community*viz*®

Working with formulas and functions in

Scenario 360

Working with formulas and functions in Scenario 360
Support Document v2.0
Updated November 2024

Table of Contents

About formulas

Formulas are expressions that specify how the elements of an analysis depend upon one another. They are statements in an equation of facts, rules, principles, or other logical relationships. The ability of Scenario 360 to calculate values dynamically, using formulas, is a powerful and unique tool; it enables you to make changes in the analysis and see the results immediately.

Types of formulas
- **Indicator formulas** specify the value of indicators, which quantify information that pertains to a scenario as a whole. Example indicators include cost of roads, number of school-age children in a neighborhood, or town tax revenues. Indicator values are often charted in Scenario 360 analyses. Each indicator in an analysis is unique; that is, one indicator has one name and one value per scenario.
- **Attribute formulas** specify the value of dynamic attributes, which are changeable characteristics associated with particular features on the map. Example attributes include name of a road, number of children living in a house, or taxes for a particular lot. Attribute values are usually found by looking at symbols on a map or by clicking on a particular feature to open its attribute table.

For example, an attribute formula might be used to calculate the cost of each proposed road feature on a map. An indicator formula might be used to sum the total costs for <u>all</u> roads in a scenario.

It is usually easiest to create a formula using the **Formula Wizard**. Follow the on-screen directions to create a formula, and if you wish, use the **Formula Editor** to make changes. You may also create formulas directly with the Formula Editor. The Formula Editor also provides access to additional functions not available in the Formula Wizard. For more information on the Formula Wizard, see page 4. For more information on the Formula Editor, see page 20.

Tips for creating formulas

Scenario 360 offers a convenient way to preview a formula you have created by providing a **Preview** button in both the Formula Wizard and Formula Editor. When you click this button, the program evaluates your formula using existing conditions in the active scenario and displays the results. Use the **Preview** button to check the results to see if they are what you expected, paying special attention to all zeroes or a long string of identical attribute values. If you click the **Preview** button when editing a formula, the prompt will appear with the first ten records for the layer you are working on. Click the **Show Me More Results** button to view the next ten records.

It is important to use consistent units throughout your formula. For example, if your lot sizes are in meters but your tax rates are per hectare, you need to use a conversion factor to produce a reasonable result. The Formula Wizard automatically checks units and prompts you for conversion factors, but it is prudent to watch for strange units or combinations of units that may have "fooled" the Wizard.

Very long formulas can be hard to read and hard to check. It is good practice to break up large formulas into smaller ones by doing smaller parts of the calculation and then combine them together. Lookup tables (more information on page 25) are another good way to shorten formulas.

Pay attention to data types. Numeric attributes cannot contain formulas that produce text, and vice versa. When working with text in formulas, enclose any text in quotes.

Formulas are normally re-evaluated, and their results updated, every time one of the following events occurs:
- One or more features is added, deleted, or changed by editing the map or linking layers
- A dynamic attribute's value is changed by editing, sketching, external table links, or other means

- One or more assumptions are changed and applied by clicking the **Apply** button
- A formula is created or changed and then saved

Only those formulas that use the affected components are re-evaluated

Formula color-coding

On the edit formula tab (indicators and attributes) and in the Formula Editor, formulas are color-coded as follows:
- Functions and operators are displayed with **bold** text
- Strings are displayed with dark blue text
- Targets, conversions, and layers are displayed with blue text
- Comments are displayed with green text

Conversion factors

The chart below lists the conversion formulas used in the Conversion Factor value generator.

Convert From	Convert To	Multiply By
Acre-Feet	Cu Feet	4.35600×10^4
	Cu Meters	1.23348×10^3
	Gallons (US)	3.25900×10^5
	Liters	1.23350×10^6
Acres	Hectares	4.04687×10^{-1}
	Sq Feet	4.35600×10^4
	Sq Feet (US Survey)	4.35598×10^4
	Sq Kilometers	4.04686×10^{-3}
	Sq Meters	4.04686×10^3
Cu Feet	Acre-Feet	2.29600×10^{-5}
	Cu Meters	2.83800×10^{-2}
	Gallons (US dry)	6.42851
	Gallons (US liquid)	7.48052
	Liters	28.3169
Feet	Kilometers	3.04800×10^{-4}
	Meters	3.04800×10^{-1}
	Miles (statute)	1.89000×10^{-4}
Gallons (US liq)	Acre-Feet	3.06890×10^{-6}
	Cu Feet	1.33680×10^{-1}
	Cu Meters	3.78500×10^{-3}
	Liters	3.78540
Hectares	Acres	2.47100
	Sq Feet	1.07639×10^5
	Sq Kilometers	1.00000×10^{-2}
	Sq Meters	1.00000×10^4
Kilometers	Feet	3.28084×10^3
	Meters	1.00000×10^3
	Miles (statute)	6.21370×10^{-1}
Liters	Cu Feet	3.53100×10^{-2}
	Cu Meters	1.00000×10^{-3}
	Gallons (US Liquid)	2.64170×10^{-1}
Meters	Feet	3.28084
	Feet (US Survey)	3.28083
	Miles (naut, int)	5.39960×10^{-4}
	Miles (statute)	6.21370×10^{-4}
Miles (Statute)	Feet	5.28000×10^3
	Kilometers	1.60935
	Meters	1.60935×10^3
Radians	Degrees	57.2958
Sq Feet	Acres	2.29570×10^{-5}
	Sq Meters	9.29000×10^{-2}
Sq Feet (US Survey)	Acres	2.29570×10^{-5}
Sq Kilometers	Acres	2.47105×10^2
	Sq Feet	1.07639×10^7
	Sq Feet (US Survey)	1.07639×10^7
	Sq Meters	1.00000×10^6
Sq Meters	Acres	2.47110×10^{-4}
	Hectares	1.00000×10^{-4}
	Sq Feet	10.7639
	Sq Kilometers	1.00000×10^{-6}

Working with the Formula Wizard

The **Formula Wizard** assists you in constructing the most common types of analysis formulas. You can access the Formula Wizard when creating dynamic attributes or indicators or editing formulas for existing dynamic attributes or indicators.
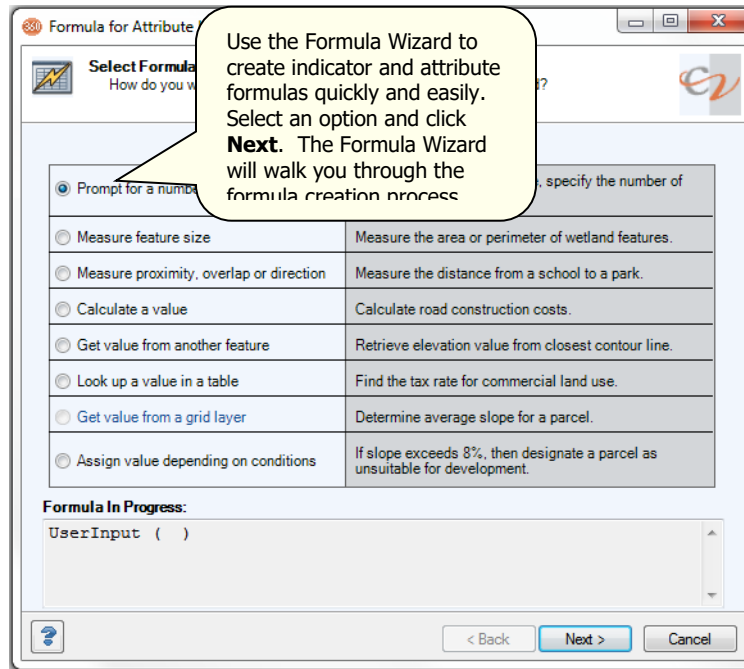
Types of formulas you can create with the Formula Wizard

Indicator formulas
- Count the features in a layer
- Add the values of an attribute
- Average the values of an attribute
- Find the maximum or minimum value of an attribute
- Reference a specific value

Attribute formulas
- Prompt for a number, text, or yes/no answer
- Measure feature size, proximity, overlap, or direction
- Calculate a value
- Get a value from another feature or a grid layer
- Look up a value in a table
- Assign a value depending on conditions
- Test proximity or overlap with other features

If the Wizard cannot create the entire desired formula, you can leave placeholders, consisting of a generic formula element, at certain points in a formula. Then, after the Wizard session, you can reopen a Wizard-created formula using the Formula Editor and edit the placeholders in any way you choose.

Finishing Formula Wizard formulas

When you finish a formula you are creating in the Formula Wizard, you will see a screen similar to the one below. Use the instructions provided in the screen below to complete your formulas.

Creating indicator formulas with the Formula Wizard

You can use the Formula Wizard to create indicator or attribute formulas. The Formula Wizard provides six formula functions for indicators: `Count, Add, Average, Maximum, Minimum,` and `Get`.

Counting the features in a layer

Use the **Count the features in a layer** option to tally the units of a group of features within a specific layer. This option will add the number of features found in the target layer that satisfy the condition specified in a where clause.

1. Open the **Formula Wizard** when creating or editing an indicator.
2. Click the **Count the features in a layer** radio button and click **Next**.
3. Click the drop-down for **Select layer** and select the data layer containing the features to use in the formula.
4. Click the **count all features** radio button.
   -OR-
   Click the **count selected features** radio button. In this case, the program will activate the drop-down fields for **Attribute**.
   a. Click the **attribute** drop-down field to select an attribute.
   b. Click the **mathematical operator** field to select an operator.
   c. Click the **condition** field to select a condition for the where clause.
   d. If you wish to add conditions, click the **New** button. Click the **Delete** button to remove any additional features.

5. View your formula using the **Formula In Progress** area at the bottom of the screen.
6. Click **Next**.
7. If you are finished with your formula, click the **No, I'm finished with this formula** radio button.

   -OR-

   If you wish to continue constructing your formula (add functions and/or syntax), click the **Yes, I want to include additional calculations or values in this formula** radio button.

   a. Click the mathematical operator field to select an operator.
   b. Select a formula function, click **Next**, and return to step #3.
9. Click the **Preview Formula Result** button to view the scenarios in your analysis and their associated indicator values, the results of the formula you just created. You can use this information to judge whether the formula is returning valid results.   When finished viewing the information, click the **Close** button to return to the Formula Wizard.

   > **Tip**: If your analysis has more information than the program can display in the results window at one time, click the **Show More Results** button to view additional results.
10. If you need to make adjustments to your formula, you can either return to previous screens in sequence using the **< Back** button or open the Formula Editor where you can revise or customize formulas to arrive at the desired results.

Warning! As you retrace your steps through the Wizard using the **< Back** button, you may lose certain elements of your formula. Furthermore, you cannot return to the Formula Wizard for the current formula once you launch the Formula Editor.

Once you have completed constructing your formula, click the **Next** button to move to the **Finish Formula** window. For information on using this window see "Finishing Formula Wizard formulas" on page 5.

You can make additional changes to your formula or add advanced calculations using the Formula Editor by clicking the **Open Formula Editor** button. Otherwise, click **Finish**.

See also: "Count function" on page 55.

Adding the values of an attribute

Use the **Add the values of an attribute** option to combine a list of figures, or numbers of a specific feature, to arrive at a sum. This option will total the values of particular attributes for multiple features

within a layer. (Note: You may find it quicker to use the **Sum** Σ button on the Attributes List.)

Open the **Formula Wizard** when creating or editing an indicator.
Click the **Add the values of an attribute** radio button and click **Next**.
Click the drop-down for **Select layer** and select the data layer containing the features to use in the formula.
Click the drop-down for **Select attribute to sum** and select an attribute.
Click the **sum all features** radio button.
-OR-
Click the **sum selected features** radio button. In this case, the program will activate the drop-down fields for **Attribute**.
Click the **attribute** drop-down field to select an attribute.
Click the **mathematical operator** field to select an operator.
Click the **condition** field to select a condition for the where clause.
If you wish to add features, click the **New** button. Click the **Delete** button to remove any additional features.
Follow steps 6-12 under "Counting the features in a layer" on page 5.

See also: "Sum function" on page 107.

Averaging the values of an attribute

Use the **Average the values of an attribute** option to determine the average value of a set of numbers, also know as the "Mean". This option will calculate the average value in a numeric attribute or the average size in a shape attribute.

Open the **Formula Wizard** when creating or editing an indicator.
Click the **Average the values of an attribute** radio button and click **Next**.
Click the drop-down for **Select layer** and select the data layer containing the features to use in the formula.
Click the drop-down for **Select attribute to average** and select an attribute.
Click the **average all features** radio button.
-OR-
Click **average selected features** radio button. In this case, the program will activate the drop-down fields for **Attribute**.
Click the **attribute** drop-down field to select an attribute.
Click the **mathematical operator** field to select an operator.
Click the **condition** field to select a condition for the where clause.
If you wish to add features, click the **New** button. Click the **Delete** button to remove any additional features.
Follow steps 6-12 under "Counting the features in a layer" on page 5.

See also: "Mean (average) function" on page 74.

Finding the maximum value of an attribute

Use the **Find the maximum value of an attribute** option to find the maximum value in a numeric attribute or the maximum size in a shape attribute.

Open the **Formula Wizard** when creating or editing an indicator.
Click the **Find the maximum value of an attribute** radio button and click **Next**.
Click the drop-down for **Select layer** and select the data layer containing the features to use in the formula.
Click the drop-down for **Select attribute to find max of** and select an attribute.
Click the **find max of all features** radio button.
-OR-
Click **find max of selected features** radio button. In this case, the program will activate the drop-down fields for **Attribute**.
Click the **attribute** drop-down field to select an attribute.
Click the **mathematical operator** field to select an operator.
Click the **condition** field to select a condition for the where clause.
If you wish to add features, click the **New** button. Click the **Delete** button to remove any additional features.
Follow steps 6-12 under "Counting the features in a layer" on page 5.

See also: "Max (maximum) function" on page 74.

Finding the minimum value of an attribute

Use the **Find the minimum value of an attribute** option to calculate the minimum value in a numeric attribute or the minimum size in a shape attribute.

Open the **Formula Wizard** when creating or editing an indicator.
Click the **Find the minimum value of an attribute** radio button and click **Next**.
Click the drop-down for **Select layer** and select the data layer containing the features to use in the formula.

Click the drop-down for **Select attribute to find min of** and select an attribute.
Click the **find min of all features** radio button.
-OR-
Click **find min of selected features** radio button. In this case, the program will activate the drop-down fields for **Attribute**.
Click the **attribute** drop-down field to select an attribute.
Click the **mathematical operator** field to select an operator.
Click the **condition** field to select a condition for the where clause.
If you wish to add features, click the **New** button. Click the **Delete** button to remove any additional features.
Follow steps 6-12 under "Counting the features in a layer" on page 5.

See also: "`Min` (minimum) function" on page 77.

Referencing a specific value

Use the **Reference a specific value** option to retrieve a value from elsewhere in the analysis for use in the formula on which you are working. This option will display a **Select Value** dialog screen. You can reference the value of an assumption, indicator, conversion factor, or a specific value that you enter.

See also: "`Get` function" on page 57 and "`GetFromClosest` (get from closest) function" on page 57.

Open the **Formula Wizard** when creating or editing an indicator.
Click the **Reference a specific value** radio button and click **Next**.
Click the **Current value of assumption** radio button, **Current value of an indicator** radio button, or **Enter a specific value** radio button.
Click the **assumption** or **indicator** drop-down field to select an assumption or indicator to include or type the value that the logical or numeric operand will use in the formula.
Click **Next**.
Follow steps 6-12 under "Counting the features in a layer" on page 5.

Creating attribute formulas with the Formula Wizard

Attribute formulas specify the value of dynamic attributes, which are changeable characteristics associated with particular features on the map. Example attributes include name of a road, number of children living in a house, or taxes for a particular lot. Attribute values are usually found by looking at symbols on a map or by clicking on a particular feature to open its attribute table.

Only dynamic data layers can contain dynamic attributes. That means a data layer must be designated as dynamic in order for you to be able to create a formula for any of its attributes.

Attributes can be designated as numeric, yes/no, or text. Each of these attribute types include different options when constructing formulas.

Creating numeric attribute formulas

The Formula Wizard includes eight numeric formula functions to choose from:

Prompt for a number
Prompting the user for a number if certain conditions are met
Prompting the user for a specific number
Prompting the user to select a numeric value from a list
Prompting the user to select a numeric value from a list if certain conditions are met
Measure size
Measure proximity, overlap, or direction
Calculate a value
Get a value from another feature
Look up a value in a table
Get a value from a grid
Assign a conditional value

Prompting the user for a number if certain conditions are met

Use the **If conditions are met, prompt user to type value** option to set up a user input screen that, in certain circumstances, will prompt users to type a specific number when they create a new feature in the analysis. An example use of this option would be prompting users to type the number of acres if the area is designated an open space area. This option uses `UserInput` (more information on this function on page 113) and `If..then` functions (more information on the If…then function on page 62).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Prompt for a number** radio button and click **Next**.
Click the **If conditions are met, prompt user to type a value** radio button and click **Next**.
Set the conditions that the user must meet in order to receive the prompt (the "If" statement).
Click the **attribute** drop-down field to select an attribute.
Click the **mathematical operator** field to select an operator.
Click the **condition** field to select a condition.
If you wish to add conditions, click the **New** button. Click the **Delete** button to remove any unwanted conditions.
Click **Next**.
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Select the type of size measurement you wish to make (area, length, or perimeter) and click **Next**.
Click the **Preview Formula Result** button to view the user input screen you created and to view the associated attribute values. You can use this information to judge whether the formula is returning valid results. When finished viewing the information, click the **Close** button to return to the Formula Wizard.
Tip: If your analysis has more information than the program can display in the results window at one time, click the **Show More Results** button to view additional results.
If you need to make adjustments to your formula, you can either return to previous screens in sequence using the **Back** button or open the Formula Editor where you can revise or customize formulas to arrive at the desired results.
Warning! As you retrace your steps through the Wizard using the **Back** button, you may lose certain elements of your formula. Furthermore, you cannot return to the Formula Wizard for the current formula once you launch the Formula Editor.
You can make additional changes to your formula or add advanced calculations using the Formula Editor by clicking the **Open Formula Editor** button. Otherwise, click **Finish**.

Prompting the user for a specific number

Use the **Prompt for a specific numeric value** option to set up a user input screen that will prompt users to input specific number when they create a new feature in the analysis. This option uses the `UserInput` function (more information on this function on page 113).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Prompt for a number** radio button and click **Next**.
Click the **Prompt for a specific numeric value** radio button and click **Next**.
Type a question that prompts the user to enter a value, for example, "How many high school-aged children per household?".
If you wish the program to display a default number, click the **Provide Default Value** check box and type a number in the provided field. An example of the user prompt screen is displayed.
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Follow steps 9-11 under "Prompting a user for a number if certain conditions are met" on page 9.

Prompting the user to select a numeric value from a list

Use the **Prompt for a number** option to set up a user input screen that will prompt users to type a number or select a specific number when they create a new feature in the analysis. Use the **Prompt to select a value from a list** sub-option to set up a user input screen that will prompt users to select a specific number from a list when they create a new feature in the analysis. This option uses the `UserChoice` function (more information on this function on page 111).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Prompt for a number** radio button and click **Next**.
Click the **Prompt to select a value from a list** radio button and click **Next**.
Type a question that prompts the user to select a value from a list of value options.
If you wish to create your own list of values, click the **Create list** radio button. Type your list of values in the Type New List Item window then click to select them and click the **Add item to list** button ➡.
Remove unwanted items from the list using the **Remove item from list** button ⬅. You can rearrange the items in your list using the **Move item up** or **Move item down** buttons.
-OR-
Click the **Get list from table or layer** radio button to create your list based on values from a table or layer in your analysis. Click the **Layer or table** drop-down field and select a layer or table, then click the **Get list from attribute** drop-down field and select an attribute.
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Follow steps 9-11 under "Prompting a user for a number if certain conditions are met" on page 9.

Prompting the user to select a numeric value from a list if certain conditions are met

Use the **Prompt for a number** option to set up a user input screen that will prompt users to type a number or select a specific number when they create a new feature in the analysis. Use the **If conditions are met, prompt user to select a value** sub-option to set up a user input screen that, in certain circumstances, will prompt users to select a specific number from a list when they create a new feature in the analysis. An example use of this option would be to prompt a user to select the number of dwelling units in a residential area. This option uses `UserChoice` (more information on this function on page 111) and `If..then` functions (more information on the If…then function on page 62).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Prompt for a number** radio button and click **Next**.
Click the **If conditions are met, Prompt to select a value** radio button and click **Next**.
Set the conditions that the user must meet in order to receive the prompt (the "If" statement).

Click the attribute drop-down field to select an attribute.
Click the mathematical operator field to select an operator.
Click the condition field to select a condition.
If you wish to add conditions, click the **New** button. Click the **Delete** button to remove any unwanted conditions.
Click **Next**.
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Select the type of size measurement you wish to make (area, length, or perimeter) and click **Next**.
Follow steps 9-11 under "Prompting a user for a number if certain conditions are met" on page 9.

Measuring feature size

An example use of the **Measure feature size** option would be to determine the size of a lake or a land parcel, or the length of a road . This option uses the `Area` (more information on this function on page 44) and `MinDistance` (more information on this function on page 78) functions.

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Measure feature size** radio button and click **Next**.
Click the radio button to select the type of size measurement you wish to make (area, length, or perimeter).
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Click the **Preview Formula Result** button to view the user input screen you created and to view the associated attribute values. You can use this information to judge whether the formula is returning valid results. When finished viewing the information, click the **Close** button to return to the Formula Wizard.
Tip: If your analysis has more information than the program can display in the results window at one time, click the **Show More Results** button to view additional results.
If you need to make adjustments to your formula, you can either return to previous screens in sequence using the **Back** button or open the Formula Editor where you can revise or customize formulas to arrive at the desired results.
Warning! As you retrace your steps through the Wizard using the **Back** button, you may lose certain elements of your formula. Furthermore, you cannot return to the Formula Wizard for the current formula once you launch the Formula Editor.
You can make additional changes to your formula or add advanced calculations using the Formula Editor by clicking the **Open Formula Editor** button. Otherwise, click **Finish**.

Measuring proximity, overlap or direction

An example use of the **Measure proximity, overlap or direction** option would be to determine the distance between bus stops or the overlap of a proposed residential parcel onto farmland. This option uses the `MinDistance` (more information on this function on page 78), `OverlapArea` (more information on this function on page 83), and `AngleTo` functions.

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Measure feature size** radio button and click **Next**.
Click the radio button to select the type of size measurement you wish to make (straight line distance or overlap).
Click the drop-down for **Measure distance to/overlap with features in layer** and select a layer.
Click the radio button to **Measure distance between/overlap with all features** to include all of the features in the layer.
-OR-

Click **Measure distance between/overlap with selected features** radio button. In this case, the program will activate the drop-down fields for **Attribute**.
Click the attribute drop-down field to select an attribute.
Click the mathematical operator field to select an operator.
Click the condition field to select a condition for the where clause.
If you wish to add features, click the **New** button. Click the **Delete** button to remove any additional features.
Follow steps 4-8 under "Measuring feature size" on page 11.

Calculating a value

Use the **Calculate a value** option to construct simple calculations by selecting attributes, assumption values, or specific values. An example use of this option would be to compare construction costs for residential vs. rural roads. Using this Formula Wizard option allows you to add, subtract, multiply, or divide specific assumption or attribute values.

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Calculate a value** radio button and click **Next**.
Click the **value** drop-down field and select the value (**Attribute Value**, **Assumption Value**, or **Other Value**) you wish to include in your calculation. The **Other Value** option allows you to type your own value.
Click the **specific assumption or attribute** drop-down field and select the assumption or attribute you wish to include in your calculation. If you selected **Other Value**, type your value into the field provided.
Click the **mathematical operator** field to select an operator.
Repeat steps 3 and 4.
If you wish to add lines, click the **Add Line** button. Click the **Remove** button to remove any additional lines.
Follow steps 4-8 under "Measuring feature size" on page 11.

Getting a value from another feature

The Formula Wizard provides an attribute formula function for determining the value of an attribute from the closest feature in another layer. This option uses the `GetFromClosest` function (more information on this function on page 57). The Formula Wizard will find the closest feature for you. For overlapping features, the largest overlapping feature will be considered the closest.

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Get value from another feature** radio button and click **Next**.
Click the **Layer containing target feature** drop-down field and select the data layer containing the feature you wish to get a value from.
Click the **Get value for attribute** drop-down field and select the attribute you wish to include in your calculation.
Follow steps 4-8 under "Measuring feature size" on page 11.

Looking up a value in a table

The Formula Wizard provides an attribute formula function for using look up tables to retrieve specific values you specify such as costs or population densities. The **Look up value in a table** option uses the `Get` function (more information on this function on page 57).

Open the **Formula Wizard** when creating or editing an indicator.
Click the **Look up value in a table** radio button and click **Next**.

Click the drop-down for **Get value from table** and select the attribute table containing the values you wish to use in the formula.

Click the drop-down for **The value of attribute** and select an attribute value.

Click the **attribute** drop-down field to select an attribute.

Click the **mathematical operator** field to select an operator.

Click the **condition** field to select a condition for the where clause.

If you wish to add features, click the **New** button. Click the **Delete** button to remove any additional features.

Follow steps 4-8 under "Measuring feature size" on page 11.

Getting a value from a grid layer

The Formula Wizard provides a numeric attribute formula function for determining the area, or the minimum, maximum, average, or median values of attributes in a grid layer that overlaps with shapes in the layer containing the attribute you are working with. The **Get value from a grid layer** option uses the `GridMax`, `GridMin`, `GridMean`, `GridMost`, or `GridOverlap` functions (more information on these functions on page 58).

The maximum value sub-option finds the largest value for a specified attribute in the cells overlapped by the current shape. You could use this formula to determine the height of the tallest tree in a forest. The function used for this formula is `GridMax`.

The minimum value sub-option finds the smallest value for a specified attribute in the cells overlapped by the current shape. Use this grid formula to determine the area of a parcel that has the lowest amount of a certain soil type. The function used for this formula is `GridMin`.

The average value sub-option finds the average value for a specified attribute in the cells overlapped by the current shape. Use this value to determine the average slope of a parcel. The function used for this formula is `GridMean`.

The most frequently occurring value sub-option measures the attribute values of all grid cells overlapped by the current shape. The value returned by the most frequently occurring value would be the median value of the cell attribute. You could use this value to find the lot that is closest in size to the average lot in a parcel. The function used for this formula is `GridMost`.

The area of all grid cells that match a certain value sub-option measures the area value of all grid cells overlapped by the current shape. If the current shape is a polygon, the function will count the cells that overlap the current shape and then multiply by the area of a cell. For lines, the program multiplies the number of overlapping cells by the width of the cells. For points, the function will return 1 or 0 (True, False). The function used for this formula is `GridOverLap`. After selecting this radio button you will select the value that overlaps the cells from a drop-down field.

Open the **Formula Wizard** when creating or editing an indicator.

Click the **Get value from a grid layer** radio button and click **Next**.

Click the **For all cells in the grid layer** drop-down field and select a grid layer.

Click the radio button to select the type of value you wish to get from the grid (maximum, minimum, average, most frequently occurring, or overlapping). If you select **The area of all grid cells that match a certain value**, you can select **All Values** from the drop-down field or type a value into the drop-down field.

View your formula using the **Formula In Progress** area at the bottom of the screen.

Click **Next**.

Click the **Preview Formula Result** button to view the user input screen you created and to view the associated attribute values. You can use this information to judge whether the formula is returning valid results.   When finished viewing the information, click the **Close** button to return to the Formula Wizard. **Tip**: If your analysis has more information than the program can display in the results window at one time, click the **Show More Results** button to view additional results.

If you need to make adjustments to your formula, you can either return to previous screens in sequence using the **Back** button or open the Formula Editor where you can revise or customize formulas to arrive at the desired results.

Warning! As you retrace your steps through the Wizard using the **Back** button, you may lose certain elements of your formula. Furthermore, you cannot return to the Formula Wizard for the current formula once you launch the Formula Editor.

You can make additional changes to your formula or add advanced calculations using the Formula Editor by clicking the **Open Formula Editor** button. Otherwise, click **Finish**.

Assigning a value depending on conditions

The Formula Wizard provides an attribute formula function for determining a value depending on a condition you set. The **Assign value depending on condition** option uses the `IfThenElse` function (more information on this function on page 63).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Assign value depending on condition** radio button and click **Next**.
Click the **Attribute** drop-down field and select the attribute for which you wish to set a condition.
Click the mathematical operator field to select an operator.
Click the **condition** drop-down field and select the condition for the `If` statement.
Select a formula for the `Then` statement and click **Next**. Follow the instructions for the formula you choose.
When you have completed the `IfThen` function, you will be asked if you are finished with this part of your formula. If you have completed your formula, click the **No, I'm finished creating If-Then conditions** radio button.
-OR-
If you wish to continue constructing your formula (add functions and/or syntax), click the **Yes, I want to include additional If-Then conditions in this formula** radio button, click **Next**, and repeat steps 3-7.
View your formula using the **Formula In Progress** area at the bottom of the screen and click **Next**.
Select a formula for the `Else ` statement and click **Next**. Follow the instructions for the formula you choose.
When you reach the Finish Formula screen, click the **Preview Formula Result** button to view the user input screen you created and to view the associated attribute values, the results of the formula you just created and input you provided in the input window. You can use this information to judge whether the formula is returning valid results.   When finished viewing the information, click the **Close** button to return to the Formula Wizard.
Tip: If your analysis has more information than the program can display in the results window at one time, click the **Show More Results** button to view additional results.
If you need to make adjustments to your formula, you can either return to previous screens in sequence using the **Back** button or open the Formula Editor where you can revise or customize formulas to arrive at the desired results.
Warning! As you retrace your steps through the Wizard using the **Back** button, you may lose certain elements of your formula. Furthermore, you cannot return to the Formula Wizard for the current formula once you launch the Formula Editor.
You can make additional changes to your formula or add advanced calculations using the Formula Editor by clicking the **Open Formula Editor** button. Otherwise, click **Finish**.

Creating Yes/No attribute formulas

The Formula Wizard includes five Yes/No formula functions to choose from:
Prompt for Yes or No
Test proximity or overlap with other features
Get value from another feature
Look up a value in a table
Assign value depending on conditions

Prompting a user for a yes or no answer

The Formula Wizard provides a yes/no attribute formula function for prompting a user to type or select a yes/no when creating a new feature in the analysis. You can also use the **Prompt an user for Yes or No** option to setup user prompts that request the user to type or select yes/no when certain conditions are met. This option uses the `UserInputB` function (for more information on this function, see page 114).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Prompt for Yes or No** radio button and click **Next**.
Type a question that prompts the user to answer yes or no, for example, "Will this parcel be zoned commercial?".
Click the **Yes** or **No** radio button to select a default response to your question. An example of the user prompt screen is displayed.
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Click the **Preview Formula Result** button to view the user input screen you created and to view the associated attribute values, the results of the formula you just created and input you provided in the input window. You can use this information to judge whether the formula is returning valid results.
When finished viewing the information, click the **Close** button to return to the Formula Wizard.
Tip: If your analysis has more information than the program can display in the results window at one time, click the **Show More Results** button to view additional results.
If you need to make adjustments to your formula, you can either return to previous screens in sequence using the **Back** button or open the Formula Editor where you can revise or customize formulas to arrive at the desired results.
Warning! As you retrace your steps through the Wizard using the **Back** button, you may lose certain elements of your formula. Furthermore, you cannot return to the Formula Wizard for the current formula once you launch the Formula Editor.
You can make additional changes to your formula or add advanced calculations using the Formula Editor by clicking the **Open Formula Editor** button. Otherwise, click **Finish**.

Testing proximity or overlap with other features

The Formula Wizard provides a yes/no attribute formula function for testing proximity or overlap with other features. Use the **Test proximity or overlap with other features** option to create a formula that tests for specific conditions (you define) that determine proximity or overlap of features.

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Test proximity or overlap with other features** radio button and click **Next**.
Click the **Determine whether each feature in the layer** drop-down field and select one of the following options:
Lies within a distance of uses the `MinDistance` function (for more information on this function, see page 78). If you select this function, type the minimum distance in the field next to the drop-down list box. The distance is in map units.

Overlaps/Intersects uses the `Intersects` function (for more information on this function, see page 65).

Has its center in uses the `IsCenterContainedIn` function (for more information on this function, see page 66).

Contains the center of uses the `CenterContains` function (for more information on this function, see page 50).

Is completely contained in uses the `IsContainedIn` function (for more information on this function, see page 68).

Completely contains uses the `Contains` function (for more information on this function, see page 52).

Click the drop-down for **features in layer** and select a layer.

Click the **for all features** radio button.

-OR-

Click the **for selected features** radio button. In this case, the program will activate the drop-down fields for **Attribute**.

Click the **attribute** drop-down field to select an attribute.

Click the **mathematical operator** field to select an operator.

Click the **condition** field to select a condition for the where clause.

If you wish to add features, click the **New** button. Click the **Delete** button to remove any additional features.

Follow steps 5-10 under "Prompting a user for a yes or no answer" on page 15.

Getting a value from another feature

For detailed information on constructing this formula using the Formula Wizard, see page 12.

Looking up a value in a table

For detailed information on constructing this formula using the Formula Wizard, see page 12.

Assigning a value depending on conditions

For detailed information on constructing this formula using the Formula Wizard, see page 14

Creating text attribute formulas

The Formula Wizard includes four text formula functions to choose from:
Prompt analysis for text
Prompting for specific text
Prompting for text if certain conditions are met
Prompting to select a text value from a list
Prompting to select a text value from a list if certain conditions are met
Get a value from another feature
Look up a value in a table
Assign value depending on condition

Prompting for specific text

Use the **Prompt for text** option to set up a user input screen that will prompt users to type or select text when they create a new feature in the analysis. Use the **Prompt for a specific text value** sub-option to set up a user input screen that will prompt users to input text when they create a new feature in the analysis. This option uses the `UserInputS` function (for more information on this function, see page 115).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Prompt for text** radio button and click **Next**.
Click the **Prompt for a specific text value** radio button and click **Next**.
Type a question that prompts the user to enter a text value, for example, "What is the name of the lake?".
If you wish the program to display a default text string, click the **Provide Default Value** check box and type the default text in the provided field. An example of the user prompt screen is displayed.
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Click the **Preview Formula Result** button to view the user input screen you created and to view the associated attribute values. You can use this information to judge whether the formula is returning valid results. When finished viewing the information, click the **Close** button to return to the Formula Wizard.
Tip: If your analysis has more information than the program can display in the results window at one time, click the **Show More Results** button to view additional results.
If you need to make adjustments to your formula, you can either return to previous screens in sequence using the **Back** button or open the Formula Editor where you can revise or customize formulas to arrive at the desired results.
Warning! As you retrace your steps through the Wizard using the **Back** button, you may lose certain elements of your formula. Furthermore, you cannot return to the Formula Wizard for the current formula once you launch the Formula Editor.
You can make additional changes to your formula or add advanced calculations using the Formula Editor by clicking the **Open Formula Editor** button. Otherwise, click **Finish**.

Prompting to select a text value from a list

Use the **Prompt for text** option to set up a user input screen that will prompt users to type or select text when they create a new feature in the analysis. Use the **Prompt to select a value from a list** sub-option to set up a user input screen that will prompt users to select text from a list when they create a new feature in the analysis. This option uses the `UserChoice` function (for more information on this function, see page 111).

Open the Formula Wizard when creating or editing a dynamic attribute.
Click the **Prompt for text** radio button and click **Next**.

Click the **Prompt to select a value from a list** radio button and click **Next**.

Type a question that prompts the user to select text from a list of text options.

If you wish to create your own list of values, click the **Create list** radio button. Type your list of values in the Type New List Item window then click to select them and click the **Add item to list** button ➡.

Remove unwanted items from the list using the **Remove item from list** button ⬅. You can rearrange the items in your list using the **Move item up** or **Move item down** buttons.

-OR-

Click the **Get list from table or layer** radio button to create your list based on values from a table or layer in your analysis. Click the **Layer or table** drop-down field and select a layer or table, then click the **Get list from attribute** drop-down field and select an attribute.

Follow steps 6-10 under "Prompting for specific text" on page 17.

Prompting the user for text if certain conditions are met

Use the **Prompt for text** option to set up a user input screen that will prompt users to type or select text when they create a new feature in the analysis. Use the **If conditions are met, prompt to type a value** sub-option to set up a user input screen that, in certain circumstances, will prompt users to type in a text value when they create a new feature in the analysis. An example use of this option would be to provide a name to a new road only if it is designated as a public road. This option uses the UserInput (for more information on this function, see page 114) and If..then (for more information on this function, see page 62) functions.

Open the Formula Wizard when creating or editing a dynamic attribute.

Click the **Prompt for text** radio button and click **Next**.

Click the **If conditions are met, prompt user to type a value** radio button and click **Next**.

Set the conditions that the user must meet in order to receive the prompt (the "If" statement).

Click the attribute drop-down field to select an attribute.

Click the mathematical operator field to select an operator.

Click the condition field to select a condition.

If you wish to add conditions, click the **New** button. Click the **Delete** button to remove any unwanted conditions.

Click **Next**.

View your formula using the **Formula In Progress** area at the bottom of the screen.

Click **Next**.

Click the drop-down for **Layer containing target feature** and select the data layer containing the features to use in the formula.

Click the drop-down for **Get value for attribute** and select an attribute.

Click **Next**.

Follow steps 8-10 under "Prompting for specific text" on page 17.

Prompting the user to select a text value from a list if certain conditions are met

Use the **Prompt for text** option to set up a user input screen that will prompt users to type or select text when they create a new feature in the analysis. Use the **If conditions are met, prompt user to select a value** sub-option to set up a user input screen that, in certain circumstances, will prompt users to select a specific text value from a list when they create a new feature in the analysis. An example use of this option would be "If this parcel is a residential zone, will the land-use designation be R-1 or R-10?". This option uses UserChoice (for more information on this function, see page 111) and If..then (for more information on this function, see page 62) functions.

Open the Formula Wizard when creating or editing a dynamic attribute.

Click the **Prompt for text** radio button and click **Next**.

Click the **If conditions are met, Prompt to select a value** radio button and click **Next**.

Set the conditions that the user must meet in order to receive the prompt (the "If" statement).
Click the **attribute** drop-down field to select an attribute.
Click the **mathematical operator** field to select an operator.
Click the **condition** field to select a condition.
If you wish to add conditions, click the **New** button. Click the **Delete** button to remove any unwanted conditions.
Click **Next**.
View your formula using the **Formula In Progress** area at the bottom of the screen.
Click **Next**.
Click the drop-down for **Layer containing target feature** and select the data layer containing the features to use in the formula.
Click the drop-down for **Get value for attribute** and select an attribute.
Click **Next**.
Follow steps 8-10 under "Prompting for specific text" on page 17.

Getting a value from another feature

For detailed information on constructing this formula using the Formula Wizard, see page 12.

Looking up a value in a table

For detailed information on constructing this formula using the Formula Wizard, see page 12.

Assigning a value depending on conditions

For detailed information on constructing this formula using the Formula Wizard, see page 14.

Working with the Formula Editor

The **Formula Editor** provides design assistance for analysis formulas. It can assist you in creating advanced or complex analysis formulas. The Formula Editor provides:
Constant access to all formula functions
Point-and-click construction for most formulas
Access to a context-appropriate set of functions
Proper syntax and functionality checking, with diagnosis messages
Search and lookup features for common formula functions
Multiple syntax editing features
Comprehensive syntax editing tools

You can access the Formula Editor when creating dynamic attributes or indicators, or editing formulas for existing dynamic attributes or indicators.
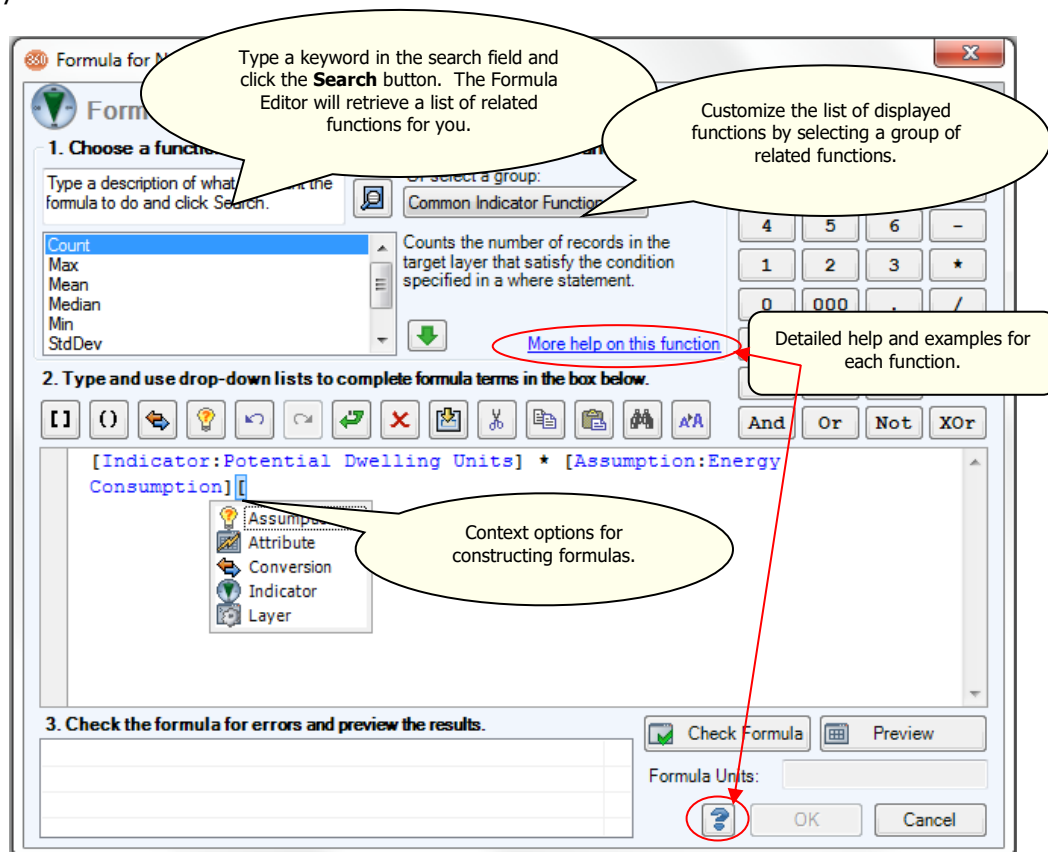
Constructing formulas using the Formula Editor

The **Formula Editor** provides design assistance for analysis formulas. It can assist you in creating advanced or complex analysis formulas. For more information on the Formula Editor, see: About the Formula Editor in the Help system.

Open the Formula Editor.

Type a keyword in the search field then click the **Search** button .
-OR-
Click the **Or select a group** drop-down field and select a function group.
-OR-
Click the **Insert Analysis Component** button [ ] to insert an assumption, attribute, indicator, layer, or conversion.

Tip Click once on any function in the function list to view a brief description of the function (to the right of the function list). For detailed information on any function, click once on the function and click the **More help on this function** link.

Highlight the desired function in the list and click the **Insert Function** button or double-click a function in the function list. The Formula Editor will automatically populate the formula box with the proper syntax and the argument template.

Tip When editing a formula, the program will insert the syntax at the insertion point indicated by your flashing cursor. In addition, the Formula Editor will replace any *selected* text. Mind your cursor!
Formula hyperlinks appear in *italics* and in curly brackets *{}*. Click or right-click on formula hyperlinks to edit arguments.

If your argument contains a `where` condition, edit or remove the where clause. The easiest way to remove a where clause is to right-click on the word "where" and choose "Remove Where Clause."
When you have finished constructing your formula, click the **Check Formula** button. This will initiate syntax and function test routines to validate the current formula. Errors will be displayed in the Error window to alert you to potential problems. You can use the **Check Formula** button repeatedly to incrementally create and validate a complex formula.
If you wish the program to evaluate your formula using existing conditions in the active scenario and display the results, click the **Preview** button.

Working with the Formula Editor toolbar

Use the Formula Editor toolbar when constructing or editing formulas. The Formula Editor toolbar is located at the top of the Formula Editor window.

[] Insert Analysis Component - allows you to select and insert a component of the current analysis (assumption, attribute, indicator, data layer) or a conversion function into the formula. When you click on this button, the Formula Editor will display a pop-up menu in the formula display window.

Double-click an item in the pop-up menu to display a menu of specific components and to add the item to your formula

- Assump...
- Attribute
- Conversion
- Indicator
- Layer

...urce)
Existing Dwelling Units (Land Use)
Main Rural Road Construction Costs (Cost)
Minor Rural Road Construction Costs (Cost)
Potential Dwelling Units (Land Use)
Residential Road Construction Costs (Cost)
Students at Buildout (Schools)
Total Road Construction Costs (Cost)
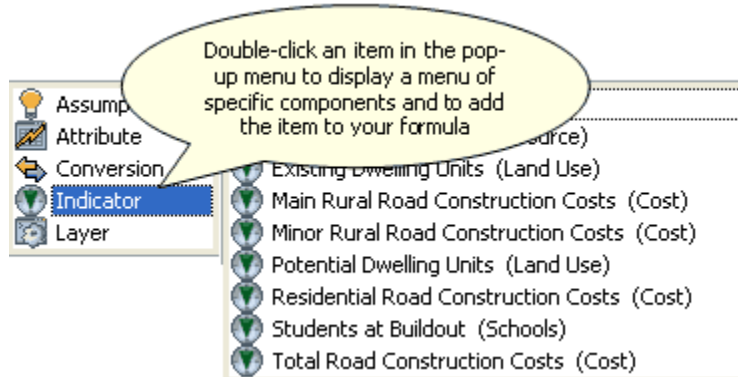
() Insert Parentheses - inserts parentheses at the insertion point indicated by your cursor in the formula display window. You can also use this button to place selected text in parentheses. The Formula Editor will attempt to apply the correct spacing and leave your cursor at the next insertion point.

Insert Conversion - inserts a conversion factor into your formula. Usually you will want to multiply this number by the rest of the formula, so insert a * before or after the conversion factor. When you click on this button, the Formula Editor will display a pop-up menu in the formula display window. For more information on conversion factors, see page 3.

Double-click an item in the pop-up menu to display a menu of conversion factors and to add the item to your formula

Acre-Feet to...
Acres to...
Cu Feet to...
Cu Meters to...
Feet to...
Gallons to...
Hectares to...
Kilometers to...
Liters to...
Meters to...

Acre-Feet
Cu Feet
Cu Meters
Gallons
Liters
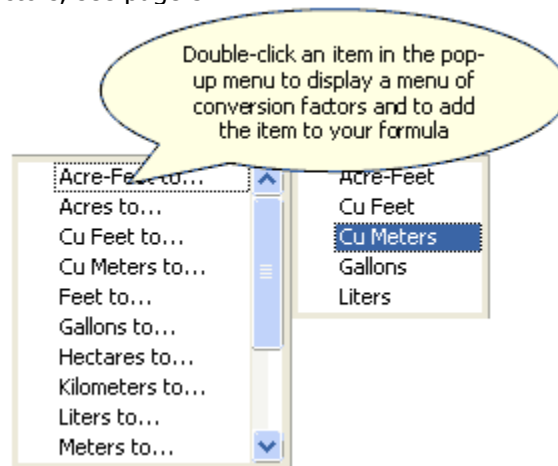
Create New Assumption - allows you to create a new assumption.

Undo - undoes the very last action you took. If you later decide you didn't want to undo an action, click the **Redo** button .

Revert to Original Formula - allows you to undo any edits and return to your original

formula.

✕ Delete - removes selected syntax.

📋 Copy Formula From Another Indicator/Attribute - copies other formulas from either the indicator or attribute list depending on where you are working. Click on this button to display a selection box allowing you to choose an existing indicator or attribute. Double click on an indicator or attribute name to copy its formula into Formula Editor display window. The syntax and format should be correct and the editor will focus your cursor at the next insertion point.

✂ 📋 📋 **Cut**, **Copy**, **Paste**

🔍 Find/Replace - Find and replace text in the formula display window.

A⁺A Font Size - Grow or shrink font in the formula display window.

Working with the Formula Editor keypad

The Formula Editor provides a keypad you can use to enter simple mathematical symbols, operands, and numbers to your formulas. This numbered key pad includes a decimal point, "000", and Boolean operators. The keypad will not perform calculations.

Click once on the keypad buttons to place symbols, operands, or numbers directly into the formula at the insertion point indicated by your cursor.
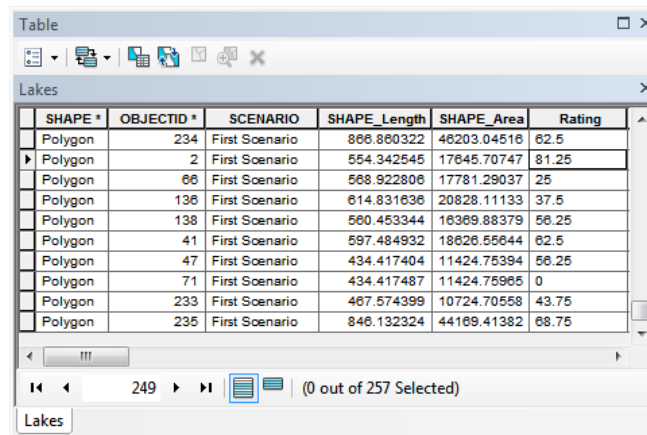
7	8	9	+
4	5	6	-
1	2	3	*
0	000	.	/
=	>	<	^
<>	>=	<=	
And	Or	Not	XOr

Working with attribute formulas

Attribute formulas specify the value of dynamic attributes, which are changeable characteristics associated with particular features on the map. Example attributes include name of a road, number of children living in a house, or taxes for a particular lot. Attribute values are usually found by looking at symbols on a map or by clicking on a particular feature to open its attribute table.

Only dynamic data layers can contain dynamic attributes. That means a data layer must be designated as dynamic in order for you to be able to create a formula for any of its attributes.

Attribute formulas apply to an entire column of attributes within a table. For example, the attribute table below is for a data layer called "Lakes." (You can open the attribute table for any layer by right-clicking on its name in the ArcMap™ table of contents and clicking **Open Attribute Table** on the pop-up menu.) Each row corresponds to one lake. "Rating" is a dynamic attribute, and the number in each row in that column is driven by the attribute formula for Rating.

Working with lookup tables

If a formula has a large number of conditional operators, If/Then statements, or where clauses, you should consider using a lookup table. For example, a long formula might read (in English):
"if land use is agricultural, then the tax rate is X%, and if land use is large-lot residential, then the tax rate is Y%, and if the land use is commercial…, etc.

You can write such a formula more efficiently by storing the tax rate for various land uses in a table and then saying, in concept, "the tax rate is the value in the tax-rate table for this land use." This method also makes it easy to change a tax rate without having to rewrite a formula. To write formulas in this style, click the **Look up a value in a table** radio button in the Formula Wizard, or use the Get function in the Formula Editor.

In this example, if the tax table looked like this…

Tax Rate Table

LAND-USE	TAX RATE
Agricultural	X%
Large-Lot Residential	Y%
Commercial	Z%

The new formula would be
```
Get ( [ Attribute:Tax Rate Table:Tax Rate ],
Where ( [ Attribute:Tax Rate Table:LAND-USE ] = [ Attribute:LAND-USE ] )
```

Note: In this example, text descriptions are used for land-use designations for the purposes of illustration. It is usually better practice to use numbers or codes for matching because there is less risk of typos or alternate spellings.

One easy way to create a lookup table is to use the Scenario 360 Add Data tool. From the 360 Analysis tab's Work Flow view, click Data and then the Add Data button. Choose to add data from a "New Data Layer," and on the following screen choose layer type "Table."

Another easy to create a lookup table is to draft it as an Excel spreadsheet and then use External Table Links to automatically create a matching table and import the values. See Scenario 360 Help on External Table Links for details.

Working with scenario-specific components

In most cases, formulas you create work independently within scenarios. When you create an indicator formula, for example, it returns one value for Scenario A, another for Scenario B, etc. In Scenario A, it uses features, attributes and assumptions (components) for Scenario A. In Scenario B, the indicator value is based on the corresponding components for Scenario B. Similarly, attribute formulas operate within the scenario of the particular feature they are evaluating. If the formula uses a variable assumption as an input, features in Scenario A use the assumption value for Scenario A, and features in Scenario B use the assumption value for Scenario B. If the attribute formula is evaluating a spatial relationship between features in two different layers, such as *MinDistance*, it only measures distances between features within the same scenario.

In some cases, however, you need to reference components from a different scenario. Scenario 360 allows you to do this by appending the scenario name to any component in a formula. For example,
>   [Attribute:Cost] becomes [Attribute:Cost:ScenarioA]

You must use the Formula Editor to add or edit scenario names; they are not supported by the Formula Wizard or the smart editing functions in the Formula Editor.

Here are some example applications:

An indicator that gives the difference in costs between Scenario A and Scenario B:

>   [Indicator:Cost Difference] = [Indicator:Cost:Scenario A] – [Indicator:Cost:Scenario B]

An indicator that finds the maximum population across all scenarios:

[Indicator:Max Population All Scenarios] = Max([Indicator:Population:Scenario A], [Indicator:Population:Scenario B], [Indicator:Population:Scenario C])

To create a chart with a threshold line that reflects the value of a scenario-specific assumption or indicator regardless of which scenario is being displayed, create an indicator whose formula is [Assumption:AssumptionName:ScenarioName] or [Indicator:IndicatorName:ScenarioName]. Use your new indicator as the threshold line in the chart.

Scenario-specific names are automatically appended by Scenario 360 in some formulas when you link layers across scenarios. Specifically, any formulas in the linked layer that reference dynamic target layers have a scenario name appended to the components in the target layer.

Working with custom scripts

Advanced. Custom script functions are an advanced feature intended for users who are familiar with Python, geoprocessing tools, or ArcGIS Model Builder.

Custom scripts let you include your own specialized functions in a Scenario 360 dynamic analysis formula. The Scenario 360 formula functions **CustomScript** (for numeric outputs), **CustomScriptB** (for Boolean outputs), and **CustomScriptS** (for text or string outputs) send parameters you specify to an external model, script, or tool stored in an ArcGIS custom toolbox, execute the script, and then use the script's results as the outputs of the function.

The custom scripts you can use are Python scripts, geoprocessing tools, or models (such as those built with ArcGIS Model Builder) stored in an ArcGIS custom toolbox. If your script is not already part of a custom toolbox, you can create one. See **Creating a Custom Toolbox and Adding Tools.**

When you include a custom script function in a formula, Scenario 360 will execute the script whenever its input parameters change as part of a dynamic update. Custom script functions are normally used in indicator formulas, but may also be used with caution in attribute formulas. Be aware, however, that the script will execute once for each feature in your layer if used in an attribute function.

The script must create a single output parameter (number, Boolean, or string), which Scenario 360 will use as the output of the function. The script may perform additional functions (e.g., creating a new feature class or raster layer), but Scenario 360 will respond only to the output parameter. If your existing script does not create an output, add one such as a parameter called "Output" whose value is 1.

Parameters

The Scenario 360 formula engine will read the parameters of the custom script from its properties. (You can view a script's properties in ArcCatalog). The number of input parameters and data types you provide in the formula must match the inputs and outputs of the script, tool, or model. Custom scripts can have multiple defined input parameters but only a single output parameter. The data type of the output parameter must match the data type of the custom function you use— Number (for the function **CustomScript**), Boolean (for **CustomScriptB**), or String (**CustomScriptS**).

Although the custom toolbox may be named anything and stored anywhere on your system, we recommend naming the toolbox 'CVCustomTools' and storing the toolbox in the C:\CVFiles folder.

Tip The Formula Editor uses a geoprocessing tool that loads all toolboxes on your system and searches for custom scripts, tools, and models. If you have any tools, scripts, or models with the same name, a message will appear warning you about the duplicate tool name. Name or rename your tools so that all names are unique.

Creating a Custom Toolbox and Adding Tools

Custom script functions require Python scripts, models, or tools to be inside an ArcGIS custom toolbox. To create a custom toolbox for use by custom script functions:

Open ArcCatalog.

 If you don't already have a connection to the C:\CVFiles folder in ArcCatalog you can add it by clicking File>Connect to Folder and navigating to it.

Right-click the C:\CVFiles folder and choose New>Toolbox.

Name the new toolbox CVCustomTools. (This is a recommended name, not a required one.) Assign the toolbox a unique Alias.

To add a new model or script tool to your custom toolbox:

To add a new model or script tool from scratch, right-click your new toolbox and select New>Model to create a new model or Add>Script to create a new tool (the Python script must already exist). Create the tool and add the inputs and outputs and appropriate data types.

To add an existing model or script tool, cut and paste an existing model or script tool into the new toolbox. **Note:** after you cut and paste, rename your model or script tool with a unique name.

Below is an example of a simple Python script that illustrates the use of the input and output parameters:

```
import arcpy

inTable = arcpy.GetParameterAsText(0)

result = arcpy.GetCount_management(inTable)

output = int(result.getOutput(0))

arcpy.SetParameter(1, output)
```

This script accesses the GetCount tool in the ArcGIS Data Management Tools toolbox. (GetCount counts the rows in a feature class). It has one input of a feature class that can be a table, shapefile or geodatabase layer, and one output that is a number. (The Scenario 360 formula function **Count** performs a similar function.)

Incremental Updates (Advanced)

Advanced
This article, intended for advanced users, provides optional guidance on how to design your analysis to take maximum advantage of incremental updates.

Incremental updates are one way *Scenario 360* performs analysis calculations under certain conditions. When available, they speed up analysis updates that occur when you edit features. The effect is particularly noticeable when using large datasets and editing individual features. Incremental updates are applied automatically with no action required by you. Advanced users, however, may want to know the conditions under which incremental updates occur in order to take advantage of them.

When incremental updates are used, only those records directly affected by the edited feature are updated, instead of all rows in a table or all features in a layer as is the case in normal dynamic updates.

Examples:

Example 1. An indicator called Smallest Parcel gives the area of the smallest parcel in a layer containing 10,000 parcels using

```
[Indicator:Smallest Parcel] = Min( [Attribute:Parcels:Shape_Area] )
```

If you edit a single parcel, incremental updates for the indicator simply recalculate that one parcel's area and compare it to the current minimum value, updating the result if necessary. This compares favorably in speed to the normal update process, which reexamines the area of all 10,000 parcels to determine a new the minimum.

Example 2. An attribute in the Parcels layer called DistanceToTransit measures distance to the nearest feature in the Transit_Stops layer using

```
[Attribute:Parcels:DistanceToTransit] = MinDistance( [Layer:Transit_Stops] )
```

If the Transit_Stops layer is edited and a new stop is added, the formula update for each parcel would normally reexamine every feature in the Transit_Stops layer to determine which transit stop is closest. When incremental updates are in use, however, only the new transit stop is examined. If the new stop's distance away from a given parcel is shorter than that parcel's current `MinDistance`, the attribute value for that parcel is updated. Otherwise, it is ignored. The incremental update process still works through every parcel, but it only looks at the single new transit stop instead of all transit stops, thus reducing processing time. (For `OverlapArea`, not even every parcel is processed—just the ones that intersect the edited feature.)

When Incremental Updates are Applied
Certain conditions are required for an attribute or indicator to incrementally update; if these are not met, the attribute or indicator will update using the regular process.

Incremental updates are used by the following functions:
```
Min
MinDistance
OverlapArea
Sum
```

In general, to enable incremental updates it is best to use simple formulas that contain exactly one of the functions stated above and nothing else. However, incremental updates can still work in some formulas that combine the function with other elements, as follows:

`Min`, `MinDistance`, and `Sum` may only have a negative sign, as in `–Min()`

`OverlapArea` can be used with a negative sign and/or with –, /, or * (but not +) together with a constant or the `Area` function (e.g.,

`OverlapArea([Layer:LayerName])/2*Area([Layer:LayerName] )`

Non-spatial Where clauses are allowed, as long as the `Where` condition refers only to attributes in the layer being edited. Incremental updates will not work if there is a spatial condition in the where clause. For example, `Where ([Attribute:Transit_Stops:Route] = "BlueLine")` works, but `Where (OverlapArea([Layer:LayerName]) > 50))` does not work.

Note that the `Sum` function is not the same as +, which prevents incremental updates from being used. Thus `Sum( [Attribute:Parcels:PersonsPerHousehold] )` can use incremental updates but `Sum( [Attribute:Parcels:PersonsPerHousehold] ) + 2` cannot.

The only layer that can be referenced in the formula is the layer being edited. If any attributes are used in the formula, they must be in the layer being edited. This includes any layer references in a where clause, as well.
Data types must be appropriate.
For the `MinDistance` function, the layer being edited must not be a shapefile.
For the `OverlapArea` function, the layer being edited must not be multipart.

Notes

Incremental updates are not performed if more than one feature is being edited in a single edit *operation* (a single edit *session* is okay). Some examples of this are if you select multiple features and move the entire group, if you apply a style to a set of selected features using the Apply Style option in Scenario Sketch Tools, or if you delete a set of features by selecting them all and then clicking Delete.

Incremental updates only affect formulas that directly reference the edited layer. Other formulas that depend on the results of an incremental update are not affected. In Example 2 above, a separate attribute called WeightedDistance in the Parcels layer with a formula like

```
                  [Attribute:WeightedDistance] =
        [Attribute:PersonsPerHousehold]*[Attribute:DistanceToTransit]
```

would not use incremental updates even though the DistanceToTransit attribute would.

Temporarily incorrect results may occur in the following procedure:

Start with an attribute or indicator that is using incremental updates

Turn off updates on that component

Edit the target layer

Turn updates back on and do not update your component (either manually or through an automatic update process that may occur)

Edit the target layer again

To avoid this, run a normal manual or automatic update the component after turning updates back on and before editing the target layer again.

Formula syntax

The following table lists the components used in Scenario 360 formulas together with their syntax or the way they you would write them in formulas. The blanks indicate where to type the name of the element.

Analysis Component	Syntax
Indicator	`[Indicator: _____ ]`
Scenario-Specific Indicator	`[Indicator: _____ :ScenarioName]`
Attribute	`[Attribute: _____ ]` *or* `[Attribute: Layer: _____ ]`
Scenario-Specific Attribute	`[Attribute: _____ :ScenarioName` *or* `[Attribute: Layer: _____ :ScenarioName]`
Assumption	`[Assumption: _____ ]`
Scenario-Specific Assumption	`[Assumption:_____ :ScenarioName]`
Layer	`[Layer: _____ ]`
Scenario-Specific Layer	`[Layer: _____ :ScenarioName]`
Conversion Factor	`[Conversion Factor: _____ ]`

The syntax used for attributes depends on whether the attribute being specified is in the same layer as the current feature. If the attribute is in the same layer, the notation is `[Attribute: _____ ]`. If the attribute is in a different layer called LayerA, the syntax is `[Attribute: LayerA:_____ ]`.

Formulas also use the following symbols:

( )	Parentheses are used to group computations together and to group together function arguments.
[ ]	Square brackets are used around names of elements.
{ }	Curly brackets are used to denote content to be provided by the person writing a formula.
,	Commas are used to separate arguments within a function. Do not use commas in numbers.
" "	Double quotes are used around strings of text
'	An apostrophe indicates the following content is a comment, not used in the calculation

Number formats

Scenario 360 formulas use "standard English" notation for numbers, with a '.' as a decimal separator and a ',' as a digit grouping symbol. If your computer has different settings, you can enter numbers in the Formula Editor according to your local conventions, but they may be displayed in standard English form. Regardless of your regional settings, you should **not** use digit grouping symbols (a comma in standard English notation) when writing numbers in formulas.

Boolean operators
Functions also include Boolean logic operators as follows:
AND (logical "and")
OR (logical "or")
NOT (logical "not")
XOR (logical "exclusive or" returns true if one but not both of the operands is true)

The Boolean functions are illustrated in the following table, using 1 for TRUE and 0 for FALSE. Each row shows one of the four possible combinations of A and B. The following columns show the result for the four Boolean logic operators.

A	B	A AND B	A OR B	A NOT B	A XOR B
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	1	1
1	1	1	1	0	0

Mathematical and logical operators

Functions include mathematical operators as follows:

+ addition	**/** division
- subtraction	**^** exponentiation – raising to a power
***** multiplication (used instead of ×)	

and logical operators as follows:

= equal to	<= less than or equal to	> greater than
<> not equal to	< less than	>= greater than or equal to
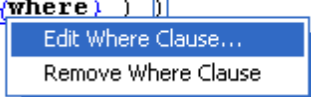
Using 'where' conditions

Where conditions let you specify one or more conditions for selecting which features to include in a calculation. For example, to count the number of buildings over four stories tall, you would write a formula to count the number of houses *where* the number of stories is greater than four.

Use Where in conjunction with other functions to mean "in the cases that…". Parts of formulas that start with Where are called *Where Clauses* or *Where Statements*. The conditions specified by where clauses are Boolean; they can be either TRUE or FALSE. Where clauses can take the form of inequalities, Boolean expressions, or Boolean functions. Conceptually (though not in formal syntax), example conditions might be Name = "Smith" or Value > 29.

When where conditions appear in a formula, you can click or right-click on the where statement and click **Edit Where Clause** or **Remove Where Clause** on the pop-up menu. If a where clause is not used, all features in the target layer are considered.

Some common functions that often include where clauses are Contains, MinDistance, and OverlapArea.

Formula syntax conversions from CommunityViz v.1.3

Type	CommunityViz 1	Scenario 360
Element References		
Conversion Factor	#Acres per Sq Foot#	[Conversion:Sq Feet to Acres]
Assumption	&Children per Household&	[Assumption:Children per Household]
Indicator	!Total Water Usage!	[Indicator:Total Water Usage]
Attribute	[Soils:Type]	[Attribute:Soils:Type]
Layer	[Soils:]	[Layer:Soils]
Table	[~LookupTable:Value]	[Attribute:LookupTable:Value]

Dynamic Attribute Formulas

	`[Shape].ReturnArea`	`Area ([Attribute:Shape])`
	`[Score1] + [Score2]`	`[Attribute:Score1] + [Attribute:Score2]`
	`([Units] / [Acres]).Round`	`Round ([Attribute:Units] / [Attribute:Acres])`
	`[Area in Sq Feet] * #Acres per Sq Foot#`	`[Attribute:Area in Sq Feet] * [Conversion:Sq Feet to Acres]`
	`[Area in Acres] * &Number of Residents per Sq Foot&`	`[Attribute:Area in Acres] * [Assumption:Number of Residents per Acre]`
	`X.UserChoice ("Select a structure type:", "Residential", "Commercial", "Other")`	`UserChoice ("Select a structure type:", "Residential", "Commercial", "Other")`

Type	CommunityViz 1	Scenario 360
	X.UserChoiceGet ([~Water Rates Table:Landuse])	UserChoiceGet ("Select a water rate", [Attribute: Water Rates Table:Landuse])
	X.UserInput ("Please enter the year this structure was built", 1999)	UserInput ("Please enter the year this structure was built", 1999)

	X.IIf ([Type] = "Residential", &Residential Water Rate&, [Type] = "Commerical", &Commercial Water Rate&, &Default Water Rate&)	IfThenElse (
		
		If ([Attribute:Type] = "Residential"), Then ([Assumption:Residential Water Rate]),
		
		If ([Attribute:Type] = "Commerical"), Then ([Assumption:Commercial Water Rate]),
		
		Else ([Assumption:Default Water Rate]))

	X.OverlapArea ([Wetlands:])	OverlapArea ([Layer:Wetlands])
	X.MinDistance ([Roads:], [Roads:Name] = "Highway 287")	MinDistance ([Layer:Roads], Where([Attribute:Roads:Name] = "Highway 287"))
	X.Count ([New Structures:], [New Structures:Type] = "Residential")	Count ([Layer:New Structures], Where([Attribute:New Structures":Type] = "Residential"))
	X.Intersects ([Wetlands:], [Wetlands:Size] > 100)	Intersects([Layer:Wetlands], Where([Attribute:Wetlands:Size] > 100))
	X.Get ([~Water Rates Table:Rate], [~Water Rates Table:Landuse] = [Landuse]) **Note**: Where Clause references Landuse attribute in host layer	Get ([Attribute:Water Rates Table:Rate], Where([Attribute:Water Rates Table:Landuse] = [Attribute:Landuse]))
	X.GetFromClosest ([Contours:Percent Slope])	GetFromClosest ([Attribute:Contours:Percent Slope)
	X.GridMax ([SlopeGrid:Value])	GridMax ([Layer:SlopeGrid])
	X.Sum ([Census Blocks:Population], Intersects ([Shape]) **Note**: Where Clause references the current shape in host layer	Sum ([Attribute:Census Blocks:Population], Where(Intersects ([Attribute:Shape])))
Type	CommunityViz 1	Scenario 360

Indicator Formulas

	X.Sum ([New Development:Water Use])	Sum ([Attribute:New Development:Water Use])
	X.Mean ([New Development:Water Use], New Development:Type] = "Commercial")	Mean ([Attribute:New Development:Water Use], Where([Attribute:New Development:Type] = "Commercial"))
	X.Count ([New Development:], [New Development:Type] = "Residential")	Count ([Layer:New Development], Where([Attribute:New Development:Type] = "Residential"))
	!New Residential Structures! * &Persons per Household&	[Indicator:New Residential Structures] * [Assumption:Persons per Household]

About functions

Indicator formulas and attribute formulas produce values based on the use of functions or formula elements. This might be written as

```
Value of Indicator (or Value of Attribute) = Function or series of functions
performed on elements
```

Functions are mathematical or spatial calculations that can range from simple addition or subtraction to complex analysis such as "standard deviation" or "overlap weighted average". Many functions can be used within one formula. Scenario 360 provides dozens of functions.

Functions, attributes, and indicators can be numeric, Boolean, or text.
Numeric components and functions use numbers
Boolean components and functions use true/false, yes/no, or 1/0
Text components and functions use words, letters, or numeric characters

Terms used in functions

Return	Functions are said to *return* a value when they are evaluated. Informally, you would say the function "gives the result…" to mean "returns."
Host	The layer or feature containing a formula.
Current	The feature whose function is being evaluated. Current and Host are used almost interchangeably.
Target	A feature or layer that is referred to in the function. In most cases the target is not the same layer as the one containing the function.
Argument	The argument of a function is the set of information upon which it operates; that is, its "inputs." In a formula, the argument is shown in parentheses after the name of the function.
Where	A condition that must be satisfied. The term is used in the logical sense, not the spatial sense.

Example

The following example formula can help illustrate these terms. A map layer called "Buildings" includes polygons representing the footprints of several buildings. The area of each building footprint is an attribute called SIZE, whose formula is

```
Area( [Attribute:Shape] )
```

Another example might be a user-variable assumption called "Roofing Unit Cost", which gives the cost of roofing per square meter of building. You could create a new attribute in the Buildings attribute table called "Building Roof Cost". Its formula could be

```
Area( [Attribute:Size] ) * [Assumption:Roofing Unit Cost]
```

In this example:
The function `Area` *returns* the number of square meters for each building.
Buildings is the *host* (*current*) layer.
The attribute called `[Attribute:Size]` is the *argument* of the function `Area`.

Now suppose you want to create an attribute called DISTANCE TO PARK in the Buildings layer that gives the distance from each building to the nearest park. You have a separate map layer called Parks. The formula for the new attribute in the Buildings layer would be

```
        MinDistance( [Layer:Parks] )
```

In this example:
The function `MinDistance` *returns* the distance to the nearest feature in the layer parks

The *host* or *current* layer is Buildings
The *target* layer is Parks
The *argument* of the function MinDistance is the layer called `[Layer:Parks]`

Functions by group

You can customize the list of functions displayed in the Formula Editor using the **Or select a group** drop-down field in the Formula Editor.

Most functions belong to more than one group. For a complete list of functions available, click the **Or select a group** drop-down field in the Formula Editor and select **All Functions**.
Group descriptions appear below.

Boolean	Tests for a condition and returns TRUE or FALSE
Common Attribute Functions	Functions of various types that are frequently used for attribute formulas
Common Indicator Functions	Functions of various types that are frequently used for indicator formulas
Conditional	Tests for a condition as part of its operation, and does not necessarily return a Boolean
Custom Scripts	Runs external Python scripts as part of its operation
Field Lookup	Finds values in a table
Grid	Numeric operations on raster or grid type layers (requires ArcGIS Spatial Analyst)
Lookup	Finds values in a table or in feature attributes
Math and Trigonometry	Performs traditional mathematical and trigonometric functions
Measurement	Calculates spatial data such as lengths or distances
Network	Performs numeric calculations on networks (requires ArcGIS Network Analyst)
Prompt	Provides an interactive window for the user during calculations
Prompt Indicator	Asks the user for inputs that affect an indicator value
Random Numbers	Generates random numbers according to particular statistical rules
Spatial Boolean	Tests for spatial conditions and returns TRUE or FALSE
Spatial Lookup	Finds values based on spatial conditions
Spatial Numeric	Performs spatial calculations and returns a number
Statistical	Performs traditional statistical functions
Text	Manipulates strings of text
User Input	Asks the user for inputs during a calculation
Variable	Generates variable results even with the same inputs; e.g., random numbers
Where	Used in conditional tests such as Where clauses

The tables below show all functions and the groups to which they belong.

FUNCTION	Boolean	Common Attribute Functions	Common Indicator Functions	Conditional	Custom Scripts	Field Lookup	Field Numeric	Grid	Lookup	Math and Trigonometry	Measurement	Network	No Indicator	Numeric	Prompt	Prompt No Indicator	Random Numbers	Spatial Boolean	Spatial Lookup	Spatial Numeric	Statistical	Text	User Input	Variable	Where
Abs										✓				✓											
Acos										✓				✓											
AngleTo		✓									✓			✓						✓					✓
Area		✓									✓			✓						✓					✓
Asin										✓				✓											
Atan										✓				✓											
AvgDistance		✓									✓			✓						✓					✓
Azimuth		✓									✓			✓						✓					✓
Ceiling										✓				✓											
Center Contains	✓	✓		✓														✓							✓
Concat																						✓			
Contains	✓	✓		✓														✓							✓
Cos										✓				✓											
Cosh										✓				✓											
Count			✓				✓			✓				✓							✓				
CustomScript					✓																				
Custom ScriptB					✓																				
Custom ScriptS					✓																	✓			
Else		✓		✓																					
Exp										✓				✓											
Floor										✓				✓											
Get		✓				✓			✓																
GetFrom Closest		✓							✓										✓						✓
GridMax								✓																	
GridMean								✓																	
GridMin								✓																	
GridMost								✓																	
GridOverlap								✓																	
If		✓		✓																					
IfError				✓																					
IfThenElse		✓		✓																					
Intersects	✓	✓		✓														✓							✓
IsCenter ContainedIn	✓	✓		✓														✓							✓
IsContainedIn	✓	✓		✓														✓							✓
IsInfinity				✓										✓											
IsNull				✓																					
IsSelected	✓	✓		✓														✓							✓
Left																						✓			

39

FUNCTION	Boolean	Common Attribute Functions	Common Indicator Functions	Conditional	Custom Scripts	Field Lookup	Field Numeric	Grid	Lookup	Math and Trigonometry	Measurement	Network	No Indicator	Numeric	Prompt	Prompt No Indicator	Random Numbers	Spatial Boolean	Spatial Lookup	Spatial Numeric	Statistical	Text	User Input	Variable	Where
Length		✓									✓			✓						✓					✓
Ln										✓				✓											
Log										✓				✓											
Log10										✓				✓											
Max			✓				✓			✓				✓							✓				
MaxDistance		✓									✓			✓						✓					✓
Mean			✓				✓			✓				✓							✓				
Median			✓				✓			✓				✓							✓				
Middle																						✓			
Min			✓				✓			✓				✓							✓				
MinDistance		✓									✓			✓						✓					✓
NetworkGetFromClosest									✓			✓							✓						
NetworkMinDistance											✓	✓								✓					
Norm									✓					✓											
OverlapArea		✓									✓			✓						✓					✓
OverlapLength		✓									✓			✓						✓					✓
OverlapMost									✓					✓					✓						✓
OverlapSum									✓					✓					✓						✓
OverlapWeightedAvg		✓							✓					✓					✓						✓
ProximityAvg									✓					✓					✓	✓					✓
ProximityCount		✓												✓						✓					✓
ProximitySum		✓					✓							✓					✓	✓					✓
ProximityWeightedAvg									✓					✓					✓	✓					✓
ProximityWeightedSum									✓					✓					✓	✓					✓
Rand														✓			✓							✓	
RandG														✓			✓							✓	
RandI														✓			✓							✓	
Right																						✓			
Round										✓				✓											
Sin										✓				✓											
Sinh										✓				✓											
Sqrt										✓				✓											
StdDev			✓				✓			✓				✓							✓				
Sum			✓				✓			✓				✓							✓				
T1F0				✓																					
Tan										✓				✓											
Tanh										✓				✓											

FUNCTION	Boolean	Common Attribute Functions	Common Indicator Functions	Conditional	Custom Scripts	Field Lookup	Field Numeric	Grid	Lookup	Math and Trigonometry	Measurement	Network	No Indicator	Numeric	Prompt	Prompt No Indicator	Random Numbers	Spatial Boolean	Spatial Lookup	Spatial Numeric	Statistical	Text	User Input	Variable	Where
Then		✓		✓																					
ToNumber																						✓			
ToString																						✓			
Trim																						✓			
Truncate										✓				✓											
UserChoice		✓											✓		✓								✓		
UserChoice Get		✓											✓		✓								✓		
UserInput		✓											✓	✓	✓								✓		
UserInputB	✓	✓											✓		✓								✓		
UserInputS		✓														✓						✓	✓		
Var			✓				✓			✓				✓							✓				
Weighted Median							✓			✓				✓							✓				
Where		✓		✓																					

Shape types allowed

Spatial Numeric and Spatial Boolean functions typically (but do not always) allow any combination of point, line, and polygon shape types between the current layer and target layer. However, the values returned in these cases may differ from the name of the function. For example, `OverlapArea` returns an area when a polygon current layer is overlapping a polygon target layer, but it returns a length when the target layer consists of lines and a count when the target layer is points. Consult the full formula descriptions for details. Grid functions all require grid (raster) target layers, and Network functions all require network current layers.

Tip It is important to use consistent units throughout your formula. For example, if your lot sizes are in meters but your tax rates are per hectare, you need to use a conversion factor to produce a reasonable result. The Formula Wizard automatically checks units and prompts you for conversion factors, but it is prudent to watch for strange units or combinations of units that may have "fooled" the Wizard.

Tip Very long formulas can be hard to read and hard to check. It is good practice to break up large formulas into smaller ones by doing smaller parts of the calculation and then combine them together. Lookup tables are another good way to shorten formulas.

Function library

Abs (absolute value) function

The `Abs` function is a **Number Request** that returns the absolute value of a number (removes the - sign from negative numbers).

The absolute value of -4 is 4; the absolute value of 4 is 4.

Formula syntax
When you add the `Abs` function to a formula, the program will display the following syntax:

    Abs( )

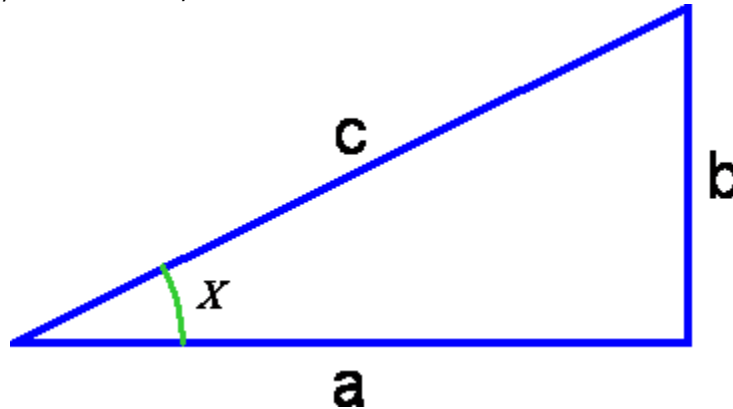To complete the function, enter a number or numeric expression in the parentheses.

Type of value returned
Numeric

Acos (arc cosine) function

The `Acos` function is a **Number Request** that calculates the value (in radians) of the angle whose cosine equals a specified number (the inverse of the cosine function).

In the diagram below, the `Acos` of a/c is X. The units of X are radians. `Acos` is sometimes written cos^{-1}.

Tip To convert X from radians into degrees, multiply it by 180/Pi. To get the value of Pi, type "Pi" in the formula.

Formula syntax
When you add the `Acos` function to a formula, the program will display the following syntax:

    Acos ( )

To complete the function, enter a value in radians in the parentheses.

Type of value returned
Numeric

AngleTo (angle to) function

The `AngleTo` function generates a value between 0 and 360 (degrees) corresponding to the direction of the nearest feature in a specified layer. The `AngleTo` function is a **Spatial Numeric** measurement that calculates the direction from the current feature to the nearest feature in the target layer. Results are provided in degrees.

Direction is measured from the center of the current feature to the nearest point on the nearest feature in the target layer. (For more information about "nearest," see the MinDistance function.) Optionally, you can place conditions on which features of the target layer to include in the calculation. If the current feature overlaps a feature in the target layer (that is not excluded by the where clause), `AngleTo` returns 0 as the result.

The output is a floating point number between 0 and 360. The geographic convention is used, which means 0 is grid north and the scale increases clockwise to 360, as illustrated below.
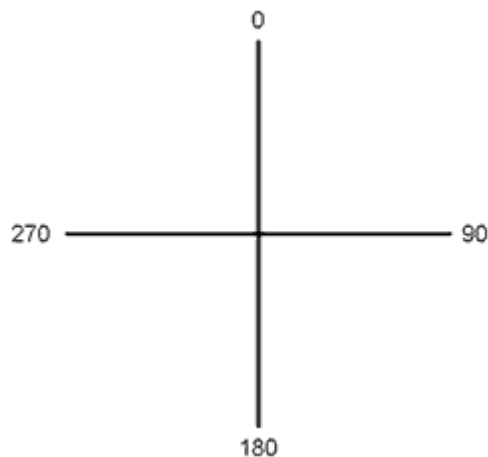
Sample applications of this formula
- Create an orientation attribute for use in 3D scenes. This can be used to orient houses to face the nearest road, or orient flat photos of distance scenery to face a scenic viewpoint.
- Calculate which side of a road or river a feature is on.

Formula syntax
When you add the `AngleTo` function to a formula, the program will display the following syntax:

```
AngleTo( {target},
Where( {where} ))
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. If you are not using the **where** clause you can delete it by right-clicking on it and clicking **Remove Where Clause** on the pop-up menu. After clicking on *{target}*, the program will display a selection box with available target layers.

For example, if a house's nearest road is due east of the house, there could be an attribute in the House layer called *DirectionToRoad* with the formula:

```
AngleTo( [ Layer:Roads ] )
```

For that particular house, the result would be 90.

Tip: An easy way to orient 3D houses in Scenario 3D so that they face the road is to create a DirectionToRoad (or similarly named) attribute for the buildings layer. In Scenario 3D 3D Scene Settings > Scenarios & Layers > 3D Layer Settings screen under Options, set Orientation to "Use this field" and choose the DirectionToRoad attribute.

Current layer shape type
Point, Line, or Polygon

Target shape type
Point, Line, or Polygon

Type of value returned
Numeric

Area function

The `Area` function calculates the flat map area of a shape. The program will display the results in map units.

`Area` is a formula function that calculates the area of a shape. The area is returned in square map units, as defined for the analysis. The area measured is that of a flat map; no topological irregularities are considered. To find the surface area of a three-dimensional terrain, use the 3D Analyst extension in ArcGIS™.

Area is a common function used in many calculations.

Sample applications of this formula
- Calculating the acreage of irregular lots to estimate tax revenue
- Calculating the area of a buildings footprint as part of a water permeability analysis
- Calculating the amount of farmland dedicated to particular crops

Formula syntax
When you add the `Area` function to a formula, the program will display the following syntax:

```
Area( [Attribute:Shape] )
```

Normally you do not need to take any further steps. However, if desired you can change the `Attribute` by right-clicking on it.

A typical application of the `Area` syntax would be to find the area of a parcel. The syntax for this would appear as:

```
Area( [ Attribute:Shape ]) x [ Conversion:Sq Meters to Hectares ]
```

This example assumes that the map units are meters, and that the desired answer is area in hectares. Therefore, it uses a conversion factor to convert square meters to hectares.

WARNING! Some imported data layers, particularly those imported from coverages, may already have existing attributes called "AREA" or "LENGTH." However, these will not be dynamic attributes and will not recalculate if the layer is edited. It is recommended you create new, Scenario 360-based dynamic attributes for `Area` and `Length` if you plan to use them in Scenario 360 formulas. You may want to consider deleting the old AREA and LENGTH fields to avoid confusion. (Also, geodatabases will automatically create attributes called `SHAPE_Area` and `SHAPE_Length`. While these attributes do

update dynamically based on editing, they may or may not use the same units as the rest of your formula.

Current layer shape type
Polygon

Target shape type
N/A

Type of value returned
Numeric

Asin (arc sine) function

The `Asin` function calculates the value (in radians) of the angle whose sine equals a specified number (the inverse of the Sin function).

In the diagram below, the `Asin` of b/c is X. The units of X are radians. `Asin` is sometimes written $\sin^{-1}$.

Tip To convert X from radians into degrees, multiply it by 180/Pi. To get the value of Pi, type "Pi" in the formula.

Formula syntax
When you add the `Asin` function to a formula, the program will display the following syntax:

```
Asin( )
```

To complete the function, enter a value in radians in the parentheses.
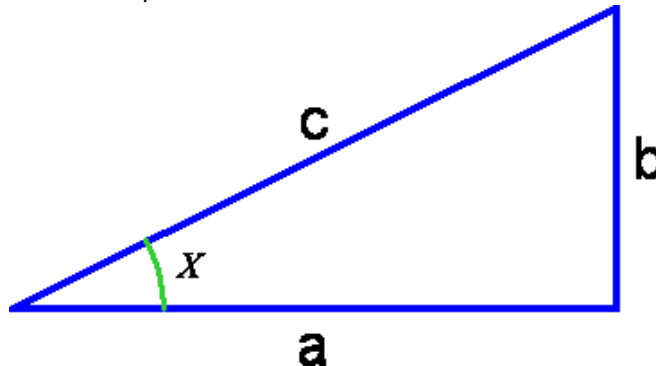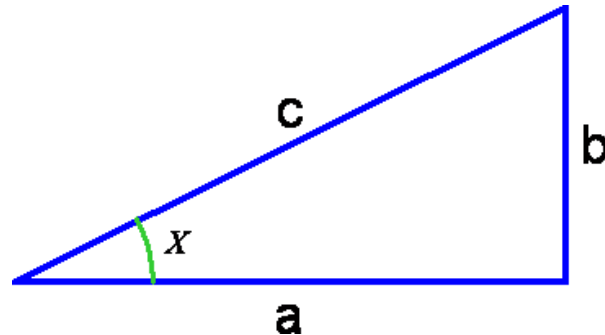
Type of value returned
Numeric

Atan (arc tangent) function

The `Atan` function calculates the value (in radians) of the angle whose tangent equals a specified number (the inverse of the Tan function).

In the diagram below, the arc tangent of b/a is X. The units of X are radians. Arc tangent is sometimes written $\tan^{-1}$.

Tip To convert X from radians into degrees, multiply it by180/Pi. To get the value of Pi, type "Pi" in the formula.

Tip `Tan` and `Atan` are useful for converting between degrees of slope and percentage of slope or "rise over run." Usually a "25% slope" means that b/a = 0.25 in the figure above. The corresponding degrees of slope would be `Atan(b/a)` converted into degrees, or `Atan(b/a) *180/Pi`. If `b/a` = 25%, then X = 14 degrees. (Be aware, however, that sometimes people use the term percent slope to mean `b/c`. If so, use the Sin and Asin functions for conversions.)

To convert degrees of slope into percentage of slope, use Slope-in-percent = `Tan(Slope-in-degrees * Pi/180)`.

To convert slope in percentage to slope in degrees, use Slope-in-degrees = `Atan(slope-in-percent)*180/Pi`.

Formula syntax
When you add the `Atan` function to a formula, the program will display the following syntax:

    Atan( )

To complete the function, enter a value in radians in the parentheses.

Type of value returned
Numeric

AvgDistance (Average Distance) function

The `AvgDistance` function identifies the average straight-line distance from the current feature in a layer to all the features in the target layer. It uses the same distance-measuring rules as `MinDistance`. Topological variations such as hills and valleys will not affect the formula. The program displays the result as a number in map units.

The `AvgDistance` function measures distances to other features, even if those features are in another layer. It uses straight-line distances.

`AvgDistance` calculates the shortest distance to the nearest edge of each feature the target layer. Target features can be lines or points as well as polygons.

When working with polygons, `AvgDistance` measures from the nearest point on the edge of the polygon. If working with lines, it measures from the nearest point on the line. If a target shape intersects the host shape, the function returns 0.

For more information on how distances are calculated by `AvgDistance`, refer to `MinDistance`.

Sample applications of this formula
- Rate many possible tornado siren sites based on their average distance to nearby buildings.
- Estimate the "wilderness factor" of parcels based on their average distance to roads.

Formula syntax
When you add the `AvgDistance` function to a formula, the program will display the following syntax:

```
AvgDistance( {target},
Where( {where} ))
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. The where clause applies to the target layer, not the current layer. After clicking on *{target}*, the program will display a selection box with available target layers.

For example, you may wish to assign a new attribute to a "lakes" layer that gives each lake's average distance to residential building. The formula for this problem would look like this:

```
AvgDistance( [ Layer:Buildings ],
Where( [ Attribute:Buildings:Type ] = "Residential" ))
```

Current layer shape type
Line, Point, or Polygon

Target shape type
Line, Point, or Polygon

Type of value returned
Numeric

Azimuth function

The `Azimuth` function generates a value between 0 and 359 (degrees) that gives the orientation or angle of rotation of a feature. The `Azimuth` function is a **Spatial Numeric** measurement.

`Azimuth` is measured as the degrees of clockwise rotation from the positive y axis. In other words, the azimuth for a line pointing straight up is 0, a line pointing straight right is 90, a line pointing straight down is 180, a line pointing straight left is 270, etc.
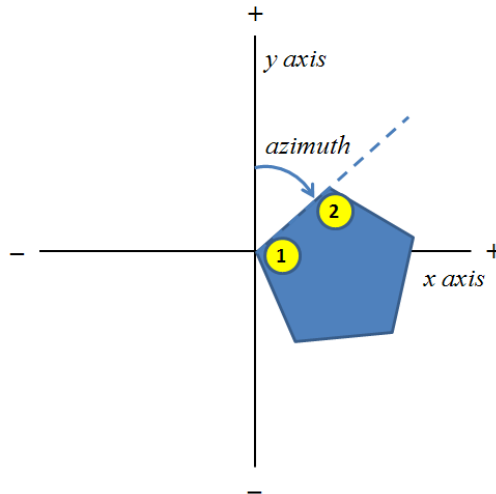
Synonyms and related terms for azimuth include rotation, compass, orientation, direction, turn, twist, north, south, east, west, angle, attitude, elevation, front, and exposure.

For points, `Azimuth` always returns -1. This can be interpreted as "false" and makes sense because points do not have a rotation.

For lines, it is important to know that ArcGIS stores a "beginning point" and an "ending point" (or "first vertex" and "second vertex") for line segments. Azimuth is measured as if the beginning point (first vertex) were at the origin of the x-y coordinates.

For polylines, the first line segment is the only one measured. For multipart lines, the first segment of the first part is used.

For polygons, it is important to know that ArcGIS stores the order (sequence) of segments (edges). The first segment (usually the edge that was drawn first) is used for azimuth measurements. ArcGIS always assumes polygons are drawn clockwise. The azimuth is measured as if the first vertex of the first segment were at the origin of the x-y coordinates. The azimuth measurement for multipart polygons is based on the first segment of the first part.

Use caution when measuring polygon donuts (also known as a torus or a polygon with a hole). The `Azimuth` function will always measure the first segment, but the segment sequence may not always be what you expect, depending on how the shape was created.

If the first segment of a line or polygon is an arc, the measurement is based on an imaginary line connecting the From and To (starting and ending) points of the arc.

The output is a floating point number between 0 and 360 (excluding 360 itself). Fractional degrees are given in decimals, not minutes and seconds.

Tip One way in ArcMap to find the first vertex of a particular line or polygon is to start editing, select the feature of interest, and choose Modify Feature. Open Sketch Properties from the button on the Editor toolbar and highlight the first part (probably numbered 0). A white square will appear on the first vertex of the associated segment.

Tip To use ArcMap to symbolize an entire layer of polygons so that the first vertex of each one is marked, first open the Symbol Selector by double-clicking on the symbol in the table of contents or on

the Symbology tab in the Layer Properties. Choose Properties… > Outline… > Properties…, and under Type choose Cartographic Line Symbol. Click the Line Properties tab and specify a left-facing arrow as the Line Decoration. Click OK several times to accept your changes. This will create an arrow pointing at the first vertex of each polygon in the layer.

Sample applications of this formula

- Create an orientation attribute for use in 3D scenes. This is most commonly used in conjunction with the "Use centroid" option in the Substitution Method for polygon layers in 3D modeling tools like Scenario 3D. To give a 3D model being placed on the centroid the same orientation as the original polygon, create an attribute called "Azimuth" in the polygon layer and then use it as the Orientation field in Scenario 3D Layer Settings.

- Calculate the solar exposure of building footprints. For this application, the footprints must have been drawn in a consistent way, such as front door first.

Tip Azimuth is related to the concepts of "front" and "elevation" in architectural drawings, but not the same. If a building's front door faces north (that is, if you are looking north as you stand inside and look out the front door), the azimuth of its front wall will be 90. If a building's front door faces south, the front wall's azimuth will be 270. Architectural convention is that the front door faces south, but the orientation of a given 3D model is set when the model is created, and not all 3D building models have their front pointing the same direction.

Formula syntax
When you add the Azimuth function to an attribute formula, the program will display the following syntax:

```
Azimuth( [Attribute:Shape] )
```

 Normally you don't need to do anything else to complete the function. Where clauses are not available for Azimuth.

 For example, if you are making a 3D scene of a city block and you have a polygon layer representing the footprints of the buildings and 3D models of each building, you can make an orientation field that Scenario 3D can use by creating a dynamic attribute called Azimuth in the footprints layer using the formula

```
Azimuth( [ Attribute:Shape ] )
```

When you are ready to set up the 3D scene, check the "Use centroid" box in Scenario 3D 3D Scene Settings > Scenarios & Layers > 3D Layer Settings for the footprints layer, and under Options > Orientation > Use this field, choose "Azimuth."

Current layer shape type
Point, Line, or Polygon

Target shape type
Point, Line, or Polygon

Type of value returned
Numeric

Ceiling function

The `Ceiling` function rounds a number up to the nearest whole number.

Ceiling rounds numbers like 4.3 up to the nearest whole number, in this case 5. If the number being rounded is negative, ceiling rounds "up" to a larger negative number, so that − 4.3 becomes − 4.

Formula syntax
When you add the `Ceiling` function to a formula, the program will display the following syntax:

    Ceiling( )

To complete the function, enter a number in the parentheses.

Type of value returned
Numeric

CenterContains (Center Contains) function

The `CenterContains` function determines whether the current shape contains the center of <u>any</u> features within the target layer.

`CenterContains` looks to see if the current shape includes the geometric center of one or more shapes in the target layer. Another name for this function might be, "Contains the center of."

In the diagram below, the current shape is a hexagon and the ellipses are features in the target layer. Each ellipse is labeled either True (indicating its center *is* contained in the current shape) or False (indicated its center is *not* contained in the current shape). `CenterContains` returns True if there are one or more shapes in the target layer with their center contained in the current shape. Target features may be points or lines as well as polygons.
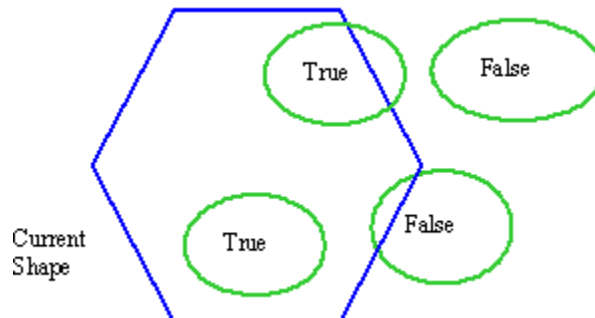
Note: The geometric center of an irregular shape may not be contained in the perimeter of the shape. The center of a shape is the center of its extent; that is, the center of the rectangle containing the shape. Refer to the example below.
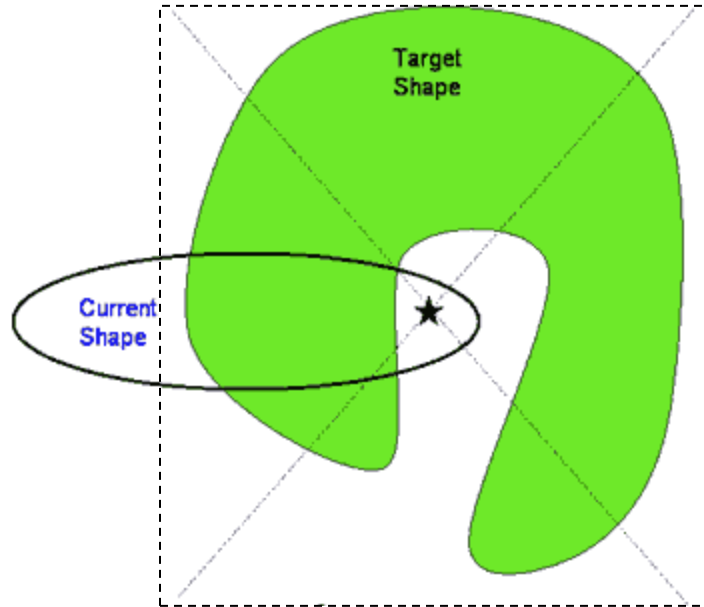
The star represents the geographic center of the irregular shape and falls within the current shape and as a result returns TRUE. Therefore, `CenterContains` should be used with caution if the target layer contains irregular shapes.

Sample applications of this formula

- Deciding whether a parcel is "in" the floodplain or not. `CenterContains` will return False (indicating *not* in the floodplain) if, for example, just one corner of the parcel is in the floodplain.
- Calculating how many lots are within a proposed new school district boundary. `CenterContains` can be used to decide whether a given lot is either in or out of the district.
- Part of a suitability analysis rating the appropriateness of parcels for building based upon whether they are "on" steep slopes.

Formula syntax

When you add the `CenterContains` function to a formula, the program will display the following syntax:

```
CenterContains( {target},
    Where( {where} ) )
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

A typical application using this function might be to find out whether the floodplain covers the center of a parcel zoned for light industry. The sample syntax would appear similar to the following:

```
CenterContains( [ Layer:Parcel Zoning ],
    Where(  [ Attribute:Parcel Zoning:ZONE_TYPE ] = "Industrial Light" )
    )
```

Current layer shape type

Polygon

Target shape type

Polygon

Type of value returned
Boolean (TRUE or FALSE)

Concat (Concatenate) function

The Concat function joins several strings into one text string.

Sample applications of this formula
- Combining "Block" and "Lot" codes into a single "Parcel" designation
- Combining first and last names into a single name field

Formula syntax
When you add the Concat function to a formula, the program will display the following syntax:
Concat( )

To complete the function, enter two or more text strings in the parentheses. Enter text strings separated by commas. For example, Concat([Attribute:Zone], [Attribute:Building Type]) might return values like "CommercialStore" and "ResidentialRanch". The formula accepts numbers (like [Indicator:Floors]) but treats them as text, including any decimal points. You can have multiple entries. Concatenated items do not have any spaces between them.

Type of value returned
String

Contains function

Contains determines whether a shape *completely* contains any features within the target layer. It returns a value of either TRUE or FALSE.

Contains looks to see if the current shape includes the entirety of one or more features (lines, points, or polygons) in the target layer.

In the diagram below, the current shape is a hexagon and the ellipses are features in the target layer. The ellipse labeled True is *entirely* contained within the current shape. The ellipses labeled False are not. The target feature may share a boundary with the current layer and still return True. `Contains` returns True if there are one or more shapes in the target layer are completely contained in the host layer. Target features can also be lines or points.
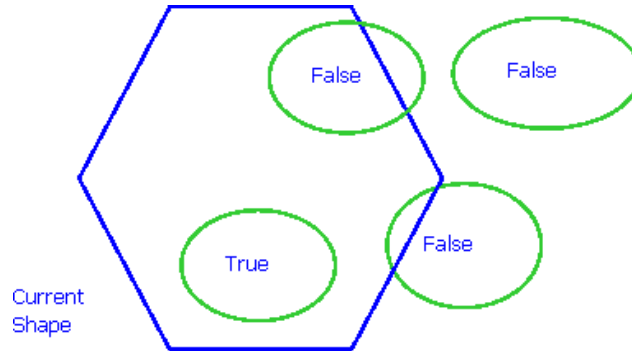
Tip `Contains` is most often used to check whether points (trees, buildings) are contained within a particular area, but it can also be used on lines and polygons.

Sample applications of this formula
- Deciding whether a construction site is within an appropriately zoned parcel
- Part of an analysis counting trees is in a specified vegetation treatment area
- Testing whether a proposed conservation area completely includes an endangered species habitat

Formula syntax
When you add the `Contains` function to a formula, the program will display the following syntax:

```
Contains( {target},
    Where( {where} ) )
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

As an example, the `Contains` function might be used to create an attribute that specifies whether a ranch contains a spring with flow of greater than 10 gallons per minute. The syntax would appear as:

```
Contains( [ Layer:Springs ],
     Where(  [ Attribute:Springs:FLOW ] >= 10 ) )
```

Current layer shape type
Polygon

Target shape type
Polygon, Line, or Point

Type of value returned
Boolean (TRUE or FALSE)

Cos (Cosine) function

The `Cos` function calculates the cosine of an angle (the length of the adjacent side).

In the diagram below, the cosine of X is given by the ratio a/c. The units of X are assumed to be radians. Radians equal the ratio of an arc of a circle to the radius of the circle.
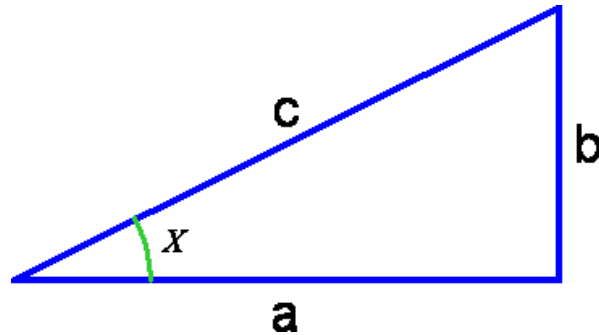
If X is in degrees, multiply it by Pi/180 to convert it to radians.

Formula syntax
When you add the `Cos` function to a formula, the program will display the following syntax:

    Cos( )

To complete the function, enter a value in radians in the parentheses.

Type of value returned
Numeric

Cosh (Hyperbolic Cosine) function

The `Cosh` function calculates the hyperbolic cosine of the angle.

The hyperbolic cosine of x is $(e^x + e^{-x})/2$. It is usually written as `Cosh`. The units of x are assumed to be radians.

Tip If x is in degrees, multiply it by Pi/180 to convert it to radians. To get the value of Pi, type "Pi" in the formula.

Formula syntax
When you add the `Cosh` function to a formula, the program will display the following syntax:

    Cosh( )

To complete the function, enter a value in radians in the parentheses.

Type of value returned
Numeric

Count function

`Count` tallies the number of records found in the target layer that satisfies the condition specified in a `where` statement.

Use this function to count features. `Count` works on a specified target layer and optionally includes a `where` clause that specifies characteristics of features that are to be counted.

Sample applications of this formula
- Counting the number of residential buildings in a buildings layer
- Counting number of parking lots larger than 100,000 square feet
- Counting number of bus stops near an affordable housing development

Formula syntax
When you add the `Count` function to a formula, the program will display the following syntax:
```
Count( {target},
    Where( {where} ) )
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

As an example, you may use this function to count all the streets with "Peachtree" in their name. The syntax would appear as follows:
```
Count( [ Layer:Street ],
    Where(  [ Attribute:Street:NAME] = "Peachtree") )
```

Type of value returned
Numeric

CustomScript (Custom Script) functions

The `CustomScript` function runs an ArcGIS custom script tools and returns a number. `CustomScriptB` and `CustomScriptS` use the same syntax as `CustomScript`, but `CustomScriptB` returns a Boolean value and `CustomScriptS` returns a text (string) value.

Formula syntax
When you add the `CustomScript` functions to a formula, the program will display the following syntax:
```
CustomScript( {script} )
```

To complete the function:
1. Click on the {script} link to open the Select Custom Script form.
2. Click the Select Custom Toolbox button, navigate to your custom toolbox and click Open.
3. A list of all the available tools will appear under the Script Name column as well as the required input parameters under the Arguments column – multiple parameters are separated by commas.
4. Double-click or simply select the tool in the list and click OK.
5. The script will be added to the Formula Editor in the following format:

> CustomScript ("C:\CVFiles\CVCustomTools.tbx\MyCustomScript")

Type in a comma after the script name (and after the quotation marks), then the value of each input parameter in order, separated by commas. Text or string parameters must be in quotation marks. After you add the parameter(s) to complete the formula, it will look something like this:
CustomScript ("C:\CVFiles\CVCustomTools.tbx\MyCustomScript", [Layer: Buildings], "Tower", 17)

Inputs can be constants, layers, attributes, assumptions, or indicators. The script's output must be a single number.

Type of value returned
Numeric

Else function

The conditional `Else` statement specifies the value to return within an `IfThenElse` formula if no condition is found to be true.

For more information, see **IfThenElse**

Exp (Natural Exponential) function

The `Exp` function calculates "e" raised to the power of a specified number.

The natural exponential `Exp` of x is usually written e^x. It is the inverse function of the natural logarithm, Ln. In other words, if $e^x = y$, then `Ln` $y = x$. The value of e is approximately 2.71828 and is available by typing e in the formula.

When you add the `Exp` function to a formula, the program will display the following syntax:
Exp( )

To complete the function, enter a number in the parentheses.

Type of value returned
Numeric

Floor function

The `Floor` rounds a number down to the nearest whole number.

`Floor` rounds numbers like 4.7 down to the nearest whole number, in this case 4. If the number being rounded is negative, floor rounds "down" to a smaller negative number, so that – 4.7 becomes – 5.

Formula syntax
When you add the `Floor` function to a formula, the program will display the following syntax:
Floor( )

To complete the function, enter a number in the parentheses.

Type of value returned
Numeric

Get function

`Get` returns the first value found in the target layer attribute table that satisfies the condition specified in a `where` statement.

This function is a lookup function.

Sample applications of this formula
```
Get( [ Attribute: ],
    Where(  [ Attribute:] =) )
```

Formula syntax
When you add the `Get` function to a formula, the program will display the following syntax:
```
Get( {attribute},
    Where( {where} ) )
```

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

Type of value returned
Field Lookup

GetFromClosest (Get From Closest) function

This lookup function retrieves an attribute value, such as a name or size, from the nearest feature in the target layer. If more than one feature in the target layer overlaps the current shape, the function will get the value from the feature with the most overlapping area or length. In the case of a point target layer, minimum distance to center is used.

The `GetFromClosest` function determines the closest feature to the current shape in the current layer by using the same logic as expressed in the MinDistance (Minimum Distance) formula. This logic uses straight line measurement and ignores the effects of terrain.

If a feature in the target layer overlaps the current shape, that feature is considered the closest. If two or more features overlap the control shape, the formula will determine the feature with the largest amount of overlap and designate that feature as the closest.

Once the formula determines the closest feature, it will look up and return the value of the specified attribute for that feature.

Sample applications of the formula
1. Finding the name of the nearest school
2. Determining the size or type of the nearest road
3. Locating the nearest state-owned park

Formula syntax
When you add the `GetFromClosest` function to a formula, the program will display the following syntax:
```
GetFromClosest( {attribute},
    Where( {where} ) )
```

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

Type of value returned
This formula will return the same type of value as the target attribute.

GridMax (Grid Maximum) function

`GridMax` calculates the maximum value among all grid cells that have centers overlapped by the current shape.

`GridMax` measures the area value of all cells with centers overlapped by the current shape. The function calculates which cell with a center overlapped by the current shape has the highest attribute value and returns that value.

In the example provided below, each cell is identified by a letter and has an attribute value represented by a number. The current shape is the polygon overlapping the centers of eight grid cells – B, C, E, F, G, J, K, and O. GridMax would determine that grid F has the highest value and therefore return the value 28.
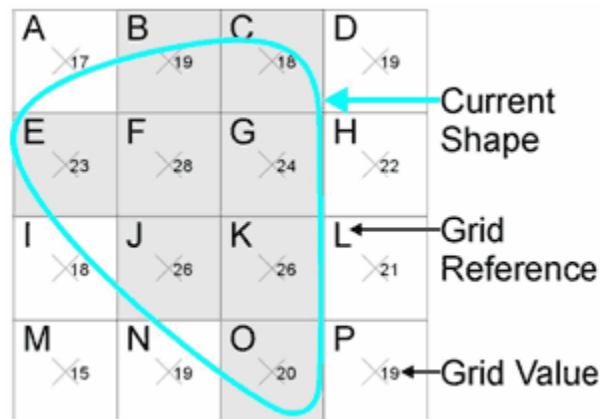
Sample applications of this formula
- Determine steepest slope on a parcel.
- Determine the highest altitude in a county.

Formula syntax
When you add the `GridMax` function to a formula, the program will display the following syntax:

    GridMax({layer})

To complete the function, specify a target layer by right-clicking on *{target}*. After clicking on *{target}*, the program will display a selection box with available target layers.

Only grid (raster) layers are allowed as the target of a grid function.

Type of value returned

GridMean (Grid Mean) function

`GridMean` calculates the average value among all grid cells overlapped by the current shape.

`GridMean` measures the area value of all cells overlapped by the current shape. The function calculates the average value of all cells overlapped by the current shape and returns that value.

In the example provided below, each cell is identified by a letter and has an attribute value represented by numbers. The current shape is the polygon overlapping the centers of eight grid cells – B, C, E, F, G, J, K, and O. GridMean would total the values of the attribute for all cells with centers overlapped by the current shape. The program would then divide the total by eight (number of grid centers overlapped) to determine the overlap average.

Sample applications of this formula
- Determine average slope of a parcel.

Formula syntax
When you add the `GridMean` function to a formula, the program will display the following syntax:

        GridMean(*{layer}*)

To complete the function, specify a target layer by right-clicking on *{target}*. After clicking on *{target}*, the program will display a selection box with available target layers.

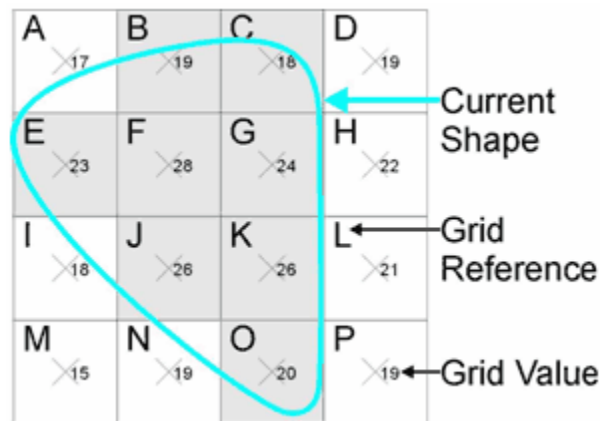Only grid (raster) layers are allowed as the target of a grid function.

Type of value returned
Numeric

GridMin (Grid Minimum) function

`GridMin` calculates the minimum value among all grid cells overlapped by the current shape.

`GridMin` measures the area value of all cells overlapped by the current shape. The function calculates which cell overlapped by the current shape has the lowest attribute value and returns that value.

In the example provided below, each cell is identified by a letter and has an attribute value represented by numbers. The current shape is the polygon overlapping the centers of eight grid cells – B, C, E, F, G, J, K, and O. GridMin would determine that grid C has the lowest value and therefore return the value 18.

Sample applications of this formula
- Determine lowest point on a parcel.
- Determine the lowest altitude in a county.

Formula syntax
When you add the `GridMin` function to a formula, the program will display the following syntax:

    GridMin({layer})

To complete the function, specify a target layer by right-clicking on *{target}*. After clicking on *{target}*, the program will display a selection box with available target layers.

Only grid (raster) layers are allowed as the target of a grid function.
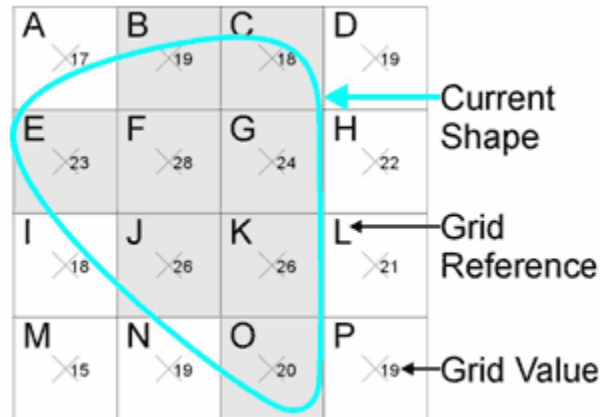
Type of value returned
Numeric

GridMost (Grid Most) function

`GridMost` calculates the value occurring most among all grid cells overlapped by the current shape.

In the example provided below, each cell is identified by a letter and has an attribute value represented by numbers. The current shape is the polygon overlapping the centers of eight grid cells – B, C, E, F, G, J, K, and O. The attribute values are 18, 19, 20, 23, 24, 26, 26, 28. GridMost would return 26.
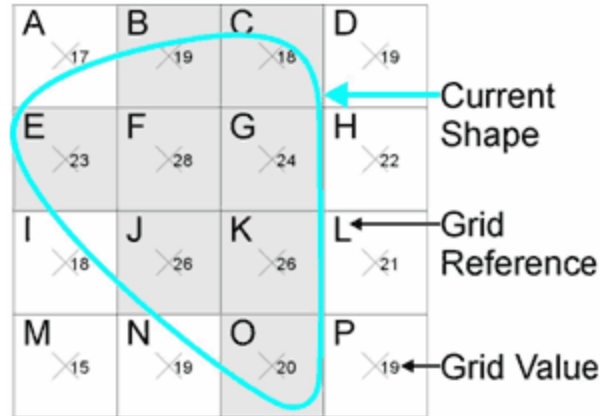
Sample applications of this formula

- Find the median slope in a parcel.

Formula syntax

When you add the `GridMost` function to a formula, the program will display the following syntax:

```
GridMost( {layer} )
```

To complete the function, specify a target layer by right-clicking on *{target}*. After clicking on *{target}*, the program will display a selection box with available target layers.

Only grid (raster) layers are allowed as the target of a grid function. This function is only valud for integer grid layers.

Type of value returned

Numeric

GridOverlap (Grid Overlap Area) function

`GridOverlap` measures the area value of all grids overlapped by the current shape. If the current shape is a polygon, the function will count the cells that overlap the current shape and then multiply by the area of a cell. For lines, the program multiplies the number of overlapped cells by the width of the cell. For points, the function returns 1 or 0 (true or false).

In the example provided below, each cell is identified by a letter and has an attribute value represented by numbers. The current shape is the polygon overlapping eight grid cells – B, C, E, F, G, J, K, and O.
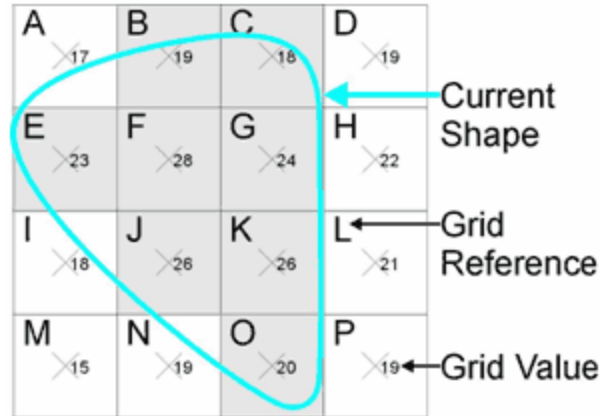
Formula syntax

When you add the `GridOverlap` function to a formula, the program will display the following syntax:

    GridOverlap({layer})

To complete the function, specify a target layer by right-clicking on *{target}*. After clicking on *{target}*, the program will display a selection box with available target layers.

Optionally, you may add a Where condition to the function that causes it to measure overlap only with cells that have a particular value. To add a Where condition, place a comma after the target layer name and then type the value you want to use. For example, to measure overlap with cells whose value is 100, write:

    GridOverlap([Layer:ExampleRasterLayerName], 100)

Only grid (raster) layers are allowed as the target of a grid function. The `GridOverlap` function is only valid for integer grid layers.

## Type of value returned
Numeric or true/false

If (If...Then) function

Use the `If` statement to conduct a conditional test on values and formulas. The function will return the specified value if the condition evaluates to TRUE.

In English, an If statement might read:
    "If it is true that _____, then return the following answer: _____."

The first blank is the conditional text, and the second blank is the specified value. Conditional tests can be any Boolean equation that returns True or False. In the case that the condition is *not* true, `If` returns the value specified after "else."

In some cases it may be more appropriate to use a conditional where clause rather than an `If` function.

Tip To test more than one condition at once, use `IfThenElse`.

Sample applications of this formula
- Marking a list according to which items are larger than a maximum allowable value
- Assigning a rating to a feature based on a test of its properties

Formula syntax
When you add the `If` function to a formula, the program will display the following syntax:
```
If(    ,
    Then( ) ,
    Else (  ))
```

To complete the function, specify a condition after `If (`, specify a value to return if the condition is TRUE after `Then (`, and specify a value to return if the condition is FALSE after `Else (`.

For example, you would use `If` to create a new attribute in a parcels layer that is similar to an existing "Buildable Area" attribute but has a value of 0 if the slope is too steep. Write this formula as:
```
If( [Attribute:Slope] < 25% ,
    Then([Attribute:BuildableArea]),
    Else (0))
```

Type of value returned
Any value

See also: IfThenElse (if...then...else) function

IfError (If Error) function

`IfError` protects numeric formulas from returning invalid results including "Infinity," undefined values, errors, nulls, and '1.#NF00e+000'. If its input formula would otherwise return one of those results, IfError returns a user-specified default value. Otherwise, it returns the ordinary formula results.

The most common application is protecting against undefined values resulting from dividing by zero, but this function can also be used to guard against, or produce special results for, log of zero, null inputs, or other potential difficulties. Writing a formula using the `IfError` function is often shorter than writing out the equivalent If/Then statement.

Example
Suppose `[Indicator:A] = 200/([Indicator:B] – [Indicator:C])`. In the case that `[Indicator:B] = [Indicator:C]`, the value of `[Indicator:A]` is undefined and will return 'IsInfinity' in most cases. Often, analysts want to return a different value, so they write a formula such as:
`If([Indicator:B] = [Indicator:C], Then (200), Else (200/([Indicator:B] – [Indicator:C])))`
The same result can be achieved using the `IfError` function as follows:
`IfError ((200/([Indicator:B] – [Indicator:C])), 200)`

Notes
- Choose the default output value with care. There is a risk of masking true problems or errors that you would want to address if you were aware of them. If the goal is to guard against unexpected results, some analysts prefer using an unusual number such as 99999999.
- The default output value may be a dynamic component such as an indicator or assumption.

Formula syntax

When you add the `IfError` function to a formula, the program will display the following syntax:

<div align="center">IfError(, -1)</div>

To complete the function:
- Specify an input function or formula just before the comma using the Formula Editor.
- Optionally, change the default output from -1 to your desired value or an analysis component.

Type of value returned

Boolean, Text, or Numeric

IfThenElse (If…Then…Else) function

Use `IfThenElse` to conduct multiple conditional tests on values and formulas. The function returns the specified value if one of the conditions evaluates to TRUE. If no condition is TRUE, then the formula returns the value for Else.

In English, an `IfThenElse` statement might read:

> If {Condition A} is true, then return {Value A}. If {Condition A} is false, but {Condition B} is true, then return {Value B}.
>
> If {Condition A} and {Condition B} are both false but {Condition C} is true, then return {Value C}, etc.
>
> Finally, if all conditions are false, return the value given after "else."

When using this function, it is important to remember that the conditions are evaluated in order, and only as necessary. As soon as an `If` condition is true, evaluation is not performed for any remaining conditions.

Also, see `If`. In some cases, it may be more appropriate to use a conditional where clause.

Sample applications of this formula
- Marking a list according to which items are larger than a maximum allowable value.
- Assigning a rating to a feature based on a test of its properties.

Formula syntax

When you add the `IfThenElse` function to a formula, the program will display the following syntax:

```
IfThenElse( If( ),
    Then( ),
    If( ),
    Then( ),
    Else( ) )
```

The following are some examples using `IfThenElse` statements:

To put a cap, called "MaxUnits," on the number of building units allowed on any parcel, you could create a new attribute in the parcels layer called "adjusted unit capacity" with the following formula:

```
IfThenElse(If ( [ Attribute:UnitCapacity ] > [ Attribute:MaxUnits] ),
    Then ( [ Attribute:MaxUnits ] ),
    Else ( [Attribute:UnitCapacity]) )
```

To create an attribute that assigns numeric ratings to one of three categories (high, medium, or low), where the category boundaries are given by assumptions:

```
IfThenElse( If( [Attribute:Rating] > [Assumption:Medium Threshold] ),
     Then ("High"),
     If( [Attribute:Rating] > [Assumption:Low Threshold] ),
     Then ("Medium"),
Else ("Low"))
```

Type of value returned
Any value

See also: `If` (if...then) function

Intersects function

`Intersects` determines whether a shape *overlaps* any features on the target layer. This function returns a value of either TRUE or FALSE.

`Intersects` uses the broadest possible definition: if the current shape touches any part of a feature in the target layer, it counts as intersecting.

In the diagram below, the current shape is a hexagon and the other shapes show example features in the target layer, each labeled True (indicating that they *do* intersect by this definition) or False (indicating otherwise).

Note that a shape that shares an edge with the current shape counts as True, as does a line that ends at the edge of the current shape. The current and target shapes can be a point, line or a polygon.
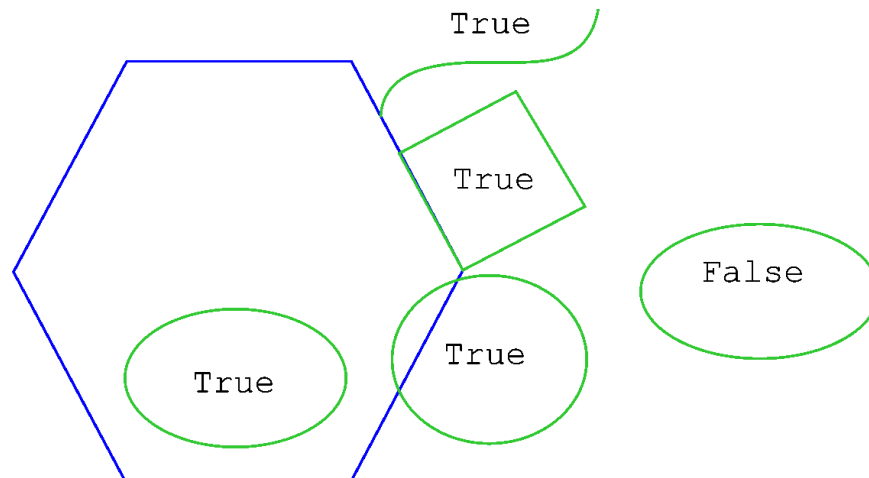
Notice the target shape needs only to overlap the current shape, not necessarily intersect through the border or perimeter.

Tip This is not the same as the geoprocessing intersect function, which creates a new shape from overlapping areas. For help on geoprocessing functions, see the ArcMap™ help.

Sample applications of this formula
- Noting whether a trail crosses a river
- Determining if a proposed burn area is within a habitat buffer zone
- Determining if a building footprint is on top of a utilities right-of-way or other right of egress

Formula syntax

When you add the `Intersects` function to a formula, the program will display the following syntax:

```
Intersects( {target},
     Where( {where} ) )
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

As an example, the `Intersects` function might be used to create an attribute that specifies whether a trail crosses the Bluewater Creek. The syntax for this formula would appear as:

```
Intersects( [ Layer:Streams ],
     Where( [ Attribute:Streams:NAME ] = "Bluewater Creek" ) )
```

Current layer shape type
Polygon, Line, or Point

Target shape type
Polygon, Line, or Point

Type of value returned
Boolean (TRUE or FALSE)

IsCenterContainedIn (Is Center Contained In) function

`IsCenterContainedIn` determines whether the center of the current shape is contained in any features in the target layer. This function returns a value of either TRUE or FALSE.

`IsCenterContainedIn` determines whether the center of the current shape lies within one or more features in the target layer. Compare it to CenterContains, which checks for the opposite condition.

In the diagram below, the current shape is the hexagon, with the star representing the center of the shape. The three ellipses are example features in the target layer. If the target layer contains either of the larger ellipses, `IsCenterContainedIn` will return True. The center of the hexagon lies within the two larger ellipses. However, if the target layer only contains the smaller ellipses, `IsCenterContainedIn` will return False since none contain the center (star) of the hexagon.
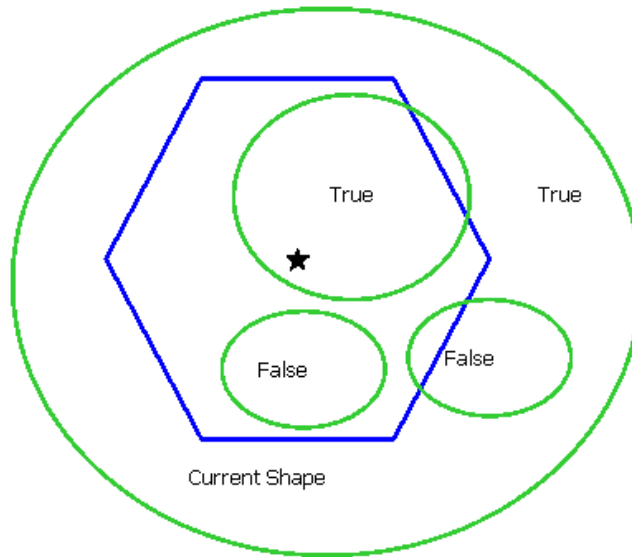
The current shape can also be a point, line, or polygon. The target shape must be a polygon.

Sample applications of this formula

- Deciding whether a parcel is "in" the floodplain or not. `IsCenterContainedIn` will return False (indicating *not* in the floodplain) if, for example, just one corner of the parcel is in the floodplain.
- Part of an analysis counting how many lots are within a proposed new school district boundary. Planners can use `IsCenterContainedIn` to decide whether a given lot is either in or out of the district.
- Part of a suitability analysis rating the appropriateness of parcels for building based upon whether they are "on" steep slopes.

Formula syntax

When you add the `IsCenterContainedIn` function to a formula, the program will display the following syntax:

```
IsCenterContainedIn( {target},
    Where( {where} ) )
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

An example using this function would be to find out whether the 30-year floodplain covers the center of a parcel: (This example includes the use of a "where clause" to specify the 30-year floodplain as opposed to the 100-year floodplain.) You would write the syntax for this formula like this:

```
IsCenterContainedIn( [ Layer:Floodplain ],
    Where( [ Attribute:Floodplain:LEVEL ] = "30-year" ) )
```

Current layer shape type

Polygon, Line, or Point

Target shape type

Polygon

Type of value returned

Boolean (TRUE or FALSE)

IsContainedIn (Is Contained In) function

`IsContainedIn` determines whether a current shape is *completely* contained in any features in the target layer. This function returns a value of either TRUE or FALSE.

`IsContainedIn` determines whether the current shape lies within one or more features in the target layer. Compare it to the function Contains, which checks for the opposite condition.

In the diagram below, the current shape is the hexagon, and three ellipses show example features within the larger target layer. If the target layer contains the large ellipse, `IsContainedIn` will return True since it contains the current shape. However, if the target layer only contains the smaller ellipses, `IsContainedIn` will return False since the current shape is not contained in any of the smaller features.

A target shape that shares an edge with the current shape *can* count as True. The current shape can also be a point, line, or polygon. The target shape must be a polygon.

Tip Use `IsContainedIn` to check whether points are contained within a given area. It can also be used on lines and polygons.

Sample applications of this formula
- Deciding whether a construction site is within an appropriately zoned parcel.
- Part of an analysis counting trees in a specified vegetation treatment area.
- Testing whether a proposed conservation area completely includes an endangered species habitat.

Formula syntax
When you add the `IsContainedIn` function to a formula, the program will display the following syntax:

```
IsContainedIn( {target},
    Where( {where} ) )
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

To create an attribute that specifies whether a spring lies within privately owned land:

```
IsContainedIn( [ Layer:Ownership ],
Where( [ Attribute:Ownership ] = "Private" ) )
```

Current layer shape type
Polygon, Line, or Point

Target shape type
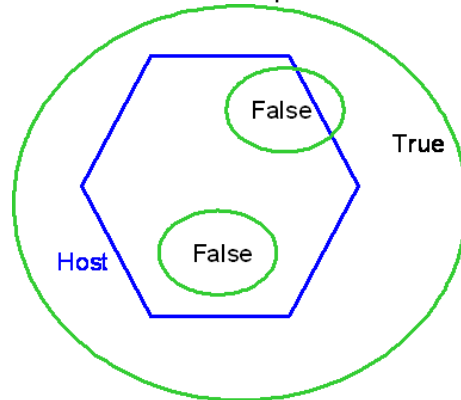Polygon

Type of value returned
Boolean (TRUE or FALSE)

IsInfinity (Is Infinity) function

`IsInfinity` checks whether an attribute value is very large due to dividing by zero. If so, the function returns True; otherwise, it returns False.

Sample applications of this formula:

- IsInfinity(9/[Attribute:Cost]) = True if the attribute cost has a value of 0.

Type of value returned
Boolean (TRUE or FALSE)

IsNull (Is Null) function

`IsNull` checks whether an attribute value is missing (that is, "<null>"). If so, the function returns True; otherwise, it returns False.

Sample applications of this formula:

- If(IsNull([Attribute:Buildable Area:Existing Floor Area]),
  Then(0),
  Else(12)) gives an answer of 0 if the "Existing Floor Area" attribute is missing (that is, null),
and otherwise gives an answer of 12.

Type of value returned
Boolean (TRUE or FALSE)

IsSelected (Is Selected) function

The `IsSelected` function tests whether a feature or a row in a table is currently selected. The function returns True (or 1) if the feature is selected, and False (or 0) if the feature is not selected.

Note: The function does not automatically update when the selection changes. You must click the **Update IsSelected Formulas**   button on the Scenario 360 toolbar or Scenario 360 toolbar dropdown Data list, or update the formula in some other way, to receive new values. The most common way to update a formula is to click the **Update Now** button on the Edit Attribute or Edit Indicator window. For more information on updating formulas, see Managing dynamic updates.

A common way to select a feature is to start editing and use the Selection cursor to click on the feature or to drag a box around it. Selected features normally appear on the map with a turquoise outline. You may need to Set Selectable Layers. There are many other options for selecting features available from the ArcMap **Selection** menu. See ArcMap help for details.

`IsSelected` is a useful way to make your analysis more interactive for users.

Sample applications of this function

- Provide an easy way for users to get indicator information from a subset of features on the map, such as features within a particular township or block. To make this work, create indicator formulas that include a `Where( IsSelected( ) )` clause of an `If ( IsSelected ( ) )`.
- Create clickable graphics on the map that launch an executable (such as an audio or video file or a website) when selected. To make this work, make a new dynamic attribute in the graphics layer called Selected whose formula is simply `IsSelected( )`. Then make an Alert that executes a file when Selected = 1.

Formula syntax
When you add the IsSelected function to a formula, the program will display the following syntax:
`IsSelected ( )`

There are no further steps. Do not place anything inside in the parentheses or brackets; they are there for formatting reasons only.
To make `IsSelected` a condition of a *Where* clause, type IsSelected( ) inside the brackets of the Where clause, like this: `Where( IsSelected( ) )`.

Current layer shape type
Point, Line, Polygon or Table

Type of value returned
Boolean

Left function

The `Left` function returns a specified number of characters from the left side of a string (text field). The `Left` function is a Text function.

`Left` requires two input parameters: the text to be processed (called the "input string"), and the number of characters to keep (called the "length"). Characters are counted from left to right starting at 1. Spaces, including leading and trailing spaces, count as characters.

The input string is normally a text attribute, such as `[Attribute:StringName]`. However, any string value is allowed, so you may enter text enclosed in double quotes, such as "ABCD FG", or you may

enter a function that returns text, such as `GetFromClosest (
[Attribute:SchoolLayer:SchoolName])`.

The length must be zero or a positive integer and is normally typed into the formula, but it may also be given by a function that returns an integer.

In the example below, the input string is 7 characters of text with a space in position 5.

A	B	C	D		F	G
1	2	3	4	5	6	7

`Left ("ABCD FG", 2)` = AB
`Left ("ABCD FG", 0)` = (blank)
`Left ("ABCD FG", 5)` = ABCD followed by a single space
`Left ("ABCD FG", 8)` =ABCD FG with *no* trailing space added

Sample applications of this formula

- Abbreviate long names to save space or time
- Extract the street number from a complete street address

Formula syntax
When you add the `Left` function to a formula, the program will display the following syntax :
`Left ( , ) ' ( string, length )`

The green text starting from the single quote is a comment to inform you that the input string goes before the comma and the length goes after the comma. To complete the function, specify an input string and a length in that order, separated by a comma:

`Left ( string, length)`

If you are entering a static string value, place double quotes ("text") around the text. Neither the string nor the length is verified as part of the formula creation process. If either is invalid, an error will appear during the update process. Choose **Abort** and return to the Formula Editor to correct your formula.

Current layer shape type
Point, Line, Polygon, or Table

Type of value returned
Text

Length function

The `Length` function calculates the flat map length of a shape. The program will display the results in map units.

`Length` is a formula function that calculates the length of a shape and is usually used to measure lines. If the specified shape is a polygon, `Length` returns its perimeter. If the specified shape is a point, `Length` returns a value of 1 (counts it). <u>The area measured is that of a flat map no topological irregularities are considered.</u> To find the surface area of a three-dimensional terrain, use the 3D Analyst extension in ArcGIS™.

`Length` is often a pre-existing attribute in imported data and may not require recalculating.

`Length` is a common function used in many calculations.

Sample applications of this formula
- Calculating the length of a road to estimate its maintenance costs
- Calculating the length of a ferry route to estimate its operating costs
- Calculating the length of a hiking trail for recreation

Formula syntax
When you add the `Length` function to a formula, the program will display the following syntax:

```
Length( [Attribute:Shape] )
```

Normally you do not need to take any further steps. However, if desired you can change the `Attribute` by right-clicking on it.

WARNING! Some imported data layers, particularly those imported from coverages, may already have existing attributes called "AREA" or "LENGTH." However, these will not be dynamic attributes and will not recalculate if the layer is edited. It is recommended you create new, Scenario 360-based dynamic attributes for `Area` and `Length` if you plan to use them in Scenario 360 formulas. You may want to consider deleting the old AREA and LENGTH fields to avoid confusion. (Also, geodatabases will automatically create attributes called `SHAPE_Area` and `SHAPE_Length`. While these attributes do update dynamically based on editing, they may or may not use the same units as the rest of your formula.

Current layer shape type
Polygon, Line, or Point

Type of value returned
Numeric

Ln (Natural Logarithm) function

The `Ln` function calculates the natural logarithm of a specified number.

Natural log is the inverse function of the natural exponential, e^x. In other words, if $e^x = y$, then $ln \ y = x$. Natural log is sometimes referred to as "log base e." The value of e is approximately 2.71828 and is available by typing e in the formula.

Formula syntax

When you add the `Ln` function to a formula, the program will display the following syntax:

    Ln( )

To complete the function, enter a positive number in the parentheses.

Type of value returned

Numeric

Log (Logarithm) function

The `Log` function calculates the logarithm of a specified number.

The logarithm-base-y of x, usually written $log_y \ x$, is the number to which power y must be raised to produce x. If $log_y \ x = z$, then $y^z = x$. In this example y is the "base" of the log function.

Formula syntax

When you add the `Log` function to a formula, the program will display the following syntax:

    Log({number}, {base})

To complete the function, enter a positive number in the parentheses.

Type of value returned

Numeric

Log10 (Base 10 Logarithm) function

The `Log10` function calculates the base 10 logarithm of a specified number.

The base 10 logarithm of x, usually written $log \ x$, is the number to which power 10 must be raised to produce x. If $log \ x = z$, then $10^z = x$.

Formula syntax

When you add the `Log10` function to a formula, the program will display the following syntax:

    Log10( )

To complete the function, enter a positive number in the parentheses.

Type of value returned

Numeric

Max (Maximum) function

`Max` calculates the maximum value in a numeric attribute or the maximum size of a shape attribute.

If it is operating on a shape attribute, `Max` finds the size of the largest feature in a layer. If it is operating on a numeric attribute, it finds the largest number value in the layer.

Sample applications of this formula
- Finding the steepest slope in a region
- Finding the size of the largest remaining agricultural parcel that still has development rights for sale

Formula syntax
When you add the `Max` function to a formula, the program will display the following syntax:

```
Max( {attribute},
    Where( {where} ) )
```
To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

You could use the `Max` formula to find the size of the largest lot in a zoning plan that contains one existing dwelling unit (DU). In this case, the formula would be:

```
Max ( [ Attribute:Parcel Zoning:ACRES ],
    Where ( [ Attribute:Parcel Zoning:EXISTING DU ] = 1 ) )
```

Type of value returned
Numeric

MaxDistance (Maximum Distance) function

The `MaxDistance` function identifies the straight-line distance from the current feature in a layer to the furthest feature in the target layer. Topological variations such as hills and valleys will not affect the formula. The program displays the result as a number in map units. For details, refer to `MinDistance`, which works the same way except for finding the nearest feature instead of the furthest feature..

Mean (average) function

`Mean` calculates the average value of a numeric attribute or the average size of a shape attribute. The mean is the average value of a set of numbers.

If `Mean` is operating on a shape attribute, this formula finds the average area, average length for lines, and/or the average number of points of specified features in a layer. If `Mean` is operating on a numeric attribute, it finds the average number value in the layer. The program calculates the `Mean` (or average) by adding all the values in the list, then dividing by the number of values.

Tip If you are looking for a representative value within a list, consider using `Median` rather than `Mean`. For example, consider the following list:

```
5   5   6   6   7   7   7   8   100
```

The mean value of this list is 16.8, while the median is 7. If these were the ages of people visiting a park, park designers would anticipate elementary school age children if using the `Median` formula. However, if using the `Mean`, designers might propose a park for teenage visitors.

Sample applications of this formula
- Finding the average slope in a region
- Finding the average lot size in a parcel
- Creating a maximum threshold condition for an alert

Formula syntax
When you add the `Mean` function to a formula, the program will display the following syntax:

```
Mean( {attribute},
     Where( {where} ) )
```

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

You could use `Mean`, to find the average size of largest lot in a zoning plan that has one existing dwelling unit (DU) on it. The formula would be:

```
Mean ( [ Attribute:Parcel Zoning:ACRES ],
     Where ( [ Attribute:Parcel Zoning:EXISTING DU ] = 1 ) )
```

Type of value returned

Numeric

Median (middle number) function

`Median` identifies the middle value in a numeric attribute or the median size of a shape attribute. An equal number of values lie above and below the `Median` value.

The `Median` value in a list of values is the center one when the values are listed in order by size. There are the same number of entries in the list that are larger than the `Median` as there are entries that are smaller than the `Median` . If it is operating on a shape attribute, `Median` finds the median size of features in a layer. If it is operating on a numeric attribute, it finds the median number value in the layer.

Tip Consider using Mean rather than `Median` if you will use the answer elsewhere for calculating totals. For example, consider the following list:

```
5   5   6   6   7   7   7   8   100
```

There are 9 items in the list. The Mean value of this list is 16.8, while the `Median` is 7. If these numbers represented traffic, measured in vehicle trips per day per dwelling unit, the calculation be as follows:

     9 dwelling units x `Median` vehicle trips per day = 56

This represents a very different estimate for total traffic than the more appropriate calculation:

     9 dwelling units x `Mean` vehicle trips per day = 151.

Sample applications of this formula
- Finding the median income for a neighborhood
- Finding the median age of trees in a forest

Formula syntax

When you add the `Median` function to a formula, the program will display the following syntax:

```
Median( {attribute},
    Where( {where} ) )
```

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

You could use the `Median` function to find the median size of the lots in a zoning plan that contains one existing dwelling unit (DU). In this case, the formula would be:

```
Median ( [ Attribute:Parcel Zoning:ACRES ],
    Where ( [ Attribute:Parcel Zoning:EXISTING DU ] = 1 ) )
```

Type of value returned

Numeric

Middle function

The `Middle` function returns a specified number of characters from the middle of a string (text field). The `Middle` function is a Text function.

`Middle` requires three input parameters: the text to be processed (called the "input string"), the character position to start at (called the "start position"), and the number of characters to keep (called the "length"). Characters are counted from left to right starting at 1. Spaces, including leading and trailing spaces, count as characters.

The input string is normally a text attribute, such as `[Attribute:StringName]`. However, any string value is allowed, so you may enter text enclosed in double quotes, such as "ABCD FG", or you may enter a function that returns text, such as `GetFromClosest ( [Attribute:SchoolLayer:SchoolName] )`.

The start position must be a positive integer greater than 0 and is normally typed into the formula, but it may also be given by a function that returns a non-zero positive integer.

The length must be zero or a positive integer and is normally typed into the formula, but it may also be given by a function that returns an integer.

In the example below, the input string is 7 characters of text with a space in position 5.

A	B	C	D		F	G
1	2	3	4	5	6	7

```
Middle ("ABCD FG", 3, 4) = CD F
Middle ("ABCD FG", 1, 6) = ABCD F
Middle ("ABCD FG", 2, 10) = BCD FG with no following spaces
Middle ("ABCD FG", 100, 1) = (blank)
```

Sample applications of this formula

- Extract parts of a zoning code
- Divide a series of characters into individual fields by creating a Middle function to pull out each character into its own attribute

Formula syntax

When you add the Middle function to a formula, the program will display the following syntax:
```
Middle( , , ) ' ( string, start position, length )
```

To complete the function, specify an input string, a start position, and a length in that order, separated by commas:
```
Middle ( string, start position, length)
```

If you are entering a static string value, place double quotes ("text") around the text. Neither the string nor the length is verified as part of the formula creation process. If either is invalid, an error will appear during the update process. Choose Abort and return to the Formula Editor to correct your formula.

Current layer shape type

Point, Line, Polygon or Table

Type of value returned

Text

Min (Minimum) function

`Min` calculates the minimum value in a numeric attribute or the minimum size in a shape attribute or the minimum of a list of values.

If `Min` is operating on a shape attribute, it finds the size of the smallest feature in a layer. If `Min` is operating on a numeric attribute, it finds the smallest number value in the layer. If `Min` is operating on a list of values, it finds the smallest value.

Formulas using the `Min` function may update faster in some circumstances if they can perform incremental updates. See the **Incremental Update** article earlier in this guide.

Sample applications of this formula
- Finding the smallest contained region specified as habitat for an endangered species.
- Checking to see if any lots in a subdivision proposal are smaller than the zoned minimum.
- Finding the shortest distance to a train or bus stop.

Formula syntax
When you add the `Min` function to a formula, the program will display the following syntax:
```
Min( {attribute},
```

```
            Where( {where} ) )
```

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

As an example, you may use the `Median` function to find the smallest lot in a zoning plan that contains one existing dwelling unit (DU). In this case, the syntax would appear as:
```
    Min ( [ Attribute:Parcel Zoning:ACRES ],
        Where ( [ Attribute:Parcel Zoning:EXISTING DU ] = 1 ) )
```

To use the function to find the minimum of a list of values, enter the values, separate by commas, in place of the attribute field. For example, to find the minimum distance to a train or bus stop, use this formula:
```
        Min ( MinDistance([Layer:Train_Stations]),
        MinDistance([Layer:Bus_Stops])
```

Type of value returned
Numeric

MinDistance (Minimum Distance) function

The `MinDistance` function identifies the shortest straight-line distance from the current feature in a layer to the closest feature in the target layer. Topological variations such as hills and valleys will not affect the formula. The program displays the result as a number in map units.

The `MinDistance` function measures how far it is to another feature, even if the feature is in another layer. It uses straight-line distances, and if the target layer contains more than one feature, it calculates the distance to the nearest one.

In the diagram below, the current feature is on the left and features from the target layer are on the right. `MinDistance` is shown calculating the shortest distance to the nearest edge of the nearest feature in the target layer. Target features can be lines or points as well as polygons.

MinDistance

Current Feature   Nearest Feature in Target Layer
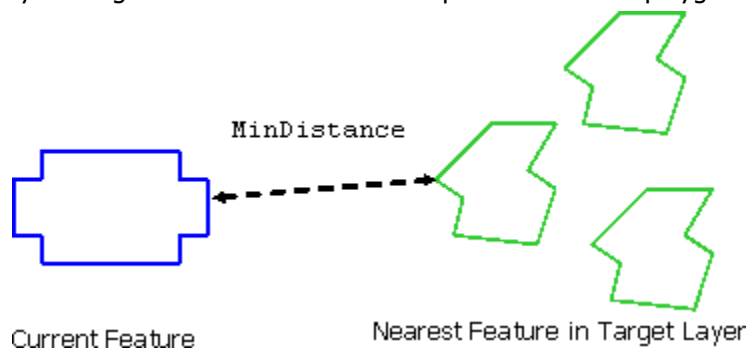
Tip The `MinDistance` measurement is based on flat maps, therefore if line goes over hills or valleys `MinDistance` will give a shorter distance than a path along the three-dimensional surface. To calculate long distances along 3D surfaces, use the 3D Analyst extension for ArcGIS™.

Note: `MinDistance` does not measure the distance between two features along a road or other designated path; rather, it measures the shortest straight-line distance. To calculate actual distance along a road, use the NetworkMinDistance (minimum distance along a network) function or see the ArcMap™ help topic on Linear Referencing/Routes and Measures.

Tip To calculate distance to the center of a polygon rather than its edge, you can calculate an approximate distance from the edge to the center and add it to `MinDistance`. A reasonable approximation for the center-to-edge distance is

```
Sqrt( Area ([Attribute:Shape] ))/∏
```

See `Sqrt` (Square Root) for more information.

When working with polygons, `MinDistance` measures from the nearest point on the edge of the polygon. If working with lines, it measures from the nearest point on the line. If a target shape intersects the host shape, the function returns 0.

Sample applications of this formula
- Determine the distance from a structure to the nearest well
- Measure the distance from a parcel to the nearest road
- Determine the distance from a nesting site to the nearest stream

Formula syntax
When you add the `MinDistance` function to a formula, the program will display the following syntax:

```
MinDistance( {target},
     Where( {where} ))
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

For example, you may wish to find the closest lake that is at least one acre in size. (The where clause also uses a conversion factor to convert acres to square feet.) The formula for this problem would look like:

```
MinDistance( [ Layer:Lakes ],
   Where( [ Attribute:Lakes:AREA ] x [ Conversion:Sq Feet to Acres] >= 1)
   )
```

Current layer shape type
Line, Point, or Polygon

Target shape type
Line, Point, or Polygon

Type of value returned
Numeric

NetworkGetFromClosest (Network Get From Closest) function

`NetworkGetFromClosest` gets an attribute value from the closest feature in the target layer, with distance measured along a network. (Requires the Network Analyst extension.)

This is a **Spatial Lookup** function that retrieves an attribute value, such as a name or size, from the nearest feature in the target layer, with distance measured along a network. If more than one feature in the target layer overlaps the current shape, the function will get the value from the feature with the most overlapping area or length. In the case of a point target layer, minimum distance to center is used.

The `NetworkGetFromClosest` function determines the closest feature to the current shape in the current layer by using the same logic as expressed in the `NetworkMinDistance` (Network Minimum Distance) formula. Instead of distance, other network attributes such as travel time or cost can be used.

Once the formula determines the closest feature, it will look up and return the value of the specified attribute for that feature.

Optionally, you can place conditions on which features of the target layer to include in the calculation.

Sample applications of this formula
- Find the name of the park within the shortest walking distance along sidewalks.
- Find the capacity of the nearest gravel pit assuming travel via roads.
- Assign a school district to houses based on which school is closest to drive to.

Formula syntax
When you add the `NetworkGetFromClosest` function to a formula, the program will display the following syntax:

```
NetworkGetFromClosest( {attribute}, {network attribute},
        50, ' Network search tolerance
        0, ' Network search cutoff distance
              Where ( {where} ) )
```

To complete the function, specify the attribute you want to get by right-clicking on *{attribute}*. Specify the network layer and network attribute by clicking on *{network attribute}*. (Typically the network attribute will be a "distance" attribute in your network, but you may use other attributes if you wish.) The program will display a selection box with available target layers and network layer attributes, respectively. If no network layer is available in your analysis, the function cannot work and clicking on the network attribute will delete it.

To specify a "search tolerance", click on the prompt and either accept the default of 50 map units or type in a new value. Search tolerance determines how far away a feature can be away from the nearest point in the network and still be considered "on" the network. Features that lie outside the search tolerance boundary are not considered accessible to the network. For example, if houses are set back 75 feet from the road centerline, the search tolerance should be at least 75 feet. The units of search tolerance are based on map units.

Specifying a "search cutoff distance" limits the search to a maximum allowed distance between origins and target destinations. If you use the default cutoff distance of '0', no cutoff distance will be used and network distance to the nearest target destination will be calculated for every origin feature. The cutoff distance can be a constant, an indicator, or an assumption. If no target destinations are found within the cutoff distance, a value of '0' will be returned for a numeric and Boolean fields and an empty string for text fields,

Optionally, specify a where clause by right-clicking on *{where}*. If you are not using the **where** clause you can delete it by right-clicking on it and clicking **Remove Where Clause** on the pop-up menu. After clicking on *{target}*, the program will display a selection box with available target layers.

Note: The where clause applies to the target layer, not the network.

For example, to create an attribute for houses the name of the nearest elementary school based on driving time, create an attribute called "Name of Elementary School" with the following formula:

```
NetworkGetFromClosest ([Attribute:Schools:Name],
[Attribute:Roads:DrivingTime], 150, [Attribute:NetworkSearchDistance],
Where([Attribute:Schools:Type]= "Elementary")
```

Current layer shape type
Point, Line, or Polygon

Target shape type
Point, Line, or Polygon

Note: Network layer must be of Network type and Network Analyst must be available

Type of value returned
Numeric

NetworkMinDistance (Network Minimum Distance) function

`NetworkMinDistance` calculates the distance along a network to the nearest feature in the target layer. (Requires the Network Analyst extension.)

The `NetworkMinDistance` function is a **Spatial Numeric** measurement that finds the shortest distance (or other network attribute) from the current feature to the closest feature in the target layer, following a network path. Results are provided in map units. This function requires the Network Analyst extension, a network dataset, and a network cost attribute.

Distance is measured from the center of the current feature to the nearest point on the nearest feature in the target layer, measured along the specified network layer. If the network does not intersect the current feature or the target feature, distance is measured along the network from the nearest point on the network to the feature. The distance from the feature to the network is not included in the measurement.

While distance is the most common attribute to measure along a network, this function can also be used to measure minimum travel time, cost, or other attribute present in the network layer.

Optionally, you can place conditions on which features of the target layer to include in the calculation.

Sample applications of this formula
- Find the driving distance from a house to the nearest park.
- Find the walking time from an office to the nearest transit stop.
- Estimate average vehicle miles traveled (VMT) for a new subdivision.

Formula syntax
When you add the `NetworkMinDistance` function to a formula, the program will display the following syntax:

```
NetworkMinDistance( {target}, {network attribute},
        50, ' Network search tolerance
        0, ' Network search cutoff distance
        Where ( {where} ) )
```

To complete the function, specify a target layer by right-clicking on *{target}* and the network attribute by clicking on *{network attribute}*. (Typically the network attribute will be a "distance" attribute in your network, but you may use other attributes if you wish.) The program will display a selection box with available target layers and network layer attributes, respectively. If no network layer is available in your analysis, the function cannot work and clicking on the network attribute will delete it.

To specify a "search tolerance", click on the prompt and either accept the default or 50 map units or type in a new value. Search tolerance determines how far away a feature can be away from the nearest point in the network and still be considered "on" the network. Features that lie outside the search tolerance boundary are not considered accessible to the network. For example, if houses are set back 75 feet from the road centerline, the search tolerance should be at least 75 feet. The units of search tolerance are based on map units.

Specifying a "search cutoff distance" limits the search to a maximum allowed distance between origins and target destinations. If no target destinations are found within the cutoff distance, a value of '-1' will be returned. If you use the default cutoff distance of '0', no cutoff distance will be used and network

distance to the nearest target destination will be calculated for every origin feature. The cutoff distance can be a constant, an indicator, or an assumption.

Optionally, you can specify a **where** clause by right-clicking on *{where}*. If you are not using the **where** clause you can delete it by right-clicking on it and clicking **Remove Where Clause** on the pop-up menu. After clicking on *{target}*, the program will display a selection box with available target layers.

> Note: The where clause applies to the target layer, not the network.

For example, to create an attribute for houses giving their driving time to the nearest elementary school, create an attribute called "Time to Elementary School" with the following formula:

```
NetworkMinDistance([Layer:Schools], [Attribute:RoadsNetwork:DriveTimes], 150,
500, Where([Attribute:Schools:Type]= "Elementary")
```

Current layer shape type
Point, Line, or Polygon

Target shape type
Point, Line, or Polygon

> Note: Network layer must be of Network type, and Network Analyst must be available

Type of value returned
Numeric

Norm (Normalize) function

`Norm` proportionally adjusts a set of attributes so that they range between specified *minimum* and *maximum* values. The minimum and maximum values of the range are user-specified parameters. This function performs a linear renormalization.

The function works by first determining the original minimum value (Min_0) and original maximum value (Max_0) in the list of attribute values. It uses as input parameters the new, rescaled minimum value (Min_R) and maximum value (Max_R). Then, for each original attribute value $(Attr_0)$, it determines a rescaled value $(Attr_R)$ using:

$$Attr_R = (Min_R) + [(Max_R) - Min_R)]*[(Attr_0) - (Min_0)] / [(Max_0) - (Min_0)]$$

Example
In the following table, Original is an existing field in an attribute table, and Normalized is the result of a `Norm` calculation with parameters *minimum* = 0 and *maximum* = 100.

Original	Normalized
30	0
35	10
80	100
65	50

Notes

- the minimum and maximum parameters may be any numeric values, including negative values and fractional values
- the minimum and maximum parameters may be analysis components such as indicators or assumptions, but not attributes
- the minimum and maximum values in the original list must not be equal
- to generate results in reverse order from the original, choose a minimum greater than the maximum
- this function only works for attributes
- this function and its input attribute must be in the same layer

Sample applications of this formula

- Creating a "dashboard" of several indicators that all have the same scale for easy charting and display
- Normalizing suitability scores for several factors so they can be fairly compared

Formula syntax

When you add the `Norm` function to a formula, the program will display the following syntax:

```
Norm( {attribute}, 0, ' Minimum
      100, ' Maximum
      Where( {where} ) )
```

To complete the function:
1. Specify a numeric input attribute by clicking on *{attribute}* and choosing a layer and then an attribute from the list.
2. Optionally, change the default *minimum* and *maximum* parameters to those you want to use. (The words "Minimum" and "Maximum" are just reference notes in the formula.) If desired, you may substitute a numeric assumption or indicator for the minimum and/or maximum parameters themselves.
3. Optionally, specify a *Where* condition by clicking on *{where}.* If you are not using the where clause, delete it by right-clicking on it and choosing **Remove Where Clause** from the pop-up menu.

 The *Where* clause is optional. Attribute values in features or rows that fail to meet the Where condition are left unchanged by Norm; that is, Norm returns their original, un-normalized value.

Current layer shape type
Point, Line, Polygon

Type of value returned
Numeric

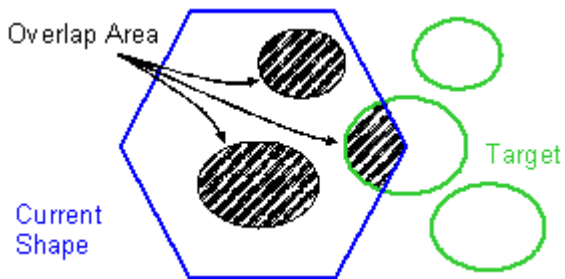
OverlapArea (Overlap Area) function

The `OverlapArea` function calculates the total area (or length or count) of overlap between features in the current layer and features in the target layer. Results are provided in map units.

`OverlapArea` calculates the area of overlap between each feature in the current layer and any feature or features in the target layer. Only the areas that overlap directly are counted. If a single feature in the

current layer overlaps multiple features in the target layer, the areas of overlap are summed. In the diagram below, the current shape is a hexagon and the target layer contains several ellipses. The overlap area is shaded.

`OverlapArea` is usually used for polygons overlapping polygons, but other combinations are allowed, as illustrated in the table below. Overlaps with lines return the length of the overlapping line(s), and overlaps with points return the number of overlapping points. For example, a polygon overlapping a line returns the length of the part of the line or lines that overlap. A polygon overlapping points returns the number of overlapping points. Note that a feature's edges and vertexes count as part of the feature for these purposes. Lines overlapping lines must be exactly coincident (i.e., lying directly on top of one another) to produce a non-zero result; lines that simply cross give a result of 0.

	Current Layer		
Target layer	**Polygon**	**Line**	**Point**
Polygon	area	length	count
Line	length	length	count
Point	count	count	count

Optionally, you can place conditions on which features to include in the calculation.
Formulas using the `OverlapArea` function may update faster in some circumstances if they can perform incremental updates. See Incremental Updates.

Sample applications of this formula
- Determine the amount of wetlands within a proposed residential parcel
- Calculate the percentage of existing farmland that falls with the boundaries of a contemplated industrial park
- Suitability analysis based on overlays

Formula syntax
When you add the `OverlapArea` function to a formula, the program will display the following syntax:
```
OverlapArea( {target},
Where( {where} ))
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

A typical application for the `OverlapArea` function might be calculating how many square feet of wetlands are in the current layer. The formula would be:
```
OverlapArea( [ Layer:Wetlands ],
```

```
Where(  [ Attribute:Wetlands:AREA ] >=  1x[ Conversion:Acres to Sq
Feet ]) )
```

This example uses a conversion factor to convert acres to square feet.

Current layer shape type
Point, Line, Polygon

Target shape type
Point, Line, Polygon

Type of value returned
Numeric

OverlapLength (Overlap Length) function

The `OverlapLength` function calculates the total length of overlap between features in the current layer and features in the target area. Results are provided in map units.

`OverlapLength` compares the current layer polygon and a target layer and then calculates the total length of overlap with the target features. If there is a partial overlap with a feature, only the area directly overlapping the current polygon is calculated. See also: `OverlapArea` (overlap area) function.

`OverlapLength` is usually used for polygons overlapping lines, but it can also be used in other combinations, as illustrated in the following table. In the case of polygons overlapping polygons, the function returns the length of the part of the perimeter of the target polygon(s) that overlap(s) the current polygon.

	Current Layer		
Target Layer	Polygon	Line	Point
Polygon	perimeter	length	n/a
Line	length	n/a	n/a
Point	count	n/a	n/a

Sample applications of this formula
- Determine the length of rivers within a county
- Determine the length of fencing required for fields of a certain type on a farm
- As part of a roads density calculation

Formula syntax
When you add the `OverlapLength` function to a formula, the program will display the following syntax:
```
OverlapLength( {target},
Where( {where} ))
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

A typical application for the `OverlapLength` function might be calculating how miles of Class A roads are in a county. The formula in the county layer would be:

```
OverlapLength( [ Layer:Roads ],
    Where(  [ Attribute:Roads:Class ] =  "A") )* [ Conversion:Feet to
    Miles]) )
```

This example uses a conversion factor to convert feet to miles.

Current layer shape type
Line, Polygon

Target shape type
Point, Line, Polygon

Type of value returned
Numeric

OverlapMost (Overlap Most) function

The `OverlapMost` function determines the value of an attribute in a target layer based on the attribute value it overlaps the most.

This is a **Spatial Lookup** function that retrieves an attribute value, such as a name or type, from features in a target layer. The attribute value retrieved is the one associated with the most overlap area (or overlap length) based on the combined value of all features. If no features are overlapped, the function returns 0. If one feature is overlapped, the function returns that feature's attribute value. This functions most useful application, however, is when multiple features are overlapped, as illustrated in the example below.
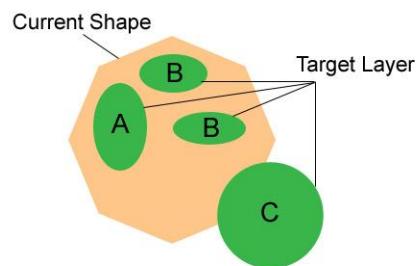
In this illustration, the current shape in the host layer is illustrated with an octagon. The target layer contains ovals, some of which overlap the current shape. Their attribute value is either A, B, or C. In this case, the feature with attribute value C is the largest, but only a small part of it overlaps the current shape. The feature with attribute value A is the next largest, but the two features with attribute value B have a combined overlap area greater than the area of the feature with attribute value A. Therefore, the `OverlapMost` function returns the value B. Although this illustration uses polygons, the function also works for lines and points. Its overlap rules follow the same functional logic as `OverlapArea`.

Comparing OverlapMost to GetFromClosest

`OverlapMost` and `GetFromClosest` both retrieve an attribute value from the target layer based on spatial relationships. `GetFromClosest` finds the attribute value of a single feature. If several features overlap the current shape, `GetFromClosest` finds the feature with the most overlap and retrieves its attribute value. `OverlapMost` finds the attribute value that has the most overlap area counting all of the features together.

Sample applications of the formula

- Determining the dominant land-use in a census block.
- Creating a land-use analysis "grid-like" layer made of regularly-shaped polygons (e.g., squares)
- Classifying development parcels based on their dominant natural resource impact.

Formula Syntax

When you add the `OverlapMost` function to a formula, the program will display the following syntax:

```
OverlapMost ( {attribute},
Where ( {where} ) )
```

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. If you are not using the where clause, you can delete it right-clicking on it and clicking **Remove Where Clause** on the pop-up menu. After clicking on *{attribute}*, the program will display a selection box with available attributes.

Type of value returned

This formula will return the same type of value as the target attribute.

OverlapSum (Overlap Sum) function

The `OverlapSum` function adds together the values of a given attribute for all features in the target layer that overlap the current feature in the host layer. For partial overlap, the attribute value is weighted by the proportion of the target feature that is overlapped.

The overlap calculation uses the same rules and logic as the `OverlapArea` function.
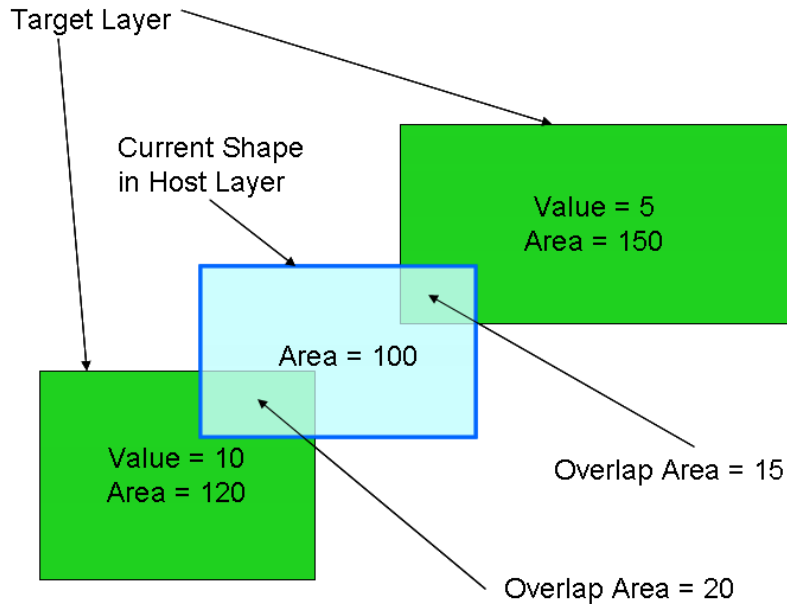
In the example above, the green shaded polygons in the target layer have an attribute called "Value" whose value is 5 in the upper feature and 10 in the lower feature. The current shape in the host layer (outlined in blue) partially overlaps both target features. Specifically, it overlaps 15/150 of the upper shape and 20/120 of the lower shape. The `OverlapSum` is given by

OverlapSum([Layer:Target Layer:Value]) = 5*(15/150) + 10*(20/120) = 2.17

Note that the area of the current shape is not used in the calculation.

Sample applications of this formula

- Estimating the number of existing buildings (given as a per-feature number in the target layer) that lie within a new land-use polygon (the host layer).
- Estimating the fair market value of a right-of-way (host layer) that will cut through tracts of private land (target layer).

Formula syntax

When you add the `OverlapSum` function to a formula, the program will display the following syntax:
```
OverlapSum( {target},
Where( {where} ))
```

To complete the function, specify a target layer by clicking on {target} and choosing a target layer from the list. (You may also right-click for other options.) Optionally, specify a *Where* condition by clicking on {where}. If you are not using the where clause, delete it by right-clicking on it and choosing **Remove Where Clause** from the pop-up menu.

The *Where* clause is optional.

Current layer shape type
Point, Line, Polygon

Target shape type
Point, Line, Polygon

Type of value returned
Numeric

OverlapWeightedAvg (Overlap Weighted Average) function

The `OverlapWeightedAvg` function determines the average weighted value of an attribute for all features within the target layer that overlap with the current shape. For a partial overlap, only the overlapping area is used in the calculation.

The `OverlapWeightedAvg` formula uses the same rules and logic as the `OverlapArea` function, which calculates the area of overlapped features within the target area.
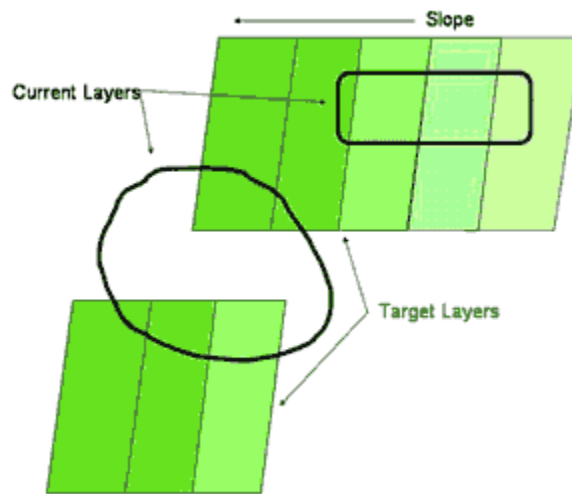
In the example above, the green shaded polygons in the target layer represent a degree of slope. The objective of the formula is to determine the average degree of slope for the current shape, represented by two examples of shapes that overlap the sloped areas. In the upper example, the current shape overlaps parts of four target shapes. The function calculates the percentage of the current shape's area that is covered by each slope, multiplies that percentage by its corresponding slope, and adds all the results. The following table shows a sample calculation.

Slope	% of Area	Weighted slope
20 degrees	10%	2
22 degrees	30%	6.6
24 degrees	40%	9.6
26 degrees	20%	5.2
	Weighted Average:	23.4 degrees

In the lower example, some parts of the current shape do not overlap any features in the target layer. Those areas would be assigned a weighting of zero.

Sample applications of this formula
- Calculating soil erosion for a construction site based on characteristics of the site such as soils and slopes

- Calculating irrigation requirements for a parcel containing several crops.

Formula syntax

When you add the `OverlapWeightedAvg` function to a formula, the program will display the following syntax:

```
OverlapWeightedAvg( {target},
    Where( {where} ))
```

To complete the function, specify a target layer by right-clicking on *{target}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{target}*, the program will display a selection box with available target layers.

In the above example, the syntax for the function would appear as:

```
OverlapWeightedAvg( [ Layer:Slope]),
```

The `where` clause is optional.

Current layer shape type

Polygon

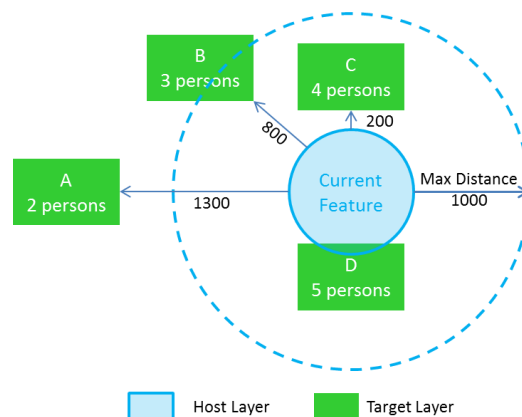## Target shape type

Polygon

Type of value returned

Numeric

ProximityAvg (Proximity Average) function

`ProximityAvg` gives the average values of a given target attribute for all features in the target layer that are "near" the current feature in the host layer. "Near" means equal to or less than the *maximum distance* away, which is an optional, user-specified parameter. Average is a synonym for mean.

The target layer can be any vector data type (point, line, polygon), but note that in the case of lines and polygons, the function includes all features that have any point lying within the maximum distance. Thus this function is not necessarily a direct replacement of a buffer overlap calculation.

Example

In the example illustration, the blue circle is a feature in the current layer (also known as host layer) and the green rectangles are features in the target layer. The target layer includes an attribute giving the number of persons residing in each feature. The *maximum distance* parameter for the function has been set to 1000 map units (e.g., feet if the map is in feet), illustrated by the dotted circle. Details on the target features are:

Target Feature Name	Persons	Distance Away	Notes
A	2	1300	Beyond maximum distance
B	3	800	At least one point within maximum distance
C	4	200	At least one point within maximum distance
D	5	0	Touches or overlaps current feature

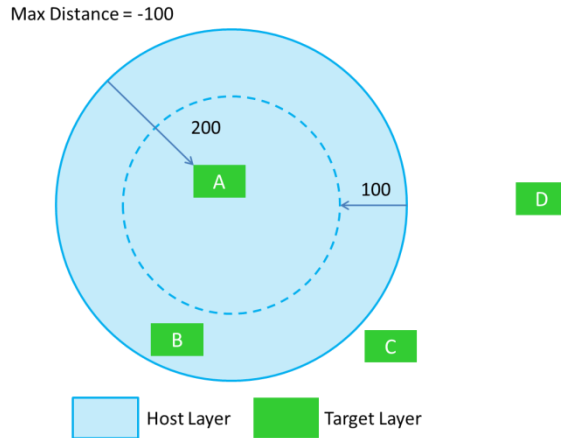The `ProximityAvg` for the current feature is given by the average of the values for features B, C, and D; that is:

```
ProximityAvg([Attribute:Target Layer:Persons], 1000) = (3 + 4 + 5)/3 = 4
```

Notes
- the lengths or areas of the target shapes are not used in the calculation
- maximum distances are calculated according to the same rules as `MinDistance`
- this function only works for attributes or as the target of an aggregate function in an indicator
- if there are no target features within the maximum distance, the function gives a result of 0
- if the target attribute is empty for a given feature, that feature is ignored
- the maximum distance is optional. If none is specified, the function considers all features in the target layer.

Advanced: Maximum distances can be negative. For host or current layers that are points, this is the same as using 0 for the maximum distance. For line layers, this is the same as using the absolute value of the maximum distance. For polygon layers, this is treated as the distance inward from the edge of the polygon. Features that lie within the polygon, and farther from the edge than the absolute value of the maximum distance specified, are considered. In the example below, target feature A would be considered, while target features B, C and D would not.

Max Distance = -100

200

100

A

B

C

D

Host Layer   Target Layer

Sample applications of this formula
- Estimating the average household size in the vicinity of each parcel in a layer as a factor in a school site suitability analysis
- Estimating the average household income in the vicinity public parks as part of an equity assessment

Formula syntax
When you add the `ProximityAvg` function to a formula, the program will display the following syntax:

```
ProximityAvg( {attribute}, 1000, ' Maximum distance
        Where( {where} ) )
```

To complete the function:
1. specify a numeric target attribute by clicking on *{attribute}* and choosing a target layer and then a target attribute from the list.
2. Change the default 1000 to the value (in map units) you want to use for maximum distance. If desired, you may substitute an assumption, indicator, or attribute for the number itself. Alternatively, you may delete the maximum distance and the comma before it.
3. Optionally, specify a *Where* condition by clicking on *{where}.* If you are not using the where clause, delete it by right-clicking on it and choosing **Remove Where Clause** from the pop-up menu.

The *Where* clause is optional.

Current layer shape type
Point, Line, Polygon

Target shape type
Point, Line, Polygon

## Type of value returned
Numeric

ProximityCount (Proximity Count) function

`ProximityCount` counts the number of features in the target layer that are "near" the current feature in the host layer. "Near" means equal to or less than the maximum distance away, which is an optional, user-specified parameter.

The target layer can be any vector data type (point, line, polygon), but note that in the case of lines and polygons, the function includes all features that have any point lying within the maximum distance. It is not necessary for a target feature to have a vertex or centroid within the maximum distance for it to be counted; any point along an edge qualifies. Multipart features are counted only once.

The function's process is similar to buffering the current feature using standard ArcGIS tools and then counting overlaps with the buffer.

Example
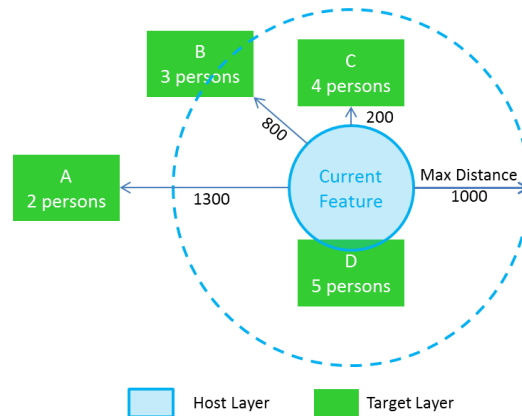
In the example illustration, the blue circle is a feature in the host layer and the green rectangles are features in the target layer. The maximum distance parameter for the function has been set to 1000 map units (e.g., feet if the map is in feet), illustrated by the dotted circle. Details on the target features are:

Target Feature Name	Distance Away	Notes
A	1300	Beyond maximum distance
B	800	At least one point within maximum distance
C	200	At least one point within maximum distance
D	0	Touches or overlaps current feature

The `ProximityCount` for the current feature is the number of features in the target layer that have any point lying within 1000 units of the edge of the current feature, which in this case are features B, C, and D. Thus

$$\text{ProximityCount ([Layer:Target Layer], 1000)} = 3$$

Notes
- the lengths or areas of the target shapes are not used in the calculation

94

- maximum distances are calculated according to the same rules as `MinDistance`.
- Maximum distance is optional. If no value is specified, all features in the target layer are considered.

Advanced: Maximum distances can be negative. For host or current layers that are points, this is the same as using 0 for the maximum distance. For line layers, this is the same as using the absolute value of the maximum distance. For polygon layers, this is treated as the distance inward from the edge of the polygon. Features that lie within the polygon, and farther from the edge than the absolute value of the maximum distance specified, are considered.   In the example below, target feature A would be considered, while target features B, C and D would not.
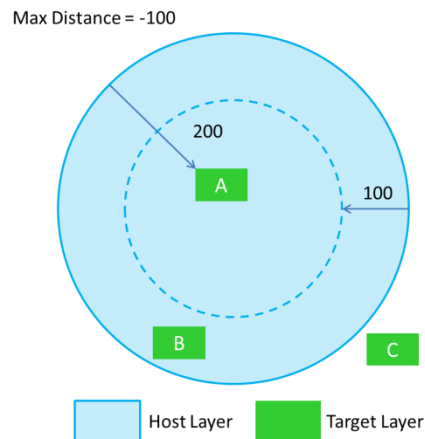
Sample applications of this formula

- Assessing the crime status of parcels by counting the number of crime events nearby.
- Estimating the likelihood of development based on proximity to a large number of existing buildings

Formula syntax

When you add the `ProximityCount` function to a formula, the program will display the following syntax:

```
ProximityCount( {attribute}, 1000, ' Maximum distance
        Where( {where} ) )
```

To complete the function:
4. specify a numeric target attribute by clicking on *{attribute}* and choosing a target layer and then a target attribute from the list.
5. Change the default 1000 to the value (in map units) you want to use for maximum distance. If desired, you may substitute an assumption, indicator, or attribute for the number itself. Alternatively, you may delete the maximum distance and the comma before it.
6. Optionally, specify a *Where* condition by clicking on *{where}.* If you are not using the where clause, delete it by right-clicking on it and choosing **Remove Where Clause** from the pop-up menu.

The *Where* clause is optional.

Current layer shape type
Point, Line, Polygon

Target shape type
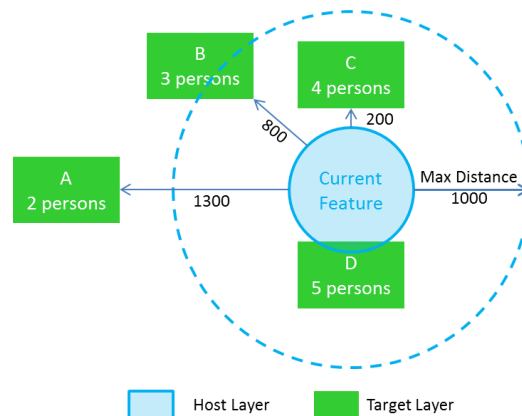Point, Line, Polygon

Type of value returned
Numeric

ProximitySum (Proximity Sum) function

`ProximitySum` adds together the values of a given target attribute for all features in the target the target layer that are "near" the current feature in the host layer. "Near" means equal to or less than the *maximum distance* away, which is an optional, user-specified parameter.

The target layer can be any vector data type (point, line, polygon), but note that in the case of lines and polygons, the function includes all features that have any point lying within the maximum distance. Thus this function is not necessarily a direct replacement of a buffer overlap calculation.

Example

In the example illustration, the blue circle is a feature in the current layer (also known as host layer) and the green rectangles are features in the target layer. The target layer includes an attribute giving the number of persons residing in each feature. The *maximum distance* parameter for the function has been set to 1000 map units (e.g., feet if the map is in feet), illustrated by the dotted circle. Details on the target features are:

Target Feature Name	Persons	Distance Away	Notes
A	2	1300	Beyond maximum distance
B	3	800	At least one point within maximum distance
C	4	200	At least one point within maximum distance

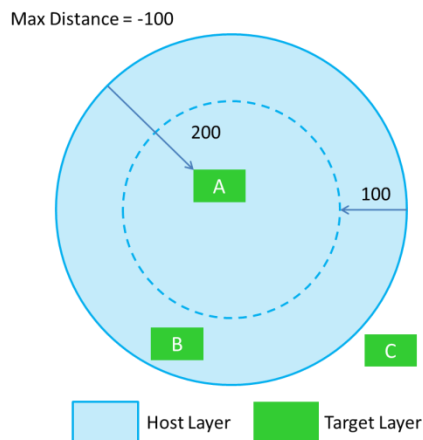| D	5	0	Touches or overlaps current feature

The `ProximitySum` for the current feature is given by the average of the values for features B, C, and D; that is:

```
ProximitySum([Attribute:Target Layer:Persons], 1000) = 3 + 4 + 5= 12
```

Notes
- the lengths or areas of the target shapes are not used in the calculation
- maximum distances are calculated according to the same rules as `MinDistance`
- this function only works for attributes or as the target of an aggregate function in an indicator
- if there are no target features within the maximum distance, the function gives a result of 0
- if the target attribute is empty for a given feature, that feature is ignored
- the maximum distance is optional. If none is specified, the function considers all features in the target layer.

Advanced: Maximum distances can be negative. For host or current layers that are points, this is the same as using 0 for the maximum distance. For line layers, this is the same as using the absolute value of the maximum distance. For polygon layers, this is treated as the distance inward from the edge of the polygon. Features that lie within the polygon, and farther from the edge than the absolute value of the maximum distance specified, are considered.   In the example below, target feature A would be considered, while target features B, C and D would not.

Sample applications of this formula
- Estimating the number of potential customers near each parcel in a layer as a factor in a retail site suitability analysis by summing the number of residents in nearby houses
- Finding the total area of parks nearby each parcel by summing the parks' areas

Formula syntax
When you add the `ProximitySum` function to a formula, the program will display the following syntax:

```
ProximitySum( {attribute}, 1000, ' Maximum distance
        Where( {where} ) )
```

To complete the function:

7. specify a numeric target attribute by clicking on *{attribute}* and choosing a target layer and then a target attribute from the list.
8. Change the default 1000 to the value (in map units) you want to use for maximum distance. If desired, you may substitute an assumption, indicator, or attribute for the number itself. Alternatively, you may delete the maximum distance and the comma before it.
9. Optionally, specify a *Where* condition by clicking on *{where}.* If you are not using the where clause, delete it by right-clicking on it and choosing **Remove Where Clause** from the pop-up menu.

The *Where* clause is optional.

Current layer shape type
Point, Line, Polygon

Target shape type
Point, Line, Polygon

Type of value returned
Numeric

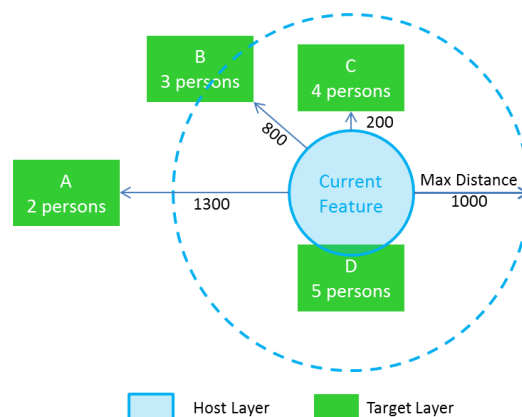
ProximityWeightedAvg (Proximity Weighted Average) function

`ProximityWeightedAvg` averages the weighted values of a given target attribute for all features in the target layer that are "near" the current feature in the host layer, weighting the values inversely with their distance from the current feature. The weighting factor is $1/(1+r)$, where r is the distance in map units. "Near" means equal to or less than the maximum distance away, which is an optional, user-specified parameter. Average is a synonym of mean.

The target layer can be any vector data type (point, line, polygon), but note that in the case of lines and polygons, the function includes all features that have any point lying within the maximum distance.

Example

In the example illustration, the blue circle is a feature in the current layer (also known as host layer) and the green rectangles are features in the target layer. The target layer includes an attribute giving the

number of persons residing in each feature. The *maximum distance* parameter for the function has been set to 1000 map units (e.g., feet if the map is in feet), illustrated by the dotted circle. Details on the target features are:

Target Feature Name	Persons	Distance Away	Notes
A	2	1300	Beyond maximum distance
B	3	800	At least one point within maximum distance
C	4	200	At least one point within maximum distance
D	5	0	Touches or overlaps current feature

The `ProximityWeightedAvg` for the current feature is given by the inverse-distance-weighted average of the values for features B, C, and D; that is:

$$\text{ProximityWeightedAvg}([\text{Attribute:Target Layer:Persons}], 1000) = \{3/(1+800) + 4/(1+200) + 5/(1+0)\}/3 = 1.6745$$

> **Tip:** Notice that the result of `ProximityWeightedAvg` often grows rapidly smaller as *maximum distance* increases. For a more intuitive result, consider using a small *maximum distance* or using the `ProximityAvg` function.

Notes
- the lengths or areas of the target shapes are not used in the calculation
- maximum distances are calculated according to the same rules as `MinDistance`
- this function only works for attributes or as the target of an aggregate function in an indicator
- if there are no target features within the maximum distance, the function gives a result of 0
- if the target attribute is empty for a given feature, that feature is ignored
- the maximum distance is optional. If none is specified, the function considers all features in the target layer.

Advanced: Maximum distances can be negative, as with `ProximityAvg`. However, for host layers that are points or polygons, this implies using target features that overlap the host feature and therefore have a distance away of r = 0. Thus if the maximum distance is negative and the host layer type is a point or polygon, `ProximityWeightedAvg` and `ProximityAvg` produce the same results.

Sample application of this formula
- Looking for areas to designate a wildlife corridor, identify the average value of nearby habitat patches. Use the function to find a neighborhood habitat score where habitat patches have been scored based on size of the habitat patch, ecological condition or other variables.

Formula syntax

When you add the `ProximityWeightedAvg` function to a formula, the program will display the following syntax:

```
ProximityWeightedAvg( {attribute}, 1000, ' Maximum distance
        Where( {where} ) )
```

To complete the function:

1. specify a numeric target attribute by clicking on *{attribute}* and choosing a target layer and then a target attribute from the list.
2. Change the default 1000 to the value (in map units) you want to use for maximum distance. If desired, you may substitute an assumption, indicator, or attribute for the number itself. Alternatively, you may delete the maximum distance and the comma before it.
3. Optionally, specify a *Where* condition by clicking on *{where}.* If you are not using the where clause, delete it by right-clicking on it and choosing **Remove Where Clause** from the pop-up menu.

The *Where* clause is optional.

Current layer shape type
Point, Line, Polygon

Target shape type
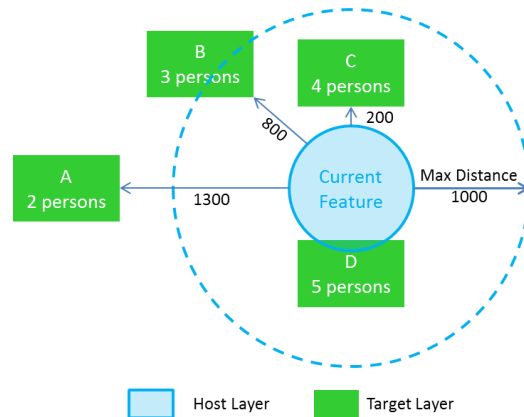Point, Line, Polygon

Type of value returned
Numeric

ProximityWeightedSum (Proximity Weighted Sum) function

`ProximityWeightedSum` adds together the weighted values of a given target attribute for all features in the target layer that are "near" the current feature in the host layer, weighting the values inversely with their distance from the current feature. The weighting factor is $1/(1+r)$, where r is the distance in map units. "Near" means equal to or less than the maximum distance away, which is an optional, user-specified parameter.

The target layer can be any vector data type (point, line, polygon), but note that in the case of lines and polygons, the function includes all features that have any point lying within the maximum distance.

Example

In the example illustration, the blue circle is a feature in the current layer (also known as host layer) and the green rectangles are features in the target layer. The target layer includes an attribute giving the number of persons residing in each feature. The *maximum distance* parameter for the function has been set to 1000 map units (e.g., feet if the map is in feet), illustrated by the dotted circle. Details on the target features are:

Target Feature Name	Persons	Distance Away	Notes
A	2	1300	Beyond maximum distance
B	3	800	At least one point within maximum distance
C	4	200	At least one point within maximum distance
D	5	0	Touches or overlaps current feature

The `ProximityWeightedSum` for the current feature is given by the inverse-distance-weighted sum of the values for features B, C, and D; that is:

```
ProximityWeightedSum([Attribute:Target Layer:Persons], 1000) =

    3/(1+800) + 4/(1+200) + 5/(1+0) = 5.0236
```

Notes
- the lengths or areas of the target shapes are not used in the calculation
- maximum distances are calculated according to the same rules as `MinDistance`
- this function only works for attributes or as the target of an aggregate function in an indicator
- if there are no target features within the maximum distance, the function gives a result of 0
- if the target attribute is empty for a given feature, that feature is ignored
- the maximum distance is optional. If none is specified, the function considers all features in the target layer.

Advanced: Maximum distances can be negative, as with `ProximitySum`. However, for host layers that are points or polygons, this implies using target features that overlap the host feature and therefore

have a distance away of r = 0. Thus if the maximum distance is negative and the host layer type is a point or polygon, `ProximityWeightedSum` and `ProximitySum` produce the same results.

Sample applications of this formula
- Estimating the noise level at individual parks based on their distance away from various noise sources such as roads
- Rating the "park score" of parcels based on their proximity to parks of various sizes, with nearby parks counting more than those far away

Formula syntax
When you add the `ProximityWeightedSum` function to a formula, the program will display the following syntax:

```
ProximityWeightedSum( {attribute}, 1000, ' Maximum distance
        Where( {where} ) )
```

To complete the function:
1. specify a numeric target attribute by clicking on *{attribute}* and choosing a target layer and then a target attribute from the list.
2. Change the default 1000 to the value (in map units) you want to use for maximum distance. If desired, you may substitute an assumption, indicator, or attribute for the number itself. Alternatively, you may delete the maximum distance and the comma before it.
3. Optionally, specify a *Where* condition by clicking on *{where}.* If you are not using the where clause, delete it by right-clicking on it and choosing **Remove Where Clause** from the pop-up menu.

The *Where* clause is optional.

Current layer shape type
Point, Line, Polygon

Target shape type
Point, Line, Polygon

Type of value returned
Numeric

Rand (Random Number) function

The `Rand` function generates a uniformly distributed random number between a minimum and maximum value (inclusive).

"Uniformly distributed" means that if you were to run `Rand` many times, the list of numbers it generated would be uniformly distributed across the range. The minimum is the lowest number `Rand` will generate; the maximum is the highest number it will generate.

Formula syntax
When you add the `Rand` function to a formula, the program will display the following syntax:

```
Rand( {minimum}, {maximum})
```

Type of value returned
Numeric

RandG (Random Number - Gaussian) function

The `RandG` function generates a Gaussian distributed random number with specified mean and standard deviation.

"Gaussian distributed" means that if you were to run `RandG` many times, the list of numbers it generated would produce a Gaussian distribution with the specified mean and standard deviation.

Formula syntax
When you add the `RandG` function to a formula, the program will display the following syntax:
    Randg({mean}, {standard deviation})

Type of value returned
Numeric

RandI (Random Number - Integer) function

The `RandI` function generates a uniformly distributed random integer between a minimum and maximum value (inclusive).

"Uniformly distributed" means that if you were to run `RandI` many times, the list of numbers it generated would be uniformly distributed across the range. `RandI` generates only integers (whole numbers). The minimum is the lowest number `RandI` will generate; the maximum is the highest number it will generate.

Formula syntax
When you add the `RandI` function to a formula, the program will display the following syntax:
    RandI({minimum}, {maximum})

Type of value returned
Integer Number

Right function

The `Right` function returns a specified number of characters from the right side of a string (text field). The `Right` function is a Text function.

Right requires two input parameters: the text to be processed (called the "input string"), and the number of characters to keep on the right side of the string (called the "length"). Characters are counted from right to left starting at 1. Spaces, including leading and trailing spaces, count as characters.

The input string is normally a text attribute, such as `[Attribute:StringName]`. However, any string value is allowed, so you may enter text enclosed in double quotes, such as `"ABCD FG"`, or you may enter a function that returns text, such as `GetFromClosest ( [Attribute:SchoolLayer:SchoolName] )`.

The length must be zero or a positive integer and is normally typed into the formula, but it may also be given by a function that returns an integer.

In the example below, the input string is 7 characters of text with a space in position 5.

A	B	C	D		F	G
1	2	3	4	5	6	7

```
Right ("ABCD FG", 0) = (blank)
Right ("ABCD FG", 2) = FG
Right ("ABCD FG", 4) = D FG
Right ("ABCD FG", 8) = ABCD FG
```

Sample applications of this formula

- Remove direction abbreviations such as S. and N. from street names
- Extract the street name from a complete street address that also contains a number

Formula syntax
When you add the Right function to a formula, the program will display the following syntax :
```
Right( , ) ' ( string, length )
```

The green text starting from the single quote is a comment to inform you that the input string goes before the comma and the length goes after the comma. To complete the function, specify an input string and a length in that order, separated by a comma:
```
Right ( string, length )
```

If you are entering a static string value, place double quotes ("text") around the text. Neither the string nor the length is verified as part of the formula creation process. If either is invalid, an error will appear during the update process. Choose Abort and return to the Formula Editor to correct your formula.

Current layer shape type
Point, Line, Polygon or Table

Type of value returned
Text

Round function

The `Round` function rounds a number up or down to the nearest whole number or specified decimal place.

`Round` converts 1.1 and 1.49 to 1. It converts 1.5 and 1.8 to 2, and it converts −3.5 to −4. A .5 value rounds to the next whole number (1.5 rounds to 2). The number of decimal places to round to can be specified by the second field of the function (it is optional). Rnd (23.777, 1) returns 23.8 (one decimal place), while Rnd (23.777,2) returns 23.78. The default value is 0.

Formula syntax

When you add the `Round` function to a formula, the program will display the following syntax:

```
Round( {number} , {decimal places} )
```

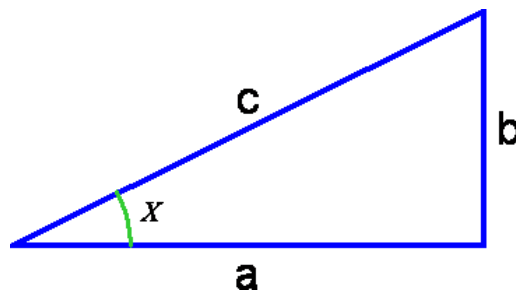To complete the function, enter a number and the number of decimal places in the parentheses.

Type of value returned
Numeric

Sin (Sine) function

The `Sin` function calculates the sine of an angle (in a right triangle the length of a side opposite an angle divided by the length of the hypotenuse of the triangle).

In the diagram below, the sine of X is given by the ratio b/c. The units of X are assumed to be radians.

Tip If X is in degrees, multiply it by Pi/180 to convert it to radians. To get the value of Pi, type "Pi" in the formula.

Formula syntax
When you add the `Sin` function to a formula, the program will display the following syntax:

```
Sin( )
```

To complete the function, enter a value in radians in the parentheses.

Type of value returned
Numeric

Sinh (Hyperbolic Sine) function

The `Sinh` function is a **Number Request** that calculates the hyperbolic sine of the angle.

The hyperbolic sine of x is $(e^x - e^{-x})/2$. It is usually written as `Sinh`. The units of x are assumed to be radians.

Tip If x is in degrees, multiply it by Pi/180 to convert it to radians. To get the value of Pi, type "Pi" in the formula.

Formula syntax
When you add the `Sinh` function to a formula, the program will display the following syntax:

```
Sinh( )
```

To complete the function, enter a value in radians in the parentheses.

Type of value returned

Numeric

Sqrt (Square Root) function

The `Sqrt` function calculates the positive square root of a number.

Formula syntax

When you add the `Sqrt` function to a formula, the program will display the following syntax:

    Sqrt( )

To complete the function, enter a positive number in the parentheses.

Type of value returned

Numeric

StdDev (Standard Deviation) function

`StdDev` calculates the standard deviation of all values in a numeric attribute or standard deviation of sizes in a shape attribute.

Standard deviation is a measure of how widely values within a list are dispersed from the average value (the mean). It uses the following formula:

$$\sqrt{\frac{\sum(value - mean)^2}{n}}$$

Where n is the number of items in the list, $value$ represents the value of the items in the list, and $mean$ represents the average of the values in the list. A large standard deviation usually implies the values in the list are widely dispersed, while a small standard deviation usually implies the values are tightly clustered around the mean. The standard deviation is the positive square root of Variance.

Sample applications of this formula

- Finding how much variation there is in the distance people need to walk to a transit stop.
- Part of a scientific model estimating fire risk in a neighborhood based on a large number of geographic factors.

Formula syntax

`StdDev` can be used in several ways.

Finding the standard deviation within a single attribute field

When you add the `StdDev` function to a formula, the program will display the following syntax:

    StdDev({attribute},
        Where({where}))

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

Example:
Use StdDev to find the standard deviation of lot sizes for lots that contain one existing dwelling unit (DU). You would write the syntax for this formula as:

```
StdDev ( [ Attribute:Parcel Zoning:ACRES ],Where ( [ Attribute:Parcel
Zoning:EXISTING DU ] = 1 ) )
```

Finding the standard deviation of numbers in a list
To find the standard deviation of the values in a list you provide, remove the "*{attribute}*,

Where(*{where}*) " from the StdDev( ) syntax and replace it with one or more numbers separated by commas. The function will use the values of all numbers in the list. The numbers can be regular numbers, indicators, or assumptions.

```
                          StdDev( number, number, number )
```

Examples:
```
StdDev( 23, 45, 280, 3 ) = 111.99
```

```
 StdDev( [Indicator:Spring Cost], [Indicator:Summer Cost],
[Indicator:Fall Cost], [Indicator:Winter Cost] ) = 217.33
```

Finding the standard deviation of several attribute values of each feature
When StdDev is used for several attributes of a feature, it can be thought of as finding the standard deviation of numbers in a row of the Attribute Table rather than finding the standard deviation of values in a column. For this purpose, the attribute names must be enclosed in their own parentheses:

```
 StdDev( ([Attribute:Value1]), ([Attribute:Value2]), ([Attribute:Value3]) )
```

Example:

```
StdDev( ( [Attribute:DistanceToLocalRoad] ), (
[Attribute:DistanceToArterialRoad] ), (
[Attribute:DistanceToCollectorRoad] ) )
```

Failure to include parentheses around each attribute name in the syntax for this form results in the formula evaluating values for the first record (feature) it encounters, which is rarely the intended result.

Type of value returned
Field Numeric

Sum function

Sum calculates the total of all values in a numeric attribute or the sum of all sizes in a shape attribute. The Sum is the whole amount, quantity, or number; it is an aggregate value.

If it is operating on a shape attribute, Sum adds the sizes of all the shapes. If Sum is operating on a numeric attribute, it adds all the values.

Sample applications of this formula
- Finding the total area of agricultural lands in a comprehensive plan
- Finding the total road mileage on a roads layer
- Finding the total population of a neighborhood

Formula syntax
When you add the Sum function to a formula, the program will display the following syntax:

```
Sum( {attribute},
     Where( {where} ) )
```

To complete the function, specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

Tip Use Sum to find the total cost of roads designated "Minor Rural". You would write the syntax for this formula as:

```
Sum ( [ Attribute:Road_Networks:COST ],
     Where ( [ Attribute:Road_Networks:DESIGNATION ] = "Minor Rural" ) )
```

Tip It is much more efficient to create an Area attribute, and sum it, rather than target a shape. The latter requires recomputing all shape areas each time it is evaluated.

Type of value returned
Field Numeric

T1F0 (True=1, False=0) function

The T1F0 converts a Boolean statement to a number; it is a conditional function. T1F0 returns 1 if the condition is true or returns 0 if the condition is false.

This function is convenient when using conditional tests in numerical formulas. Many numerical functions, such as Sum and Min, only work on numbers. It can also be used to multiply other numbers, and act as a switch.

In some cases, it may be easier to use a conditional where clause.

Formula syntax
When you add the T1F0 function to a formula, the program will display the following syntax:
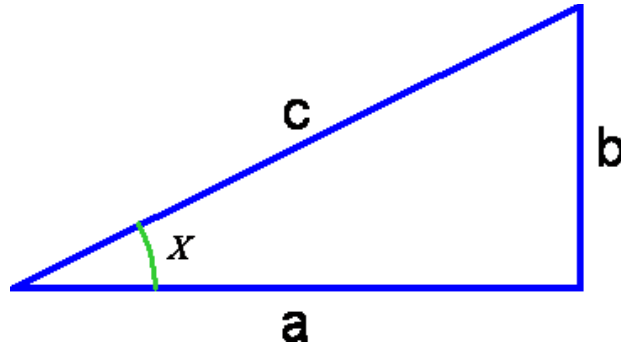
```
T1F0( )
```

To complete the function, insert a Boolean statement in the parenthesis.

Type of value returned
Numeric

Tan (Tangent) function

The `Tan` function calculates the tangent of an angle (in a right triangle the side opposite an angle divided by the side adjacent the angle). In the diagram below, the tangent of X is given by the ratio b/a. The units of X are assumed to be radians.

Tip If X is in degrees, multiply it by Pi/180 to convert it to radians. To get the value of Pi, type "Pi" in the formula.

Tip `Tan` and `Atan` are useful for converting between degrees of slope and percentage of slope or "rise over run." Usually a "25% slope" means that b/a = 0.25 in the figure above. The corresponding degrees of slope would be `Atan(b/a)` converted into degrees, or `Atan(b/a) *180/Pi`. If b/a = 25%, then X = 14 degrees. (Be aware, however, that sometimes people use the term percent slope to mean b/c. If so, use the Sin and Asin functions for conversions.)

To convert degrees of slope into percentage of slope, use `Slope-in-percent = Tan(Slope-in-degrees * Pi/180)`.

To convert slope in percentage to slope in degrees, use `Slope-in-degrees = Atan(slope-in-percent)*180/Pi`.

Formula syntax
When you add the `Tan` function to a formula, the program will display the following syntax:
    Tan( )

To complete the function, enter a value in radians in the parentheses.

Type of value returned
Numeric

Tanh (Hyperbolic Tangent) function

The `Tanh` function calculates the hyperbolic tangent of the angle.

The hyperbolic tangent of x is $(e^x - e^{-x})/(e^x + e^{-x})$. It is usually written as `Tanh`. The units of x are assumed to be radians.

Tip If x is in degrees, multiply it by Pi/180 to convert it to radians. To get the value of Pi, type "Pi" in the formula.

Formula syntax
When you add the `Tanh` function to a formula, the program will display the following syntax:
    Tanh( )

To complete the function, enter a value in radians in the parentheses.

Type of value returned
Numeric

ToNumber (Convert to Number) function

The `ToNumber` function converts a text string that represents a number to a number.

The typical application is converting a number that is FORMATTED as a string back to a number. If the input is text characters, the function returns a 0.

Examples:

`ToNumber`("Wristwatch") = 0

`ToNumber` ("123456") = 123456

`ToNumber` ([Attribute:Tree_Species]) = 0 (assuming "Tree Species" is a name).

Formula syntax
When you add the `ToNumber` function to a formula, the program will display the following syntax:
    ToNumber( )

To complete the function, enter a text attribute or text typed in quotes in the parentheses.

Type of value returned
Numeric

ToString (Convert to String) function

The `ToString` function converts a value to text format.

The typical application is converting a number into text format for further processing.

Formula syntax
When you add the `ToString` function to a formula, the program will display the following syntax:
    ToString( )

To complete the function, enter a number in the parentheses.

Type of value returned
String

Trim function

The `Truncate` function removes all leading and trailing spaces from a text value; in other words, it removes all spaces except spaces between words.

`Trim` converts " Lincoln Street " to "Lincoln Street".

Formula syntax
When you add the `Trim` function to a formula, the program will display the following syntax:
    Trim( )

To complete the function, enter a text value in the parentheses.

Type of value returned
String

Truncate function

The `Truncate` function truncates a number to a whole number by removing the decimal or fractional portion of the number.

`Truncate` converts 12.982 to 12.

Formula syntax
When you add the `Truncate` function to a formula, the program will display the following syntax:
    Truncate( )

To complete the function, enter a positive number in the parentheses.

Type of value returned
Numeric

UserChoice (User Choice) function

The `UserChoice` function presents the user with a prompt and a drop down box for the user to choose an attribute value from a list of valid values.

Inserting this function into a formula causes a dialog box, as shown below, to appear each time the user creates a new feature in the current layer.
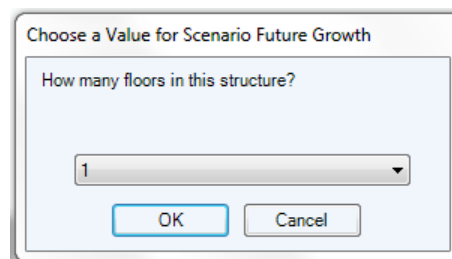
The text prompt "How many floors in this structure?", is specified in the function arguments, as are the values that appear in the drop-down box.

If the formula is being evaluated for more than one feature, the program will display the dialog box once and use the same value for each feature. Typically, the program will display the `UserChoice` prompt to collect attributes during the creation of a new feature.

Sample applications of this formula

- Allowing an user to specify properties of a new feature (such as a building, road, or land-use area) to the map
- Allowing a user to specify inputs to a scientific model (such as time of year, type of mitigation treatment, or implementation rule) being applied to an analysis

Formula syntax

When you add the `UserChoice` function to a formula, the program will display the following syntax:

```
UserChoice( "prompt", 1, 2, 3 )
```

To complete the function, replace `prompt` with the text you wish to appear on the prompt box. Next, replace the numbers `1, 2, 3` with the choices you wish to display. Any number of additional choices can be added; simply separate them with commas. The choices can be numbers or text, depending on the type of attribute being prompted. Text values must be enclosed in double quotes (e.g. "cat", "dog", etc.). All choices must be the same type.

For example, to create the prompt box shown above, type the following:

```
UserChoice( "How many floors in this structure?", 1, 2, 3, 4 )
```

You cannot use this function in indicator formulas.

Type of value returned

Numeric or text

UserChoiceGet (User Choice Get) function

The `UserChoiceGet` function prompts the user to select an attribute value from a list of valid values. The program collects (gets) the list of valid values from an attribute in another layer or table.

This function is like `UserChoice` except that the valid values come from an attribute in another layer or table. Inserting this function into a formula causes a dialog box, similar to the one below to appear when a user adds a feature to the current layer.
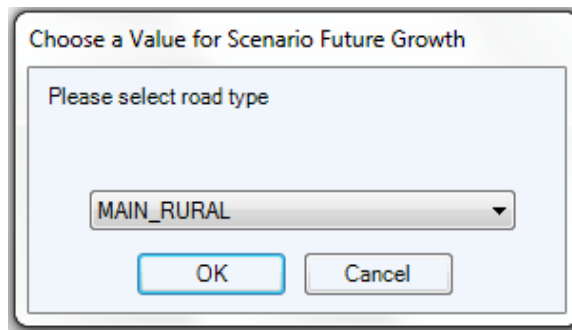
The text prompt that appears in the box, "Please select road type" is specified in the function. The values in the drop-down box are the current values of the specified attribute.

If the formula is being evaluated for more than one feature, the program will continue to display the dialog box prompting the user to select a value for the next feature.

Sample applications of this formula

- Allowing an user to specify properties of a new feature (such as a building, road, or land-use area) to the map.
- Allowing a user to specify inputs to a scientific model (such as time of year, type of mitigation treatment, or implementation rule) being applied to an analysis.

Formula syntax

When you add the `UserChoiceGet` function to a formula, the program will display the following syntax:

```
UserChoiceGet( "prompt",
    {attribute},
    Where( {where} ) )
```

To complete the function, replace `prompt` with the text you wish to appear on the prompt box. Specify an attribute by right-clicking on *{attribute}* and, optionally, specify a where clause by right-clicking on *{where}*. After clicking on *{attribute}*, the program will display a selection box with available attributes.

For example, to create the prompt box shown above, type the following syntax:

```
UserChoiceGet ( "Please select road type",
    [ Attribute:Road_Networks:DESIGNATION ] )
```
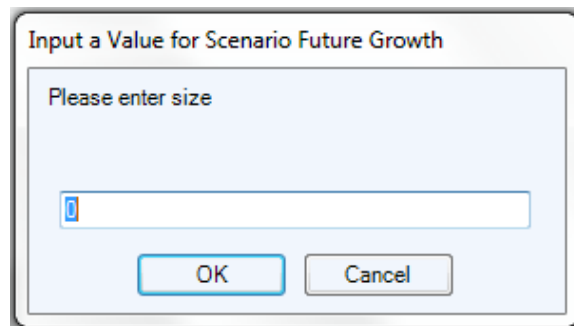
You cannot use this function in indicator formulas.

Type of value returned

Numeric, Text, or Boolean

UserInput (User Input) function

The `UserInput` function prompts the user to enter a numeric attribute value when adding features to a map.

This function generates a dialog box similar to the one below when a user adds a feature to the map while doing analysis.

The text "Please specify size" that appears in the prompt box is specified in the function. During analysis, the user will enter numbers to replace the "0" that the program displays by default.

This dialog box appears each time the user creates a feature.

Sample applications of this formula

- Allowing an user to specify numeric properties of a newly created feature, such as height of tree, depth of well, number of rooms in a building, or capacity of a wastewater treatment plant.

Formula syntax

When you add the `UserInput` function to a formula, the program will display the following syntax:

    UserInput("prompt", 0)

To complete the function, replace `prompt` with the text you wish to appear on the prompt box. You may change 0 to any other value you would like to serve as the default.

For example, to create the prompt box shown above, type the following syntax:

    UserInput ("Please specify size", 0)

You cannot use this function in indicator formulas.

Type of value returned

Numeric

UserInputB (User Input - Boolean) function

The `UserInputB` function prompts the user to select Yes or No for an attribute value when adding features to a map. This function is a Boolean input prompt attribute.

This function generates a dialog box like the one below when an user adds a feature to the map.
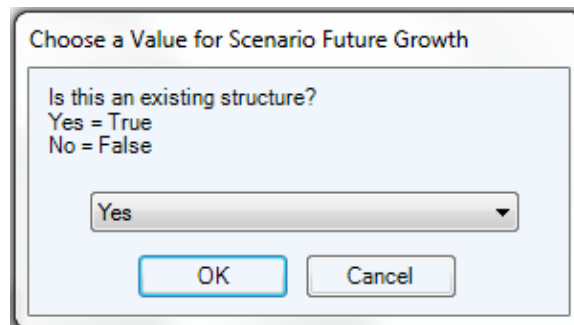
The text "Is this an existing structure? Yes = True No = False" that appears in the box is specified in the function. The user doing the analysis will choose True or False from the drop-down box.

This function is only used for attributes that are Boolean.

The dialog box appears after the user creates one feature, and then again if the user creates more features.

If you click the **Preview** button when editing this formula, the prompt will appear with the first ten records for the layer you are working on. Click the **Show Me More Results** button to view the next ten records. The program will display the prompt for each set of results.

Sample applications of this formula

- Allows the user to specify yes/no choices about a new feature such as new/existing, protected/not protected, or large/small.

Formula syntax

When you add the `UserInputB` function to a formula, the program will display the following syntax:

```
UserInputB( "prompt", True )
```

To complete the function, replace `prompt` with the text you wish to appear on the prompt box.

For example, to create the prompt box shown above, type the following syntax:

```
UserInputB ( "Is this an existing structure?
Yes = True
No = False", True )
```

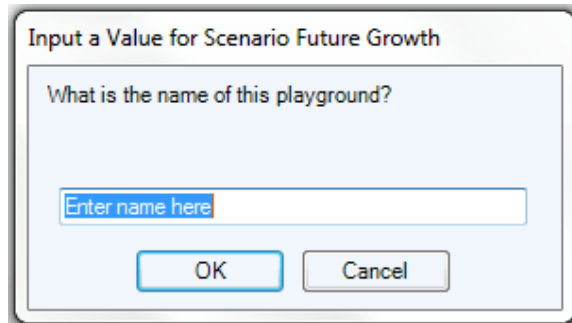You cannot use this function in indicator formulas.

Type of value returned

Boolean

UserInputS (User Input - String) function

The `UserInputS` function prompts the user to type a text value when adding features to a map.

This function generates a dialog box similar to the one below when the user doing the analysis adds a feature to the map.

The text "What is the name of this playground?" that appears in the prompt box is specified in the function.

The user doing the analysis types text into the dialog box. If the user presses "OK" without typing in alternate text, the function returns the default value (for example, "Enter name here").

This function is only used for attributes that are text.

The dialog box appears after the user creates one feature, and then again if the user creates more features.

If you click the **Preview** button when editing this formula, the prompt will appear with the first ten records for the layer you are working on. Click the **Show Me More Results** button to view the next ten records. The program will display the prompt for each set of results.

Sample applications of this formula
- Allowing an user to give a name or specify the name of an owner for a new feature

You cannot use this function in indicator formulas.

Formula syntax
When you add the *UserInputS* function to a formula, the program will display the following syntax:

```
UserInputS( "prompt", "default" )
```

To complete the function, replace `prompt` with the text you wish to appear on the prompt box.

For example, to create the prompt box shown above, type the following syntax:

```
UserInputS ( "What is the name of this playground?", "Enter name here" )
```

Type of value returned
Text

Var (Variance) function

Variance is a measurement of the differences of values from the expected average. `Var` calculates the variance of all values of a numeric attribute or the variance of all sizes of a shape attribute.

Variance is a measure of how widely values within a list are dispersed from the average value (the Mean). It uses the formula:

$$\frac{\sum(value - mean)^2}{n}$$

where n is the number of items in the list, *value* represents the value of the items in the list, and *mean* represents the average of the values in the list. A large variance usually implies the values in the list are widely dispersed, while a small variance usually implies the values are tightly clustered around the mean. Variance computed by `Var` is the square of standard deviation as calculated by the `StdDev` function.

For more information and examples, see **StdDev.**

Type of value returned
Numeric

WeightedMedian (Weighted Median) function

`WeightedMedian` identifies the middle value of a numeric attribute or the median size of a shape attribute, weighted by the value of another attribute in the same layer.

The median, or middle, value in a list of numbers is the center one when the values are listed in order by size. There are the same number of entries in the list that are larger than the median as there are entries that are smaller than the median. An equal number of values lie above and below the median value in a list. The `WeightedMedian` function effectively creates a long list in which the number of entries with the value of a particular feature's attribute value is given by the weighting value for that feature. Once that long list is created, its median is the value returned by the `WeightedMedian` function.

Sample applications of this formula
- Finding the median rate of attrition for public schools across several districts
- Finding the median household income in a metro area

Example:
Imagine a layer called "Parcels" which includes an attribute describing how many buildings are in each parcel and another attribute specifying how many windows are in each of those buildings.

Buildings	Windows_per_Building
1	20
5	15
2	40

The `WeightedMedian` function can be used to find the median number of windows per building across the entire layer using the formula

[Indicator:Median Windows per Building] = `WeightedMedian`(
[Attribute:Parcels:Windows_per_Building], [Attribute:Parcels:Buildings])

In effect, this formula creates a list like this:
20 15 15 15 15 15 40 40

In this list, the number of times "20" appears is given by the corresponding weighting factor of 1. The value "15" appears 5 times because its weighting factor is 5, etc. The median value of this list is 15, which is therefore the result given by the **WeightedMedian** function. Note that this result is different from the simple median of the Windows_per_Building attribute, which is 20.

Formula syntax
When you add the `WeightedMedian` function to a formula, the program will display the following syntax:

`WeightedMedian(` {*attribute*},

{*attribute*}, ' Weighting attribute

Where({*where*}))

The green text "Weighting attribute" after the apostrophe is simple a helpful comment telling you what the second attribute is.

To complete the function, specify an attribute by clicking on {attribute} and, optionally, specify a where clause by clicking on {where}. If you are not using the Where clause you can delete it by right-clicking on it and clicking Remove Where Clause on the pop-up menu. After clicking on {attribute}, the program will display a selection box with available attributes.

Type of value returned:
Numeric

Where function

`Where` is a special function that can only be used within other functions. It lets you specify one or more conditions for selecting which features or rows to include in a calculation.

Refer to the section called **Using 'where' conditions** in the chapter on formula syntax.

Some common functions that often include where clauses are Contains, MinDistance, IsSelected, and OverlapArea.

Glossary

A

assumption: An assumption is a value that is used as input to an analysis and it is often changeable. Assumptions apply to an entire scenario. For example, your assumptions about water consumption per household will impact the indicator for total water consumption for a scenario. Assumptions can also be a way to express subjective inputs, such as how much weighting to give to a particular community value like open space or economic development. Output values that depend on a particular assumption are automatically updated when the assumption is changed and you click the Apply button.

attribute: 1. A piece of information describing a map feature. The attributes of a census tract, for example, might include its area, population, and average per capita income. 2. A characteristic of a geographic feature described by numbers, characters, images, and CAD drawings, typically stored in tabular format and linked to the feature by a user-assigned identifier. For example, the attributes of a well might include depth and gallons per minute. 3. A column in a table.

B

Boolean: A logical statement, or a condition, that is either true or false. Often represented as yes/no or 1/0.

C

conditional operator: A symbol or keyword that specifies the relationship between two values. Examples include = (equal to), < (less than), > (greater than).

D

dynamic attribute: A dynamic attribute is an attribute that is automatically updated as changes are made in the analysis using the unique capabilities of Scenario 360. For example, a proposed road layer may contain dynamic attributes for length, pavement type, intersecting slopes, and construction costs. As each new road segment is added or modified, each of these dynamic attributes will be updated automatically. A formula is associated with each dynamic attribute that specifies how the attribute is calculated.

dynamic data layer: A dynamic data layer is a layer that is stored in your personal geodatabase. Only dynamic data layers can contain dynamic attributes. That is, a data layer must be designated as dynamic in order for you to be able to create a formula for any of its attributes.

F

feature: Features on a map are individual points, lines, or polygons. A typical map layer contains multiple features that together illustrate geographic information. In the 3D viewer, a feature is a shape that represents an actual or imaginary object such as a building, road, or tree.

fixed assumption: A fixed assumption is an input to the analysis that will not likely change, such as the municipal water supply. A fixed assumption value cannot be altered and has the same value across all scenarios.

I

indicator: Indicators are impact or performance measures. They can help people choose alternatives that best match their objectives or desired outcomes. Indicator values are automatically recalculated as you experiment with alternatives and the values can be displayed in a chart. Indicators apply to an entire scenario. An indicator might be used to evaluate costs, revenues, average household size, "community benefit", or total daily auto trips.

L

layer: Geographic information is displayed on a map as layers; each layer represents a particular type of feature such as streams, lakes, or highways.

line feature: A line connects two or more x,y coordinate pairs. Rivers and roads are both line features.

M

multi-point feature: A feature that consists of more than one point but only references one set of attributes in the database. For example, Hawaii, as there is a single set of attributes for multiple islands.

P

personal geodatabase: A geodatabase stored in Microsoft Access database format that supports many readers and a single editor. See also: geodatabase.

point feature: A single x,y coordinate pair representing a single geographic feature such as a tree.

polygon feature: An enclosed shape that may have any number of sides such as a shape representing a state.

S

spatial: Pertaining to space or a geographic area or the relationships between those geographic features. For example, distance, slope, or density.

syntax: The structure used for a formula, including the order of operations, the parentheses and commas required, and the element types allowed in each place.

T

target (formulas): In a formula, the target is the element that a function evaluates. Often attributes or features of the target layer, formally called target layer attributes or target layer features, are simply called target attributes or target features. Attribute formulas can be understood as taking the form, Current Attribute = Function operating on Target Layer.

V

variable assumption: A variable assumption is an input to the analysis that might change as part of the analysis, such as the current interest rate, seasonal resource consumption values, or

120

residential density. A variable assumption may be altered (using a slider bar or other method) during analysis, and can vary across scenarios.

W

weighting factor: A factor assigned to a number in a computation to make the number's effect on the computation reflect its importance.

where clause (formulas): "Where" conditions let you specify one or more conditions for selecting which features to include in a calculation. For example, to count the number of buildings over four stories tall, you would write a formula to count the number of houses "where" the number of stories is greater than four.