Creating an Iterative Macro - Transcript

Iterative Macros

In this lesson, we're going to explore the third and final type of macro in Designer: The Iterative Macro. Just like standard and batch macros, iterative macros are a group of tools or workflow, that have been packaged into a single tool. Standard macros run an entire incoming dataset through a single process. Batch macros can organize the data into groups and apply a distinct process to each group. An iterative macro on the other hand, runs the incoming dataset through a process but then feeds records back through the process over and over in a loop until a certain condition is met, or until a specified number of loops have been achieved.

Iterative macros rely on true/false logic somewhere in the process to determine when to stop running. For example, sending inventory to your highest volume store location until they are at capacity, then moving on to the next highest volume store until it is at capacity, and so on. Once a store location reaches capacity it exits the macro's output. The macro will loop until all stores are processed or until you run out of inventory. Iterative macros are extremely useful when you need to process records in a certain order, such as by priority or distance.

Another common example of iterative calculations are amortization tables for loans and investments. In a mortgage, for example, the next pay period's principal and interest amounts are determined by the balance from the previous pay period. Those calculations are run every month, until the end of the mortgage's term – for example, 360 months or iterations.

Use Case

For the purposes of this lesson, we're going to look at a very simple math problem to illustrate how iterative macros are built in Designer. The question is this: someone offers you a penny that will double in value every day for 30 days – OR – that person will give you 1 million dollars flat. Which option do you choose? With an iterative macro we can even look at other questions like what if the number of days, or the flat rate amount changes? When does one option become a better deal than the other?

Rendering Reports

From the Interface Tool Palette, drag a Macro Input tool onto the Canvas

Iterative macros are the only workflow type that will not be automatically configured by Designer. Dragging a Macro Input tool onto a canvas will convert the workflow to a standard macro. Adding a Control Parameter to that canvas will convert the workflow to a batch macro. Iterative macros have no such shortcut.

1) In the Canvas Workflow configuration, open the Macro dropdown

2) Select “Iterative Macro”

Setting your workflow to an Iterative Marco unlocks a new variable in Alteryx Designer: Iteration Number. This constant is very useful because it counts how many loops have been run. We’ll leverage this variable in an expression as we build the workflow.

Building the Macro

Since we just have one record to process, we can easily create our Template Input, rather than copying it from an Input Data tool. In the Macro Input tool’s configuration:

1) Click the “Text Input” radio button

2) Click “Edit Data”

You will recognize this as the same configuration as a standard Text Input tool. We’ll create a header called “Money,” and enter .01 in the first row – one penny. We’ll give the input a custom name: “Macro Input.” Next, drag a Formula tool onto the canvas and connect it to the Macro Input tool. Here, we’ll apply the expression [Money] * 2, since the penny will double in value every day.

In the same Formula tool, we’ll create a new expression and add a new column called “Day Number.” Since the number of loops in our macro represents one day, we can utilize that new variable we mentioned: [engine.iterationNumber]. This will count the number of times the dataset goes through the workflow, which can stand in as our number of days. Set the datatype to “Int16”. You’ll notice the default value for this variable is 0. Computers like to start counting at 0 but humans don’t, so we can add a simple “+1” to that variable, to make the day count more intuitive.

Finally, to make the results output easier to read, we’ll add a 3rd column with the same currency formatting expression we used in the standard macro lesson. Running the workflow shows that after one day, our penny is now two pennies, and we also have a nicely formatted dollar amount.

Creating a Loop

Next, connect a Filter tool to the Formula tool. We want our data to loop through this calculation until the number of days reaches 30, at which point the data should exit the workflow. We'll make a basic filter that passes the record through the true output when [Day Number] = 30. As mentioned, we want the data to exit the macro when it meets the condition or loop around until it does so. This means that we need two macro outputs.

Back in the Interface palette, drag two Macro Output tools onto the canvas

Because there are multiple output tools on the canvas, it's important to give each one a name – We'll call the first one "Exit," and the second, "Loop." Connect the true anchor from the Filter tool to the "Exit" Macro Output tool, and the false anchor to the "Loop" Macro Output. Dragging your loop Macro Output tool near your input can serve as a visual reminder of which output is which, but we also need to inform Designer of this distinction.

The Interface Designer

1) Open the Interface Designer window by clicking View > Interface Designer

2) Click on Settings (gear) icon

3) In the Iteration Input dropdown, select "Macro Input"

4) In the Iteration Output dropdown, select "Loop"

In this Properties window, you can also force Designer to stop after a maximum number of iterations and set what kinds of error messages display if that ceiling is hit. The default values are just fine for this example. With that, our macro is now complete! Save this workflow as "Penny Iterative Macro."

Results

In a new canvas, we have a Text Input tool that matches the Macro Input tool we just used. Insert the Penny Iterative Macro. After running the workflow, we can see the results! After 30 iterations, our initial penny has exponentially multiplied to over 10 million dollars! I think we have an answer to the original question - take the penny, not the million dollars.

Customization

This is a simple example of an iterative macro, but we can add as much complexity as we'd like! As with all macro types, we can add various interface elements to give the user additional control over the macro's configuration. For example, we could connect a Text Box to the Formula tool so the user can manually set the multiplier used in the Formula tool's expression. Another Text Box could be connected to the Filter tool so the user can set how many loops, or days the macro should calculate. By customizing your macro with interface options, you can easily find different values for any variation of this problem.

In the next lesson, we will finish this course with how to package, share, and deploy your macros.