crashing errors I will need to use validation and verification for my data. I will use these mostly for the user input data so that the user cannot input extreme or incorrect values. This will prevent the user from giving the program problems (equations) it cannot solve. Since in algebra fractions often replace decimals I will use the evaluation function for the text inputs. This will allow the user to easily input numbers such as 1/3, instead of having to type 0.33333.

**Identification of Storage Media**

The solution will be stored on the school’s network, in the “Shared Documents” location. The teachers will be told how to reach the file(s). The teachers will then be able to tell their students where the files are in lessons. A USB stick is recommended to students who want to use the program outside of school for revision. Since my system won’t have any need to store records it won’t need much storage space. However the coding will need a small amount of memory to store.

**Class Diagram**

This class diagram shows the classes in the system and where they interact.

**Class Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Length** | **Validation** | **Comments** |
| XMin | 5 | Integer | Stores smallest x value on the graph axes |
| XMax | 5 | Integer | Stores largest x value on the graph axes |
| YMin | 5 | Integer | Smallest value on y axis. |
| YMax | 5 | Integer | Largest value on y axis. |
| Equation | 10 | None | Equation becomes a class when radio buttons are selected by the user. Becomes Polynomial or Exponential. |
| Graph | 10 | List | This holds the coordinates of all of the points on the graph to draw them. |

**Application Class**

**Exponential**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Length** | **Validation** | **Comments** |
| eCoefficient | 5 | Float | Stores the coefficient of ex. |
| YTranslation | 5 | Float | Stores the value for the graph’s translation in the y-direction |
| Limit | 10 | Float | Stores the root (the limit) of the equation. |
| TempYValue | 10 | Float | Stores a temporary value for the result of the ExponentialRoot method. |
| DiagramCoordinates | Varying | List | Stores the x and y results from the ExponentialRoot method. Used to plot the diagrams. |
|  |  |  |  |

**Polynomial**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Length** | **Validation** | **Comments** |
| XCubedCoefficient | 5 | Float | Stores the coefficient of x3. |
| XSquaredCoefficient | 5 | Float | Stores the coefficient of x2. |
| XCoefficient | 5 | Float | Stores the coefficient of x. |
| YIntercept | 5 | Float | Stores the value for the y-intercept. |
| Limit | 10 | Float | Stores the value of the root of the equation (its limit). This value is calculated by one of the class methods. |
| X1 | 5 | Float | The first value of x used to solve the limit. |
| TempYValue | 10 | Float | Temporary value for the result of one of the methods. |
| DiagramCoordinates | Varying | List | Stores the x and y value results from the PolynomialRoot method. Used when plotting the staircase/cobweb diagrams. |

**Class Methods**

**Polynomial**

|  |  |  |
| --- | --- | --- |
| XCubedInput | self, Frame | Creates a text entry box for the user to put the x3 coefficient. |
| XSquaredInput | self, Frame | Text entry box for x2 coefficient. |
| XInput | self, Frame | Text entry box for coefficient of x. |
| YInterceptInput | self, Frame | Text entry box for the y-intercept value. |
| GetPolyInputs | self | Assigns the values from the entry boxes to their respective attributes. |
| PolynomialRoot | self, Xn | Uses recursion to find the root of the input equation. |
| FindTheRootCallback | self | Event that calls the PolynomialRoot method. |

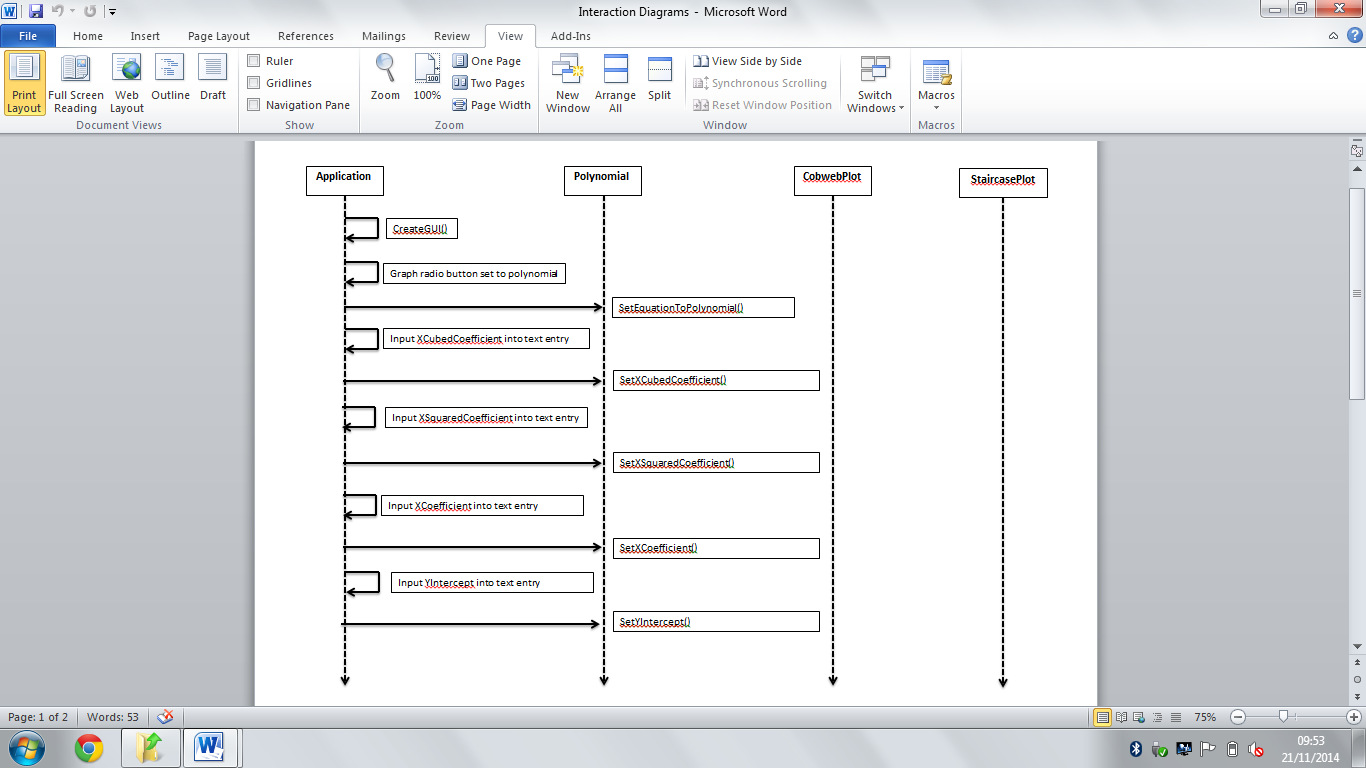
**Exponential**

|  |  |  |
| --- | --- | --- |
| eCoefficientInput | self, Frame | Creates an entry box for the coefficient for e. |
| YTranslationInput | self, Frame | Creates an entry box for the value the graph is translated by along the y axis. |
| GetExpoInputs | self | Assigns the inputs to their associated attributes. |
| ExponentialRoot | self, Xn | Finds the root for an exponential graph using recursion. |

|  |  |  |
| --- | --- | --- |
| **Method** | **Parameters** | **Comments** |
| CreateGUI | self | Calls all of the other methods that create buttons, text, etc. for the GUI. Also creates the axes for the graph. |
| CanvasX | self, x |  |
| CanvasY | self, y |  |
| ActualX | self, x |  |
| PlotPolynomial | self | Plots the points for a polynomial graph. |
| PlotExponential | self | Plots the points for an exponential graph. |
| PlotGraph | self | An event that calls either PlotPolynomial or PlotExponential depending on what object Equation is. |
| PlotLine | self | Plots the y = x line. |
| SolveLimit | self | Calls the SolveRoot method from the equation object based on which type of equation it is. |
| CreateSolveButton | self, Frame | Creates a button that solves the root of the given equation. |

**Application**

**Interaction Diagrams**

This interaction diagram shows how my classes will interact when the user chooses to draw a cobweb diagram that involves a polynomial equation. First the type of equation being used is set as a polynomial. Next the user inputs the required values for the coefficients. Then the program uses an algorithm (in the PolynomialRoot method) to find the limit. The algorithm stores its result from each iteration so the CobwebPlot class can access them and plot the diagram afterwards.

**Identification of suitable algorithms and pseudocode**

The program will need to be able to solve polynomial and exponential equations; I will need to use appropriate algorithms so it can find the roots for both equation types. Here are the algorithms I have made for this.

**PolynomialRoot(Xn)**

**Variables**

The customer told me that polynomial equations will not have powers that are above 3, meaning the program will need coefficients for x3, x2 and x, as well as a y-intercept. These are input by the user.

XCubedCoefficient : float

XSquaredCoefficient : float

XCoefficient : float

YIntercept : float

Xn : float

TempYValue : None

**START**

YValue **🡨** XCubedCoefficient \* Xn \*\*3 + XSquaredCoefficient \* Xn \*\* 2 + XCoefficient \* Xn + YIntercept

**IF** YValue (to 6 d.p) **EQUAL** TempYValue (to 6 d.p.) **THEN**

Questions usually ask for the answer to 4 significant figures or 4 decimal places (d.p.). I chose to use 6 d.p to avoid rounding errors without needing to make another method. Also because roots theoretically cannot be reached, it stops the program from looping infinitely/until it crashes.

Limit **🡨** YValue

**OUTPUT** Limit

**ELSE**

TempYValue **🡨** YValue

PolynomialRoot(TempYValue)

This line calls the function again, creating a recursive loop until the root (the limit) has been found.

**ENDIF**

**END**

**ExponentialRoot(Xn)**

**Variables**

An exponential equation is written in the form:

y = bx + c

Here, ‘b’ is usually given the value of ‘e’ (Euler’s number). ‘c’ is the translation in the y direction (i.e. how much the graph moves up or down).

eCoefficient : float

e : float

Xn : float

YTranslation : float

TempYValue : None

**START**

YValue **🡨** eCoefficient \* e \*\* Xn + YTranslation

**IF** TempYValue (to 6 d.p.) **EQUAL** YValue (to 6 d.p.)

Limit **🡨** YValue

**ELSE**

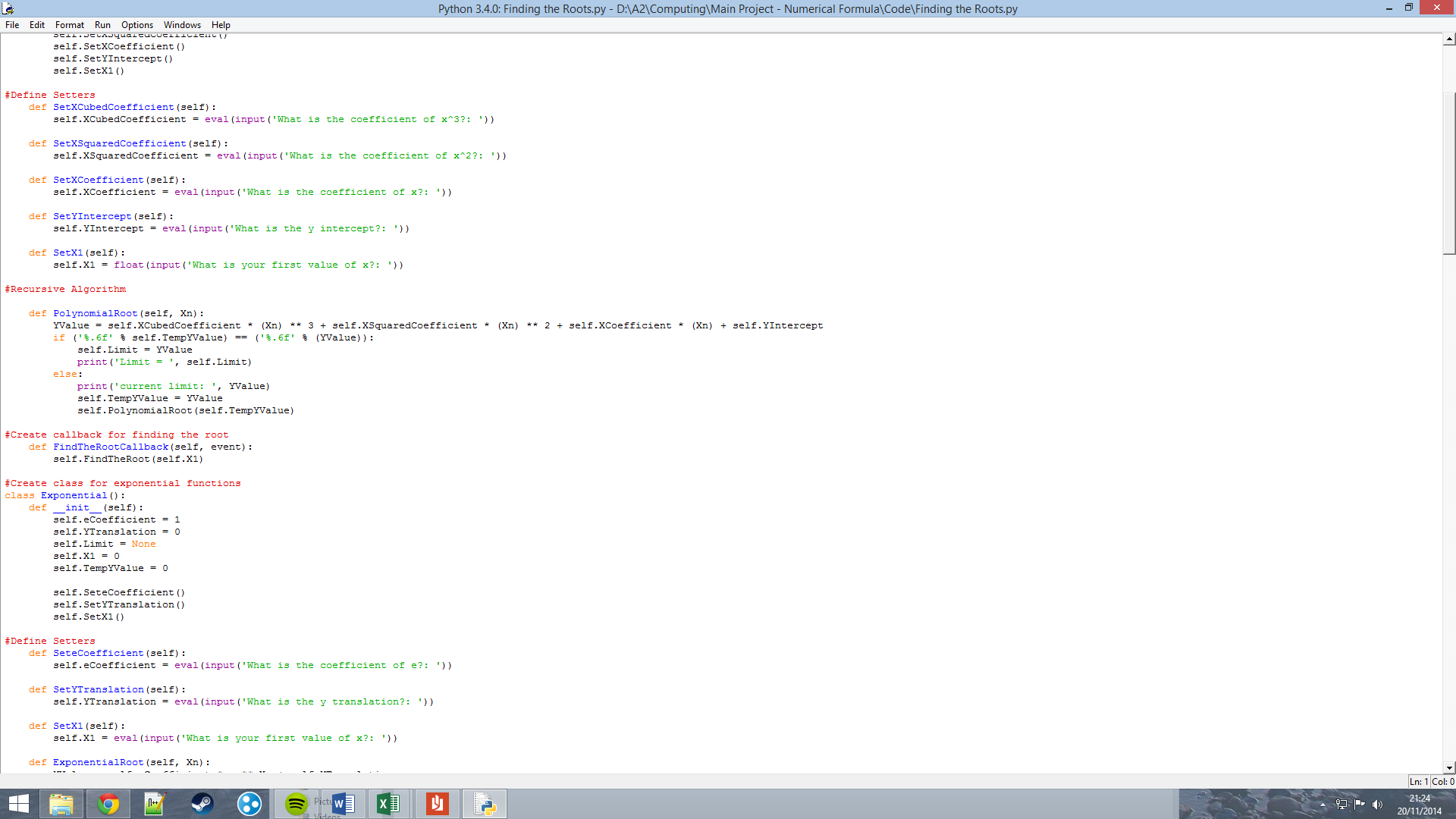
Like the PolynomialRoot method, the ExponentialRoot method calls itself recursively to find the root of the graph.

TempYValue **🡨** YValue

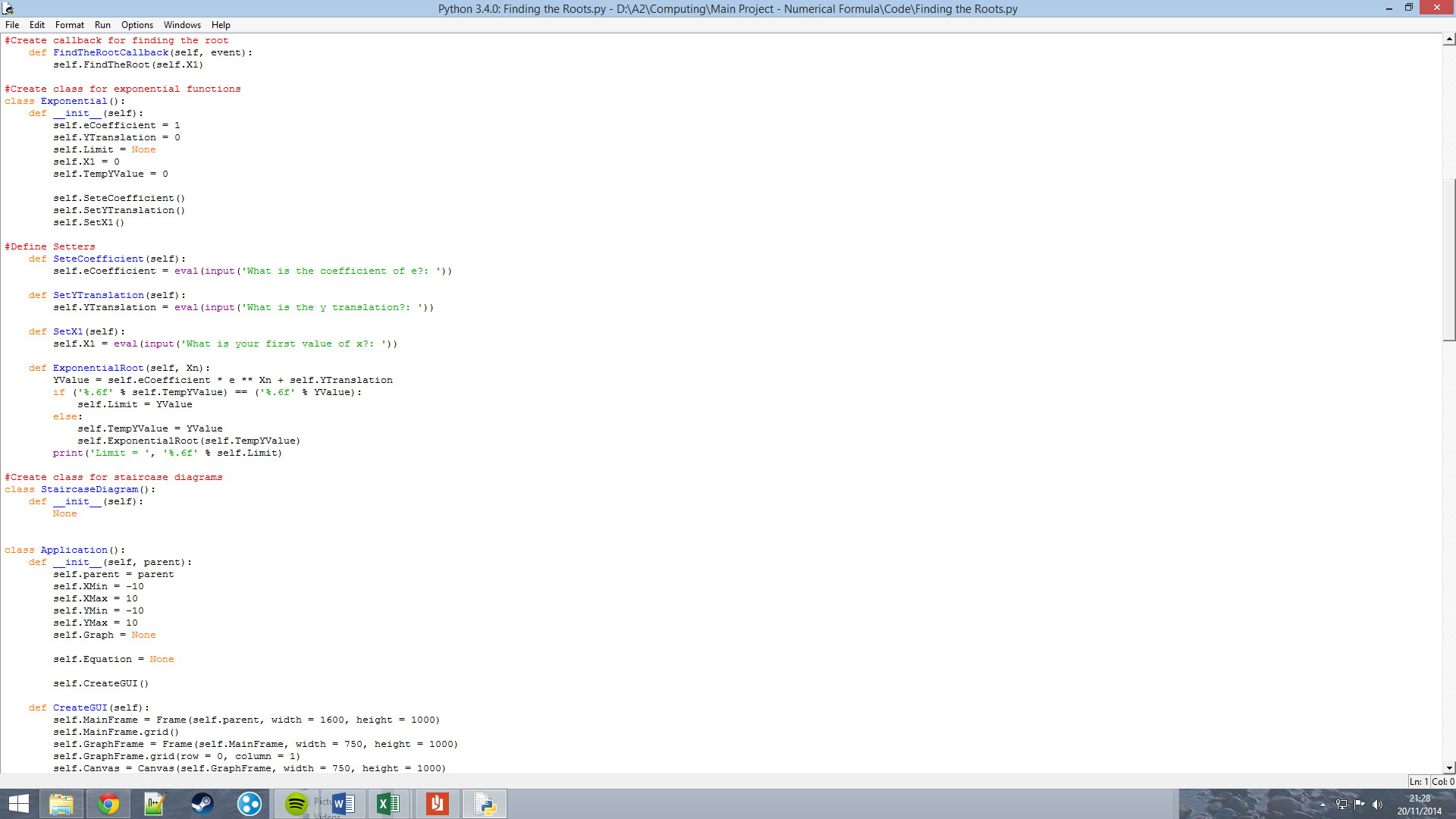
ExponentialRoot(TempYValue)

**Prototype Algorithms**

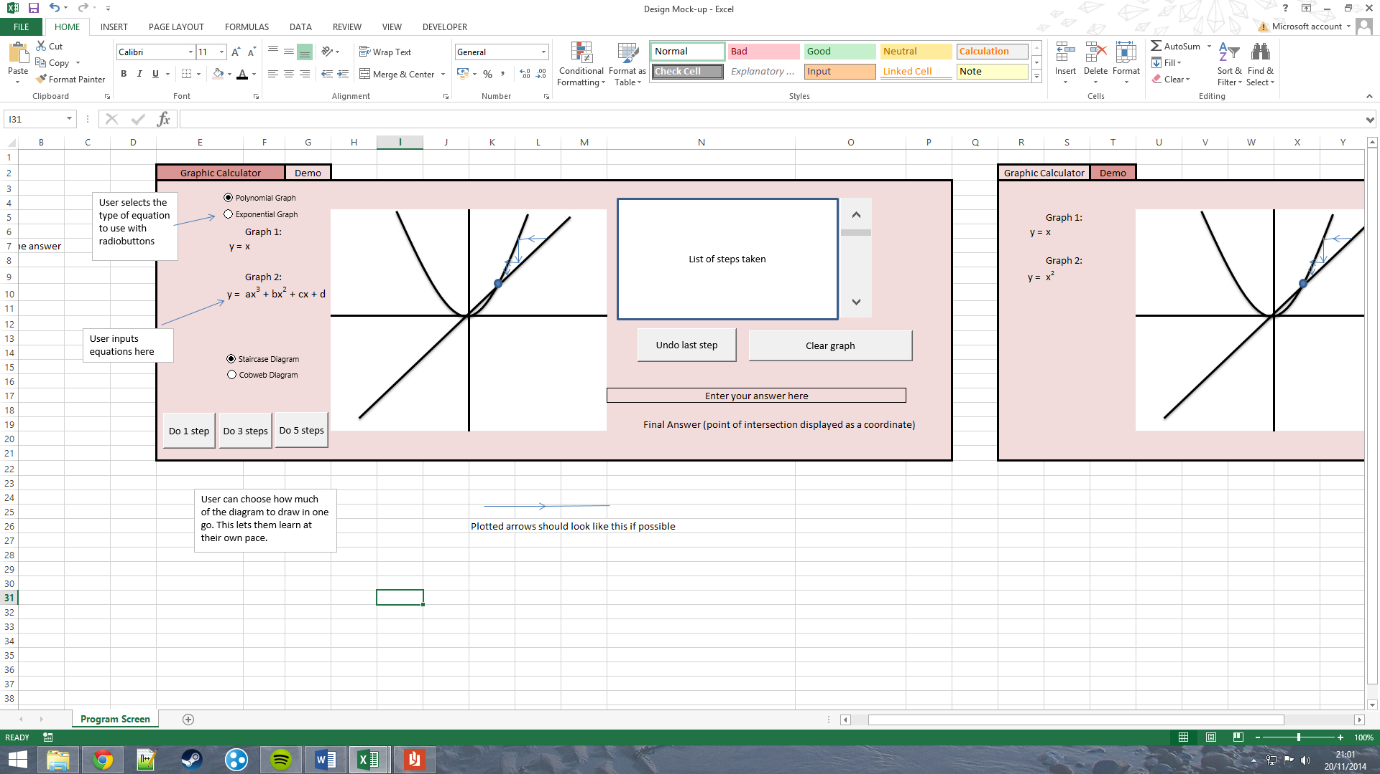
This section will detail the code I used to create the algorithms I previously specified.

PolynomialRoot

The print screen of this code includes print functions which were used to show me the algorithm worked when testing. I later removed it.

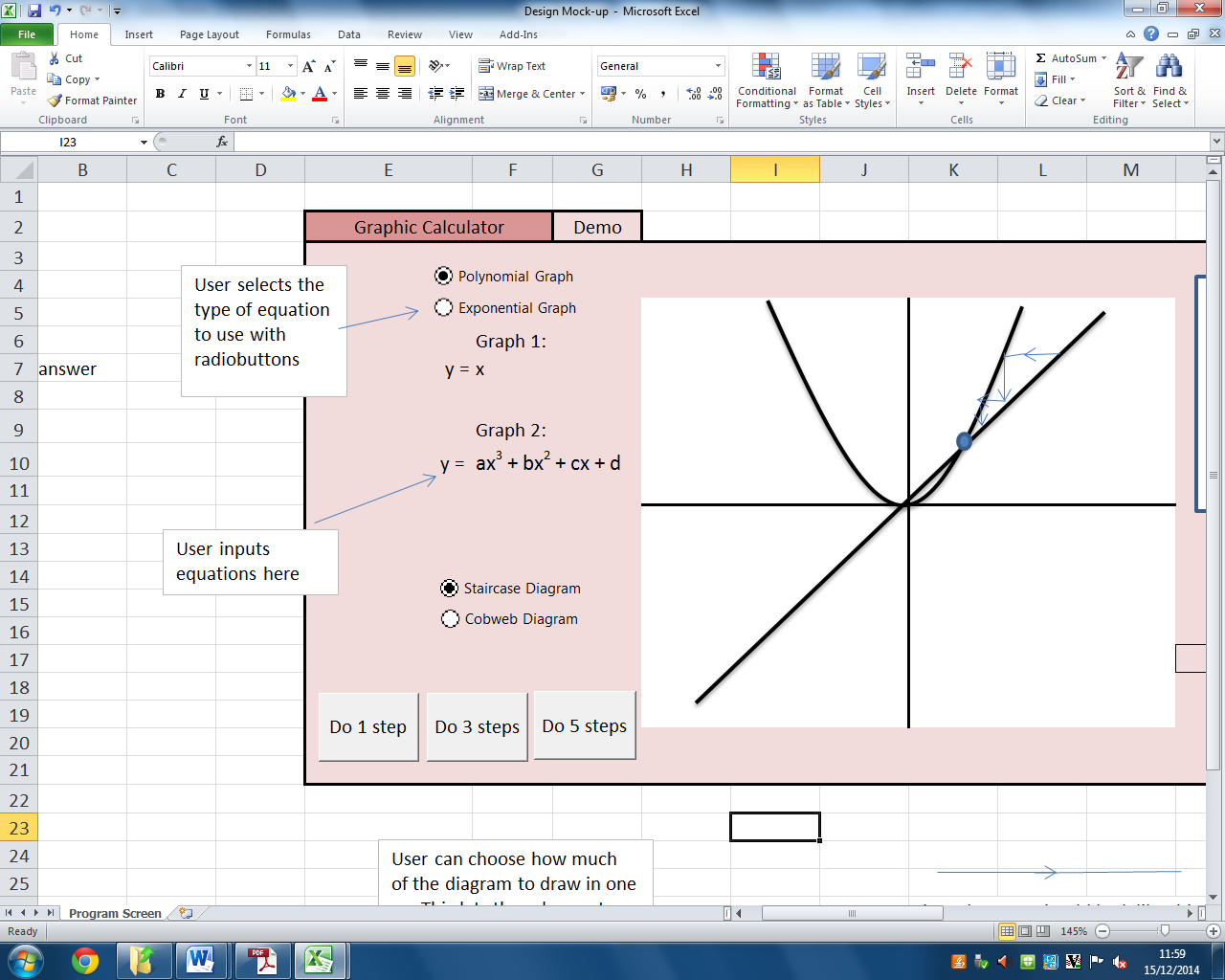
ExponentialRoot

This is the current code for the ExponentialRoot method (part of the Exponential class). The code itself is almost identical to the PolynomialRoot method’s code, apart from the calculations used to solve YValue.

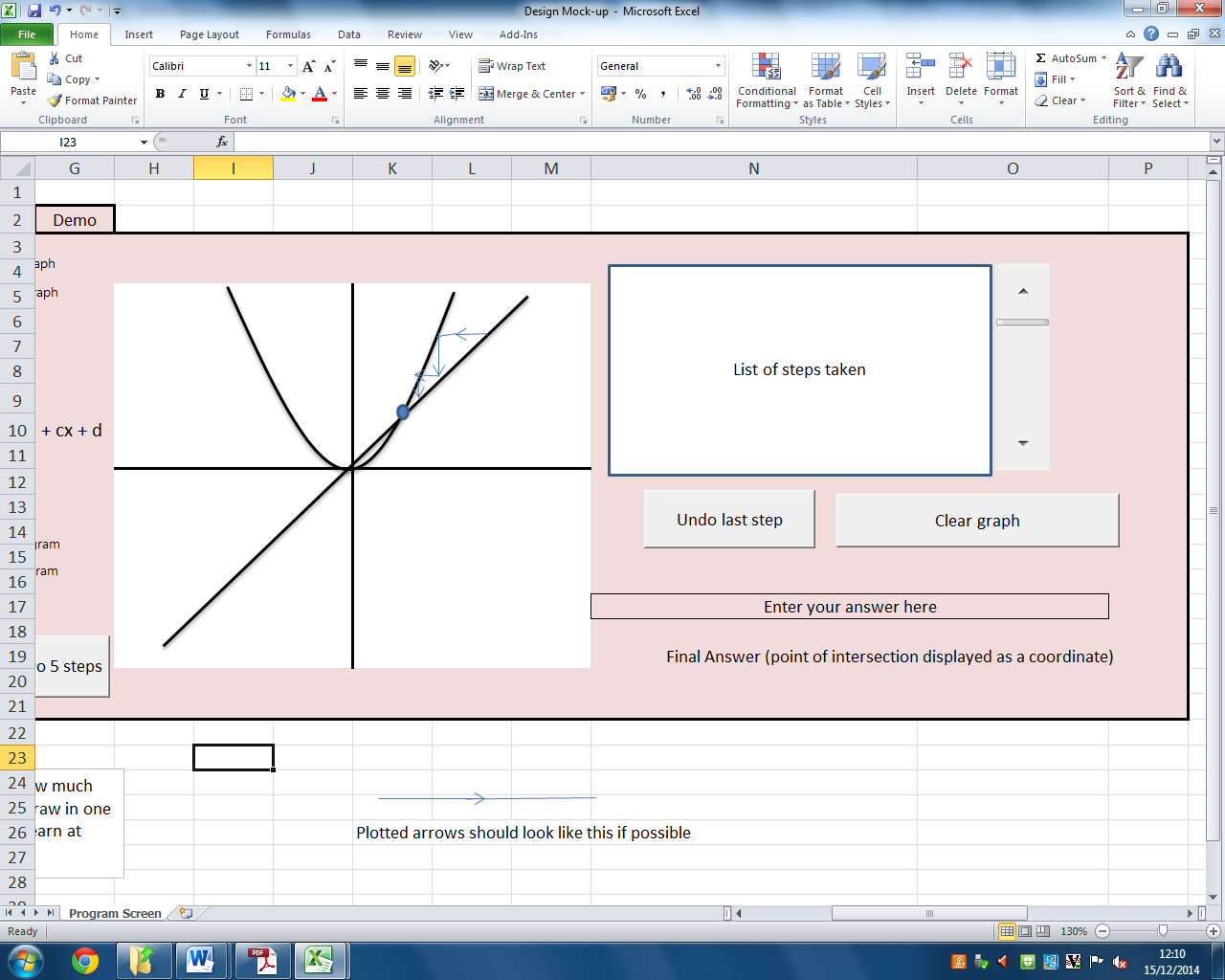
**GUI Designs**

This is a mock-up of what I imagine the GUI of the program will look like. There will be two tabs – one for the main program and one for a demo if the user needs help.

This is a close-up of the input side of the GUI. The user will be able to input parameters for each coefficient in the equation. These will be entered in text boxes; there will be a text box for each coefficient.



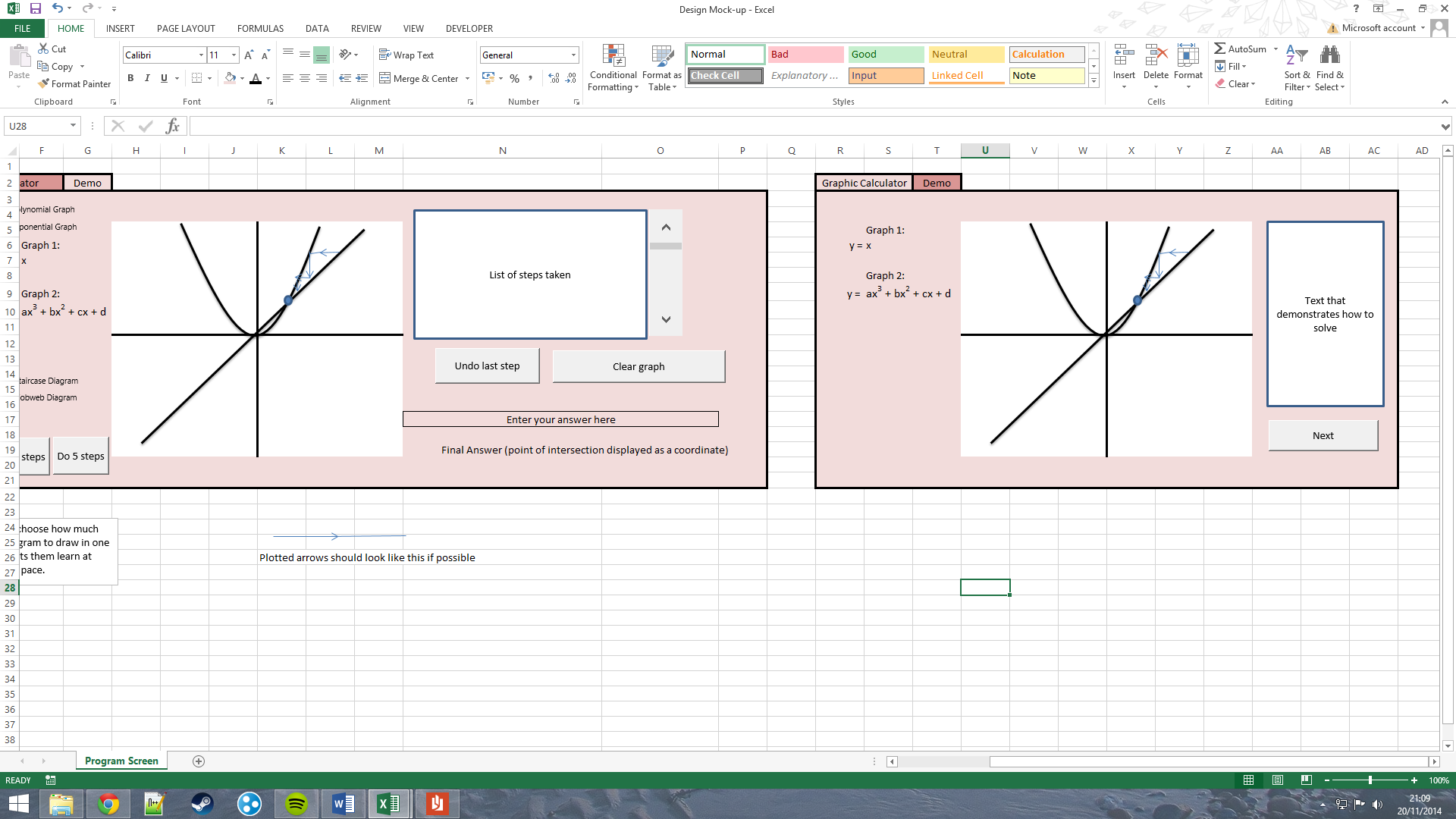
The customer’s request to be able to show steps of for method led me to design these “Do n steps” buttons. Each button will perform the specified number of steps for the user. This allows the user to set the pace for answering each question. I might also add a “Solve” button for users who want the answer straight away.



This is the output side of the GUI. The graphs and diagrams will be drawn onto the axes in the centre of the layout. In the top right is a text box that will display the calculations done for each step taken. The user will be able to scroll back and forth between each step so they can see the convergence patterns more easily.

The “Undo last step” button fulfils the objective for being able to work backwards and the “Clear Graph” button can be used to delete the diagram.

The user can enter their answer (assuming they have one) into a text box in the bottom right of the screen. They can then compare their answer to the actual answer. The customer asked for a simple feedback mechanism for this section when I showed it to him. This will probably be a small “Correct/Incorrect” message near their answer.



This is a draft of what the demo tab will look like when opened. I thought about making the user input the equation into the demo so the demo doesn’t become repetitive, however due to time constraints it is likely this won’t be implemented. For now I will use one pre-defined demonstration for a staircase diagram and one for a cobweb diagram.

**Overall Test Strategy**

To make sure the program works as intended, and to deal with possible errors, I will need to test many of the methods of the classes. I will use black box testing for this so that I can quickly identify errors and add error trapping where needed. My testing will need to make sure the following works:

* Graphs being plotted correctly
* Limits/roots being solved correctly
* The input boxes accepting the right data types
* The text box where the method is outputted prints the right text

**Planned Measures for System Security and Integrity of Data**

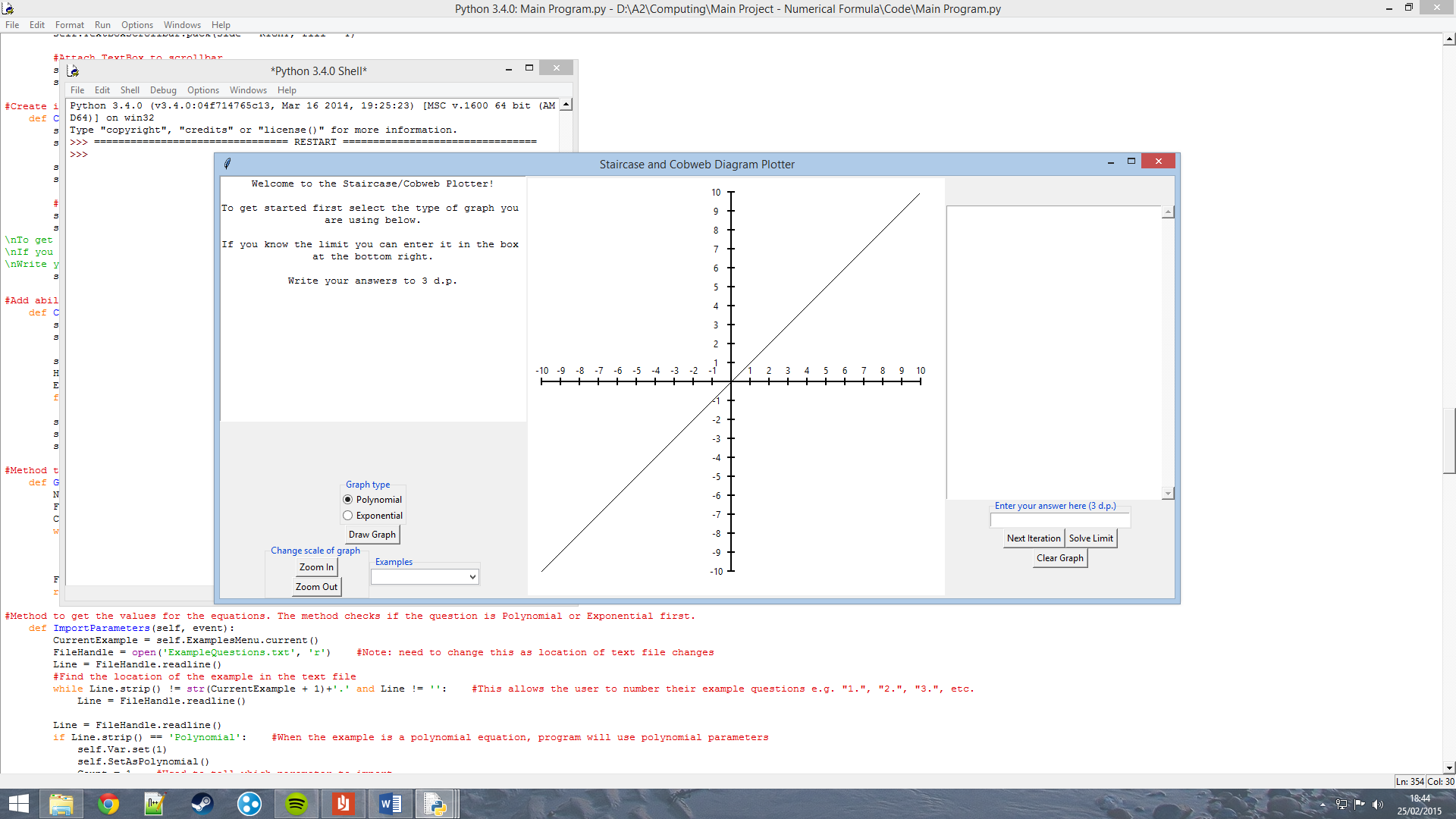
My system is to be used as a learning resource. It will be available to anyone who would want or need to use it (although it will still be targeted at my customers). The system will not store personal details about the user, therefore the amount of security the system will have will be minimal.

In the case that the program becomes corrupt I will remove it from the shared area on the school’s network and re-upload a version that is not corrupt.

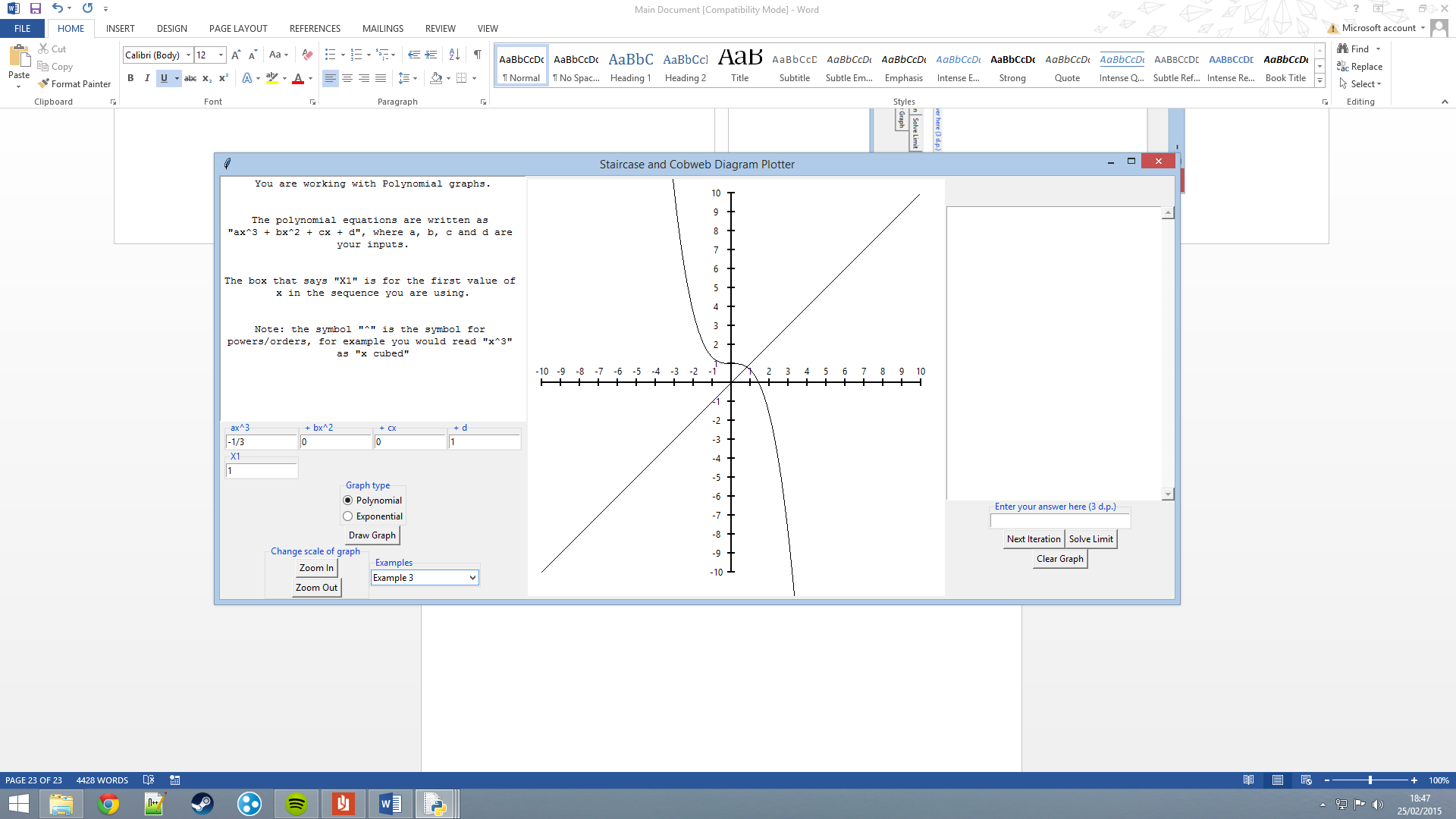
I will use data validation in my program to make sure the user’s inputs are of the correct data type. Fields that will use data validation include: the graph input boxes, inputs into these boxes will be floating point types (to allow the user to input fractions/decimals); the user’s answer, check as a floating point when checking against the program’s result; and the input for what the starting point of x will be (x1), which will also be checked as a floating point for the same reason as the graph equation inputs.

**Technical Solution**

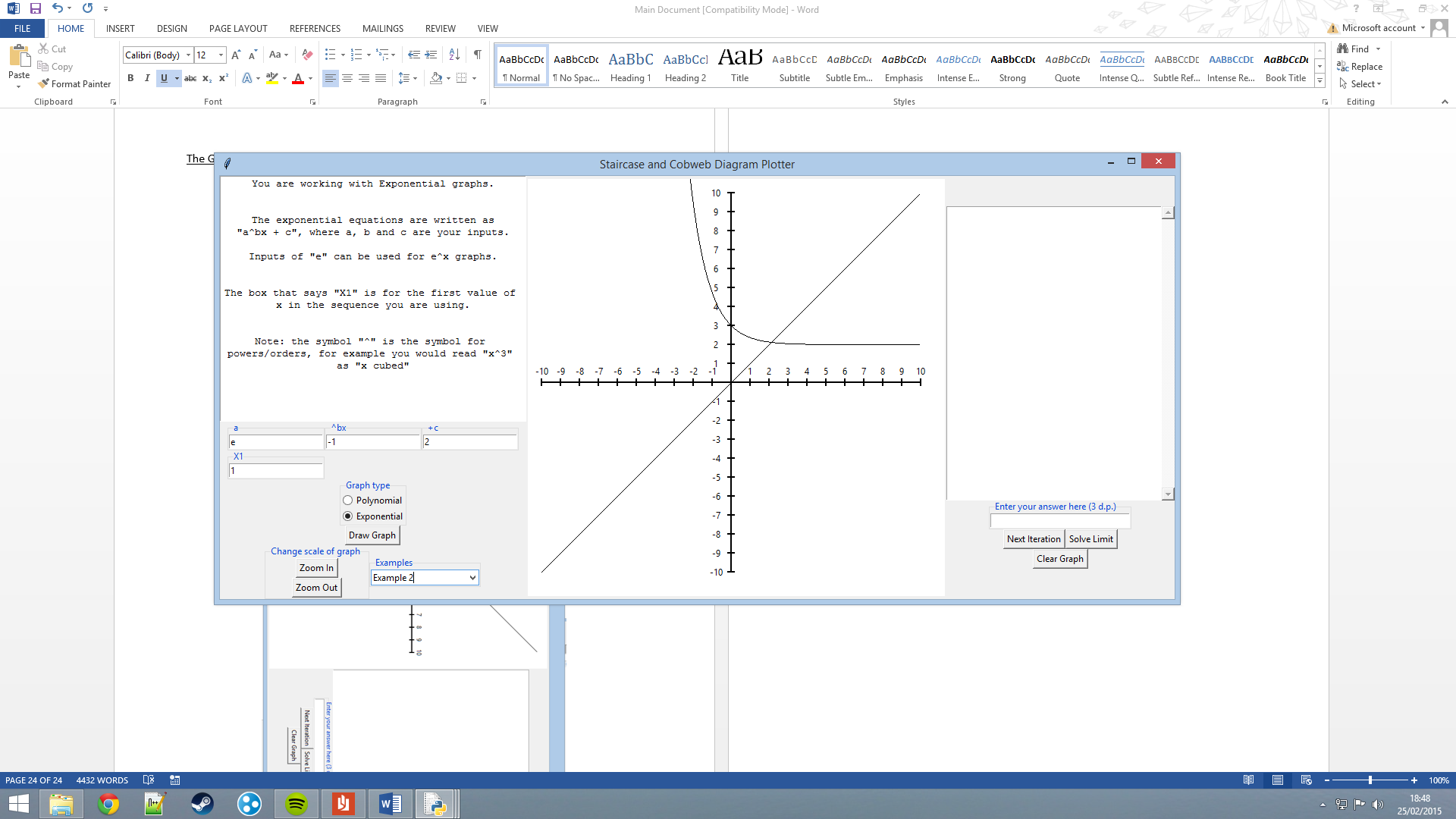
**The GUI (on start-up)**



**The GUI (Polynomial equations)**



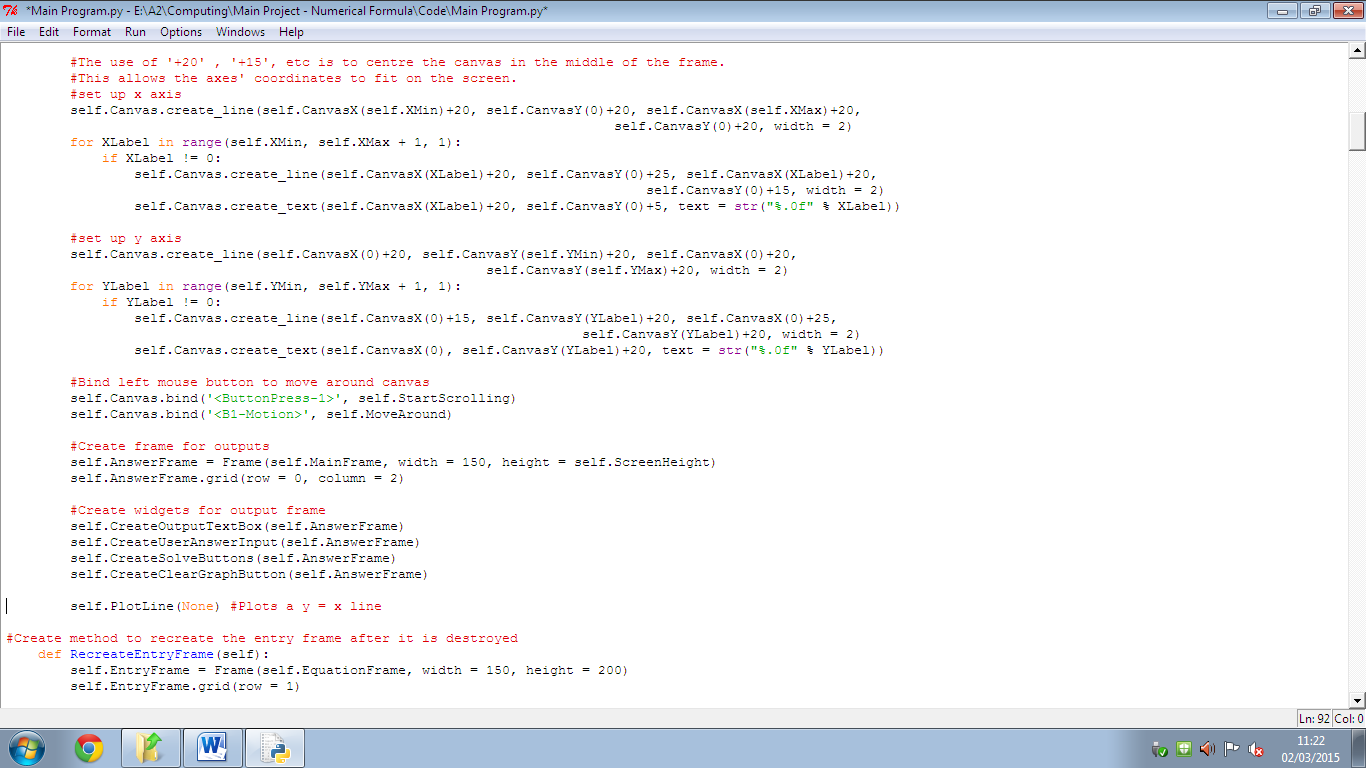
**The GUI (Exponential Equations)**

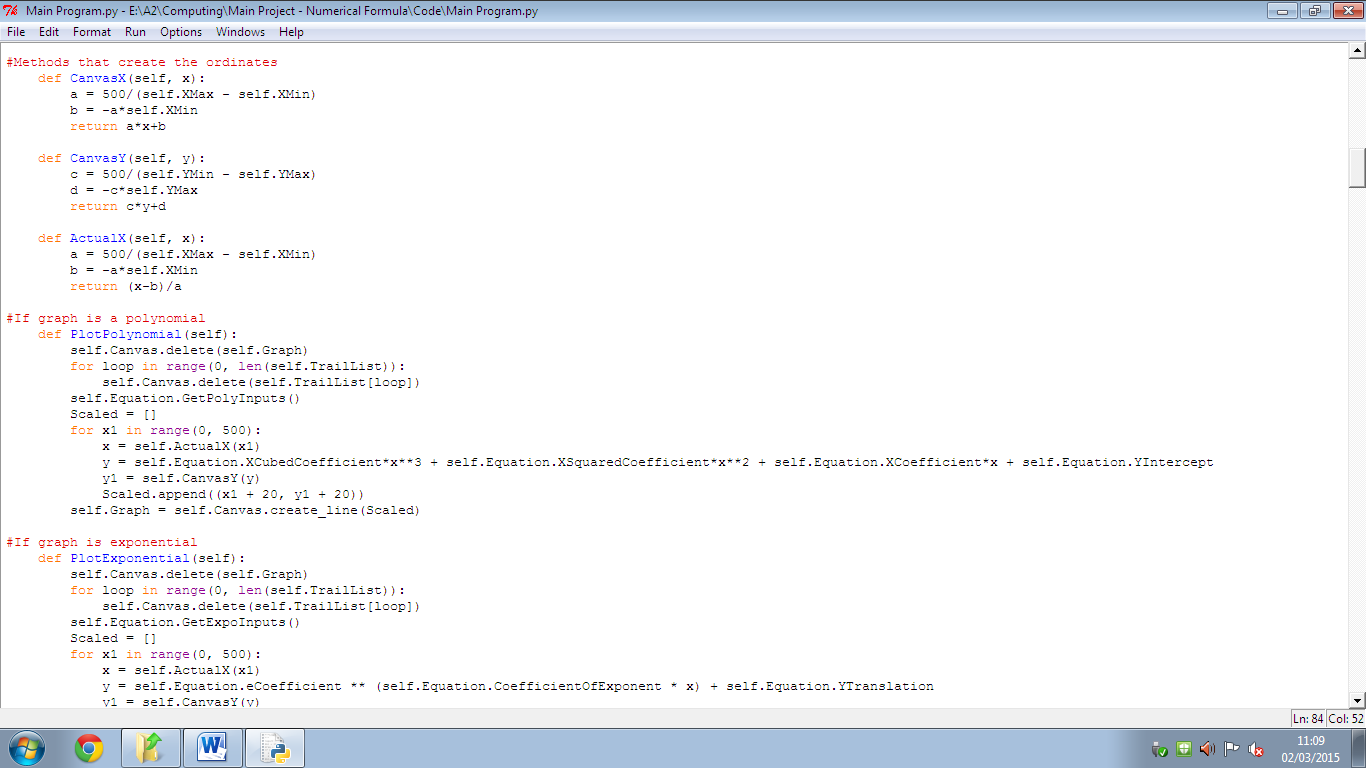


**Program Code**

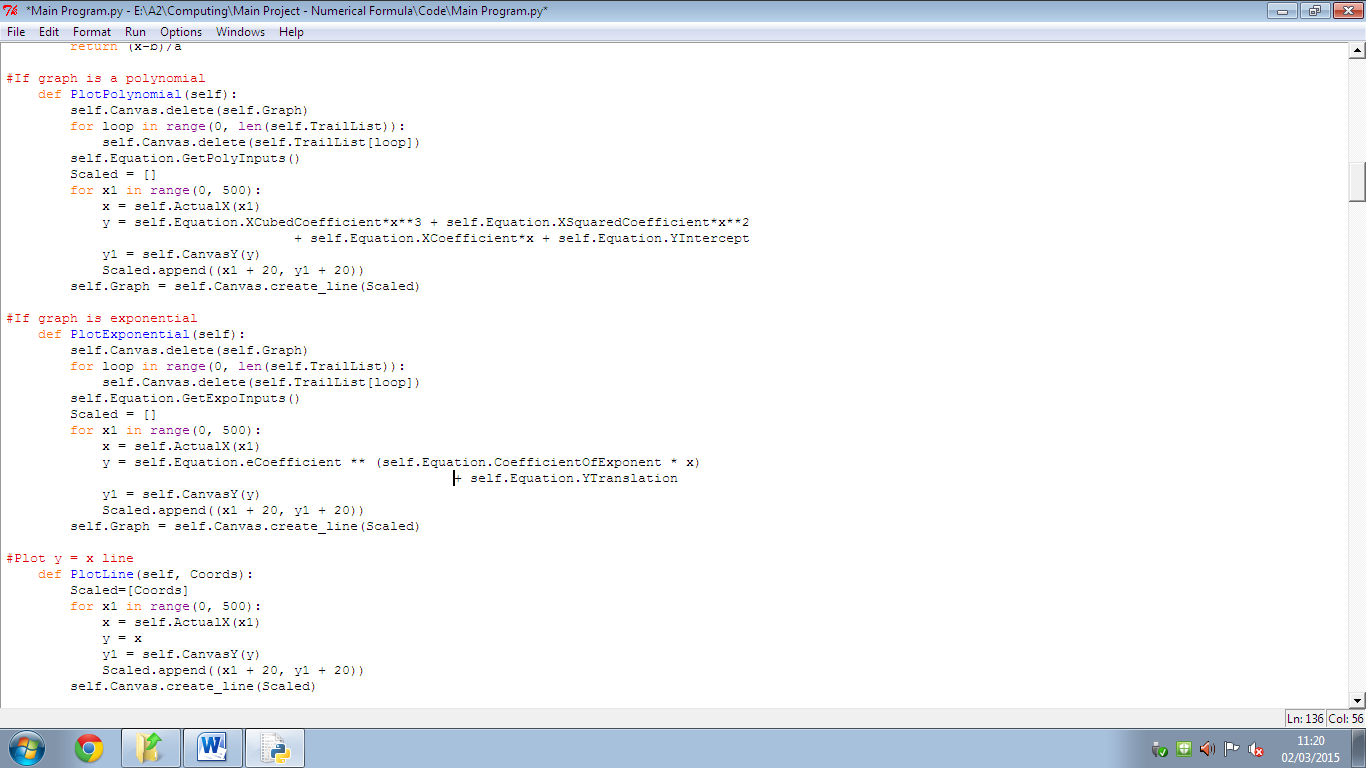
**Application Class**

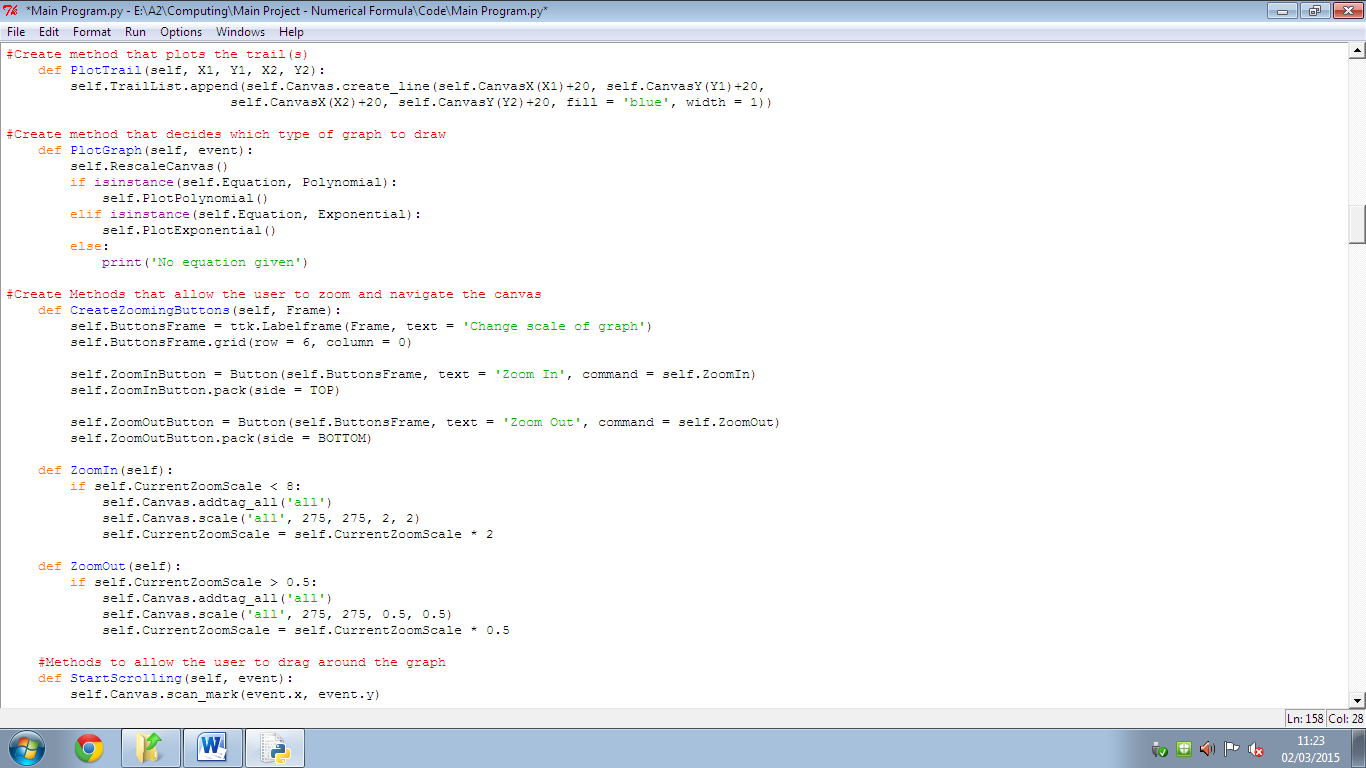
The application class is the system’s main class. The GUI is created and manipulated via methods within this class. The application class also stores the Equation object, which changes its own class during runtime to be what the user chooses it to be.

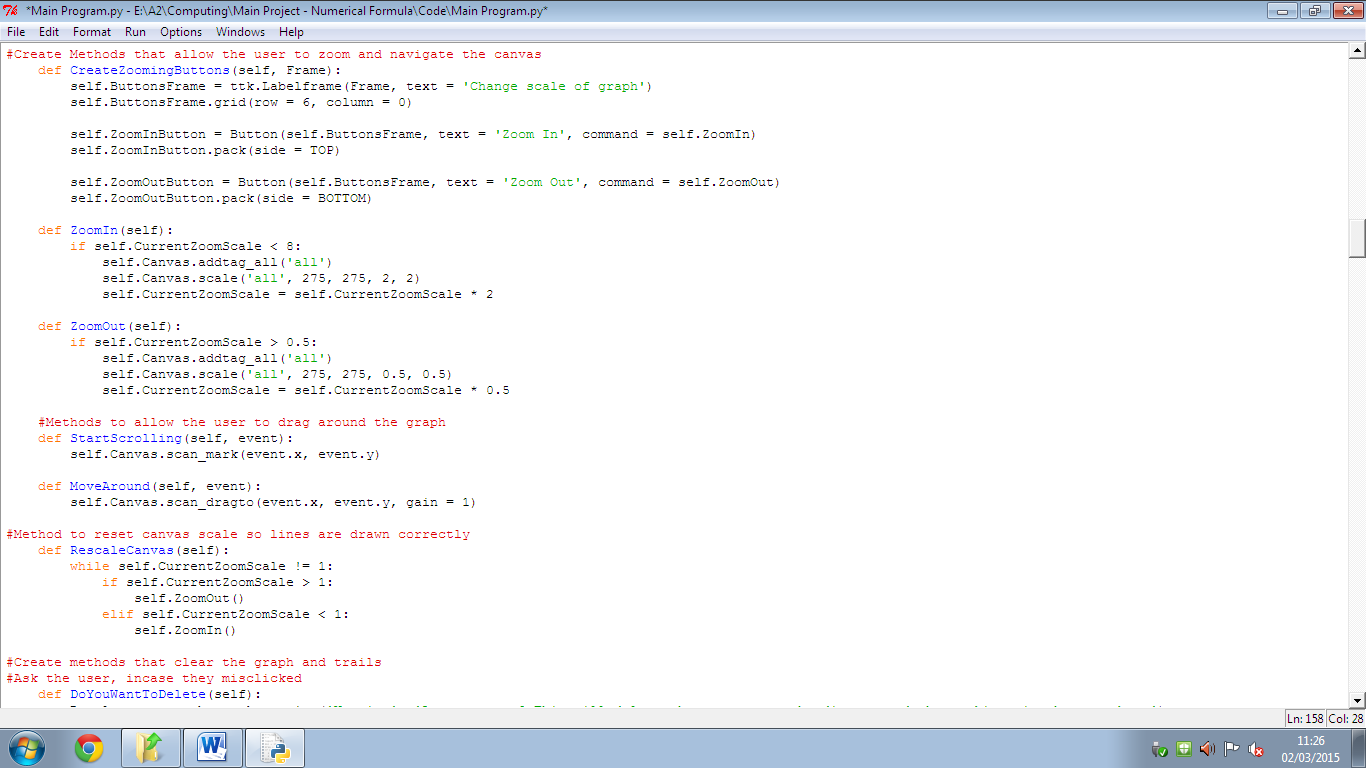
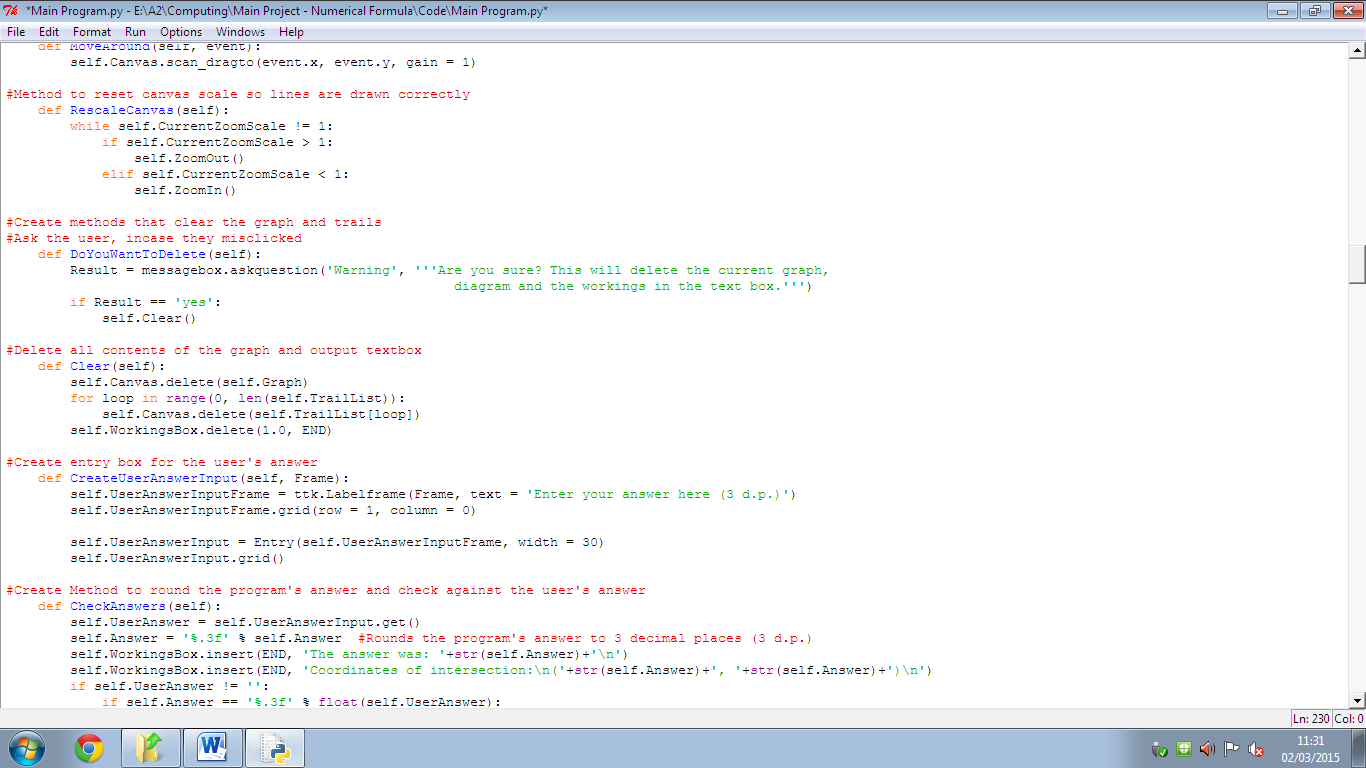
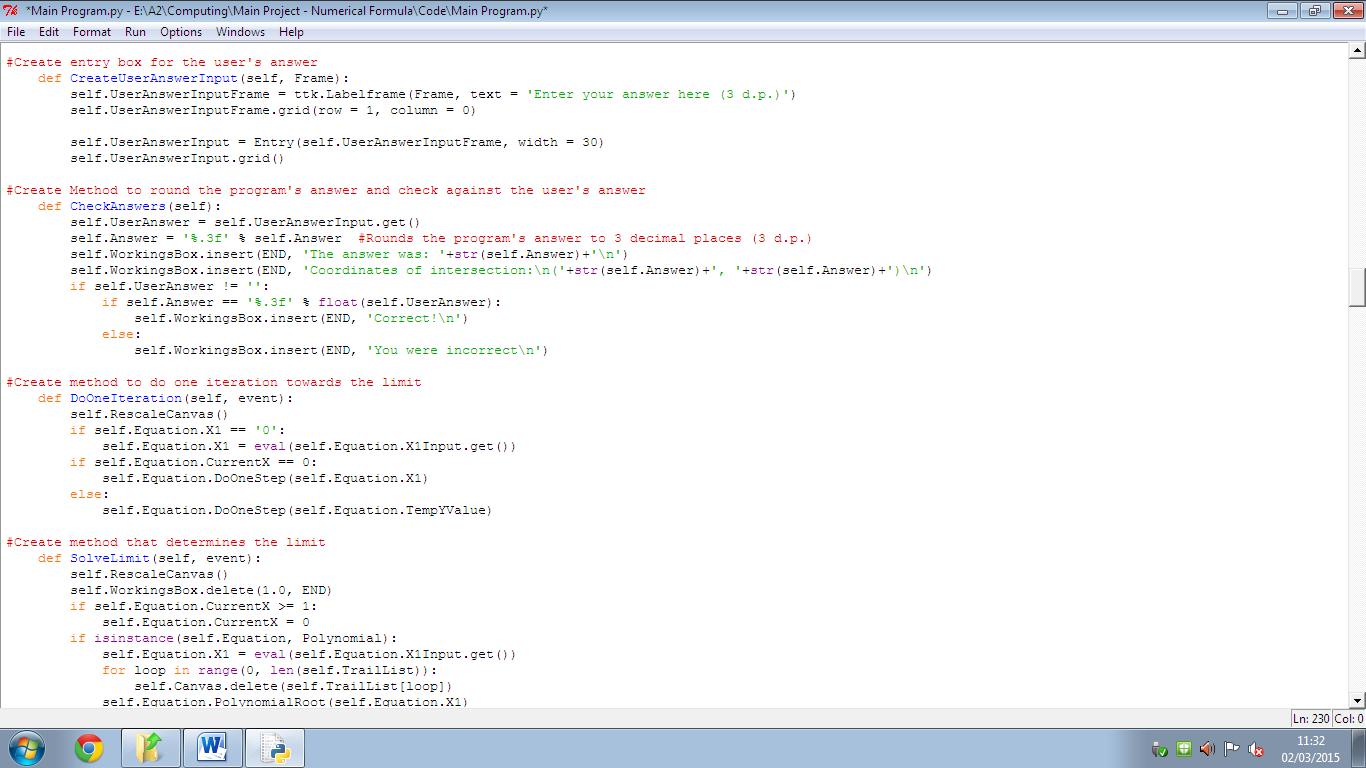


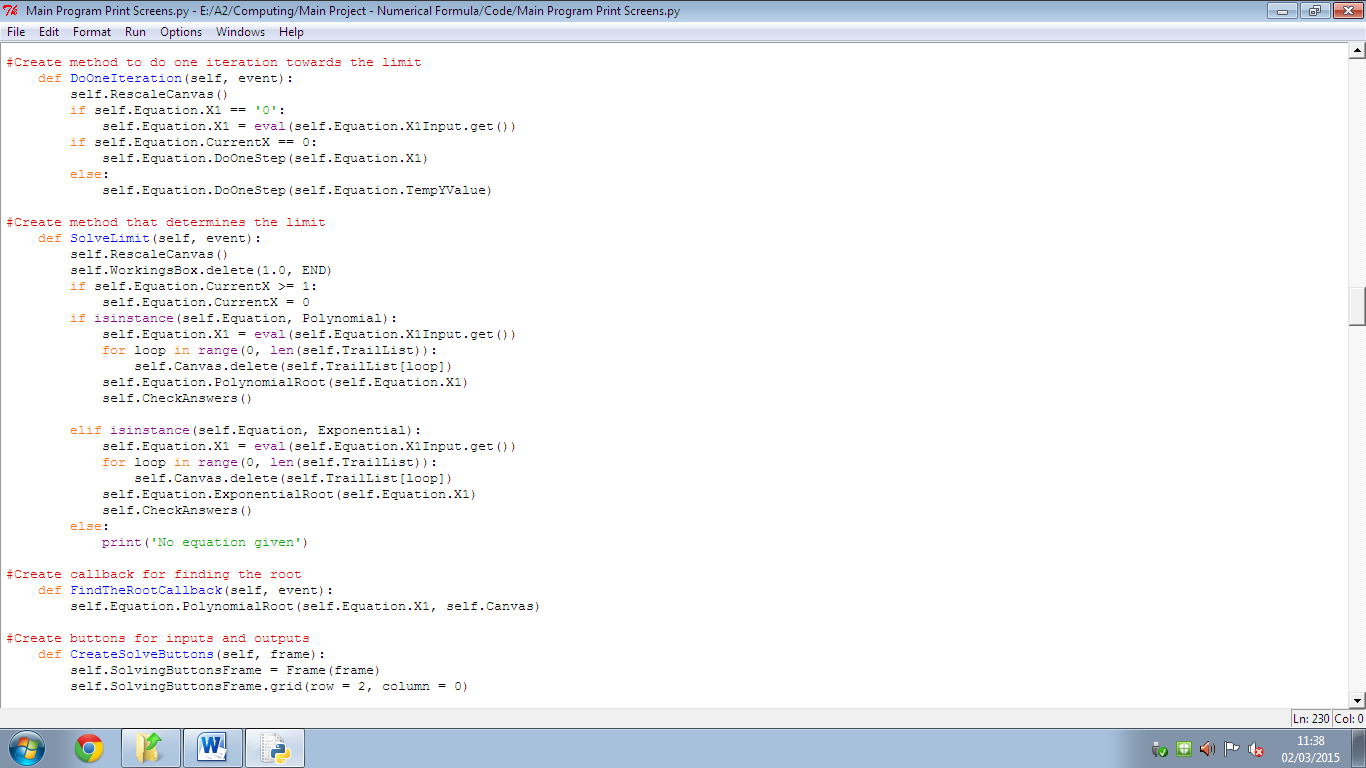


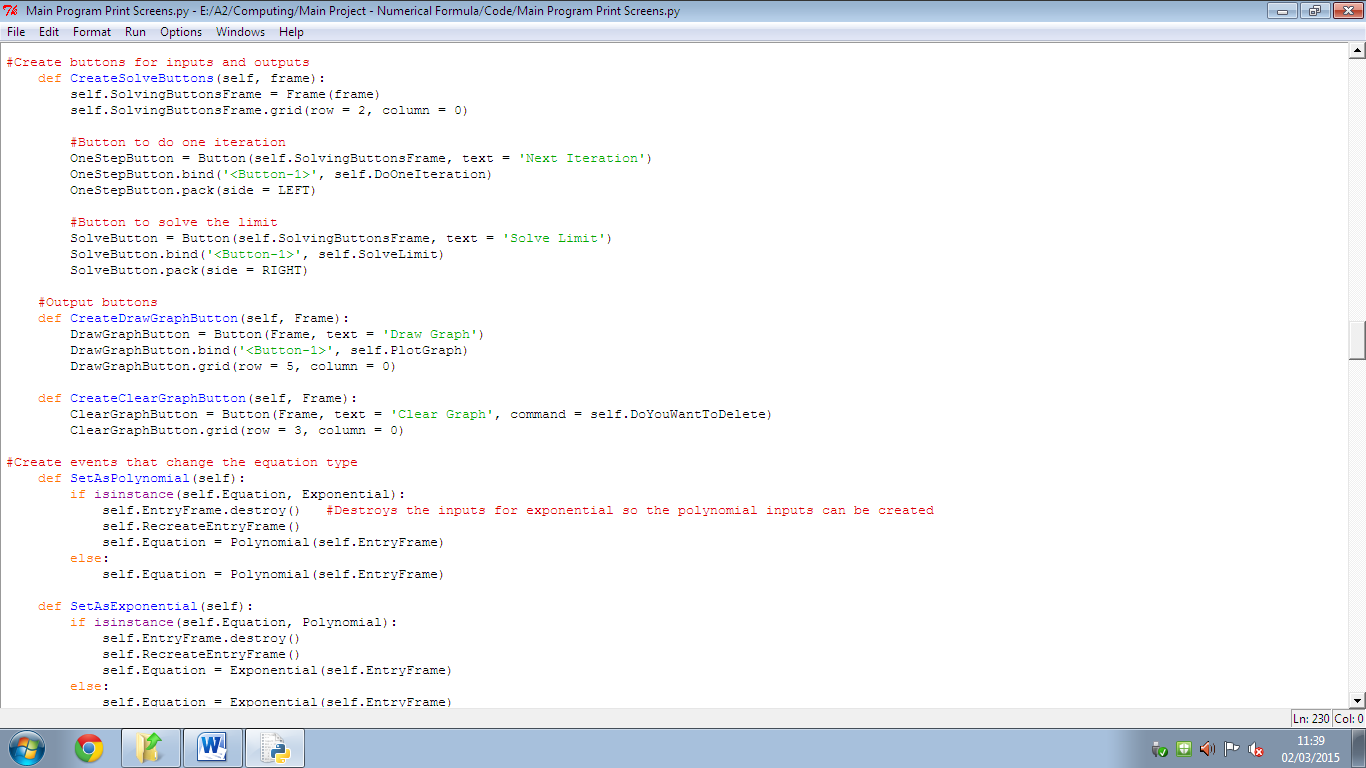
This section of code was taken from a tutorial file for drawing graphs in the school’s shared documents directory. In Python’s tkinter, the y axis is different to what is studied at Maths A-Level (as the y ordinate increases the object moves down as opposed to up). These methods change the coordinates Python uses into the standard coordinates.

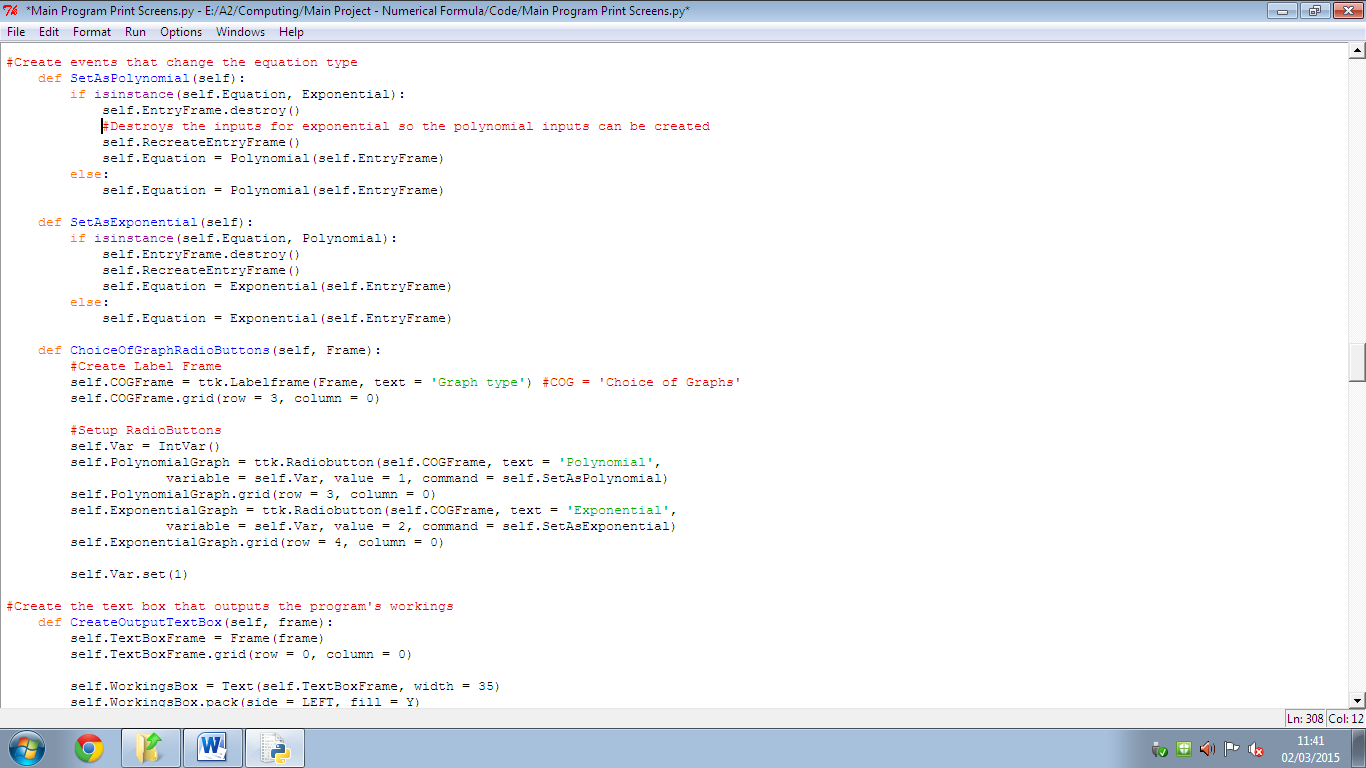


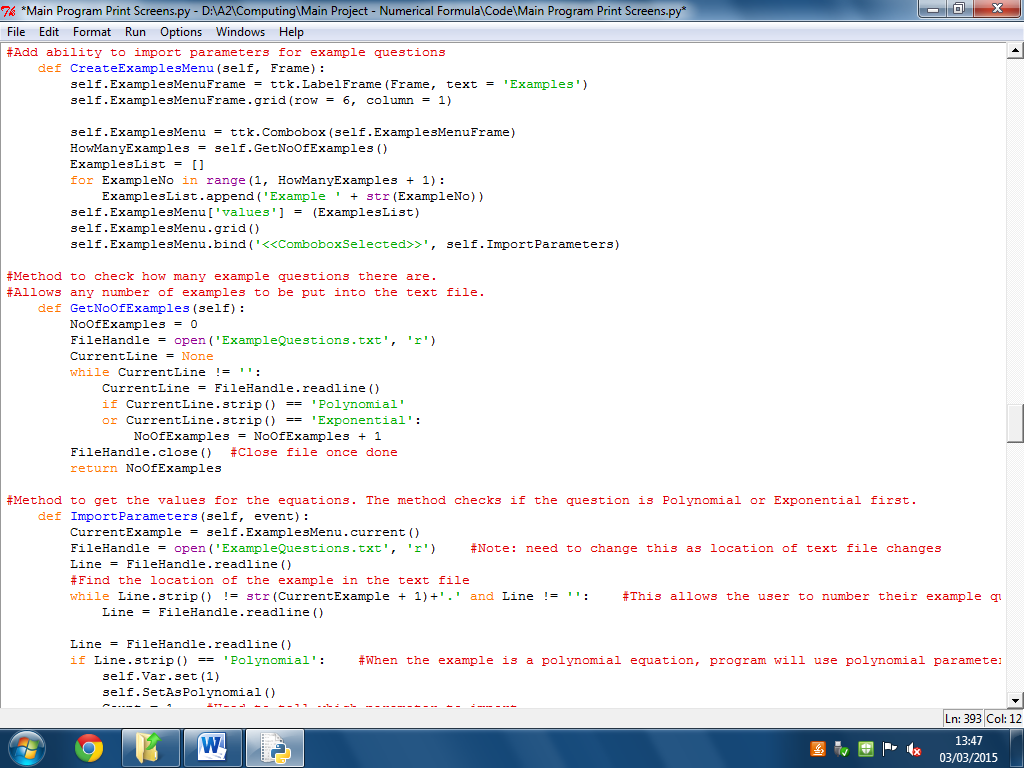


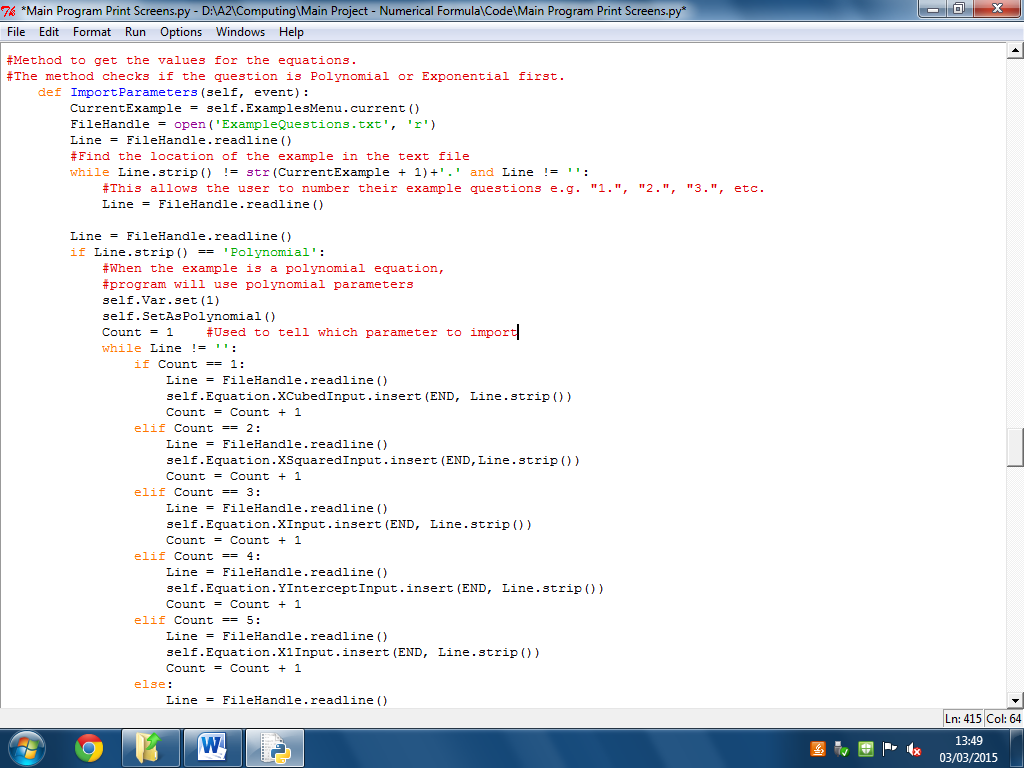
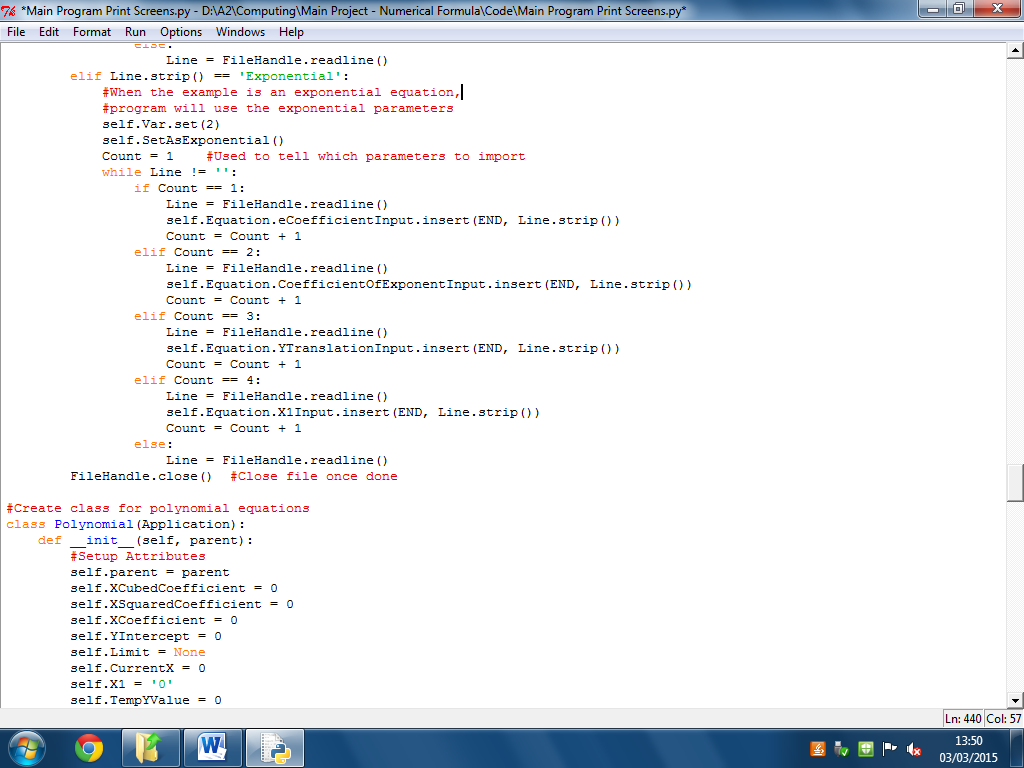




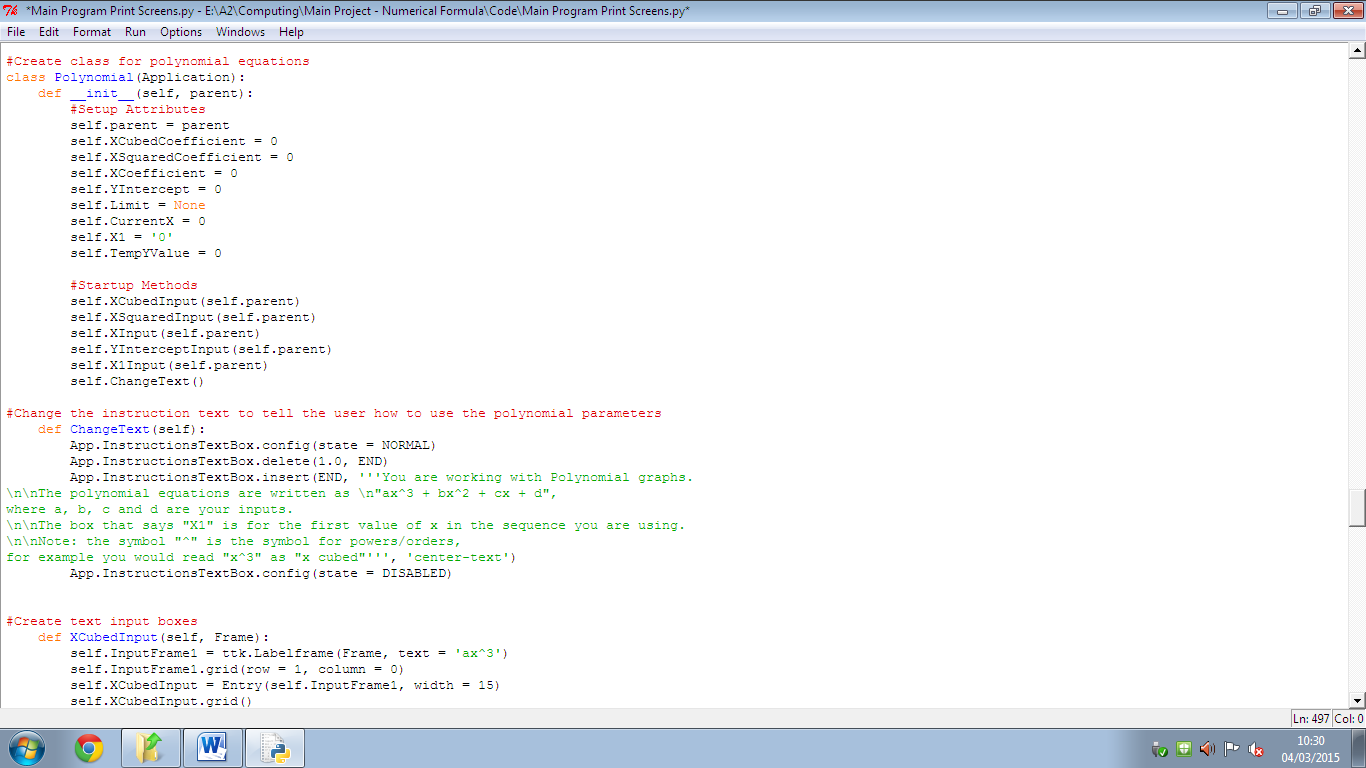


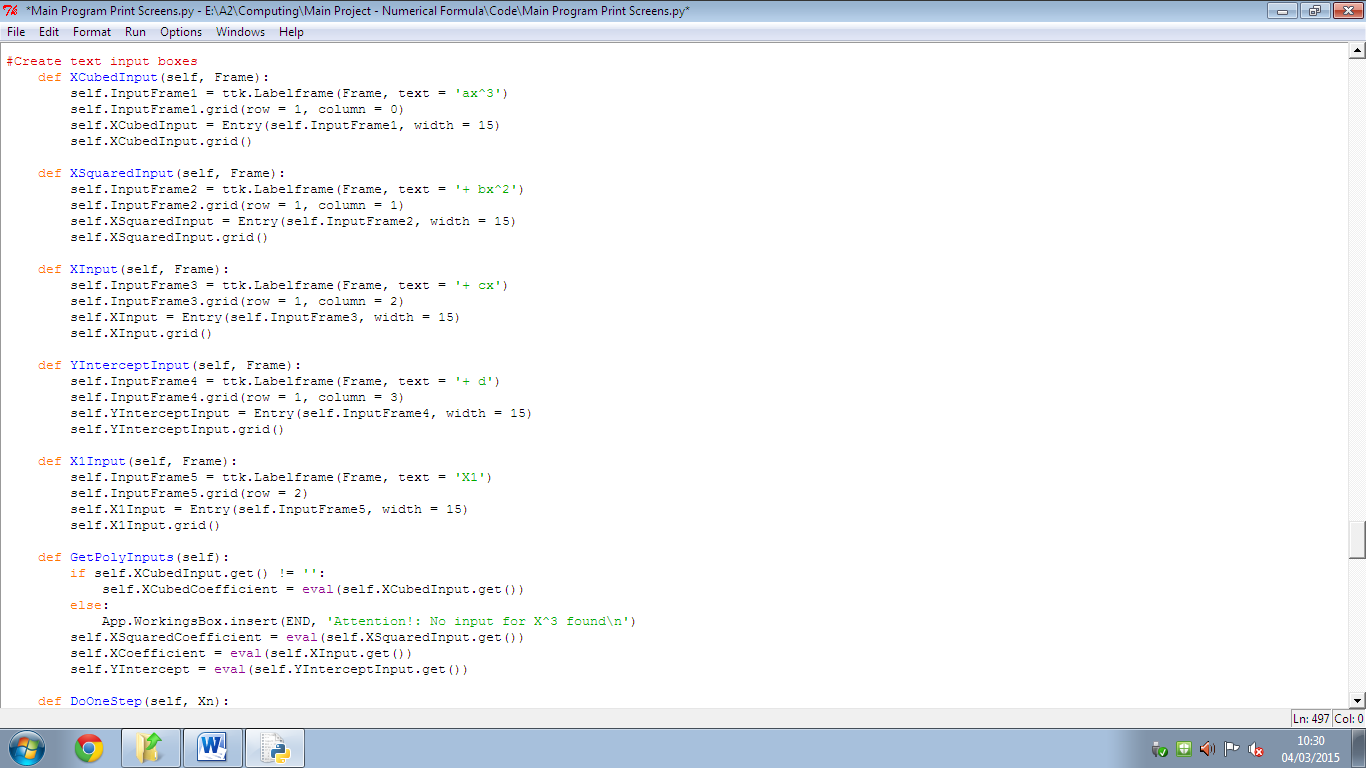






**Polynomial Class**

The object Equation (part of the Application class) becomes a polynomial when the user clicks the polynomial radio button in the GUI. This class has a set of attributes that would be seen from a polynomial equation, as well as methods that are used to find the limit when the equation is a polynomial.



**Testing**

**GUI/Display Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Description | Expected Result | Actual Result |
| **GUI** 1 | Click “Polynomial” radio button | Make sure the equation changes to the correct type when the radio buttons are selected by the user. | GUI displays inputs for polynomial equations | Polynomial inputs displayed |
| **GUI** 2 | Click “Exponential” radio button | Make sure the equation changes to the correct type when the radio buttons are selected by the user. | GUI displays inputs for exponential equations | Exponential inputs displayed |
| **GUI** 3 | Instructions Textbox (top left of GUI) on program start-up | Check that the text is correct for what the user has currently selected | Instructions about the system’s general use is displayed | Instructions are displayed |
| **GUI** 4 | Instructions Textbox (polynomial equations) | Check that the text is correct for what the user has currently selected | Guide for the polynomial inputs given to the user | Relevant guide is displayed |
| **GUI** 5 | Instructions Textbox (exponential equations) | Check that the text is correct for what the user has currently selected | Guide for the exponential inputs displayed | Relevant instructions shown to user |
| **GUI** 6 | Examples list | Check that the predefined examples in the Examples text file import correctly. | Each example loads in and works with the rest of the program | Each example is put into the list and displayed correctly when prompted |
| **GUI** 7 | Draw Graph button; input values for  y = 2x +2 graph | Make sure graphs are drawn correctly when the button is pressed | The appropriate graph is drawn when the inputs have been entered | Graph is drawn correctly |
| **GUI** 8 | Zoom In button | Make sure the canvas zooms in when button is pressed | Zoom of the current diagram is increased (by factor of 2) | The canvas zooms in as expected |
| **GUI** 9 | Zoom Out button | Make sure canvas is zoomed out when button is pressed | Zoom of the current diagram is decreased (by factor of 2) | The canvas zooms out as expected |
| **GUI** 10 | ZoomIn method | Check that user cannot zoom in indefinitely | Scale of the diagram is limited (pre-set max scale of x16) | The scale of the diagram is limited, further use of the zoom in button does not scale. |
| **GUI** 11 | ZoomOut method | Check that user cannot zoom out indefinitely | Scale of diagram is limited (pre-set min scale of x0.5) | The scale of the diagram is limited, cannot zoom out farther than the limit. |
| **GUI** 12 | Navigating the graph by dragging | Check that the user can look at different parts of the graph | Scrolling the canvas can be done by clicking and dragging the mouse | Canvas is moved when I click and drag. |
| **GUI** 13 | Scroll boundaries | Make sure the user doesn’t lose the graph by scrolling too far | Scrolling is limited, user does not lose sight of the graph | Scrolling is NOT limited to the user. Place scroll region boundaries to fix. |
| **GUI** 14 | Draw Graph method while zoomed | Lines are plotted incorrectly when the graph is scaled, make sure graph resizes to scale/zoom of x1 before plotting | Graph rescales correctly and lines plotted correctly | Lines are plotted correctly; however the view is not centred. Fix with a method to centre the canvas. |
| **GUI** 15 | Solve Limit button | Check Solve Limit button calls the method to solve the limit of the given equation/sequence | Solve Limit method called when button is clicked | Limit of the equation is solved and displayed to the user |
| **GUI** 16 | Outputted limit from given equation | Do each example manually then compare with the program’s answers | Manually worked out answers are the same as the program’s outputs | My manual answer was the same as each of the examples’ answers |
| **GUI** 17 | Next iteration button | Check only one step is shown to the user; compare with own results for accuracy. | A single iteration is performed and displayed. The output is correct. | One iteration performed, each iteration displayed correct results (based on manual answers) |
| **GUI** 18 | Repeated use of next iteration button (testing the DoOneStep method) | Check to see what happens when the Next Iteration button is clicked after the limit has been solved | Output telling the user that they have already solved the limit. | Output telling the user that the limit has been reached is displayed. No further iterations are performed. |
| **GUI** 19 | Clear Graph button is pressed | Check that the program does not delete everything straight away | Program asks the user if they are sure they want to clear | User is prompted with the yes/no question. |
| **GUI** 20 | Clear graph question answered “Yes” | Make sure the program deletes the correct data. | Program deletes the plotted diagram (and trails if any), as well as the contents of the output textbox | Trails, graph and contents of workings textbox are deleted. |
| **GUI** 21 | Clear Graph question answered “No” | Make sure the program doesn’t clear the graph if the user doesn’t want it to | Program does nothing to its current data | No change to the data. |
| **GUI** 22 | Switching radio buttons (“Polynomial” 🡪 “Exponential” and vice versa) | Make sure the input boxes are changed correctly | Current inputs are removed and the boxes are changed appropriately | Equation object is instantiated with empty inputs after each switch. |

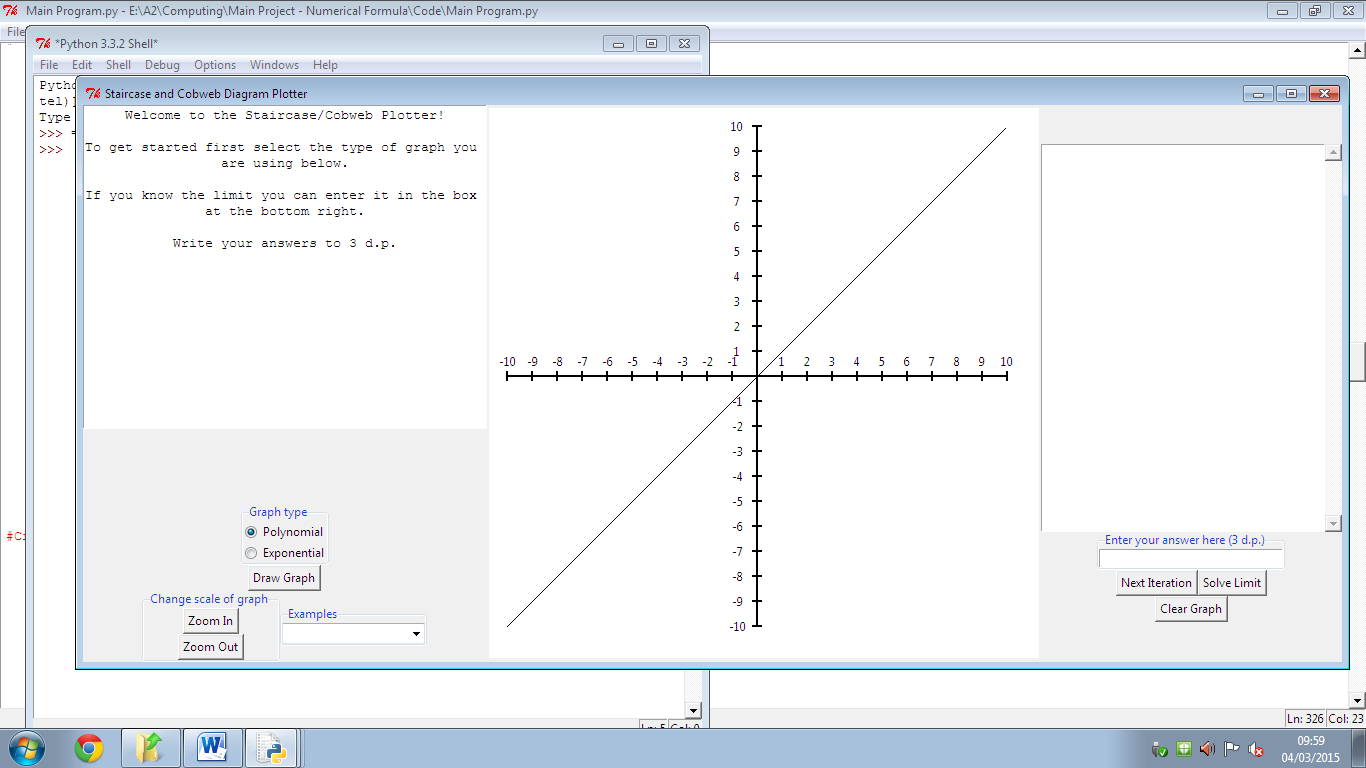
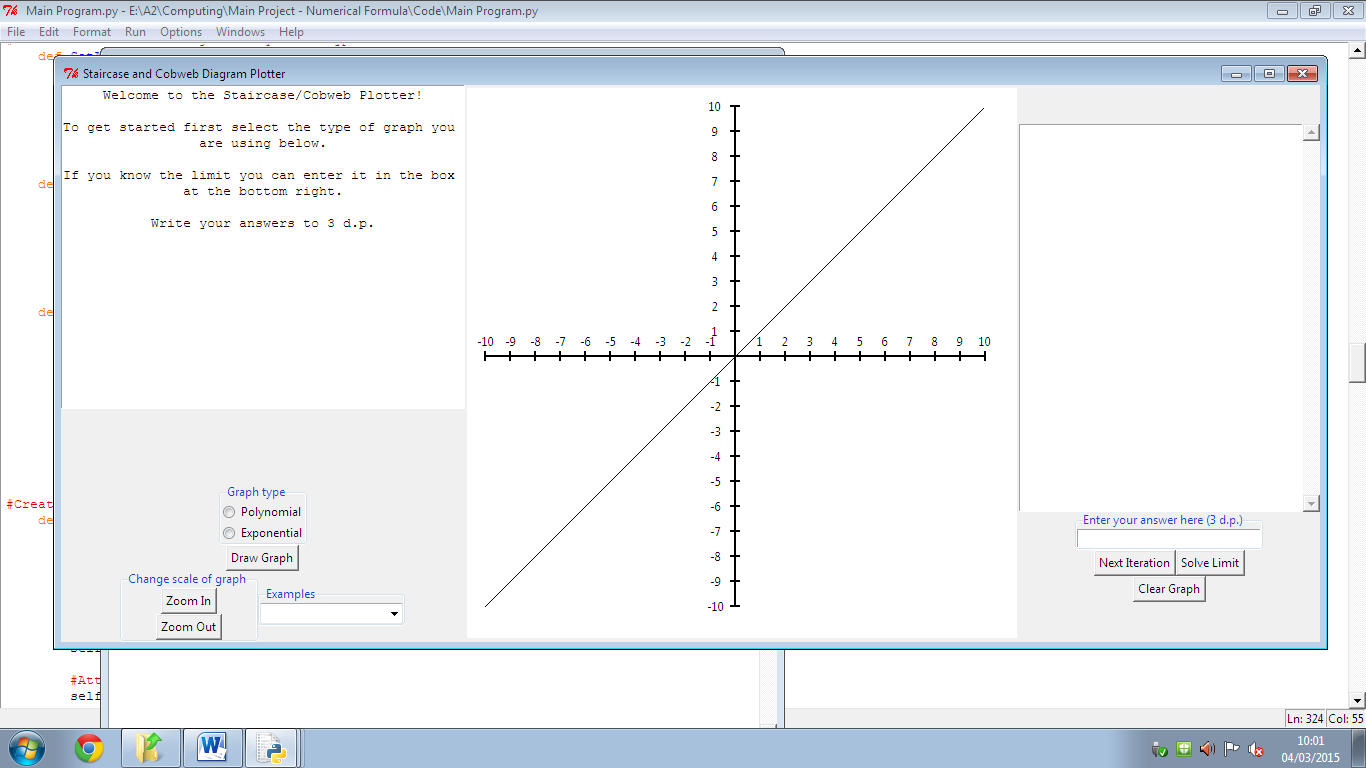
**Input Tests**

These are tests where the user would be asked to enter values into the system. I will use a set of normal data, boundary data and erroneous data for these tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Data | Description | Expected Result | Actual Result |
| **INP** 1 | Polynomial input boxes (normal data) | Inputs of 2 (in each box) | Inputs accepted | Inputs accepted, graph of y=2x3+2x2+2x+2 drawn |
| **INP** 2 | Polynomial input boxes (boundary data) | Inputs of 10 (in each box) | Inputs accepted, part of graph drawn off screen (can be scrolled to) | Inputs accepted, graph of 10x3+10x2+10x+10 drawn |
| **INP** 3 | Polynomial input boxes (erroneous data) | Box(es) left empty/ input of ‘n’ | Error message displayed in output textbox. | “No input found” message for each empty box displayed. “Incorrect type of input” message displayed for each box with ‘n’ as its value. |
| **INP** 4 | Polynomial X1 input box (normal data) | Input of 1 | Input accepted | Input is accepted |
| **INP** 5 | Polynomial X1 input box (boundary data) | Input of -5 | Input accepted | Input is accepted |
| **INP** 6 | Polynomial X1 input box (erroneous data) | Box(es) left empty/ input of ‘q’ | Input rejected, error message displayed in output text box | Error message not accounted for, add in later. |
| **INP** 7 | Exponential input boxes (normal data) | Inputs of 2 (in each box) | Inputs accepted | Inputs accepted, graph of 22x+2 drawn |
| **INP** 8 | Exponential input boxes (boundary data) | Inputs of 5 (in each box) | Inputs accepted, time to draw graph may be long due to large exponent | Program was still calculating coordinates after 2 minutes (due to large exponent).  In theory graph of 55x+5 would have been drawn. |
| **INP** 9 | Exponential input boxes (erroneous data) | Box(es) left empty/ input of ‘a’ | Error(s) displayed in output textbox. | “No input found” message displayed for each empty box. “Incorrect type of input” message displayed for each box with ‘a’ as its value. |
| **INP** 10 | Exponential X1 input box (normal data) | Input of 1 | Input accepted | Input was accepted |
| **INP** 11 | Exponential X1 input box (boundary data) | Input of ‘0’ | Input accepted | Input was accepted |
| **INP** 12 | Exponential X1 input box (erroneous data) | Box left empty/ input of ‘f’ | Error message displayed in output textbox | Error checks not accounted for, add in later. |
| **INP** 13 | Answer input box (normal data) | Input of correct/incorrect answer | ‘Correct’ or ‘Incorrect’ message displayed in output textbox accordingly | “Correct!” message displayed when the answers match (done during test GUI 16).  “Incorrect” message shown when user’s answer is wrong. |
| **INP** 14 | Answer input box (boundary data) | No input given | No comparison of an answer to the limit | System’s answer is still displayed to the user but no comparisons are made |
| **INP** 15 | Answer input box (erroneous data) | Input of ‘k’ | Error message displayed in output textbox | Error checks not accounted for, add in later. |

**User Testing Feedback**

As part of the testing phase of my system development cycle I showed my main customer the system and let them try and use it with minimal guidance from myself. He found some small changes I needed to make to the program, mostly to do with making it clearer on how to use it. A simple example is to not set the radio buttons when the program is started, as the users might think it is already selected.

**Before: After:**

The main customer assumed that the graph type was already selected here so I adjusted it as shown on the right.

Some other changes that my customer wanted included:

* Tell the user to enter a 0 into the input box if there is a term has no coefficient (i.e. if the equation is x2 + 1, the x3 and x input boxes should have 0’s in them)
* Reset the next iteration sequence when the user changes the parameters. This came up because the next iteration button was not executing as intended. This was the first time the next iteration algorithm had been properly tested.
* Find a way of centring the diagram when zooming in/out or if the user scrolls too far.
* Try and make the diagram centre over the position where the graphs converge so the user can easily identify it.
* Make the program more user-friendly. By this the customer meant that I needed to supply more information to the user so that they have a clearer idea of how to use the system.

We agreed that most of the instructional changes can be done by adding some more text into the “how to use” textbox. The centring the diagram addition was a must because the program would have to be restarted if the user loses where the axes for the graphs were. However, my customer said that the centring over the limit change was an optional extra.

I decided not to include the “centre over the limit” functionality to the system because it would not have been completed in the timescale remaining after the testing phase. The way that canvases in Python work also would have made it difficult to implement; where the system plots the graphs is based on where the centre of the canvas is, so changing the canvas’ origin would change where the lines are drawn.

Overall the main customer liked the program at this stage.

**System Maintenance**

**System Overview**

The system that I have made contains a single page that can change dynamically based on the type of graph the user wishes to draw (see design section). There is also a separate text file to go with the main program that has some examples (taken from the AQA Core 3 book). These examples can be loaded into the system by the user; the examples can then be drawn to the axes displayed on-screen while the user attempts to solve the example manually (like how they would in the exam). Students can use the system as practice for staircase diagrams and cobweb diagrams while the maths teachers can also use it for demonstrations to the class.

The main page of the system is split up into three sections.

The left side contains a textbox to act as a tutorial for the program and a variety of different inputs: input boxes, radio buttons and buttons for the user to use.

In the centre of the page is a pair of axes where the inputted graphs are drawn. The user can navigate this part of the page by clicking and dragging with the mouse and can also zoom in and out with the buttons on the left side of the page.

The right side of the page is the “calculations” side. Here there is a textbox that will output the system’s results from calculating the given equation/sequence, some buttons to instruct the program to perform the calculations, and an answer box for the user to input their manually worked out result into.

**System Classes**

My system was designed with object oriented in mind. Below are the classes that my system contains and what they are used for.

|  |  |  |
| --- | --- | --- |
| Class name | Purpose of the class | When is it used? |
| Application | The main part of the system. Contains methods used to create and manipulate the GUI. | The system creates an object with this class on start-up. |
| Polynomial | Stores the inputs for polynomial equations and contains methods specific to solving limits for polynomial equations. | When the user selects to use polynomial equations via the radio button displayed on screen or when an example of a polynomial is loaded. |
| Exponential | Stores the inputs for exponential equations and contains methods specific to solving limits for exponential equations. | When the user selects to use exponential equations via the radio button displayed on screen or when an example of an exponential is loaded. |

**Methods/Procedures in Application class**

|  |  |  |  |
| --- | --- | --- | --- |
| Name of method/procedure | Purpose of method/procedure | Variables used | Purpose of Variables |
| CreateGUI | To setup the GUI on start-up – frames, canvases, buttons, etc. | MainFrame  EquationFrame  InstructionsFrame  EntryFrame  BottomLeftFrame  GraphFrame  Canvas | Frames are created to layout the system’s widgets.  Canvas is the main part of the GUI where the graphs/diagrams are drawn. |
| RecreateEntryFrame | Used when switching between polynomial and exponential graphs. Previous input boxes are destroyed when changing so the frame must be remade. | EntryFrame | EntryFrame is created again in the process of changing graph types. |
| CanvasX (taken from tutorial program) | Used in drawing the canvas, calculates the x ordinates | x (parameter)  a  b | Variables used in a calculation that returns an x coordinate |
| CanvasY (taken from tutorial program) | Used in drawing the canvas, calculates the y ordinates | y (parameter)  c  d | Variables used in a calculation that returns a y coordinate |
| ActualX (taken from tutorial program) | Used when plotting the graphs, scales the x ordinates of the graph so that the lines are drawn realistically | x (parameter)  a  b | Variables used in a calculation (different to CanvasX’s) that returns an x coordinate |
| PlotPolynomial | Plots the graph of a polynomial function | Canvas  Scaled  x  y  y1  Graph | Scaled is a list that stores the values of the graph’s coordinates.  x is the value used in the graph’s equation to calculate the y coordinate.  y is the coordinate produced from substituting x into the given equation.  y1 is the adapted version of y, used because in Python the y axis is inverted (positive y coordinates go down the axis).  Graph is the line drawn using the coordinates in Scaled. |
| PlotExponential | Plots the graph of an exponential function | Canvas  Scaled  x  y  y1  Graph | Each variable serves the same purpose as in PlotPolynomial. The given equation used to work out the y coordinate here is an exponential function. |
| PlotLine | Plots a line between two points on the graph. Only used to draw the y=x line on start-up | Canvas  Scaled  x  y  y1 | Each variable has the same use as they do in the previous methods. The equation to work out the y coordinate here is simply “y = x” |
| PlotTrail | Plots blue lines to represent the trails of the staircase/cobweb diagrams | Canvas  TrailList | A list used to store the coordinates of each of the drawn trails. Makes manipulating the trails later easier. |
| PlotGraph | Decides the type of graph to draw (polynomial or exponential) and calls the appropriate “Plot…” method | N/A | N/A |
| CreateZoomingButtons | Creates the buttons for changing the zoom of the canvas | ButtonsFrame  ZoomInButton  ZoomOutButton | ButtonsFrame for the layout of the buttons. The last two variables create the corresponding buttons. |
| ZoomIn | Zooms the canvas in by doubling the scale of all items inside it. | CurrentZoomScale  Canvas | Checks if CurrentZoomScale is not too large (I pre-set the maximum zoom to x8) before scaling Canvas by factor of 2 |
| ZoomOut | Zooms the canvas out by halving the scale of all items inside it. | CurrentZoomScale  Canvas | Checks if CurrentZoomScale is not too small (I pre-set the minimum zoom to x0.5) before scaling Canvas by factor of 0.5 |
| CentreCanvas | WIP | WIP | WIP |
| StartScrolling | Records where the user clicked so that the Canvas is dragged from that point. |  |  |
| MoveAround | The process for scrolling around the Canvas. |  |  |
| RescaleCanvas | Changes the scale/zoom of Canvas back to x1 so that graph lines are drawn accurately. |  |  |
| DoYouWantToDelete | Message box appears to ask the user if they are sure they want to delete the current outputs of the system. Called when the Clear button is pressed. |  |  |
| Clear | The procedure for deleting the system’s currently displayed graph and text outputs. |  |  |
| CreateUserAnswerInput | Creates the input box where the user can enter their answer for the given equation/sequence’s limit. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |