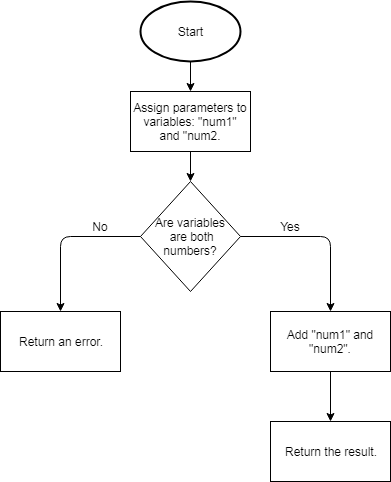
## 2.2 Algorithms

### 2.2.1 Basic Operations, e.g. Addition

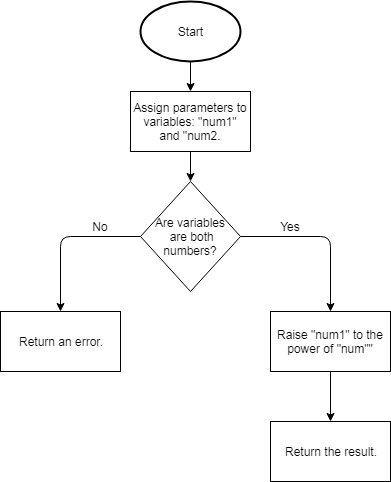


This flowchart shows that two variables are created and added together, then the method is returned.

A similar algorithm will be used for the other basic mathematical operations such as subtraction, division, and multiplication. This is done by changing the operation that is done between the two numbers.

This method is required because basic mathematical operations are the minimum requirement for a calculator to be of any use. Even if the calculator had more advanced features such as converting numbers to different number bases, the calculator would still require basic functionality for it to be helpful to any user.

### 2.2.2 Powers

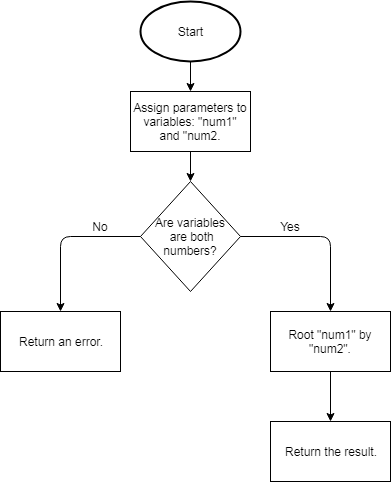


This flowchart shows how the initial parameters that are passed to the method are assigned to the variables, “base” and “power”. After validation checks are performed on the two input numbers, the calculator will raise the base to the order of the power.

This method is required for the calculator app because the ability for the user to user more advanced maths (compared to basic functionality) is very important to for students in their GCSEs or A Levels.

Powers are used in almost every aspect of a STEM subject in GCSE and therefore this functionality is required for the calculator to be of use in more situations.

### 2.2.3 Roots

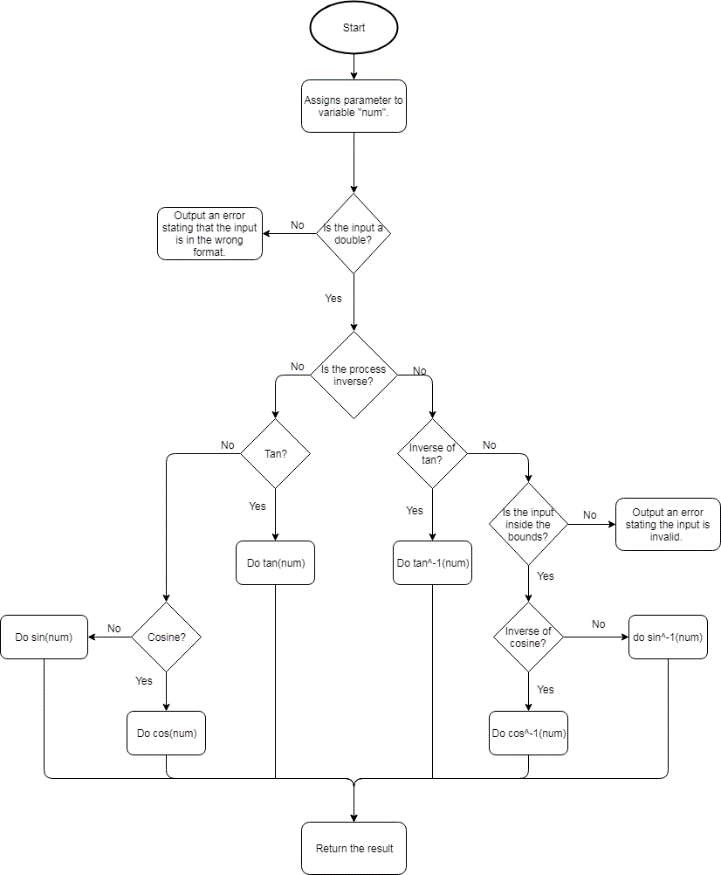


This method takes the parameter and assigns them to the variables ‘base’ and ‘root’. The two variables under go a validation check to ensure that using them won’t result in an error that could lead to a crash.

After the validation is confirmed, the the “base “variable will be rooted by the “root” variable.

This method is required so the calculator can be used and be more helpful in more situations. This is because the functionality to root numbers is often needed during school lessons and for schoolwork. Therefore, without this feature, the calculator would be useless in many situations.

### 2.2.4 Trigonometry



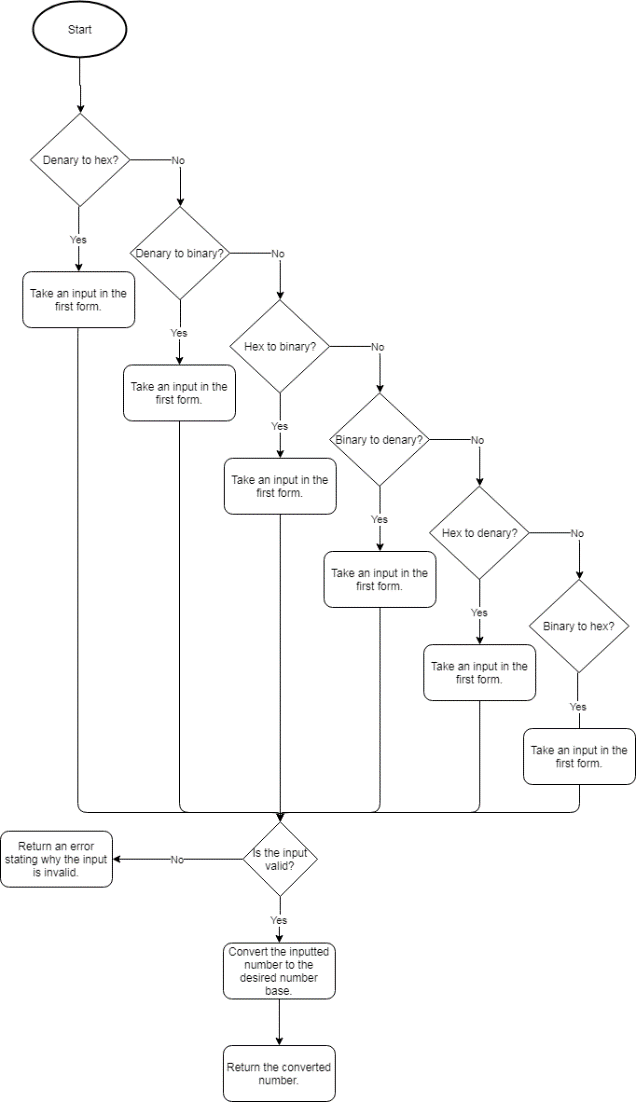
This flowchart shows how the function for calculating trigonometric expressions will be carried out.

This includes two forms of validation. The intial validation ensures that the input is of the correct data type so the function does not attempt to perform a calculation on a string.

The second form of validation is for the sine and cosine calculations only. This validation ensures that the input is between 0 and 1 for inverse calculations of sine and cosine. The validation is only required for the inverse versions of sine and cosine as they must be within the maximum and minimum bounds of their waves whereas the tan wave increases to infinity.

This method is required because trigonometric functions are often used in GCSE and A Level subjects. For example, trigonometry is a large of the Maths GCSE specification and every student will need to learn how to use it. Having this feature in the calculator will help the user understand trigonometry and so help them in their exams.

### 2.2.5 Number Base Conversions



This flowchart shows how the process of the user selecting a type of conversion, then entering a number which then gets converted.

This algorithm enables the user to perform conversions to different bases such as binary to hex.

There is one instance of validation in this algorithm. After the user inputs their number, it will be run through the validity check. This checks that the input is valid for the different types of conversion, for example, binary to denary will only accept the numbers ‘1’ and ‘0’.

This algorithm is especially required for Computer Science GCSE and A Level students. This is because the Computer Science specification requires students to convert denary to hex, binary to denary etc. The specification also includes basic binary operations.

Therefore this feature will be very helpful to those students as it will allow them to check their working so they can see their mistakes and improve upon them.

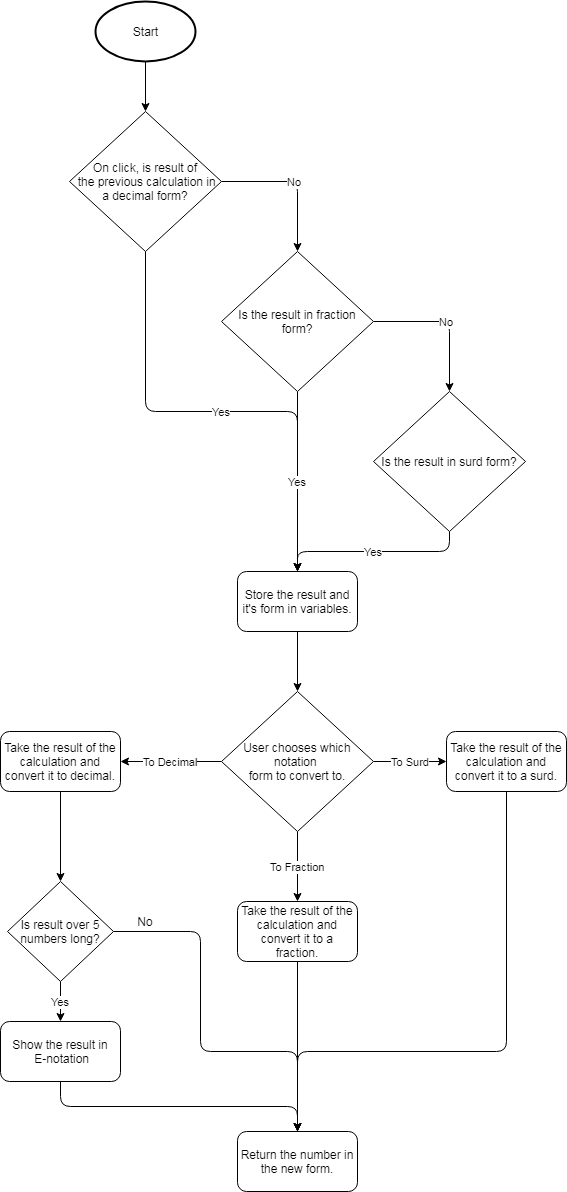
### 2.2.6 Number Base Conversions Validity CheckC:\Users\2dennyd\Downloads\Number Conversion Validity Check.png

This flowchart shows how the input for the base number conversion will be validated. The algorithm will employ different validation methods depending on the resulting number base that the user wants.

For example, if user selects binary to denary conversion, then the input number must only contain the digits “1” and “0”. This is because binary is a base 2 number system.

This function works in conjunction with the number base conversions. Therefore, this algorithm is required so that the number base conversions can be used without any errors occurring. This potentially prevents errors that could crash the app or otherwise influence the user’s experience.

### 2.2.7 Notation Conversion



This flowchart shows the process of converting values between different types of notation.

For example, if the user chooses to convert a fraction to a decimal, the first decision (is the input decimal) will be a no so the flowchart carries on to the next one. This one is positive because the decision is whether the input is fractional. The input and which notational form it’s in is stored in variables. Then the user chooses what notation to convert it to.

In this example, the user chooses decimal notation. The input is converted to decimal and another decision occurs, if the decimal result is over 9 digits long, it is converted and returned in scientific notation (used when the decimal number is too big to be conveniently written).

This method is required because throughout school, the students are required to be able to convert fractions to decimals and vice versa and to be able to use surds, etc. Therefore, having a method which will display the other forms of their calculations will help the students become familiar with all the different forms of notation.

### 2.2.8 Unit ConversionC:\Users\2dennyd\Downloads\Convert Units.png

This flowchart shows how the method for converting units will be carried out.

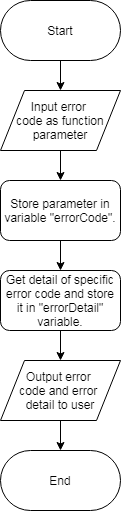
For example, the user will choose their conversion such as millilitres to litres. The method will check if there is currently a number stored in the answer variable. If so, the method will display the conversion using the answer variable as the input number.

After displaying it (or straight away if the variable is empty), the method will then ask the user for input.

If this is denied, the method will exit and the user will be returned to the main calculator screen.

This method is required because in STEM subjects, especially ones like Physics, equations are often used with different units. The specification requires the student to understand how to convert units and so this calculator will help them understand the process and let them focus on harder parts of the problem.

### 2.2.9 Display Error



This flowchart shows the process of an error occurring and it being represented to the user.

For example, this algorithm will be a method that is called when an error occurs. The method call will have a parameter which will be an integer error code.

Each unique error code will have a corresponding string that can be displayed to the user. This will explain the reason of the error and how the user can fix it.

This algorithm is required because having a clear and understandable error messages will be very important when it comes to the apps usability and the user’s experience.

For example, the user makes a mistake when they write out their expression and it doesn’t work. If the app doesn’t have any intuitive error messages the user will become frustrated and annoyed. This ruins the user’s experience and makes it less likely that the user will continue to use my app.

Therefore, having clear, concise and helpful error messages is an important requirement of a successful solution because it greatly improves the UX and makes it more likely that the app will be used.