Testing and Evaluation

Blackbox testing (Input and Output)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Description | User Input | Expected Result | Actual Result |
| 1 | Testing the menu’s ‘Calculator’ button. | Clicking the ‘Calculator’ button. | The user is moved to the calculator screen. | As expected. |
| Testing the menu’s ‘Graph’ button. | Clicking the ‘Graph’ button. | The user is moved to the graph creation screen. | As expected. |
| Testing the menu’s ‘Exit’ button. | Clicking the ‘Exit’ button. | The program closes. | As expected. |
| 2.1.1 | Testing the calculator’s numerical buttons (0, 1, 2, 3, etc.). | Clicking the number buttons. | The numbers are appended to the expression to be calculated. | As expected. |
| Testing the calculator’s operator buttons. (+, -, xn, etc.). | Clicking the operator buttons. | The operators are appended when appropriate to the expression to be calculated. | As expected. |
| 2.1.2 | Testing the calculator’s sine function. | Inputting ‘sin(90)’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| Inputting ‘sin(0)’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘sin(180)’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘sin(54.235)’ into the calculator. | The calculator outputs ’0.8114209979’ or some expanded or contracted variation of it. | As expected. |
| Inputting ‘sin(45)’ into the calculator. | The calculator outputs ‘0.7071067812’ or some expanded or contracted variation of it. | As expected. |
| 2.2.1 | Testing the calculator’s cosine function. | Inputting ‘cos(90)’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘cos(0)’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| Inputting ‘cos(180)’ into the calculator. | The calculator outputs ‘-1’. | As expected. |
| Inputting ‘cos(54.235)’ into the calculator. | The calculator outputs ’0.5844621152’ or some expanded or contracted variation of it. | As expected. |
| Inputting ‘cos(45)’ into the calculator. | The calculator outputs ‘0.7071067812’ or some expanded or contracted variation of it. | As expected. |
| 2.2.2 | Testing the calculator’s tangent function. | Inputting ‘tan(90)’ into the calculator. | The calculator expresses a ‘Complex Error’ in red text. | As expected. |
| Inputting ‘tan(0)’ into the calculator. | The calculator outputs ‘0’. | The calculator expresses a ‘Complex Error’ in red text. |
| Inputting ‘tan(180)’ into the calculator. | The calculator outputs ‘0’. | The calculator expresses a ‘Complex Error’ in red text. |
| Inputting ‘tan(54.235)’ into the calculator. | The calculator outputs ’1.388320948’ or some expanded or contracted variation of it. | As expected. |
| Inputting ‘tan(45)’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| 2.2.3 | Testing the calculator’s logarithm function. | Inputting ‘log(1)’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘log(10)’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| Inputting ‘log(100)’ into the calculator. | The calculator outputs ‘2’. | As expected. |
| Inputting ‘log(107)’ into the calculator. | The calculator outputs ‘7’. | As expected. |
| Inputting ‘log(65)’ into the calculator. | The calculator outputs ’1.812913357’ or some expanded or contracted variation of it. | As expected. Gave a more precise value for the output than expected (‘…566’ instead of ‘…57’). |
| 2.2.4 | Testing the calculator’s natural logarithm function. | Inputting ‘ln(1)’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘ln(e)’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| Inputting ‘ln(√e)’ into the calculator. | The calculator outputs ‘0.5’. | As expected. |
| Inputting ‘ln(107)’ into the calculator. | The calculator outputs ’16.11809565’ or some expanded or contracted variation of it. | As expected. Gave a more precise value for the output than expected (‘…651’ instead of ‘…65’). |
| Inputting ‘ln(65)’ into the calculator. | The calculator outputs ‘4.17438727’ or some expanded or contracted variation of it. | As expected. Gave a more precise value for the output than expected (‘…2699’ instead of ‘…27’). |
| 2.2.5 | Testing the calculator’s modulus (absolute value) function. | Inputting ‘abs(1)’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| Inputting ‘abs(-1)’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| Inputting ‘abs(-√e)’ into the calculator. | The calculator outputs ‘1.648721271’ or some expanded or contracted variation of it. | As expected. |
| Inputting ‘abs(-107)’ into the calculator. | The calculator outputs ’10000000’. | As expected. |
| Inputting ‘abs(-π)’ into the calculator. | The calculator outputs ‘3.141592654’ or some expanded or contracted variation of it. | As expected. Gave a more precise value for the output than expected (‘…536’ instead of ‘…54’). |
| 2.3.1 | Testing for a correct formatting of multiplied variable terms. | Inputting ‘2\*x’ into the calculator. | The calculator formats to ‘2x’ as expression is being inputted. | As expected. |
| 2.3.2 | Testing for a correct formatting of multiplied numerical terms. | Inputting ‘2\*2’ into the calculator. | The calculator formats to ‘2×2’ as expression is being inputted. | As expected. |
| 2.3.3 | Testing for a correct formatting of multiplied open brackets | Inputting ‘2\*(’ into the calculator. | The calculator formats to ‘2(’ as expression is being inputted. | As expected. |
| 2.3.4 | Testing for a correct formatting of multiplied open brackets | Inputting ‘…)\*2’ into the calculator. | The calculator formats to ‘…)×2’ as expression is being inputted. | As expected. |
| 2.4.1 | Testing for a correct formatting of single character indices. | Inputting ‘2^2’ into the calculator. | The calculator formats to ‘22’ as expression is being inputted. | As expected. |
| 2.4.2 | Testing for a correct formatting of bracketed (multi-character) indices. | Inputting ‘2^(2\*x)’ into the calculator. | The calculator formats to ‘2(2x)’ as expression is being inputted. | As expected. |
| 2.4.3 | Testing for a correct formatting of single character indices when there is another character after the power. | Inputting ‘2^2+’ into the calculator. | The calculator formats to ‘22+’ as expression is being inputted. | As expected. |
| 2.4.4 | Testing for a correct formatting of single character indices when there is another character after the power. | Inputting ‘2^(2\*x)+’ into the calculator. | The calculator formats to ‘2(2x)+’ as expression is being inputted. | As expected. |
| 2.5.1 | Testing the addition operator. | Inputting ‘2+2’ into the calculator. | The calculator outputs ‘4’. | As expected. |
| Inputting ‘2+(-2)’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘2+-2’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘-2+2’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘2+2.5’ into the calculator. | The calculator outputs ‘4.5’. | As expected. |
| Inputting ‘2+(-2.5)’ into the calculator. | The calculator outputs ‘-0.5’. | As expected. |
| 2.5.2 | Testing the subtraction operator. | Inputting ‘2-2’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘2-(-2)’ into the calculator. | The calculator outputs ‘0’. | As expected. |
| Inputting ‘2--2’ into the calculator. | The calculator outputs ‘4’. | As expected. |
| Inputting ‘-2-2’ into the calculator. | The calculator outputs ‘-4’. | As expected. |
| Inputting ‘2-2.5’ into the calculator. | The calculator outputs ‘-0.5’. | As expected. |
| Inputting ‘2-(-2.5)’ into the calculator. | The calculator outputs ‘4.5’. | As expected. |
| 2.5.3 | Testing the multiplication operator. | Inputting ‘2\*2’ into the calculator. | The calculator outputs ‘4’. | As expected. |
| Inputting ‘2\*(-2)’ into the calculator. | The calculator outputs ‘-4’. | As expected. |
| Inputting ‘-2\*2’ into the calculator. | The calculator outputs ‘-4’. | As expected. |
| Inputting ‘2\*2.5’ into the calculator. | The calculator outputs ‘5’. | As expected. |
| Inputting ‘2\*(-2.5)’ into the calculator. | The calculator outputs ‘-5’. | As expected. |
| 2.5.4 | Testing the division operator. | Inputting ‘2/2’ into the calculator. | The calculator outputs ‘1’. | As expected. |
| Inputting ‘2/(-2)’ into the calculator. | The calculator outputs ‘-1’. | As expected. |
| Inputting ‘-2/2’ into the calculator. | The calculator outputs ‘-1’. | As expected. |
| Inputting ‘2/2.5’ into the calculator. | The calculator outputs ‘0.8’. | As expected. |
| Inputting ‘2/(-2.5)’ into the calculator. | The calculator outputs ‘-0.8’. | As expected. |
| 2.5.4 | Testing the index operator. | Inputting ‘2^2’ into the calculator. | The calculator outputs ‘4’. | As expected. |
| Inputting ‘2/(-2)’ into the calculator. | The calculator outputs ‘0.25’. | As expected. |
| Inputting ‘-2^(0.5)’ into the calculator. | The calculator expresses a ‘Complex Error’ in red text. | As expected. |
| Inputting ‘-2^2.5’ into the calculator. | The calculator expresses a ‘Complex Error’ in red text. | As expected. |
| Inputting ‘2^(-2.5)’ into the calculator. | The calculator outputs ‘0.1767766953’ or some expanded or contracted variation of it. | As expected. Gave a more precise value for the output than expected (‘…52966369’ instead of ‘…53’). |
| 3.1.1 | Testing the graph creation functionality with algebraic graphs. All of these bounds are (-10, 10). | Creating a graph of y=2\*x. | The calculator creates a straight diagonal line from the bottom left to the top right. | As expected. |
| Creating the graph of y=x^2 | The graph creates an upward curving parabola that touches the axes at O(0, 0). | As expected. |
| Creating the graph of y=1/x | The graph creates a standard reciprocal centred on O(0, 0). | As expected. |
|  | The graph creates a standard cubic passing through O(0, 0). | As expected. |
| 3.1.2 | Testing the graph creation functionality with trigonometric graphs. All of these bounds are (-90, 90). | Creating a graph of y=sin(x). | The graph creates a standard sine wave. | As expected. |
| Creating a graph of y=sin(x)+1. | The graph creates a sine wave, shifted up by 1. | As expected. |
| Creating a graph of y=sin(x-10). | The graph creates a sine wave, shifted right slightly. | As expected. |
| Creating a graph of y=cos(x). | The graph creates a standard cosine wave. | As expected. |
| Creating a graph of y=tan(x). (Bounds (0, 360)) | The graph creates a standard tangent wave. | As expected. |

Whitebox testing (Process)

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test Description | Process Output | Final Output |
| 1 | Testing the calculator for the way it handles brackets.  (Inputted “5\*(2-2)”) | 1. STR: [[5, '\*', '(', 2, '-', 2, ')']]  Length: 7  y: 0 1 2B 3 4 5 6E  [0 2] [0 6]1  (...)  1. eu: ['(', 2, '-', 2, ')']2  1a. eu: [2, '-', 2, ')']  1b. eu: [2, '-', 2]  2. eu: [[2, '-', 2]]  3. e\_temp: [[5, '\*', '(', 2, '-', 2, ')']]  4. eu: [[2, '-', 2]]  5. eo: [[2, '-', 2]]  6. eu: [[2, '-', 2]]  7. eo: [['2', '-', '2']]  8. ej: 2-2  2  e: [[2, '-', 2]]  pre-res: [[2, '-', 2]]  S  S: [[0]]  RTRN: 0  res: 0  brac: [[5, '\*', 0]]3  F  [[5, '\*', 0]]  S  [[5, '\*', 0]]  C  [[5, '\*', 0]]  T  [[5, '\*', 0]]  L  [[5, '\*', 0]]  E  [[5, '\*', 0]]  A  [[5, '\*', 0]]  !  [[5, '\*', 0]]  func: [[5, '\*', 0]]  e: [[5, '\*', 0]]  pre-res: [[5, '\*', 0]]  M  M: [[0]]  RTRN: 0  calc: 0  l: 0  0 | 0, the brackets were handled as expected, with the program finding the location of the bracketed expression and isolating it to calculate on it before substituting it back into the expression3. |
| 2 | Test for the way that the calculator substitutes in for variables.  (Inputted “y-5\*x”, where x=3 and y=1) | y-5\*x  0 0 0  y ['x', 3]  0 1 0  - ['x', 3]  0 2 0  5 ['x', 3]  0 3 0  \* ['x', 3]  0 4 0  x1 ['x', 3]  0 0 1  y2 ['y', 1]  0 1 1  - ['y', 1]  0 2 1  5 ['y', 1]  0 3 1  \* ['y', 1]  0 4 1  32 ['y', 1]  0 0 2  14 ['π', 3.14159265359]  0 1 2  - ['π', 3.14159265359]  0 2 2  5 ['π', 3.14159265359]  0 3 2  \* ['π', 3.14159265359]  0 4 2  3 ['π', 3.14159265359]  0 0 3  1 ['φ', 1.61803398875]  0 1 3  - ['φ', 1.61803398875]  0 2 3  5 ['φ', 1.61803398875]  0 3 3  \* ['φ', 1.61803398875]  0 4 3  3 ['φ', 1.61803398875]  0 0 4  1 ['e', 2.71828182846]  0 1 4  - ['e', 2.71828182846]  0 2 4  5 ['e', 2.71828182846]  0 3 4  \* ['e', 2.71828182846]  0 4 4  3 ['e', 2.71828182846]  Combination:  1 (0, 0, 0)  - (0, 1, 0)  5 (0, 2, 0)  \* (0, 3, 0)  3 (0, 4, 0)  INT: [[1, '-', '5', '\*', '3']] | -14, the program worked as expected, running through each character in the expression and matching it with a character contained within the list of variables, and substituting it with the value associated with it, as can be seen by the difference in item value between 1 and 3, and 2 and 4. The final line shows the fully substituted expression – “1-5\*3”. |
| 3 | Test for the way that the calculator conjoins individual characters to form a multiple digit number.  (Inputted “546-123”) | 5 (0, 0, 0)  [['54', '6', '-', '1', '2', '3']]1  54 (0, 1, 1)  [['546', '-', '1', '2', '3']]2  546 (0, 2, 2)  - (0, 3, 2)  1 (0, 4, 2)  [['546', '-', '12', '3']]  12 (0, 5, 3)  [['546', '-', '123']]  123 (0, 6, 4)  INT: [[546, '-', '123']]  INT: [[546, '-', 123]]3 | The progam worked as expected, searching through the list and combining each individual digit character into a string of digits, ignoring non-numerical characters and substitutes it back into the expression.2 It then converts it into an integer.3 |
| 4 | Test for the way that the program creates curve coordinates.  (Created a graph of “y=2\*x”, bounds -10 and 10) | ['2\*x']  ['2\*x']  Equation:  Curves: ['2\*x']  Given Bounds: ['-10', '10']  -10  10  Calculated Bounds: [-10, 10]  2  0  2  y: 2\*x  dy/dx: 2  -20  2  2\*x: -20 2 0%  -19.916666666666668  2  2\*x: -19.916666666666668 2 0.208%  -19.833333333333332  2  2\*x: -19.833333333333332 2 0.417%  -19.75  2  2\*x: -19.75 2 0.625%  -19.666666666666668  2  2\*x: -19.666666666666668 2 0.833%  -19.583333333333332  2  2\*x: -19.583333333333332 2 1.042%  -19.5  2  2\*x: -19.5 2 1.25%  -19.416666666666668  2  2\*x: -19.416666666666668 2 1.458%  -19.333333333333332  2  2\*x: -19.333333333333332 2 1.667%  -19.25  2  2\*x: -19.25 2 1.875%  -19.166666666666668  2  2\*x: -19.166666666666668 2 2.083%  -19.083333333333332  2  2\*x: -19.083333333333332 2 2.292%  -19  2  2\*x: -19 2 2.5%  -18.916666666666668  2  2\*x: -18.916666666666668 2 2.708%  -18.833333333333332  2  2\*x: -18.833333333333332 2 2.917%  -18.75  Etc. | Created as expected, the program took the curves given to it in a list, and also took the bounds in a list in string format and calculated them. It then found the expression for y as the equation given, and found its differential as dy/dx. It then calculated y for each value of x as expected, and increased the completion percentage by 0.208% for each iteration, as that is 100 / 480, the screen width. |

User Review Questionnaire

These are a series of questions about the efficacy and usability of the graphing calculator program. Please answer them fully and accurately.

1.       Do you feel that the program carries out its intended tasks well? How do you think these tasks are met?

*In some cases it works well, performing calculations correctly but it does not always give correct answers, eg cos(60)=1 (which wouldn’t be true for degrees or radians), log(0)=0, e(-2)=0.*

2.       Do you feel that the program is intuitive and easy to use? If so, please state any features to this effect. If not, please state how you think usability could be improved.

*For one step arithmetic, yes. It is not always clear how to enter an argument, for example using e. Also, once you have a calculation and get a response, you can then start adding further digits which doesn’t make sense. As an example input 5x6= and get 30, but then if I then input 1 I get a display of 5x61.*

3.       How do you feel the program could be improved in order to improve its effectiveness?

*Inputs need to be stricter and errors more explicit.*

4.       Have you encountered any bugs or crashes while using this program? If so, please state them.

*Some examples of bugs are given above. There were no crashes.*

5.       Do you have any features that you would like to see added in the future? If so, please state them.

* *Scales added to the graphing feature.*
* *The ability to zoom in or out.*

Evaluation and issues with the system

I believe the solution generally matches the requirements given in the design section by the stakeholders, and by my own metric and estimation. The program correctly calculates input given to it, and correctly performs the calculations necessary to properly graph multiple and various different types of equations. The program also formats the expressions given to it using algebraic convention, properly superscripting indices and properly eliminating multiplication symbols in between variables and brackets.

However, there are some issues with the underlying design of the solution itself which as outlined in the user review questionnaires, would make it seem as if the program performed an incorrect calculation. This, I believe is to do with the rounding functionality of the solution, where the answer is automatically rounded to a certain number of decimal places upon calculation. This could make it seem as if the program incorrectly calculated decimal answers as whole numbers, especially because by default, the program rounds numbers to a whole number instead of to the maximum of 10 decimal places.

There are multiple ways this could be fixed, including either making the program by default set the value in the calculator’s spin box to 10 instead of 0 in case the user forgets to change its value, or making the user have to manually round the answer to their desired number of decimal places instead of it being automatically rounded. This second solution would allow for the user to first see the more precise answer to their problem, before deciding to round it, rather than mistakenly believing that the imprecise answer was the solution instead.

In addition, the program also allows one to continue typing out an expression even when an answer has been calculated, despite most calculators clearing the display after calculating an answer to allow for the input of a new expression. This may be a problem for those who would expect the display to be cleared, however it requires only a simple fix, only requiring the redefinition of the self.Equation variable to “” after an answer has been calculated. This would clear the display, whilst also removing the value for self.CalculatorExpression.

There are multiple other ways that this program could be improved. There have been many issues with the distribution of the software, mostly owing to the difficulty in adapting it to different systems. This was mainly due to the lack of an inherent installation for PyQt4 in the Python directory. This could be solved by packaging the software as an executable file, bundling it with the PyQt4 folder so as to not require its installation on each system it was run on.

In addition, to improve the graphing of curves, it could also be useful to colour each of the curves drawn to allow for easier differentiation by the end user. This would allow for the easier perception of the shape of the curves, as well as a simpler differentiation between the coloured curves and the black axes. In addition, it would also allow for intersection points to be more easily determined as well, as the overlap of colours would mark these points.

Additionally, Issues with graph presentation arise when graphing equations in the software. As the difference in bounds becomes larger, the curves would become squashed due to the width of the window being constant instead of changing. This would be fixed in future versions of the program, where the graph bounds could become moveable using a scroll wheel that would move through the graph. However, if this was done, the end user would only be able to see one section of the graph at a time.

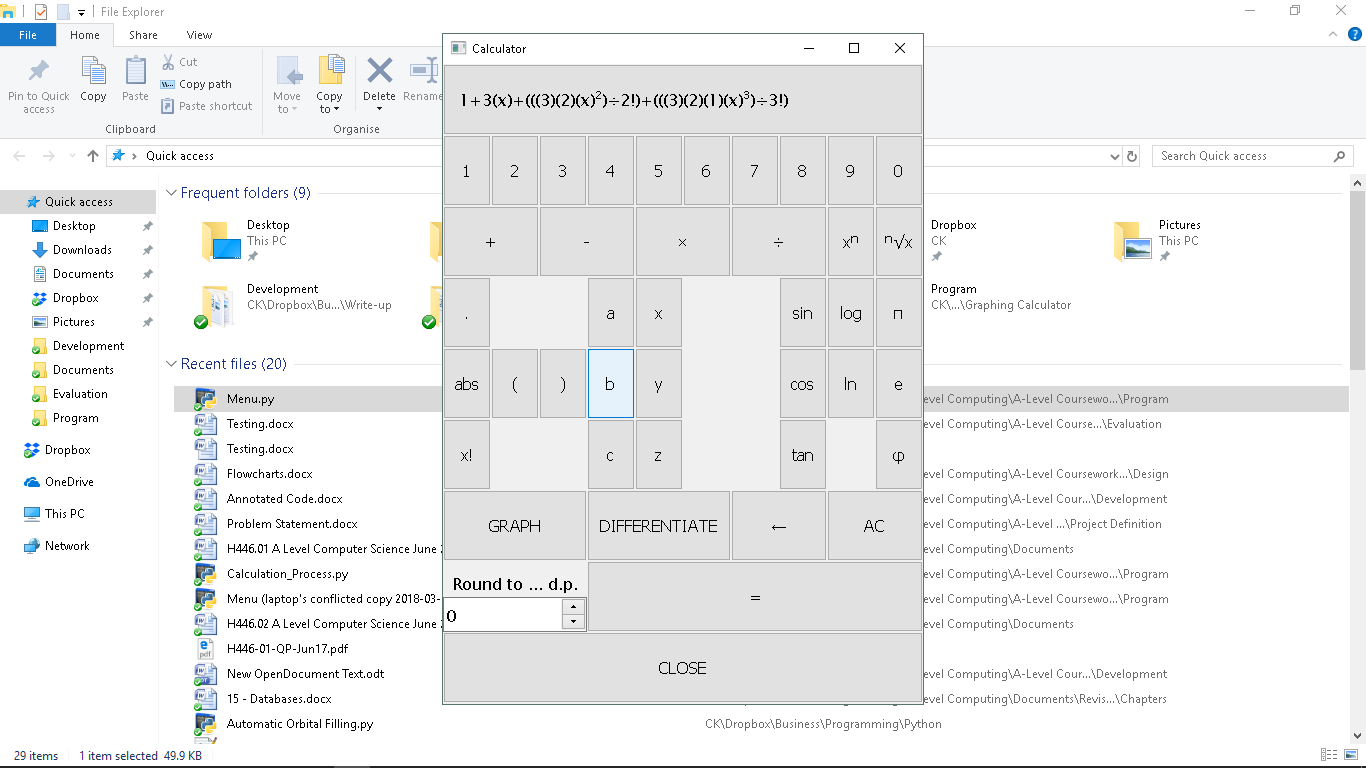
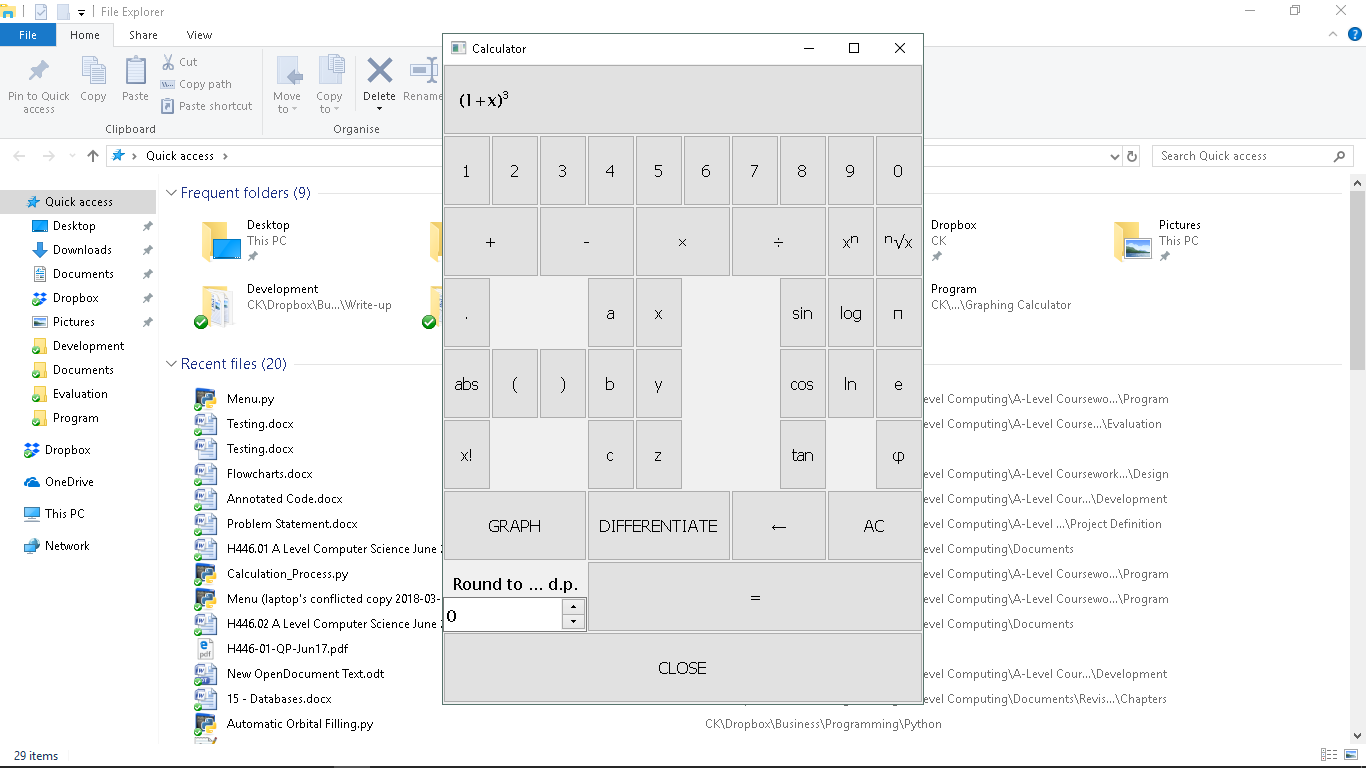
This could be alleviated by following the request in the answer to question five and including an ability to zoom in and out to magnify or minimise focus on certain sections or the graph as a whole. This would also allow for closer inspection of the graph as well. However, with the way that the graph is currently drawn, it would be difficult to provide this functionality without revealing the imprecise nature of the curves, as the trapeziums that composed the curves would be revealed. This would also compromise and diminish the usefulness of this functionality anyway, as it would become more difficult to determine the precise nature of the curves by zooming in anyways. Ways to fix this without altering the entire drawing process would be very inefficient and lag-prone, and so this feature may be very difficult to implement without a total rewrite of the graph creation algorithm. As such, this feature would likely not be added without extensive revisions to the code.

However, one other essential feature that would need to be added could be implemented without too much hassle. Adding scales to the graphs with the current design would be much easier, as it would only require drawing a label the intercepts where and, already determined by the axis. Alternatively, I could also draw labels intermittently along the axes to act as the scale intervals, which would be a much more simple iterative process than solving for y. This would likely be implemented much sooner than the magnification functionality.

Changes during development

The addition of an All Clear (‘AC’) button to the calculator setup during development was also a useful change, as it allowed users to more easily clear expressions if there was a difficult mistake to fix, where before the use of the backspace button would have been too cumbersome, especially where it had to be used to delete large expressions, which would take a while due to the inability to hold down the button to more quickly delete the expression.

In addition, as per request I made an ‘’ button that would calculate the factorial of inputs given. This was useful, as it would enable the user to perform binomial expansions much more easily, as could be seen here:



This would make it much more simple to estimate the indices of more unusual numbers, such as , where this would be translated as , where . This process was prevalent in the A-Level Core Mathematics 4 syllabus, which would make the factorial function very useful for teachers and sixth-form students, my end users.