PHYSICS GRAPH PLOTTER SOFTWARE

William Taylor, Candidate number 2468

ROYAL GRAMMAR SCHOOL HIGH WYCOMBE

Contents

Analysis

The Problem

The physics department completes class experiments with results that need to be collated. Currently each student in the class writes their results onto the board resulting in a temporary record of the class' results which can only be left on the board for the lesson. Writing on the board is also slow and messy and results in students losing lesson time. Sometimes excel is used, however this also takes a large amount of time, and requires the teacher to be able to trust the students to use their laptop one at a time. This can be averted by polling each student one by one, but as each student often has ten or more data points this can either become unfeasible or result in time being lost which could be used to teach. This also doesn't allow for students to submit their results for the teacher for homework should the time of the lesson run out. Lines of best fit produced by students are often imprecise, resulting in neither method being well suited to the class. This is a big problem as experiments are a major part of scientific subjects, especially physics, where good experimentation is tested both in exams and in "core practicals" (effectively experimental coursework). Any class wide errors need to be fixed quickly and effectively, and currently there are not sufficient tools to cope with this.

Design of the solution

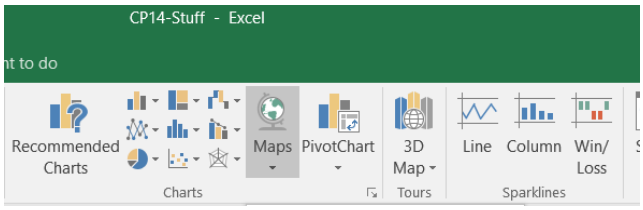
The solution would be a GUI based graph plotter. It would allow students to log in and to submit work to their respective teachers. This can be plotted as a class to check if the results are generally good. A linear or polynomial regression line could then be used to create a line of best fit for the class' data in order to give a perfect line of best fit. This would allow the class to quickly see how well their points correlate with the others and also see if their results fit the model. These results can then be saved. This would result in a class graph which could be quickly created. The students would be able to register and there would be different permissions for staff and students, such as teachers being able to remove data points. Teachers will require another teacher to create an account for them in order to ensure that students are not able to gain administrative privileges. All passwords will be hashed to improve the security of the system. This would mean that should someone gain access to the database, it would be less likely that the person would know login details. Maximum and minimum points can be added to graphs to ensure that students do not submit unrealistic data points, which may heavily alter the line of best fit. The colour of the lines and points would also be changeable to accommodate for preference or requirements. The user will be able to switch between graphs however they choose, however they should only be allowed to view graphs that they have permission for. Teachers will be able to view any graph.

Analysis of other similar methods

Microsoft excel

Microsoft excel offers an extremely wide range of tools for statistical analysis and graph plotting, which are well beyond my capabilities. It contains a wide variety of graph types e.g:

Bar, Pie, Column, Line, Area, Scatter Surface, Doughnut, Bubble, and Radar graphs

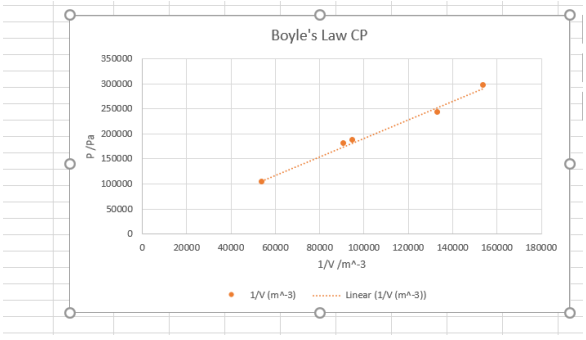


Users select their data by highlighting it in the data table below, and then selecting a chart type in the menu at the top of the interface. It then automatically labels and draws the graphs based upon values given by the user in the cells highlighted.

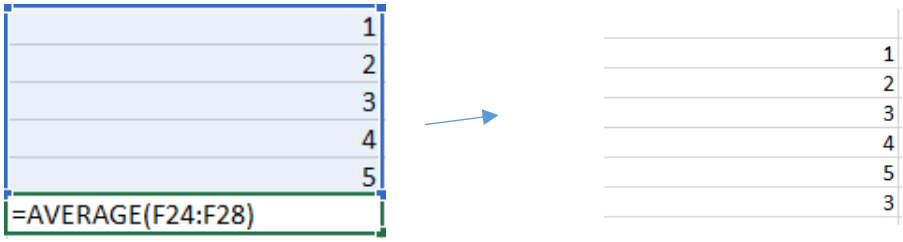
			T = 15		288.2
P /lbin^	P /Pa	V /cm^	V /m^3	n	1/V m^
15	103425	18.5	0.0000185	8E-04	54054.05
26	179270	11	0.000011	8E-04	90909.09
27	186165	10.5	0.0000105	8E-04	95238.1
35	241325	7.5	0.0000075	8E-04	133333.3
43	296485	6.5	0.0000065	8E-04	153846.2

Above the data for an experiment comparing pressure against 1 / volume is selected. These results have already been calculated using manipulation of our initial results with inbuilt data processing.

A line chart is then selected and the graph is automatically generated with correct lines, x and y values, and titles. An option can then be selected to draw a line of best fit across the data



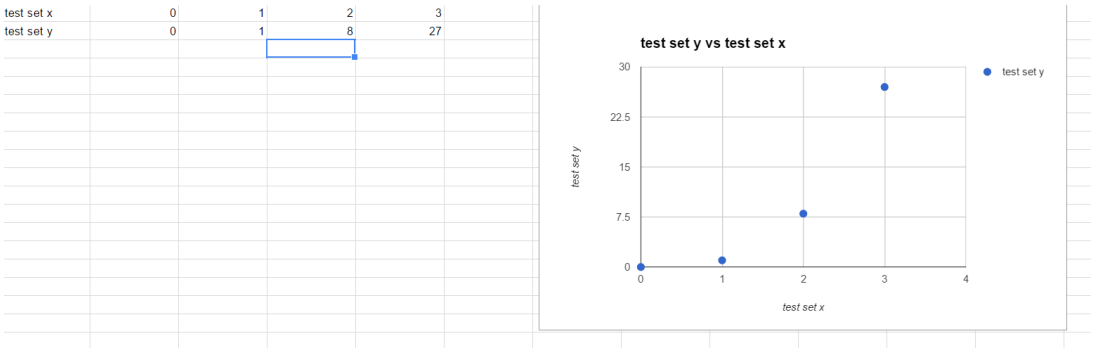
Their solution to data analysis is by a text bar which can be used to enter commands for data analysis such as mean. The range of values can be specified and the type of analysis done and it is completed automatically even if values are changed



This provides a much larger range of statistical methods and graphs possible for me to create in my project, however it lacks one piece of crucial functionality that is required to function as my program would – it offers no option to collaborate as my program would allow. It offers the ability to collaborate on data, however it does not allow for the creation of graphs while collaboration is being used. This would mean that the file would need to be done without collaboration. Collaboration is crucial to the physics department as experiments are often done in groups and collating data would require a lot of work on behalf of the students or teacher. To just view the graph the whole file would need to be sent to the user or it would need to be printed, wasting time, and losing ink. This would be too much for the department to handle as the use of my system would be much easier to manage. They can also maintain admin privileges and can delete data points that they find to ruin the line of best fit for every member of the graph, not just for themselves. This means that Microsoft Excel is not suitable for the task and another program is required.

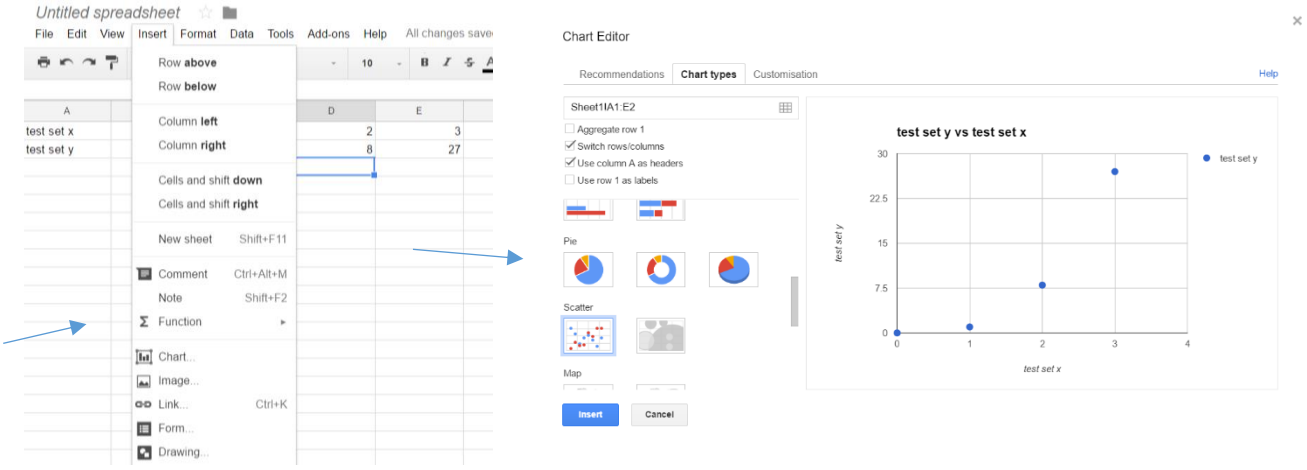
Google Sheets

Google sheets is designed to look very similar to excel, however it is slightly different in some areas. It still offers almost the same options for statistical analysis, however it allows for a more customised graph. It offers a similar cell style sheet, which may prove to be too complicated for some year 7s who have no background in using these systems.

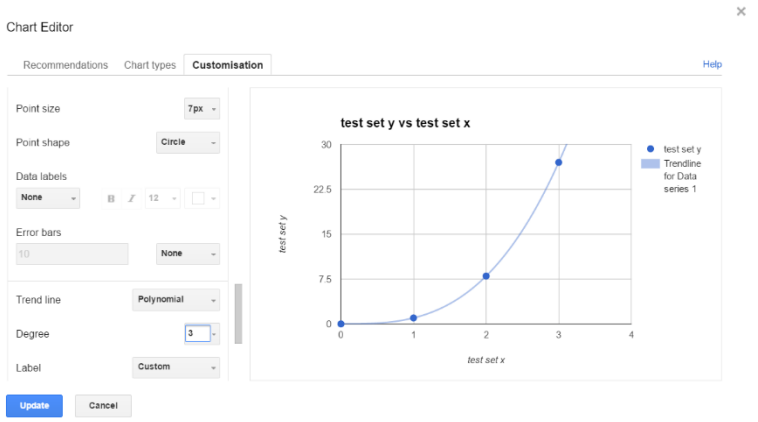


Here I have entered some test data to be used for the graph fit. And created a scatter plot.

The software does this via a set of drop down menus and then a popout window for the graph selection.



Then a line of best fit type can be selected. Here I select polynomial and then an order of 3 as I want a X^3 fit.



This is much more alike to what I had planned, and google sheets also offers the user to collaborate, while still allowing graph plotting. This at first seems like an ideal solution to the physics departments problems, however there is no option for user permissions to a shared graph. A viewer can either view or edit the graph with no customisation. This means that students could just delete data as they see fit and could add text values, breaking both the graph and the statistical analysis set up by the teacher. This would mean that students would need to fully understand every part of the software package, however this is unrealistic with students of lower years, who may not understand the consequences of deleting the data, or imputing incorrect data. If I were to create my own solution I would be able to validate every piece of data being entered, ensuring that It meets the requirements set by the teacher.

Reasons for a computational approach

This problem is suited for computational approaches because it requires the ability for multiple people to add data to the system all at the same time. It also requires heavy use of iteration to optimise the line of best fit. This would take too long for a human to calculate, and needs to be done with a good level of precision. The use of gradient descent is ideal for computers as once the

gradient formula is given it can be calculated quickly with much higher precision than a human could without a computational aid. The speed at which computer software will be able to plot the points and also the line of best fits would be much quicker than a physics teacher doing the same, with saveable results, rather than the graphs being rubbed off a whiteboard at the end of use. The problem can be heavily abstracted as there is no need to understand the data being used, only to produce a good result based on the data. The system will perform well using only low order polynomial terms, meaning that fewer resources are required while still maintaining a good result. The program can be broken down into many different sub sections. The user interface and database interaction can be mostly separated from the graphing. The changes in users will not influence the data being processed by the graph drawing methods. The ability to store data is much better than using paper methods as storage is much more efficient and it is much easier to search for values. The processing is also much quicker for even simple calculations such as averages.

Results

After reviewing these programs the importance of statistical generation is evident. I shall include some basic statistical tools in my program which may be relevant to physics. These will be added as a precaution in order to ensure that the program can be used even if the syllabus changes, as currently there is not a need for these statistics. I also have understood the great importance of polynomial graphing as it allows the department to display even the most complicated trends shown in the A level specification. I will also ensure that the students can only access the functions that they need to try to reduce the risk of abuse of the system.

Platform chosen

I chose to use python for my back end as it allows for a very quick development cycle. As this is a prototype of the end product it means that I am able to produce it quickly and check for errors before possibly porting my program to another language. It also allows me to use PyQT, a GUI library which has many features useful to me such as a tab system. It also allows me to use “web views” which can let me display my generated graph graphics. These will be produced in JavaScript as it is a very effective front end language. This may also make it easier to port my program to web based languages for use on phones or tablets should I develop the solution further

Computational methods

The program can be broken down into the following problems (the plausibility of solving them computationally will also be explained):

Login

The user will be able to have their own credentials unique to their user account. This will be stored in a database along with any other information required for the system (e.g. date of birth, class). The password should also be hashed in order to provide security in the case of unauthorised access to the database. This is advantageous against paper as each user can have their own graphs which can only be seen by people specified, whereas on paper it would result in large amounts of graph paper being used. This also means that the graph cannot be easily lost which prevents students from using it as an excuse for not completing homework.

Registration

Students must enter details specified to create the account which will be linked to the database. This is necessary for the solution to the program as students must only be able to views that staff want them to adjust. This is disadvantageous as it requires each student to go through extra effort and

may require teachers to have to chase up students. Teachers would also have to get other teachers to register them. I chose this as it means that students can't guess or overhear a code that would be used when registering. This on the other hand would need to be done every time a new teacher joins the school in departments that use the system. This is required for the system to be practical, and to allow maintenance by staff. Without user privileges (staff accounts) it would be impossible to remove points as students could not be trusted to maintain the graphs by themselves.

Graph creation

A real benefit of the program is the ability to create a graph which can then be viewed and adjusted by anyone who has a code. If the graph code were to be sufficiently long it would mean that users couldn't guess the codes, making the graphs private without a code. This is beneficial as it means that the graphs can be quickly shared between users and each user can add to it without further effort. This can be done by using another table which links graphs and students. This means that a many to many relationship is avoided.

Class creation

Classes will be groups of students who are linked as a group. This will consist of a class table denoting these groups, with each user being linked via a permission table. This would mean that staff are quickly able to assign a graph to a large group of people. This is better than manually getting each person to create a graph separately where results can't be collated, and is also better than getting them to manually collate the data which would be extremely time consuming.

Statistical analysis

Statistical analysis is extremely fast compared to human analysis when precise results are required. The results will also be guaranteed to be correct should the algorithm be correct, removing human error. This data would not be feasible should the user do it themselves as it would result in a large amount of time taken, which would either be unexpected from homework, or be impossible in a lesson. These statistics on the other hand are very useful when reliable for determining values of results and finding further results. Below I list the time complexities of the simplest time efficient algorithms for each of the following statistics:

Mean – O(n)

Number of values – O(n)

Sum of X - O(n)

Sum of X^2 - O(n)

Sum of y - O(n)

Sum of xy - O(n)

Minimum of x – O(n)

Maximum of x – O(n)

Minimum of y – O(n)

Maximum of y – O(n)

These important statistics for finding a gradient are all o(n) meaning that the system will likely be very fast for linear regression for lines. This is because the gradient and intercept are both found by using these results using multiplication or addition, meaning that further complexity is O(1). This results in the whole system for finding the gradient and intercept O(n) as this is the highest degree of complexity. This is therefore a fully tractable problem in 2 dimensions.

On the other hand, I plan to use gradient descent to find the values of the coefficients for higher order polynomial terms. This algorithm is much more complex and requires a large amount of iteration in order to find the coefficients. This could not even converge as the rate of descent could be too high, meaning that the algorithm fails to find the optimal coefficients. As a comparison the "normal equation" method is used to understand how quickly the algorithm will converge. I plan to use gradient descent rather than the normal equation as should a large amount of numbers be used it would less likely to crash the computer due to overuse of memory. This is because the gradient descent uses one point at a time, whereas the normal equation uses every point at once to find the optimum.

The complexity of the normal equation is $O(mn^2)$ where m = number of data points (e.g x and y values) and n is the number of polynomial coefficients. This will be in the form:

$$A_6 X^5 + A_5X^4+A_4X^3 + A_3X^2 + A_2X + A_1$$

Where A_k denotes the kth coefficient of X. as we can see there are 6 coefficients no matter how many data points there are, meaning that this problem is very easy to solve with our given data points. I will choose gradient descent for scalability purposes, however should it be found to be too intractable to be manageable then it will be replaced with the normal equation method. This should mean that the algorithm is much quicker than a person calculating the values, or even sketching a curve based upon eye.

Graph sketching

I plan to use javascript to plot my graphs. To calculate the positions of the axis I can just use the previously calculated statistics. Then I can use these values to transform numerical values to x and y positions on the graph. These can then be used to calculate y values based on the x values every pixel. These can then be joined together using javascripts "lineto" this would result in a continuous line. This would result in an O(n) complexity for drawing the line, making it easily managed and very quick this would result in an extremely fast sketch, which has a big advantage over hand drawing the graph.

Stakeholders

The main people interested in the development of my program are the physics department, including both teaching staff and students. The teachers may require to use it in class, and students will use it at home as part of homework. Other departments may also show interest including the biology, chemistry, and mathematics departments. These could be used in a similar way in the other departments.

Proposed users

Users

My proposed users will be the Physics department of the Royal Grammar School High Wycombe. As a representative Mrs Dove, a long-standing physics teacher with interests in improving teaching methods, will check my work and sign it off against requirements. I will also ask my intended users, students of the Royal Grammar School, to test my program after development. The physics department is relatively large with roughly 800 students. Therefore scalability is important for numbers of this size, however larger numbers are not expected.

Researching their needs

To research the needs of my clients I shall interview Mrs Dove on the features she desires and query about the usefulness of my program. I chose an interview as it allows her to say anything she wants, rather than just ticking boxes, and allows her to discuss solutions with me to help her to understand my proposition. I will also ask students more general questions to get an understanding of what they want to get out of the graph program.

Interviews

Staff interview

I spoke to Mrs Dove, a teacher in the physics department, and asked her a few questions about her concerns and expectations for the project. Below are the questions (underlined) and the answers.

<u>What do you currently do when plotting graphs with your class?</u>

Currently we ask the students to come up and write their results down individually, or just take the results from one student. This means that the results either take a long time to plot or only use a small amount of data. After this the line of best fit is plotted freehand by either myself or a student. This often isn't mathematically correct. On other occasions we sometimes use Microsoft Excel to create a regression line, however this takes a long time as one person has to input all of the data and also Microsoft excel is expensive to purchase, which could be a barrier to many users.

<u>Are there any initial concerns you have about the program?</u>

Students are always looking to try to experiment with things available to them. I would be concerned if it were possible for users to apply for admin anonymously with names which could cause offence. I would also hope that the ranges of dates that a pupil selects are reasonable as they would help us to search through the users by their date of birth. It would also be useful if I could allow single students access to graphs which they could not access as they are not part of a class. This could mean that students could look ahead to see what is done in the future to aid self-study for eager students.

<u>Are there any features that you would like to be added which are not described as above?</u>

I think the objectives cover most of the usage that I would see in my classroom; however, I would definitely like the option to be able to remove the line of best fit all together. I would also like to be able set a range of data in which the points can be, as this stops anomalies from destroying the line of best fit for the graph.

Would you consider this program to be useful?

Initially after hearing the idea I was sceptical as I like to get students to draw their own graphs so that they get practice, however the option to be able to have multiple students submitting data and collating it means that after they have drawn their own graphs it offers a good way to see how the class did as a whole. The class graph could be compared to scientifically accurate graphs on the internet, allowing for errors in experimentation to be shown. Another reason why this could be useful is that the program can be used by students at home. This means that they can submit data after the lesson has ended so that the data can be collated even if the experiment ends very late.

Could you think of any disadvantages such a program could have?

I have very few disadvantages I can think of, except that students may not submit work to the class and I may not be able to know.

To address the concerns that Mrs Dove had about my program I have added a few objectives to ensure that my program remains relevant and useful to the department. It is difficult to stop users from creating names which may offend people, as many can adjust the words to avoid filters, and also filters may unintentionally reject valid names. In order to stop this from happening on teacher account creation requests I have made it possible for only other teachers to submit new admin users. This should prevent any spamming of requests. I have also added an objective to validate the date of birth for registering students to only dates that are feasibly possible to be in school for that period. I will also add an option for allowing specific students to be added to a graph. This will allow individuals to be added to a graph. I will also allow users to view the graph with no line of best fit as it will allow the users to view just the points without the distraction of the line. A range for both axis will be added so that large values for either x or y will not stretch the axis, obscuring the data.

I will also add an option to search for users who have been permitted to view the graph, but have not actually submitted data. This could allow teachers to set homework for the plotter to submit data when they get home.

Student interviews

I also spoke to two different students of the school who study physics – Finlay Franks (year 13)(FF), Bryan Potter (year 11) (BP). Here are the answers provided by the students:

How important are graphs for your current studies in physics?

FF – I am currently studying physics and I see that graphs are becoming increasingly important to my studies. In exams graphs are often used to not only display the relationship between different values, but also to calculate further values. An example is using the gradient of a graph comparing voltage and current to find the resistance in the circuit. Graphs are also important in exams as it is important to understand how to read a graph.

BP – currently I am studying my GCSEs, meaning that my skills in physics are important for my grades. I do occasional experiments in class, and when we do these we normally draw up graphs on paper. Graphs often helps me to understand the concepts, because a straight line helps me to visualise how things work.

Do you draw graphs in class and if so what are the limitations?

FF- We draw graphs very frequently in class and sometimes at home for homework. We are doing "Core practicals" which is a log of our experimental work which needs to be passed for my A levels. These require a lot of graph work; however, they must be drawn. Currently doing graphs in class takes a long time as it must be done on graph paper. Graphs are never done together as it would take way too long. Once I attempted to do a graph with a few friends. We did it using excel, however it involved lots of emailing results, and then copying and pasting them into a single excel file. This took a long time and resulted in a lot of email requests for the results.

BP – we occasionally draw graphs. I wish we could draw them more but it normally takes too long for the teachers to justify it. We do just enough to learn the skills. When it is done it often takes about 40 minutes to do. Once in a class we tried to do a graph with everyone's results, but it was done by polling each member of the class individually and adding them to an excel spreadsheet. This took a long time and although the results are impressive, my teacher said we would not do it again because it used up all of the lesson.

Would a graph based solution as proposed be useful to you and what

FF – In its currently proposed form the graph plotter would be mainly useful for check my graphs against others doing "core practicals". If my friends were to add data to the graph then I could check my results against the graph between all of us. With this large collation of data error would be removed and it would be similar to doing lots of experiments, a good way to get reliable results. To make it more useful to me I would like to be able to see the equation of the line of best fit formed. This would mean that I could compare it to my own so that I can understand whether my calculated results are reliable.

BP – This would be useful to me as I struggle to understand what is in the textbook without visual help. The large graph from class results could be projected onto the board and it would help me to understand how the equations work. The ability to change colour for the lines might help me to visualise the results when trying to remember then in an exam.

Objectives

In response to the interviews from the staff and students and my research into similar programs I have deemed to be important to both the stakeholders and also the operation of the program.

1. Collect username for registration
2. Collect password for registration
3. Collect Forename for registration
4. Collect Surname for registration
5. Input date of birth
6. Validate string data from registration against pre-set rules such as length.

7. Existence check user names to ensure that they aren't in use
8. Check that the date of birth of users is not too early to currently be in year 7
9. Check that the date of birth is not too late by comparing it against the date at the time of use
10. Provide an error message should registration data be invalid.
11. If validation is successful, the student account should be added to the database with a unique primary key
12. Allow the user to leave the registration screen should they want to stop creating an account
13. Provide an option for student or teacher login (each requires different data so should not share a table)
14. Hash the password entered by the user
15. Check the username in the database that has been entered for a password
16. Compare the hashed password in the database against the hash of the password provided by the user
17. If the user enters the correct details, change the user to the username entered and open the main window
18. If the password entered was incorrect an error message should be provided
19. Allow teachers to create classes
20. Allow the facility for teachers to input class name
21. Allow the teacher to input a subject name
22. Allow the teacher to input the year of the students in the class
23. Validate that the year is between 7 and 13
24. Generate a code for the class
25. Check that the code is unique
26. Add the class to the database using the fields that were entered
27. Allow students to enter the code of a class
28. Check the database for a class corresponding to said code
29. Should this code correspond to a class the user should be given permissions for that class
30. Allow graphs to be created by teachers
31. Provide the option to allow all members of a class to view the graph
32. Generate a unique code to join the graph
33. Allow the user to type in codes for graphs
34. If this corresponds to a graph let the user join it
35. Display the graphs available to the users for selection
36. Allow the user to select the graph by clicking on it
37. Display to the user which graph is selected
38. Allow students to add points to the graph
39. Plot the x axis with correct scale
40. Add text for the maximum and minimum values
41. Select good increments for the x and y axis
42. Write sensible values on the scale for different values
43. Plot the y axis with correct scale
44. Add X title to the graph visualisation
45. Add Y title to the graph visualisation
46. Add main title to the graph visualisation
47. Allow the user to select no line of best fit
48. Allow the user to select linear line of best fit
49. Allow the user to select polynomial line of best fit

50. Allow the user to select line of best fit colour from a predetermined selection
51. Allow the user to select point colour from a predetermined selection
52. Plot points onto correct point based upon the scale
53. Plot points with the correct colour as selected by the user
54. Display the line of best fit based upon the user selected colour.
55. Calculate the statistical value of the gradient and intercept for a linear line of best fit
56. Use these values to calculate the y values for each x value within the range
57. Not plot a part of the line if it is not within the y axis.
58. Calculate the polynomial values for the line of best fit when polynomial is selected
59. "Regularise" the values in order to ensure that the line of best fit isn't too biased.
60. Plot the polynomial line of best fit by calculating y value based upon x value
61. If the calculated y value is out of range the value shouldn't be plotted.
62. Calculate the mean value of the data points
63. Display the mean value of the data points
64. Calculate the number of points on the current graph
65. Display the number of data points for the current graph
66. Allow a user to change their password
67. Validate the password entered against requirements
68. Display the equation of the line of best fit provided
69. Interpolate X values given

Limitations

The main limitation for my program is being unable to label the axis on my graph. At my current skill level, it would be extremely challenging to be able to give sensible answers with a realistic scale and fitting exactly on the axis. I would also need to ensure that these numbers do not interfere with the points.

Proposed hardware and software requirements

Requirement	**Reason**
Windows	To handle the opening of files and for opening python and PyQT files
Processor with clock speed over 1Ghz	The minimum requirement for running python, as stated in the python documentation
Ram over 512mb in size	The minimum requirement for running python, as stated in the python documentation
100mb available hard drive space	To store the python files, python itself, and the PyQT files
Mouse	For interacting with the UI
keyboard	For entering details in text boxes
Monitor with 1366x768 or greater resolution	So that the GUI can all fit on the screen at once

Design

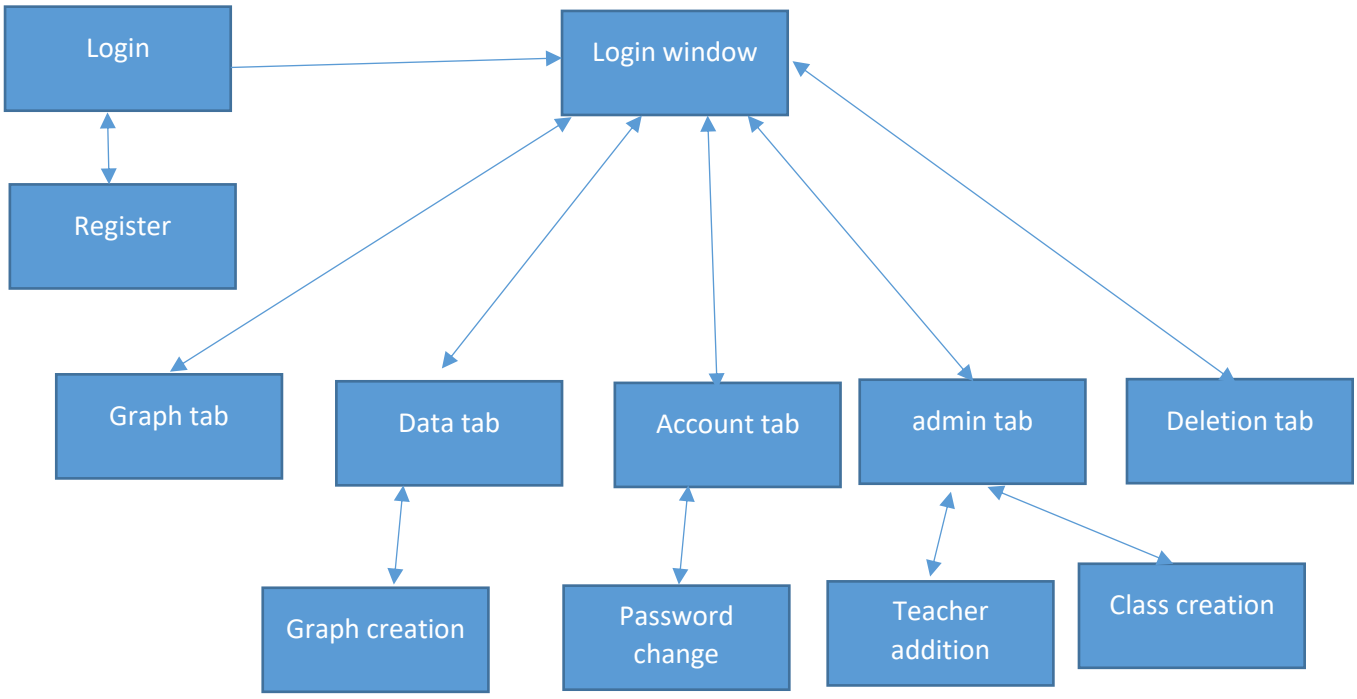
Algorithms used

- Registration validation
- Database check for login
- Graph line sketching
- Point to coordinate
- Point sketching
- Line of best fit drawing
- Statistical calculation
- Linear regression
- Polynomial regression – gradient descent
- Graph/class code generation
- Graph joining validation
- class creation
- graph creation validation
- point enter validation

User interface

My program can be broken down into further sub-interfaces which will then be discussed and evaluated individually per both the users' and technical requirements. These will be listed below and then further expanded upon.

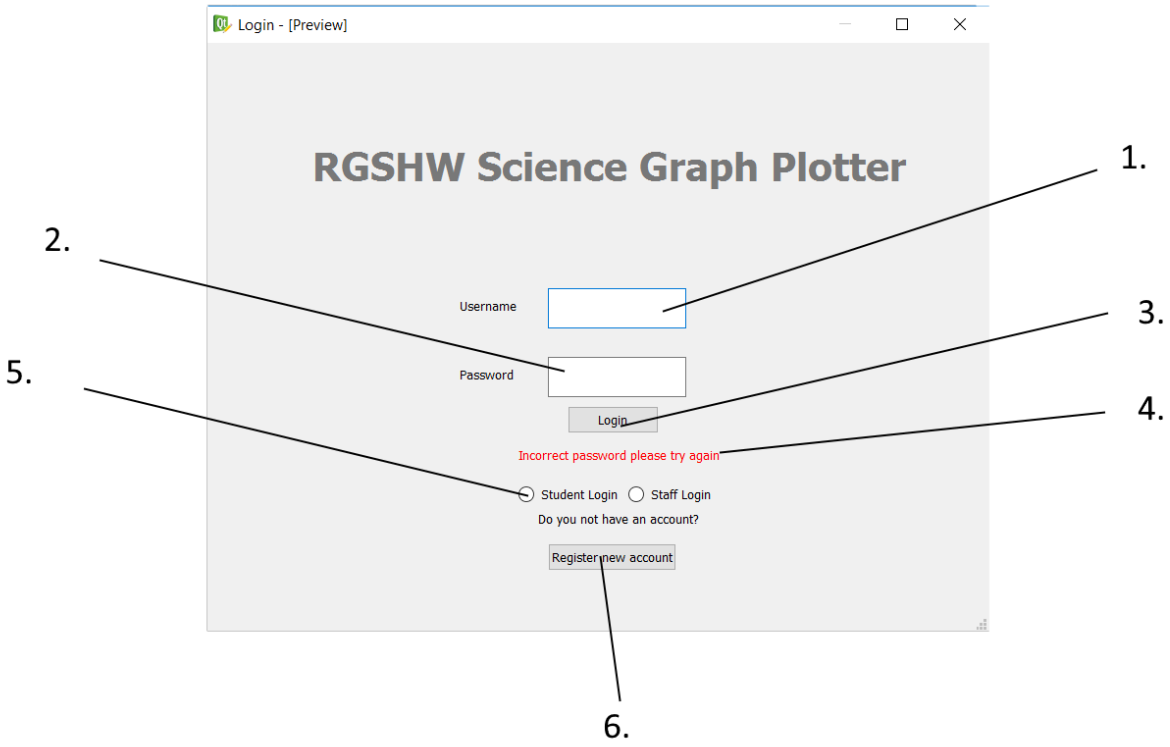
User interface diagram



Login

The user should be allowed to login on the home page. There shall be 2 login versions – staff and student. Each should check against their respective databases to validate user credentials. The passwords in the database will be hashed as a security measure to ensure that anybody viewing the database would be unable to see users' passwords.

Login interface



---

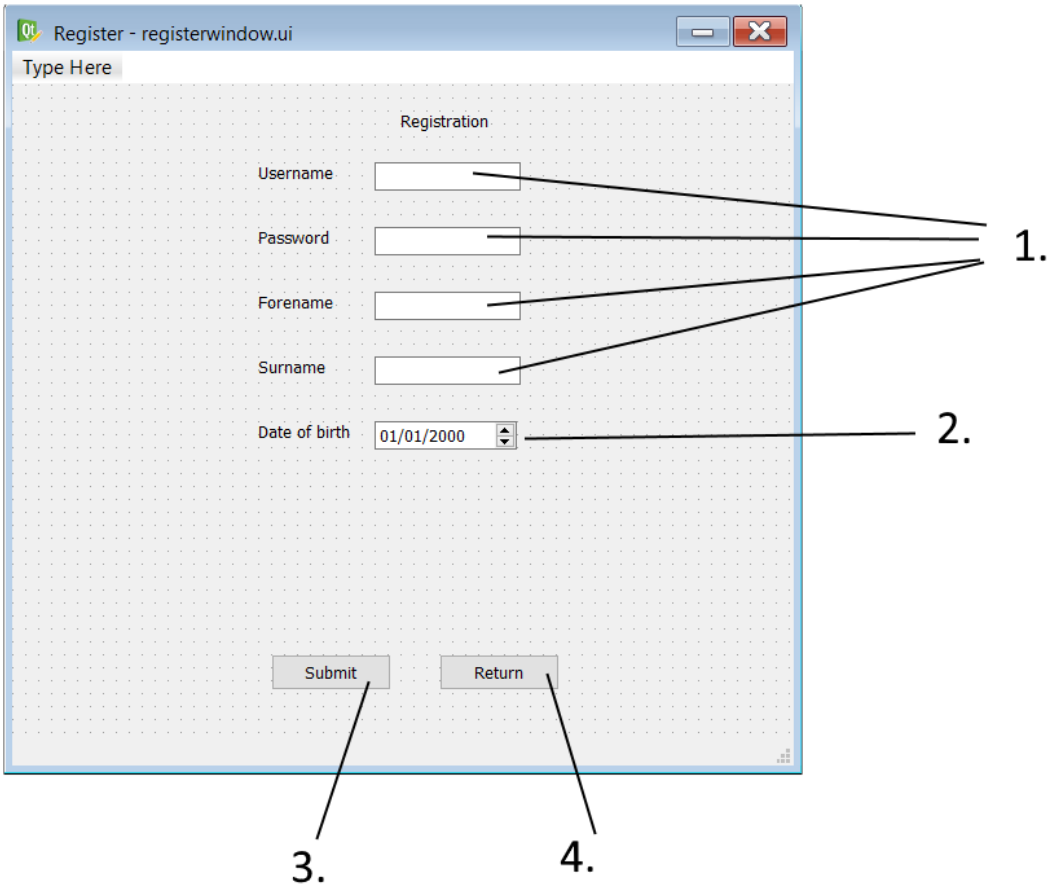
1. The username box will be a pyqt line edit box which allows the user to enter their details
2. The password box will be the same as above and will have its contents hashed for comparison with the database password
3. The Login button will start the process of comparison against the database. It will first check if the user wants to login as a teacher or student and then compare hashed passwords for a given username. It is in a position which is often used in login screens to make the interface seem familiar to the user without using the program before.
4. When the login button has been pressed, it will either open a new window and close the Login window or it will show this message. This message will originally be invisible, however if an incorrect password is given it will be shown to the user. Red has been chosen to make the text easily readable and noticeable by the user
5. These two radio buttons offer the user a method by which to change between the two login states – Staff and Student. Only one may be pressed at any one time meaning that the user

will be checked against the right database. These are both stored in separate databases as each user must input different data when registering. I chose radio dials as it means that the user can only select 1 option. It is also in a visible position meaning that the user is likely to swap between them if they are a teacher. It is most likely that the user is a student, therefore the student radio dial will be selected automatically, reducing movement of the mouse for most users.

6. This button opens a new window for a new student to register. This only caters for students as a new staff member requires another staff member to add them as a user. This prevents students from attempting to gain admin privileges. This button is also in a prominent position making it more clear to new users. This means that most people will notice it if they want to register as they scan the page for the option. The text above it also explains to the user that they need to register a new account if they have not used the program before, making sure that the user understands the functionality of the interface.

Register

This UI file will allow the user to create a new account for my program. This will add them to the students database and check to see if what they have entered is valid. I chose to keep buttons central as it means that the text inputs stand out to the user.



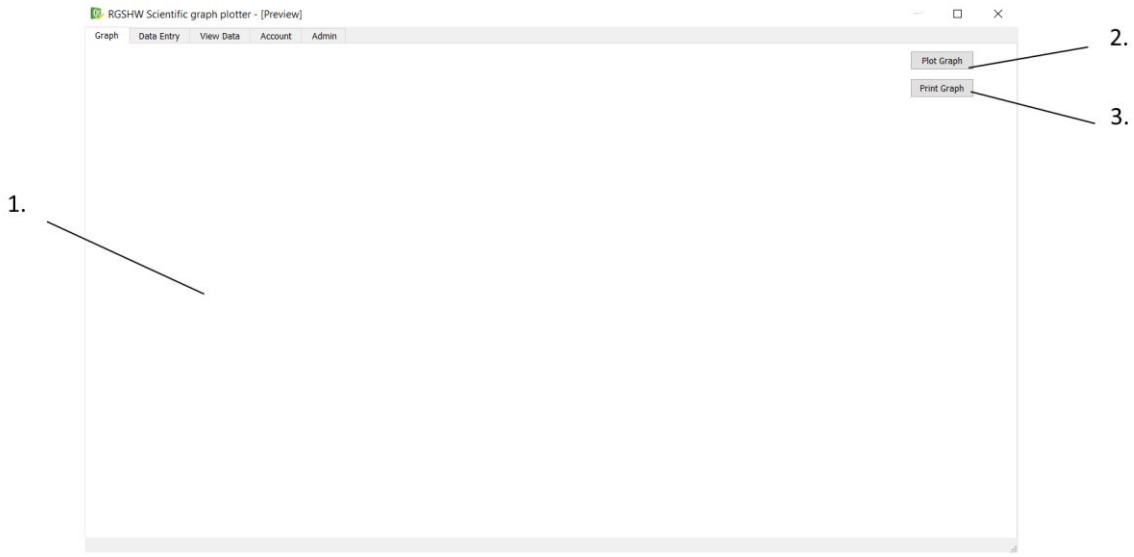
1. These fields offer the user a way of entering data referring to themselves. Each shall have a different check for validation.
   Username- This must be between 7 and 20 characters long. These characters will also have to alphanumeric. It must also be unique and not be the same as any username currently in the database
   Password – The password must also be between 7 and 20 characters long, however has no character restriction to improve security. Before addition to the database this must be hashed to add security. Sha512 will be used as it is secure enough for this program.
   Forename – This can be between 2 and 15 characters long (allowing for large variation in name lengths) It will also have to be alphabetical as names may only contain letters.
   Surname – This will be validated in the same way as the forename.
2. Date of birth – This UI module only allows valid dates to be entered, however extra validation will be used in order to ensure that the dates are in the range of what students can expect to be (e.g. born more than 10 years ago, and less than 20 years ago). I chose the date widget as it means that less validation is required as the widget ensures that all inputs fit the data structure that I am using. It also means that the user fully understands the format that I require and also makes it easier to select the date.
3. The submit button executes a function which validates all of the other fields as explained above. If successful it will add the user to the database, otherwise it will display the error. I chose to place this button and the return button below the rest of the inputs by quite a bit to make the text stand out more. This ensures that the user will fill out the text boxes before then searching for the submit buttons. The button is on the same level as the return button for structure, and by convention the submit button is on the right, and the reject button is

on the left, meaning that the interface seems familiar to most people who have used programs or similar software before.

4. The return button goes back to the login screen

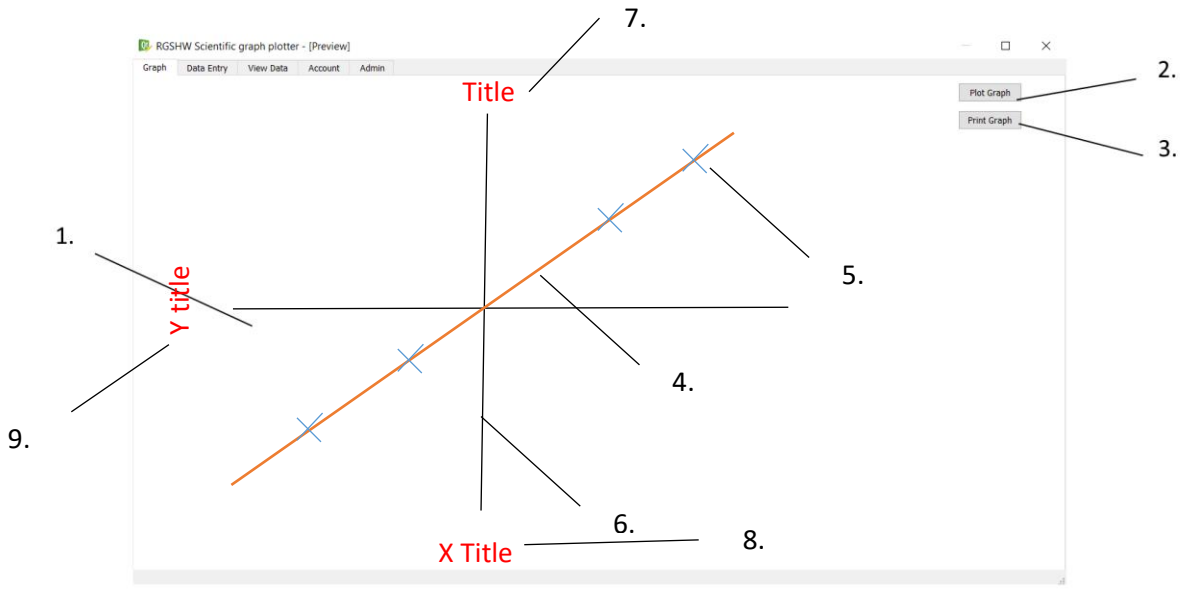
Main Window

This is where the main functionality will be displayed including the graph and the user settings



---

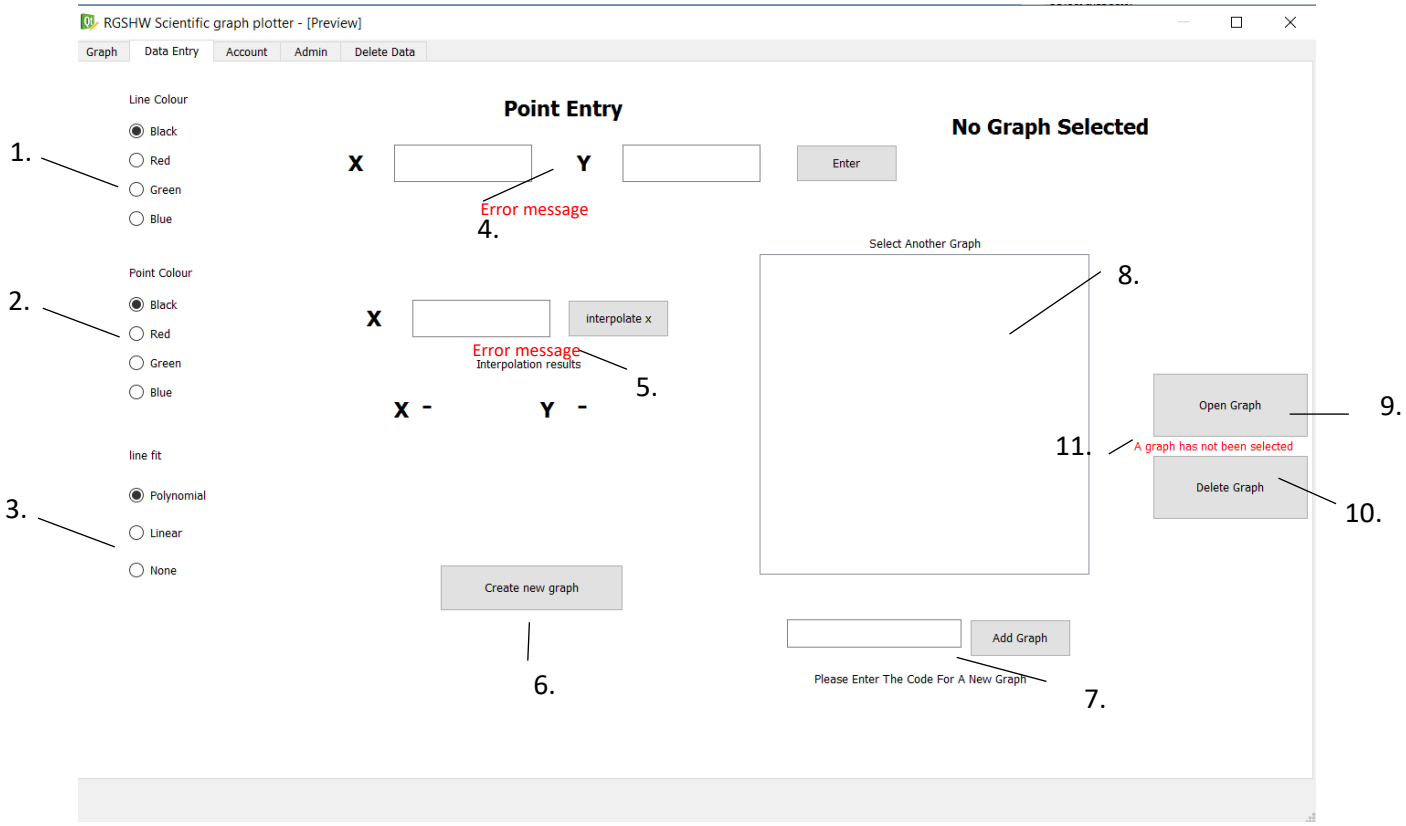
1. This is a web view where the graph will be outputted on to. The web view will use a html file which contains a html 5 canvas. This canvas will have the data points displayed as crosses on a 2d plane to show where the users' inputs lie. There will also be a plotted line of best fit. This will be calculated in the python code and then y values for small increments of x will be used to create a curve for the overall graph. There is also the possibility for no line of best fit if decided upon by the user. The data will then be fed into javascript where an algorithm to plot the data will then be run. This will use settings set by the user such as point and line colours. It will also use X, Y, and title labels in order to add text to the graph. I chose to make this take up most of the page as I believe that this shows the importance of the graph before it is even drawn. The blank space where the graph would be denoting to the user that this is a key piece of functionality and must not be overlooked. When drawn it also means that the graph is more visible, making it easier to use the graph on projectors which fits in with the idea of it being used in schools.
2. This button plots the data as described above
3. This button will use the inbuilt pyqt print function to print the webview to a selected printer. This is useful for showing data to teachers or a class.



4. The line of best fit is displayed to the user in the form of a line of the graph. It will be fitted to the points as shown. Here the points show a linear pattern and therefore the line linearly fits the data. This will have its colour selectable by the user to ensure that the users feel as if the program is customisable to their needs. This benefits the user as they can display it in a way that they prefer rather than having a fixed colour such as black
5. The data points for the current graph are displayed to the user so that they can see how the points contributed to the line of best fit. This is a crucial part to the graph drawing program as should the user select “no line of best fit” they will still be able to view the points
6. The axes of the graph are shown here. They will be dynamic in order to show as many of the points as possible without losing space to redundant places on the graph. This means that if more of the range is to the left of the graph, then the axes will shift to the right to display them more. This means that data is less bunched and can be viewed more easily
7. The title of the graph will be shown here with the text chosen by the user. This is in red as black text would be easily missed amongst the other lines and it adds character to the graph. This both improves readability of the program and looks. It is centred by convention for graphs which also improves looks and helps the user to understand that it is the title of the graph.
8. The X title specified by the user will be displayed here. This is coloured and positioned with the same reasoning as the title, but Is placed at the bottom to convey that it is indeed the x title by convention
9. The Y title for the graph is shown at the side. It is rotated by 90^{o} by convention to both display that it is denoting the y axis, but also to take up less space. It is also centred on the y axis to provide good visibility and is red for reasons shown above.

Data entry and graph settings

This section of the GUI is within the main window. It offers the options for the current graph and the options for adding new data points for students. This is one of the most key parts of the program as it means that the user can do most of the functionality on this section of GUI. It is important that this section looks presentable and is understandable as the user will most likely spend most of the time on this screen.



As you can see this page has a large amount of functionality. This is the core piece of the program and should be presentable and understandable. As each piece of functionality on this page is linked in its function it is important that each piece is on the same page. This means that the page could appear cluttered, however it is not as I have ensured that each piece fits into the design of the interface as a whole.

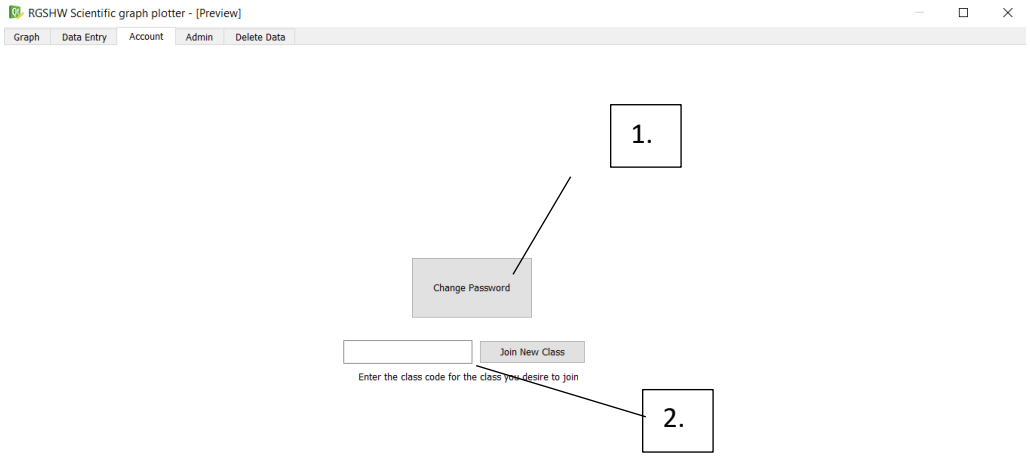
1. These radio dials provide the options for the colour of the line of best fit. These are important as explained on the graph page section. It is crucial that the buttons are radio dials as it visually explains to the user that only one button should be pressed. This is ingrained in most users as it is convention that radio dials are used for individual options
2. Point colour also has an option for colours. These are also using radio dials to display to the user that only one option is selectable. When these buttons are pressed, they change a global variable to a string connected to that value. This can then be added to the javascript to change the colour.
3. The graph type is also important and is placed next to the colour options. This visually denotes that the settings are all linked to the graph, and looks aesthetically pleasing, as all radio buttons are in a vertical line. I am attempting to use a uniform standard for options by using the same type of button to display the same type of operation. This means that the user will quickly understand how my program works, and as one thing is understood it quickly shows how the other things also work to the user
4. These inputs are there to enter points to the currently selected graph. The X and Y are shown in large bold lettering so that the user looks at the data entry first. This means that the user sees the point entry first, denoting its importance. The button is to the right in order to show the user the inputs first to try to ensure that the user inputs some values before clicking the button. As the button is close it visually denotes that the button is related to the values. The error message will be changed depending upon the error with the data

that the user has entered. This means that the user will understand what exactly they have entered incorrectly. This will also be in red to match my convention for error messages, not only ensuring that the user knows that an error is present, but also ensuring that the user spots it quickly, reducing frustration. This will be hidden for teacher accounts as it is necessary for only students to enter points, not only for database integrity, but also because the tool should be used for class experiments and homework rather than just to display result

5. This is the interface for interpolating values. The value for interpolation is limited to x values being entered rather than x and y as the y value would result in multiple x values and may not actually be calculatable. The error message uses previous conventions explained. The results are in bold with the values labelled to ensure that the values are seen.
6. This button links to the create a graph window described later so that new graphs can be created. This will be hidden from students so that only teachers can add graphs. This is moved to a separate position so that it stands out and seems enticing for teachers. This is important as the only time when teachers will struggle with the interface is when they first start using it, and making it stand out will help with this.
7. This text box allows the user to enter the code for a graph that they want to join. This code will be checked against the database and if the code corresponds to a graph it will give them position to add points and view it. It is positioned below the graph selection table, helping users to understand the relationship between the two.
8. This tableview shows each of the current graphs that the user has access to and information about them. This allows the user to view each of the graphs. The user will be able to click on a row and the row will be highlighted. When the graph is clicked, it can be loaded and the bold writing above will change to the graph which is selected.
9. When this button is pressed the currently highlighted graph is loaded for calculations. This means that the graph can then be plotted and viewed. This graph will then be used for any functions where a graph is required. If no graph is selected the error message below will be shown (it is automatically hidden)
10. This option will be hidden for students, but is available for teachers. When a teacher selects a graph this button is pressed and the graph is removed including any related records. This means that the user cannot then load that graph

Account page

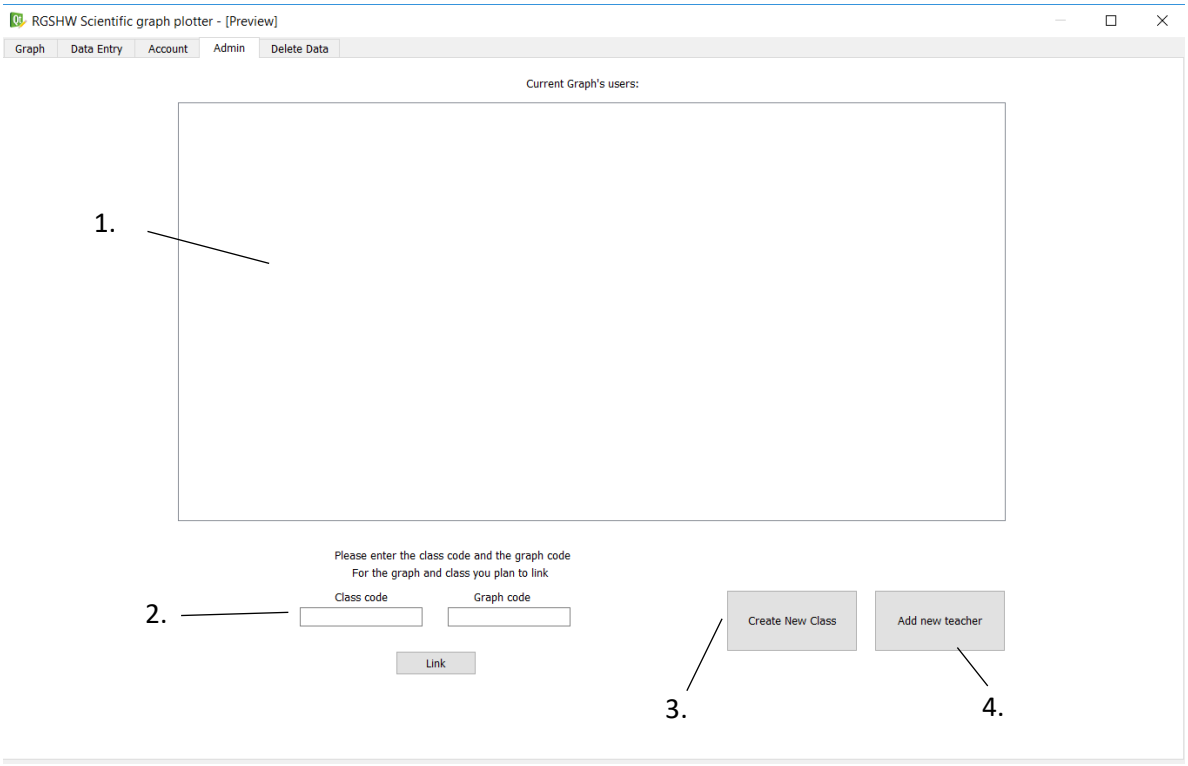
This screen contains the displays related to the user's account. This includes password changing and joining classes. This page is only visible to students as it is not relevant to teachers, who do not require the option to join a class as they see all of the graphs anyway.



1. This button links to the window designed for changing password for a user. It closes the main window and opens the password changing window. It is centralised and enlarged to denote its importance to the user
2. Here a user can add a class code to join the class. If this is correct the user is given permission to see all of the graphs for that class

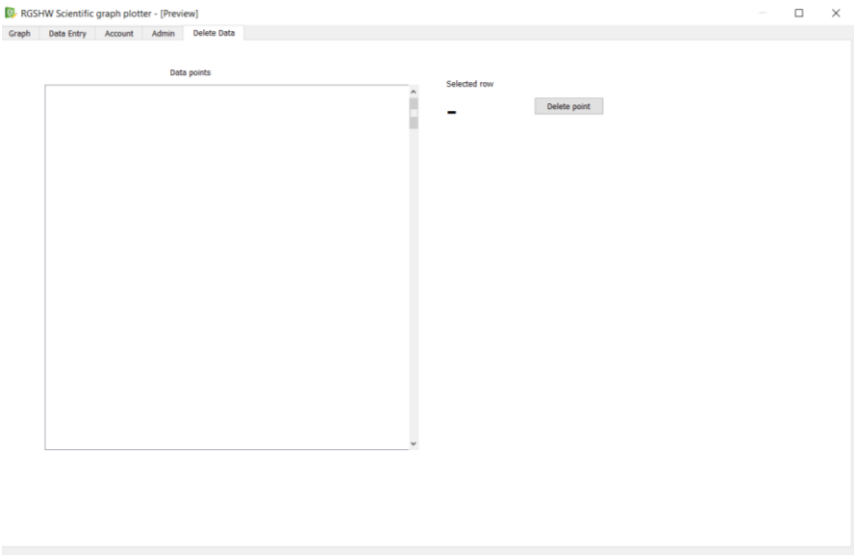
Admin page

This contains all of the information required for teachers to use the system. This is important as it allows teachers to not only create class functionality but also to view who has submitted points to the current graph out of who currently has permission to view the graph



1. This table shows all of the information about the user stored in the database about the user (Excluding their password) To teachers. This also displays how many points each user displayed has contributed to the currently selected graph. This is useful as it allows the teachers to set graph contribution as homework to their class, and then check if the student has done it. It also allows the teacher to check who has done work in lessons. This is large and centralised to make it easy to see each6
2. This allows the teacher to add the graph code and class code to link the two. This means that the graph can be linked to a whole class, saving time, and also ensuring that the teacher can see if all of the users in the class have contributed points. This saves a lot of time compared to getting every user to join the graph separately. This includes an explanation to avoid confusion in the case of the teacher.
3. This box links to the class creation screen. It is positioned with the teacher adding button with the same style as they both serve similar functions
4. The add a teacher button links to the teacher addition GUI file. This allows teachers to add other teachers. I chose to put this within the UI for teachers as it prevents students from attempting to gain teacher privileges.

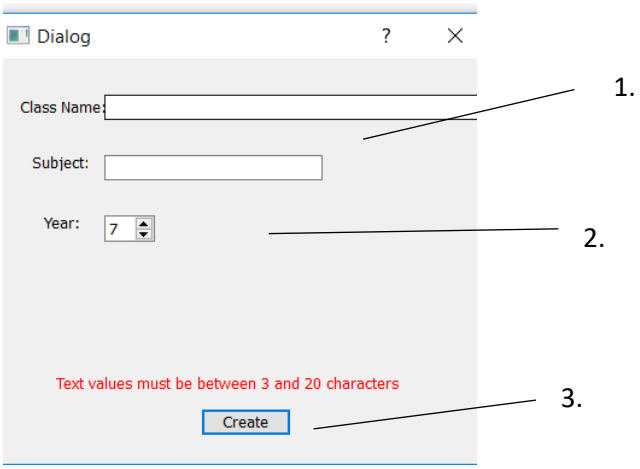
Data deletion screen



This screen is designed to provide the teacher with the option of deleting data points from students. This could be because the data point causes a bad line of best fit, or even is thought to be added to sabotage the results. The table view used to display the data points is elongated to provide the maximum number of points without scrolling, and takes up a large amount of the screen. I chose a new tab for this table view as it is important to provide plenty of space for it for easy viewing, The selected row is shown to attempt to make it more clear to the user which point is selected, and then the button next to it allows the teacher to delete the point. This can also be used for solely viewing the data points as well.

Class creation window

This window is designed for teachers to create a new class. It is in a separate window so that it can be focused on before normal operation is continued. This is important as classes are a core piece of functionality to the program, and creating them should happen frequently at the start of a school year.



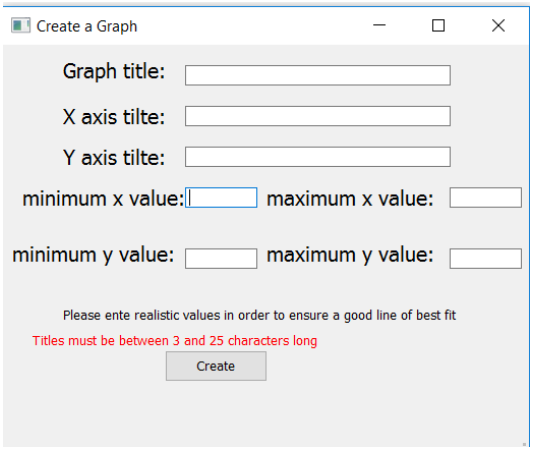
1. The string input boxes allow the teachers to select the names of the class and subject that they would like. I do not provide any drop-down menus for subjects as this would add very little functionality for a lot of development time. An attempt to do this may even limit

functionality and increase frustration as a teacher may want to be more descriptive than just the subject name in the subject box, however would have to create a whole new subject just for this. It would also require further storage as there would need to be another table just for subjects.

2. This year selection is chosen from a range of 7-13 this means that there is no need for extra validation as it is required to be an integer in this range. This ensures that no matter what the input is from the user it must be correct. This also explains the function of the year to the teacher as it may confuse them if they believe the input to be for date
3. The create button can be pressed and the class will be added to the database. This also triggers the validation method in order to ensure that values are as desired. If they are not the data is not submitted to the database and the error message is shown in red.

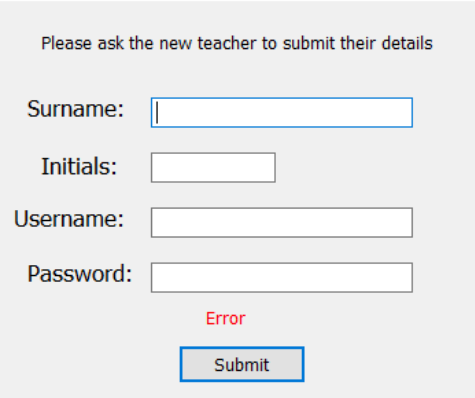
Graph creation window

Teachers use this window to produce new graphs. It contains a large amount of inputs and is therefore wider than the class creation screen. Maximum and minimum values for the same variable are put next to one another to try to denote the similarity.



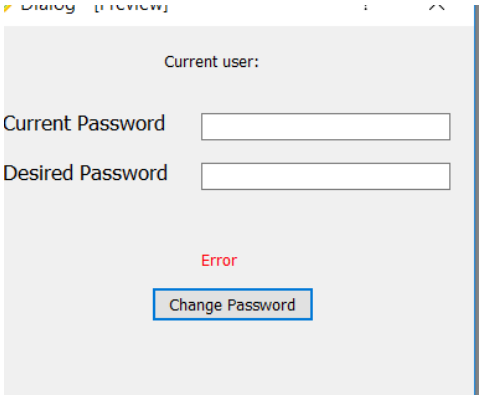
The titles are given in a column in order to show their similarity. As all of them are strings of similar size they are given the same length boxes and are relatively close to each other. The maximum and minimum values for x and y are shown together with spacing between them to group the two types of point visually. The text boxes are small to try to denote a number, using my standard for inputs. A reminder is placed at the bottom to enter realistic values, as large ranges make it much more difficult to plot an accurate line of best fit. Warning teachers about this avoids later confusion and therefore doesn't frustrate teachers. The error message is in the standard red for clarity.

Teacher addition



This window allows teachers to add other teachers. Surname and username use normal line edits; however, initials use a shorter box to denote the smaller input required. The password box will also automatically star the input so that the user's password cannot be seen by a colleague looking over their shoulder. The error message is in red as I use as standard and will display the required message for the mistake made in entry.

Password change



This window is used for changing the password for the current user. The text at the top will be changed to "current user:" and then the user's username. This means that the user does not accidentally try to change another person's password.

Pseudocode

Login pressed

```
username = getusernameboxvalue()

password = getpasswordboxvalue()

hashedpass = utf8hash(password)

 If Studentselected == true then

        data = SQLExecute("SELECT * FROM Students WHERE username = (username)")// This
        //should return a single list as each username is unique

        accesslevel = "student"
```

```
        databasepassword = data[2] // The 3rd database item contains the password in this case(see
        //below)
    Else
        data = SQLExecute("SELECT * FROM Teachers WHERE username =(username)")
        accesslevel = "staff"
        databasepassword = data[3] // The 4th database item contains the password in this
        //case(see below)
    Endif
    IF databasepassword == hashedpass THEN
        Login(username,accesslevel)
    Else
        Showloginfailedtext()
    Endif
```

Register pressed

When the register button is pressed each entered field needs to be validated so that it can be used. Most of the fields will only be used so that teachers can distinguish between users, however it is important to still maintain a standard for visuals.

```
    username = getproposedusername()
    password = getproposedpassword()
    forename = getproposedforename()
    surname = getproposedsurname()
    dobyear = getproposeddateyear()
    dobmonth = getproposeddatemonth()
    dobday = getproposeddateday()
    dob = getproposeddate()
    errortext = ""
    data = sqlexecute("select * from Students where username = ?;",(username,))
    if username.isalphanumeric() == False then
        errortext = "This username isn't alphanumeric"
    else if username.length() > 20 or username.length() >7 then
        errortext = "This username isn't between 7 and 20 characters long"
```

```
else if data != [] then
    errortext = 'This username is already in use'
else if password.length() >20 or password.length() <7 then
    errortext = "This password isn't between 7 and 20 characters long"
else if forename.isalphabetical() == False then
    errortext = “This forename isn’t alphabetical”
else if surname.isalphabetical() == False then
    errortext = “This surname isn’t alphabetical”
else if forename.length() >15 or forename.length() <2 then
    errortext = "This forename isn't between 2 and 15 characters long"
else if Surname.length() >15 or Surname.length() <2 then
    errortext = "This Surname isn't between 2 and 15 characters long"
else if dobyear < currentyear-19 or dobyear> currentyear -11 then
    errortext = “This date is either too early or too late”
end if

If errortext=”” then
    hashedpass = utf8hash(password)
    sqlexecute("INSERT INTO Students(Username,Password,Forename,Surname,DOB) Values
    (?,?,?,?,?);",(username,password,forename,surname,dob))
    showloginscreen()
    hideregisterscreen()
 else:
    ErrorLabel.SetText(errortext)
```

Setting up values for axis sketching:

Refer to statistics generation for the values that will be taken as parameters. The maximum and minimum values need to be adjusted for sketching such that the lines go through the point (0,0)

```
Function setupvalues(maxx,minx,maxy,miny)
    if (maxx<0) then
     maxx=0
```

```
end if
if (minx>0) then
  minx=0
end if
if (maxy<0) then
  maxy=0;
end if
if (miny>0) then
  miny=0
end if
return minx,maxx,miny,maxy
end function
```

plotting the axes

This function is designed to plot the axes of the graph on the 800x600px canvas the x and y ranges are taken from the function above

```
Function plotaxes(minx,maxx,miny,maxy)
    Xpositionofaxes = 100-(minimumofx/rangeofx)*800 // as 800px is the width of the javascript
    //window and there is a border of 100px it makes sense  that the position
    Ypositionofaxes = 100+(maxy/rangey)*600
    // this follows the same understanding, however as the margins are measured from the top
    //maximum values are used instead
    Drawlinebetween([Xpositionofaxes,100],[ Xpositionofaxes,700])
    // draws the vertical line using the x position calculated
    Drawlinebetween([100, Ypositionofaxes],[900, Ypositionofaxes])
    // does the same for the horizontal line
End function
```

Transferring a data point to a position value

This function takes a data point and then converts is to a position on the canvas

```
Function pointtographvalue(xvalue,yvalue,minx,rangex,maxy,rangey):
        Xpos = 100+((xvalue-minx)/rangex)*800
        Ypos = 100+((maxy-yvalue)/rangey)*600
        Cords = [xpos,ypos]
```

```
        Return coords
End function
```

Plot the point given

Takes a point given and plots it to the scale. It draws a line about that point

```
Function plotpoint(cords):
        Xongraph=coords[0]
        Yongraph=cords[1]
        Lineto((xongraph+4,yongraph+4))
        Lineto(xongraph-4,yongraph-4)
        Endline()
        Lineto(xongraph-4,yongraph+4)
        Lineto(xongraph+4,yongraph-4)
End function
```

Plot line of best fit

This algorithm plots the line of best fit based upon the polynomial equation given in the form [x^5coefficient,x^4coefficient,x^3coefficient,x^2coefficient,xcoefficient,constant]

```
 Function LineOfBestFitPlot(equation xvalue,yvalue,minx,rangex,maxy,rangey):
        flag="False" // the flag becomes true if the line was previously above or below the canvas,
//denoting that the line must be plotted from a position other than the previous one
        i = 0
        while(i<801)
                yvalue = 0
                xvalue = (i*rangex)/800 +minx // finds x value based upon canvas position
                //from 0-800
                J=0
                For j<7, j=j+1
                        Yvalue = yvalue + ((xvalue)^(5-j))*equation[j]
                End for
                // finds the y position for that x value using the equation
                if (yvalue<=maxy && yvalue>=miny) then
                        if (flag == "above") then
```

```
                lineTo(pointtographvalue(xvalue,yvalue,minx,rangex,maxy,rangey)[0],100)
                // draws the line from above the point if it was above before
                else if (flag == "below") then
                        lineTo(pointtographvalue(xvalue,yvalue)[0],700)
                        // draws the line from underneath the point if it was below before
                end if
        lineTo(pointtographvalue(xvalue,yvalue)[0],pointtographvalue(xvalue,yvalue)[1])
        // goes to the x and y value of the new point
                flag ="False"

                else if (yvalue>maxy) then
                        flag="above"
                        // if the line is above it sets the flag to above and doesn't plot it
                else
                         flag = "below"
                        // if the line is below it sets the flag to below and doesn't plot it
                end if
end function
```

Statistical Generation

Generates the statistics for the rest of the program

```
function statisticsgeneration(GraphID)
   xlist=  cursor.execute("SELECT XVal FROM DataPoints WHERE GraphID =?;",(GraphID,) //fetches Xs
   ylist= cursor.execute("SELECT YVal FROM DataPoints WHERE GraphID =?;",(GraphID,)) //fetches Ys
   sumx = 0
   sumxsquared=0
   for each in xlist
      sumx = sumx+each // calculates sum of X
      sumxsquared= sumxsquared+ each**2 // calculates sum of x^2
   sumy = 0
   for each in ylist
```

```
        sumy = sumy+float(each) // calculates sum of y
    sumxy=0
    for i in range(len(xlist))
        sumxy+= xlist[i]*ylist[i] // calculates the sum of each x value * y value
    n = len(xlist)
    minx = 100000000000000000000000
    maxx = -100000000000000000000000
    miny = 100000000000000000000000
    maxy = -100000000000000000000000
    points=[]
    for i in range(len(ylist))
        newrow=[xlist[i],ylist[i]]
        points.append(newrow)
        if xlist[i]<minx
            minx = xlist[i] //if the x value is the new minimum value the minimum value is replaced
        if xlist[i] > maxx
            maxx=xlist[i] if the x value is the new maximum value the maximum value is replaced
        if ylist[i] > maxy
            maxy = ylist[i] if the y value is the new maximum value the maximum value is replaced
        if ylist[i]< miny
            miny = ylist[i] if the y value is the new minimum value the minimum value is replaced
    return xlist,sumx,sumxsquared,sumy,sumxy,minx,maxx,miny,maxy,n,points ,ylist
end function
```

linear regression

calculates the coefficients for the linear regression and changes the global equation variable

uses the mathematical equation for the coefficients

```
function linearregression(sumx,sumy,sumxy,sumxsquared,n):
        Sxy = sumxy - ((sumx*sumy)/n)
        Sxx = sumxsquared - ((sumx**2)/n)
        b = Sxy / Sxx
        a = (sumy/n)-(b*(sumx/n))
```

```
    global currentrepresentation
    currentrepresentation = [0,0,0,0,b,a] // changes the values to the ones desired
end function
```

$$y = a + bx$$

Not surprisingly, this is the equation of a straight line!

Where a and b are calculated using the following formulae:

$$a = \bar{y} - b\bar{x} \quad \text{and} \quad b = \frac{S_{xy}}{S_{xx}}$$

$$\text{Where, } S_{xy} = \Sigma xy - \frac{\Sigma x \Sigma y}{n}$$

$$\text{And, } S_{xx} = \Sigma x^2 - \frac{(\Sigma x)^2}{n}$$

http://www.s-cool.co.uk/a-level/maths/bivariate-data/revise-it/regression

Polynomial regression

Calculates the polynomial values using the statistics calculated before.

```
Function polynomialregression()
    global equation
    alpha=0.1 // sets alpha which is the learning rate. This determines how quickly the algorithm
//goes at the expense of accuracy
    theta = [0,0,0,0,0,0] // the current values for the equation
    for i in range(10000)
        thetatemp=[0,0,0,0,0,0]
        total = 0
        hypothesis = 0
        for j in range(len(xlist))
            for k in range(len(theta))
                hypothesis += ((xlist[j]**(5-j))*theta[k])  // calculates the expected
//values of y based on the equation
            cost = (ylist[j]-hypothesis) //calculates the distance between this value and y
            for l in range(len(theta))
                thetatemp[l]=thetatemp[l]+(alpha*cost*(xlist[j]**(5-l)))
                // changes the equation based upon the distance
            End for
            End for
```

```
                theta = thetatemp
        end for
        // does 10,000 iterations of gradient descent, getting the real value closer to the expected
        //one
        equation=theta // sets the equation to the calculated one
end function
```

Class / graph code generation

This generates the unique class/ graph code to be used for students to join that class/graph

```
function codecreate()
        newcode = ""
        for i<11, i++
                newcode = newcode + string(randomnumordigit)
        end for
        // generates a 10 digit code
        newcode = string(newcode)
        tempfetchgraph  =cursor.execute("select count(GraphCode) from Graphs where
                                        GraphCode=?;",(newcode,)
 // fetches all records where the graph code is the same as the current code
        tempfetchclass =cur.execute("select count(ClassCode) from Classes where
                        ClassCode=?;",(newcode,))
// fetches all records where the class code is the same as the current code
        if string(tempfetchgraph== "()" and string(tempfetchclass)== "()" then
                return newcode
                // if none are found the result is returned
    else
       codecreate()
// otherwise the code is generated again
```

graph joining

When a code is entered, the database is checked for graphs with that code. This is the same with classes, except the class table is checked. This is within the graph window class so parts of the class are referenced

```
function addgraph(self)

        error = False

        trialcode = self.AddGraphEdit.gettext() //gets the text from the textbox

        AddGraphResult = cur.execute("select GraphID from Graphs where
GraphCode=?;",(trialcode,))

          If addgraph result == “()” then

                    Error = True

          End if

        if error == False then

           global currentuser // gets the current username

           userid=cur.execute("select StudentID from Students where Username=?;",(currentuser,))

           // gets the userid

           try

              existencechecker = cursor.execute("select PermissionID from StudentPermissions where
GraphID=? and studentID=?;",(AddGraphResult,userid,))

              self.NewGraphStatus.setText("You already have permission for this graph")

          // if a value is returned then there must already be a permission.

           except

              cur.execute("insert into
StudentPermissions(GraphID,StudentID)values(?,?);",(AddGraphResult,userid,))

//otherwise the student is given the permission

              database.commit()

              self.changetableview(("select Graphs.* from Graphs INNER JOIN StudentPermissions ON
Graphs.GraphID = StudentPermissions.GraphID where StudentPermissions.StudentID = ? order by
GraphID Desc;").replace("?",userid))

// changes the table view to show all new graphs

              self.AddGraphEdit.setText("")
```

```
                // sets the entered text to nothing
            self.NewGraphStatus.setText("Please Enter The Code For A New Graph")
                // removes error messages
End function
```

Class creation

This uses the classcreate class so therefore uses the objects associated with it.

```
Function classvalidation
        ClassNameString = string(self.ClassName.text())
        SubjectString = string(self.Subject.text())
        YearVal = integer(self.SpinYear.value())
      if ClassNameString.length()> 3 and ClassNameString.length()< 20 and SubjectString.length()> 3
and SubjectString.length()< 20 then
                NewClassCode = codecreate()
                // if all validation is met the code is generated
                cursor.execute("insert into Classes(ClassName,Year,Subject,Classcode) values
        (?,?,?,?);",(ClassNameString,YearVal,SubjectString,NewClassCode))
                databse.commit()
                self.ClassName.setText("")
                self.Subject.setText("")
                self.hide()
// changes all of the text to new values, and then closes the window
        else:
                self.Error.setText("Text values must be between 3 and 20 characters")
                // otherwise explains to the user the error
        End if
end function
```

Point entry validation

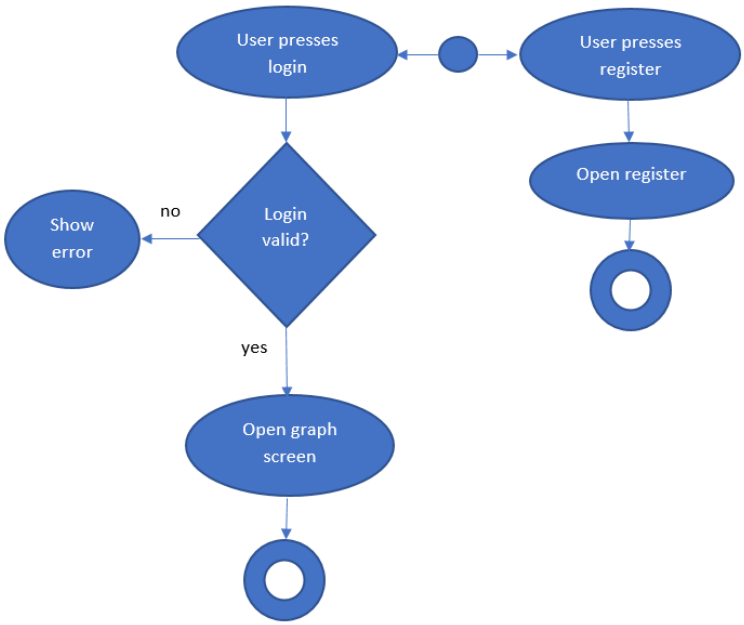
Validates the points entered. Uses the graph window class

```
Function pointentry()
```

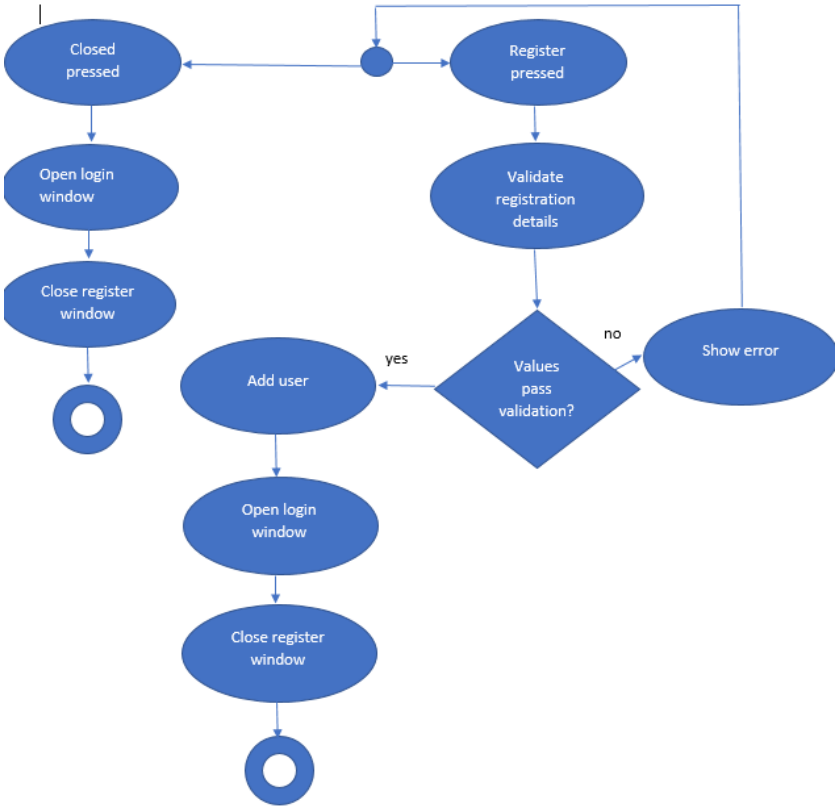
```
global userid
xpoint = self.XValueEdit.text()
ypoint = self.YValueEdit.text()
if self.currentgraph== -1 then
        self.PointEntryError.setText("No Graph is selected") // checks if the user is connected to a
        //graph
 else
       comparisondata=  cursor.execute("Select XValMin, XValMax, YValMin, YValMax from Graphs
where GraphID =?;",(self.currentgraph) //gets the values which the points need to be compared with
         comparisonlist = comparisondata.split(" ")
         try
            xpoint = float(xpoint)
            ypoint = float(ypoint)
            if xpoint >= comparisonlist[0] and xpoint <= comparisonlist[1] and ypoint >
comparisonlist[2] and ypoint < comparisonlist[3] then
                cursor.execute("INSERT INTO DataPoints(GraphID,XVal,YVal,UserID) VALUES
(?,?,?,?);",(self.currentgraph,xpoint,ypoint,userid))
                   // if the datapoint is within the ranges it is added to the graph
                database.commit()
                self.PointEntryError.setText("")
                self.YValueEdit.setText("")
                self.XValueEdit.setText("") // sets the text values
            else
                self.PointEntryError.setText("The values given are not within the range specified")
            end if
         except
            self.PointEntryError.setText("The entry is not a number")
// if something is incorrect the error messages are changed
end function
```

UML diagrams

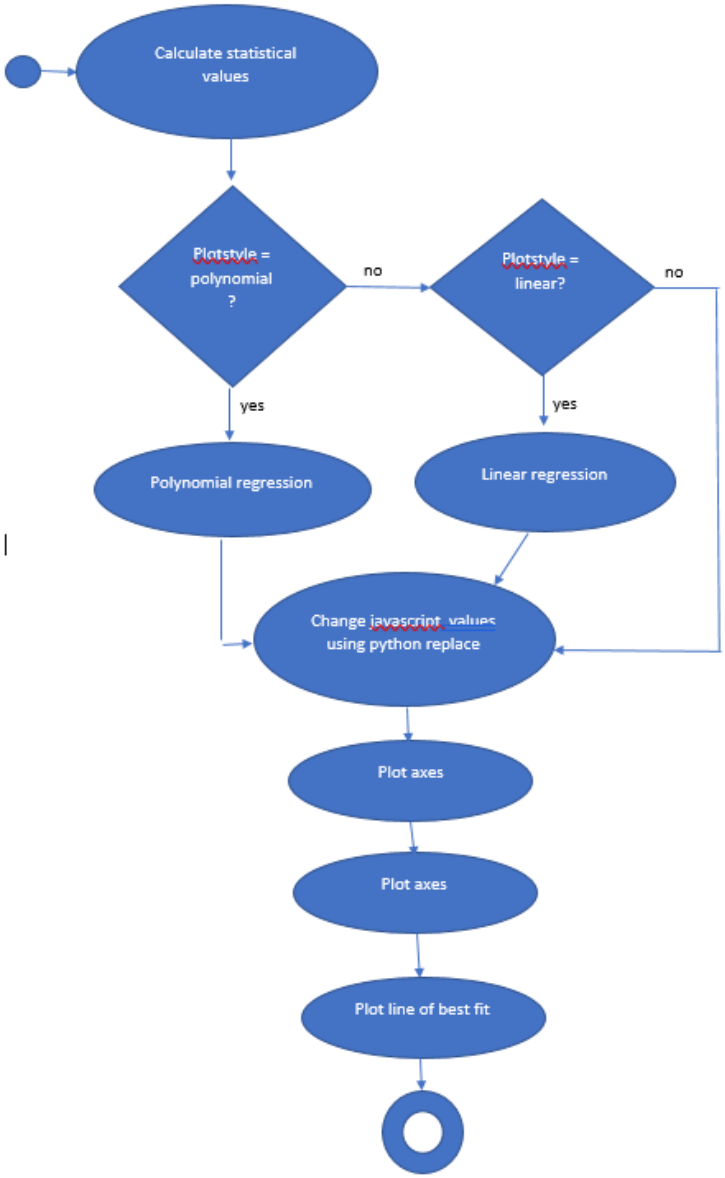
Login screen (window start)



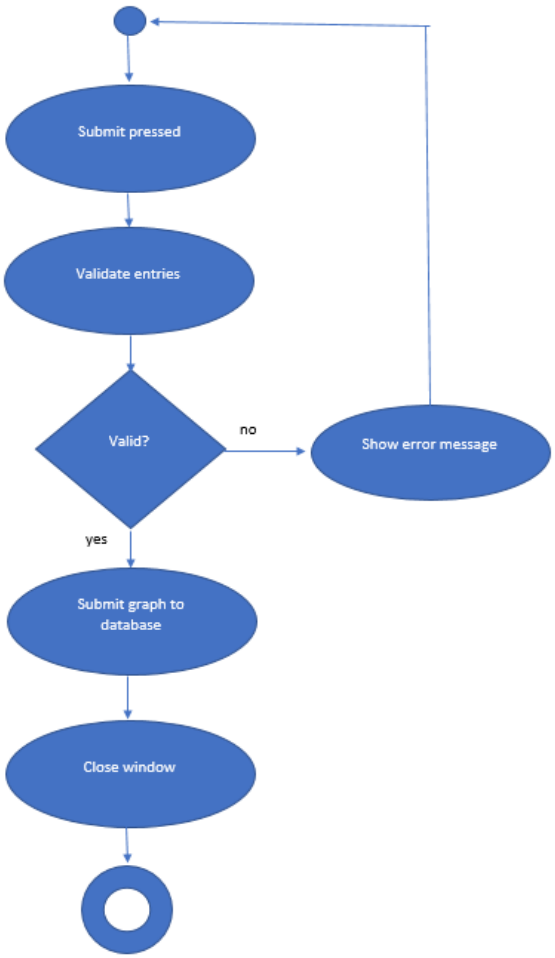
Register screen



Graph plotting



Graph creation

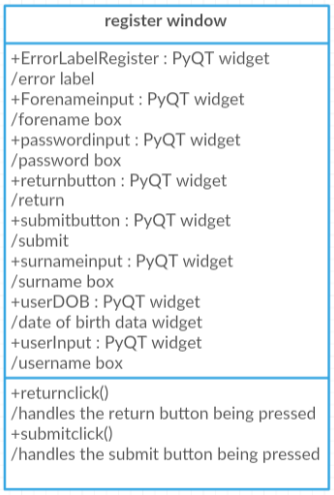


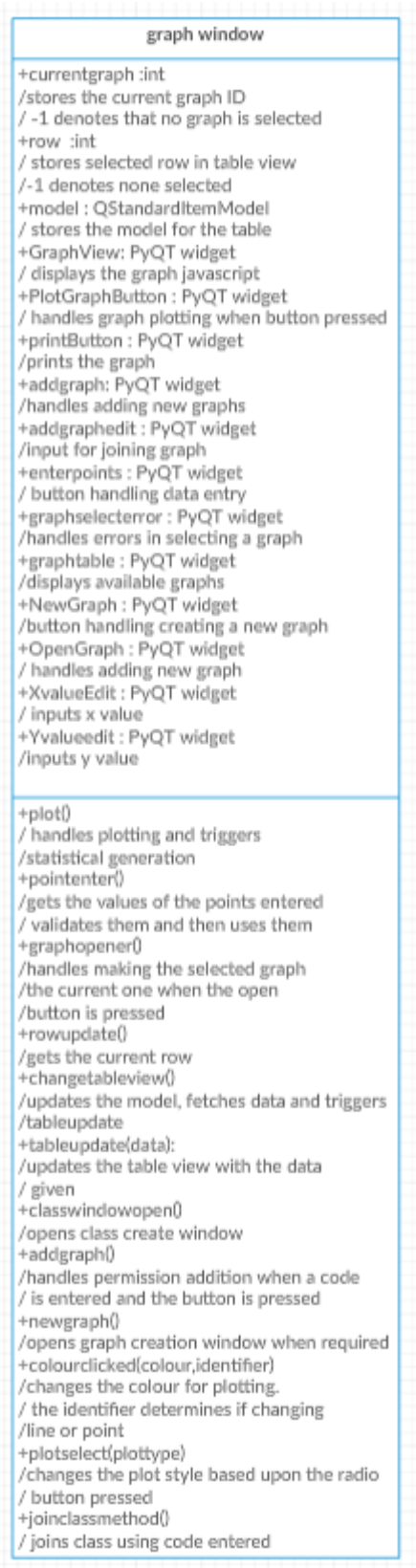
Create class , add teacher, and change password use the same structure, except with their respective data

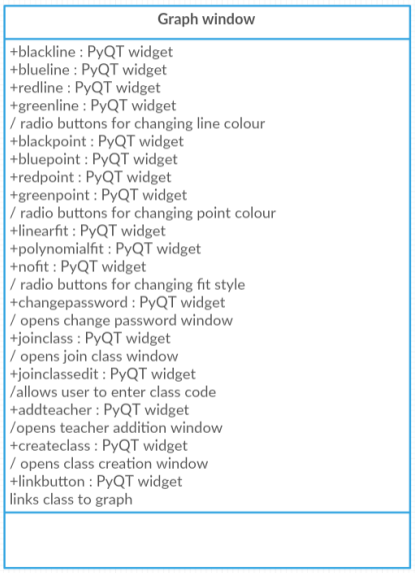
Class UML diagrams

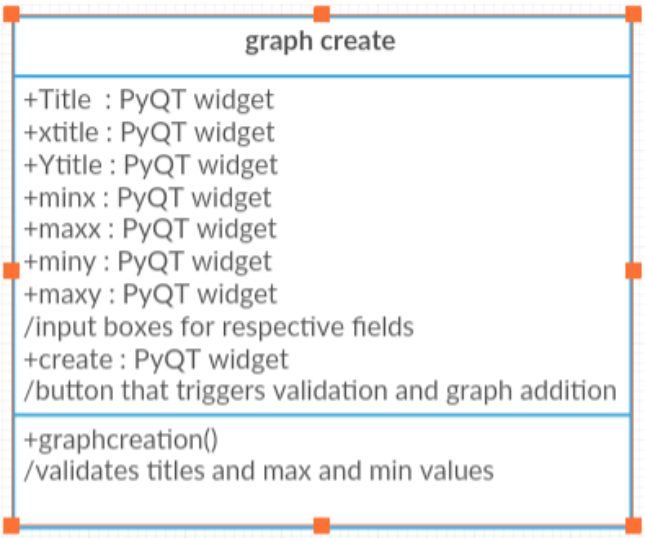
Each of my ui files has its own class to handle both the UI objects and any variables that need to be stored in the class and that should not be global

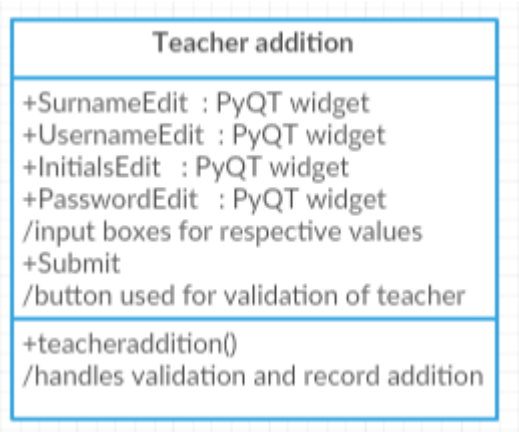
login window
- user :string +incorrect : PyQT widget / error message +loginbutton : PyQT widget +newaccountbutton : PyQT widget +password : PyQT widget /password box +username : PyQT widget /username box
+staffbuttonclicked() / when staff radio dial pressed +studentbuttonclicked() /when student radio dial pressed +loginclick() / handles login validation +registerclick() / closes screen and opens register screen

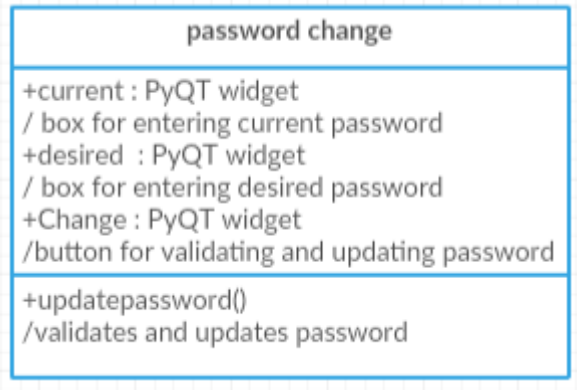

register window
+ErrorLabelRegister : PyQT widget
/error label
+Forenameinput : PyQT widget
/forename box
+passwordinput : PyQT widget
/password box
+returnbutton : PyQT widget
/return
+submitbutton : PyQT widget
/submit
+surnameinput : PyQT widget
/surname box
+userDOB : PyQT widget
/date of birth data widget
+userInput : PyQT widget
/username box
+returnclick()
/handles the return button being pressed
+submitclick()
/handles the submit button being pressed

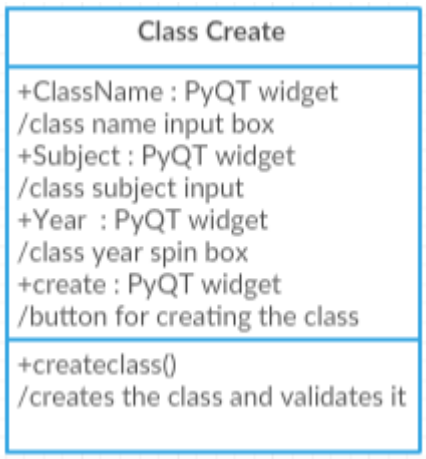

graph window
+currentgraph :int
/stores the current graph ID
/ -1 denotes that no graph is selected
+row :int
/ stores selected row in table view
/-1 denotes none selected
+model : QStandardItemModel
/ stores the model for the table
+GraphView: PyQT widget
/ displays the graph javascript
+PlotGraphButton : PyQT widget
/ handles graph plotting when button pressed
+printButton : PyQT widget
/prints the graph
+addgraph: PyQT widget
/handles adding new graphs
+addgraphedit : PyQT widget
/input for joining graph
+enterpoints : PyQT widget
/ button handling data entry
+graphselecterror : PyQT widget
/handles errors in selecting a graph
+graphtable : PyQT widget
/displays available graphs
+NewGraph : PyQT widget
/button handling creating a new graph
+OpenGraph : PyQT widget
/ handles adding new graph
+XvalueEdit : PyQT widget
/ inputs x value
+Yvalueedit : PyQT widget
/inputs y value
+plot()
/ handles plotting and triggers
/statistical generation
+pointenter()
/gets the values of the points entered
/ validates them and then uses them
+graphopener()
/handles making the selected graph
/the current one when the open
/button is pressed
+rowupdate()
/gets the current row
+changetableview()
/updates the model, fetches data and triggers
/tableupdate
+tableupdate(data):
/updates the table view with the data
/ given
+classwindowopen()
/opens class create window
+addgraph()
/handles permission addition when a code
/ is entered and the button is pressed
+newgraph()
/opens graph creation window when required
+colourclicked(colour,identifier)
/changes the colour for plotting.
/ the identifier determines if changing
/line or point
+plotselect(plottype)
/changes the plot style based upon the radio
/ button pressed
+joinclassmethod()
/ joins class using code entered


Graph window
+blackline : PyQT widget
+blueline : PyQT widget
+redline : PyQT widget
+greenline : PyQT widget
/ radio buttons for changing line colour
+blackpoint : PyQT widget
+bluepoint : PyQT widget
+redpoint : PyQT widget
+greenpoint : PyQT widget
/ radio buttons for changing point colour
+linearfit : PyQT widget
+polynomialfit : PyQT widget
+nofit : PyQT widget
/ radio buttons for changing fit style
+changepassword : PyQT widget
/ opens change password window
+joinclass : PyQT widget
/ opens join class window
+joinclassedit : PyQT widget
/allows user to enter class code
+addteacher : PyQT widget
/opens teacher addition window
+createclass : PyQT widget
/ opens class creation window
+linkbutton : PyQT widget
links class to graph

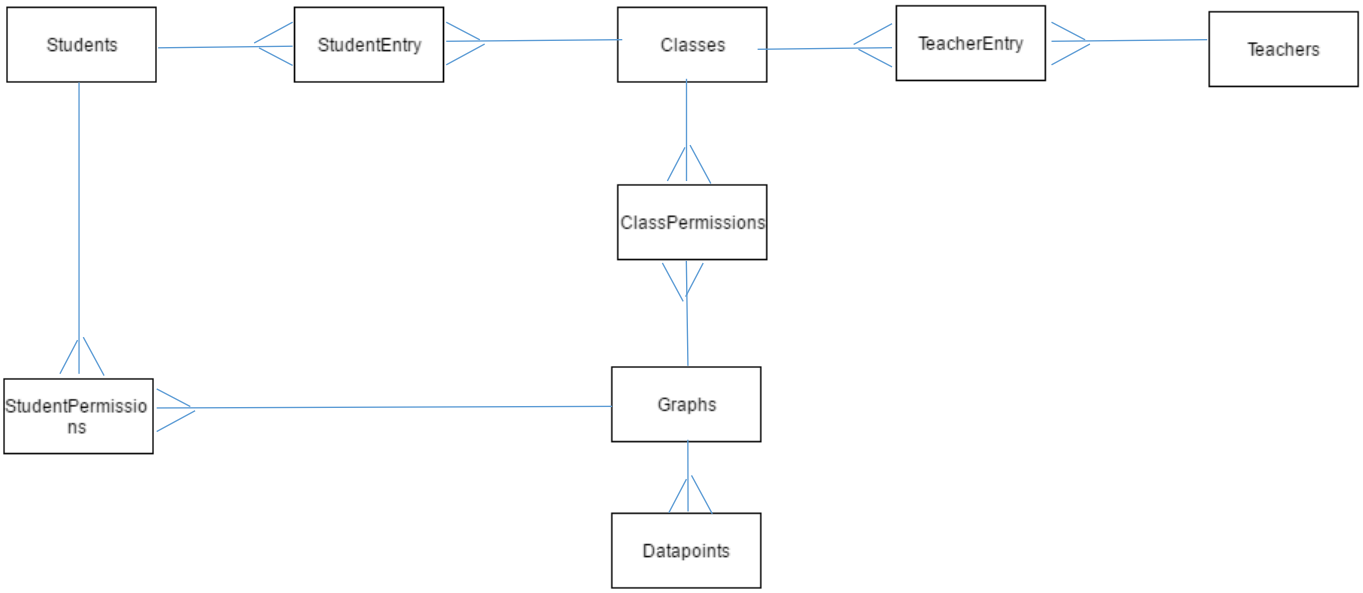

graph create
+Title : PyQT widget
+xtitle : PyQT widget
+Ytitle : PyQT widget
+minx : PyQT widget
+maxx : PyQT widget
+miny : PyQT widget
+maxy : PyQT widget
/input boxes for respective fields
+create : PyQT widget
/button that triggers validation and graph addition
+graphcreation()
/validates titles and max and min values

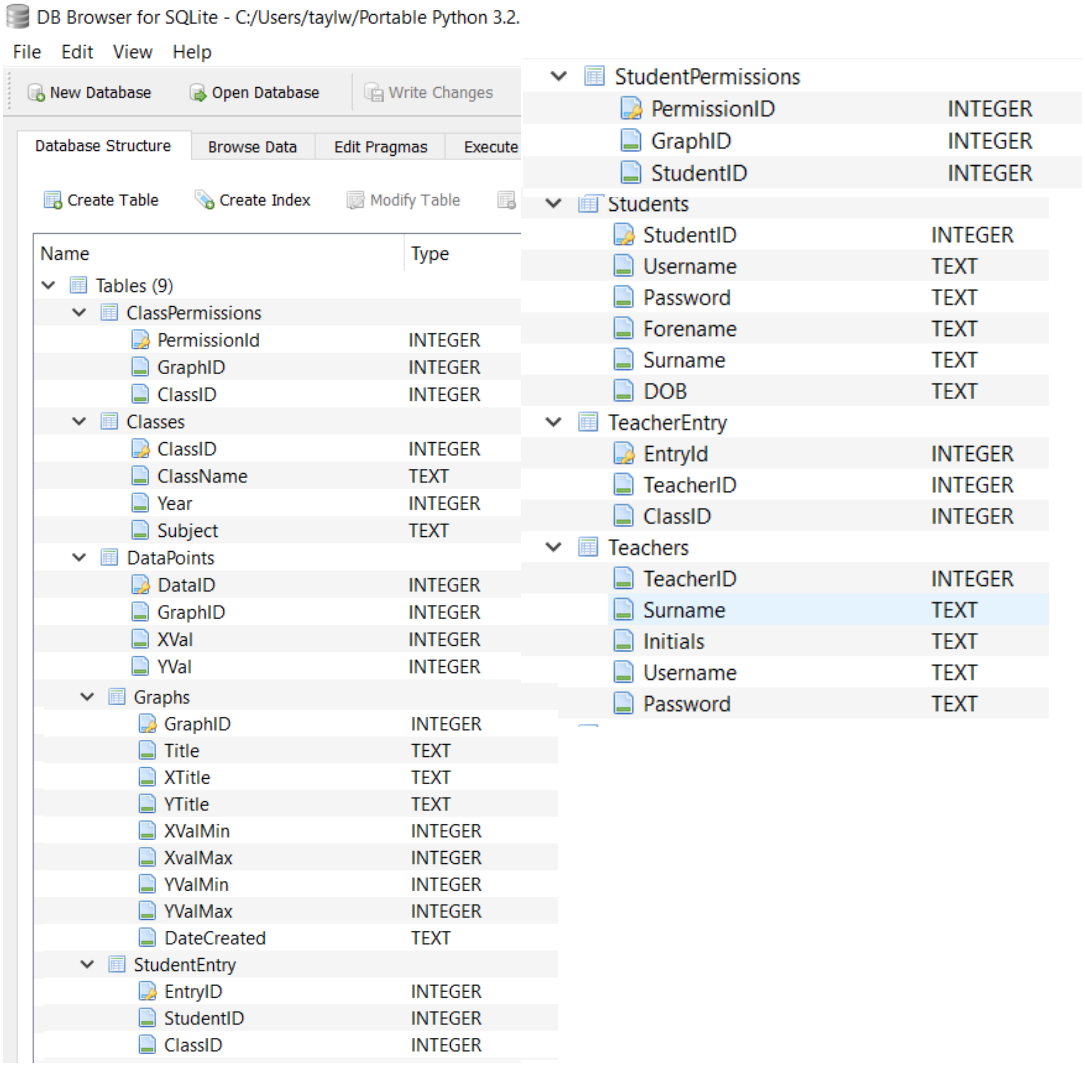

Teacher addition
+SurnameEdit : PyQT widget
+UsernameEdit : PyQT widget
+InitialsEdit : PyQT widget
+PasswordEdit : PyQT widget
/input boxes for respective values
+Submit
/button used for validation of teacher
+teacheraddition()
/handles validation and record addition


password change
+current : PyQT widget
/ box for entering current password
+desired : PyQT widget
/ box for entering desired password
+Change : PyQT widget
/button for validating and updating password
+updatepassword()
/validates and updates password



Database Structure





Database Structure Explained

Students

This contains logins and basic information about that student.

StudentID- I decided to create an individual studentID to act as the primary key for students in order to be used as a foreign key in the student entry table for adding students into classes.

Username- This should store alphanumeric values for the user's username which they use to login to the program.

Password- This contains a hashed version of the user's password to prevent obvious security flaws. This should be able to hold any character in order to allow for strong passwords.

DOB- The data of birth of the student. When entered it shall be in the form “YYYY-MM-DD” in order to allow sql sorting based on the date of birth of the student.

Forename- The forename of the student

Surname- The surname of the student

This then links to the student entry table using student ID as a foreign key

Student Entry

This table is used to avoid many to many relationships between students and classes. Initially I decided that students should only be able to join 1 class as I intended for the program to be only used for physics, however I decided to add this table in order to increase the scope of my program to allow it to be used in a multitude of classes. This also avoids a problem that may arise when students aren’t removed from groups as they enter new years.

EntryID- The primary key of this table, which allows each entry to be distinguished from each of the others.

StudentID- This acts as a foreign key to create a one to many relationship between Students and Student Entry.

ClassID- This foreign key creates a one to many relationship to the classes table which is described next.

Classes

This table contains information for a class or groups of people who may want to access a graph. This table is linked to Teacher entry in a one to many relationship as multiple teachers may teach one group of people. It is linked to Student entry in a many to one relationship and it is linked to class permissions in a one to many relationship.

classID- the primary key for classes.

ClassName-The unique class name for the class which a teacher will assign before creating it

Year- The school year which the class is for

Subject – The subject for which the class is for. This is was originally only going to be physics, however I introduced this field to accommodate for more broad use of the program

Teachers

This has a many to one relationship to teacher entries. This contains the necessary information about the teachers that is used in the program.

TeacherID- This is the unique teacher id that identifies the teacher that acts as a primary key. This is used in the teacher entries table to prevent a many to many relationsip between teachers and classes

Surname- The teachers surname

Initials – The initials of the teacher to be show to students

Username – The username of the teacher for login

Password – The hashed version of the teachers password for login.

TeacherEntry

This table is in a many to one relationship with classes and a many to one relationship with Teachers. It prevents a many to many relationship when entering multiple teachers into different classes.

EntryId- The unique primary key of the teacherentry

TeacherID- The foreign key to link to teachers

ClassID- The foreign key to the class table which links the teacher specified to the necessary class

Class permissions

This links many classes to one class permission and many class permissions to each graph. This allows multiple classes to be linked to the graph without a many to many relationship.

PermissionID- The primary key for the permission

GraphID- the foreign key to link to the graph

ClassID- the foreign key to the class which will be linked to a graph

Student Permissions

This works in much the same way as class permissions but allows individual students to be added to a graph using StudentID as a foreign key rather than ClassID

Graphs

This is the table which stores the basic information about each graph.

GraphID- The unique identifier for a graph

Title- The title for the graph which is shown at the top of the graph.

XTitle- the title for the X axis which will show below the x axis

Ytitle- the same as Xtitle but for the Y axis

XvalMin,YValMin,XValMax,YValMax – the lowest values for x and y and the highest values for x and y respectively allowed to be entered into the graph in order to prevent huge data which obscures the rest.

DateCreated-The date for when the graph was created.This shall be in the form YYYY-MM-DD in order to allow the program to sort by age so newest graphs a user is allowed to edit show up at the top

DataPoints

This table contains a set of data about each set of points in each graph. It is in a many to one relationship with Graphs.

DataID-The unique identifier for the data points

GraphID- The ID of the graph in which the points are needed.

XVal,Yval- the x and y values respectively of the data point.

Schedule

1. Create the database as shown above
2. Create the user interface as designed above
3. Import the necessary files for the database and GUI into python and write code to set them up
4. Create basic code for a login screen that checks inputs
5. Check that text inputs work and can be validated against strings
6. Link the login window to the register window in both directions
7. Store data inputted by the user in the registration screen
8. Validate the inputs against the test data
9. Introduce password hashing
10. Create code for entering valid data into database
11. add functionality for displaying an error message for invalid data
12. add functionality for swapping between staff and student logins
13. access database and check login credentials against entered ones (including hashed password)
14. show the main screen if the entered credentials are correct
15. add code for error message for invalid data
16. add functionality to radio buttons for colours
17. add error messages if the files are not there
18. allow users to only see the UI that they have access to.
19. Allow users to open the graph creation screen from the main window
20. Allow input into graph creation
21. Validate graph creation inputs
22. Create a code for joining the graph
23. Add graph to database if valid
24. Show error message if invalid
25. Load allowed graphs to the table view
26. Allow a user to enter a code for a graph
27. If the graph is valid give permission
28. Update the tale view if valid
29. Show error if invalid
30. Allow teachers to open class creation screen
31. Validate inputs for created class
32. If valid add the class to the database
33. If valid create a class code
34. Otherwise show error message
35. Allow the user to join the class using the code
36. Show graphs available to the user with said code
37. Allow user to select a graph from a table view
38. Allow users to add points to the graph
39. When plot is clicked, if linear is selected calculate linear regression values
40. Add polynomial regression for data points
41. create graph drawer in javascript

42. generate statistics required for graph drawer.
43. Plot graph based upon points entered
44. Change colours based upon radio dials
45. Add line of best fit based upon values provided
46. Allow teachers to add a class to a graph using the codes for each one
47. Allow teachers to see all graphs
48. Allow teachers to delete points on the selected graph
49. Allow the user to print the graph
50. Add value interpolation
51. Graph deletion
52. Password change

Test Plan

Success criteria	Tests	When	Explanation
70. Collect username for registration	Username too short i.e “a” username too long i.e. “aaaaaaaaaaaaaaaaaaaa” (21 characters) Username contains symbols “@”,”?”,”!”,”£”,”””” Username already in use “newusername” Then “newusername” Username that has not been used and is of the right length with no symbols “WilliamT99” Should be accepted	Once the register screen has been completed	The username box can be tested as soon as all the register screen has been created as it requires existence checks. These are not possible without entering data beforehand. It could be tested by entering directly into the database and comparing, however this could be achieved to the same level just by waiting
71. Collect password for registration	Password too short i.e “a” password is too long i.e. “aaaaaaaaaaaaaaaaaaaa”	As soon as password is implemented	The conditions can be checked immediately as no existence checks are required

	password of required length “williamspass” should be accepted		
3.Collect Forename for registration	Forename too short i.e. “a” forename too long i.e. “aaaaaaaaaaaaaaaa”(16 characters) forename is not letters “111”,”@@@”,”...” forename of required length with all letters “William” should be accepted	As soon as forename is implemented	The conditions can be checked immediately as no existence checks are required and there are no prerequisites
4. Collect Surname for registration	surname too short i.e. “a” surname too long i.e. “aaaaaaaaaaaaaaaa”(16 characters) surname is not letters “1”,”@”,”.” Surname of required length with all letters “Taylor” should be accepted	As soon as surname is implemented	The conditions can be checked immediately as no existence checks are required and there are no prerequisites
5. Input date of birth	Try to press the buttons quickly to try to input invalid values. Try date 20 years ago “01/01/1997” Try date this year “01/01/2017” Date of “01/01/1999” should be accepted	As soon as date of birth is implemented	The conditions can be checked immediately as no existence checks are required and there are no prerequisites
6,7,8,9	Tested above		
10. Provide an error message should registration data be invalid.	Values above should be tested again and respective error message should be given	As soon as errors are implemented	The values can be tested again, as long as the previous features have all been added then they can be tested against
11. If validation is successful, the student account should be added to	The correct values described above should be entered and the user should be added to the database. The window	Once everything above is added	Once the rest of the registration screen has been added the final

the database with a unique primary key	should be closed and the database should be checked for the new record Spamming submit button shouldn't enter multiple records.		step can be completed.
12. Allow the user to leave the registration screen should they want to stop creating an account	Press the return button. This should take the user back to the previous screen. The button should be spammed to no adverse effect	As soon as the return button is created	
13. Provide an option for student or teacher login	Pressing the radio button on the login screen should change the value of the permission variable. This should be tested with a print command	As soon as the radio's buttons' functions are added	
14. Hash the password entered by the user	Check that a password entered creates the correct hash. This can be done by printing the hashed password when entering "testpassword" And checking it against the value of an online hasher using the same algorithm	As soon as the hash algorithm has been added.	
15. Check the username in the database that has been entered for a password	Print the return of the sql query and check it against the username entered "WilliamT99" Should return a value	As soon as registration testing is done	Registration testing must be done to test against this value
16. Compare the hashed password in the database against the hash of the password provided by the user	Enter "WilliamT99" and "williamspass" And the password should be considered correct and not return an error	As soon as login is finished	
17. If the user enters the correct details, change the user to the username entered and open the main window	Enter "WilliamT99" and "williamspass" The next screen should be shown and a printed version of the variable currentuser should be "WilliamT99"	As soon as login is finished	

	Spamming login should produce no adverse effects.		
18. If the password entered was incorrect an error message should be provided	Enter “WilliamT99” and “wrongpassword” The error message should show	As soon as login is finished	
19. Allow teachers to create classes	Class create button should not produce adverse effects when spammed and should open the class creation window	As soon as the class create button is implemented.	
20. Allow the facility for teachers to input class name	Check that the class name is between 4 and 19 characters “aaa”, “aaaaaaaaaaaaaaaaaaaa” (20 characters) Input correct value “year 13 2017”	As soon as the class name is added	
21. Allow the teacher to input a subject name	Check that the subject name is between 4 and 19 characters “aaa”, “aaaaaaaaaaaaaaaaaaaa” (20 characters) “Physics” should provide no incorrect values	As soon as implemented	
22,23. Validate that the year is between 7 and 13	The integer box widgets should validate for me, but test for any unintended flaws when spamming buttons	Right away	
24. Generate a code for the class	Test code generation function to see that it creates a new code by printing the output.	At any time	the function is independent so can be tested at any point after coding
25. Check that the code is unique	Test that codes are different from ones in current classes or graphs	Once class and graph creation is complete and data has been entered	The program checks against currently created classes and graphs so these must be implemented
26. Add the class to the database using the fields that were entered	Check with previous correct values – “year 13 2017”, “Physics”, “13”	Once the rest is implemented	

27,28,29 Allow students to enter the code of a class	Enter an invalid code (check database first as codes are random). This should produce an error message. Entering a valid code should add a permission between the current user and the graph in the database	Once class create is implemented	Class must be created to test current codes.
30. Allow graphs to be created by teachers	Ensure that the new graph button opens the graph window and does not produce adverse effects when spammed.	As soon as implemented	
31. Provide the option to allow all members of a class to view the graph	Add class link to graph. Then log in as a user in said class. If the user sees the graph, then the test is a success	After class creation, class codes, and class permissions are added	These are requisites of the function
32. Generate a unique code to join the graph	Same function as 24, ensure that code is added to graph's record in DBMS	After code function is added	
33,34,35: display graph when code is entered	Check that entering a valid graph code adds a permission in the database, and then check that it gets displayed. Check that incorrect codes do not cause unanticipated errors	After graph codes are implemented and the graph view updates	
36.Allow the user to select the graph by clicking on it	Use variable watch to see if the currentgraph variable changes when the user clicks. Pressing open graph when the program starts and nothing is clicked should produce no error, and should display the red text message	Once graph codes and graph permissions are added	
37. Display to the user which graph is selected	The text above the graph should change when a new graph is loaded	Right away	
38. Allow students to add points to the graph	Enter value below range e.g. "-1", should be rejected. Value below range e.g. "20000" should be rejected, strings should be rejected i.e "a"	Once point entry, graph create, and graph loading are done	

39. Plot the x axis with correct scale	Compare calculated percentage of x values on right side with visual representation	Check immediately for hard coded, and then later when values are inputted from python	This seems to be the only feasible way. The algorithm is derived mathematically however there is the possibility of syntax errors.
40. Plot the y axis with correct scale	Compare calculated percentage of y values on top of graph with visual representation	Check immediately for hard coded, and then later when values are inputted from python	This seems to be the only feasible way. The algorithm is derived mathematically however there is the possibility of syntax errors.
41. Add X title to the graph visualisation	Check that x title is correct and in reasonable place	Check immediately for hard coded, and then later when values are inputted from python	
42. Add Y title to the graph visualisation	Check that y title is correct and in reasonable place	Check immediately for hard coded, and then later when values are inputted from python	
43. Add main title to the graph visualisation	Check that main title is correct and in reasonable place	Check immediately for hard coded, and then later when values are inputted from python	
44. Allow the user to select no line of best fit	Use variable watch to check that the global variable changes to “none”	Right away	
45. Allow the user to select linear line of best fit	Use variable watch to check that the global variable changes to “linear”	Right away	
46. Allow the user to select polynomial line of best fit	Use variable watch to check that the global variable changes to “polynomial”	Right away	
47. Allow the user to select line of best fit colour from a predetermined selection	Use variable watch to check that the global variable changes to respective line colour, and can change multiple times	Right away	

48. Allow the user to select point colour from a predetermined selection	Use variable watch to check that the global variable changes to respective point colour, and can change multiple times	Right away	
49. Plot points onto correct point based upon the scale	Test multiple values at once hard coded into the javascript, then test again when values are transferred from python	Check immediately for hard coded, and then later when values are inputted from python	
50. Plot points with the correct colour as selected by the user	Check that colours reflect a change in the variable	Check immediately for hard coded, and then later when values are inputted from python	
51. Display the line of best fit based upon the user selected colour.	Check that line colours reflect a change in the variable	Check immediately for hard coded, and then later when values are inputted from python	
52. Calculate the statistical value of the gradient and intercept for a linear line of best fit	Check that the values calculated by the function are in line with online calculators	As soon as graphs are fully implemented. Initially check with hard coded values. Then attempt with graph codes.	Graphs must be fully functional before the graph code can be tested, however the function itself only relies on statistical analysis for hard coded values
53. Use these values to calculate the y values for each x value within the range	This is tested within the graph plotting function. Once the line of best fit is showing correctly this value can be transferred to the javascript. Check that the line of best fit looks correct.	As soon as plotting points and this function is added.	
54. Not plot a part of the line if it is not within the y axis.	Check graph visually to see if the point is shown if it is out of range.	As soon as plotting points is added	
55,57. Calculate the polynomial values for the line of best fit when polynomial is selected	Input hard values of linear line such as “[0.1,0.1], [0.2,0.2] … [0.5,0.5]” Afterwards use graph for data. Check polynomial	Can be done as soon as statistics are implemented.	This requires the statistics from the graph e.g. number of points

	lines against values from websites.		
56.“Regularise” the values in order to ensure that the line of best fit isn’t too biased.	Check that values seem to fit the points better on graph	As soon as I am confident in my polynomial regression outputs	This is hard to test non-visually, therefore I need to know that the polynomial regression display is correct
58. If the calculated y value is out of range the value shouldn’t be plotted.	In the same way as 54	As soon as 56 is completed.	
59.60. Calculate the mean value of the data points And display it	Should be correct after statistical generation is done. Test using calculator and checking against small amounts of data. Afterwards ensure that the data is outputted	As soon as statistical calculation is done.	
61,62. Calculate the number of points on the current graph and display it	Manually count points on the current graph in the database. If it matches the displayed value for multiple cases, then the algorithm must be correct	As soon as statistical calculation is done.	
63,64. Allow a user to change their password and validate it against requirements	Check that incorrect password produces an error. Check that correct password does not for multiple users. Check that the new password follows the same validation as before by doing the same tests described in 2. Then check that the hashed password is added to the database	Right away	
65. Display the equation of the line of best fit provided	Check line of best fit in variable watch and then compare to the value displayed. This will be known true from previous tests	As soon as all regression types are completed and display is added	
66. Interpolate X values given	Test that the data will produce an error if they are not numbers. E.g. “a”,”*”. Test with data		

	outside of the range e.g. "-10000000" and "100000000000". Try with values within range. If they result in an error then this must be fixed, otherwise the functionality is correct		

Extra OOP pseudocode

I have created some OOP pseudocode in addition to my pseudocode. The class with anything more than trivial pseudocode is the main window.

Main Window

Class mainwindow

```
Public currentgraph // current graph selected by user

Public row // current row clicked on in graph table

PlotGraphButton.clicked(Call plot())

EnterPoints.clicked(call pointentry())

AddGraph.clicked(call addgraph())

Public procedure new()

    Currentgraph = -1

    Row = -1 // sets to -1 to denote none selected

End procedure

Public procedure plot()

    //as described in main pseudocode

End procedure

Public procedure pointentry()

    // as described in main pseudocode

End procedure

Public procedure addgraph()

    //as described in main pseudocode for joining graph

End procedure
```

Development

Version 0.1

What the aim is:

To set up the login screen UI and attempt to read from login credentials and check that validation works. I also check to see that the ui files are loadable and that I can import pyqt and sqlite.

```
from PyQt4 import QtCore, QtGui, uic
import sys
import sqlite3 as lite
# con and cur for sql lite
tempuser = "user"
temppass = "pass"
loginclass = uic.loadUiType("loginwindow.ui")[0]

class LoginWindowClass(QtGui.QMainWindow,loginclass):
    def __init__(self,parent=None):
        QtGui.QMainWindow.__init__(self,parent)
        self.setupUi(self)
        self.Incorrect.hide()
        self.LoginButton.clicked.connect(self.loginclick)

    def loginclick(self,curr):
        username=self.Username.text()
        password=self.Password.text()
        if username == tempuser and password == temppass:
            print("login successful")
            loginscreen.show()
        else:
            self.Incorrect.show()

loginapp = QtGui.QApplication(sys.argv)
loginscreen = LoginWindowClass(None)
loginscreen.show()
loginapp.exec()
```

Loads pyqt for handling the UI, system for handling the loading of the files for the UI, and sqlite for communication with and editing of the database

Sets a temporary password just for validation against the inputs

Loads the loginwindow in order to show it to the user

Creates a class for the login window with its own method

The constructor method for the loginwindow class. It calls the method “setupUi” creates the ui and makes it visible

1

2

3

4

5

6

explanation

1 – Hides the text box which displays the error message when the object is created so that the user knows if they have made an error or not

2 – When the button is pressed it calls the method “loginclick” which has the main functionality of the button press

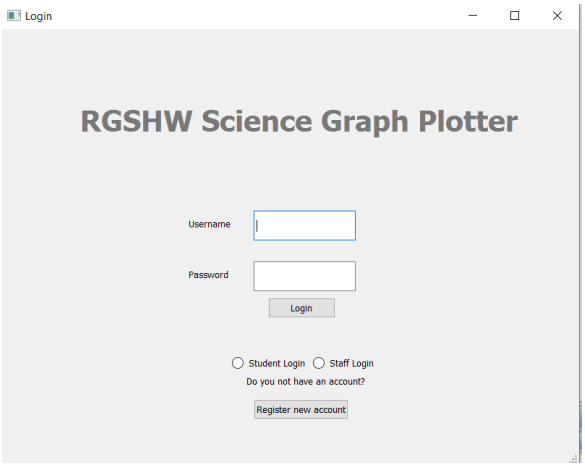
3- Defines the method of the loginwindow class “loginclick”. If the button is pressed this method’s contents are called

4- The variables “username” and “password” are set to the values that have been typed into the text edits in the pyqt GUI. This can be used to validate that the values are the same as the temporary values used for testing.

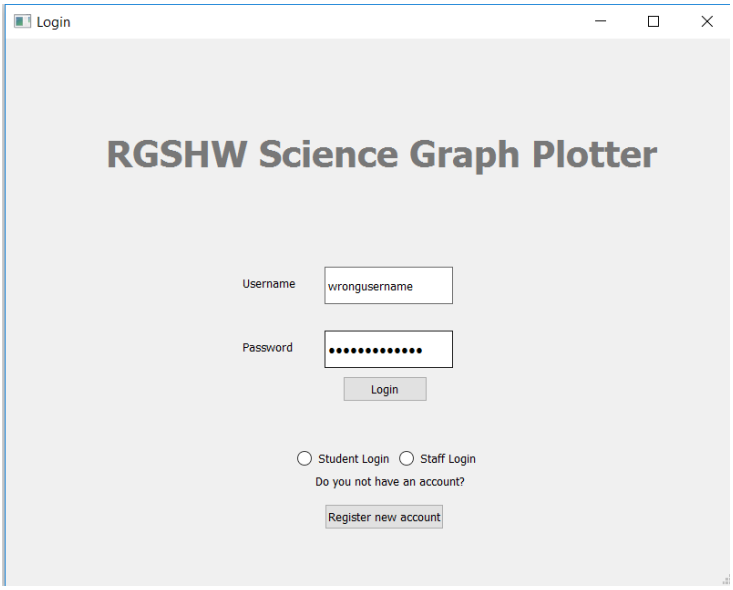
5- If the username and password entered by the user are the same as the temporary values used for testing, then “login successful” is printed in the console for the sake of the developer. This will later be used to load data from the current user from the sql server. If these values are incorrect then the incorrect message is shown

6- The login app is defined as the pyqt application which contains all of the UI files. The loginscreen object is created based upon the class previously defined. This screen is then showed and the application is then executed so that all of the UI files are opened.

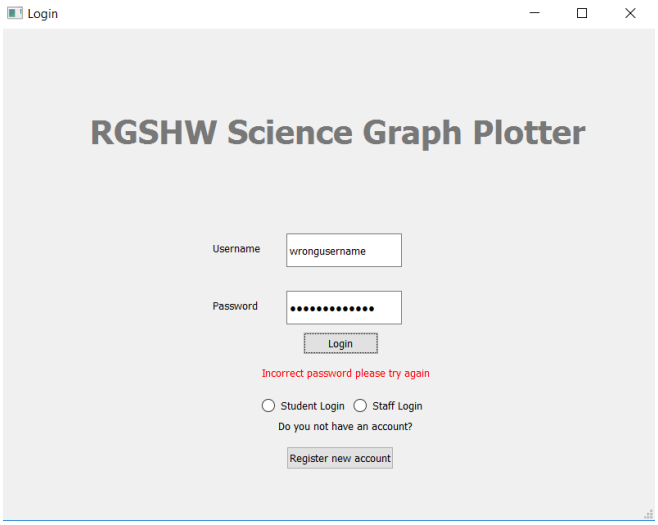
Version 0.1 testing



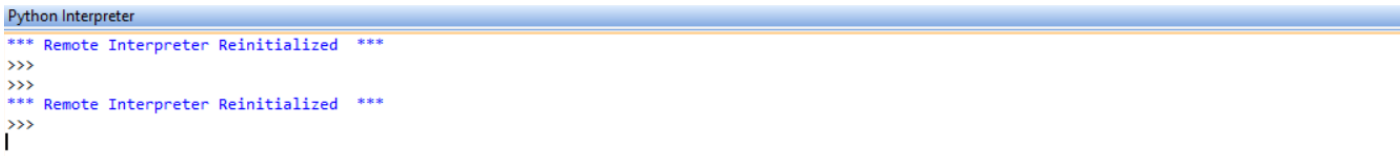
When the program is run the ui file is opened, showing that the loginclass , the file loading, and the pyQT is working.



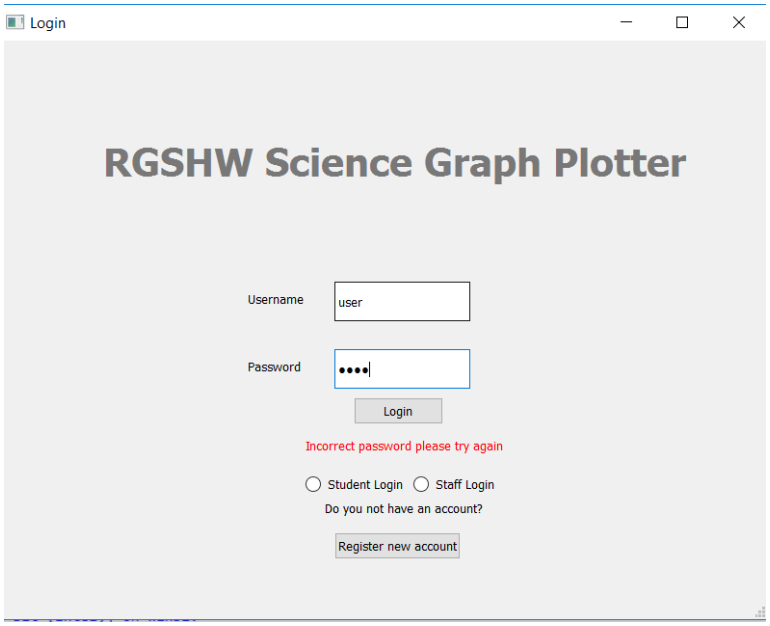
The values “Wrongusername” and “wrongpassword” are entered into the username and password boxes respectively.



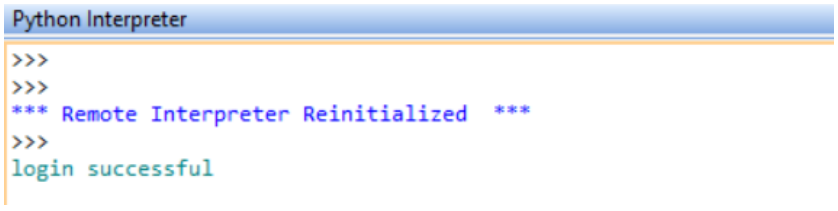
As “wrongusername” is not the same as “user” and “wrongpassword” is not the same as “pass” it means that the if statement is false, showing the error message “Incorrect password please try again”. This is as was expected meaning that this test has worked correctly.



The console does not print any value, meaning that the print must not have been executed.



The values “user” and “pass” are entered into the UI. The “incorrect password please try again” message is still showing from last test. These are the correct username and password values so I expect the print to be shown, reassuring me that the values have been read correctly.



As you can see the code has been run for when the login values are the same as the temporary values. This shows that the values are being read from the PyQT line edits. This works so therefore can be used in later iterations of the program.

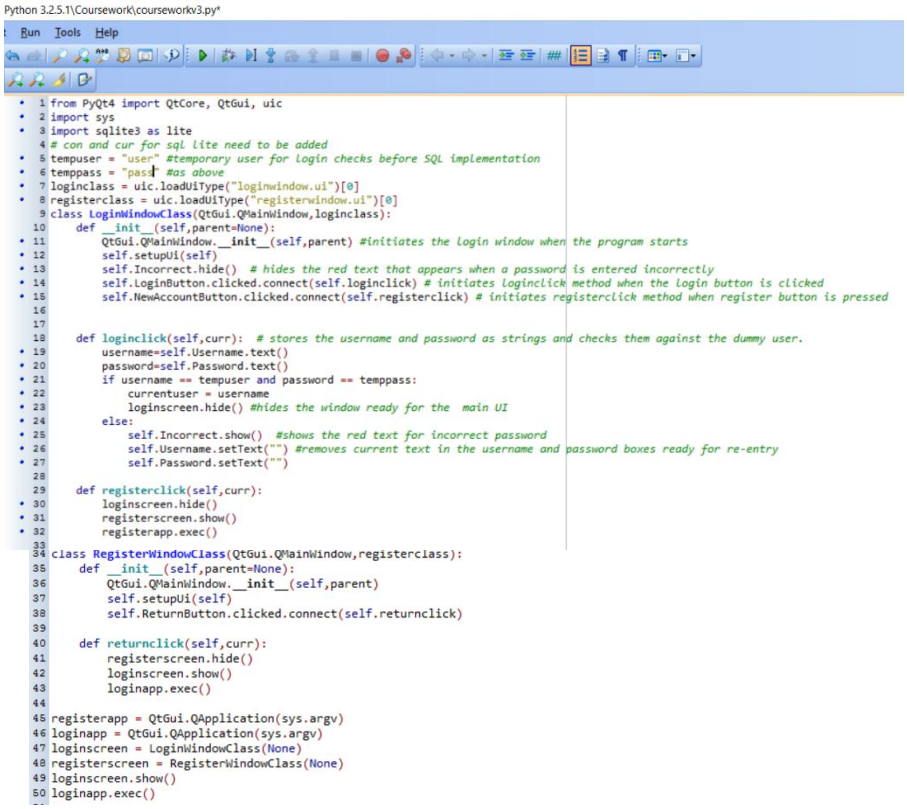
Version 0.2

The aim of this iteration:

To open the register window should the user presses the button.

To hide the login window when the value is correct.

To return to the login screen when the return button is pressed




```
1  from PyQt4 import QtCore, QtGui, uic
2  import sys
3  import sqlite3 as lite
4  # con and cur for sql lite need to be added
5  tempuser = "user" #temporary user for login checks before SQL implementation
6  temppass = "pass" #as above
7  loginclass = uic.loadUiType("loginwindow.ui")[0]
8  registerclass = uic.loadUiType("registerwindow.ui")[0]
9  class LoginWindowClass(QtGui.QMainWindow,loginclass):
10     def __init__(self,parent=None):
11         QtGui.QMainWindow.__init__(self,parent) #initiates the login window when the program starts
12         self.setupUi(self)
13         self.Incorrect.hide()  # hides the red text that appears when a password is entered incorrectly
14         self.LoginButton.clicked.connect(self.loginclick) # initiates loginclick method when the login button is clicked
15         self.NewAccountButton.clicked.connect(self.registerclick) # initiates registerclick method when register button is pressed
16
17
18     def loginclick(self,curr):  # stores the username and password as strings and checks them against the dummy user.
19         username=self.Username.text()
20         password=self.Password.text()
21         if username == tempuser and password == temppass:
22             currentuser = username
23             loginscreen.hide() #hides the window ready for the  main UI
24         else:
25             self.Incorrect.show()  #shows the red text for incorrect password
26             self.Username.setText("") #removes current text in the username and password boxes ready for re-entry
27             self.Password.setText("")
28
29     def registerclick(self,curr):
30         loginscreen.hide()
31         registerscreen.show()
32         registerapp.exec()
33
34 class RegisterWindowClass(QtGui.QMainWindow,registerclass):
35     def __init__(self,parent=None):
36         QtGui.QMainWindow.__init__(self,parent)
37         self.setupUi(self)
38         self.ReturnButton.clicked.connect(self.returnclick)
39
40     def returnclick(self,curr):
41         registerscreen.hide()
42         loginscreen.show()
43         loginapp.exec()
44
45 registerapp = QtGui.QApplication(sys.argv)
46 loginapp = QtGui.QApplication(sys.argv)
47 loginscreen = LoginWindowClass(None)
48 registerscreen = RegisterWindowClass(None)
49 loginscreen.show()
50 loginapp.exec()
```

explanation

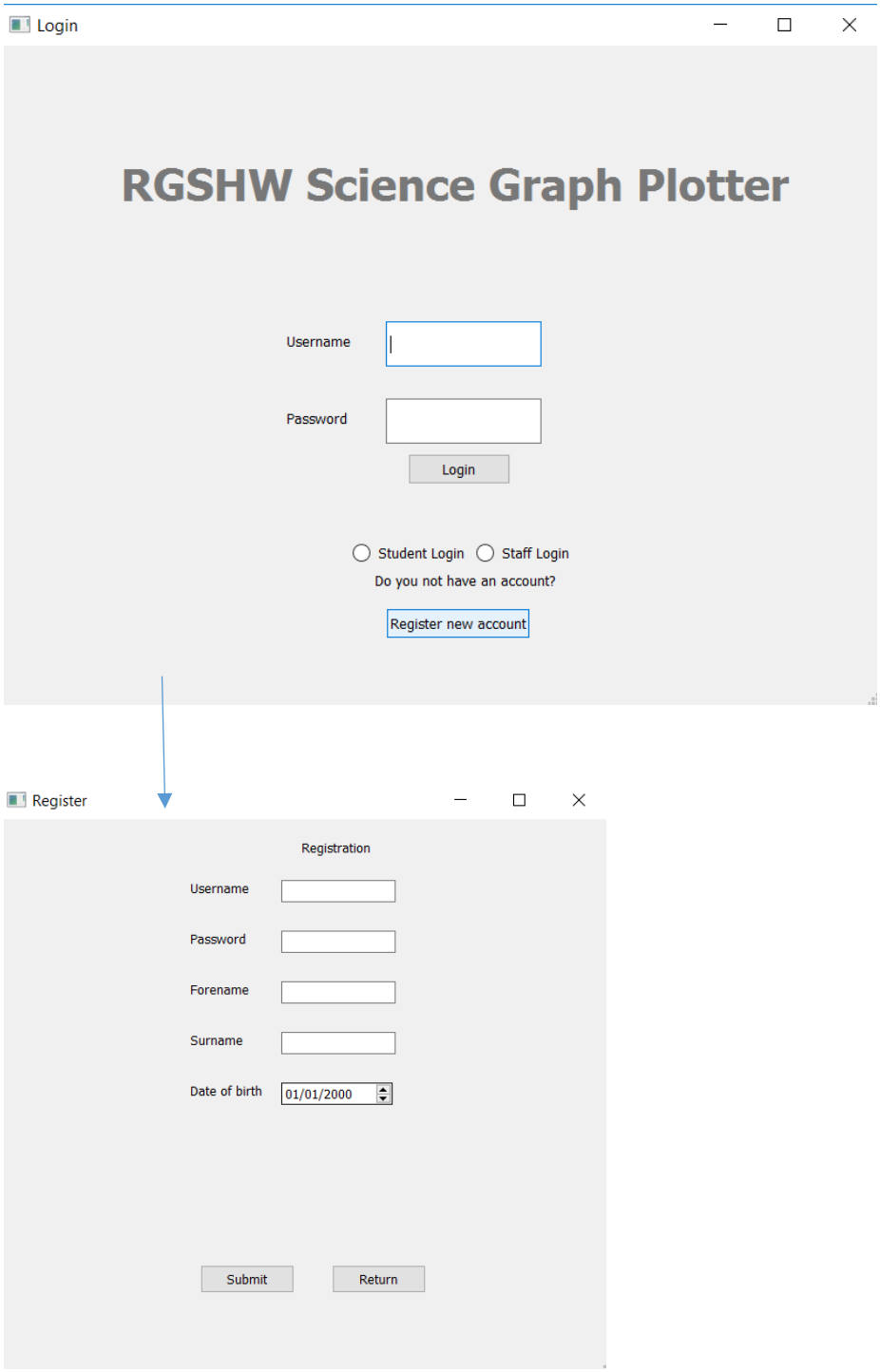
Line 29 – The method for opening the register window when “register” is clicked. This opens the register window object, and allows the user to see the window.

Line 34-38 - Defines the class “RegisterWindowClass”. When the object is constructed it sets up the UI, and then calls the method “ReturnButton” should the button to go back to the login screen is pressed.

Line 40-43 This is the definition of the method “returnclick”. When this button is pressed the register screen is hidden so it cannot be seen and then the loginscreen is shown again.

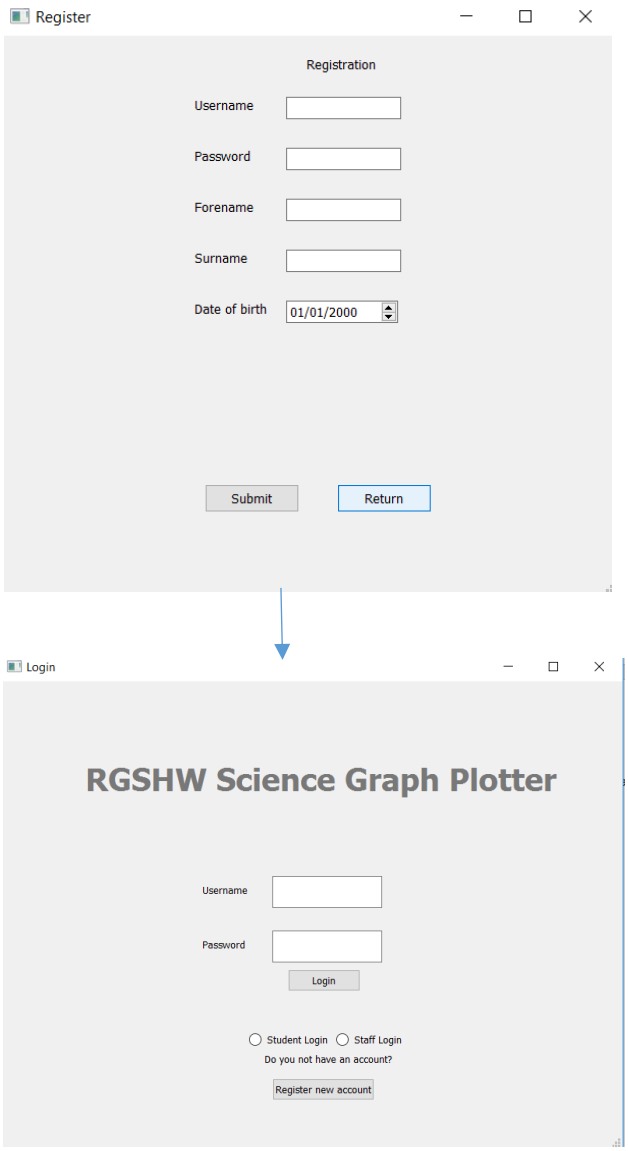
Tests:

Open the register window



When the “register new account” is pressed it opens the register window. This test has been successful.

Return button



When return is pressed the register window is closed and the login window is opened. This is as expected.

Version 0.3

aim:

To connect to the database

Input all of the values required for registering

Validate the values entered into the register screen by users

Hash the password

Insert the new user into the database if the values are correct.

```
40 class RegisterWindowClass(QtGui.QMainWindow,registerclass):
41     def __init__(self,parent=None):
42         QtGui.QMainWindow.__init__(self,parent)
43         self.setupUi(self)
44         self.ReturnButton.clicked.connect(self.returnclick)
45         self.SubmitButton.clicked.connect(self.submitclick)
46
47     def returnclick(self,curr):
48         registerscreen.hide()
49         loginscreen.show()
50
51     def submitclick(self,curr):
52         registeruser= self.UserInput.text()
53         registerpass= self.PasswordInput.text()
54         registerforename=self.ForenameInput.text()
55         registersurname=self.SurnameInput.text()
56         registerdate = str(self.UserDOB.date())
57         registerdate=registerdate.replace('PyQt4.QtCore.QDate(','')
58         registerdate=registerdate.replace(')','')
59         registerdate = registerdate.split(',')
60         self.ErrorLabelRegister.setText('')
61         newtext=('')
62         cur.execute("select StudentID from Students where StudentID = ?",(registeruser,))
63         data = cur.fetchall()
64         if registeruser.isalnum() == False:
65             newtext="This username isn't alphanumeric"
66         elif len(registeruser)> 20 or len(registeruser) < 7:
67             newtext = "This username isn't between 7 and 20 characters long"
68         elif data != []:
69             newtext = 'This username is already in use'
70         elif len(registerpass)> 20 or len(registerpass) < 7:
71             newtext = "This password isn't between 7 and 20 characters long"
72         elif registerforename.isalpha()== False:
```

```
73             newtext = "This forename is not alphabetical"
74         elif len(registerforename)>20 or len(registerforename)<2:
75             newtext = "This forename is not between 2 and 25 characters long"
76         elif registersurname.isalpha()== False:
77             newtext = "This surname is not alphabetical"
78         elif len(registersurname)>20 or len(registersurname)<2:
79             newtext = "This surname is not between 2 and 25 characters long"
80         elif int(registerdate[0])< (datetime.datetime.now().year -19) or int(registerdate[0])> (datetime.datetime.now().year -11):
81             newtext = "This date is either too early or too late"
82         self.ErrorLabelRegister.setText(newtext)
83         if newtext == '':
84             registerpass=registerpass.encode('utf-8')
85             registerpass = hashlib.sha512(registerpass).hexdigest()
86             registerdate = registerdate[0] + '-'+registerdate[1].strip() + '-' + registerdate[2].strip()
87             cur.execute("INSERT INTO Students(Username,Password,Forename,Surname,DOB) Values (?,?,?,?,?)",(registeruser,registerpass,registerforename,registersurname,registerdate
88             loginscreen.show()
89             registerscreen.hide()
90         else:
91             self.ErrorLabelRegister.show()
92
93
94 class GraphWindowClass(QtGui.QMainWindow,graphclass):
95     def __init__(self,parent=None):
96         QtGui.QMainWindow.__init__(self,parent)
97         self.setupUi(self)
98
99
```

```
102 loginscreen = LoginWindowClass(None)
103 registerscreen = RegisterWindowClass(None)
104 graphscreen = GraphWindowClass(None)
105 loginscreen.show()
106 apptemplate.exec()
```

Explanation

The validation has been done using the pseudocode stated in my design. This is done to ensure that values that are either incorrect, or values that wouldn't work with the database are rejected.

Lines 50 – 56 are the lines which run when the submit button is clicked. Each variable is set to a value of a text book. This is used to validate the values

Lines 57-59 manipulate the string value for date so that it cannot only be validated (as it is in the form of a list) but also so that it is entered into the database in the correct format for the database. This order means that the string can be sorted by sqlite's built in date handling.

Line 60 – the label's value for the error is changed to nothing, so that it can be changed to the right value when the text is determined.

line 62 -63 – The user is selected from the sql where the username is the same as the one for the current user.

64-65 - checks if the username is alphanumeric as the username should only contain letters and numbers. If not an error message is set to the error text.

66-67 – checks that the username is within the right range. If not, it changes the error text to tell the user the correct range and which errors there are.

68-69 - checks that the username doesn't exist. If the value returned is an empty list it means that the username does not exist, otherwise it means the username is already in use.

70-71 – checks that the password is the right length – there is no alphanumeric check as symbols in the password is fine.

72-73 – checks that the forename of the person is alphanumerical.

74-75 - Checks that the forename of the person is within a reasonable range (between 20 and 2 characters)

76-77 – checks that the surname is alphabetical.

78-79 – checks that the surname entered is in the desired number of characters.

80-81 – checks that the date of birth is within the window in which a student could be studying at the school. This is done by checking the year entered compared to the current year. If the date of birth means that the user would be over 18 or under 11 then the programs rejects this and creates an error message

82 – Sets the text of the error label to the value of the error text produced by the error. I decided that 1 error message would suffice as it would be more difficult to design the UI around it.

83 – If the new error statement is still equal to nothing then there must have been no error produced. This means that there must have been no error validated against. This triggers the new addition of a user.

84-86 – the data is changed ready for insertion into the database – The password is hashed and the date is further edited to make it database compatible

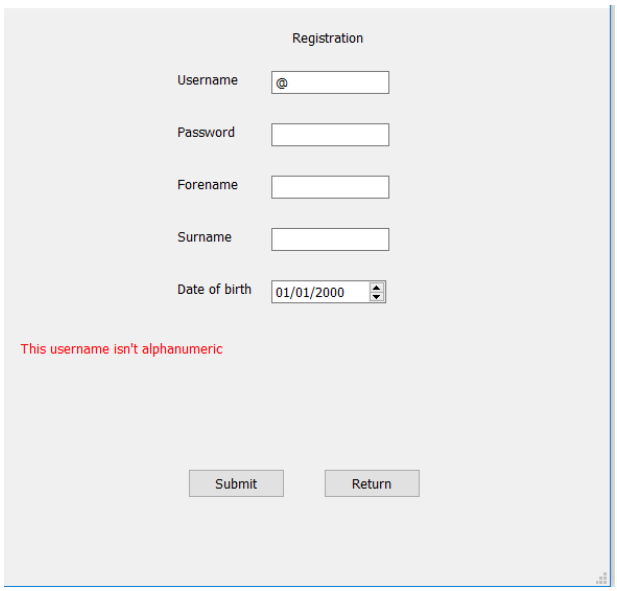
87-89 – The details are added to the "students" table in the database. The register user UI is then closed, and the login window is opened.

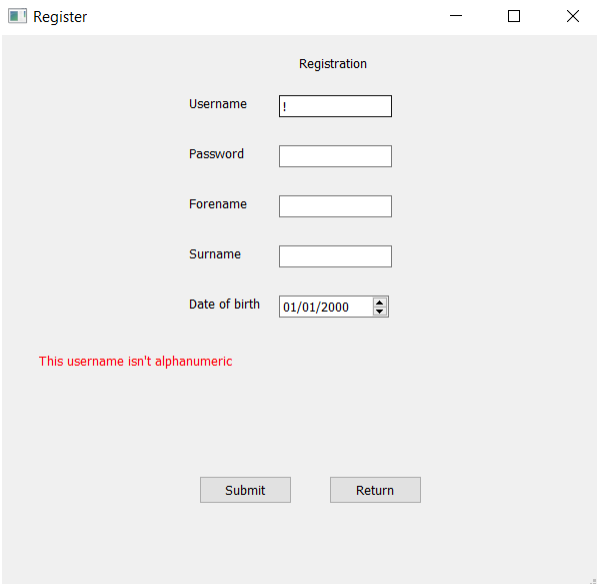
90-91 – If there is an error then the error message is shown.

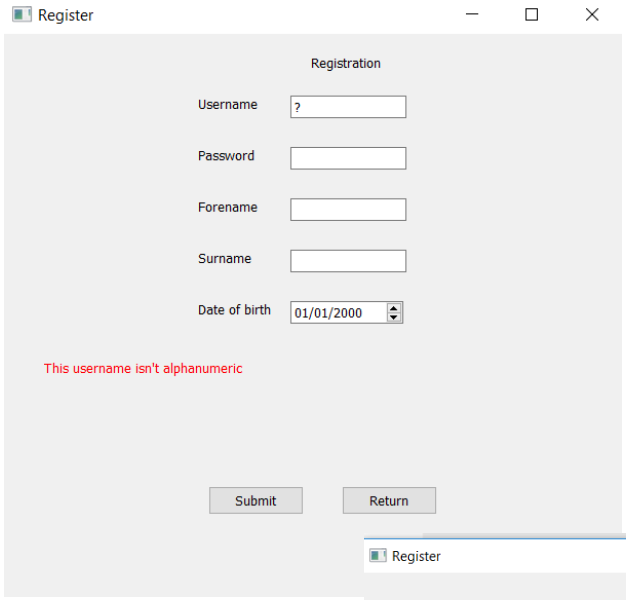
Testing

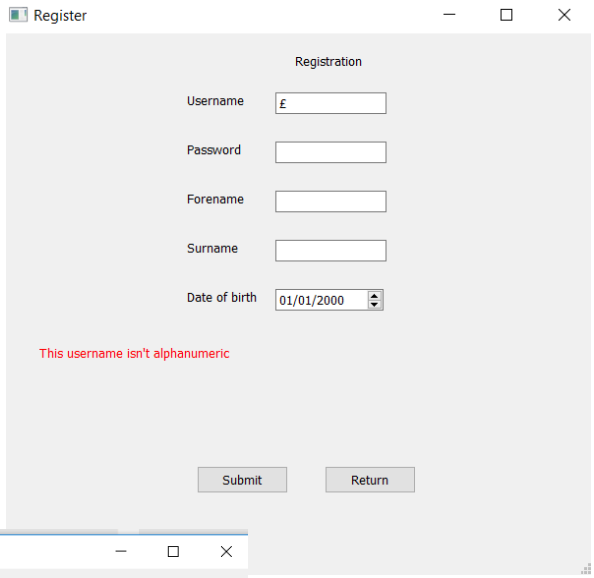
Enter symbols in username

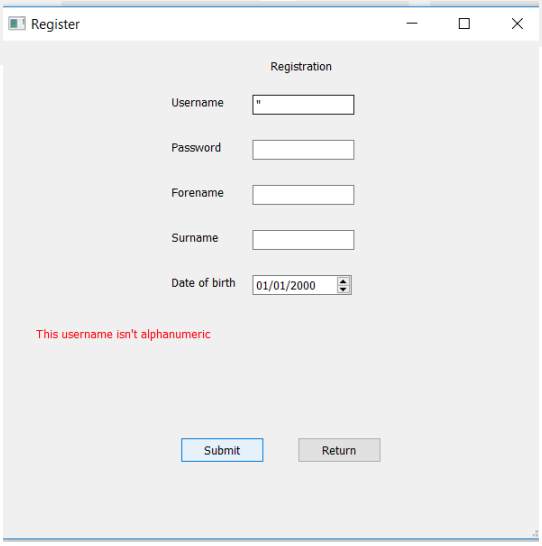
The username can’t contain a symbol so it should not be accepted







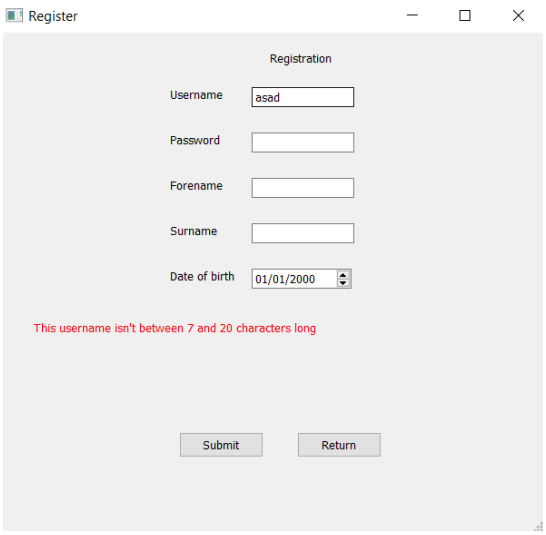


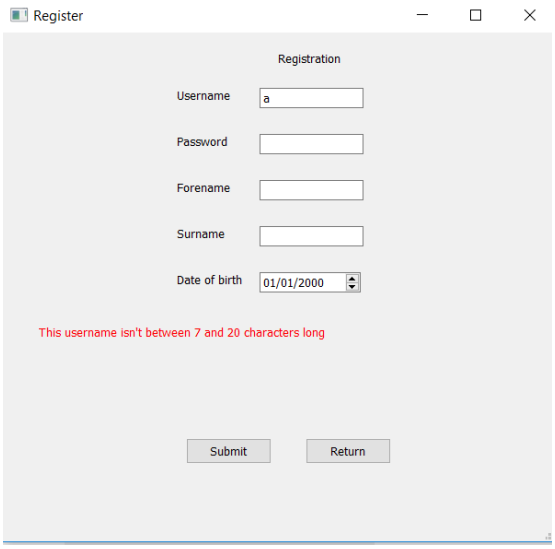


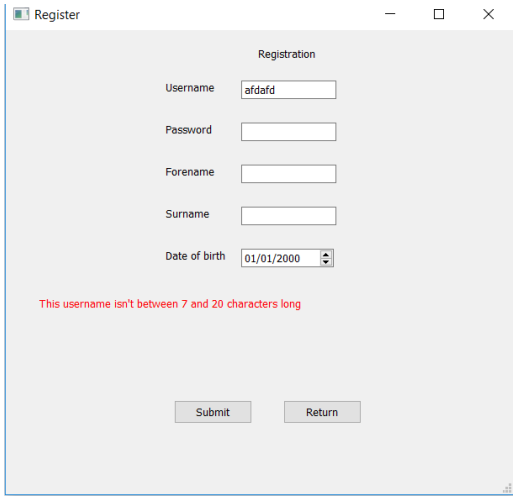
The error message shows that the username is rejected should symbols be entered into the lineedit. This means that there should not be any usernames entered with symbols.

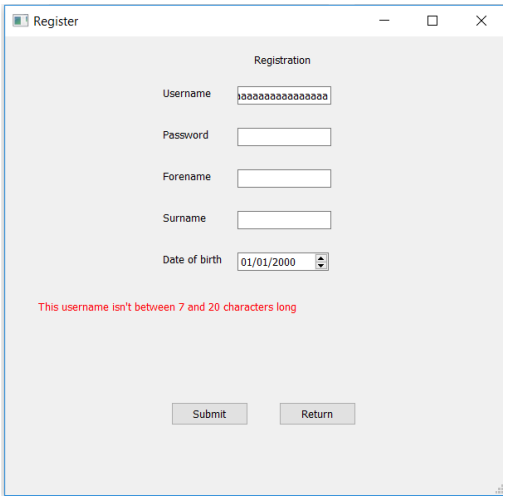
Check that the username is the right length

The text should be between 7 characters and 20.

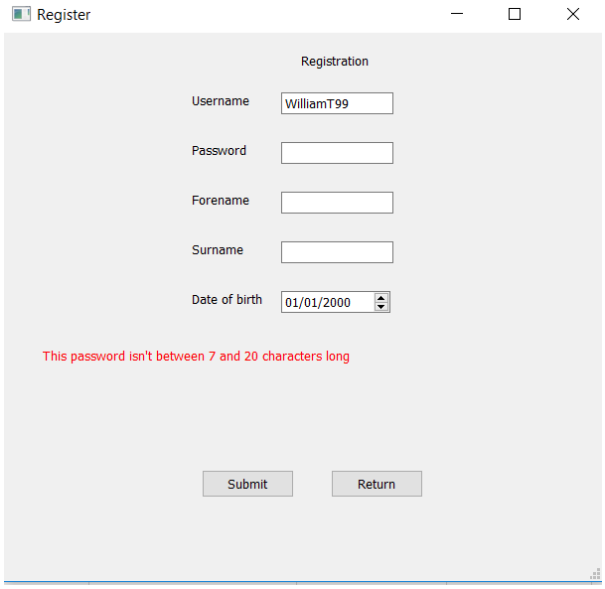








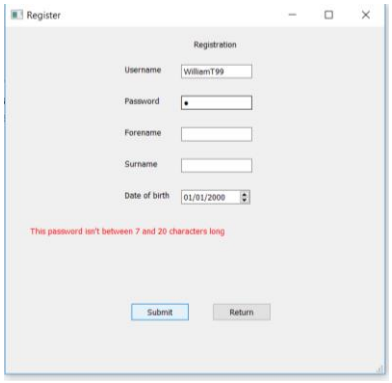
Here we can see that the test has been successful. When the username was over the character limit the username was rejected, similarly it was rejected if the length was too short.



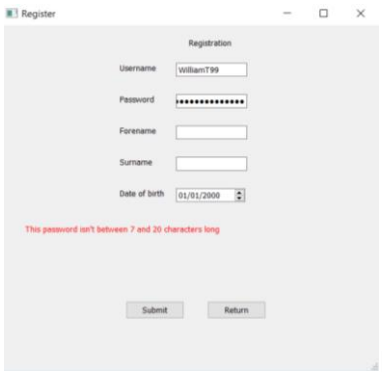
Here a password is valid, as explained in the development. The username has been accepted as the error has moved onto the password rather than the username.

Check that the password is the right length

Here the length of password is being tested. It needs to be between 7 and 20 characters long to fit requirements for password strength.

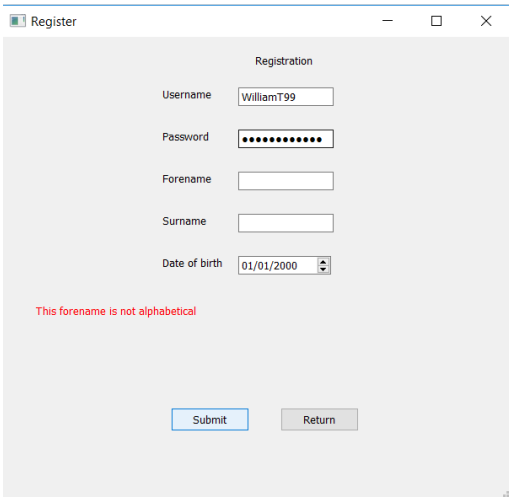


here “a” is entered into the password box



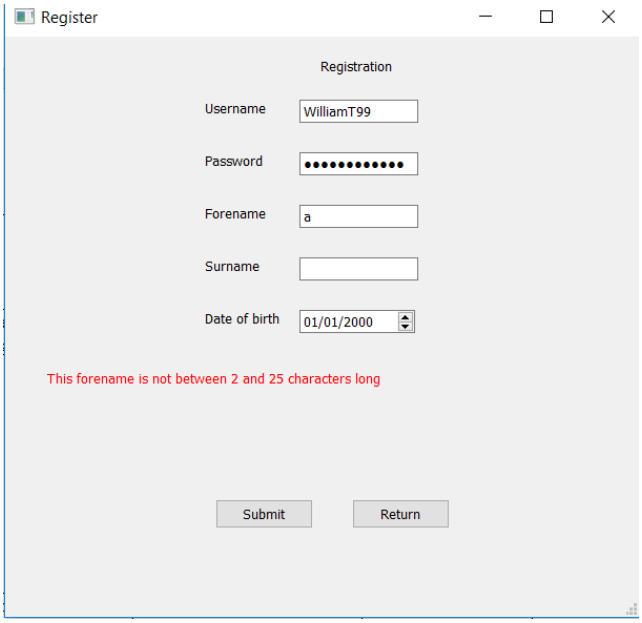
here “aaaaaaaaaaaaaaaaaaaa” is entered into the password box

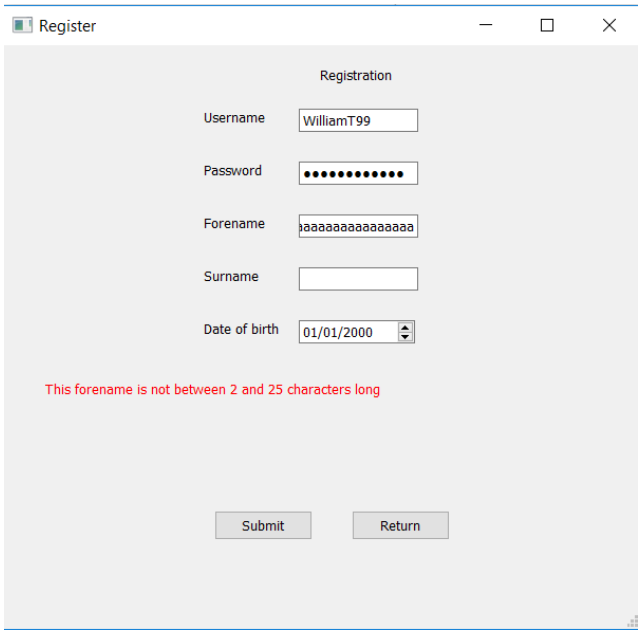
The length of the passwords here are validated as being unacceptable. This is expected as the passwords do not meet the strength required.



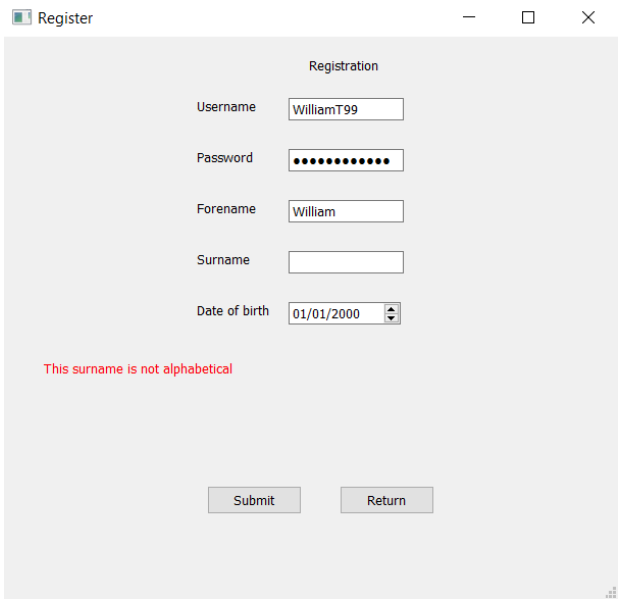
The password “williamspass” is entered. This is of acceptable length and therefore is accepted. The next error message “The forename is not alphabetical” is shown, denoting that the password has been accepted. This is what was expected.

Check that the forename is the right length





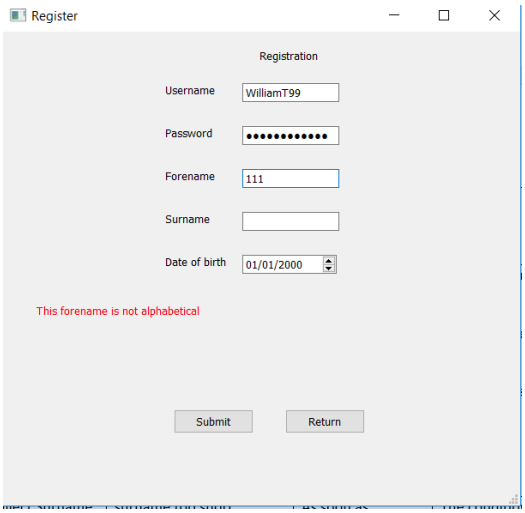
Here incorrect lengths of forenames are tested. They are rejected as the length of the forename is not correct. This is as expected and requires no changes.

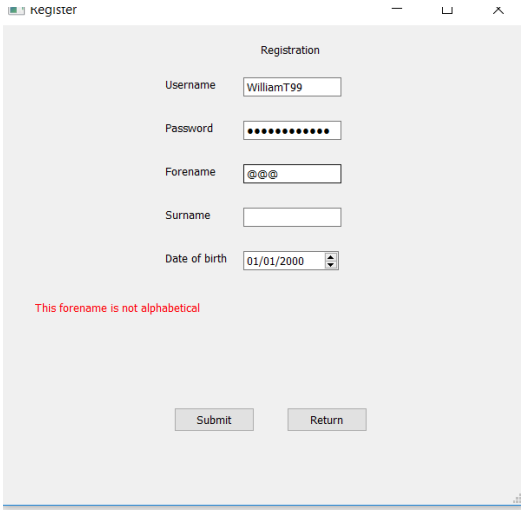


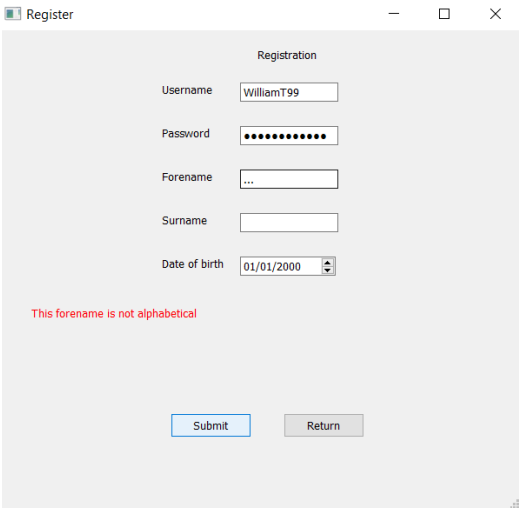
Here a forename of the correct length is used. The next error message is shown denoting that the error is accepted.

Testing that the forename is only alphabetical

The password is expected to be only alphabetical. Values should only contain letters. It has been proven to work when alphabetical values are entered in the test above, therefore only incorrect values need to be tested



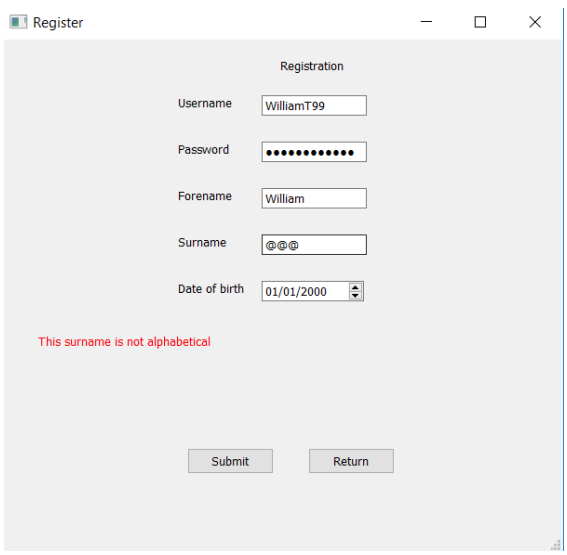


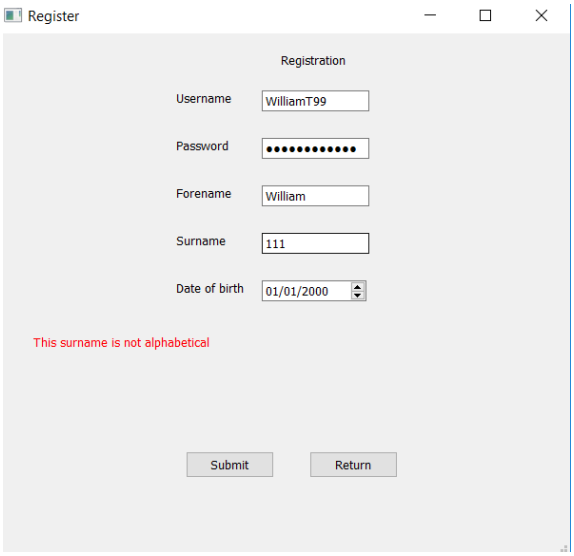


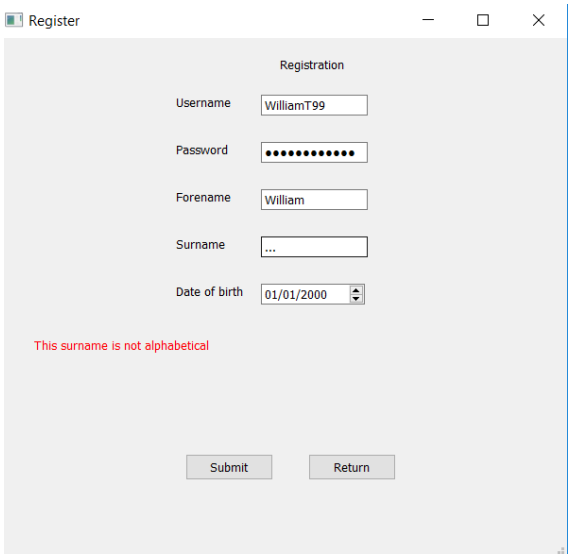
Here non alphanumerical values are entered into the text box. They are not accepted and therefore the test is successful.

Checking that the Surname is alphabetical

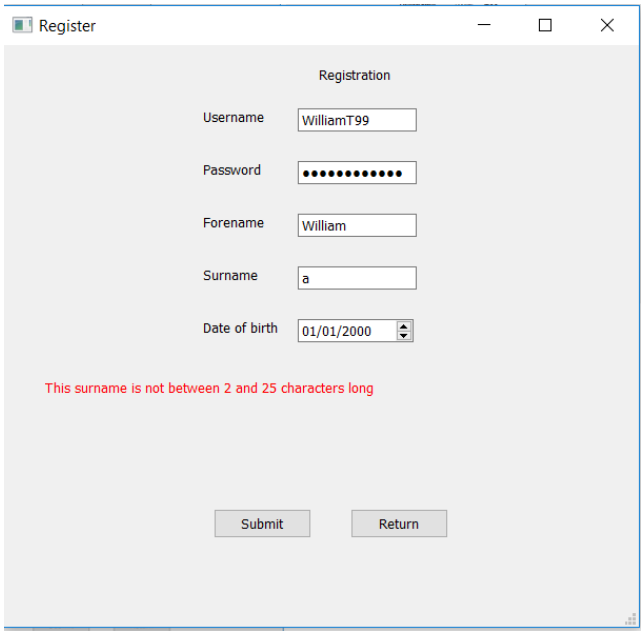
The surname is expected to be alphabetical. Both incorrect and correct values will be tested.







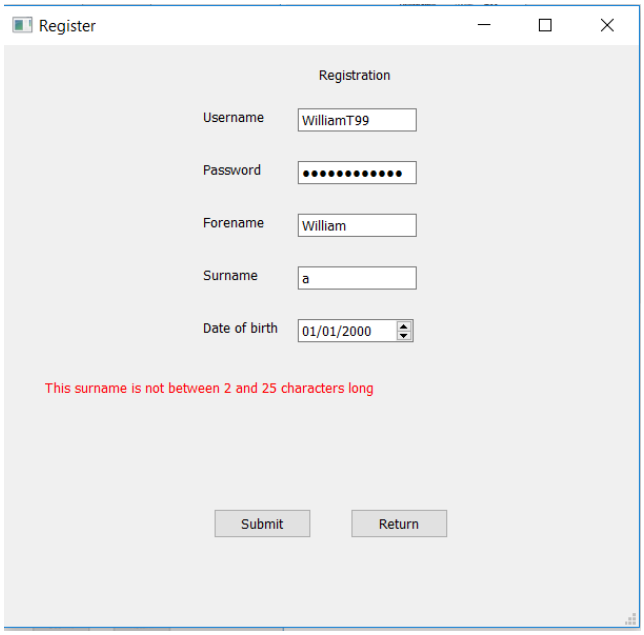
Here non-alphabetical values. All values are rejected which is a successful test.

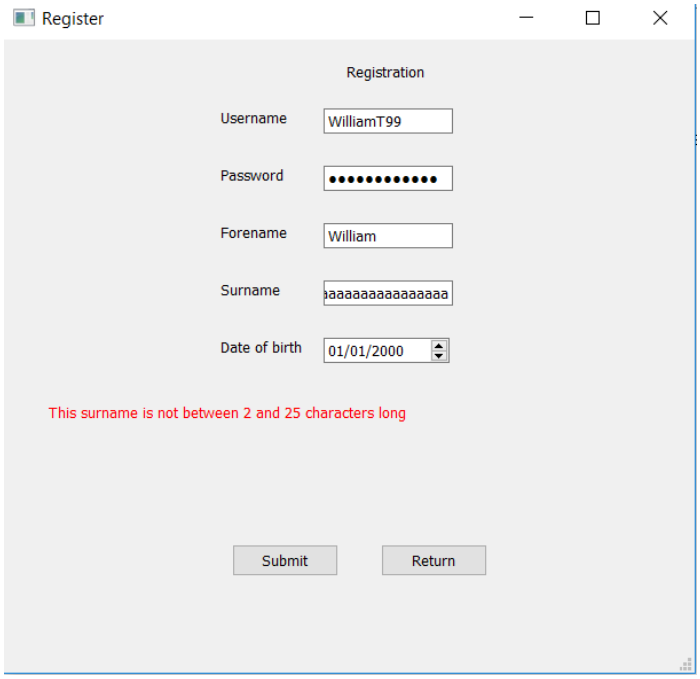


Here a surname that is alphabetical is entered. This is as expected and no errors are found with the code. The next error is shown.

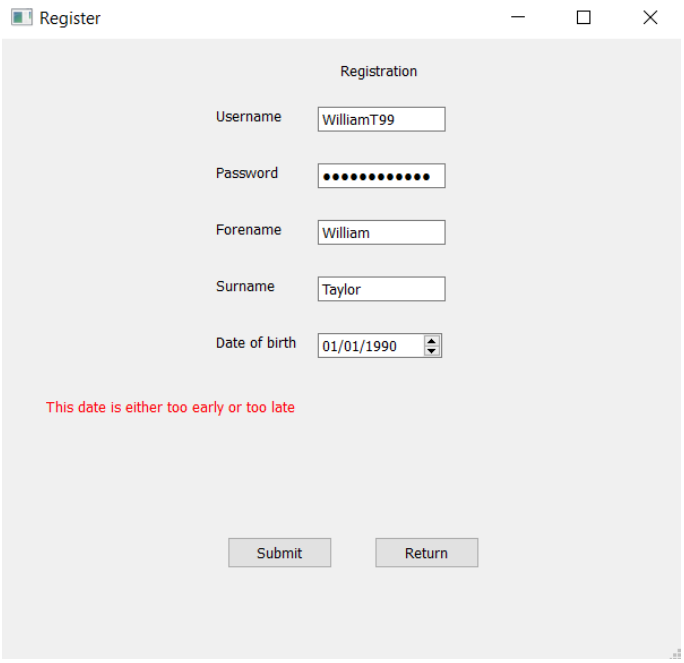
Checking surname lengths.

The surname is expected to be between 2 and 20 letters long. Here the length is tested.





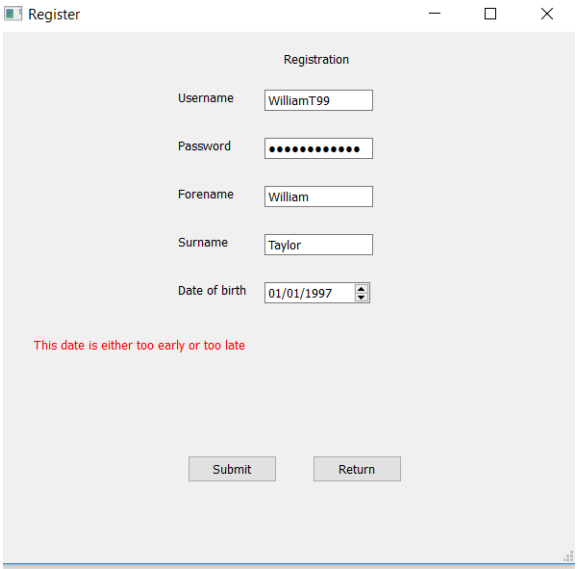
Here incorrect lengths are rejected. Therefore the tests are successful.

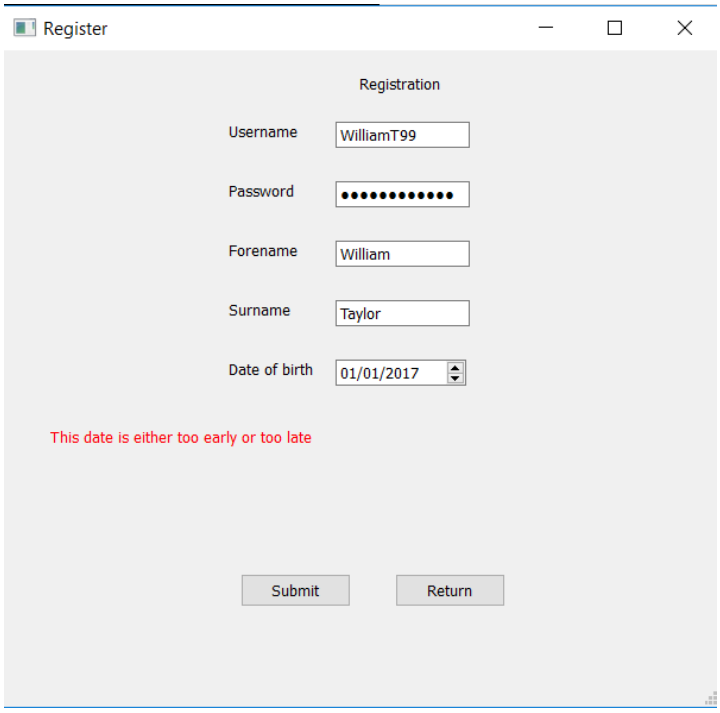


Here a correct value for a password is accepted as the next error message is shown. Therefore the next error is shown.

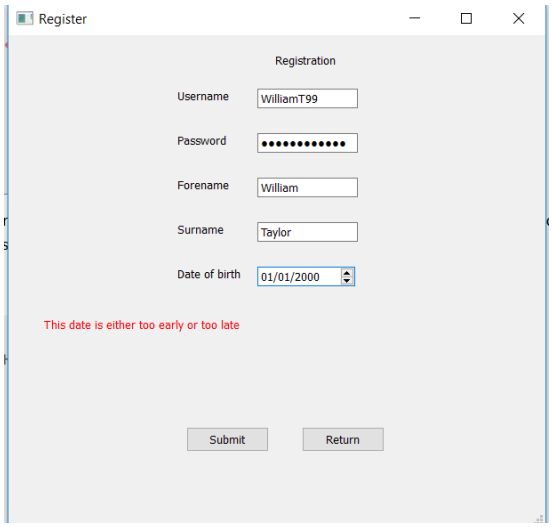
Checking that the date is within the correct range

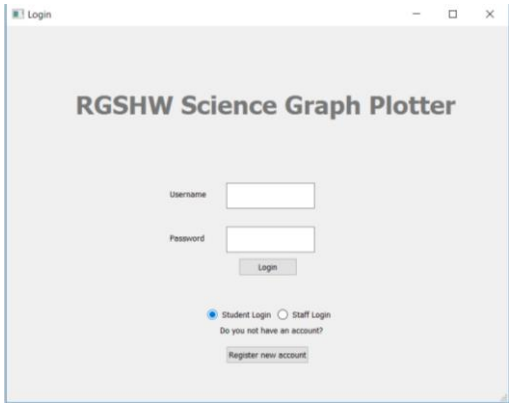
This program expects students to be within the age range of school. This means that the age must be between 19 years ago and 11 years ago.



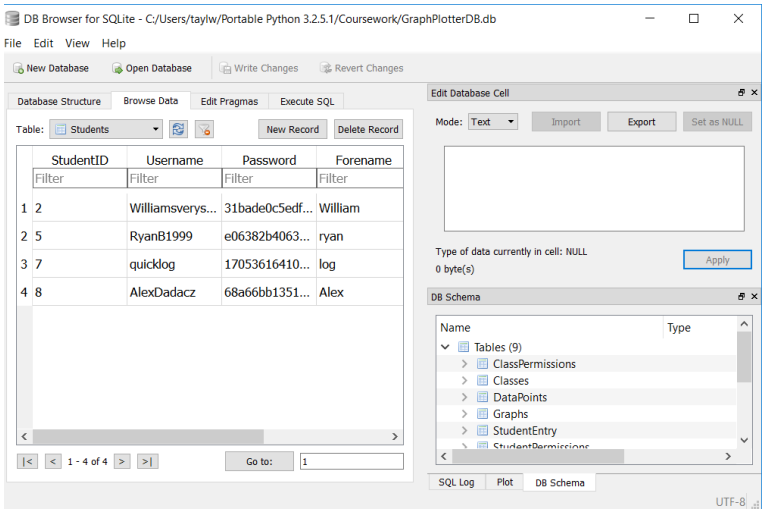


Here incorrect values for dates which would not be used by current students are rejected. This test is successful





When a correct value is entered the user is returned to the home screen. This is as expected as it means that the code for a correct user has been executed.



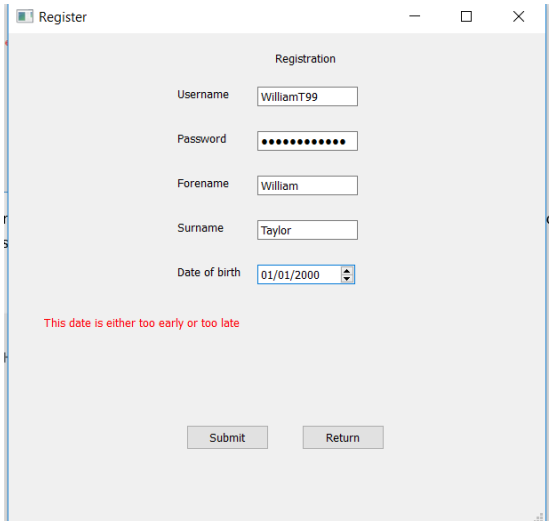
However after testing this the user was not added to the database. This means that the test was unsuccessful as the new user should've been added.

My first suspicion as to why the value was not entered to the database was because the changes were not committed to the database. The code was therefore changed to :

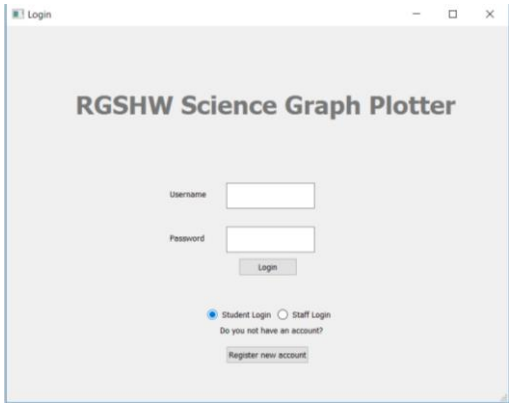
```
83          if newtext == '':
84              registerpass=registerpass.encode('utf-8')
85              registerpass = hashlib.sha512(registerpass).hexdigest()
86              registerdate = registerdate[0] + '-'+registerdate[1].strip() + '-' + registerdate[2].strip()
87              cur.execute("INSERT INTO Students(Username,Password,Forename,Surname,DOB) Values (?,?,?,?,?)",(registeruser,registerpass,registerforename,registersurname,registerdate
88              con.commit()
89              loginscreen.show()
```

The con.commit() should now commit the changes to the database. This is now tested below

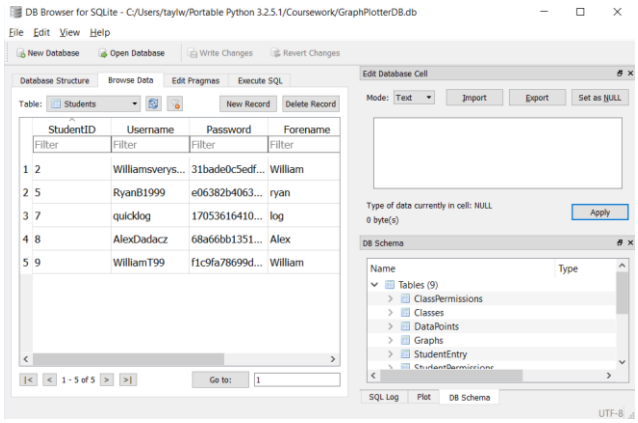
Testing that data is now entered into the database after changes



The same data is now entered and as this is what is expected the data should be entered into the database.

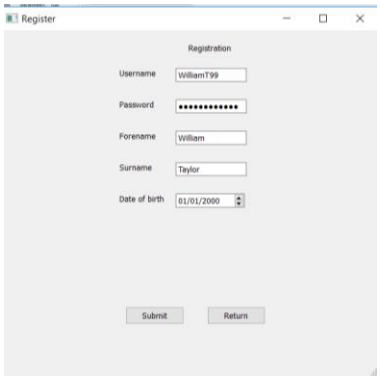


The program goes to the home screen again after submit is clicked.

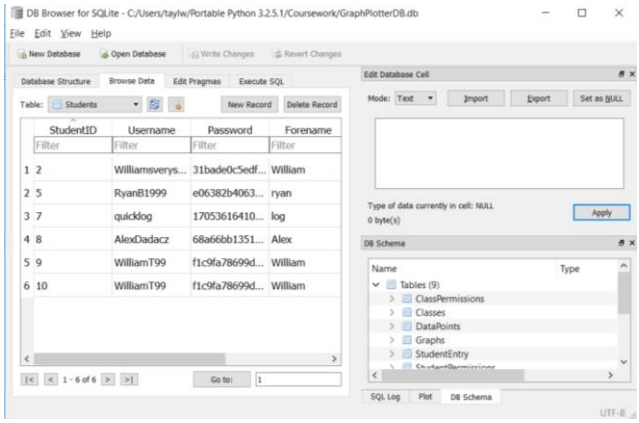


Here the value has been added to the database with all of the correct values. This is as expected and the test has been successful

Checking that the username is not entered if it is not unique



The same data is entered as before. This should be rejected as it is not unique.



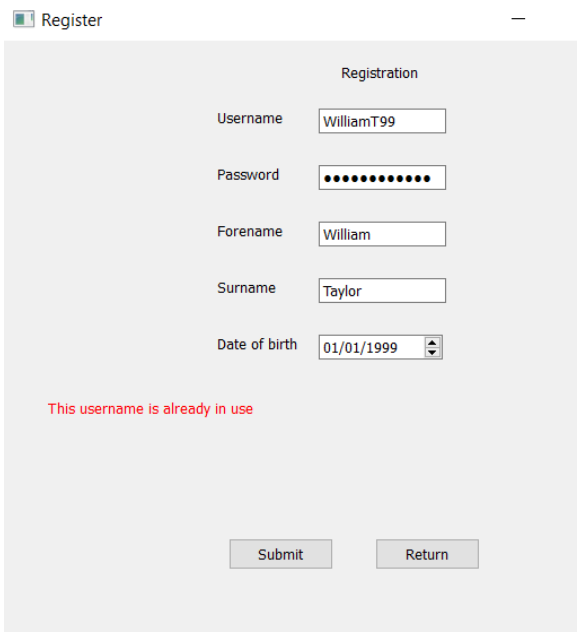
This is not correct as the new data should've been rejected. Upon further inspection of the code I noticed that the line

"cur.execute("select StudentID from Students where StudentID = ?",(registeruser,))"

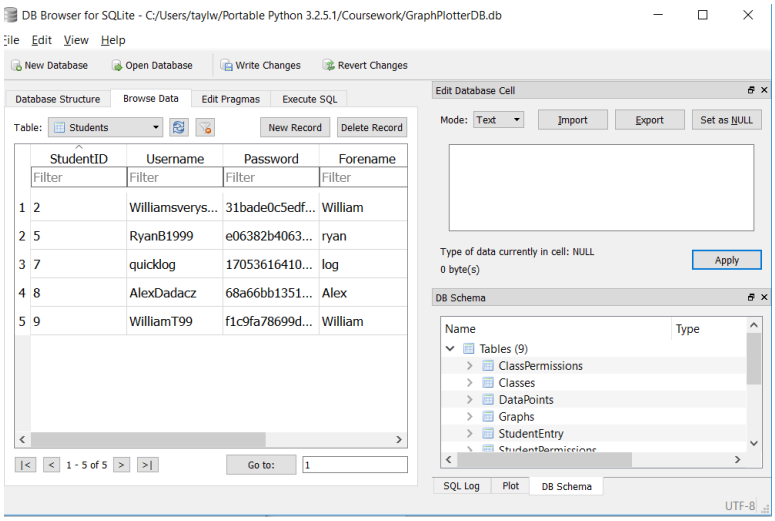
Is incorrect as I should be checking against Username rather than StudentID. This is because the studentID is nothing to do with the username and is autoincremented so the username will always appear unique. This was changed to

cur.execute("select StudentID from Students where Username = ?",(registeruser,))

The previous duplicate was deleted from the database and it was tested again.



The correct error message is now shown. Upon inspection the value is also not added to the database.



The test has now therefore been successful.

Checking that hashing is correct.

I will now check if the sha512 hash is correct for the data I remembered. I will decrypt the values in the database using “https://md5hashing.net/hash/sha512” to obtain the original password. If this is the same as what was entered before then the test has been successful.

f1c9fa78699d76d1cc445f2855a034a34b90b946176b549a932f867ece55c3c67a68140c6d3c9e3b10918b13e34f52fe35b3ec09630e1ffa4732f736d0c55215

was entered into the decrypter.

Williamspass was returned, denoting that the encryption was a success.

I entered “PasswordTest” as a password into my program and received the sha512 hash of

3cc77174a566ce88e22ba9da987f8a69ebb01f164ee20e3e5ded3fef4f42b9e70529e4de3b6baf04a016adc14f000786eadb396c801c720d3cb44790673d2265

When this is decrypted I received

“PasswordTest”

As the original entry. This means that the encrypting is working as expected

Version 0.4

Aim: To input login credentials from the user. To search the database for the username entered by the user and to compare the two hashed passwords against one another

```
13 class LoginWindowClass(QtGui.QMainWindow,loginclass):
14     def __init__(self,parent=None):
15         self.user = "student"
16         QtGui.QMainWindow.__init__(self,parent) #initiates the login window when the program starts
17         self.setupUi(self)
18         self.StudentOption.setChecked(True)
19         self.Incorrect.hide()  # hides the red text that appears when a password is entered incorrectly
20         self.LoginButton.clicked.connect(self.loginclick) # initiates loginclick method when the login button is clicked
21         self.NewAccountButton.clicked.connect(self.registerclick) # initiates registerclick method when register button is pressed
22         self.StaffOption.clicked.connect(self.staffclicked)
23         self.StudentOption.clicked.connect(self.studentclicked)
24     def studentclicked(self,curr):
25         self.user = 'student'
26     def staffclicked(self,curr):
27         self.user = 'staff'
28
29
30     def loginclick(self,curr):  # stores the username and password as strings and checks them against the dummy user.
31         print(self.user)
32         username=self.Username.text()
33         password=self.Password.text()
34         cur.execute("SELECT Password FROM Students WHERE Username = ?",(username,))
35         temporarypass = cur.fetchall()
36         password=password.encode('utf-8')
37         password = hashlib.sha512(password).hexdigest()
38         if temporarypass == password:
39             currentuser=username
40             loginscreen.hide() #hides the window ready for the  main UI
41             graphscreen.show()
42         else:
43             self.Incorrect.show()  #shows the red text for incorrect password
44             self.Username.setText("") #removes current text in the username and password boxes ready for re-entry
45             self.Password.setText("")
```

Lines explanation

31- prints the current user access level selected via the radio button. This is used for testing to see if pressing the radio button changes the user attribute.

32-33 the values entered by the user are retrieved and stored for validation

34-35 accesses the SQLLite database and finds the password where the username is equal to the currently entered one

36-37 hashes the user entered password.

38-41 checks if the hashed version is the same as the hashed one in the database. If it is the current user is set the username value and the graph screen is opened.

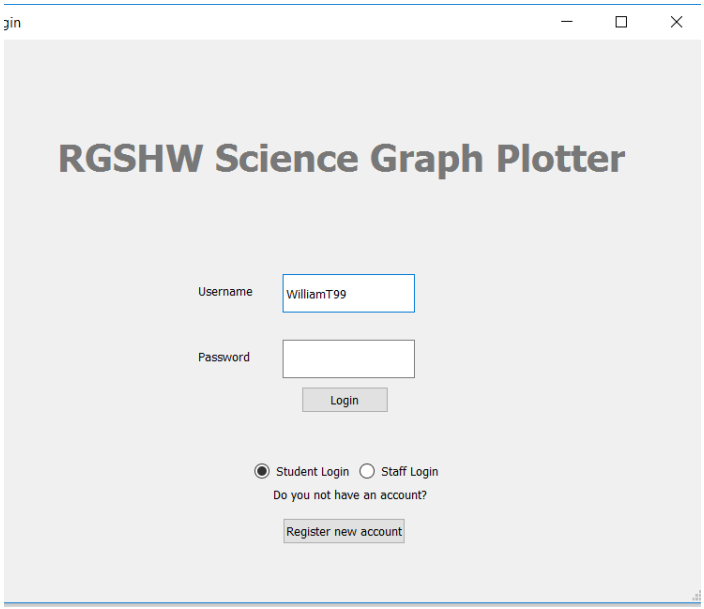
42-45 otherwise the incorrect text is shown and the username and password are overwritten so that the user must re-enter them

To test the values entered by the user I will use a combination of breakpoints and variable watch to see the values of the variables at key points of validation

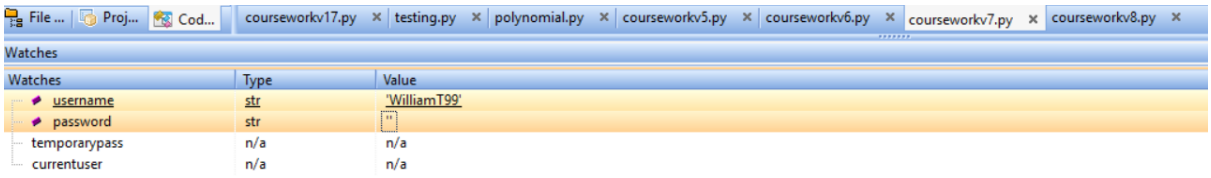
Testing that the correct password is fetched from the database.

In this example I will be using the username WilliamT99

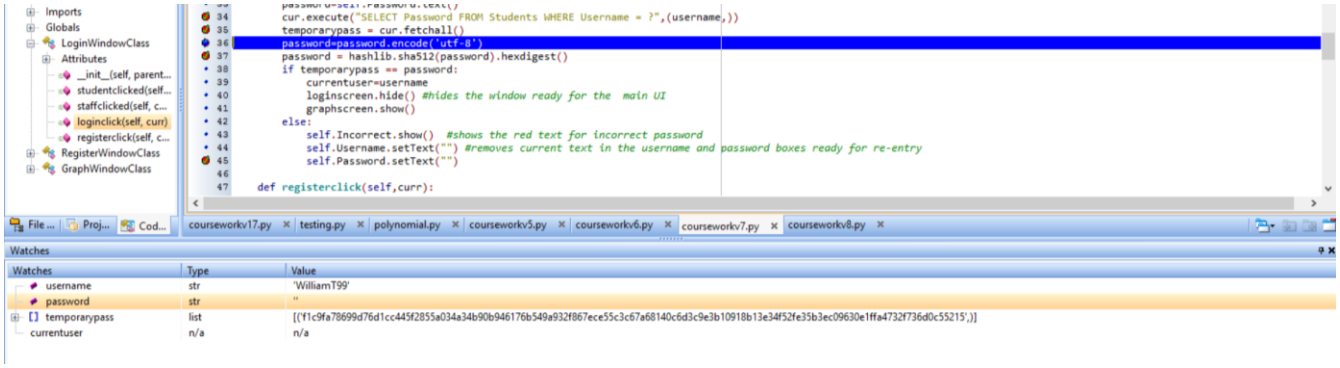
And the password williamspass



The username is entered and login is pressed



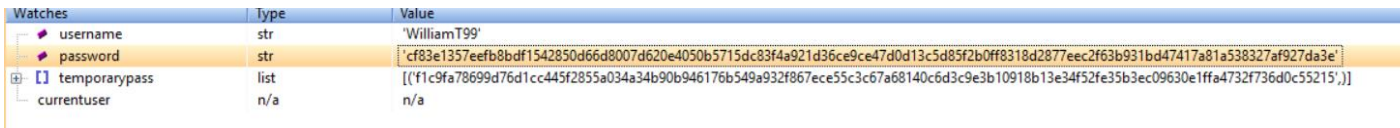
At the first breakpoint on line 34 the values are correct as entered



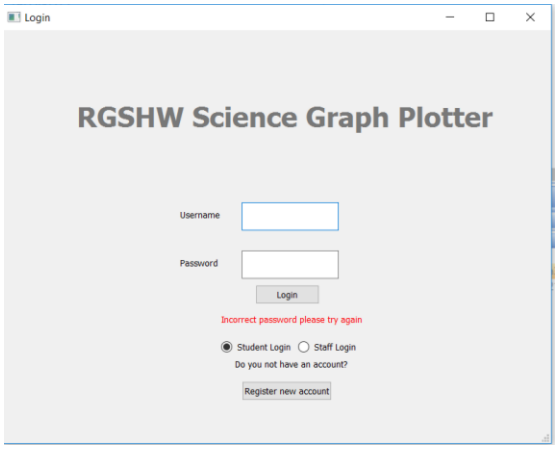
At line 36 the value for temporary pass is

[('f1c9fa78699d76d1cc445f2855a034a34b90b946176b549a932f867ece55c3c67a68140c6d3c9e3b10918b13e34f52fe35b3ec09630e1ffa4732f736d0c55215',)]

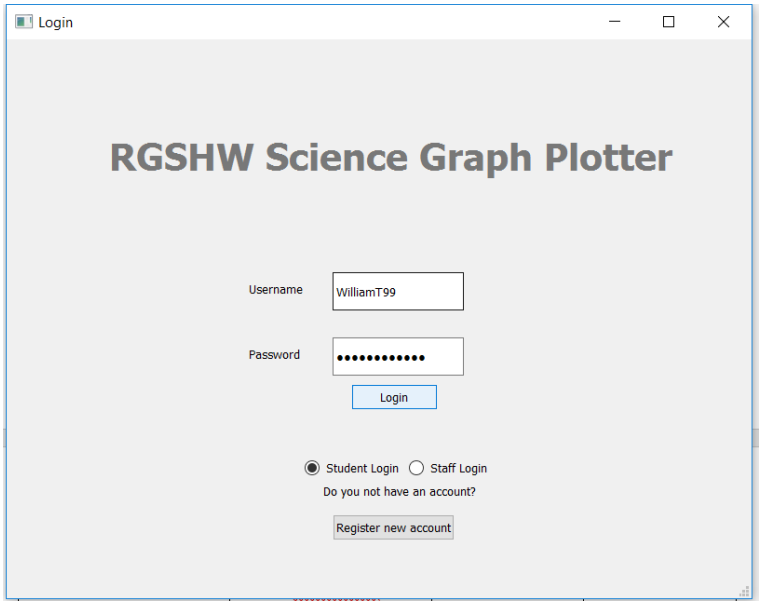
This is as expected as this is how pyscripter formats its variables.



The password doesn’t match the temporary password and should be rejected.



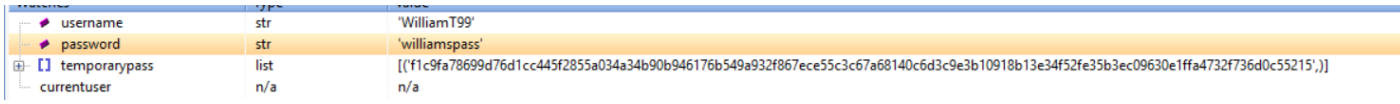
This is as expected as the variable should be rejected.



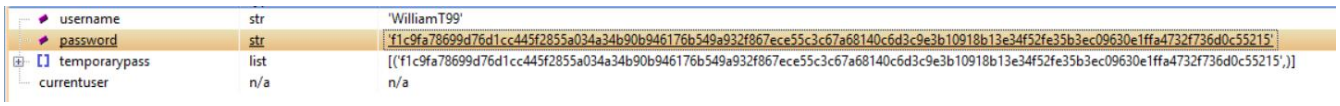
Now the correct value is entered “williamspass” for “WilliamT99”

This should result in the data being accepted

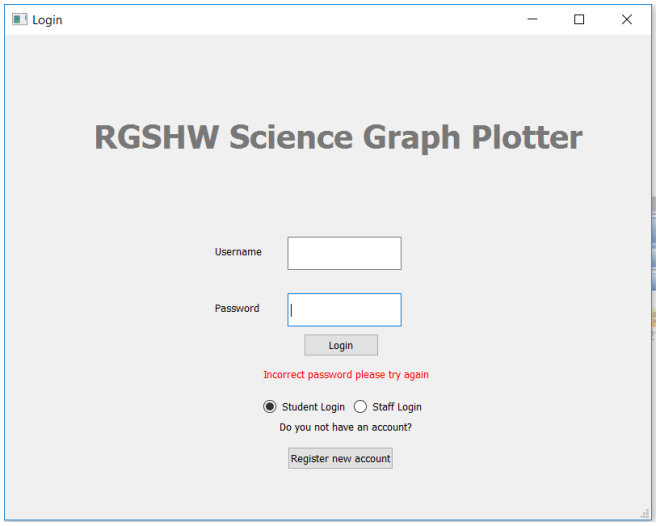
The correct values are in the variable watch after login is clicked.



The correct value for temporary pass is found



The password is then hashed to the right password.



The incorrect message is returned as the two passwords do not match. This is possibly because the formatting is incorrect for the comparison, and also the list is checked instead of its first object.

I shall fix this in the next version

Version 0.5

Aim : To fix the login error

To distinguish between student and teacher on login

```
22          self.StaffOption.clicked.connect(self.staffclicked)
23          self.StudentOption.clicked.connect(self.studentclicked)
24      def studentclicked(self,curr):
25          self.user = 'Students'
26      def staffclicked(self,curr):
27          self.user = 'Teachers'
28
29
30      def loginclick(self,curr):
31          username=self.Username.text()
32          password=self.Password.text()
33          if self.user == 'Students':
34              cur.execute("SELECT Password FROM Students WHERE Username = ?;",(username,))
35          else:
36              cur.execute("SELECT Password FROM Teachers WHERE Username = ?;",(username,))
37          try:
38              temporarypass = cur.fetchone()[0].replace("'","").replace(",","").replace("(","").replace(")","")
39              password=password.encode('utf-8')
40              password = hashlib.sha512(password).hexdigest()
41              if temporarypass == password:
42                  currentuser=username
43                  loginscreen.hide() #hides the window ready for the  main UI
44                  graphscreen.show()
45              else:
46                  self.Incorrect.show()  #shows the red text for incorrect password
47                  self.Username.setText("") #removes current text in the username and password boxes ready for re-entry
48                  self.Password.setText("")
49          except:
50              self.Incorrect.show()
51              self.Username.setText("")
52              self.Password.setText("")
```

Explanation

24-25 changes the user to a student if the student radio dial is pressed

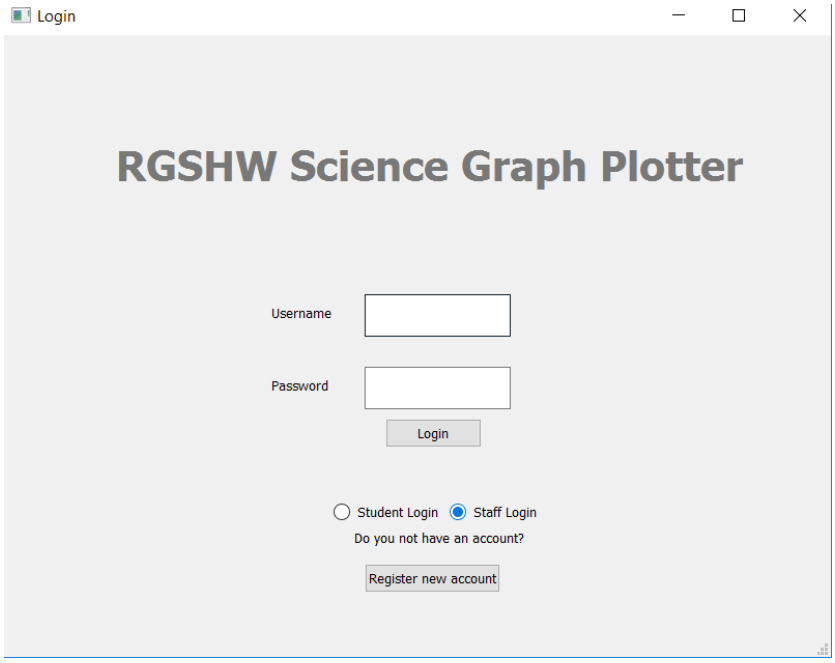
26-27 changes the user to a teacher if the teacher radio dial is pressed

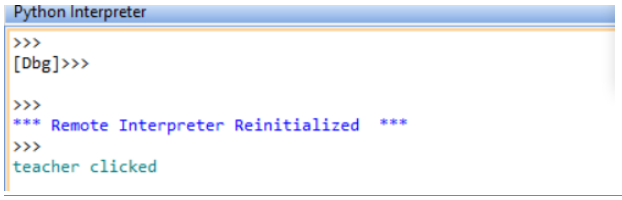
33-36 sets the sql query to finding the hashed password with the teacher/ student with entered username depending upon what was selected.

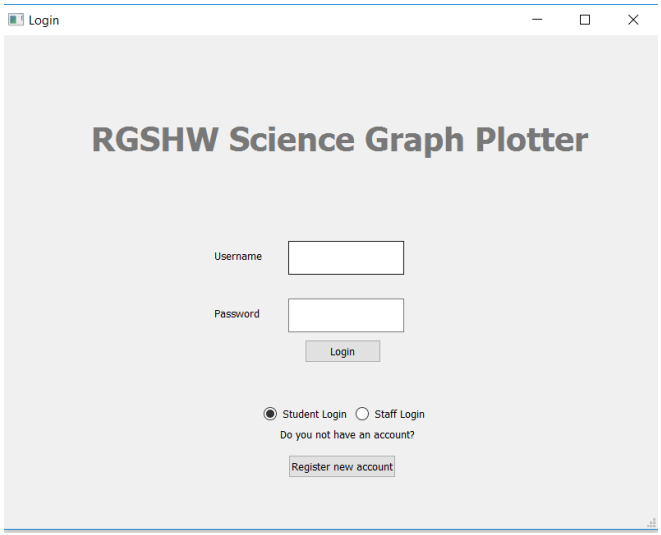
38 – Reformats the list returned by sql lite so that it can be used for comparison

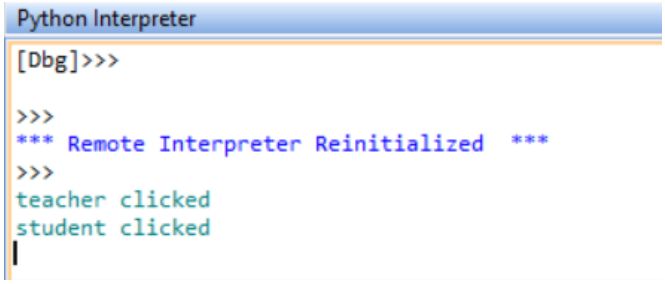
Checking that radio dial functionality works

I will check that the radio dials work correctly by changing to the selected role. I have added a print statement to validate that it has been pressed.





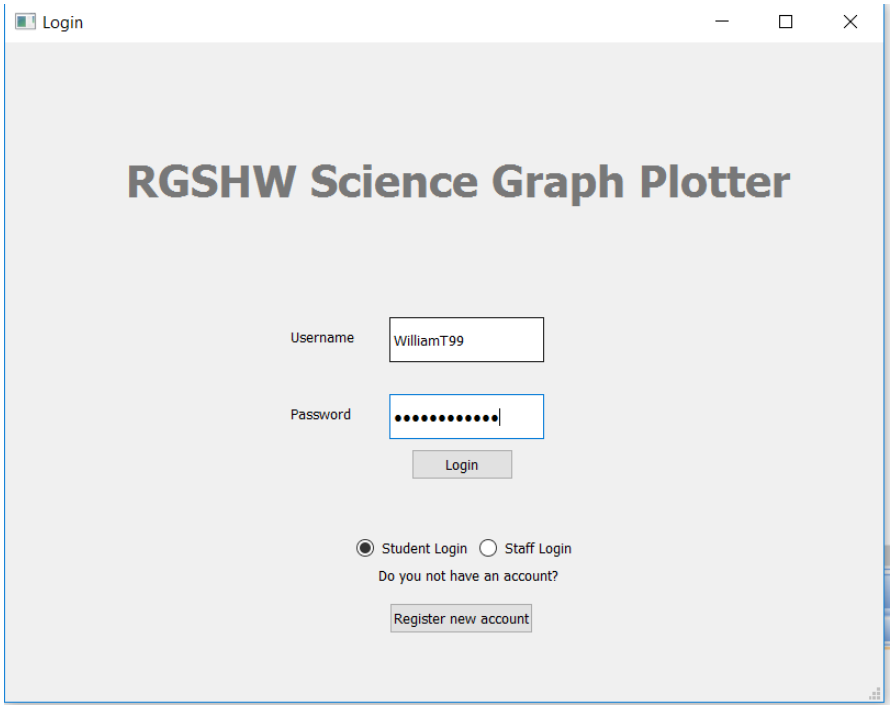


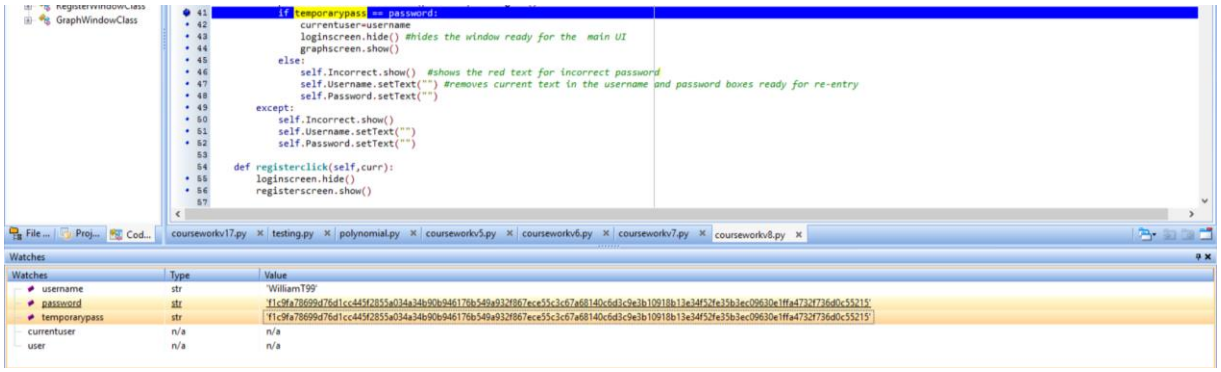


Here we can see that the code is being run for the respective radio dials and therefore the assignments are being run.

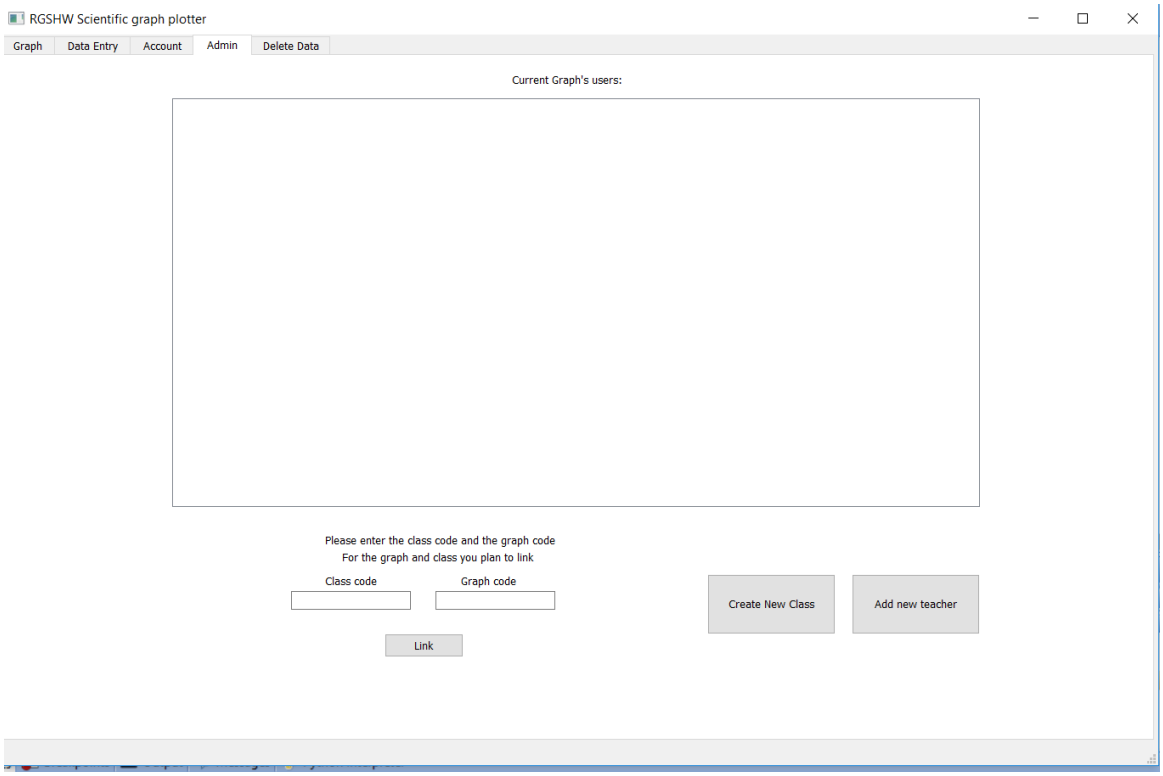
Testing that login works with students

I will be logging in with the user “WilliamT99” with the password “williamspass”



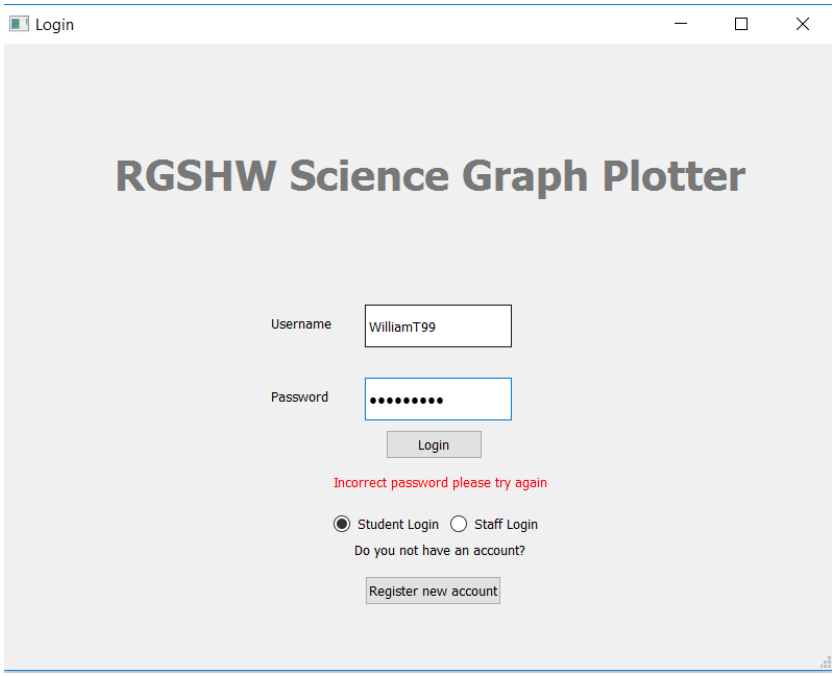


Here we see that the database value for password has been formatted correctly at this break point and the values should allow the user to login.



The graph window has been opened therefore the login has been accepted as expected.

This time I enter incorrect details. The user is “WilliamT99” and the password I use is “wrongpass” This should be rejected



This shows that the test has been successful.

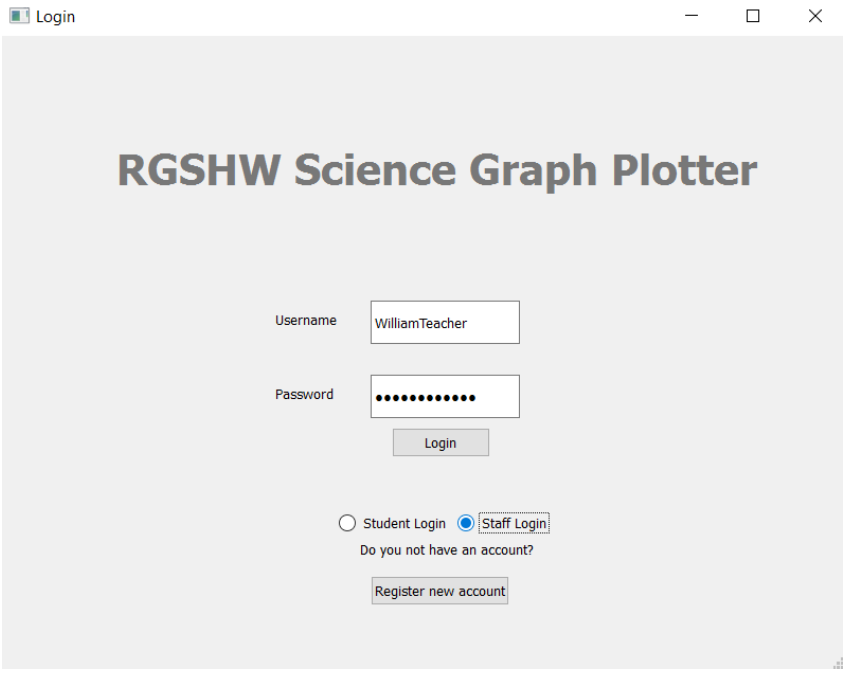
Testing that teacher login works

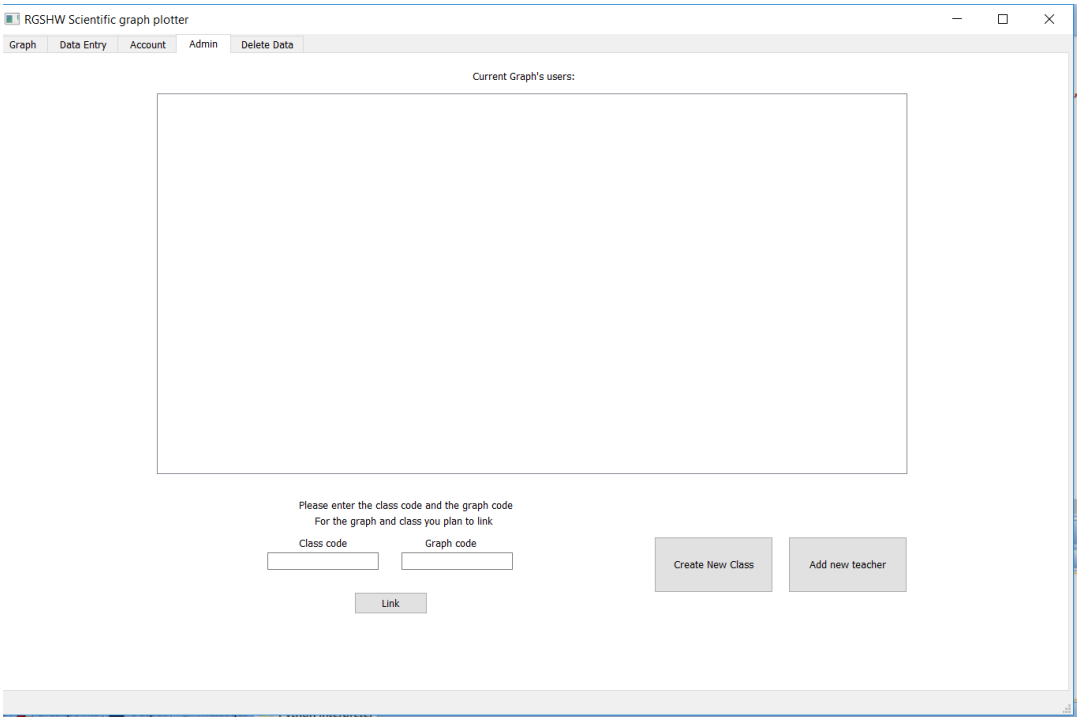
I have added a teacher for testing purposes as shown below

3	4	Taylor	WT	WilliamTeacher	f1c9fa78699d...

The username is “WilliamTeacher” and password “williamspass”

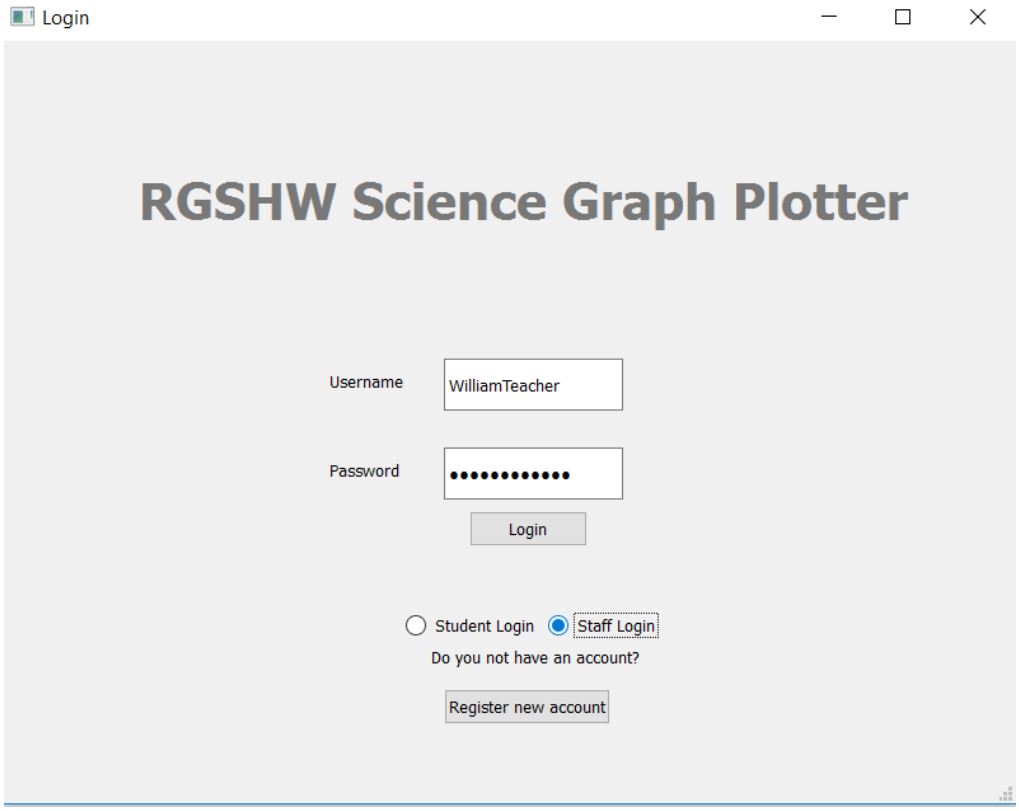
First I will check that these details allow me to login when staff is selected.

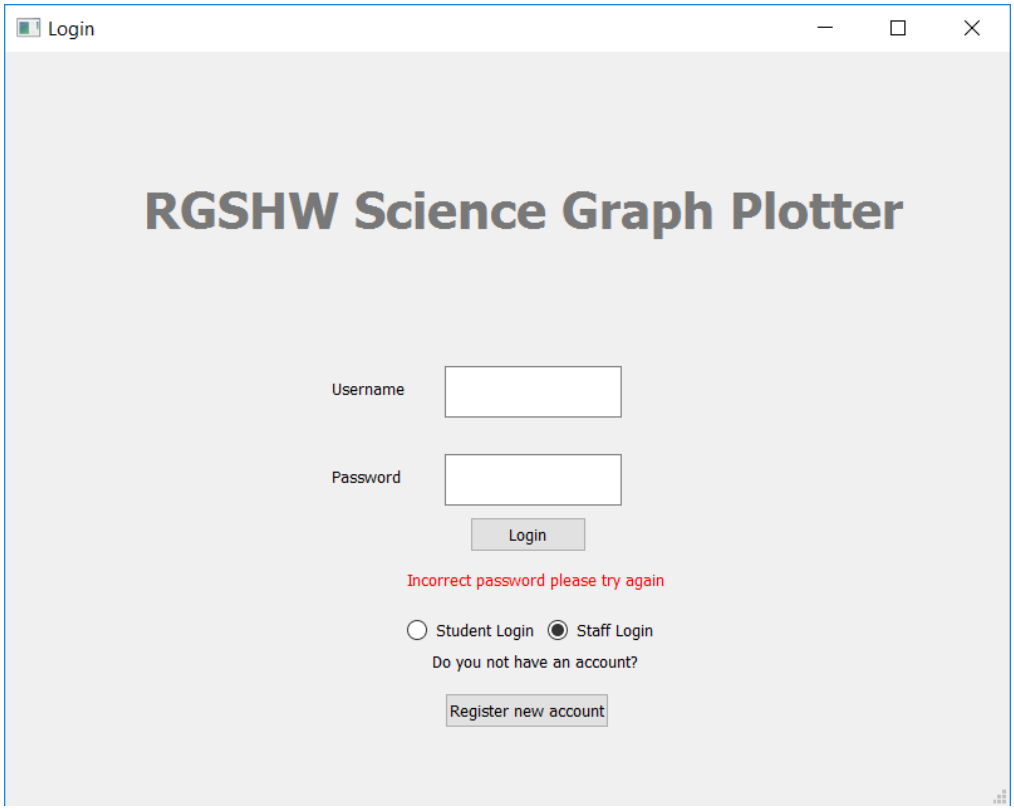




The user is logged in therefore the test is a success

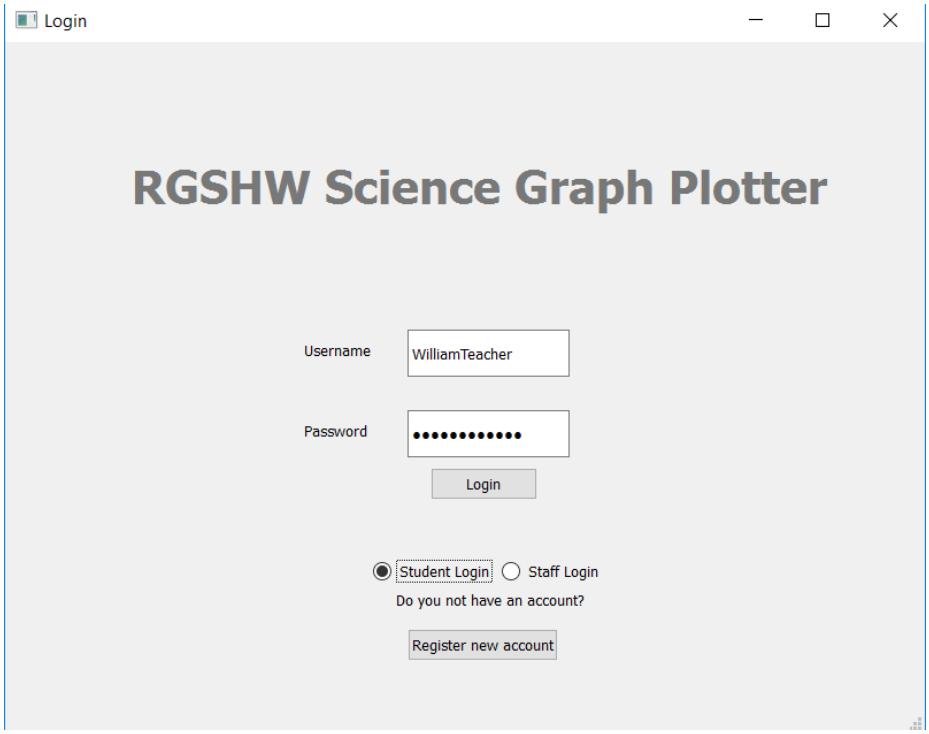
Now I shall test it with incorrect credentials: “WilliamTeacher” and “williamspass”



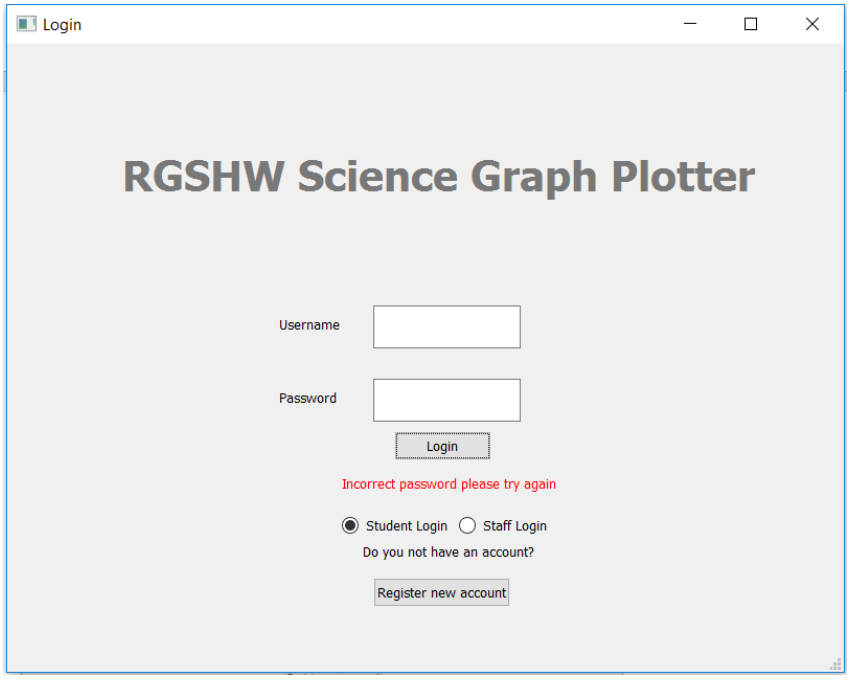


The credentials have been rejected, showing that the test is a success.

I will now test it by logging in as a student using the teacher credentials



Here I have entered “WilliamTeacher” and “williamspass” but as a student login



The username was rejected meaning that the data is definitely checking against the database.

Version 0.6

Aim : to ensure that radio buttons work. Ensure that the program doesn't crash if files are missing

```
try:
    from PyQt4 import QtCore, QtGui, uic
    import sys
    import hashlib
    import datetime
    import sqlite3 as lite
    error=False
    con = lite.connect('GraphPlotterDB.db')
    cur = con.cursor()
    cur.execute("select * from Students")
    datatest=cur.fetchall()
    if not datatest:
        print("CRITICAL ERROR : Database MISSING")
        error = True
    loginclass = uic.loadUiType("loginwindow.ui")[0]
    registerclass = uic.loadUiType("registerwindow.ui")[0]     #Loads the uis ready for use in classes
    graphclass=uic.loadUiType("graphwindow.ui")[0]
    graphcreate=uic.loadUiType("newgraphwindow.ui")[0]
except:
    print("CRITICAL ERROR : FILES MISSING")
    error = True

currentuser = ""
permissions = "Students"
colours = ["Black","Black"]
plotstyle="Polynomial"
```

```
    class GraphWindowClass(QtGui.QMainWindow,graphclass):
        def __init__(self,parent=None):
            QtGui.QMainWindow.__init__(self,parent)
            self.setupUi(self)
            self.BlackLine.clicked.connect(lambda:self.colourclicked("Black",0))
            self.BlueLine.clicked.connect(lambda:self.colourclicked("Blue",0))
            self.RedLine.clicked.connect(lambda:self.colourclicked("Red",0))
            self.GreenLine.clicked.connect(lambda:self.colourclicked("Green",0))
            self.BlackPoint.clicked.connect(lambda:self.colourclicked("Black",1))
            self.BluePoint.clicked.connect(lambda:self.colourclicked("Blue",1))
            self.RedPoint.clicked.connect(lambda:self.colourclicked("Red",1))
            self.GreenPoint.clicked.connect(lambda:self.colourclicked("Green",1))
            self.LinearFit.clicked.connect(lambda:self.plotselect("Linear"))
            self.PolynomialFit.clicked.connect(lambda:self.plotselect("Polynomial"))
            self.NoFit.clicked.connect(lambda:self.plotselect("None"))
            self.NewGraph.clicked.connect(self.newgraph)
        def newgraph(self):
            graphcreatescreen.show()
        def colourclicked(self,colour,identifier):
            global colours
            colours[identifier]=colour
            print(colours)
        def plotselect(self,plottype):
            global plotstyle
            plotstyle = plottype
            print(plotstyle)
```

```
if error != True:
    apptemplate = QtGui.QApplication(sys.argv)
    loginscreen = LoginWindowClass(None)
    registerscreen = RegisterWindowClass(None)
    graphscreen = GraphWindowClass(None)
    graphcreatescreen = GraphCreateClass(None)
    loginscreen.show()
    apptemplate.exec()
```

Explanations:

1-22: wraps a try except clause around initialising database connection and opening files. If the database is missing it prints "error database missing" Otherwise if other files are missing the try fails and the program prints "CRITICAL ERROR : FILES MISSING" and also sets the variable "error" to true for later use

23-26: global variables are assigned for the current user, user privilege, line and point colours, and the method of plotting.

148-158 the methods are called to change the variables required by the radio buttons to their respective values

159-160: method for opening the currently unfunctional graph creation screen

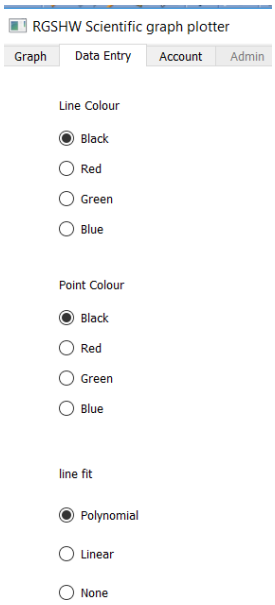
161-164: changes the global variable colours when the buttons are pressed. Takes the arguments colour, and then identifier: 1=point 0 = line. It changes the colour for that identifier. It prints the result for testing.

165-169: changes the global variable plotstyle when the buttons are pressed.

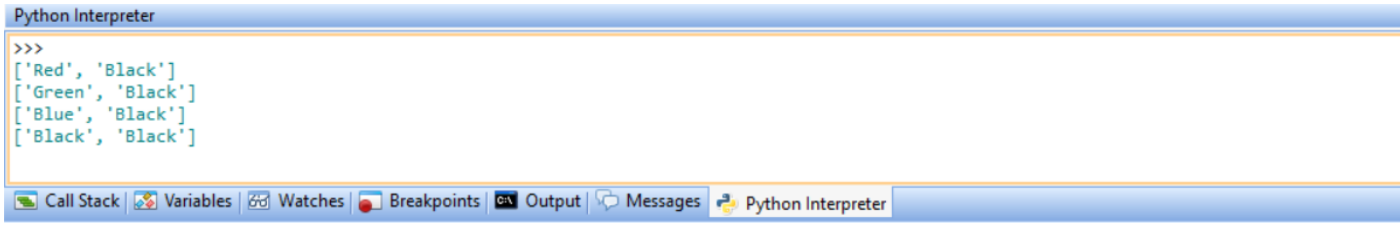
174-181: If there is an error previously the graphics objects are not created and therefore there should be no errors

Testing if radio buttons work

I shall press the radio buttons in the order “red, green, blue, black” for colours and “polynomial, none, linear, polynomial” on line fit.

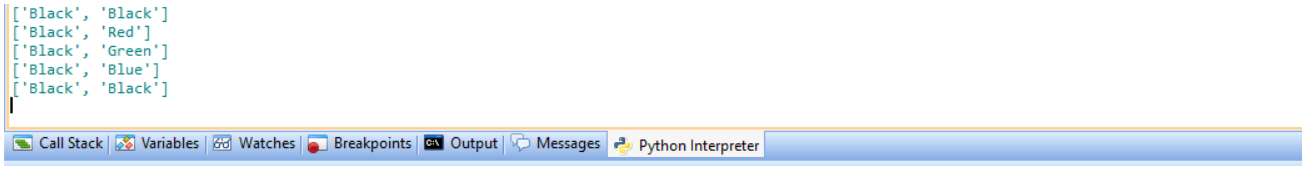


Line colour test:



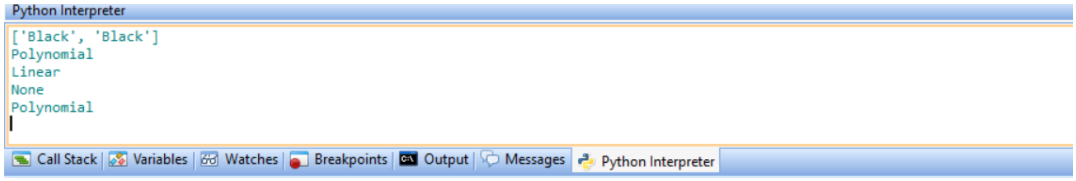
This is as expected

Point colour test:



This is as expected

Fit style test:

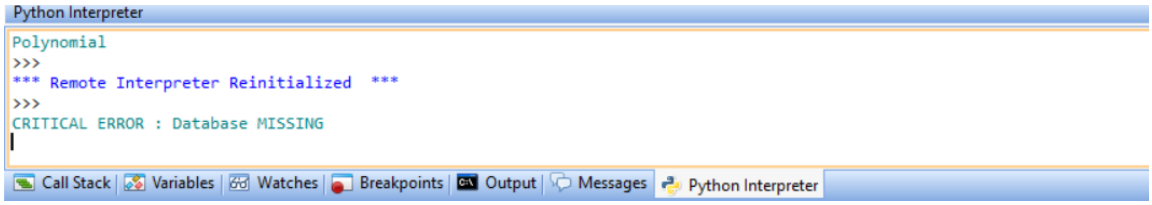


This is as expected

Therefore, these tests have passed.

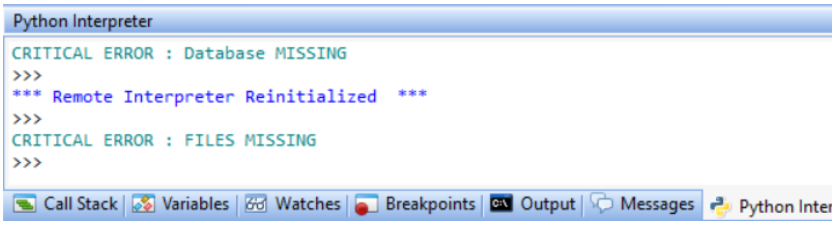
Testing the program when files are missing

I will first remove the database file and see what the response is



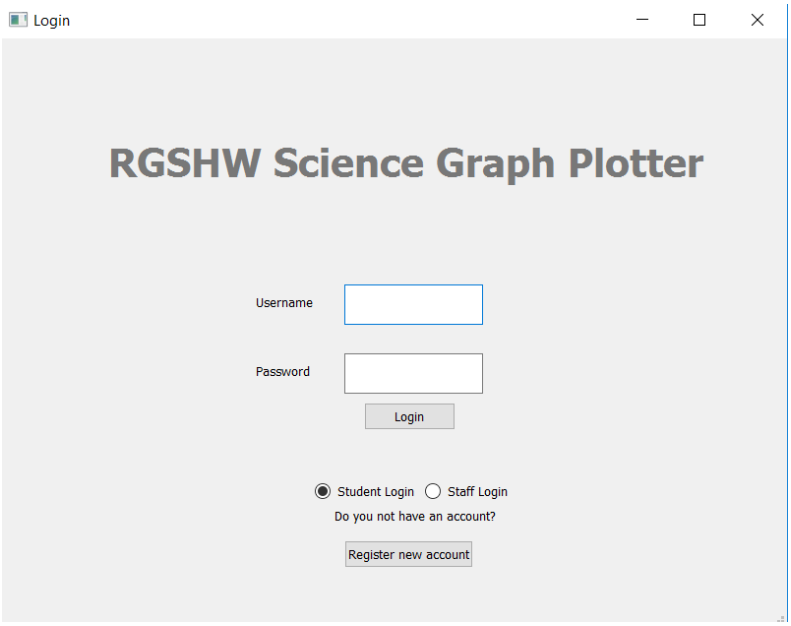
This test is therefore a success

I will now remove a UI file and replace the database file and see what happens



This is therefore a success.

I will now ensure that all the files are there



The program works as normal; therefore, this version has passed all tests and is complete

Version 0.7

Aim: To allow the teachers to add a new class to the database. To create a code for the creation of a graph or class.

```
28 def codecreate():
29         newcode = ""
30         for i in range(10):
31             newcode = newcode + str(random.choice(string.ascii_uppercase + string.digits))
32         newcode = str(newcode)
33         cur.execute("select count(GraphCode) from Graphs where GraphCode=?;",(newcode,))
34         tempfetchgraph = cur.fetchall()[0]
35         cur.execute("select count(ClassCode) from Classes where ClassCode=?;",(newcode,))
36         tempfetchclass = cur.fetchall()[0]
37         if str(tempfetchgraph).strip() == "(0,)" and str(tempfetchclass).strip() == "(0,)":
38             return newcode
39         else:
40             codecreate()
```

Code creation – I wanted to add a code generator that works for both classes and graph.

Line 29 – defines the new code as an empty string

30-31 repeatedly adds a random object from the list of every letter and digit combined to the new code until it is 10 in length

32: converts the code into a string

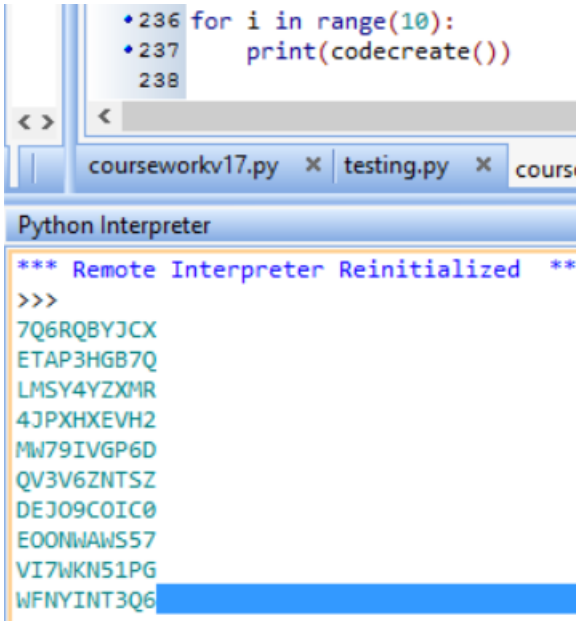
33-36: searches for graph/class with the same code.

37-38: if the code is unique then the code is returned by the function for use

39-40: otherwise the function is called again to return a new value.

Testing code create

To test the creation of codes I must first check that it can generate a 10 digit code. I must check that it is unique later when it is included when creating graphs.



These codes are unique and contain letters and numbers and are 10 digits long. Therefore, the program is creating codes as expected.

Graph creation

```
185 class GraphCreateClass(QtGui.QMainWindow,graphcreate):
186     def __init__(self,parent=None):
187         QtGui.QMainWindow.__init__(self,parent)
188         self.setupUi(self)
189         self.Create.clicked.connect(self.graphcreation)
190     def graphcreation(self):
191         date = str(datetime.datetime.now().year)+"-"+str(datetime.datetime.now().month)+"-"+str(datetime.datetime.now().day)
192         error = False
193         maxminlist= [self.MinX,self.MinY,self.MaxX,self.MaxY]
194         titlelist = [self.Title,self.XTitle,self.YTitle]
195         for each in titlelist:
196             if len(each.text())<3 or len(each.text())>25:
197                 self.Error.setText("Titles must be between 3 and 25 characters long")
198                 error = True
199         if error == False:
200             for each in maxminlist:
201                 try:
202                     float(each.text().strip())
203                 except:
204                     self.Error.setText("Max and min values must be real numbers")
205                     error = True
206         if error == False:
207             if self.MinX.text() >= self.MaxX.text() or self.MinY.text()>=self.MaxY.text():
208                 error = True
209                 self.Error.setText("Maximum must be greater than Minimum")
210         if error == False:
211             dbvals = []
212             self.Error.setText("")
213             for each in maxminlist:
214                 dbvals.append(each.text())
215                 each.setText("")
216             for each in titlelist:
217                 dbvals.append(each.text())
218                 each.setText("")
219             newcode = codecreate()
220             cur.execute("INSERT INTO Graphs(Title,XTitle,YTitle,XValMin,XValMax,YValMin,YValMax,DateCreated,GraphCode) Values (?,?,?,?,?,?,?,?,?,?);",(dbvals[4],dbvals[5],dbvals[6],float(dbvals[0]),
221             con.commit()
222             graphscreen.changetableview("select * from Graphs order by GraphID Desc")
223             self.hide
```

line 220 continued:

```
dbvals[0]),float(dbvals[2]),float(dbvals[1]),float(dbvals[3]),date,newcode,))
```

Explanation:

189: When the create graph button is pressed it calls the method for validating the details.

191: gets the current date and stores it in the way required by sql

193-194: creates a list of the objects required to be checked that are similar.

195-198: Checks each of the titles to check that each is between 3 and 25 characters. If they are not an error is created and displayed to the user.

199-205: checks that each of the max and min values are floats. If they are not an error is shown to the user

206-209: checks that the max values are greater than min values. If they are not then an error is shown

210: if no errors are found the graph is added to the database.

213-218 adds the max and mins and titles to a list for database addition.

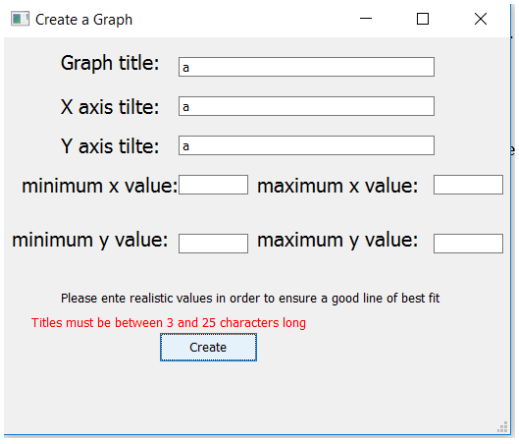
219 – generates a code for the graph using codecreate

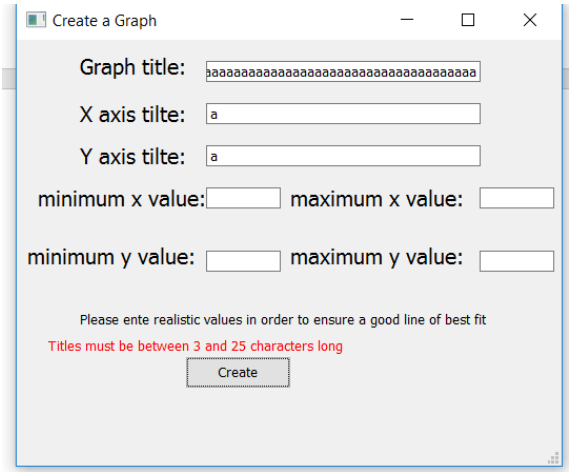
220-221: uses these details to add the graph to the database

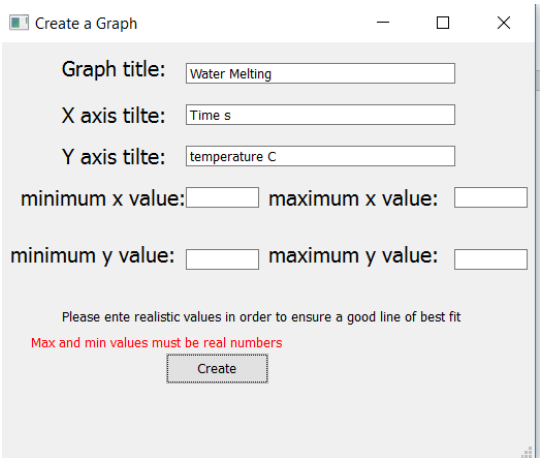
222: changes the table view. Currently I make it show all graphs to show it has been added.

223: closes the window.

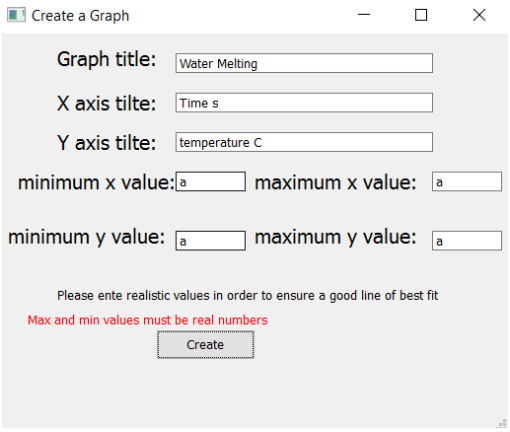
Testing that the titles are the right length.

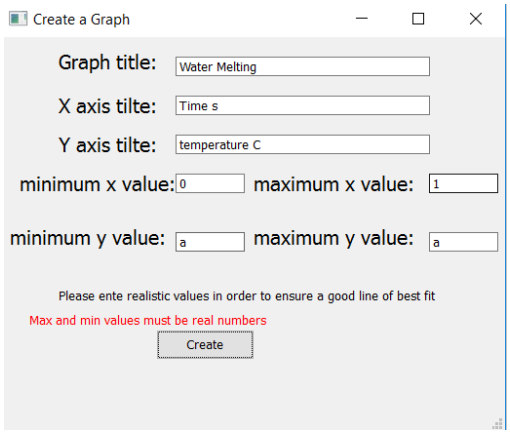


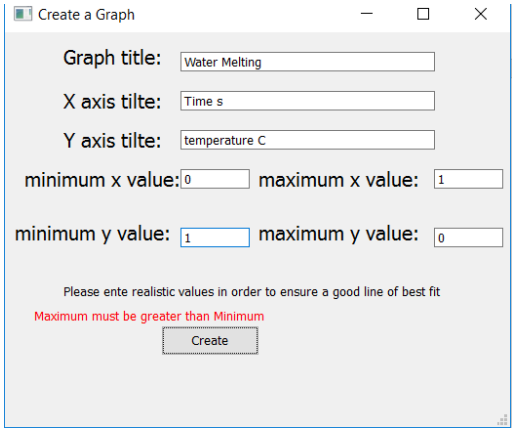




The test has been successful as the incorrect values have returned an error, whereas the correct ones haven't meaning that the algorithm is successful

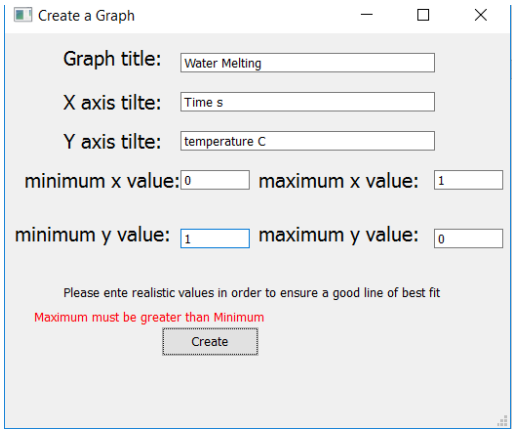


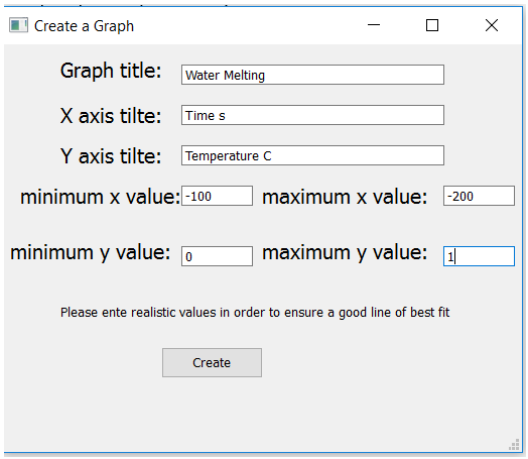




The test has been successful as the incorrect values have returned an error, whereas the correct ones haven't meaning that the test has passed.

Now I will test to see if the max>min validation works

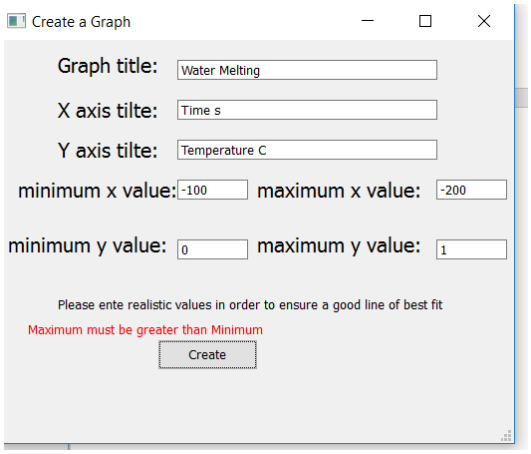


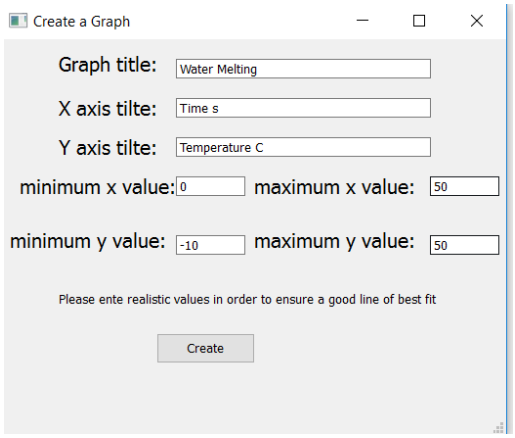


This was accepted and was entered into the database, which is incorrect.

I noticed the program was just comparing the text rather than the float values. It was changed to

```
if float(self.MinX.text()) >= float(self.MaxX.text()) or float(self.MinY.text())>=float(self.MaxY.text()):
```



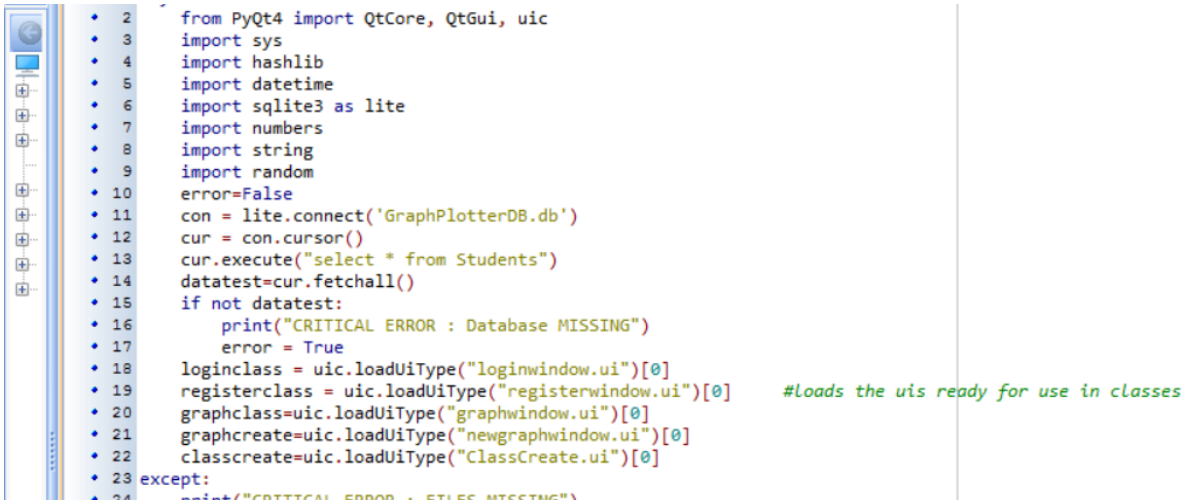


GraphID	Title	XTitle	YTitle	XValMin	XValMax	YValMin	YValMax	DateCreated	GraphCode
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
11	Melting Water	Time s	Temperature C	0.0	50.0	-10.0	50.0	2017-4-14	ETPRV2V00P

This test is now successful as invalid data has been rejected and the valid data has been accepted.

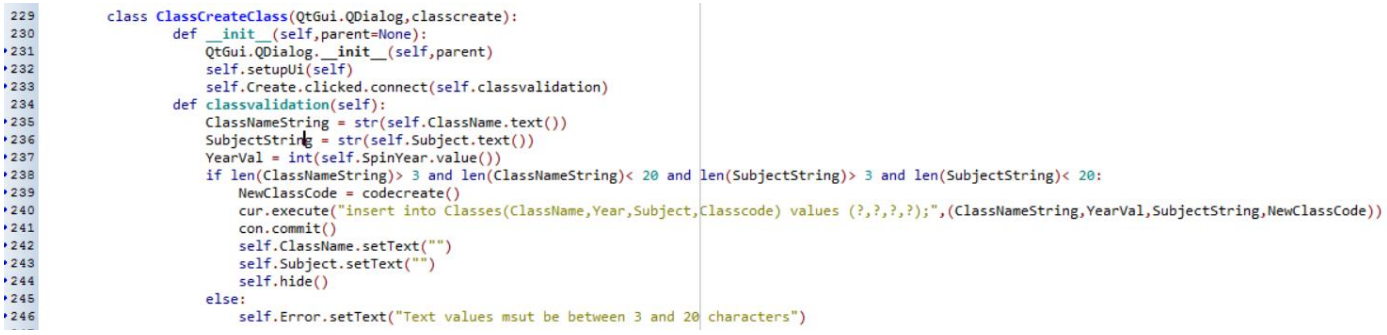
Version 0.8

Aim: add class creation functionality.



```
2     from PyQt4 import QtCore, QtGui, uic
3     import sys
4     import hashlib
5     import datetime
6     import sqlite3 as lite
7     import numbers
8     import string
9     import random
10    error=False
11    con = lite.connect('GraphPlotterDB.db')
12    cur = con.cursor()
13    cur.execute("select * from Students")
14    datatest=cur.fetchall()
15    if not datatest:
16        print("CRITICAL ERROR : Database MISSING")
17        error = True
18    loginclass = uic.loadUiType("loginwindow.ui")[0]
19    registerclass = uic.loadUiType("registerwindow.ui")[0]     #loads the uis ready for use in classes
20    graphclass=uic.loadUiType("graphwindow.ui")[0]
21    graphcreate=uic.loadUiType("newgraphwindow.ui")[0]
22    classcreate=uic.loadUiType("ClassCreate.ui")[0]
23 except:
```

line 22: loads class create UI file



```
229     class ClassCreateClass(QtGui.QDialog,classcreate):
230         def __init__(self,parent=None):
231             QtGui.QDialog.__init__(self,parent)
232             self.setupUi(self)
233             self.Create.clicked.connect(self.classvalidation)
234         def classvalidation(self):
235             ClassNameString = str(self.ClassName.text())
236             SubjectString = str(self.Subject.text())
237             YearVal = int(self.SpinYear.value())
238             if len(ClassNameString)> 3 and len(ClassNameString)< 20 and len(SubjectString)> 3 and len(SubjectString)< 20:
239                 NewClassCode = codecreate()
240                 cur.execute("insert into Classes(ClassName,Year,Subject,Classcode) values (?,?,?,?);",(ClassNameString,YearVal,SubjectString,NewClassCode))
241                 con.commit()
242                 self.ClassName.setText("")
243                 self.Subject.setText("")
244                 self.hide()
245             else:
246                 self.Error.setText("Text values msut be between 3 and 20 characters")
```

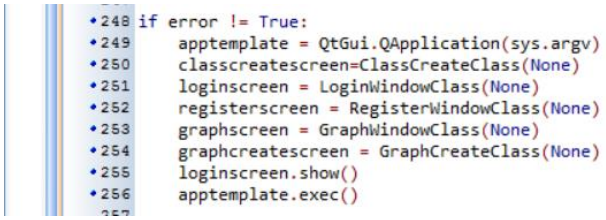
229-232: initialises the class “classcreateclass” which handles the class create window

233: when the classcreate button is pressed the classvalidation method is called

235-237: fetches the text values for validation

238-244: checks that the lengths of both the class name and the subject name are between 4 and 19 characters. If they are the class is added to the database and the values are set to blank for re-entry. The window is then hidden.

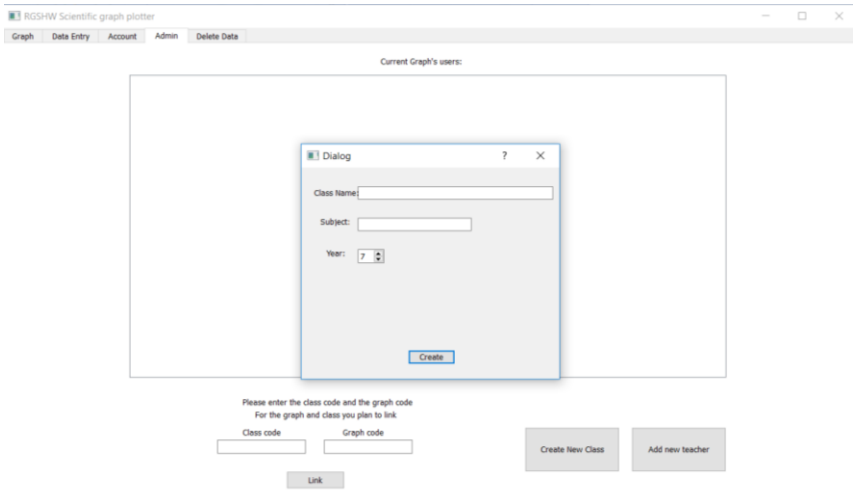
245-246: otherwise errors are shown.



```
248 if error != True:
249     apptemplate = QtGui.QApplication(sys.argv)
250     classcreatescreen=ClassCreateClass(None)
251     loginscreen = LoginWindowClass(None)
252     registerscreen = RegisterWindowClass(None)
253     graphscreen = GraphWindowClass(None)
254     graphcreatescreen = GraphCreateClass(None)
255     loginscreen.show()
256     apptemplate.exec()
```

250: creates an object of the classcreateclass class for use as a window.

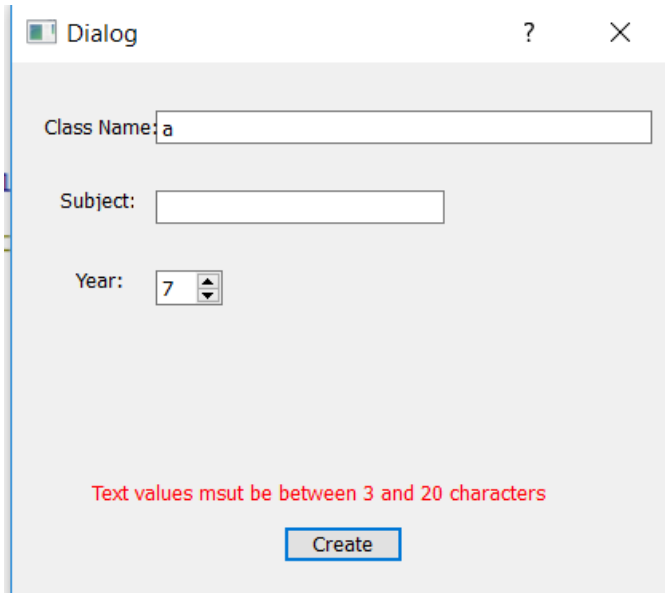
Testing that the class create screen opens as expected.



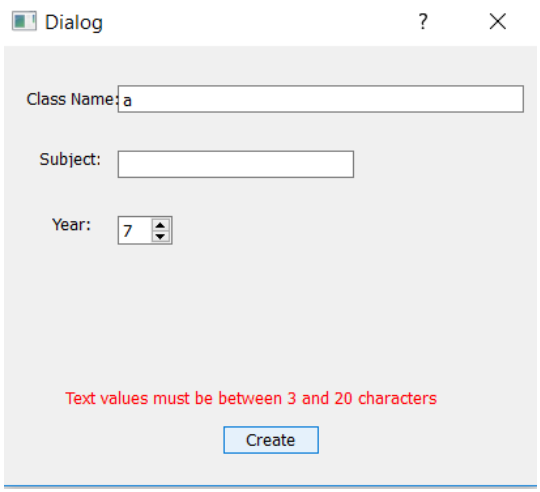
When the create new class button is pressed the window opens. Spamming this button has no adverse effects.

Testing that the validation is working correctly

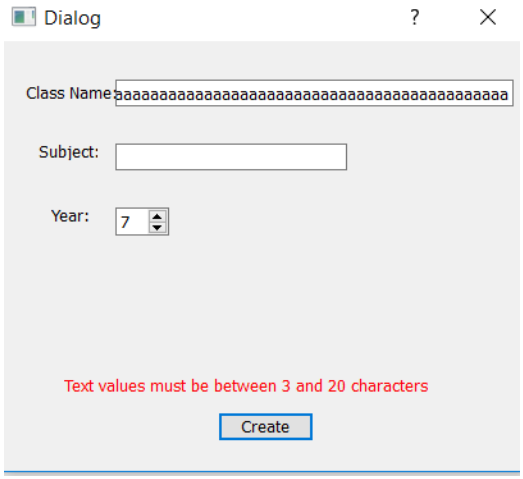
I will first test that the class name and subject name are validated to be of the correct size

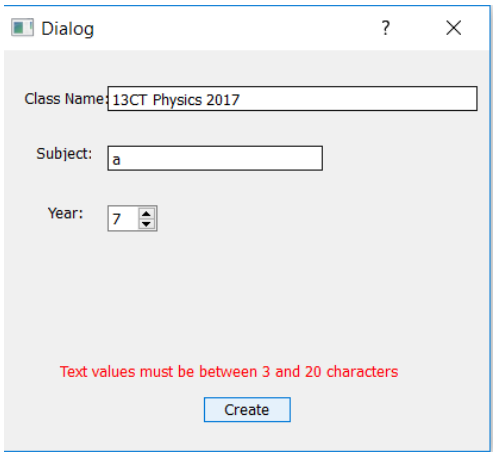


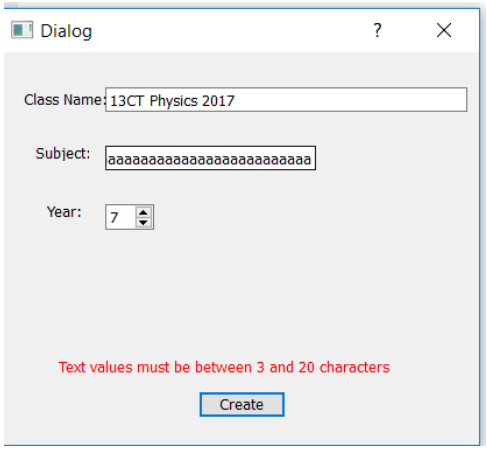
I notice a typo and replace “msut” with “must”

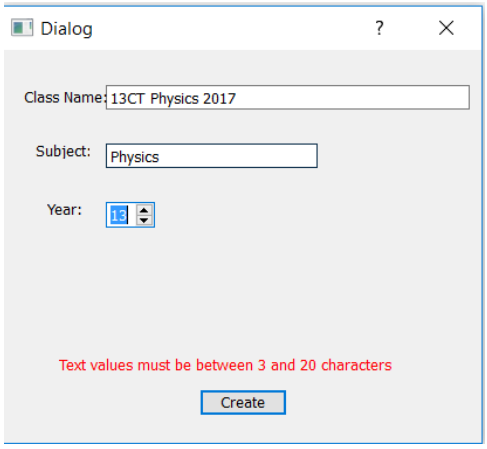


This results in the expected error message now.









Note: error remains from before

Table: Classes

	ClassID	ClassName	Year	Subject	ClassCode
	Filter	Filter	Filter	Filter	Filter
1	3	13CT Physics ...	13	Physics	GWYN9BK3I5

The value has been entered into the database. Therefore, this test has been successful.

I was also unable to produce any adverse effects from button mashing.

Version 0.9

Aim: To add graph table update: for viewing by students.

```
161    class GraphWindowClass(QtGui.QMainWindow,graphclass):
162        def __init__(self,parent=None):
163            self.currentgraph=-1
164            self.row = -1
165            global currentuser
166            global userid
```

163-164: sets the variables for the selected graph and row to -1. This denotes to the program that none have been selected.

```
167         self.GraphSelectError.hide()
168         self.model = QtGui.QStandardItemModel(self)
169         self.GraphTable.setModel(self.model)
```

167-169: hides error label, sets the model of the graph table to standard item so that it can be updated.

```
199     def rowupdate(self):
200         self.row = self.GraphTable.currentIndex().row()
```

Method of graphwindowclass. When a row is clicked on the table this method is called. It updates the current row to the one currently clicked.

```
207     def graphopener(self):
208         if self.row!=-1:
209             self.currentgraph = str(self.model.data(self.model.index(self.row,0)))
210             self.GraphSelectError.hide()
211             self.CurrentGraphLabel.setText("Graph " + self.currentgraph + " Selected")
212         else:
213             self.GraphSelectError.show()
```

207: graphopener is the method for opening a selected graph when the button is pressed.

208-211: If a row is selected the current graph label changes to display the current graph. The current graph is set to the graph id and the error is hidden

212-213: otherwise an error is shown.

```
215     def changetableview(self,query):
216         cur.execute(query)
217         data = cur.fetchall()
218         self.model.clear()
219         self.tableupdate(data)
220
221     def tableupdate(self,data):
222         for row in data:
223             items = []
224             for field in row:
225                 items.append(QtGui.QStandardItem(str(field)))
226             self.model.appendRow(items)
227         titles = ["GraphID","Title","X Title","Y Title","Min X","Max X","Min Y","max Y", "Date Created", "Graph Code"]
228         self.model.setHorizontalHeaderLabels(titles)
229         self.proxy = QtGui.QSortFilterProxyModel(self)
230         self.proxy.setSourceModel(self.model)
```

215-219: executes the query and receives the results. Resets the table model and then calls the method tableupdate to change the model

222-226: adds each row to the table view model

227-230: changes the titles and updates the table view

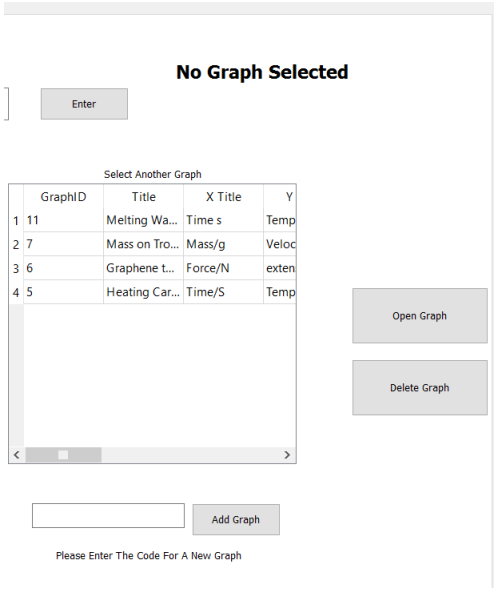
```
84  if permissions =='Students':
85      graphscreen.changetableview(("select Graphs.* from Graphs INNER JOIN StudentPermissions ON Graphs.GraphID = StudentPermissions.GraphID where StudentPermissions.StudentID = ? order by GraphID Desc;"
86      #replace function used as impossible to use as function without and the userid is a result from a query which is automatically set and not user editable. no Chance of SQL injection
87      graphscreen.Graph.setTabEnabled(3,False)
88      graphscreen.Graph.setTabEnabled(4,False)
89      graphscreen.NewGraph.hide()
90      graphscreen.DeleteGraph.hide()
91  else:
92      graphscreen.changetableview("select * from Graphs order by GraphID Desc")
```

When someone logs in the table view is updated. If it is a student the graphs which they have permissions for in the database are shown. If the user is staff all the graphs will be shown. They are shown newest first to make it easier to find graphs.

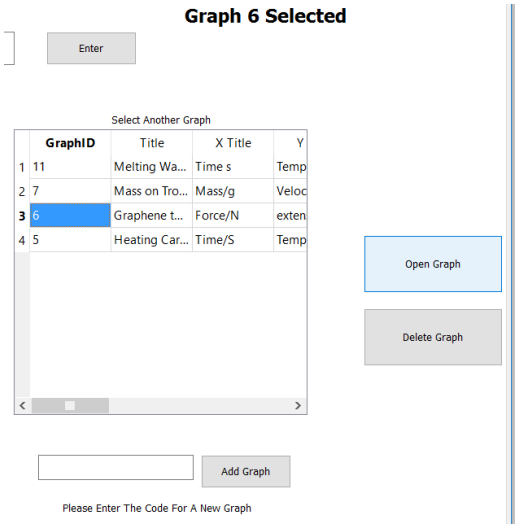
```
• 85  Graphs.* from Graphs INNER JOIN StudentPermissions ON Graphs.GraphID = StudentPermissions.GraphID where StudentPermissions.StudentID = ? order by GraphID Desc;").replace("?",userid))
```

Because the SQL is not called and rather the query is just stored I am forced to use a replace query. This poses no chance of SQL injection as the userID is auto generated and not user created.

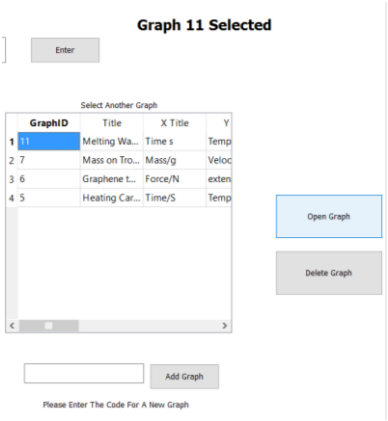
Testing graph loading.



When the program is opened as a teacher all graphs are shown as expected.



When the graph 6 is opened the open graph function changes the text to “graph 6 selected” as expected. This means that row update and graphopener must work. Also as the graphs have been loaded tableupdate and changetableview must be correct.



Opening another graph afterwards works as normal

Spamming buttons also proves to produce no errors.

Version 0.10

Aim: To allow users to add points to a graph.

First off before completing this task I noticed that I was frequently removing and stripping the same parts of the tuple returned by a fetchall function and requiring it as a string. I therefore decided to create a global function that takes the results of the query and returns a string with these modifications.

```
48 def fetchallstripper(oldtuple):
49     newstring = str(oldtuple)
50     newstring = newstring.replace("(","")
51     newstring = newstring.replace(")","")
52     newstring = newstring.replace(",","")
53     return newstring
```

I know that the function works as it is simple string replacement and the same code has been used previously in a form that was not modular, however it will be used in the point entry method so it will be tested as part of that.

```
87                 global currentuser
88                 currentuser=username
89                 global userid
90                 cur.execute("select StudentID from Students where Username=?;",(currentuser,))
91                 userid=fetchallstripper(cur.fetchall()[0])
```

Here in the function loginclick this new function is now used to change the global variable “userid” to the value of the user logging in.

This is then used for the relationships in the database as a foreign key.

```
245         def pointenter(self):
246                 global userid
247                 xpoint = self.XValueEdit.text()
248                 ypoint = self.YValueEdit.text()
249                 if self.currentgraph== -1:
250                     self.PointEntryError.setText("No Graph is selected")
251                 else:
252                     cur.execute("Select XValMin, XValMax, YValMin, YValMax from Graphs where GraphID =?;",(self.currentgraph))
253                     comparisondata= fetchallstripper(cur.fetchall()[0])
254                     comparisonlist = comparisondata.split(" ")
255                     try:
256                         xpoint = float(xpoint)
257                         ypoint = float(ypoint)
258                         for i in range(len(comparisonlist)):
259                             comparisonlist[i] = float(comparisonlist[i])
260                         if xpoint >= comparisonlist[0] and xpoint <= comparisonlist[1] and ypoint > comparisonlist[2] and ypoint < comparisonlist[3]:
261                             print("running sql")
262                             cur.execute("INSERT INTO DataPoints(GraphID,XVal,YVal,UserID) VALUES (?,?,?,?);",(self.currentgraph,xpoint,ypoint,userid))
263                             print("sql fine")
264                             con.commit()
265                             self.PointEntryError.setText("")
266                             self.YValueEdit.setText("")
267                             self.XValueEdit.setText("")
268                         else:
269                             self.PointEntryError.setText("The values given are not within the range specified")
270                     except:
271                         self.PointEntryError.setText("The entry is not a number")
```

The procedure pointenter is a method of the class “graphwindowclass” It uses the global userid to add the entered values to the graph.

Lines 247-248 retrieve the values from the UI for x value and y value.

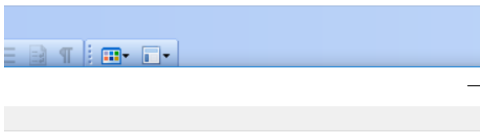
250-251: if there is no graph selected an error is shown.

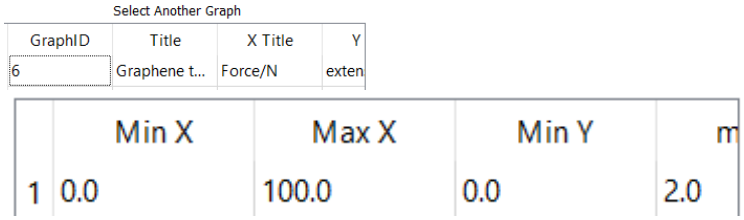
252-254: forms a list of the min and max values for x and y

260: Checks if the values are within the limits set. If not an error message is given, otherwise the values are added to the database and errors are removed.

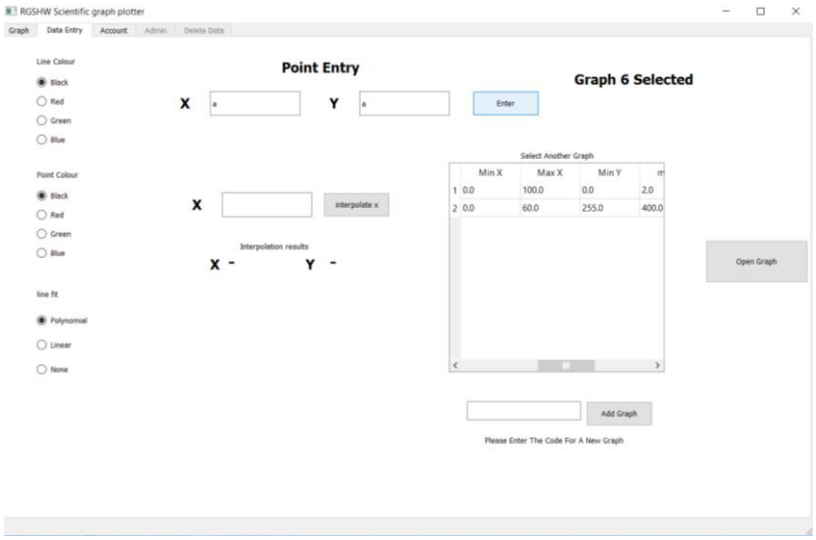
271: If the procedure creates an error then the numbers entered must be a float due to the xpoint and ypoint not being convertible to float.

Testing

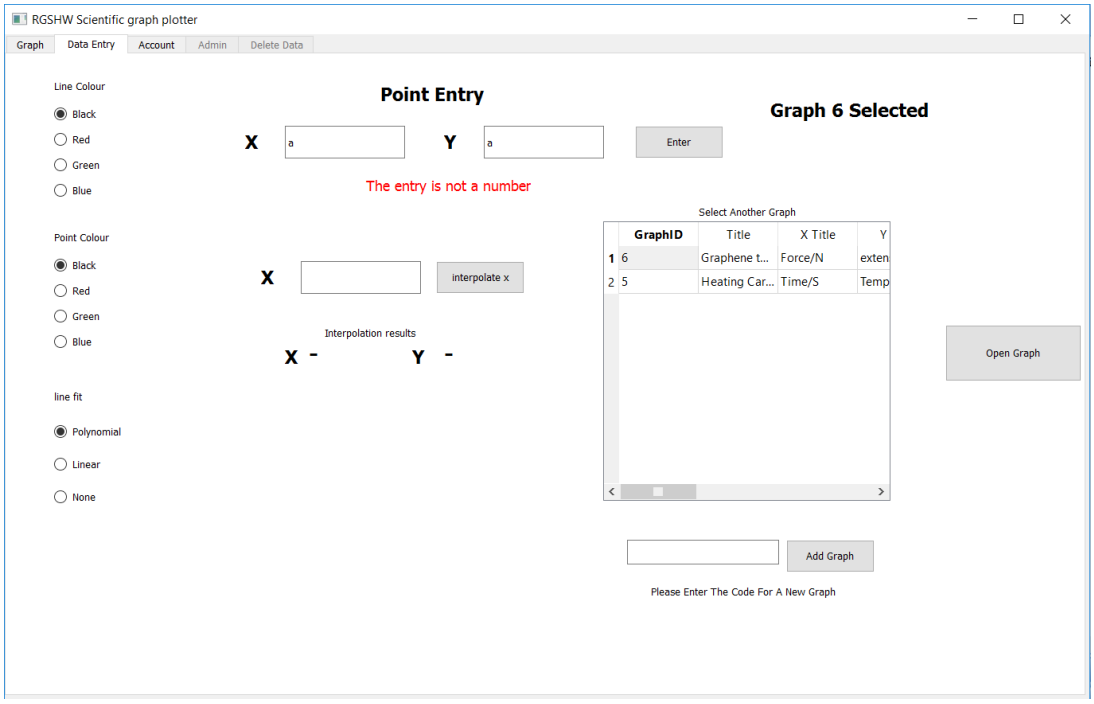




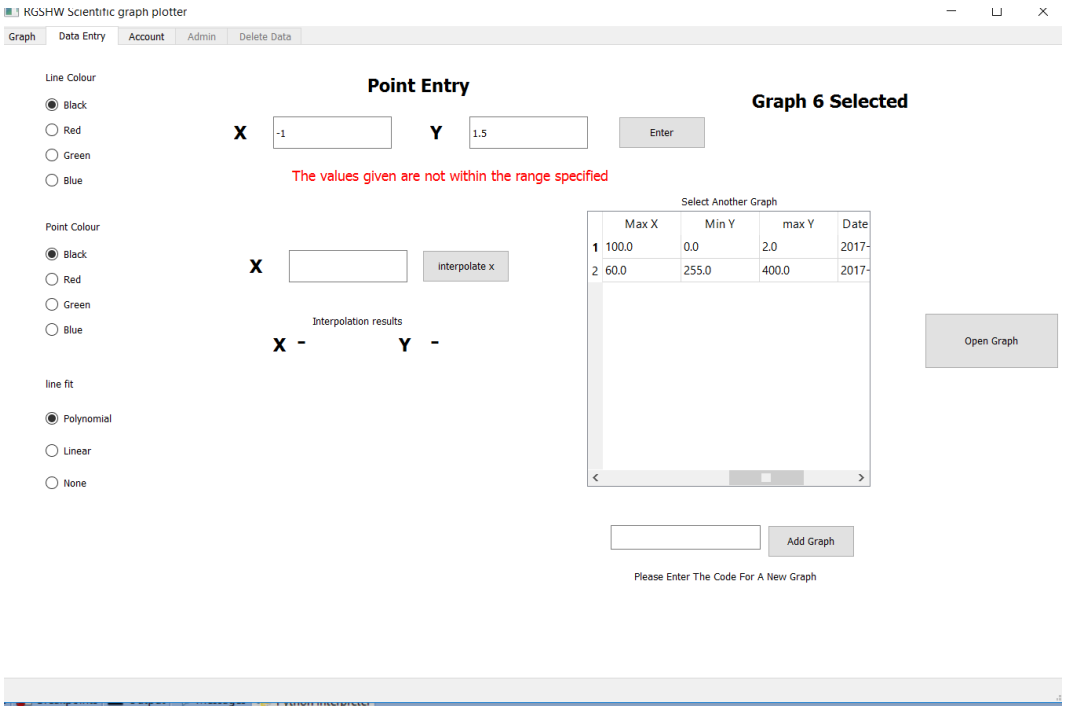
Graph 6 is selected with min x of 0, maxx of 100, miny of 0, max y of 2



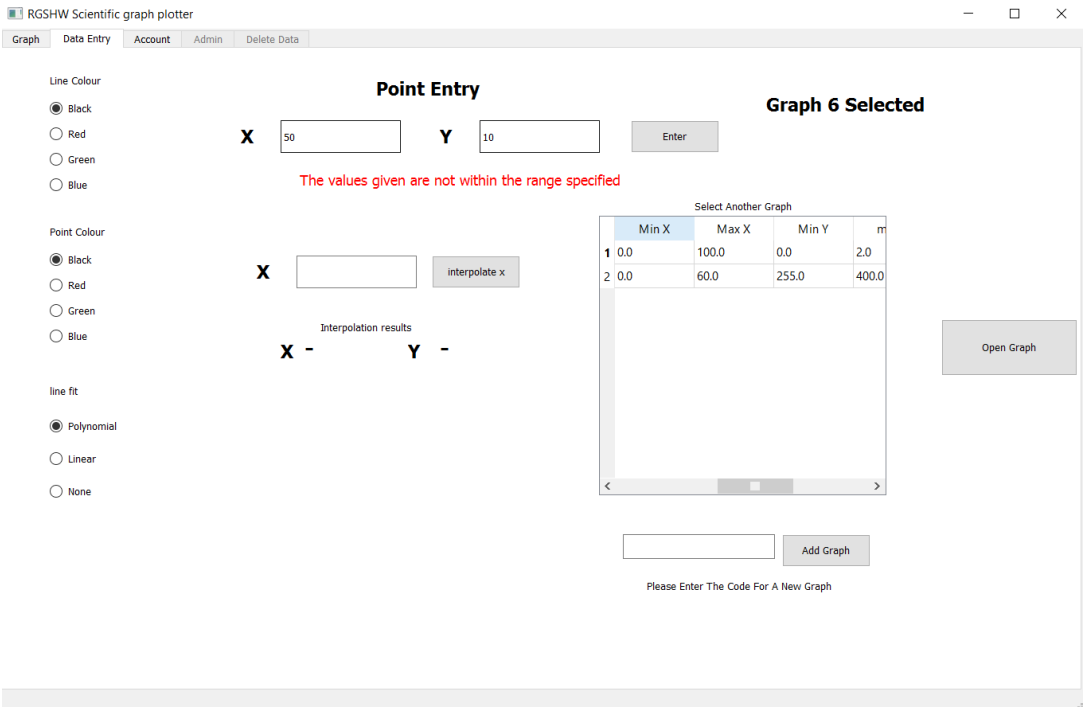
Initially when enter was pressed it produced no results. This was because I forgot to make the button press call the function.



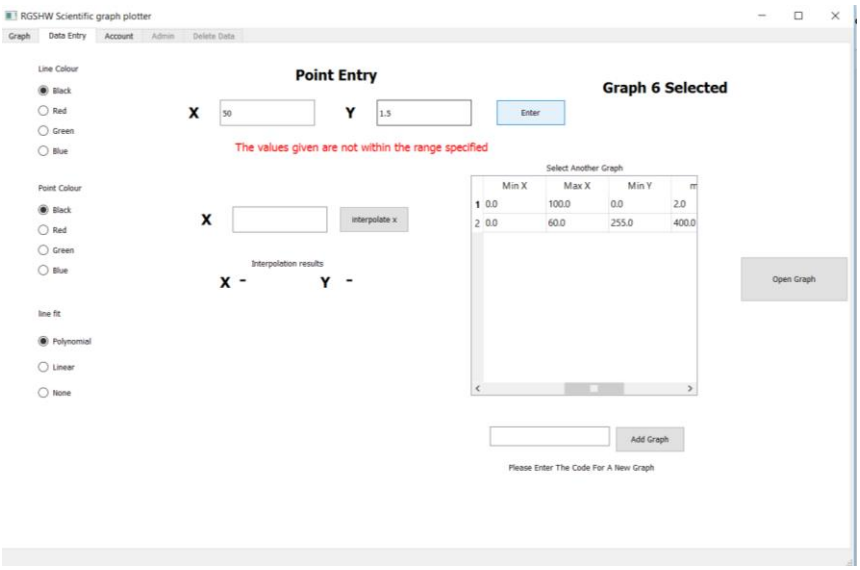
Test successful – text provides error.



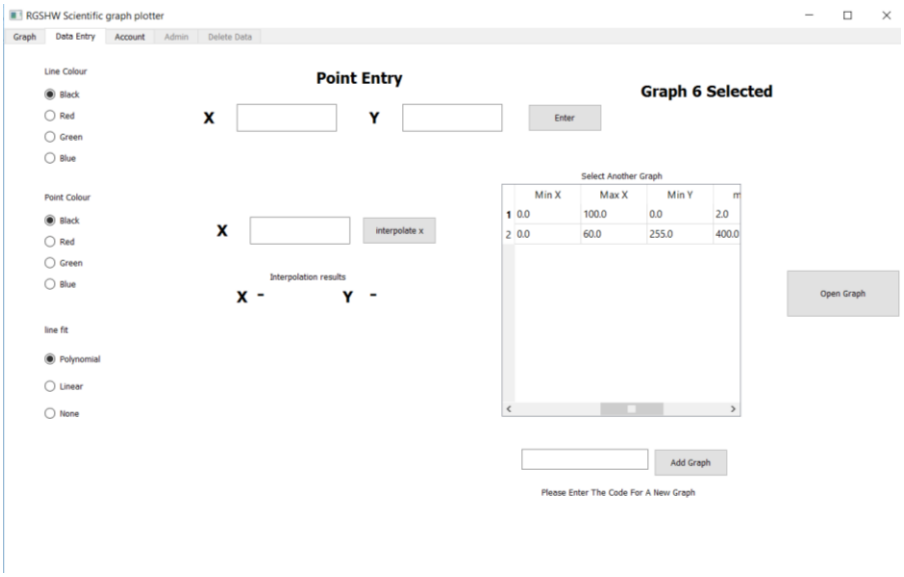
Success – x out of range = error



Y out of range – error. Therefore, the test is successful



Error showing from last time.



This input clears the screen, denoting that the input was accepted.

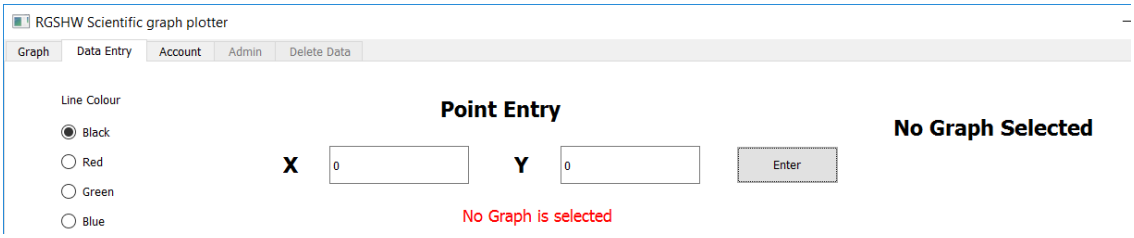
	DataID	GraphID	XVal	YVal	UserID
	Filter	Filter	Filter	Filter	Filter
1	11	6	50	1.5	7

The new data point is added to graph 6, with the correct xval and yval. The userID is 7, corresponding to the user used – quicklog(used for quick test login)

Table: Students

	StudentID	Username	Password	Forename	Surname	DOB
	Filter	Filter	Filter	Filter	Filter	Filter
1	2	Williamsverys...	31bade0c5edf...	William	Taylor	1999-4-8
2	5	RyanB1999	e06382b4063...	ryan	bown	1999-9-3
3	7	quicklog	17053616410...	log	quick	1999-9-3

This means that the test is successful



When no graph has been selected the no graph error shows. This is a success.

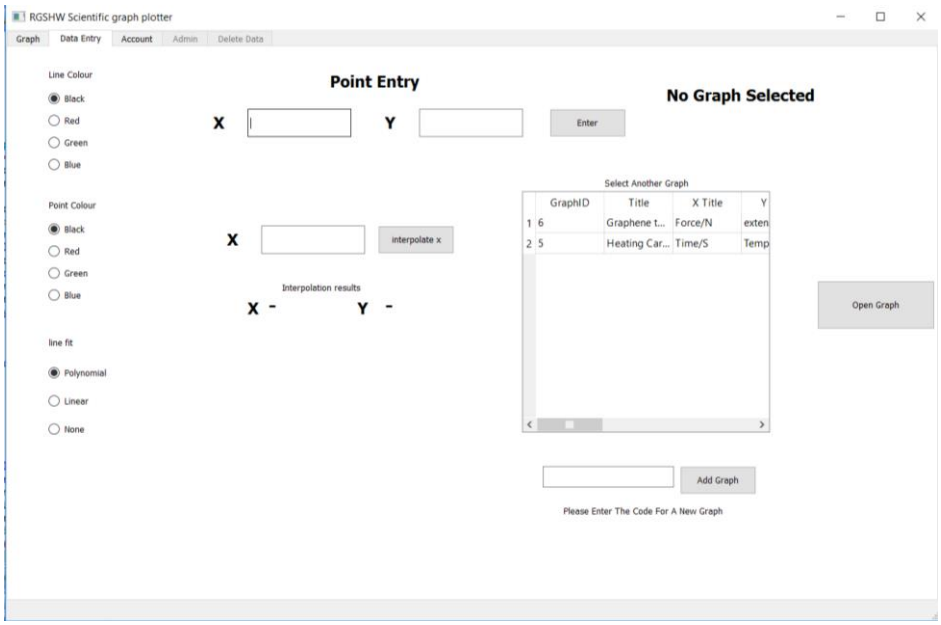
Version 0.11

Aim: To allow users to join graphs using a graph code supplied by a teacher.

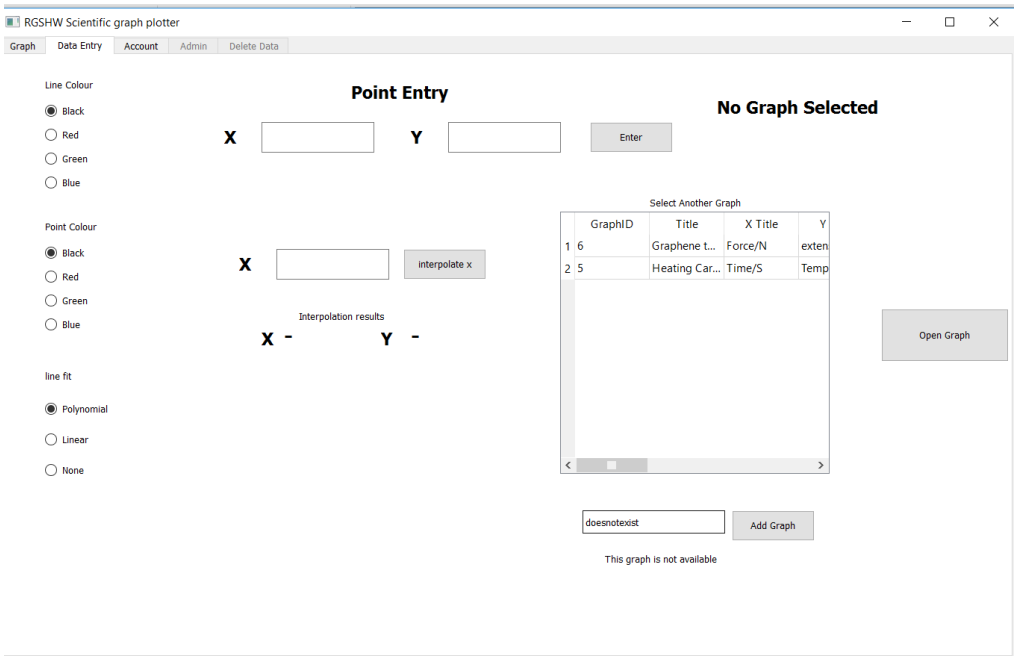
```
def addgraph(self):
    error = False
    trialcode = self.AddGraphEdit.text()
    cur.execute("select GraphID from Graphs where GraphCode=?;",(trialcode,))
    try:
        AddGraphResult = fetchallstripper(cur.fetchall()[0])
    except:
        error = True
        self.NewGraphStatus.setText("This graph is not available")
    if error == False:
        global currentuser
        cur.execute("select StudentID from Students where Username=?;",(currentuser,))
        userid=fetchallstripper(cur.fetchall()[0])
        try:
            cur.execute("select PermissionID from StudentPermissions where GraphID=? and studentID=?;",(AddGraphResult,userid,))
            existencechecker = fetchallstripper(cur.fetchall()[0])
            self.NewGraphStatus.setText("You already have permission for this graph")
        except:
            cur.execute("insert into StudentPermissions(GraphID,StudentID)values(?,?);",(AddGraphResult,userid,))
            con.commit()
            self.changetableview("""select Graphs.* from Graphs INNER JOIN StudentPermissions
                                    ON Graphs.GraphID = StudentPermissions.GraphID where StudentPermissions.StudentID = ? order by GraphID Desc;""".replace("
            self.AddGraphEdit.setText("")
            self.NewGraphStatus.setText("Please Enter The Code For A New Graph")
```

Checks the database for the graph code entered. If it exists get ID, otherwise output error.

If there is no error the current user is loaded and the userID is found. This is then used to add the student to the graph and update the table view with their graphs available.

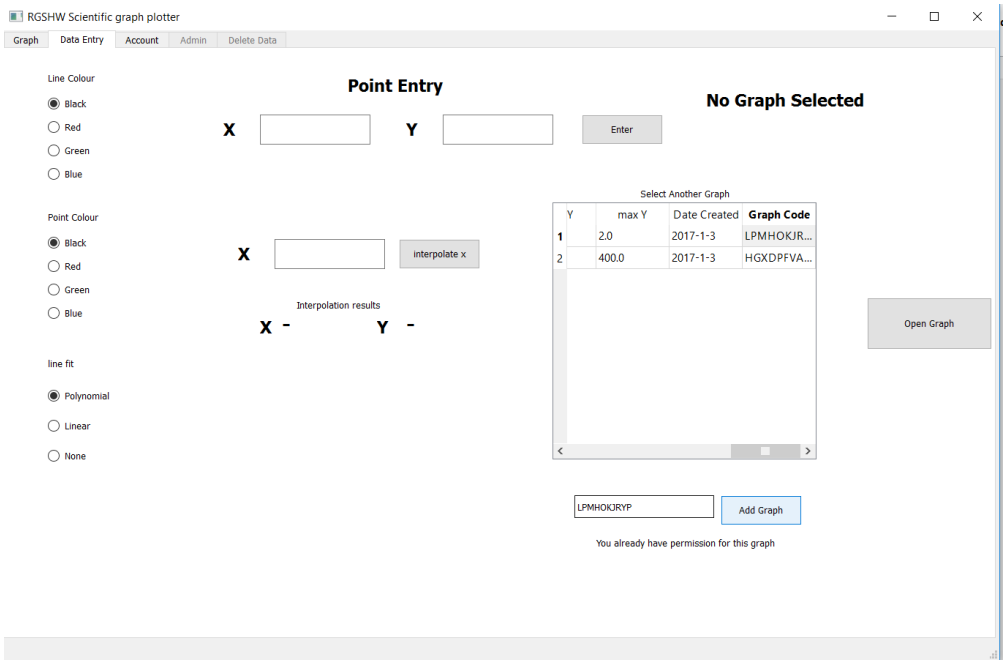


Adding bad graph code.



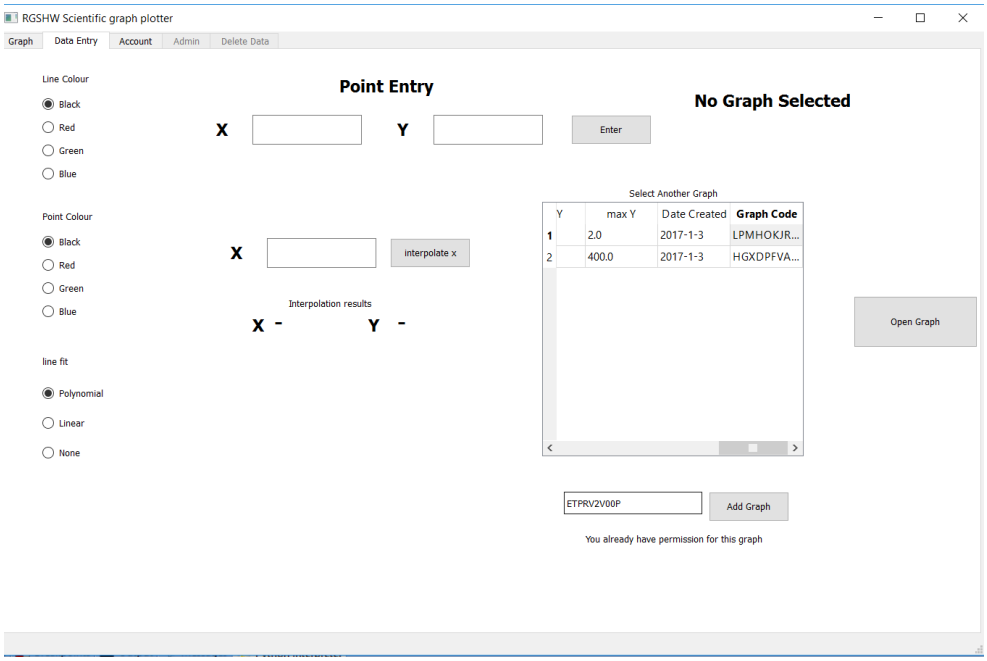
Success

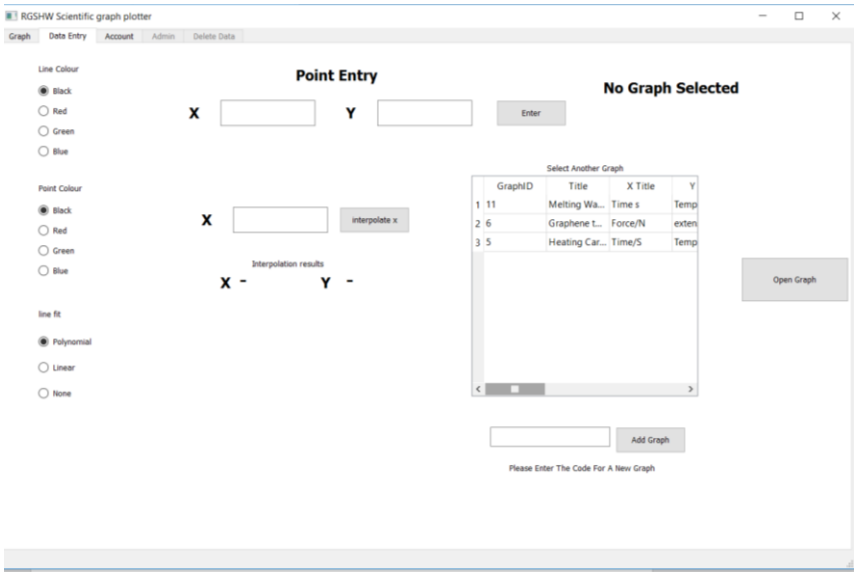
Adding already used graph code



Success.

Adding graph code for the melting water experiment.





Graph added – success.

Version 0.12

Aim : to plot the graphs in javascript

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Graph</title>
</head>
<body>
<canvas id="myCanvas" width="1000" height="800">
An error has occurred in showing javascript.
</canvas>
</body>
</html>
```

The HTML for this is very simple as it consists of creating an empty document with only a 1000x800 canvas. I chose to embed my CSS within the HTML as this is the only CSS being used. If the canvas fails to load it will display an error message.

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "20px Comic Sans MS";
ctx.fillStyle = "red";
ctx.textAlign = "center";
ctx.fillText("Graphene tension", 500, 50);  // change title
ctx.fillText("xtitle", 500, 750); //change xtitle
ctx.translate( 0, 0 );
PI = math.PI
ctx.rotate( PI / 2 );
ctx.fillText("ytitle", 400, -50); // change ytitle
ctx.rotate(-PI / 2 );
points=[[-20,-100],[0,0],[-11,25],[18,-22]]; //change to points
equation=[-6.93345381e-12, 1.00000000e+00,-5.14262410e-10,  2.51822030e-09,-2.07410267e-09,1.11469944e-09]; // change to equation
maxx=0.5; // change to maxx
minx = 0; //change to minx
maxy =0.5; // change to maxy
miny = -0; //change to miny
if (maxx<0){
  maxx=0;
  }
if (minx>0){
  minx=0;
}
if (maxy<0){
  maxy=0;
}
if (miny>0){
  miny=0;
}
rangex=maxx-minx;
rangey=maxy-miny;
plotaxis(rangex,rangey,minx,miny);
for (i = 0; i < points.length; i++) {
  coords=pointtographvalue(points[i][0],points[i][1]);
  plotpoint(coords);
}
```

I set up my canvas to have the font and styles I require. I have hard coded some values for titles and gradients for testing. These will be changed in the actual versions. If the axis would not work I change the minimum and maximum values such that the axis are the min/ max so that calculations can be done accordingly. I then calculate range and call the later functions. I used the capitalisation of “PI” to conform to the standard that constants are capitalised.

```
function plotaxis(){
  ctx.strokeStyle = '#000000';
  xpos = 100+(-minx/rangex)*800;
  ypos = 100+(maxy/rangey)*600;
  ctx.beginPath();
  ctx.lineTo(xpos,100);
  ctx.lineTo(xpos,700);
  ctx.closePath();
  ctx.stroke();
  ctx.beginPath();
  ctx.lineTo(100,ypos);
  ctx.lineTo(900,ypos);
  ctx.closePath();
  ctx.stroke();
}

function pointtographvalue(xvalue,yvalue){
  xpos = 100+((xvalue-minx)/rangex)*800;
  ypos = 100+((maxy-yvalue)/rangey)*600;
  return [xpos,ypos];
}
function plotpoint(positions){
  ctx.strokeStyle = '#000000'; //change to point colour
  xongraph=positions[0];
  yongraph=positions[1];
  ctx.beginPath();
  ctx.lineTo(xongraph+4,yongraph+4);
  ctx.lineTo(xongraph-4,yongraph-4);
  ctx.closePath();
  ctx.stroke();
  ctx.beginPath();
  ctx.lineTo(xongraph-4,yongraph+4);
  ctx.lineTo(xongraph+4,yongraph-4);
  ctx.closePath();
  ctx.stroke();
}
```

Plot axis takes the max ,min, and range values and uses them to calculate pixel values for the axis. They are then plotted.

Pointtographvalue takes a point and returns the x and y coordinates it would be translated to on the canvas.

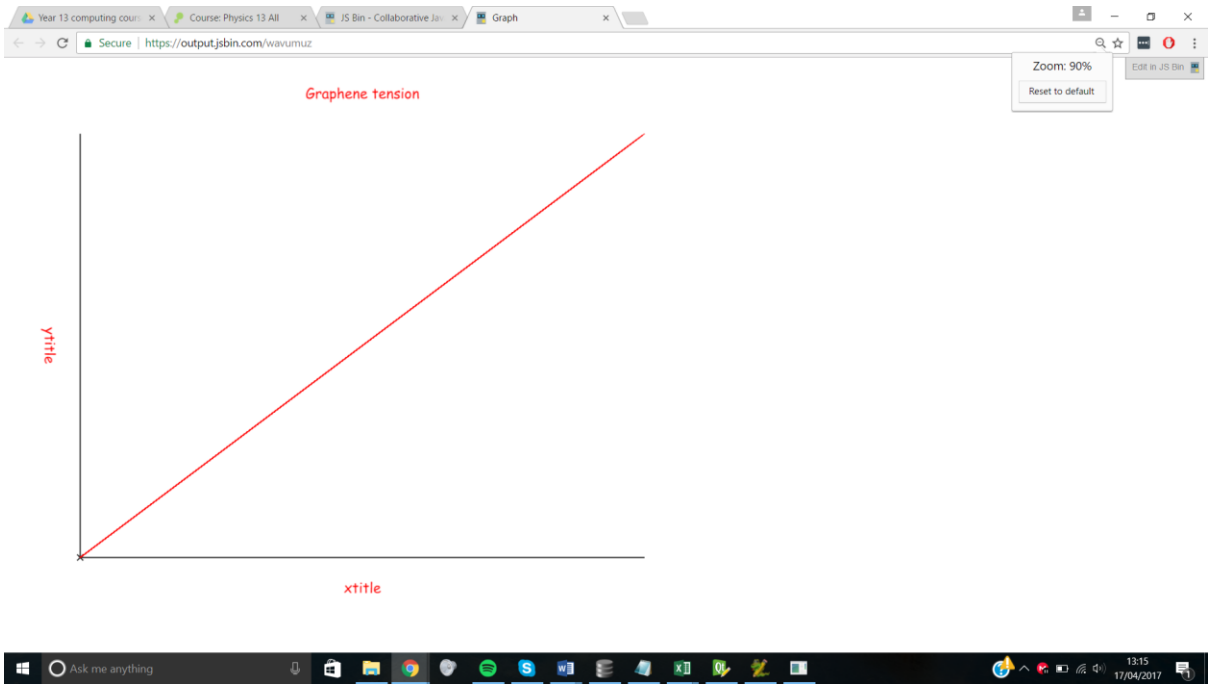
Plotpoint sets a point colour and then finds the x and y positions on the canvas. It then plots a cross on the graph where the point would be.

```
function regressionplot(){
  flag="False"
  ctx.strokeStyle = '#ff0000'; //change to line colour
  ctx.closePath();
  ctx.beginPath();
  for (i = 0; i < 801; i=i+2) {
    yvalue=0;
    xvalue=((i*rangex)/800)+minx;
    for (j = 0; j < equation.length; j++) {
      yvalue=yvalue+Math.pow(xvalue, (5-j))*equation[j];
    }
    if (yvalue<=maxy && yvalue>=miny){
      if (flag == "above"){
        ctx.closePath();
        ctx.beginPath();
        ctx.lineTo(pointtographvalue(xvalue,yvalue)[0],100);
      }
      if (flag == "below"){
        ctx.closePath();
        ctx.beginPath();
        ctx.lineTo(pointtographvalue(xvalue,yvalue)[0],700);
      }
      ctx.lineTo(pointtographvalue(xvalue,yvalue)[0],pointtographvalue(xvalue,yvalue)[1]);
      ctx.stroke();
      flag ="False";
    }
    else{
      if (yvalue>maxy){
        flag="above";
      }
      else{
        flag = "below";
      }
      ctx.closePath();
      ctx.beginPath();
    }
  }
}
```

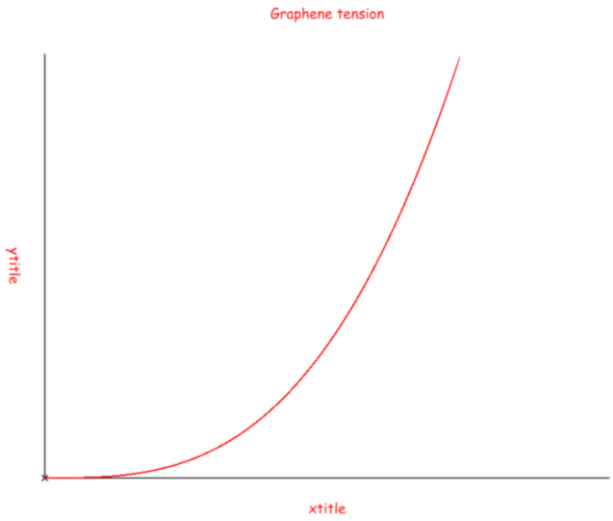
For each 2nd pixel on the x axis of the graph I calculate the y value using the equation given. If the value is out of the range given then the value is not plotted and a flag is given for above or below. Next time the graph plots, if the value is now in range the line is drawn from above the graph such that it appears that the point is not coming from nowhere.

To test this I am using the values above. If the graph appears to show properly according to my knowledge of graphs I shall give it a success as that is what others will judge it upon

Graph of X:



This is a success as it looks correct.



X^2 looks correct

X^3 looks correct

Therefore I can conclude that it plots correctly.

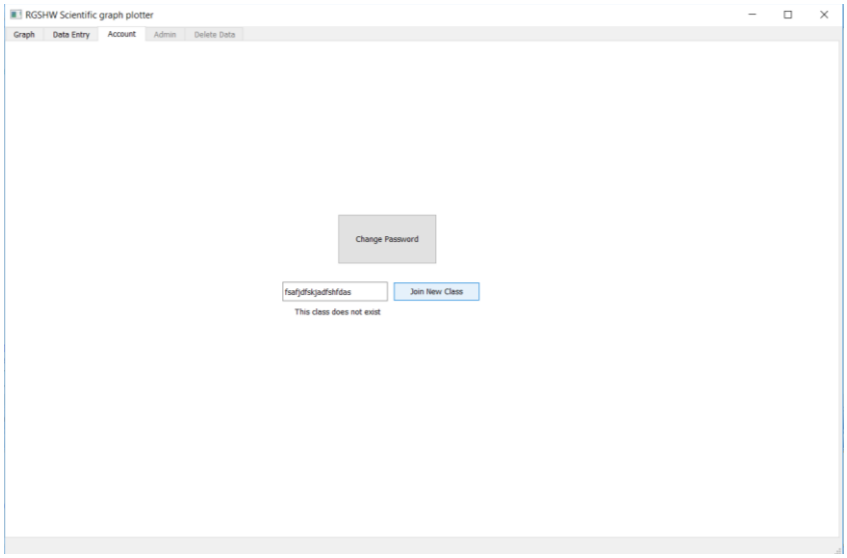
This can therefore be saved for use later in the program.

V0.13

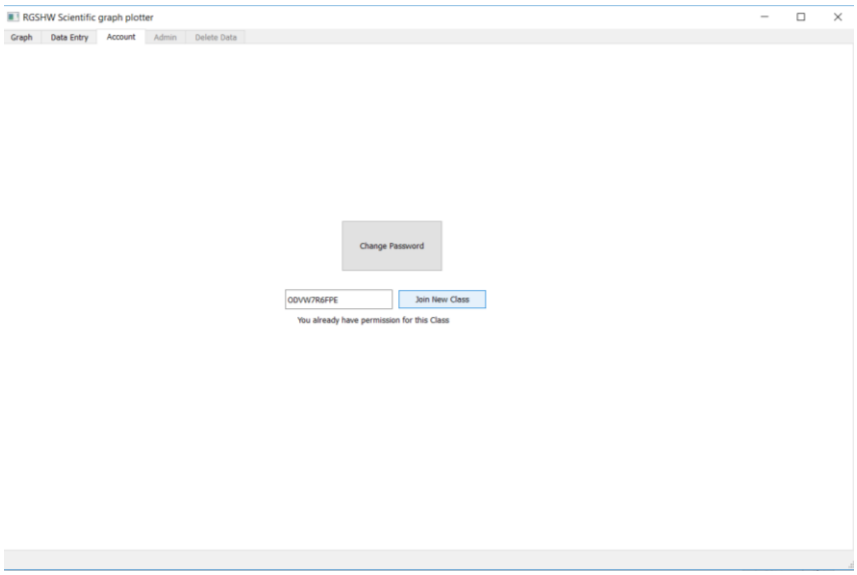
Aim: to allow students to join a class.

```
300     def joinclassmethod(self):
301             error = False
302             ClassCode = self.JoinClassEdit.text()
303             self.JoinClassStatus.setText("")
304             cur.execute("select ClassID from Classes where ClassCode=?;",(ClassCode,))
305             try:
306                 QueryResult = fetchallstripper(cur.fetchall()[0])
307             except:
308                 error = True
309                 self.JoinClassStatus.setText("This class does not exist")
310             if error == False:
311                 global currentuser
312                 cur.execute("select StudentID from Students where Username=?;",(currentuser,))
313                 userid=fetchallstripper(cur.fetchall()[0])
314                 try:
315                     cur.execute("select EntryID from StudentEntry where ClassID=? and studentID=?;",(QueryResult,userid,))
316                     existencechecker = fetchallstripper(cur.fetchall()[0])
317                     self.JoinClassStatus.setText("You already have permission for this Class")
318                 except:
319                     cur.execute("insert into StudentEntry(StudentID,ClassID)values(?,?);",(userid,QueryResult,))
320                     con.commit()
321                     self.JoinClassEdit.setText("")
322                     self.JoinClassStatus.setText("Enter the class code for the class you desire to join")
```

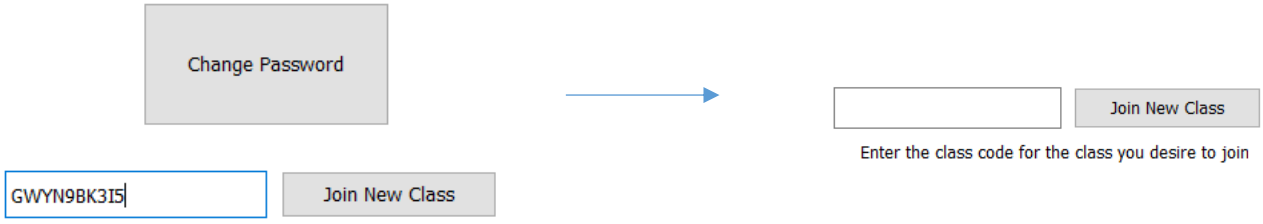
Accesses the class code that the user entered. If it does not exist an error is returned. If they already have permission for the graph an error is also shown. Otherwise the student is given a permission using their found StudentID.



Success – code is not used



Class with permission already existing – success



Success- the code was accepted.

Version 0.14

Aim : to generate statistics for later use.

```
def statisticsgeneration(GraphID):
    cur.execute("SELECT XVal FROM DataPoints WHERE GraphID =?;",(GraphID,))
    xlist=fetchallstripper(cur.fetchall())
    if len(xlist)==2:
        print("zero list")
        return "Error","Error"
    xlist = xlist.replace("[","")
    xlist = xlist.replace("]","")
    xlist=xlist.split(" ")#
    for i in range(len(xlist)):
        xlist[i]=float(xlist[i])
    cur.execute("SELECT YVal FROM DataPoints WHERE GraphID =?;",(GraphID,))
    ylist=fetchallstripper(cur.fetchall())
    ylist = ylist.replace("[","")
    ylist = ylist.replace("]","")
    ylist=ylist.split(" ")
    for i in range(len(ylist)):
        ylist[i]=float(ylist[i])
    sumx = 0
    sumxsquared=0
    if len(xlist)==0:
        return "Error","Error" #twice to ensure that list[0] finds "Error" to avoid crashes
    for each in xlist:
        sumx = sumx+each
        sumxsquared+= each**2
    sumy = 0
    for each in ylist:
        sumy = sumy+float(each)
    sumxy=0
    for i in range(len(xlist)):
        sumxy+= xlist[i]*ylist[i]
    n = len(xlist)
    minx = 10000000000000000000000
    maxx = -10000000000000000000000
    miny = 10000000000000000000000
    maxy = -10000000000000000000000
    points=[]
```

First off the function finds the xvalues for the current list. If xlist is 2 long then it is an empty list so therefore an error is returned. The xlist is then splitted into an actual list and each value is turned into a float. The same is done for the yvalues. The sum of x and sum of x squared are then found by adding the values throught the list. The sumy is then found and sumxy as well. The maximum and minimum values are then found for x and y (continued)

```
    for i in range(len(ylist)):
        newrow=[xlist[i],ylist[i]]
        points.append(newrow)
        if xlist[i]<minx:
            minx = xlist[i]
        if xlist[i] > maxx:
            maxx=xlist[i]
        if ylist[i] > maxy:
            maxy = ylist[i]
        if ylist[i]< miny:
            miny = ylist[i]
    return xlist,sumx,sumxsquared,sumy,sumxy,minx,maxx,miny,maxy,n,points ,ylist
```

These values are then returned from the function for later use.

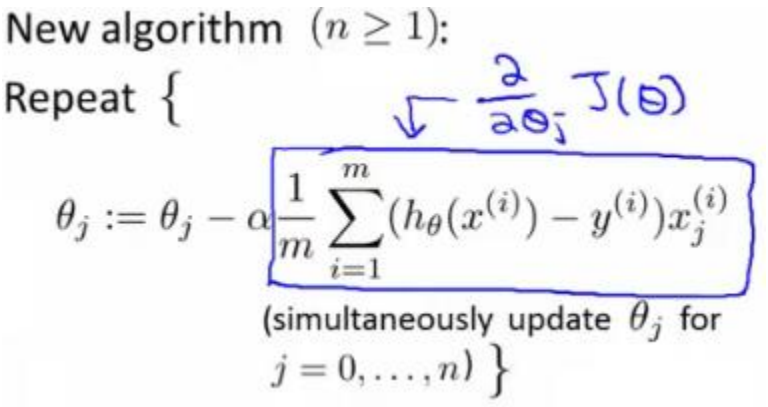
Version 0.15

Aim: to add a polynomial regression calculator.

```
99  def polynomialregression(xlist,ylist):
100     global currentrepresentation
101     alpha=0.1
102     theta = [0,0,0,0,0,0]
103     for i in range(10000):
104         thetatemp=[0,0,0,0,0,0]
105         total = 0
106         hypothesis = 0
107         for j in range(len(xlist)):
108             for k in range(len(theta)):
109                 hypothesis += ((xlist[j]**(5-j))*theta[k])
110             cost = (ylist[j]-hypothesis)
111             for l in range(len(theta)):
112                 thetatemp[l]=thetatemp[l]+(alpha*cost*(xlist[j]**(5-l)))
113         theta = thetatemp
114     print(theta)
115     currentrepresentation=theta
```

The polynomial regression function uses the loops to create sums of values. It then simultaneously updates the theta values to reflect new changes. 10,000 iterations are done to make it as accurate as possible. I am printing theta for testing purposes

It uses the following equation:



<u>Testing</u>

I will first test some hard coded values

```
533  polynomialregression([0.1,0.2,0.3,0.4],[0.1,0.2,0.3,0.4])
```

This should return a line similar to [0,0,0,0,1,0]

```
[0.0032, 0.0027, 0, 0.0012, 1.002, 0.014]
```

These values seem realistic, therefore it is successful for the hard coded values.

This will be tested with real graphs later when plotting is implemented.

Version 0.16

Aim: To add linear regression

```
42 def linearregression(sumx,sumy,sumxy,sumxsquared,n):
43     Sxy = sumxy - ((sumx*sumy)/n)
44     Sxx = sumxsquared - ((sumx**2)/n)
45     b = Sxy / Sxx
46     a = (sumy/n)-(b*(sumx/n))
47     global currentrepresentation
48     currentrepresentation = [0,0,0,0,b,a]
```

This uses the statistical values to calculate the next values, which are then used to calculate gradient and intercept.

I will now integrate this into the program to plot.

```
294         def plot(self):
295             global plotsyle
296             global currentrepresentation
297             currentrepresentation =[0,0,0,0,0,0]
298             if self.currentgraph!=-1:
299                 statistics =statisticsgeneration(self.currentgraph)
300                 if statistics[0]!="Error":
301                     xlist=statistics[0]
302                     sumx=statistics[1]
303                     sumxsquared=statistics[2]
304                     sumy=statistics[3]
305                     sumxy=statistics[4]
306                     minx=statistics[5]
307                     maxx=statistics[6]
308                     miny=statistics[7]
309                     maxy=statistics[8]
310                     n=statistics[9]
311                     points=statistics[10]
312                     ylist=statistics[11]
313                     self.MeanxLbl.setText(str(sumx/n))
314                     self.MeanyLbl.setText(str(sumy/n))
315                     self.nofpointsLbl.setText(str(n))
316                     cur.execute("select Title,XTitle,YTitle from Graphs where GraphID=?;",(self.currentgraph))
317                     titles = cur.fetchall()
318                     titles=str(titles[0])
319                     titles=titles.replace("'","")
320                     titles=titles.replace("(","")
321                     titles=titles.replace(")","")
322                     titles=titles.split(",")
323                     title =  titles[0]
324                     title = "'"+title+"'"
325                     xtitle= titles[1]
326                     xtitle = "'" +xtitle+"'"
327                     ytitle=titles[2]
328                     ytitle = "'" +ytitle+"'"
```

Calculates the statistical values and create variables for each of the values. I then set the values for the display for mean and number of points.

Then finds the titles.

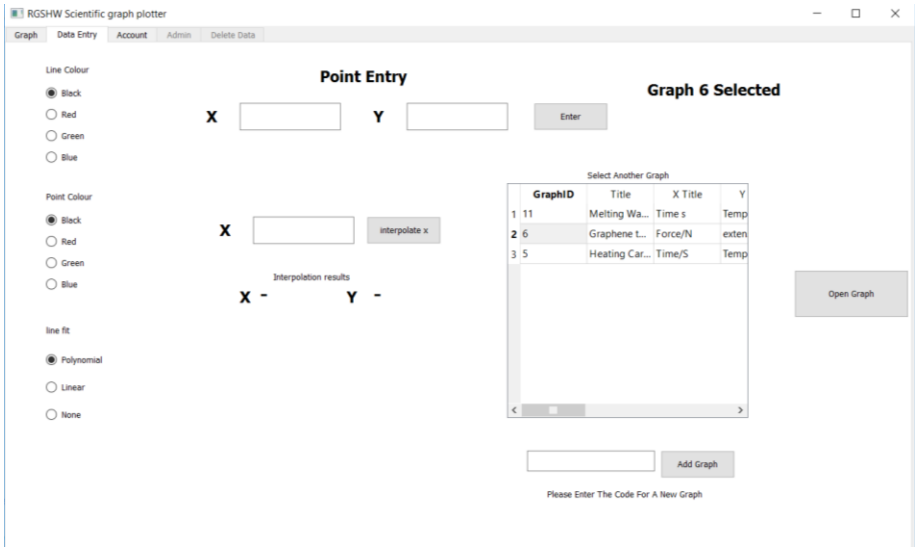
```
if plotstyle == "Linear":
    linearregression(sumx,sumy,sumxy,sumxsquared,n)
if plotstyle == "Polynomial":
    polynomialregression(xlist,ylist)
```

Sets the current representation using the functions.

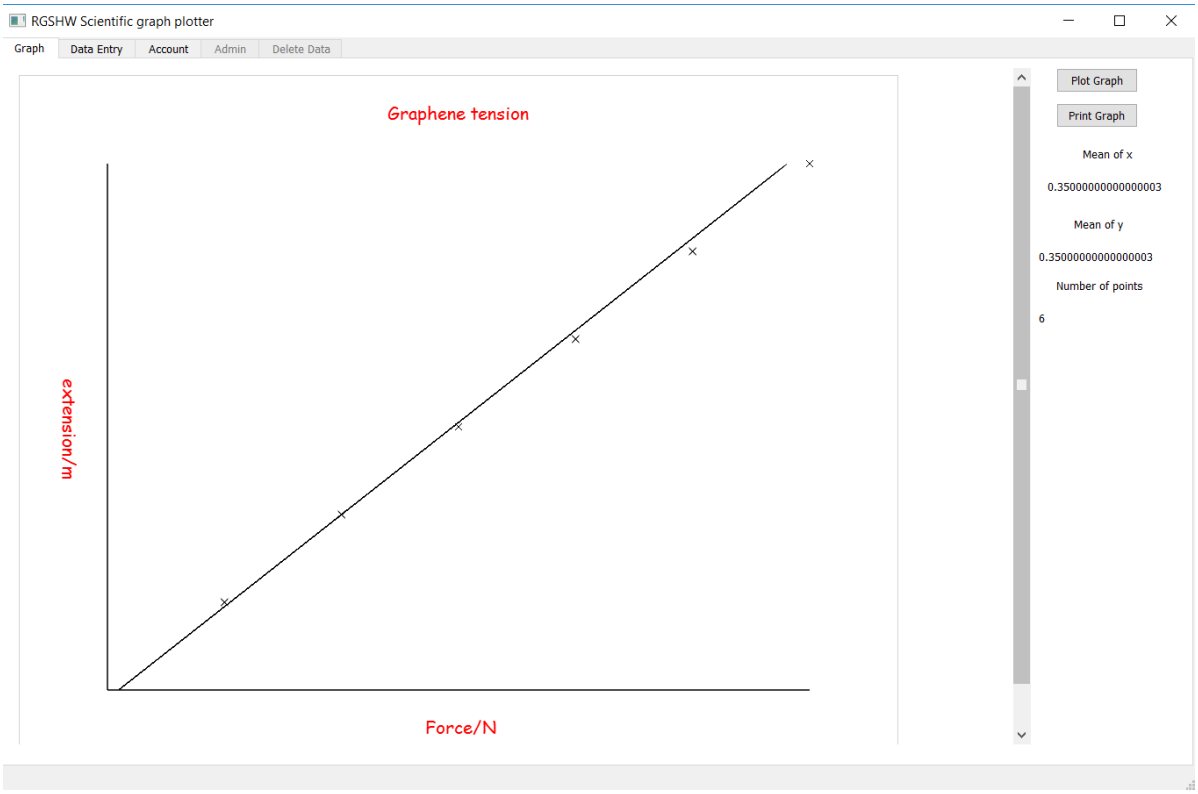
```
                file = open('javascript3.txt', 'r')
                htmltext=file.read()
                global colours
                #0 - line 1 - point
                print(plotstyle)
                values=[maxx,minx,maxy,miny,points,currentrepresentation,title,xtitle,ytitle,colours[0],colours[1]]
                replacelist=["#maxx#","#minx#","#maxy#","#miny#","#points#","#equation#","#title#","#xtitle#","#ytitle#","#linecol#","#pointcol#"]
                for i in range(len(replacelist)):
                    htmltext = htmltext.replace(replacelist[i],str(values[i]))
                self.GraphView.setHtml(htmltext)
            else:
                self.GraphView.setHtml("")
```

Reads the javascript file and sets the html to it. The javascript is then edited and the values required are edited. This is then displayed to the user.

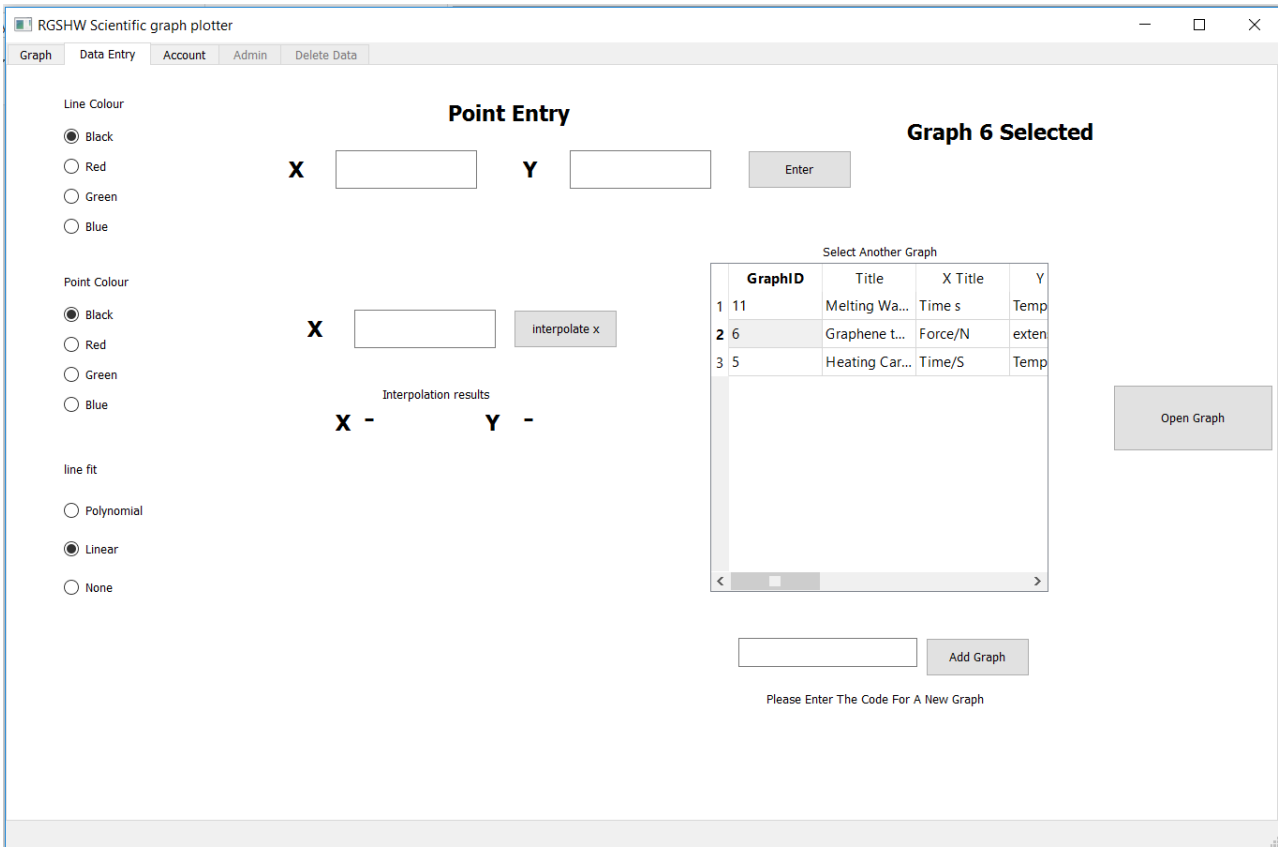
Testing



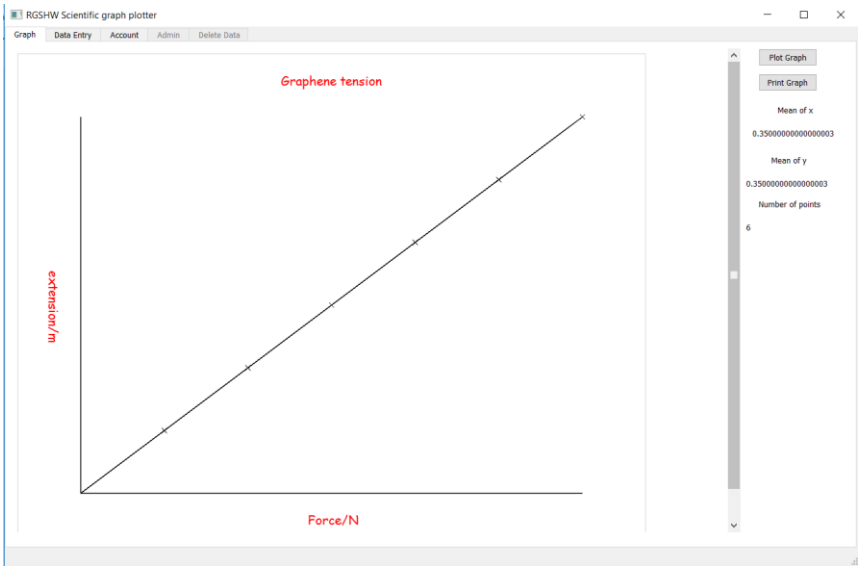
Selected choices for first test



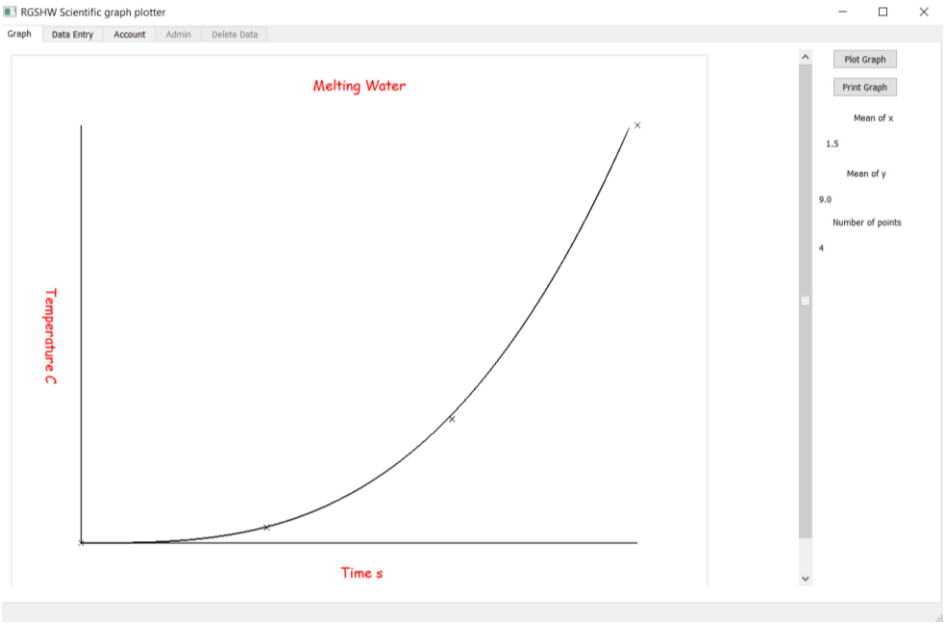
All of the calculated values are correct and the line of best fit fits correctly with sensible axes.



The same graph is tested with linear results



This version is more precise as it doesn't use gradient descent optimisation which uses actual calculations.



Polynomial works as expected.

Linear works as expected.

V 0.17

Aim: interpolate values

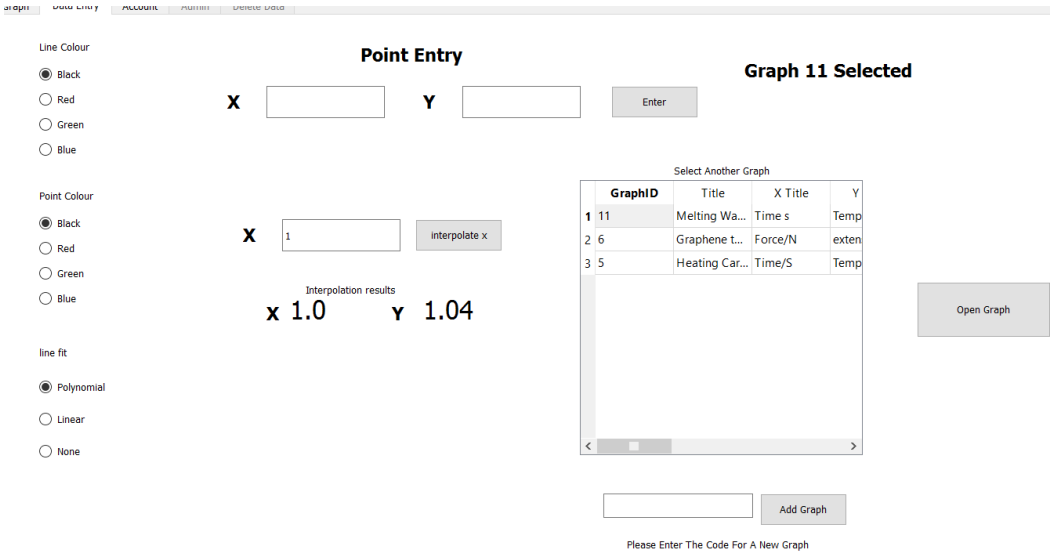
```
459         def interpolate(self):
460             xval = self.XEdit.text()
461             yval = 0
462             try:
463                 xval = float(xval)
464                 for i in range(len(currentrepresentation)):
465                     yval = yval + currentrepresentation[i]*(xval**(5-i))
466                 self.Xinterp.setText(xval)
467                 self.Yinterp.setText(yval)
468                 self.InterpError.setText("")
469             except:
470                 self.InterpError.setText("This value is not a float")
```

The function gets the x value entered. It then calculates the y values and display the values. If there is an error the value must not be a float so the error is displayed

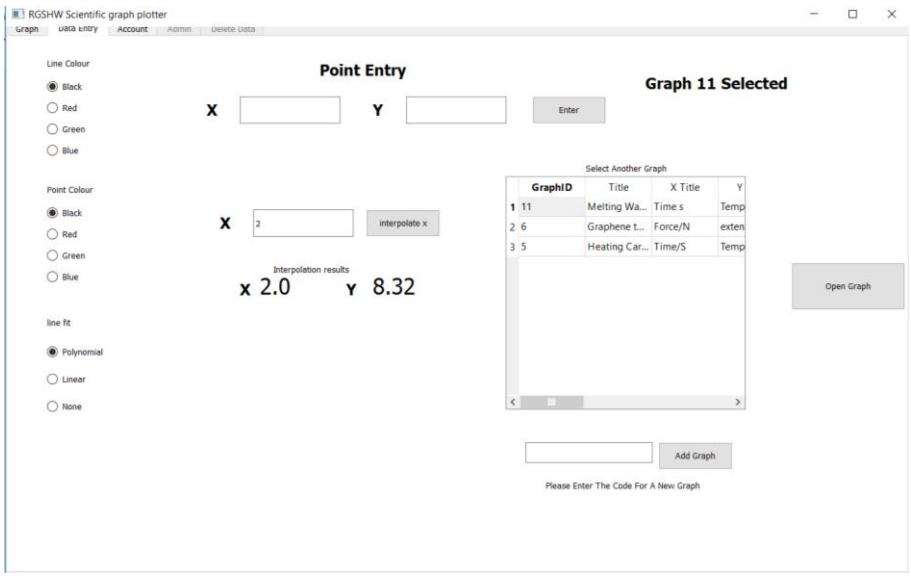
Initially there was an error every time for valid answers. This was because I didn't add

Global CurrentRepresentation

At the top.



This is as expected



This is as expected

Version 0.18

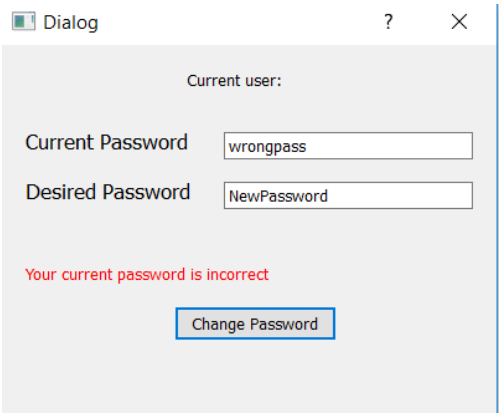
Aim: to set up password changes.

```
536    class ClassCreateClass(QtGui.QDialog,classcreate):
537        def __init__(self,parent=None):
538            QtGui.QDialog.__init__(self,parent)
539            self.setupUi(self)
540            self.Change.clicked.connect(self.passwordchange)
541        def passwordchange(self):
542            global currentuser
543            trialpass = self.Current.text()
544            desiredpass = self.Desired.text()
545            cur.execute("SELECT Password FROM Students WHERE Username = ?;",(currentuser,))
546            dbpass = fetchallstripper(cur.fetchall()[0])
547            trialpass=trialpass.encode('utf-8')
548            trialpass = hashlib.sha512(trialpass).hexdigest()
549            if trialpass == dbpass:
550                if len(desiredpass)> 20 or len(desiredpass) < 7:
551                    self.errorlabel.setText("This password isn't between 7 and 20 characters long")
552                else:
553                    desiredpass=desiredpass.encode('utf-8')
554                    desiredpass = hashlib.sha512(desiredpass).hexdigest()
555                    cur.execute("Update Students Set Password =? where username = ?",(desiredpass,currentuser,))
556            else:
557                self.errorlabel.setText("Your current password is incorrect")
558
```

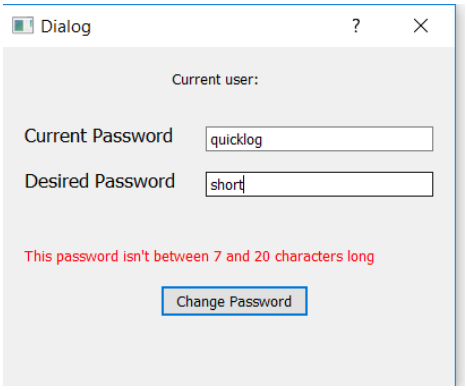
Sets up the password change UI

Then validates the passwords against eachother. If they match the new password is validated and then entered into the database.

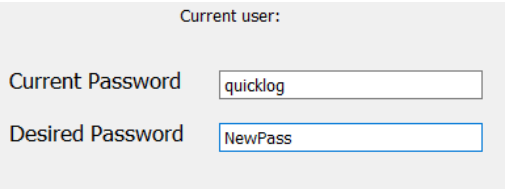
Testing



Success – entered password is wrong



Success



The value was then changed in the database. Success.

Editions

Capitalising constants

In order to match industry standards and improve readability for maintenance I have decided to capitalise constants.

PI –

```
PI = Math.PI
ctx.rotate( PI / 2 );
ctx.fillText("ytitle", 400, -50); // change ytitle
ctx.rotate(-PI / 2 );
```

Pi is used for rotations in my javascript. I have capitalised the name and then used it later in the javascript. This makes it much easier for readers to understand that Math.PI refers to the constant and not another function.

CANVWIDTH and CANVHEIGHT-

```
CANVWIDTH=800;
CANVHEIGHT=600;
```

CANVWIDTH and CANVHEIGHT are constants added to denote the canvas width and height. These are used later so it makes it much easier to adjust sizes when changes are required. The capitalisation is also used.

```
function pointtographvalue(xvalue,yvalue){
  xpos = 100+((xvalue-minx)/rangex)*CANVWIDTH;
  ypos = 100+((maxy-yvalue)/rangey)*CANVHEIGHT;
  return [xpos,ypos];
}
```

An example of CANVWIDTH and CANVHEIGHT being used in calculation to work out where the point needs to be plotted.

Polynomial regression constants-

```
def polynomialregression(xlist,ylist):
    global currentrepresentation
    ITERATIONS = 10000;
    ALPHA=0.1
    theta = [0,0,0,0,0,0]
    for i in range(ITERATIONS):
        thetatemp=[0,0,0,0,0,0]
        total = 0
        hypothesis = 0
        for j in range(len(xlist)):
            for k in range(len(theta)):
                hypothesis += ((xlist[j]**(5-j))*theta[k])
            cost = (ylist[j]-hypothesis)
            for l in range(len(theta)):
                thetatemp[l]=thetatemp[l]+(ALPHA*cost*(xlist[j]**(5-l)))
        theta = thetatemp
    print(theta)
    currentrepresentation=theta
```

ITERATIONS are the number of iterations of gradient descent required. More requires more time than fewer, however produces better results.

ALPHA is the learning rate, the speed at which the values of the “thetas” change. Lower produces better results, whereas higher means results are found more quickly.

Adding these as constants results in easier maintenance of the polynomial regression function.

Hungarian notation

Below I show some examples of my use of Hungarian notation.

```
100 def polynomialregression(xlist,ylist):
101     global currentrepresentation
102     ITERATIONS = 10000;
103     ALPHA=0.1
104     theta = [0,0,0,0,0,0]
105     for i in range(ITERATIONS):
106         thetatemp=[0,0,0,0,0,0]
107         fhypothesis = 0
108         for j in range(len(xlist)):
109             for k in range(len(theta)):
110                 fhypothesis += ((xlist[j]**(5-j))*theta[k])
111             fcost = (ylist[j]-fhypothesis)
112             for l in range(len(theta)):
113                 thetatemp[l]=thetatemp[l]+(ALPHA*fcost*(xlist[j]**(5-l)))
114         theta = thetatemp
115     print(theta)
116     currentrepresentation=theta
```

Here I have used the prefix of “f” for hypothesis and cost to show that they are floats. This uses standard Hungarian notation, improving readability.

```
fsumx = 0
fsumxsquared=0
if len(xlist)==0:
    return "Error","Error" #twice to ensure that list[0] finds "Error" to avoid crashes
for each in xlist:
    fsumx = fsumx+each
    fsumxsquared+= each**2
fsumy = 0
for each in ylist:
    fsumy = fsumy+float(each)
fsumxy=0
for i in range(len(xlist)):
    fsumxy+= xlist[i]*ylist[i]
n = len(xlist)
```

Here I used f again to denote floats. I decided not to replace “n” with “in” as this could confuse the mathematical use of this variable.

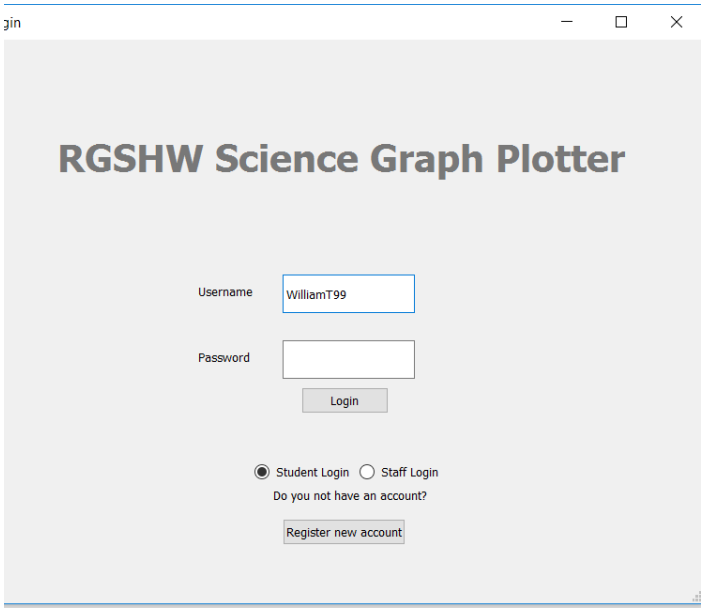
```
def loginclick(self,curr):
    strusername=self.Username.text()
    strpassword=self.Password.text()
    if permissions == 'Students':
        cur.execute("SELECT Password FROM Students WHERE Username = ?;",(strusername,))
    else:
        cur.execute("SELECT Password FROM Teachers WHERE Username = ?;",(strusername,))
    try:
        strtemporarypass = cur.fetchone()[0].replace("'","").replace(",","").replace("(","").replace(")","")
        strpassword=strpassword.encode('utf-8')
        strpassword = hashlib.sha512(strpassword).hexdigest()
        if strtemporarypass == strpassword:
            global currentuser
            currentuser=strusername
```

Here I use the str prefix to denote the strings used in logging in. This shows that I haven’t used another datatype to store passwords for people maintaining the code.

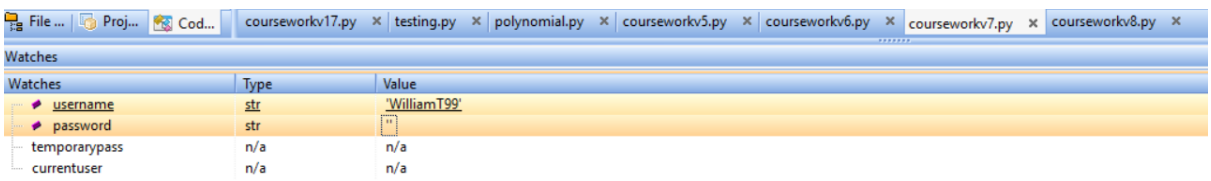
Variable watch and breakpoints.

I previously used variable watch and break points, however I would like to show its further use in this section.

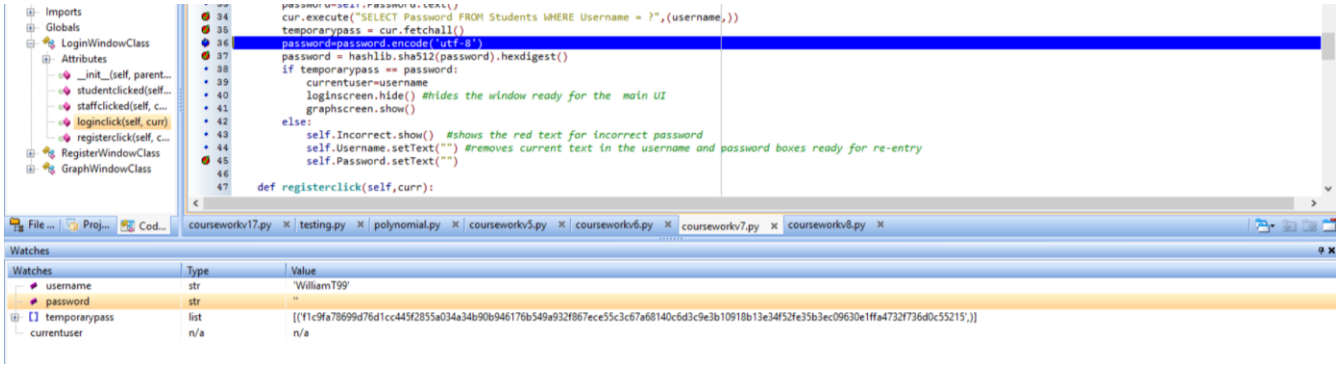
Here is a copy of the previous example:



The username is entered and login is pressed



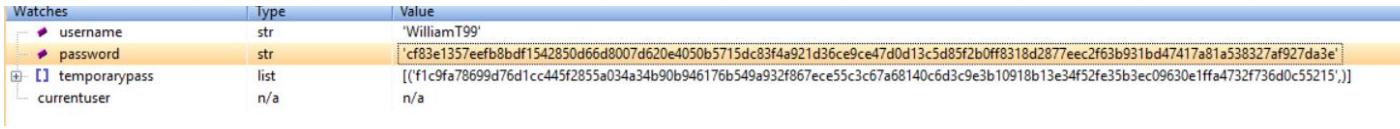
At the first breakpoint on line 34 the values are correct as entered



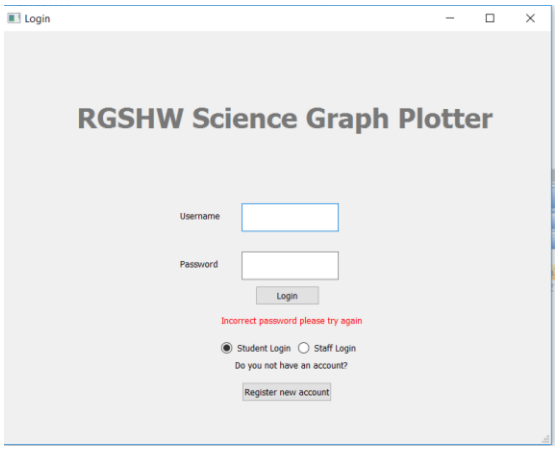
At line 36 the value for temporary pass is

[('f1c9fa78699d76d1cc445f2855a034a34b90b946176b549a932f867ece55c3c67a68140c6d3c9e3b10918b13e34f52fe35b3ec09630e1ffa4732f736d0c55215',)]

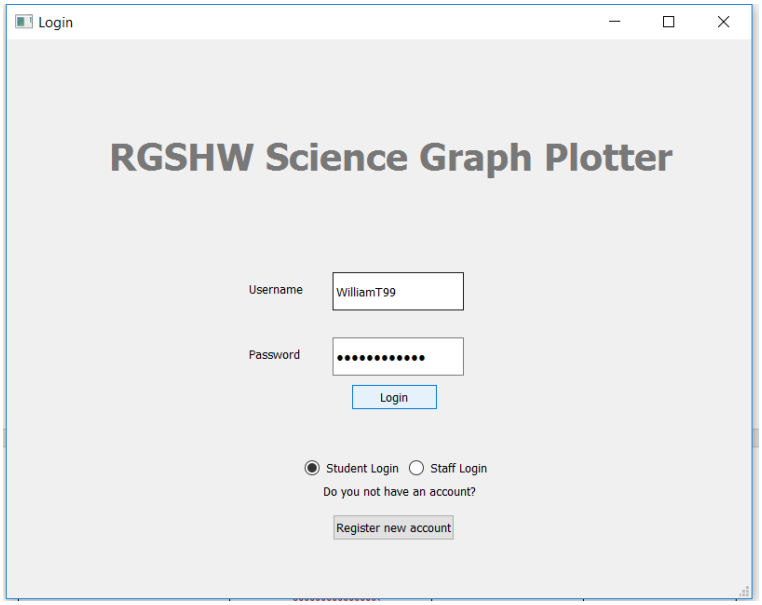
This is as expected as this is how pyscripter formats its variables.



The password doesn't match the temporary password and should be rejected.

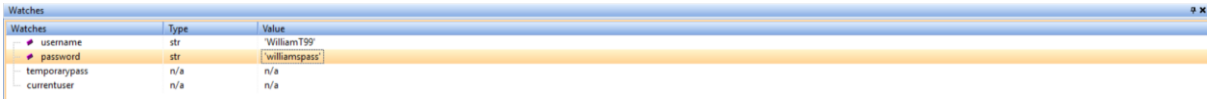


This is as expected as the variable should be rejected.

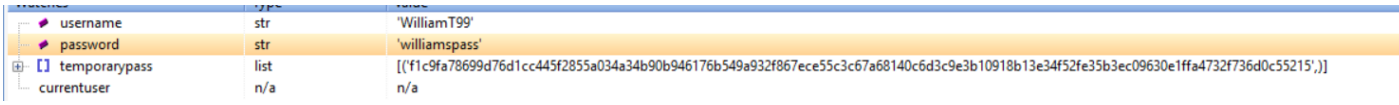


Now the correct value is entered “williamspass” for “WilliamT99”

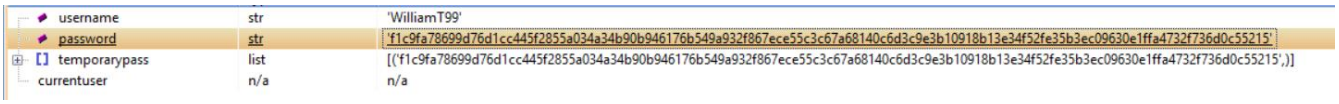
This should result in the data being accepted



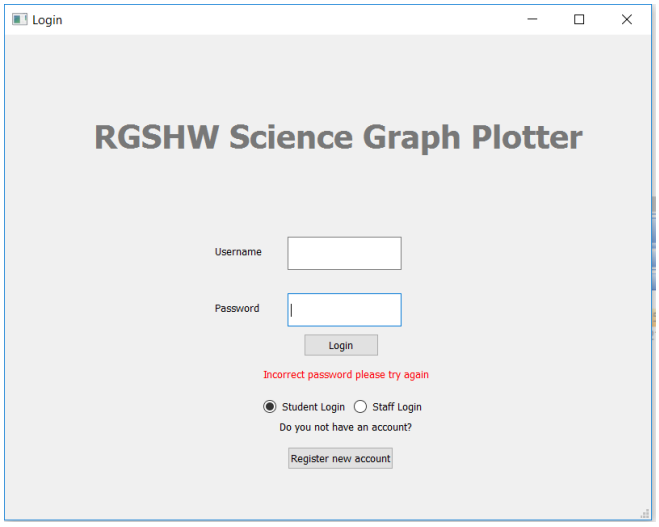
The correct values are in the variable watch after login is clicked.



The correct value for temporary pass is found



The password is then hashed to the right password.



The incorrect message is returned as the two passwords do not match. This is possibly because the formatting is incorrect for the comparison, and also the list is checked instead of its first object.

As you can see the variable watch in combination with the break points was very useful.

I would also like to check my code create function. Without breakpoints it would not be possible to check if the function recurs without prints. This makes it much easier to see exactly what the function is doing.

```
120 def codecreate():
121         newcode = ""
122         for i in range(10):
123             newcode = newcode + str(random.choice(string.ascii_uppercase + string.digits))
124         newcode = str(newcode)
125         cur.execute("select count(GraphCode) from Graphs where GraphCode=?;",(newcode,))
126         tempfetchgraph = cur.fetchall()[0]
127         cur.execute("select count(ClassCode) from Classes where ClassCode=?;",(newcode,))
128         tempfetchclass = cur.fetchall()[0]
129         if str(tempfetchgraph).strip() == "(0,)" and str(tempfetchclass).strip() == "(0,)":
130             return newcode
131         else:
132             codecreate()
```

Here we can see that breakpoints have been added on line 121, 123, 127, 129, and 132. This is to test to see if the function actually recurs.

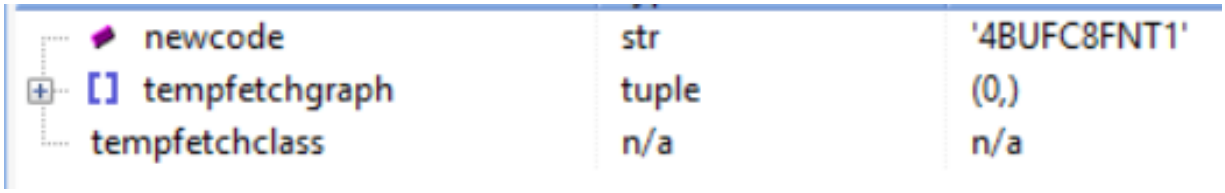
newcode	str	''
tempfetchgraph	n/a	n/a
tempfetchclass	n/a	n/a

Initially the code starts as “” as expected.

Watches	Type	Value
newcode	str	'4'
tempfetchgraph	n/a	n/a
tempfetchclass	n/a	n/a

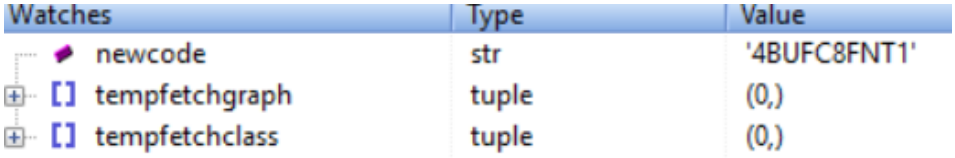
Then a character is added as expected.

```
120 def codecreate():
121         newcode = ""
122         for i in range(10):
123             newcode = newcode + str(random.choice(string.ascii_uppercase + string.digits))
124         newcode = str(newcode)
125         cur.execute("select count(GraphCode) from Graphs where GraphCode=?;",(newcode,))
126         tempfetchgraph = cur.fetchall()[0]
127         cur.execute("select count(ClassCode) from Classes where ClassCode=?;",(newcode,))
128         tempfetchclass = cur.fetchall()[0]
129         if str(tempfetchgraph).strip() == "(0,)" and str(tempfetchclass).strip() == "(0,)":
130             return newcode
131         else:
132             codecreate()
```



This continues until 10 characters are added as expected.

The tempfetchgraph is also “(0,)” which is as expected as there are no graphs with this code



The tempfetchclass also returns this value as there are no other classes with this code.

The function then ends meaning that it works as expected!

Comments

I have added comments to each function to add readability to the code, which will help with maintenance in the future.

```
try: # if theres an error the try will fail
    import numpy.polynomial.polynomial as poly
    import numpy.linalg as linalg
    from PyQt4 import QtCore, QtGui, uic
    import numpy as np
    import sys
    import hashlib
    import datetime
    import sqlite3 as lite
    import numbers
    import string
    import random
    #import libraries
    error=False
    con = lite.connect('GraphPlotterDB.db')
    cur = con.cursor()
    #connects to database
    cur.execute("select * from Students")
    datatest=cur.fetchall()
    if not datatest:
        print("CRITICAL ERROR : Database MISSING")
        error = True
        #checks if the database contains data
    loginclass = uic.loadUiType("loginwindow.ui")[0]
    registerclass = uic.loadUiType("registerwindow.ui")[0]     #loads the uis ready for use in classes
    graphclass=uic.loadUiType("graphwindow.ui")[0]
    graphcreate=uic.loadUiType("newgraphwindow.ui")[0]
    classcreate=uic.loadUiType("ClassCreate.ui")[0]
    passwordclass = uic.loadUiType("Passwordwindow.ui")[0] #loads UIs for use in classes
except:
    print("CRITICAL ERROR : FILES MISSING") # if files are missing an error is printed
    error = True
userid=""
currentuser = ""
permissions = "Students"
colours = ["#000000","#000000"]
plotstyle="Polynomial"
currentrepresentation=[0,0,0,0,0,0] # [X^5,X^4,X^3,X^2,X,Constant]
#sets up global variables
def fetchallstripper(oldtuple):
```

```
    newstring = str(oldtuple)
    newstring = newstring.replace("(","")
    newstring = newstring.replace(")","")
    newstring = newstring.replace(",","")
    return newstring
#formats fetchall tuples into the format of strings required for comparison
def linearregression(sumx,sumy,sumxy,sumxsquared,n):
    Sxy = sumxy - ((sumx*sumy)/n)
    Sxx = sumxsquared - ((sumx**2)/n)
    b = Sxy / Sxx
    a = (sumy/n)-(b*(sumx/n))
    global currentrepresentation
    currentrepresentation = [0,0,0,0,b,a]
    #calculates the gradient and intercept using the mathematical formulae.
def statisticsgeneration(GraphID):
    cur.execute("SELECT XVal FROM DataPoints WHERE GraphID =?;",(GraphID,))
    xlist=fetchallstripper(cur.fetchall())
    #creates a list of all of the x values
    if len(xlist)==2:
        print("zero list")
        return "Error","Error"
        #if the string returned only has 2 values then there must be an error. Returned in list form so i can keep datatype consistent as list
    xlist = xlist.replace("[","")
    xlist = xlist.replace("]","")
    xlist=xlist.split(" ")
    #finishes creating list of x values
    for i in range(len(xlist)):
        xlist[i]=float(xlist[i])
        #turns each x value in the list into a float for statistical analysis
    cur.execute("SELECT YVal FROM DataPoints WHERE GraphID =?;",(GraphID,))
    ylist=fetchallstripper(cur.fetchall())
    ylist = ylist.replace("[","")
    ylist = ylist.replace("]","")
    ylist=ylist.split(" ")
    #creates a list of y values from the SQL query.
    for i in range(len(ylist)):
        ylist[i]=float(ylist[i])
        #turns each of the y values into floats of each value
    fsumx = 0
    fsumxsquared=0 # initialises variables
```

```
    if len(xlist)==0:
        return "Error","Error" #twice to ensure that list[0] finds "Error" to avoid crashes
    for each in xlist:
        fsumx = fsumx+each
        fsumxsquared+= each**2 #calculates sum of x and sum of x squared by adding each value for the sum for each value.
    fsumy = 0
    for each in ylist:
        fsumy = fsumy+float(each) # calculates sumy in the same way
    fsumxy=0
    for i in range(len(xlist)):
        fsumxy+= xlist[i]*ylist[i] #calculates the sum of x multiplied by y
    n = len(xlist) # finds the length of the list, which becomes the statistical value n- the number of points
    minx = 100000000000000000000000
    maxx = -100000000000000000000000
    miny = 100000000000000000000000
    maxy = -100000000000000000000000
    points=[]
    for i in range(len(ylist)):
        newrow=[xlist[i],ylist[i]]
        points.append(newrow)
        if xlist[i]<minx:
            minx = xlist[i]
        if xlist[i] > maxx:
            maxx=xlist[i]
        if ylist[i] > maxy:
            maxy = ylist[i]
        if ylist[i]< miny:
            miny = ylist[i]
        # for each of the points, check if they would be a new value of min or max for x and y
    return xlist,fsumx,fsumxsquared,fsumy,fsumxy,minx,maxx,miny,maxy,n,points ,ylist

def polynomialregression(xlist,ylist):
    global currentrepresentation
    ITERATIONS = 10000; #initialises number of iterations and learning rate
    ALPHA=0.1
    theta = [0,0,0,0,0,0] # sets the representation to nothing for analysis.
    for i in range(ITERATIONS):
        thetatemp=[0,0,0,0,0,0] # creates a new temporary value for calculation
        fhypothesis = 0
        for j in range(len(xlist)):
            for k in range(len(theta)):
                fhypothesis += ((xlist[j]**(5-j))*theta[k]) # calculates the hypothesis based upon the values of x and theta
            fcost = (ylist[j]-fhypothesis) # calculates the distance between y and the hypothesis
            for l in range(len(theta)):
                thetatemp[l]=thetatemp[l]+(ALPHA*fcost*(xlist[j]**(5-l))) # uses this cost and the learning rate to update each thetatemp
        theta = thetatemp # replaces theta with theta temp
    print(theta)
    currentrepresentation=theta# finally changes the representation used in the graph to the value required



def codecreate():
        newcode = ""
        for i in range(10):
            newcode = newcode + str(random.choice(string.ascii_uppercase + string.digits)) # adds a number or letter to the end of the code 10 times
        newcode = str(newcode)
        cur.execute("select count(GraphCode) from Graphs where GraphCode=?;",(newcode,)) #finds graphs with this code
        tempfetchgraph = cur.fetchall()[0]
        cur.execute("select count(ClassCode) from Classes where ClassCode=?;",(newcode,)) # finds classes with this code
        tempfetchclass = cur.fetchall()[0]
        if str(tempfetchgraph).strip() == "(0,)" and str(tempfetchclass).strip() == "(0,)": # if both have no results, then the code must be unique
            return newcode
        else:
            codecreate() # calls the function again if the code is not unique

if error != True:
    class LoginWindowClass(QtGui.QMainWindow,loginclass):
        user="Students"
        def __init__(self,parent=None):
            QtGui.QMainWindow.__init__(self,parent) #initiates the login window when the program starts
            self.setupUi(self)
            self.StudentOption.setChecked(True)
            self.Incorrect.hide()  # hides the red text that appears when a password is entered incorrectly
            self.LoginButton.clicked.connect(self.loginclick) # initiates loginclick method when the login button is clicked
            self.NewAccountButton.clicked.connect(self.registerclick) # initiates registerclick method when register button is pressed
            self.StaffOption.clicked.connect(self.staffclicked) #changes the options when the radio button is pressed.
            self.StudentOption.clicked.connect(self.studentclicked)
        def studentclicked(self,curr):
            global permissions
            permissions = "Students"
```

```
        if str(tempfetchgraph).strip() == "(0,)" and str(tempfetchclass).strip() == "(0,)": # if both have no results, then the code must be unique
            return newcode
        else:
            codecreate() # calls the function again if the code is not unique

if error != True:
    class LoginWindowClass(QtGui.QMainWindow,loginclass):
        user="Students"
        def __init__(self,parent=None):
            QtGui.QMainWindow.__init__(self,parent) #initiates the login window when the program starts
            self.setupUi(self)
            self.StudentOption.setChecked(True)
            self.Incorrect.hide()  # hides the red text that appears when a password is entered incorrectly
            self.LoginButton.clicked.connect(self.loginclick) # initiates loginclick method when the login button is clicked
            self.NewAccountButton.clicked.connect(self.registerclick) # initiates registerclick method when register button is pressed
            self.StaffOption.clicked.connect(self.staffclicked) #changes the options when the radio button is pressed.
            self.StudentOption.clicked.connect(self.studentclicked)
        def studentclicked(self,curr):
            global permissions
            permissions = "Students"
        def staffclicked(self,curr): # changes permissions based upon radio button
            global permissions
            permissions = "Teacher"


        def loginclick(self,curr):
            strusername=self.Username.text()
            strpassword=self.Password.text() # recieves username and password from UI
            if permissions == 'Students':
                cur.execute("SELECT Password FROM Students WHERE Username = ?;",(strusername,))
            else:                                                                    # finds passwords for the current user
                cur.execute("SELECT Password FROM Teachers WHERE Username = ?;",(strusername,))
            try:
                strtemporarypass = cur.fetchone()[0].replace("'","").replace(",","").replace("(","").replace(")","")
                strpassword=strpassword.encode('utf-8')
                strpassword = hashlib.sha512(strpassword).hexdigest() #hashes entered password
                if strtemporarypass == strpassword:
                    global currentuser
                    currentuser=strusername # if the password is correct the current user is updated
                    global userid
```

ter - C:\Users\taylw\Portable Python 3.2.5.1\Coursework\courseworkv17.py

it Search View Project Run Tools Help

```
                    cur.execute("select StudentID from Students where Username=?;",(currentuser,))
                    userid=fetchallstripper(cur.fetchall()[0]) # finds the userID for the person logging in
                    loginscreen.hide() #hides the window ready for the  main UI
                    graphscreen.show()
                    if permissions =='Students':
                        graphscreen.changetableview(("select Graphs.* from Graphs INNER JOIN StudentPermissions ON Graphs.GraphID = StudentPermissions.GraphID where StudentPermissions.StudentID = ? order
                        #replace function used as impossible to use as function without and the userid is a result from a query which is automatically set and not user editable. no Chance of SQL infectic
                        graphscreen.Graph.setTabEnabled(3,False)
                        graphscreen.Graph.setTabEnabled(4,False)
                        graphscreen.NewGraph.hide() # hides features not avaialable to students
                        graphscreen.DeleteGraph.hide()
                    else:
                        graphscreen.changetableview("select * from Graphs order by GraphID Desc")
                        graphscreen.Graph.setTabEnabled(3,True)
                        graphscreen.Graph.setTabEnabled(4,True)
                        graphscreen.NewGraph.show() # shows features for staff
                        graphscreen.DeleteGraph.show()
                else:
                    self.Incorrect.show()  #shows the red text for incorrect password
                    self.Username.setText("") #removes current text in the username and password boxes ready for re-entry
                    self.Password.setText("")
            except:
                self.Incorrect.show()
                self.Username.setText("")
                self.Password.setText("")

        def registerclick(self,curr):
            loginscreen.hide() # when register is clicked the login is closed and register is hidden
            registerscreen.show()

    class RegisterWindowClass(QtGui.QMainWindow,registerclass):
        def __init__(self,parent=None):
            QtGui.QMainWindow.__init__(self,parent)
            self.setupUi(self)
            self.ReturnButton.clicked.connect(self.returnclick)
            self.SubmitButton.clicked.connect(self.submitclick) #sets up attributes and methods

        def returnclick(self,curr):
            self.UserInput.setText("")
            self.PasswordInput.setText("")
```

186 and 187 continued

```
= ? order by GraphID Desc;").replace("?",userid))
 injection
```

```
            self.ForenameInput.setText("")
            self.SurnameInput.setText("")
            self.UserDOB.setDate(QtCore.QDate(2000,1,1))
            registerscreen.hide()
            loginscreen.show() # if return is pressed values are reset

        def submitclick(self,curr):
            registeruser= self.UserInput.text()
            registerpass= self.PasswordInput.text()
            registerforename=self.ForenameInput.text()
            registersurname=self.SurnameInput.text()
            registerdate = str(self.UserDOB.date())
            registerdate=registerdate.replace('PyQt4.QtCore.QDate(','')
            registerdate=registerdate.replace(')','')
            registerdate = registerdate.split(',')
            self.ErrorLabelRegister.setText('') # sets up values entered
            newtext=('')
            cur.execute("select * from Students where username = ?;",(registeruser,))
            data = cur.fetchall()
            print(data)
            if registeruser.isalnum() == False:
                newtext="This username isn't alphanumeric" # checks if username is alphanumeric
            elif len(registeruser)> 20 or len(registeruser) < 7:
                newtext = "This username isn't between 7 and 20 characters long" # checks username length
            elif data != []:
                newtext = 'This username is already in use' #checks for username existence
            elif len(registerpass)> 20 or len(registerpass) < 7: #checks password length
                newtext = "This password isn't between 7 and 20 characters long"
            elif registerforename.isalpha()== False:#checks that forename is alphabetical
                newtext = "This forename is not alphabetical"
            elif len(registerforename)>15 or len(registerforename)<2: #checks that forename is of acceptable length
                newtext = "This forename is not between 2 and 15 characters long"
            elif registersurname.isalpha()== False: #checks that surname is alphabetical
                newtext = "This surname is not alphabetical"
            elif len(registersurname)>15 or len(registersurname)<2:
                newtext = "This surname is not between 2 and 15 characters long" #checks that the surname is the correct length
            elif int(registerdate[0])< (datetime.datetime.now().year -19) or int(registerdate[0])> (datetime.datetime.now().year -11): # checks that the person is born in an acceptable data.
                newtext = "This date is either too early or too late"
            self.ErrorLabelRegister.setText(newtext)
            if newtext == '':
                registerpass=registerpass.encode('utf-8')
                registerpass = hashlib.sha512(registerpass).hexdigest()
                registerdate = registerdate[0] + '-'+registerdate[1].strip() + '-' + registerdate[2].strip()
                cur.execute("INSERT INTO Students(Username,Password,Forename,Surname,DOB) Values (?,?,?,?,?);",(registeruser,registerpass,registerforename,registersurname,registerdate))
                con.commit() #adds correct users to database
                loginscreen.show()
                registerscreen.hide()
                self.UserInput.setText("")
                self.PasswordInput.setText("")
                self.ForenameInput.setText("")
                self.SurnameInput.setText("")
                self.UserDOB.setDate(QtCore.QDate(2000,1,1)) # resets text
            else:
                self.ErrorLabelRegister.show()


    class GraphWindowClass(QtGui.QMainWindow,graphclass):
        def __init__(self,parent=None):
            self.currentgraph=-1
            self.row = -1 #sets row and graph to none set
            global currentuser
            global userid
            QtGui.QMainWindow.__init__(self,parent)
            self.setupUi(self)
            self.GraphSelectError.hide()
            self.model = QtGui.QStandardItemModel(self) #creates a model for the table view
            self.GraphTable.setModel(self.model)
            self.PlotGraphButton.clicked.connect(self.plot)
            self.EnterPoints.clicked.connect(self.pointenter)
            self.GraphTable.clicked.connect(self.rowupdate)
            self.OpenGraph.clicked.connect(self.graphopener)
            self.BlackLine.clicked.connect(lambda:self.colourclicked("#000000",0))
            self.BlueLine.clicked.connect(lambda:self.colourclicked("#0000ff",0))
            self.RedLine.clicked.connect(lambda:self.colourclicked("#ff0000",0))
            self.GreenLine.clicked.connect(lambda:self.colourclicked("#00ff00",0))
            self.BlackPoint.clicked.connect(lambda:self.colourclicked("#000000",1))
            self.BluePoint.clicked.connect(lambda:self.colourclicked("#0000ff",1))
            self.RedPoint.clicked.connect(lambda:self.colourclicked("#ff0000",1))
            self.GreenPoint.clicked.connect(lambda:self.colourclicked("#00ff00",1))
            self.LinearFit.clicked.connect(lambda:self.plotselect("Linear"))
```

```
        self.PolynomialFit.clicked.connect(lambda:self.plotselect("Polynomial"))
        self.NoFit.clicked.connect(lambda:self.plotselect("None"))
        self.NewGraph.clicked.connect(self.newgraph)
        self.AddGraph.clicked.connect(self.addgraph)
        self.CreateClass.clicked.connect(self.classwindowopen)
        self.JoinClass.clicked.connect(self.joinclassmethod)
        self.InterpolateBtn.clicked.connect(self.interpolate)
        self.ChangePassword.clicked.connect(self.openpassword)
        #associates methods with buttons
    def openpassword(self):
        passwordscreen.show() #opens password change screen when button is pressed
    def plot(self):
        global plotsyle
        global currentrepresentation
        currentrepresentation =[0,0,0,0,0,0] #resets representation
        if self.currentgraph!=-1:
            statistics =statisticsgeneration(self.currentgraph) #generates statistics for graph
            if statistics[0]!="Error":
                xlist=statistics[0]
                sumx=statistics[1]
                sumxsquared=statistics[2]
                sumy=statistics[3]
                sumxy=statistics[4]
                minx=statistics[5]
                maxx=statistics[6]
                miny=statistics[7]
                maxy=statistics[8]
                n=statistics[9]
                points=statistics[10]
                ylist=statistics[11]
                self.MeanxLbl.setText(str(sumx/n)) # changes labels based upon values
                self.MeanyLbl.setText(str(sumy/n))
                self.nofpointsLbl.setText(str(n))
                cur.execute("select Title,XTitle,YTitle from Graphs where GraphID=?;",(self.currentgraph,))
                titles = cur.fetchall() # changes titles
                titles=str(titles[0])
                titles=titles.replace("'","")
                titles=titles.replace("(","")
                titles=titles.replace(")","")
                titles=titles.split(",")
                title =  titles[0]
                title = "'"+title+"'"
                xtitle= titles[1]
                xtitle = "'" +xtitle+"'"
                ytitle=titles[2]
                ytitle = "'" +ytitle+"'"
                if plotstyle == "Linear":
                    linearregression(sumx,sumy,sumxy,sumxsquared,n) #calculates values if linear is selected
                if plotstyle == "Polynomial":
                    polynomialregression(xlist,ylist)
                    # calculates polynomial values for polynomial selected
                file = open('javascript3.txt', 'r')
                htmltext=file.read()
                global colours
                #0 - line 1 - point
                print(plotstyle)
                values=[maxx,minx,maxy,miny,points,currentrepresentation,title,xtitle,ytitle,colours[0],colours[1]]
                replacelist=["#maxx#","#minx#","#maxy#","#miny#","#points#","#equation#","#title#","#xtitle#","#ytitle#","#linecol#","#pointcol#"]
                for i in range(len(replacelist)):
                    htmltext = htmltext.replace(replacelist[i],str(values[i]))
                    #replaces each value in javascript with values calculated
                self.GraphView.setHtml(htmltext)
            else:
                self.GraphView.setHtml("")

    def pointenter(self):
        global userid
        xpoint = self.XValueEdit.text()
        ypoint = self.YValueEdit.text()
        if self.currentgraph== -1:
            self.PointEntryError.setText("No Graph is selected") # checks that graph is selected
        else:
            cur.execute("Select XValMin, XValMax, YValMin, YValMax from Graphs where GraphID =?;",(self.currentgraph,))
            comparisondata= fetchallstripper(cur.fetchall()[0])
            comparisonlist = comparisondata.split(" ") # finds maximum and minimum values for graph
            try:
                xpoint = float(xpoint)
                ypoint = float(ypoint) # turns x and y into floats
                for i in range(len(comparisonlist)):
                    comparisonlist[i] = float(comparisonlist[i])  # turns each max and min into floats
```

```
                    if xpoint >= comparisonlist[0] and xpoint <= comparisonlist[1] and ypoint > comparisonlist[2] and ypoint < comparisonlist[3]: #checks thats values are within max and min
                        print("running sql")
                        cur.execute("INSERT INTO DataPoints(GraphID,XVal,YVal,UserID) VALUES (?,?,?,?);",(self.currentgraph,xpoint,ypoint,userid,))  #if they are add data point
                        print("sql fine")
                        con.commit()
                        self.PointEntryError.setText("")
                        self.YValueEdit.setText("")
                        self.XValueEdit.setText("") # resets text values
                    else:
                        self.PointEntryError.setText("The values given are not within the range specified")
                except:
                    self.PointEntryError.setText("The entry is not a number")
        def graphopener(self):
            if self.row!=-1: #if a row is selected
                self.currentgraph = str(self.model.data(self.model.index(self.row,0))) # gets graphID and sets it to current graph selected
                self.GraphSelectError.hide()
                self.CurrentGraphLabel.setText("Graph " + self.currentgraph + " Selected")
            else:
                self.GraphSelectError.show()
        def rowupdate(self):
            self.row = self.GraphTable.currentIndex().row() # sets the row to the one selected on click
        def changetableview(self,query):
            cur.execute(query)
            data = cur.fetchall()
            self.model.clear()
            self.tableupdate(data) # executes query and fetches the results, uses these for table update
        def tableupdate(self,data):
            for row in data:
                items = []
                for field in row:
                    items.append(QtGui.QStandardItem(str(field))) #adds each item to the model for the table view
                self.model.appendRow(items)
                titles = ["GraphID","Title","X Title","Y Title","Min X","Max X","Min Y","max Y", "Date Created", "Graph Code"]
                self.model.setHorizontalHeaderLabels(titles) #adds titles to model
                self.proxy = QtGui.QSortFilterProxyModel(self)
                self.proxy.setSourceModel(self.model)
        def classwindowopen(self):
            classcreatescreen.show() #opens class create screen

        def addgraph(self):
```

```
            error = False
            trialcode = self.AddGraphEdit.text()
            cur.execute("select GraphID from Graphs where GraphCode=?;",(trialcode,)) #finds graphs with same code
            try:
                AddGraphResult = fetchallstripper(cur.fetchall()[0])
            except:
                error = True
                self.NewGraphStatus.setText("This graph is not available") # checks if graph is available
            if error == False:
                global currentuser
                cur.execute("select StudentID from Students where Username=?;",(currentuser,))
                userid=fetchallstripper(cur.fetchall()[0]) # finds userID
                try:
                    cur.execute("select PermissionID from StudentPermissions where GraphID=? and studentID=?;",(AddGraphResult,userid,))
                    existencechecker = fetchallstripper(cur.fetchall()[0])
                    self.NewGraphStatus.setText("You already have permission for this graph") #checks if a permission for the student and graph already exists
                except:
                    cur.execute("insert into StudentPermissions(GraphID,StudentID)values(?,?);",(AddGraphResult,userid,))#adds a permission if it is validated succesfully
                    con.commit()
                    self.changetableview(("select Graphs.* from Graphs INNER JOIN StudentPermissions ON Graphs.GraphID = StudentPermissions.GraphID where StudentPermissions.StudentID = ? order by GraphID Desc;").replace("?",userid))
                    self.AddGraphEdit.setText("") #changes tableview to display the data
                    self.NewGraphStatus.setText("Please Enter The Code For A New Graph")
        def newgraph(self):
            graphcreatescreen.show() #shows graph create screen

        def colourclicked(self,colour,identifier):
            global colours
            colours[identifier]=colour #changes colours with radio buttons
        def plotselect(self,plottype):
            global plotstyle
            plotstyle = plottype #changes plot style with radio buttons

        def joinclassmethod(self):
            error = False
            ClassCode = self.JoinClassEdit.text()
            self.JoinClassStatus.setText("")
            cur.execute("select ClassID from Classes where ClassCode=?;",(ClassCode,)) #finds classes with the classcode entered.
            try:
                QueryResult = fetchallstripper(cur.fetchall()[0])
            except:
```

440 continued

```
tID = ? order by GraphID Desc;").replace("?",userid))
```

```
                error = True
                self.JoinClassStatus.setText("This class does not exist") #checks if class exists already
            if error == False:
                global currentuser
                cur.execute("select StudentID from Students where Username=?;",(currentuser,))#finds studentID
                userid=fetchallstripper(cur.fetchall()[0])
                try:
                    cur.execute("select EntryID from StudentEntry where ClassID=? and studentID=?;",(QueryResult,userid,))
                    existencechecker = fetchallstripper(cur.fetchall()[0])
                    self.JoinClassStatus.setText("You already have permission for this Class") #checks if a permission already exists
                except:
                    cur.execute("insert into StudentEntry(StudentID,ClassID)values(?,?);",(userid,QueryResult,))
                    con.commit()
                    self.JoinClassEdit.setText("") #adds permission if validation is successful
                    self.JoinClassStatus.setText("Enter the class code for the class you desire to join")

        def interpolate(self):
            global currentrepresentation
            xval = self.XEdit.text()
            yval = 0
            try:
                xval = float(xval) #turns x value into a float
                for i in range(len(currentrepresentation)):
                    yval = yval + currentrepresentation[i]*(xval**(5-i)) #calculates y based upon x value entered and equation
                self.Xinterp.setText(str(xval))
                self.YInterp.setText(str(yval)) #changes display values
                self.InterpError.setText("")
            except:
                self.InterpError.setText("This value is not a float")


    class GraphCreateClass(QtGui.QMainWindow,graphcreate):
        def __init__(self,parent=None):
            QtGui.QMainWindow.__init__(self,parent)
            self.setupUi(self)
            self.Create.clicked.connect(self.graphcreation)
        def graphcreation(self):
            date = str(datetime.datetime.now().year)+"-"+str(datetime.datetime.now().month)+"-"+str(datetime.datetime.now().day) #gets current date
            error = False
            maxminlist= [self.MinX,self.MinY,self.MaxX,self.MaxY]
```

```
            titlelist = [self.Title,self.XTitle,self.YTitle]
            for each in titlelist:
                if len(each.text())<3 or len(each.text())>25:
                    self.Error.setText("Titles must be between 3 and 25 characters long") #checks that titles are of a correct length
                    error = True
                else:
                    for each in maxminlist:
                        try:
                            float(each.text().strip()) #turns each value into a float
                        except:
                            self.Error.setText("Max and min values must be real numbers") #if they can't be turned into floats they must not be real values
                            error = True
                    if error == False:
                        if self.MinX.text() >= self.MaxX.text() or self.MinY.text()>=self.MaxY.text():
                            error = True
                            self.Error.setText("Maximum must be greater than Minimum") #checks that max > min
            if error == False:
                dbvals = []
                self.Error.setText("")
                for each in maxminlist:
                    dbvals.append(each.text())
                    each.setText("") # adds max and mins to database values
                for each in titlelist:
                    dbvals.append(each.text())
                    each.setText("") # adds titles to database values
                newcode = codecreate()
                cur.execute("INSERT INTO Graphs(Title,XTitle,YTitle,XValMin,XValMax,YValMin,YValMax,DateCreated,GraphCode) Values (?,?,?,?,?,?,?,?,?);",(dbvals[4],dbvals[5],dbvals[6],float(dbvals[0]),flc
                con.commit()
                #inserts graph into database
                graphscreen.changetableview("select * from Graphs order by GraphID Desc") #allows teacher to see all graphs including new graph
                self.hide

    class ClassCreateClass(QtGui.QDialog,classcreate):
        def __init__(self,parent=None):
            QtGui.QDialog.__init__(self,parent)
            self.setupUi(self)
            self.Create.clicked.connect(self.classvalidation)
        def classvalidation(self):
            ClassNameString = str(self.ClassName.text())
            SubjectString = str(self.Subject.text())
```

527 continued

```
(?,?,?,?,?,?,?,?,?);",(dbvals[4],dbvals[5],dbvals[6],float(dbvals[0]),float(dbvals[2]),float(dbvals[1]),float(dbvals[3]),date,newcode,))
```

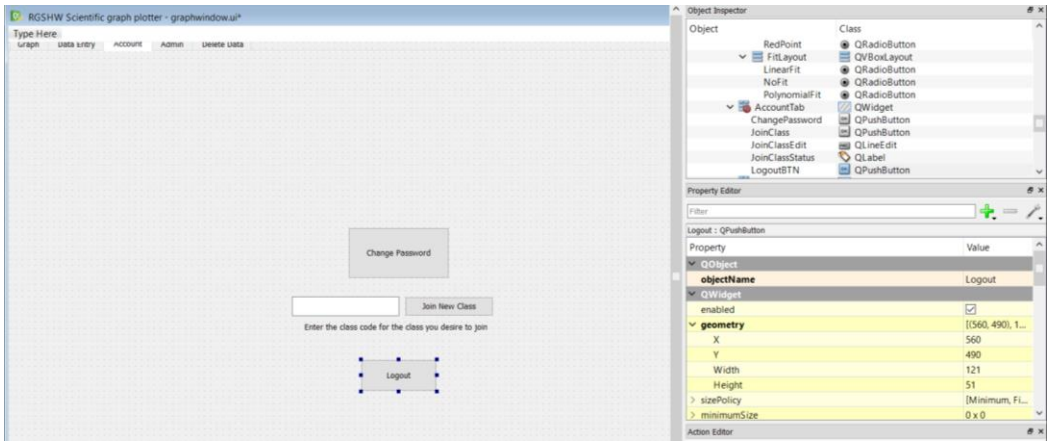
```
            YearVal = int(self.SpinYear.value())
            if len(ClassNameString)> 3 and len(ClassNameString)< 20 and len(SubjectString)> 3 and len(SubjectString)< 20: #checks that strings are correct length
                NewClassCode = codecreate()
                cur.execute("insert into Classes(ClassName,Year,Subject,Classcode) values (?,?,?,?);",(ClassNameString,YearVal,SubjectString,NewClassCode)) #adds class to database
                con.commit()
                self.ClassName.setText("")
                self.Subject.setText("")
                self.hide()
            else:
                self.Error.setText("Text values must be between 3 and 20 characters") #sets an error if the lengths are incorrect

    class PasswordWindowClass(QtGui.QDialog,passwordclass):
        def __init__(self,parent=None):
            QtGui.QDialog.__init__(self,parent)
            self.setupUi(self)
            self.Change.clicked.connect(self.passwordchange)
        def passwordchange(self):
            global currentuser
            trialpass = self.Current.text()
            desiredpass = self.Desired.text()
            cur.execute("SELECT Password FROM Students WHERE Username = ?;",(currentuser,)) #finds hashed pass for current user
            dbpass = fetchallstripper(cur.fetchall()[0])
            trialpass=trialpass.encode('utf-8')
            trialpass = hashlib.sha512(trialpass).hexdigest() #hashes entered password
            print(trialpass)
            print(dbpass)
            dbpass=dbpass.replace("'","")
            if trialpass == dbpass: #if these values are the same check new password
                if len(desiredpass)> 20 or len(desiredpass) < 7:
                    self.errorlabel.setText("This password isn't between 7 and 20 characters long") #checks strength of new password
                else:
                    desiredpass=desiredpass.encode('utf-8')
                    desiredpass = hashlib.sha512(desiredpass).hexdigest()
                    cur.execute("Update Students Set Password =? where username = ?",(desiredpass,currentuser,)) #if valid, adds new password to database
            else:
                self.errorlabel.setText("Your current password is incorrect") #shows error

if error != True:
    apptemplate = QtGui.QApplication(sys.argv)
    classcreatescreen=ClassCreateClass(None)
```

```
    loginscreen = LoginWindowClass(None)
    passwordscreen = PasswordWindowClass(None) #sets up objects for the UI
    registerscreen = RegisterWindowClass(None)
    graphscreen = GraphWindowClass(None)
    graphcreatescreen = GraphCreateClass(None)
    loginscreen.show() #shows login at start
    apptemplate.exec()
```

Log in and out

After looking back at my project I noticed that I had not provided a method for logging back out of the program once logged in. To be able to log out I need to first close the main UI file and open the login screen again. I must also remove the graphs and set variables back to initial values

I must also add the physical button in the UI file to allow them to do this.



I have added the logout button onto the screen. It has been given the name “LogoutBTN”

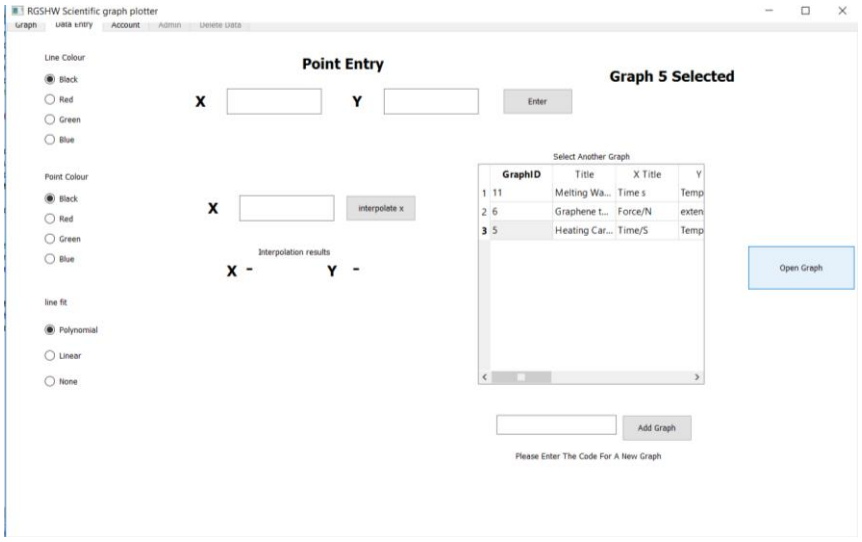
Now I need to add the code for the button being pressed.

```
self.logoutBTN.clicked.connect(self.logout)
```

I have added this so that when “logoutBTN” is clicked it calls the method logout, which I will now define.

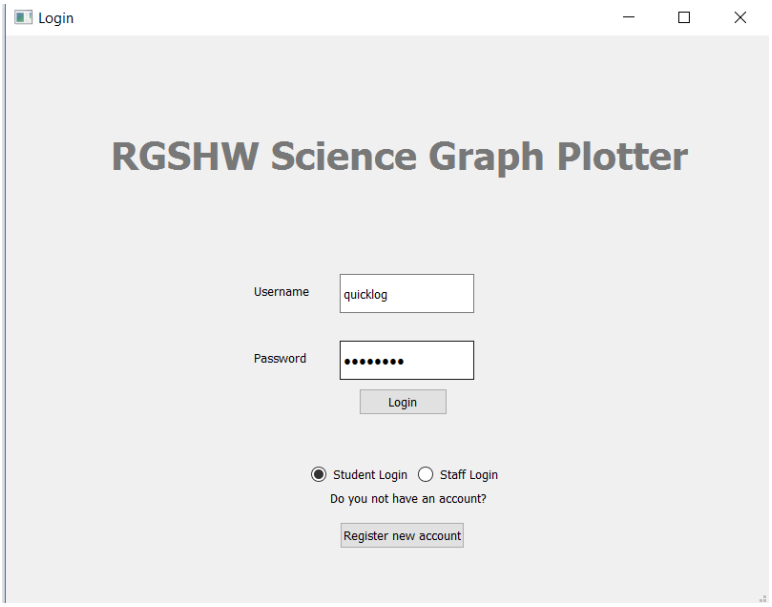
```
312        def logout(self):
313            global currentuser
314            global userid
315            graphscreen.hide() #hides graph screen
316            self.currentgraph = -1
317            self.row = -1 #sets selected graphs to not selected so that users may not access the last graph used
318            currentuser = ""
319            userid = ""
320            loginscreen.show() #shows the login screen
321
```

This resets the values that could be used by the other user and then hides the graph screen and shows the login.



I select graph 5 under the user “quicklog”

I now press logout.

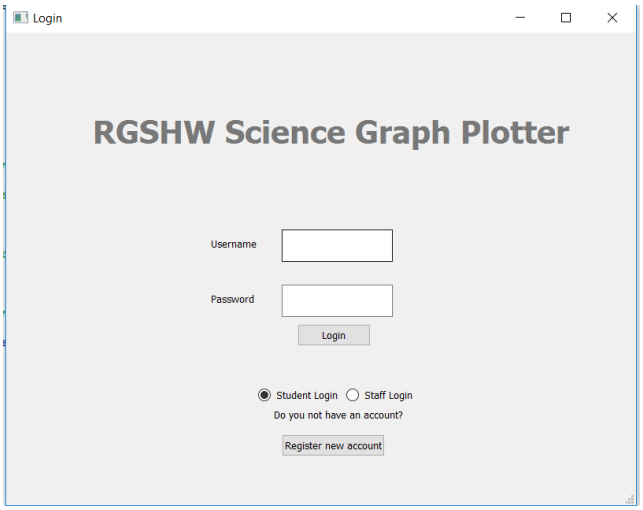


The login credentials remain in the boxes. These must be removed otherwise the next user may be able to access the account again.

```
312         def logout(self):
313             global currentuser
314             global userid
315             graphscreen.hide() #hides graph screen
316             self.currentgraph = -1
317             self.row = -1 #sets selected graphs to not selected so that users may not access the last graph used
318             currentuser = ""
319             userid = ""
320             loginscreen.show() #shows the login screen
321             loginscreen.Username.setText("")
322             loginscreen.Password.setText("")#resets username and password boxes.
```

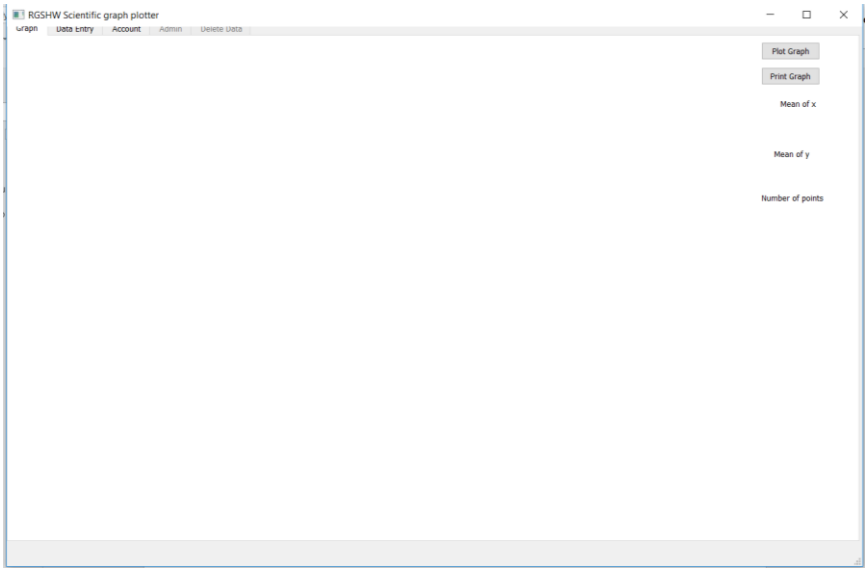
This code has now been added to address this issue.

The same test is done.



The credentials are now deleted.

I will now attempt to login using the account “WilliamT99”



I am able to log into the new account. Therefore, this is successful.

Password checking and regular expressions

I decided to check the strength of the passwords entered further. This is important such that login credentials aren't easily guessed, and also means that they are not likely in databases of reverse hashes. This improves security.

I decided that a strong password has the following features:

- At least 1 letter
- At least 1 number
- Is between 7 and 20 characters in length
- Has at least 1 capital

I decided to use regex to do this to provide a quick way of creating and testing the validation while maintaining readability by using a standard syntax.

This is the code I created:

```
import re
def passwordcheck(password):
    if re.match(r"^.*[a-z]+.*$",password) and re.match(r"^.*[0-9]+.*$",password):
        print("num and letter check passed")
        if re.match(r"^.{7,20}$",password):
            print("length test passed")
            if re.match(r"^.*[A-Z]+.*$",password):
                print("capital test passed")
passwordcheck("111111")
```

First off the regex library is imported for use

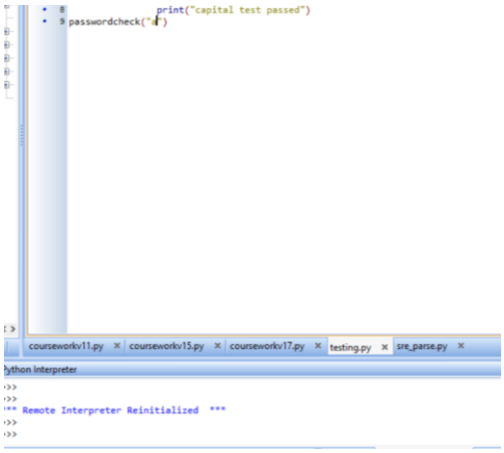
Then the function is defined, taking a string password as an input.

The first check tests if there is any number of characters, then at least one character a-z and then any number of characters. This tests that there is at least 1 letter in the password.

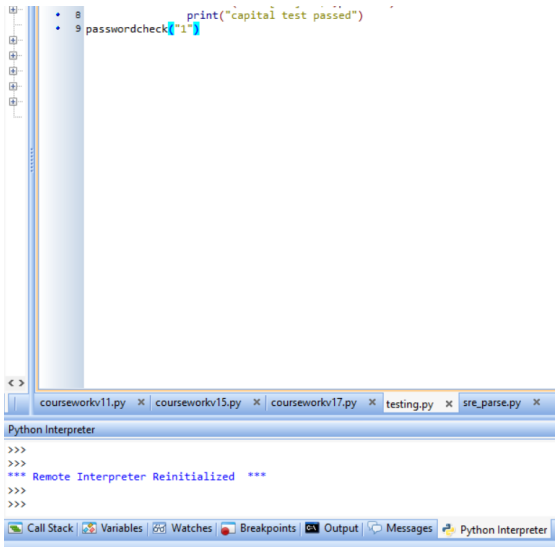
It also has a second mach. This one tests the password for at least one number using the same method. If this is successful a message is printed for testing, showing that the test was a success. The next test checks that there are between 7 and 20 characters in the string.

This also prints a validation statement. Line 7 checks the password for at least 1 capital letter. This works in the same was as line 3.

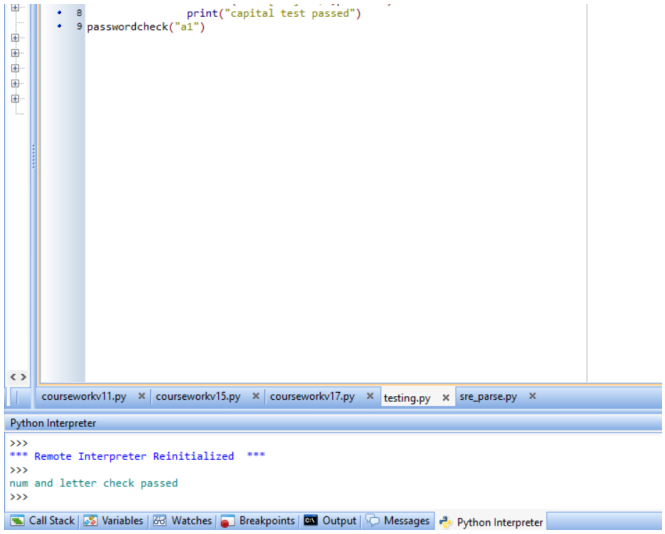
Testing



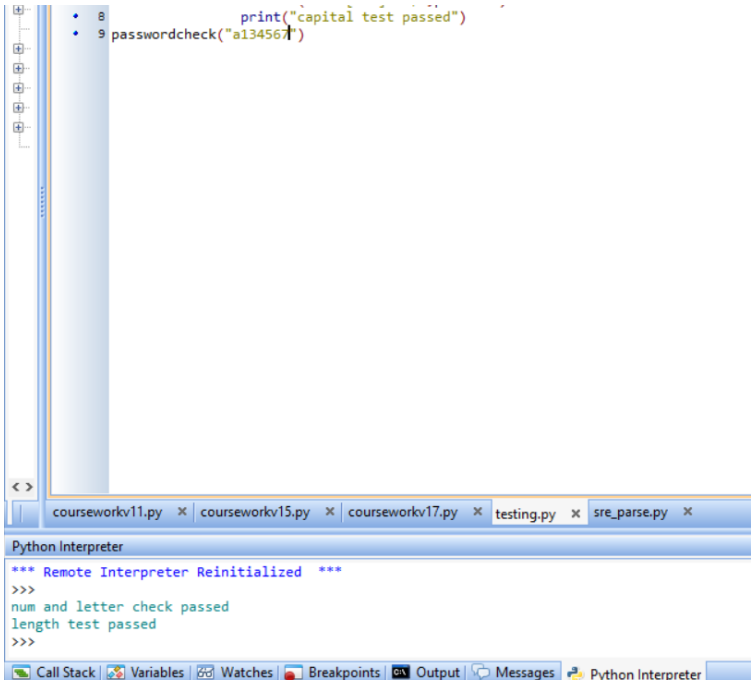
first off I provide the input “a” which produces no output. This is as expected as it does not contain a number



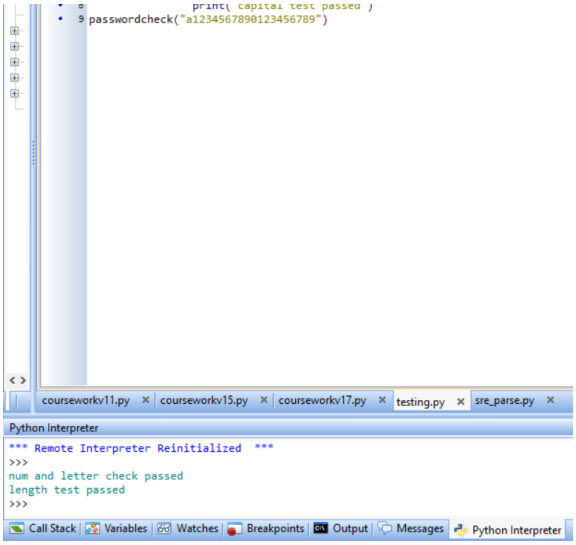
Next “1” is entered. This also fails as there are no letters.



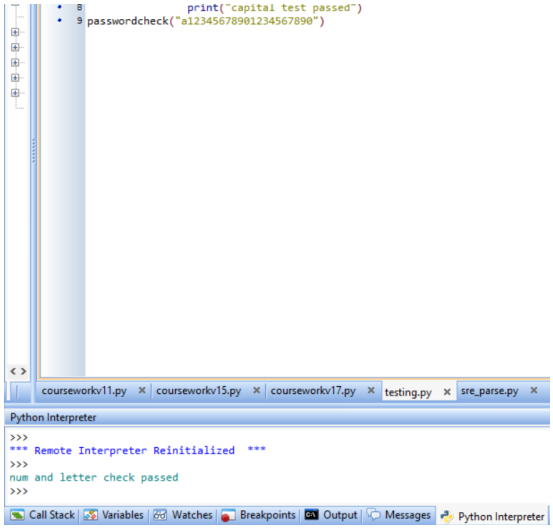
“a1” is now tested. This produces the expected message as it contains both a letter and number. It does not pass the length test which is a success as it is only 2 characters.



7 characters are accepted

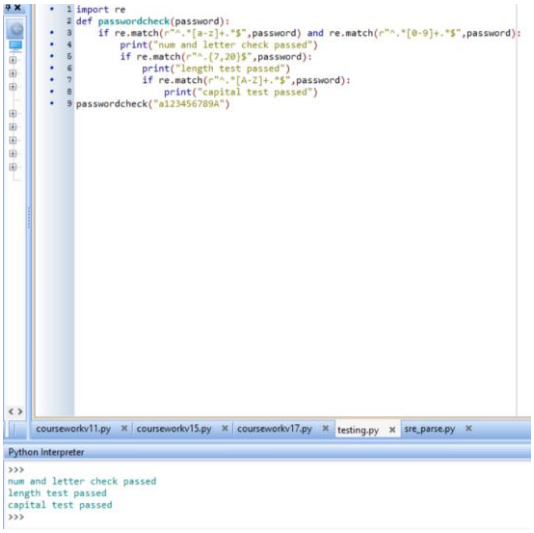


20 characters are accepted



21 characters are rejected, meaning that the length function is working correctly

This did not pass the capital test as expected.



When a capital is added and the length reduced to fit it the test passes, meaning that everything is validated correctly

I have decided to not implement this after talking to my end user (Mrs Dove) as this programme is intended to be used by many people, including year 7s who would forget their passwords with the added complexity, increasing the hassle caused by the program.

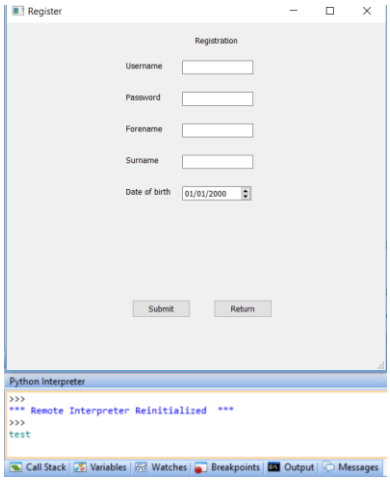
Inheritance

I don't consider a need for inheritance in my program, however after looking back at my program I noticed a place where it could be used. Each window needs to define its main window, and to setup the ui.

```
11 class window(QtGui.QMainWindow):
12     def __init__(self,parent=None):
13             QtGui.QMainWindow.__init__(self,parent)
14             self.setupUi(self)
15
```

A parent class "window" could be inherited by all other windows to avoid rewriting this, however as there are only 3 lines it may not be efficient to type the inheritance each time.

```
12 from PyQt4 import QtCore, QtGui, uic
13 import sys
14 registerclass = uic.loadUiType("registerwindow.ui")[0]
15
16 class window(QtGui.QMainWindow):
17     def __init__(self,parent=None):
18             QtGui.QMainWindow.__init__(self,parent)
19             self.setupUi(self)
20
21 class RegisterWindowClass(window,registerclass):
22     print("Test")
23
24 registerscreen = RegisterWindowClass(None)
```



Even though this works as expected, I don't think it would be useful for my program as inheritance requires a distinct hierarchy between subclasses, and there is no situation where this is needed.

Evaluation

User testing

I gave my program to Mrs Dove for use with her classes for a week for testing with her classes. She has a large number of students (roughly 120), allowing us to have a large range of students testing. We also get to test the admin accounts as Mrs Dove shares some classes with other teachers. After this I did some testing with a number of students to see how quickly they could use the software.

Registering

Student	1	2	3	4
Time(s)	40	56	32	41

This produces an average time of 42.25 seconds. This is considered to be a reasonable time considering there are multiple details to enter – Username, Password, Forename, Surname, and date of birth. Users took a relatively small amount of time to use the date feature, which I thought would take up most of the user's time, however took roughly 15s average. Once the submit button is clicked it is validated and submitted to the database with no visual lag, indicating a quick validation and insertion speed. When compared to google docs it requires a lot less time. I timed myself creating a google account, required for google docs and it takes 39 seconds to enter details for the account. This is possibly faster than the average as I am a relatively quick typer, taking 22 seconds to fill the registration on my program. This login is also followed by a delay, and the need for previous email validation, which takes time varying on email provider.

Login

Student	1	2	3	4
Time(s)	12	9	6	15

This gives an average login time of 10.5 seconds. This is pretty fast and is not much time for login. This varies, depending on size and complexity of username and password, however does not seem much as it is only required once per use of the software. There is also no visible delay once login is clicked as the main window is opened.

Adding a graph

Teacher	1	2	3	4
Time(s)	25	27	19	18

This gives an average of 22.25 seconds per graph. This takes a bit more time than excel as the graph can be created by adding a new document and saving it. That takes about 22 seconds, however with excel you have much fewer variables to enter as there is no option for maximum or minimums, which reduces functionality.

Plotting a graph

With my program, I can create a graph instantly just by pressing a button. The graph also seems to display almost immediately after "plot" is pressed. This seems trivial to time so instead I asked the students to create a graph in excel for comparison.

Student	1	2	3	4
Time(s)	6	5	6	4

This is rather quick, however definitely not as quick as my solution.

I asked Mrs Dove to fill out a questionnaire to see if I had met the success criteria from what she had seen in the classroom

Criteria	Success? Y/N/Partial
Collect username for registration	Y
Collect password for registration	y
Collect Forename for registration	y
Collect Surname for registration	y
Input date of birth	Y
Validate string data from registration against pre-set rules such as length.	Y
Existence check user names to ensure that they aren't in use	Y
Check that the date of birth of users is not too early to currently be in year 7	Y
Check that the date of birth is not too late by comparing it against the date at the time of use	Y
Provide an error message should registration data be invalid.	Y
If validation is successful, the student account should be added to the database with a unique primary key	Y
Allow the user to leave the registration screen should they want to stop creating an account	Y
Provide an option for student or teacher login (each requires different data so should not share a table)	Y
Hash the password entered by the user	Y
Check the username in the database that has been entered for a password	Y
Compare the hashed password in the database against the hash of the password provided by the user	Y
If the user enters the correct details, change the user to the username entered and open the main window	Y
If the password entered was incorrect an error message should be provided	Y
Allow teachers to create classes	Y
Allow the facility for teachers to input class name	Y
Allow the teacher to input a subject name	Y

Allow the teacher to input the year of the students in the class	Y
Validate that the year is between 7 and 13	Y
Generate a code for the class	Y
Check that the code is unique	Y
Add the class to the database using the fields that were entered	Y
Allow students to enter the code of a class	Y
Check the database for a class corresponding to said code	Y
Should this code correspond to a class the user should be given permissions for that class	Y
Allow graphs to be created by teachers	Y
Provide the option to allow all members of a class to view the graph	N
Generate a unique code to join the graph	Y
Allow the user to type in codes for graphs	Y
If this corresponds to a graph let the user join it	Y
Display the graphs available to the users for selection	Y
Allow the user to select the graph by clicking on it	Y
Display to the user which graph is selected	Y
Allow students to add points to the graph	Y
Plot the x axis with correct scale	Y
Plot the y axis with correct scale	Y
Add X title to the graph visualisation	Y
Add Y title to the graph visualisation	Y
Add main title to the graph visualisation	Y
Allow the user to select no line of best fit	Partial
Allow the user to select linear line of best fit	Partial
Allow the user to select polynomial line of best fit	Partial
Allow the user to select line of best fit colour from a predetermined selection	Partial
Allow the user to select point colour from a predetermined selection	Partial
Plot points onto correct point based upon the scale	Y
Plot points with the correct colour as selected by the user	Y
Display the line of best fit based upon the user selected colour.	Y
Calculate the statistical value of the gradient and intercept for a linear line of best fit	Y
Use these values to calculate the y values for each x value within the range	Y

	Not plot a part of the line if it is not within the y axis.	Y
	Calculate the polynomial values for the line of best fit when polynomial is selected	Y
	Plot the polynomial line of best fit by calculating y value based upon x value	Y
	If the calculated y value is out of range the value shouldn't be plotted.	Y
	Calculate the mean value of the data points	Y
	Display the mean value of the data points	Y
	Calculate the number of points on the current graph	Y
	Display the number of data points for the current graph	Y
	Allow a user to change their password	Y
	Validate the password entered against requirements	Y
	Display the mean of Y values	Y
	Interpolate X values given	Y

The only feature not implemented from the success criteria is criteria 31 - Provide the option to allow all members of a class to view the graph. This is because development time ran out to implement this. I spoke to Mrs Dove and she said it was fine to go into testing without this feature, as it can be added once I start creating the final version. This is only a minor feature, but it would be great to add this in a future part of development.

I also asked Mrs Dove the question "Is there anything that you dislike about the program in its current state?"

I found the program to be extremely useful in my classes, however there are some parts of the UI I dislike. The graph selection part of the program is small and requires a lot of scrolling to see key details. To improve this I think that this loading part should have its own screen or be larger. It would also be really useful to have the colour selections and the plot style on the screen next to the graph. This would help me to change colours quickly, but more importantly easily compare the different types of lines of best fit for my classes. This is why I have only given a partial success for this section. I would also like the tabs at the top of the page to be moved downwards. This is because the tops of the words are cut off slightly. Students with old computers at home with very small screens had an issue with the UI not fitting on the screen.

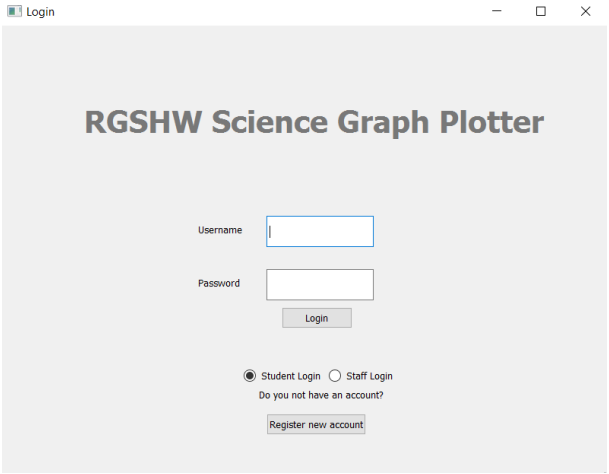
In response to this I believe that the radio button changes would be really useful for the classes. This is a relatively quick fix and may improve the usability of my program massively. This could be done by just moving the widgets in the pyQT QT designer. It would also be rather quick to move the radio buttons ready for the next release of my program. This did not occur while testing, but may have changed in the last development version or could be due to the different screen sizes used.

The problem with smaller screen sizes having broken GUIs is significant, however with my requirements I stated at least a 720p screen. Most people now have 720p screens and reducing the size of the UI would make it very difficult for it to be read. This will become less of a problem over time and will most likely not be an issue within the next few years.

Outputs

To test my outputs I will take a screenshot of each output and ask for a comment from Mrs Dove. I will also comment on the usefulness of the output types.

Login



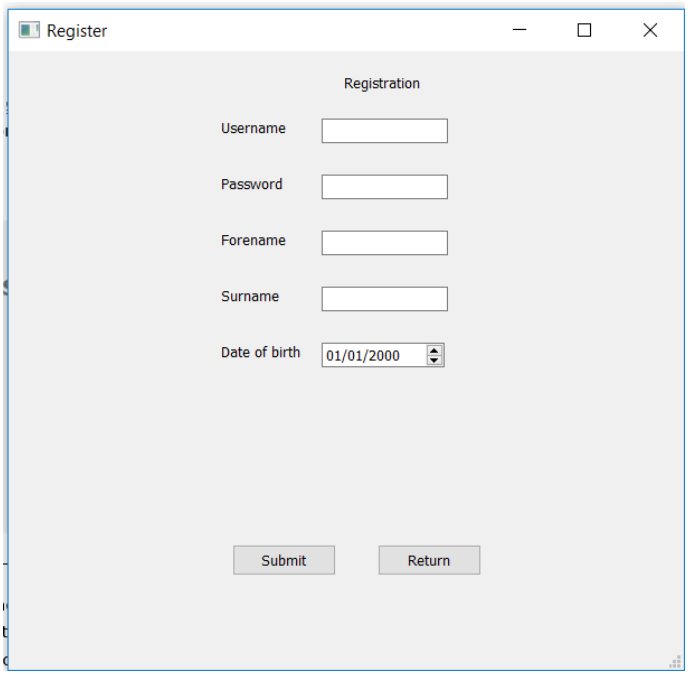
Mrs Dove –

I believe the inputs to be fit for purpose and they look good. There is sufficient space for typing and I like the buttons for student and staff. The register button is well placed. I would like to have a logo for the school under the graph plotter title would add a bit of colour.

My comment –

Adding the logo would be fine as I could move the title up and fit a decently sized logo. I think the basic theme of the screen allows the users to log in quickly.

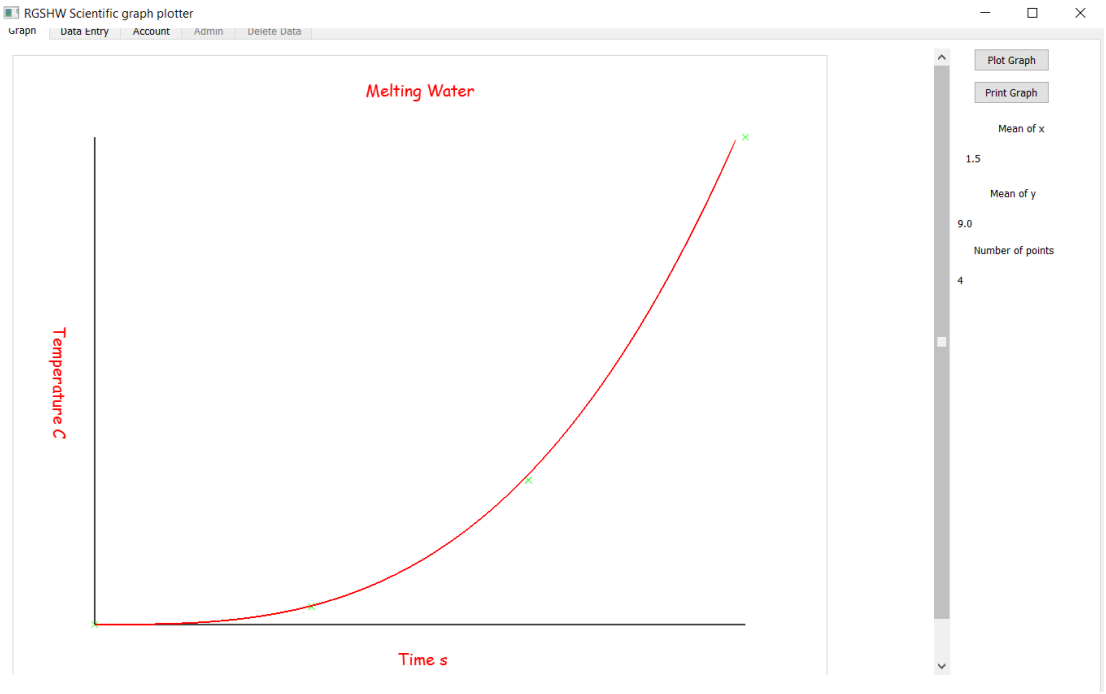
Register



Mrs Dove- I'm not so keen on this window as it seems very dull. It is not used much as each user should only need to use it once. The dimensions seem a bit odd as the submit and return are quite far away from the text inputs. The title would also be nice if it were underlined

My comment – I believe that the UI is very functional and is not good looking because it isn't used much. I have taken on board the comments. I could possibly add a bit of colour to the UI to make it more interesting. I would also resize the window. I would keep the aspect ratio the same, while making the window shorter.

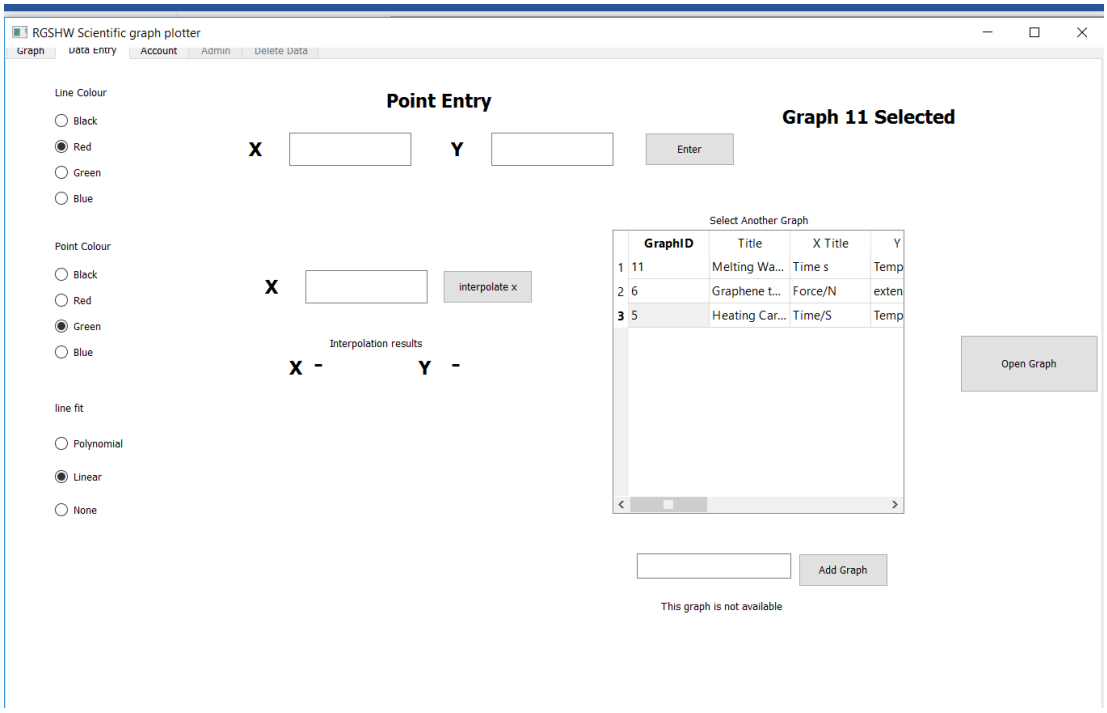
Graph screen



Mrs Dove- I love the graph plotting, it works well and looks great! I think the graphs look smooth and plots really well. The graph takes up most of the screen and the calculated values are visible but don't reduce the size of the graph itself. Changing the colours is really useful.

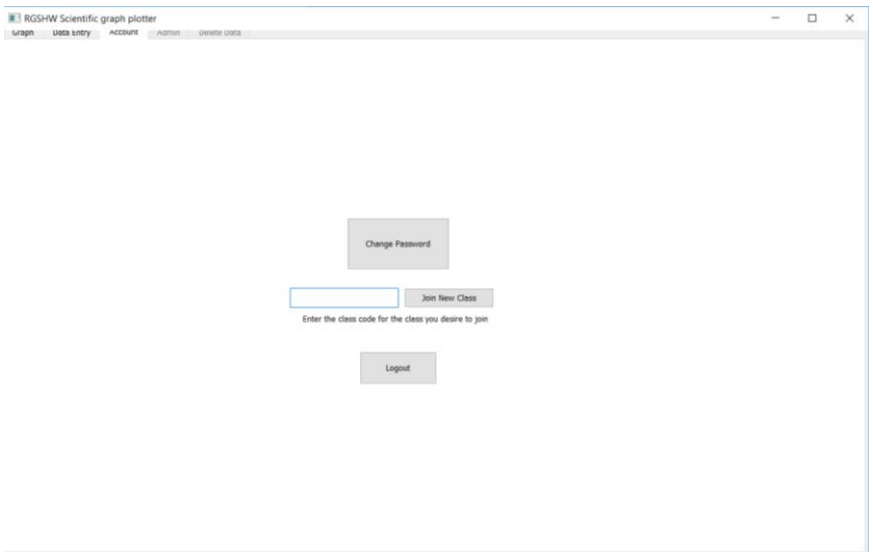
My comment – I was really pleased with how this tab turned out. It was the most important part of the program and making it look good was a big priority.

Data entry screen



Mrs Dove – This screen feels a bit cramped. I would like it if the buttons on the left were used to another screen. I really like the graph selection area, but I would like the selection part to be larger. I like that the graph selected is shown as it helps

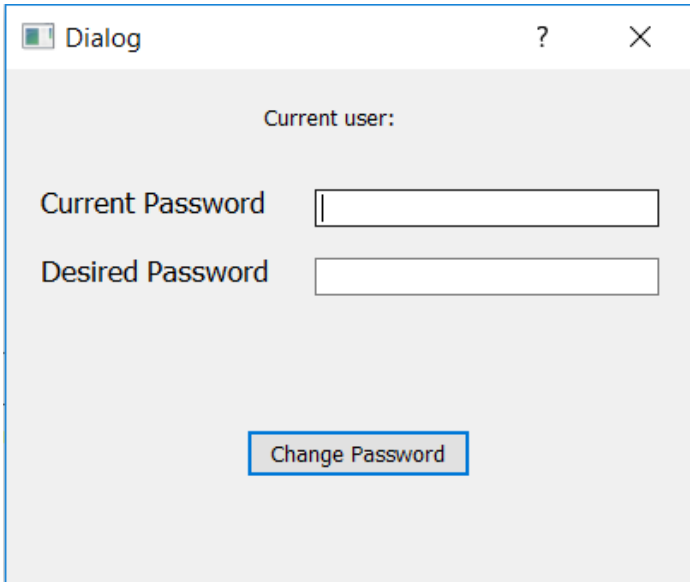
Account page



Mrs Dove – This screen seems to be the parts which don't fit. It may be better to use a drop down menu in another tab to do this.

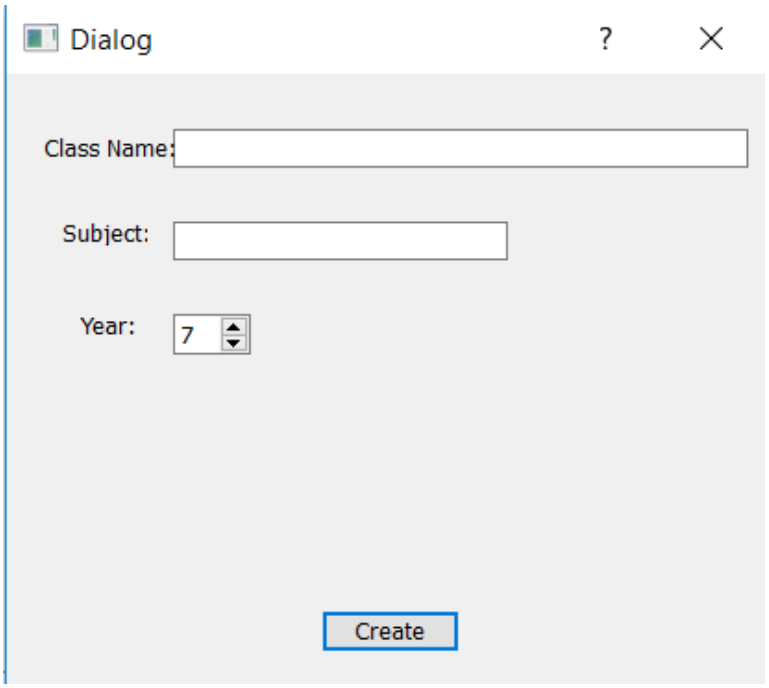
My comment – I do think this is a good idea and would have been good if planned for in the design section. This change may be more complicated but would improve the visuals.

Password change



Mrs Dove – This is a good window as it is short and to the point. It fits its purpose.

Create new class



Mrs Dove – This window fits its purpose

Further additions

To further improve my program, I would first like to add the functionality of adding whole classes to a graph. This would improve the usability and also reduce the time taken to use core functions of the program. I also believe this would reduce the stress on teachers who may need to chase up students to get them to add points to the graph. It would also allow me to add functionality whereby teachers can check which members of a class have added a value to a graph and how many. This could be used to set homework rather than just for class use. I would also add menus for more admin features such as changing students' passwords, possibly sending them an email with a temporary password. For the main graph functionality, I would also add integer values to the x and y axis. This would be more complicated as it would require an algorithm that would give reasonable results, even for graphs with odd dimensions. I would also like to add an option for exponential and logarithmic graphs, as this is often used in physics. This might help to increase the usage of my program.

Porting the program to other formats

To a website

Porting to a website is relatively easy. The whole of the graph plotting algorithm can be used as it is all in JavaScript. It is also easy to create an attractive user interface using HTML and CSS. This could be used as the inputs for my program. This could then be linked to my python using a web server. This would mean that only the user interface would need to be changed

To a mobile device

Mobile devices have much smaller screens aren't very suited to my interface. I would also have to be wary of some of my text inputs as the on-screen keyboard may cover some valuable information such that the user can't see it. The code can be used directly, however the ui would also be the wrong dimensions and may require a complete redesign. There would also be a steep learning curve in trying to use python on an IOS or android device.

To a siri-like interface

An interface like siri is a very interesting idea. I could provide solely the graph screen and the statistics and then use voice control. I could include instructions on how to input data points and other key pieces of functionality. The inputs would need to be remade, possibly using an api for voice recognition. I could use most of my code, except for that used for interface and inputs.