


# Guide to Calculations and Filters

for Leapfrog Geo Version 2021.2

© 2022 Bentley Systems, Incorporated (“**Seequent**”). All rights reserved. Unauthorised use, reproduction, or disclosure is prohibited. Seequent assumes no responsibility for errors or omissions in this document. LEAPFROG, SEEQUENT and  are trade marks owned by Seequent. All other product and company names are trade marks or registered trade marks of their respective holders. Use of these trade marks in this document does not imply any ownership of these trade marks or any affiliation or endorsement by the holders of these trade marks.

## Catalogue of Metadata, Syntax and Functions

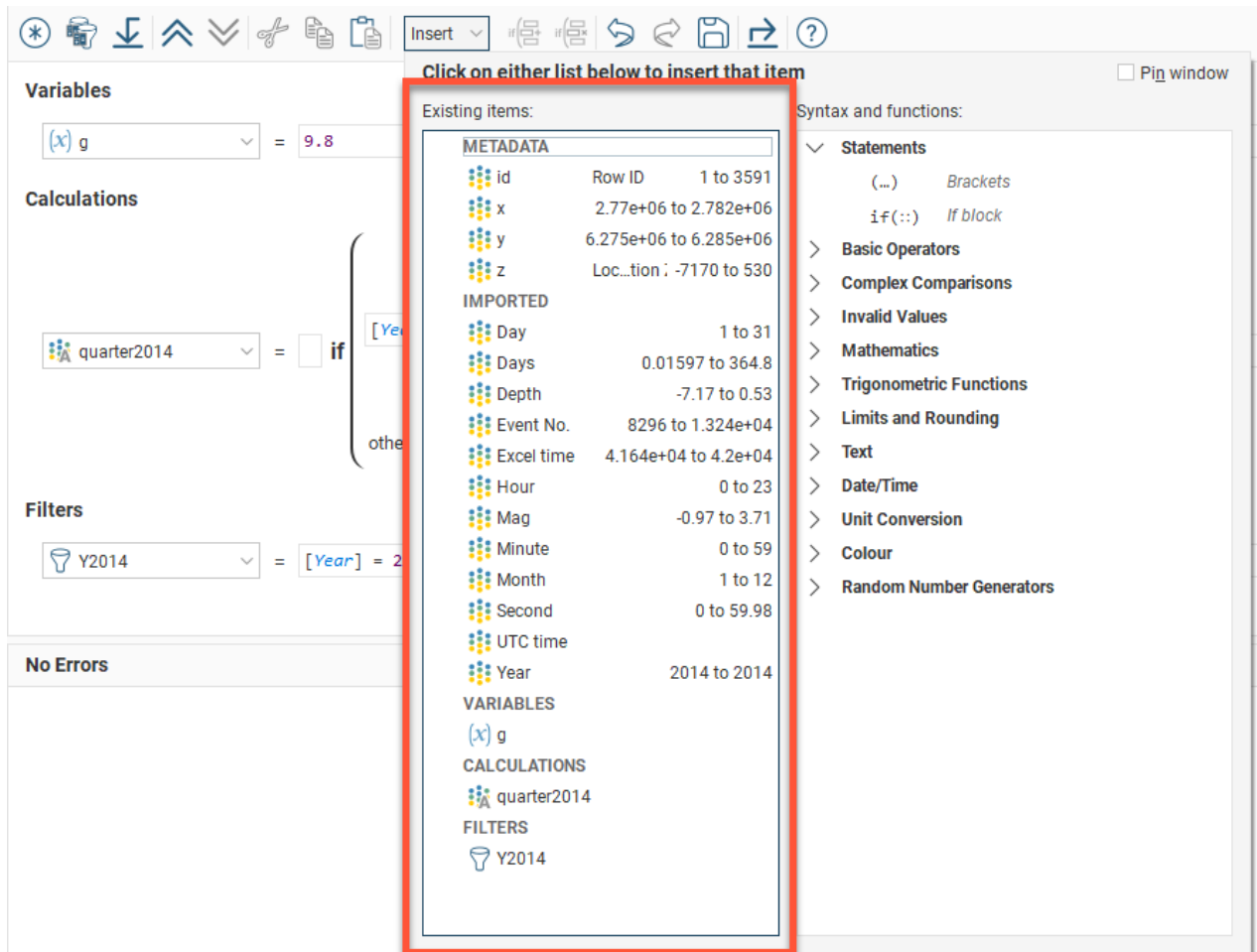
This catalogue details each of the items in the pinnable Insert list for Calculations and Filters. Each item includes an intentionally trivial example to illustrate the use of the item, along with an explanation of the effect of the expression.

The catalogue is organised as follows:

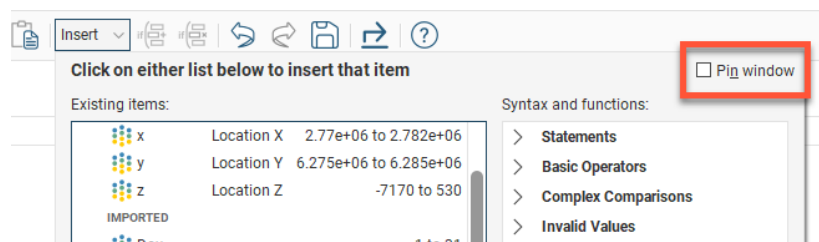
- Existing Items
  - Metadata
  - Evaluations
  - Variables, Calculations and Filters
- Syntax and Functions
  - Statements
  - Basic Operators
  - Complex Comparisons
  - Invalid Values
  - Mathematics
  - Trigonometric Functions
  - Limits and Rounding
  - Text
  - Date/Time
  - Unit Conversion
  - Colour
  - Random Number Generators

## Existing Items

This section covers the items listed in the left-hand side of the pinnable Insert list:



To pin these lists to the Calculations tab, enable the Pin Window option:



## Metadata

### id

This metadata item is available for imported points objects. It is the row ID from the points table.

*Example*



$$\text{even rows} = [id] \% 2 = 0$$

*Explanation*

The filter even rows will select for points with a row [id] that has no remainder when divided by 2.

**x, y and z**

These three metadata items are the variables for locating each point in a points object in X, Y and Z coordinates. Select an item to add it to the expression at the insertion point. Note that whenever one of these metadata items is added to an expression, it is wrapped in square brackets. This is not available for block objects; use xc, yc and zc instead.

*Example*



$$YX = [y] * [x]$$

*Explanation*

The numeric calculation YX will be assigned the value of the location of [y] multiplied by the location of [x].

**xc, yc and zc**

These three metadata items are the variables for locating the centroid of each block, in X, Y and Z coordinates. Select an item to add it to the expression at the insertion point. Note that whenever one of these metadata items is added to an expression, it is wrapped in square brackets. This is not available for points objects; use x, y and z instead.

*Example*



$$\text{top} = [zc] + ([dz] / 2)$$

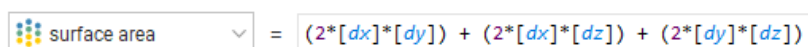
*Explanation*

The numeric calculation top will be assigned the value of the location of [zc] the altitude of each block centroid from the zero reference, plus half the height of the block.

**dx, dy and dz**

These three metadata items are the variables for the block dimensions in X, Y and Z coordinates. Select an item to add it to the expression at the insertion point. Note that whenever one of these metadata items is added to an expression, it is wrapped in square brackets. This is not available for points objects.

*Example*



$$\text{surface area} = (2*[dx]*[dy]) + (2*[dx]*[dz]) + (2*[dy]*[dz])$$

*Explanation*

The numeric calculation surface area will be calculated by figuring the area of each face of the block by multiplying the X and Y dimensions, X and Z dimensions and Y and Z dimensions and adding them together.

**volume**

This metadata item provides the volume for each block. Note that when this metadata item is added to an expression, it is wrapped in square brackets. This is not available for points objects.

*Example*

*Explanation*

The numeric calculation density will be assigned the value of the variable [mass] divided by the metadata item [volume].

**xi, yi and zi**

These three metadata items are the variables for locating each block by X, Y and Z index. Select an item to add it to the expression at the insertion point. Note that whenever one of these metadata items is added to an expression, it is wrapped in square brackets. This is not available for points objects.

*Example*

*Explanation*

The numeric calculation remaining will be assigned the value of the block's AU\_gpt evaluation, unless [zi] the Z index of the block is greater than or equal to 40, in which case the block status will be set to the invalid value outside.

**Evaluations**

Each of these items are automatically added to the list whenever an evaluation is added to the object. When added to an expression, the item represents a placeholder in the calculation for an estimated value, as the expression is evaluated for each of the locations in turn.

Note you can expand each evaluation in the list to see the attributes for the estimation evaluation that may also be selected instead of or in addition to the estimated value.

*Example*

*Explanation*

The numeric calculation halved is defined by the evaluation [AU\_gpt] divided by 2. Each location in the object will have its own value for AU\_gpt, and this calculation uses those values to create a new value named halved for each location, using the formula above.

**Variables, Calculations and Filters**

Each time you create a new variable, numeric calculation, category calculation, or filter in Calculations, it will also be added to the Existing items list. You can select them from this list and they will be inserted into your new expression at the insertion point. Note that whenever one of these named items is added to an expression, it is wrapped in square brackets.

**VARIABLES**

- (x) density

**CALCULATIONS**

- halved 0 to 14.2
- remaining 0 to 9.495
- highlight

**FILTERS**

- Less Than 1

*Example*

**Variables**

(x) AU threshold = 1 = 1

(x) CU threshold = 0.5 = 0.5

**Calculations**

highlight

if

- [Geological model] = 'Early Diorite' → if [AU over threshold] → "AU" otherwise blank
- [Geological model] = 'Dacite' → if [CU over threshold] → "CU" otherwise blank
- otherwise blank

**Filters**

AU over threshold = [AU\_gpt] > [AU threshold]

CU over threshold = [CU\_pct] > [CU threshold]

*Explanation*

This example is only attempting to illustrate how variables, calculations and filters that have already been defined can be referenced by name in new calculations; the calculation highlight is not intended to be useful.

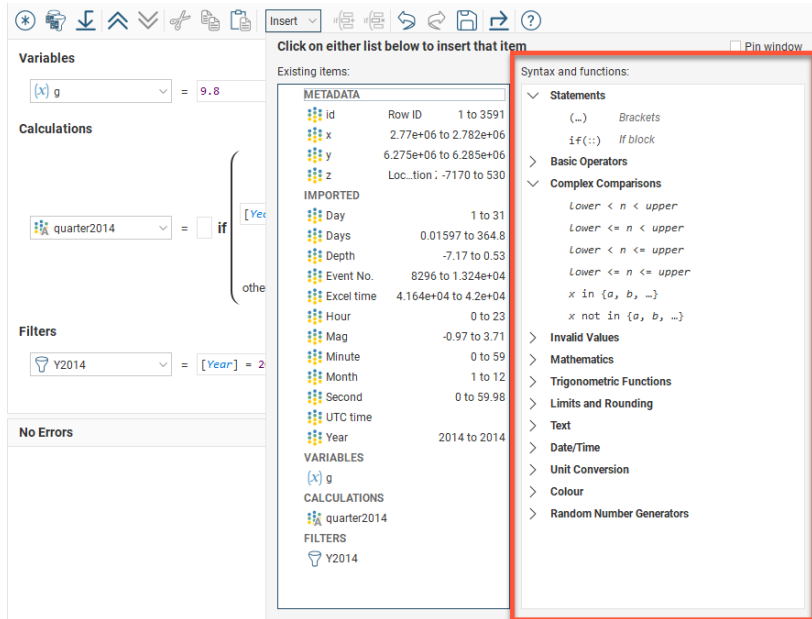
It is good practice to break your calculations down into parts, giving each part a relevant and readily identifiable, unambiguous and easily understood name. This will make your calculations more

readable and clear. Using Variables to define a constant with a name makes it easy to understand the utility of that particular constant when you use it in a calculation. You may also be able to re-use certain parts such as filters or constant variables, so you do not need to define the same thing repeatedly.

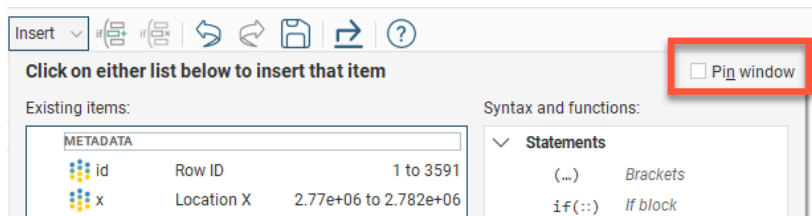
Be careful not to inadvertently name something incorrectly, such as naming a volume as "area", as this could give rise to difficult-to-locate errors in your calculation.

## Syntax and Functions

This section covers the items listed in the right-hand side of the pinnable Insert list.



To pin these lists to the Calculations tab, enable the Pin Window option:



## Statements

### (...) Brackets

Brackets are used to enclose an expression so the operations on the values within the brackets take precedence over operations outside the brackets.

Example

(x) BODMAS =  $6 / 2 * (1 + 2)$  = 9



*Explanation*

1+2 will be calculated prior to calculating the result of the expression, following the standard order of mathematical operations.

**if(::<) If block**

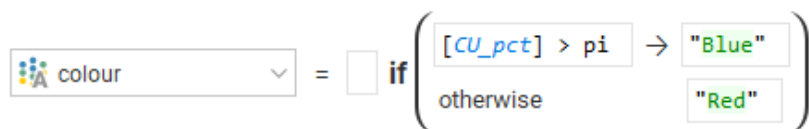
The If block is used for conditional logic. This allows multiple pathways to results depending on selected conditions, or categorisation based on values.

An If block will be evaluated by each test → result, row by row, separately and in order downwards by row. Each test has an output that can be 'true', 'false', or an invalid value 'error', 'blank', 'without\_value' or 'outside'. The result output is produced by the execution of the result expression. The If block output follows these rules:

- If a test output is 'error', the If block output is 'error' and no further processing of subsequent rows is done.
- If a test output is 'false', the result expression is not executed, and the next row is considered.
- If a test is 'true', the result expression is executed and the result output is used for the If block output and no further processing of subsequent rows is done.
- If all the tests are 'false', the 'otherwise' result expression is executed and its output is used.
- If a test output is an invalid value (without\_value, blank, outside) the result expression is not executed and the test output's invalid value is remembered. The subsequent rows are then run.
- If a subsequent test output is 'true' after an earlier one produced an invalid value, the previous test output is discarded and the new row's result expression output is used as the output for the If block.
- If all test outputs are invalid status values, the highest priority status of all the remembered invalid statuses is used as the output result. The priority of non-error invalid status values is: outside > without\_value > blank.

Additionally, it is possible to choose to produce invalid values as the output of result expressions.

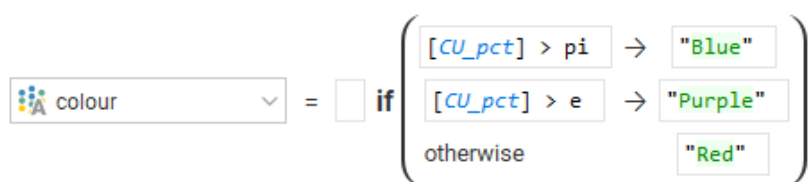
*Example*



*Explanation*

Cu\_pct is the name of an evaluation applied to all the points in a points object. As the if(::<) If block calculation is run for each point, the evaluation for each point replaces this variable name in the expression. If the value is greater than the value of the constant pi, the result for that point will be the text string "Blue". Otherwise, if the value is less than or equal to pi, the result will be the string "Red".

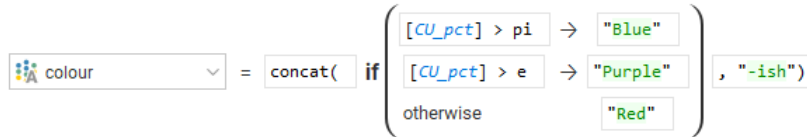
Additional rows may be added. Each row follows on from the left-over results of the line before, simplifying the logical expression that may be used.



*Explanation*

Note the addition of the line `[Cu_pct] > e → "Purple"`. This line can be interpreted to mean: if the value estimated for a point is less than or equal to pi, but greater than e the result shall be "Purple". The part about it being less than or equal to pi is implied because the line follows the previous line `[Cu_pct] > pi → "Blue"`.

Note that expression elements before and after the if expression can be entered. This allows the if(::<) If block to form part of a more complex or extensive expression.



*Explanation*

The earlier conditional classification has now been embedded within a concatenation function, forming the first part of a two-part string concatenation. The concatenation function is adding the text string "-ish" to whatever is produced by the if(::<) If block. Thus, if the value of the block being evaluated is 1.2, the result of the colour categorisation calculation will be "Red-ish".

**Basic Operators**

**+ Add**

An arithmetic addition operation.

*Example*



*Explanation*

next is assigned the value of the current imported points row [id] plus 1.

**- Subtract**

An arithmetic subtraction operation.

*Example*



*Explanation*

prev is assigned the value of the current imported points row [id] minus 1.

## \* Multiply

An arithmetic multiplication operation. Note that implied multiplication, putting factors adjacent to one another, is not supported. The \* operator must be explicitly used.

*Example*

$$\text{[x] product} = \text{[x]} * \text{[scale]}$$

*Explanation*

product is assigned the value of [x] times [scale].

## / Divide

An arithmetic division operation.

*Example*

$$\text{[x] quotient} = \text{[numerator]} / \text{[denominator]}$$

*Explanation*

quotient is assigned the value of [numerator] divided by [denominator].

## % Modulo

An arithmetic modulo operation. This is an integer division operation that returns the remainder instead of the integer quotient.

*Example*

$$\text{[x] remainder} = \text{[numerator]} \% \text{[denominator]}$$

*Explanation*

remainder is assigned the value of [numerator] modulo [denominator], or in other words, [numerator] is divided by [denominator] to produce an integer quotient, the number of times [denominator] goes into [numerator], and a remainder, which is the number returned by this modulo function.

## ^ Power

A mathematical exponentiation operation where a base is raised to the power of the exponent.

*Example*

$$\text{[x] area} = \text{pi} * \text{[radius]}^2 = 12.5663706143...$$

*Explanation*


area is assigned the value of  $\text{pi} * \text{[radius]}^2$  (because ^2 is interpreted as 'to the power of the exponent 2' or 'squared'). Because [radius] happened to be defined as equalling 2, the result of the expression

$\pi * 2^2$  is 12.56637..., as can be seen from the result at the end of the expression.

## and Logical and

A logical and operation.

### Example

 topD = `(([Geological model] = 'Dacite') and ([z] > 2800))`

### Explanation

filter will be true if the point is classified in the 'Dacite' part of the geological model AND the [z] coordinate for the point is above 2800; it will be false if either of these conditions are not true.

## or Logical or

A logical or operation.

### Example

 Dacite and ED = `(([Geological model] = 'Dacite') or ([Geological model] = 'Early Diorite'))`


### Explanation

Dacite and ED will be true if the point is classified in the 'Dacite' part of the geological model OR the point is classified in the 'Early Diorite' part of the geological model, but it will be false if neither of these conditions is true.

## not Logical not

A logical not operation.

### Example

 Not Dacite = `not ([Geological model] = 'Dacite')`

### Explanation

Not Dacite will be true if the point is classified in the geological model as anything other than 'Dacite'. The logical operator not inverts the logical expression that follows the operator.

## = Equal

A logical equality operator.

### Example

 Dacite = `([Geological model] = 'Dacite')`

### Explanation

Dacite will be true if the point is classified in the 'Dacite' part of the geological model, and will be false for all other values.

**!= Not equal**

A logical not-equal operator.

*Example*

 Not Dacite = `([Geological model] != 'Dacite')`

*Explanation*

Not Dacite will be true if the point is classified in the geological model as anything other than 'Dacite', and will be false when it is 'Dacite'.

**< Less than**

A logical less-than operator.

*Example*

 density under 1 = `[density] < 1`


*Explanation*

density under 1 will be true when the variable [density] is less than 1, and false otherwise.

**< Less or equal**

A logical less-than-or-equals operator.

*Example*

 density le 1 = `[density] <= 1`

*Explanation*

density le 1 will be true when the variable [density] is less than or equal to 1, and false otherwise.

**< Greater than**

A logical greater-than operator.

*Example*

 density over 1 = `[density] > 1`

*Explanation*

density over 1 will be true when the variable [density] is more than 1, and false otherwise.

**< Greater or equal**

A logical greater-than-or-equals operator.

*Example*

=

*Explanation*

density ge 1 will be true when the variable [density] is more than or equal to 1, and false otherwise.

## Complex Comparisons

**Lower < n < upper**

A pair of comparisons, with a logical result to indicate if the tested value n is between the lower value provided and the upper value provided.

*Example*

=

*Explanation*

filter will be true when the point evaluation [AU\_gpt] is between the values of 5 and 8 (but not equalling 5 or 8); it will be false otherwise.

**Lower <= n < upper**

A pair of comparisons, with a logical result to indicate if the tested value n is between the lower value provided and the upper value provided, or equal to the lower value.

*Example*

=

*Explanation*

filter will be true when the point evaluation [AU\_gpt] is between the values of 5 and 8 (but not equalling 8); it will be false otherwise.

**Lower < n <= upper**

A pair of comparisons, with a logical result to indicate if the tested value n is between the lower value provided and the upper value provided, or equal to the upper value.

*Example*

=

*Explanation*

filter will be true when the point evaluation [AU\_gpt] is between the values of 5 and 8 (but not equalling 5); it will be false otherwise.

## Lower $\leq n \leq$ upper

A pair of comparisons, with a logical result to indicate if the tested value  $n$  is between (or equal to either) the lower value provided and the upper value provided.

*Example*

=

*Explanation*

filter will be true when the point evaluation [AU\_gpt] is greater than or equal to 5 and less than or equal to 8; it will be false otherwise.

## $x$ in {a,b,...}

A logical inclusion expression. This will return true if  $x$  matches any element of the set of listed items.

*Example*

= 

*Explanation*

If [colour] is one of the items in the list within the curly brackets, filter will be true. If [colour] is anything else, filter will be false.

## $x$ not in {a,b,...}

A logical exclusion expression. This will return true if  $x$  fails to match any element of the set of listed items.

*Example*

= 

*Explanation*

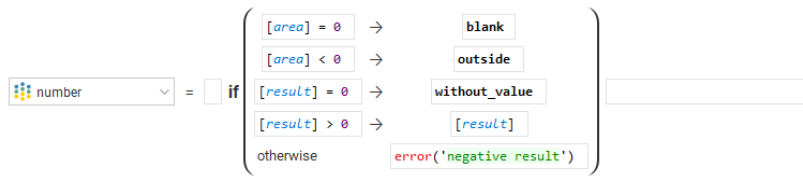
If [colour] is not one of the items in the list within the curly brackets, filter will be true. If [colour] matches any item in the list, filter will be false.

## Invalid Values

Invalid values are different types of results for “numeric” categorisation calculations that need special non-numeric results for certain category results. These have special meanings of their own without having to resort to interpreting negative numbers and zero as having special meaning.

blank means having no value, the value in the imported file is blank or has non-numeric data  
 without\_value is often used to mean the estimator cannot produce a value (specific to blocks)  
 outside is used to indicate the block is outside the boundary of the domain (specific to blocks)  
 error generates an error, and provides an ‘error’ status value for the affected block or point  
 error(‘message’) is similar to error but includes a custom message.

*Multiple case example*



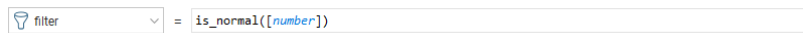
*Explanation*

If the variable `[area]` is equal to 0, the point will be marked with the special value `blank`. If the variable `[area]` is less than 0, the point will be marked with the special value `outside`. If `[area]` is greater than 0 and the variable `[result]` is equal to 0, the point will be marked with the special value `without_value`. If `[area]` is greater than 0 and `[result]` is greater than 0, the point will be assigned the value of the variable `[result]`. If `[area]` is greater than 0 and `[result]` is less than 0, the point will be given the special value `error` and status with the message 'negative result'.

**is\_normal(a)**

A function that tests `a` to see if it is a normal value or an invalid value. If `a` is normal, it returns `true`. If `a` is invalid, it returns `false`.

*Example*



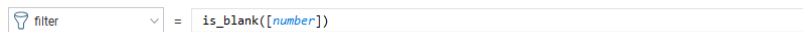
*Explanation*

If `[number]` has a normal value, `filter` will be true for that block. If it produces an invalid value, `filter` will be false.

**is\_blank(a)**

A function that tests `a` to see if it is a blank invalid value. If `a` is blank, it returns `true`. If `a` is normal or another invalid value, it returns `false`.

*Example*



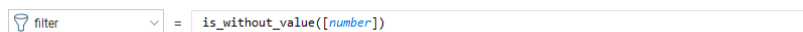
*Explanation*

If `[number]` has a blank status, `filter` will be true for that block. If it produces any other invalid value or a normal value, `filter` will be false.

**is\_without\_value(a)**

A function that tests `a` to see if it is a `without_value` invalid value. If `a` is `without_value`, it returns `true`. If `a` is normal or another invalid value, it returns `false`.

*Example*





### Explanation

If [number] has a without\_value status, filter will be true for that block. If it produces any other invalid value or a normal value, filter will be false.

### is\_outside(a)

A function that tests a to see if it is an outside invalid value. If a is outside, it returns true. If a is normal or another invalid value, it returns false.

### Example

=

### Explanation

If [number] has an outside status, filter will be true for that block. If it produces any other invalid value or a normal value, filter will be false.

## Mathematics

### pi Constant

The constant pi is an existing item you do not need to define yourself. It is defined to 15 decimal places as 3.141592653589793.

### Example

=  = 12.5663706143...

### Explanation

area will be calculated as pi multiplied by the square of [radius]. Note that pi is not enclosed in square brackets like user-created variables and constants, as it is an internal constant.

### e Constant

The constant e, the base of the natural logarithm, is an existing item you do not need to define yourself. It is defined to 15 decimal places as 2.718281828459045.

### Example

=

### Explanation

comp will be calculated as e to the power of ([rate] minus 1). Note that e is not enclosed in square brackets like user-created variables and constants, as it is an internal constant.

### log(n) Base 10

The common logarithm, the logarithm with base 10, i.e.  $\log_{10}(n)$  or  $\lg(n)$ . This function will calculate the common logarithm of the value provided as n.

*Example*

 scaled = log([measure])


*Explanation*

scaled will be calculated as  $\log_{10}$  of [measure].

**log(n, base)**

The logarithm of a number n to the base base.

*Example*

 scaled = log([measure], 2)


*Explanation*

scaled will be calculated as  $\log_2$  of [measure].

**ln(n) Natural log, base e**

The natural logarithm, the logarithm with base e, i.e.  $\log_e(n)$ . This function will calculate the natural logarithm of the value provided as n.

*Example*

 scaled = ln([measure])


*Explanation*

scaled will be calculated as  $\log_e$  of [measure].

**exp(n) Natural exponent**

The natural exponent. This function will provide the result of  $e^n$ .

*Example*

 em = exp([measure])

*Explanation*

em will be calculated as  $e$  [measure].

**sqrt(n) Square root**

The principle square root of the provided number n.

*Example*

 sqrtm = sqrt([measure])

*Explanation*

sqrtm will be calculated as the square root of [measure].

### abs(n) Absolute value

The absolute value of a number is its value with the sign of the number disregarded. The absolute value of -42 is 42. The absolute value of 42 is also 42.

*Example*

 absm = abs([measure])

*Explanation*


absm will be whatever [measure] is, but without its sign; it will always be positive as a result.

## Trigonometric Functions

### sin(x) Sine

The trigonometric sine function, expecting the parameter provided to be in degrees (not radians).

*Example*

 sinx = sin(30)

*Explanation*

sinx will be calculated as the sine of 30 degrees, which is 0.5.

### cos(x) Cosine

The trigonometric cosine function, expecting the parameter provided to be in degrees (not radians).

*Example*

 cosx = cos(60)


*Explanation*

cosx will be calculated as the cosine of 60 degrees, which is 0.5.

### tan(x) Tangent

The trigonometric tangent function, expecting the parameter provided to be in degrees (not radians).

*Example*

 tanx = tan(45)


*Explanation*

tanx will be calculated as the tangent of 45 degrees, which is 1.

**asin(x) Arcsine**

The trigonometric arcsine (or inverse sine) function, returning a result in degrees (not radians).

*Example*

 asinx = asin(0.5)


*Explanation*

asinx will be calculated as the arcsine of 0.5, which is 30 degrees.

**acos(x) Arccosine**

The trigonometric arccosine (or inverse cosine) function, returning a result in degrees (not radians).

*Example*

 acosx = acos(0.5)


*Explanation*

acosx will be calculated as the arccosine of 0.5, which is 60 degrees.

**atan(x) Arctangent**

The trigonometric arctangent (or inverse tangent) function, returning a result in degrees (not radians).

*Example*

 atanx = atan(1)

*Explanation*


atanx will be calculated as the arctangent of 1, which is 45 degrees.

## Limits and Rounding

**min (n, m, ...)**

Returns the lowest of all the values in the set provided.

*Example*

 low = min([x], [y], [z])

*Explanation*

low will be the lowest of the three values provided, the metadata items for the X, Y, and Z point coordinates.

**max (n, m, ...)**

Returns the highest of all the values in the set provided.

*Example*

=

*Explanation*

high will be the highest of the three values provided, the metadata items for the X, Y, and Z point coordinates.

**clamp(n, lower)**

This clamp function tests the value n against the threshold lower and if it is less than lower the result will be lower; otherwise the result will be n. The effect is to push all the values below the threshold up to the threshold.

*Example*

=

*Explanation*

The output for modified will range from 0.25 up to the maximum value of [AU\_gpt]. If [AU\_gpt] is less than 0.25, the output will be 0.25 instead. Otherwise, the output will be [AU\_gpt].

**clamp(n, lower, upper)**

This clamp function tests the value n against the threshold lower and if it is less than lower the result will be lower; it tests the value n against the threshold upper and if it is more than upper the result will be upper; otherwise the result will be n. The effect is to squish all the values into a box between the lower and upper thresholds.

*Example*

=

*Explanation*

The output for modified will range from 0.25 up to 8. If [AU\_gpt] is less than 0.25, the output will be 0.25 instead. If [AU\_gpt] is more than 8, the output will be 8 instead. Otherwise, the output will be [AU\_gpt].

**round(n)**

This function rounds the input value n to the nearest whole number.

*Example*

=  = -13  
 =  = 13

*Explanation*

The variable negative will be given the value -13 as the nearest whole number to -12.6789 used as the input to the function. The variable positive will be given the value 13 as the nearest whole number to 12.6789 used as the input to the function.

**round(n, dp)**

This function rounds the input value n to the number of decimal places specified by dp, a positive integer.

*Example*

=  = -12.3457

=  = 12.3457

*Explanation*

The variable negative will be given the value -12.3457, the value of -12.3456789 rounded to 4 decimal places. The variable positive will be given the value 12.3457, the value of 12.3456789 rounded to 4 decimal places.

**roundsf(n, sf)**

This function rounds the input value n to the number of significant figures specified by sf, which must be a positive integer  $\geq 1$ . Rounding to a given number of significant figures is often preferred in scientific applications over rounding to a given number of decimal places, as outputs can be rounded to the same amount of significance as the inputs.

*Example*

=  = -12.35

=  = 12.35

*Explanation*

The variable negative will be given the value -12.35, the value of -12.3456789 rounded to 4 significant figures. The variable positive will be given the value 12.35, the value of 12.3456789 rounded to 4 significant figures.

**floor(n)**

This function removes the fractional part of a real number n and returns the integer number below the real number n. This remains true when n is a negative number.

*Example*

=  = -13

=  = 12

*Explanation*

The variable negative will be given the value -13, the integer below -12.3456789. The variable positive will be given the value 12, the integer below 12.3456789.

## ceiling(n)

This function removes the fractional part of a real number  $n$  and returns the integer number above the real number  $n$ . This remains true when  $n$  is a negative number.

### Example

(x) negative = ceiling(-12.3456789) = -12  
 (x) positive = ceiling(12.3456789) = 13

### Explanation

The variable `negative` will be given the value -12, the integer above -12.3456789. The variable `positive` will be given the value 13, the integer above 12.3456789.

## truncate(n)

This function simply removes the fractional part of a real number  $n$  and returns the integer number without the fractional part. This means that for positive real numbers, the result will be the integer less than the real number  $n$ , but for negative real numbers, the result will be the integer greater than the real number  $n$ .

### Example

(x) negative = truncate(-12.3456789) = -12  
 (x) positive = truncate(12.3456789) = 12

### Explanation

The variable `negative` will be given the value -12, the integer part of -12.3456789. The variable `positive` will be given the value 12, the integer part of 12.3456789.

## Text

### 'abc' Text value

Use this item to add a text sequence to an expression. Selecting the item will add two single quotation marks with the cursor between, ready for the text sequence to be typed. You can of course simply type the quote marks into the expression yourself. Double quotation marks also work identically to the single quotation marks used by this item. If you need to include a quote mark inside your text sequence, you need to "escape" the character so it is not interpreted as the end of the text sequence, by entering two quotation marks for each quotation mark you want inside the text sequence. Alternatively, you can use a different type of quotation mark as the sequence wrappers; for instance to write Seequent's Region with an internal apostrophe, you might wrap the sequence with double quotation marks: "Seequent's Region".

### Example

(x) name1 = "Seequent's Region"  
 (x) name2 = 'Seequent''s Region'

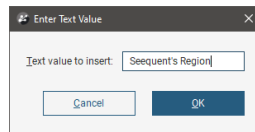
*Explanation*

The text sequence Seequent's Region (note: without the wrapping quotation marks and with only one possessive apostrophe) will be used wherever the variables name1 or name2 are used in expressions. Both techniques for including apostrophes in the text sequence have the same result.

**Enter text...**

This selection opens a dialog box prompting you for text. After you enter it and click OK, the text will be entered at the cursor, wrapped with quotation marks. This is an easy way to resolve any issues about internal quotation marks, as the dialog box will convert the text into the necessary character sequence required to generate your desired text.

*Example*



*Explanation*

This will produce a converted character sequence that produces a valid string and insert it to the expression at the insertion point.

```
(x) name2 = "Seequent's Region"
```

**concat(t, u, ...)**

This concatenates a series of text sequences together.

*Example*

```
(x) name = concat("Seequent_", [direction], "_Region")
```

*Explanation*

Each of the text sequences in the input are run together and combined. If [direction] is 'North' then name will be Seequent\_North\_Region. If [direction] is 'South' then name will be Seequent\_South\_Region.

**startswith(t, 'prefix')**

This function returns true if the text sequence t starts with prefix, and false otherwise. This is case insensitive; 'prefix' will match 'PREFIX'.

*Example*

```
filter = startswith([name], 'Seequent')
```

*Explanation*

filter will be true if [name] starts with Seequent.



**endswith(t, 'suffix')**

This function returns true if the text sequence t ends with suffix, and false otherwise. This is case insensitive; 'suffix' will match 'SUFFIX'.

*Example*

 filter = endswith([name], 'Region')

*Explanation*

filter will be true if [name] ends with Region.

**contains(t, 'part')**

This function returns true if the text sequence t contains part somewhere within, and false otherwise. This is case insensitive; 'part' will match 'PART'.

*Example*

 filter = contains([name], 'North')


*Explanation*

filter will be true if [name] contains North somewhere within the character sequence.

**like(t, 'pattern')**

This function returns true if the text sequence t matches pattern, where [pattern] is follows SQL-style LIKE matching rules. This is case insensitive, and pattern must match the whole of t, not just a portion of it. Use \_ as a wildcard for a single character, and % as a wildcard for any number of characters (including no characters).

*Example*

 filter = like([name], '%Seequent\_Region%')

*Explanation*

filter will be true if [name] matches the pattern %Seequent\_Region%. Examples of [name] that will match include:

- SEEQUENT1Region
- SEEQUENT1REGION
- Seequent2region
- NorthernSeequent1Region
- SEEQUENT3regionExtra
- #seequentXregion#

Examples that will not match include:

- Seequent12Region
- SeequentReg1ion
- SeequentRegion

**regexp(t, 'pattern')**

This function returns true if the text sequence t matches pattern, where [pattern] is follows regular expression matching rules. This is case insensitive.

*Example*

=

*Explanation*

filter will be true if [name] matches the regexp pattern Seequent.Region. Examples of [name] that will match include:

- SEEQUENT1Region
- SEEQUENT1REGION
- Seequent2region
- NorthernSeequent1Region
- SEEQUENT3regionExtra
- #seequentXregion#

Examples that will not match include:

- Seequent12Region
- SeequentReg1ion
- SeequentRegion

## Date/Time

**'now' Current timestamp**

Use this item to add the current date-and-timestamp at the insertion point.

*Example*

=

*Explanation*

While entering an expression, the 'now' selection has been chosen and a date-and-timestamp has been entered at the insertion point.

**'today' Current date**

Use this item to add the current datestamp at the insertion point.

*Example*

=

### Explanation

While entering an expression, the 'today' selection has been chosen and a date-and-timestamp has been entered at the insertion point.

### dateonly(timestamp)

This function takes a date-and-timestamp and cuts off the timestamp to leave just the date.

### Example

=

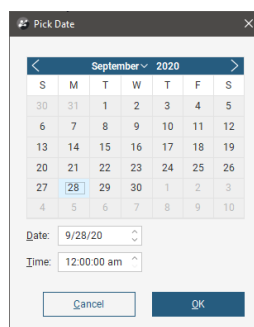
### Explanation

While the [past] variable has a full date-and-timestamp, the dateonly function strips the time off and leaves just the date.

### Pick timestamp...

This selection opens a dialog box prompting you for a date and a time that will be entered at the insertion point when you click OK.

### Example



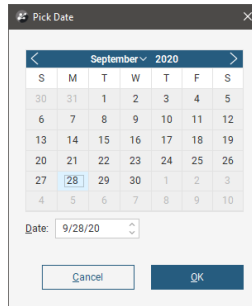
### Explanation

Use the year picker, month picker, date selector and Time field to specify a date and time. If the above date is entered, it will be represented in the expression as @2020-01-22 00:00:00.

### Pick date...

This selection opens a dialog box prompting you for a date that will be entered at the insertion point when you click OK.

*Example*



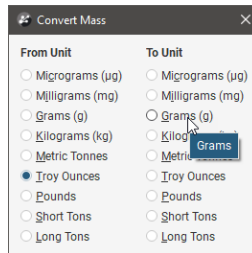
*Explanation*

Use the year picker, month picker and date selector to specify a date. If the above date is entered, it will be represented in the expression as @2020-01-22.

## Unit Conversion

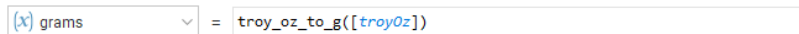
**Mass/weight...**

This selection opens a dialog box prompting for the From Unit and To Unit for the conversion.



When you have selected the two units, the dialog box disappears and the conversion function you require will have been inserted into the expression at the cursor, and the cursor repositioned so you can enter the source of the value to be converted.

*Example*

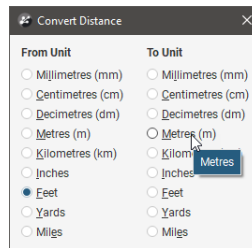


*Explanation*

The troy\_oz\_to\_g function has been entered by the dialog box, and the variable [troyOz] has been entered as the input. The numeric calculation grams will be given the output of the troy\_oz\_to\_g function.

## Distance...

This selection opens a dialog box prompting for the From Unit and To Unit for the conversion.



When you have selected the two units, the dialog box disappears and the conversion function you require will have been inserted into the expression at the cursor, and the cursor repositioned so you can enter the source of the value to be converted.

### Example

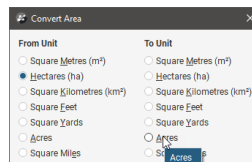
=

### Explanation

The `ft_to_m` function has been entered by the dialog box, and the variable `[feet]` has been entered as the input. The numeric calculation metres will be given the output of the `ft_to_m` function.

## Area...

This selection opens a dialog box prompting for the From Unit and To Unit for the conversion.



When you have selected the two units, the dialog box disappears and the conversion function you require will have been inserted into the expression at the cursor, and the cursor repositioned so you can enter the source of the value to be converted.

### Example

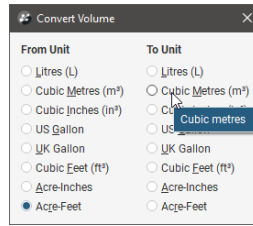
=

### Explanation

The `ha_to_acre` function has been entered by the dialog box, and the variable `[hectares]` has been entered as the input. The numeric calculation acres will be given the output of the `ha_to_acre` function.

## Volume...

This selection opens a dialog box prompting for the From Unit and To Unit for the conversion.



When you have selected the two units, the dialog box disappears and the conversion function you require will have been inserted into the expression at the cursor, and the cursor repositioned so you can enter the source of the value to be converted.

### Example

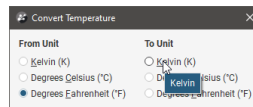
=

### Explanation

The acre\_ft\_to\_m3 function has been entered by the dialog box, and the variable [acre-ft] has been entered as the input. The numeric calculation volm3 will be given the output of the acre\_ft\_to\_m3 function.

## Temperature...

This selection opens a dialog box prompting for the From Unit and To Unit for the conversion.



When you have selected the two units, the dialog box disappears and the conversion function you require will have been inserted into the expression at the cursor, and the cursor repositioned so you can enter the source of the value to be converted.

### Example

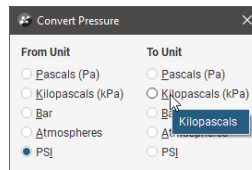
=

### Explanation

The degF\_to\_K function has been entered by the dialog box, and the variable [fahrenheit] has been entered as the input. The numeric calculation Kelvin will be given the output of the degF\_to\_K function.

## Pressure...

This selection opens a dialog box prompting for the From Unit and To Unit for the conversion.



When you have selected the two units, the dialog box disappears and the conversion function you require will have been inserted into the expression at the cursor, and the cursor repositioned so you can enter the source of the value to be converted.

### Example

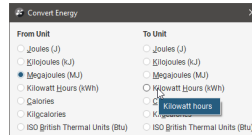
`(x) pressure_kPa` = `psi_to_kPa([pressure_psi])`

### Explanation

The `psi_to_kPa` function has been entered by the dialog box, and the variable `[pressure_psi]` has been entered as the input. The numeric calculation `pressure_kPa` will be given the output of the `psi_to_kPa` function.

## Energy...

This selection opens a dialog box prompting for the From Unit and To Unit for the conversion.



When you have selected the two units, the dialog box disappears and the conversion function you require will have been inserted into the expression at the cursor, and the cursor repositioned so you can enter the source of the value to be converted.

### Example

`(x) kiloWatt Hours` = `MJ_to_kWh([MegaJoules])`

### Explanation


The `MJ_to_kWh` function has been entered by the dialog box, and the variable `[MegaJoules]` has been entered as the input. The numeric calculation `kiloWatt Hours` will be given the output of the `MJ_to_kWh` function.

## Colour

### rgb\_to\_hex(R, G, B)

This function returns an RGB hex sequence by combining three other values, interpreting them as R (red), G (green) and B (blue) values. It requires that the new item selected when creating a calculation is of type RGB to Hex Colour, not Numeric calculation or Category calculation. While this data could be combined into a hex value using Excel prior to import, point clouds from sources such as LIDAR data can be millions of rows in length, beyond the capacity of Excel. This function allows data collected as separate RGB values to be combined into a RGB hex value that can be used within Leapfrog Geo.

#### Example

 r2h = rgb\_to\_hex([R] , [G] , [B])

#### Explanation




Each row will get a new column combining the data in columns R, B and G into r2h. For instance a row with values R=41, G=61 and B=60 will be assigned the value #293d3c (41 dec is 29 hex, 61 dec is 3d hex, 60 dec is 3c hex). The resulting RGB hex column can then be selected as a colouration in the Select display field in the shape list.

## Random Number Generators

### random\_float(start, end, seed)

This function returns a pseudo-random real number between the start and end values (inclusive). When the random\_float function is added to a calculation, by default the seed value is set to a number that relates to the exact time the function was added. This seed value can be changed. Because the function is a pseudo-random number generator, the use of the same seed will result in the same series of random numbers. Use a different seed number to get a different series of randomised values.

#### Example

 rand\_float1 = random\_float(100, 105, 42)  
 rand\_float2 = random\_float(100, 105, 42)  
 rand\_float3 = random\_float(100, 105, 1627616440868)

#### Explanation

Each row will get three new columns identified as rand\_float1, rand\_float2 and rand\_float3, and the values in each column will be populated using the pseudo-random number generator to have real values between 100 and 105 inclusive. Because rand\_float1 and rand\_float2 have been seeded with the same value, both columns will be identical, but the rand\_float3 column will have different values because the default seed was used.



rand_float1	rand_float2	rand_float3
103.869780243	103.869780243	102.721440018
102.194392199	102.194392199	104.473263145
104.2929896	104.2929896	101.2581808
103.486840145	103.486840145	103.725987018
100.470886739	100.470886739	104.580258067
104.878111758	104.878111758	100.760961314
103.80569851	103.80569851	101.190408922
103.930321526	103.930321526	102.345841052
100.640568163	100.640568163	103.11564955
102.251929689	102.251929689	102.277267318
101.853990121	101.853990121	103.463173082
104.633824944	104.633824944	101.725028594
103.2193256	103.2193256	101.290210761
104.113808066	104.113808066	104.165937719
102.217070994	102.217070994	100.393105835

### random\_integer(start, end, seed)

This function returns a pseudo-random whole number between the start and end values (inclusive). When the random\_integer function is added to a calculation, by default the seed value is set to a number that relates to the exact time the function was added. This seed value can be changed. Because the function is a pseudo-random number generator, the use of the same seed will result in the same series of random numbers. Use a different seed number to get a different series of randomised values.

#### Example

rand_int1	=	random_integer(100, 105, 42)
rand_int2	=	random_integer(100, 105, 42)
rand_int3	=	random_integer(100, 105, 1627616999023)

#### Explanation

Each row will get three new columns identified as rand\_int1, rand\_int2 and rand\_int3, and the values in each column will be populated using the pseudo-random number generator to have whole numbers between 100 and 105 inclusive. Because rand\_int1 and rand\_int2 have been seeded with the same value, both columns will be identical, but the rand\_int3 column will have different values because the default seed was used.

rand_int1	rand_int2	rand_int3
100.0	100.0	103.0
104.0	104.0	101.0
103.0	103.0	100.0
102.0	102.0	101.0
102.0	102.0	101.0
105.0	105.0	105.0
100.0	100.0	105.0
104.0	104.0	103.0
101.0	101.0	105.0
100.0	100.0	105.0
103.0	103.0	101.0
105.0	105.0	101.0
104.0	104.0	103.0
104.0	104.0	104.0
104.0	104.0	105.0

**random\_normal(mean, stddev, seed)**

This function returns a pseudo-random real number conforming to a normal distribution centred around the mean with a standard deviation stddev. When the random\_normal function is added to a calculation, by default the seed value is set to a number that relates to the exact time the function was added. This seed value can be changed. Because the function is a pseudo-random number generator, the use of the same seed will result in the same series of random numbers. Use a different seed number to get a different series of randomised values.

*Example*

```

rand_norm1 = random_normal(75, 5, 42)
rand_norm2 = random_normal(75, 5, 42)
rand_norm3 = random_normal(75, 5, 1627617529726)
    
```

*Explanation*

Each row will get three new columns identified as rand\_norm1, rand\_norm2 and rand\_norm3, and the values in each column will be populated using the pseudo-random number generator to have real values in a normal distribution centred on 75 with a standard deviation of 5. Because rand\_norm1 and rand\_norm2 have been seeded with the same value, both columns will be identical, but the rand\_norm3 column will have different values because the default seed was used.

rand_norm1	rand_norm2	rand_norm3
76.5235853988	76.5235853988	76.2716369854
69.8000794688	69.8000794688	79.7006526432
78.752255979	78.752255979	72.9053594793
79.702823582	79.702823582	78.7379871609
65.2448240567	65.2448240567	75.0241169211
68.4891024657	68.4891024657	68.4630512157
75.6392020158	75.6392020158	72.2035708489
73.4187870383	73.4187870383	70.6880847977
74.9159942125	74.9159942125	80.2305542732
70.7347803621	70.7347803621	73.1696053327
79.3969898743	79.3969898743	72.0383011632
78.8889596771	78.8889596771	74.8238341785
75.3301534878	75.3301534878	74.0294041393
80.6362060348	80.6362060348	78.20390226
77.3375467113	77.3375467113	73.5473604839

Here univariate statistics histograms are shown demonstrating the resulting normal distributions produced (note the first two are identical because the same seed was used):

