Syntax Guide: Operators

Last Updated: Sep 27, 2016 03:03PM EDT

Designer has built-in operators that can be used in enabling conditions or validations.

Arithmetic Operators

The following table shows the arithmetic operators supported by Survey Solutions. Arithmetic expressions are evaluated from left to right. For the example, ***assume A = 20 and B = 10***

Operator	Description	Example
+	Addition: returns the sum	`A + B` = 35
-	Subtraction: returns the difference	`A - B` = 10
*	Multiplication: returns the product	`A * B` = 200
/	Division: returns the quotient	`B / A` = 2
%	Modulus: returns the remainder after an integer division	`B % A` = 0 (The remainder of 20 divided by 10 is 0)

Relational Operators

These operators compare two values and return either TRUE or FALSE. The following comparison operators are supported by Survey Solutions:

Operator	Description	Example
==	Checks if the two values are *equal*. If values are equal, then the condition is true	`20 == 10` returns **FALSE**
!=	Checks if the two values are *not equal*. If the values are not equal, then the condition is false	`20 != 10` returns **TRUE**
>	Checks if the left value is *greater than* the right value	`20 > 10` returns **TRUE**
<	Checks if the left value is *less than* the right value	`20 < 10` returns **FALSE**
>=	Checks if the left value is *greater than or equal to* the right value	`20 >= 10` returns **TRUE**. `10 >= 10` returns **TRUE**.
<=	Checks if the left value is *less than or equal to* the right value	`20 <= 10` returns **FALSE**. `10 <= 10` returns **TRUE**.

Logical Operators

Logical values returns either **TRUE** or **FALSE**. The following tables show the logical operators supported by Survey Solutions:

Operator	Description	Example
&&	Logical AND operator. A && B returns true of *both* expression A and B are true.	`(20 == 10) && (10 > 5)` returns **FALSE** `(10 == 10) && (10 > 5)` returns **TRUE**
||	Logical OR operator A || B returns true if *either* expression A or expression B is true	`(20 == 10) || (10 > 5)` returns **TRUE** `(20 == 10) || (10 < 5)` returns **FALSE**
!	Logical NOT operator !A returns true if expression A is false. It returns false if expression A is true.	`!(20 == 10)` returns **TRUE** `!(10 == 10)` returns **FALSE**

Syntax Guide: System Generated Variables

self

For validation conditions, a very useful system generate variable is `self`. This variable denotes the value of the question being validated. Using `self` instead of the question's variable name in a question's validation conditions is advised because you will not have to change the variable name in the validation condition if the variable name for the question is changed.

Functions by question type

Numeric

- Relational Operators: less than, greater than, equal to, etc

Single-select

- Relational Operators less than, greater than, equal to, etc.

Text

- Length: *Checks the number of characters*

List

- Length.*Checks the number of items listed*

Date

- Relational Operators. *Less than, greater than, equal to, etc*
- Check against current date *Less than, greater than, equal to, etc. Example, BirthDate < (new DateTime ())*

Multi-select:

- Contains. Checks whether the answers to question contain a specified value.
- ContainsAll. Check whether the answers contain all of the specified values.
- ContainsOnly. Contains only the specified values
- ContainsAny. Contains at least one of the specified values
- !ContainsAny. Contains none of the specified values.
- Length. Computes the number of answers to a multi-select question.

Advanced

Other Useful Operators

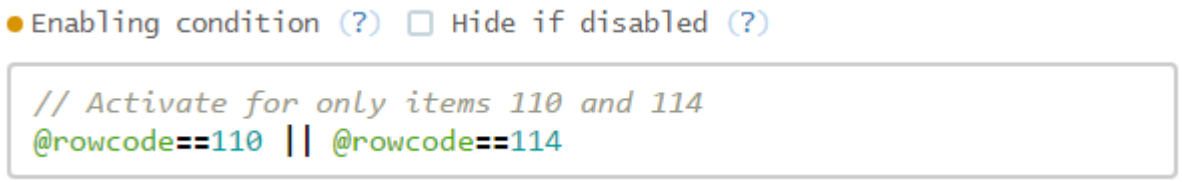
These are other operators that you might want to use in your Survey Solutions instrument.

Operator	Description	Example
?:	Conditional Expression/ The condition must evaluate to true or false. If condition is true, the *first expression* is evaluated and becomes the result. If condition is false, the *second expression* is evaluated and becomes the result. Only one of the two expressions is evaluated.	`(10 < 2)? a:b` will return b `(10 > 2)? a:b` will return a

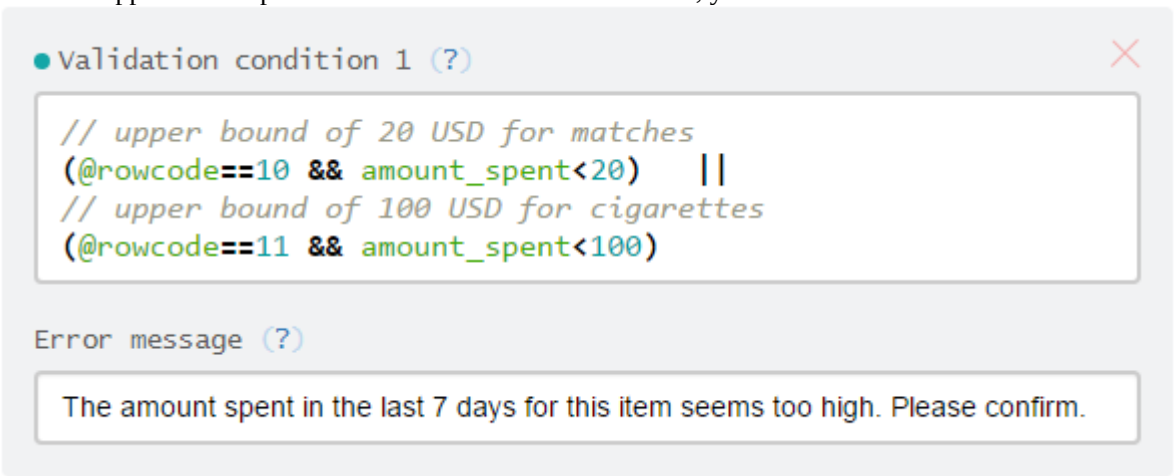
@rowcode

This system generated variable allows you to reference specific rows within a roster. This variable can be used in enabling conditions and validations. Additionally, @rowcode can also be used to refer to certain rows in a look up table.

Example 1:
Assume you have roster of items and you have a question that should only be asked of two of the items (item #110 and item #114). In this case, you would use `@rowcode` to refer to the item number. Then you would use `@rowcode` and write the enabling condition to be like below:



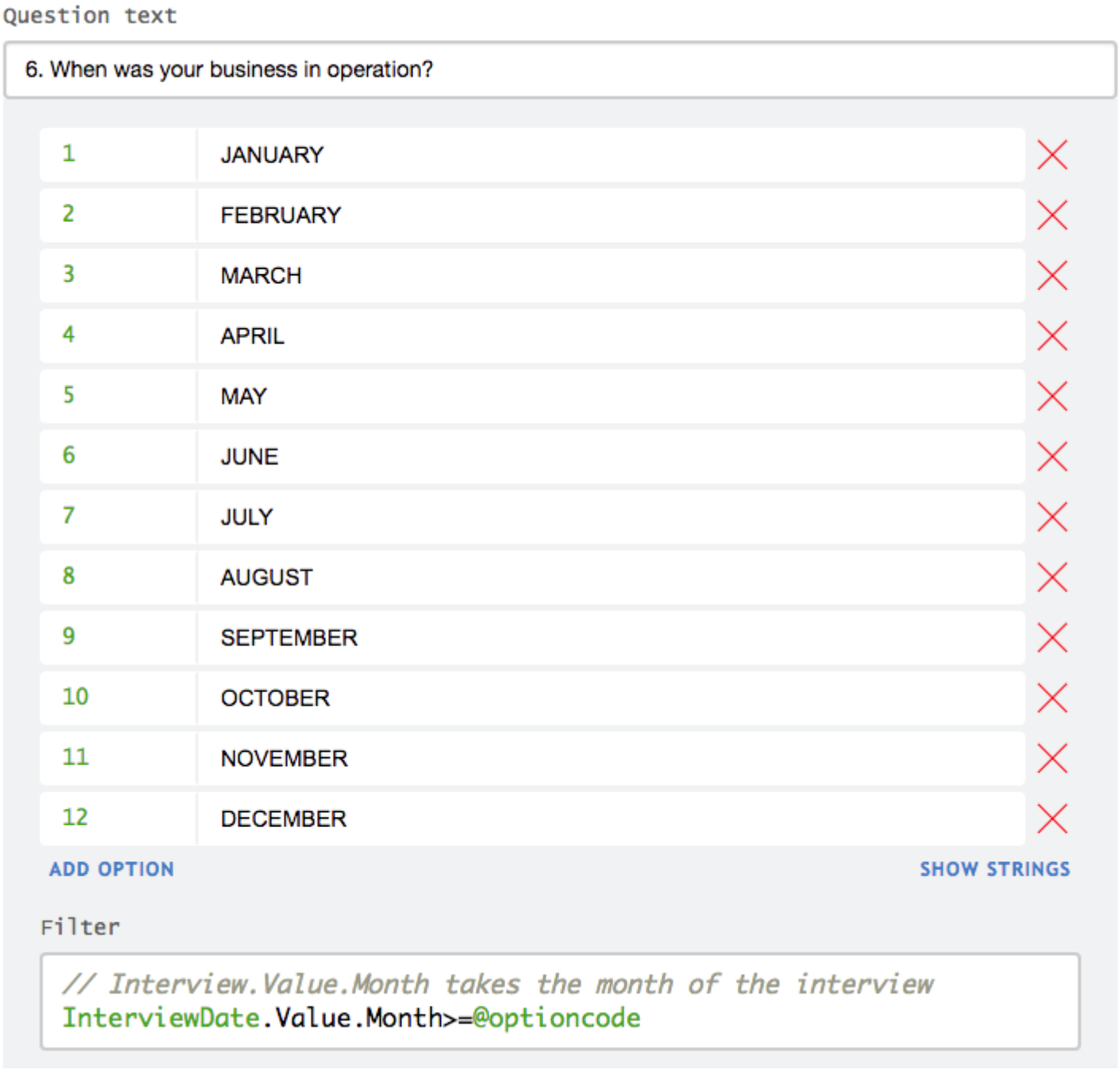
Example 2:
Assume you have a roster of items that the household could have bought in the last 7 days. You want to check that the amount of money spent is not too high. Two of the items on the list are matches (item# 10) and cigarettes (item #11). Cigarettes are more expensive than matches so the largest amount that could be spent would be different for each item. Assume that you want an upper bound of 20 USD for matches and 100 USD for cigarettes. You would use `@rowcode` to code the upper bound specific to that item. To code this check, you would write the validation condition to be like below:



@optioncode

This system generated variable, @optioncode, is used to filter answer options for single select questions and multi-select questions. @optioncode refers to the numeric code for each answer option.

Example:
Assume you have a question that asks when in the past calendar year a business has been in operation. You want to filter the answer options so that only the months in the past appears. For example, if the interviewer occurs in October, you do not want the month November and December to appear because those are months in the future. Assume you have a question that captures the the date of the interview (`InterviewDate`). You will use date time functions to code this filter. You should write the filter to look like below:



The code`InterviewDate.Value.Month`takes the numeric month value from the InterviewDate variable. The filter will be evaluated for every option code and it will only display the options that it evaluate to true. In this case, months in the future will evaluate to false and will not appear as an answer option.

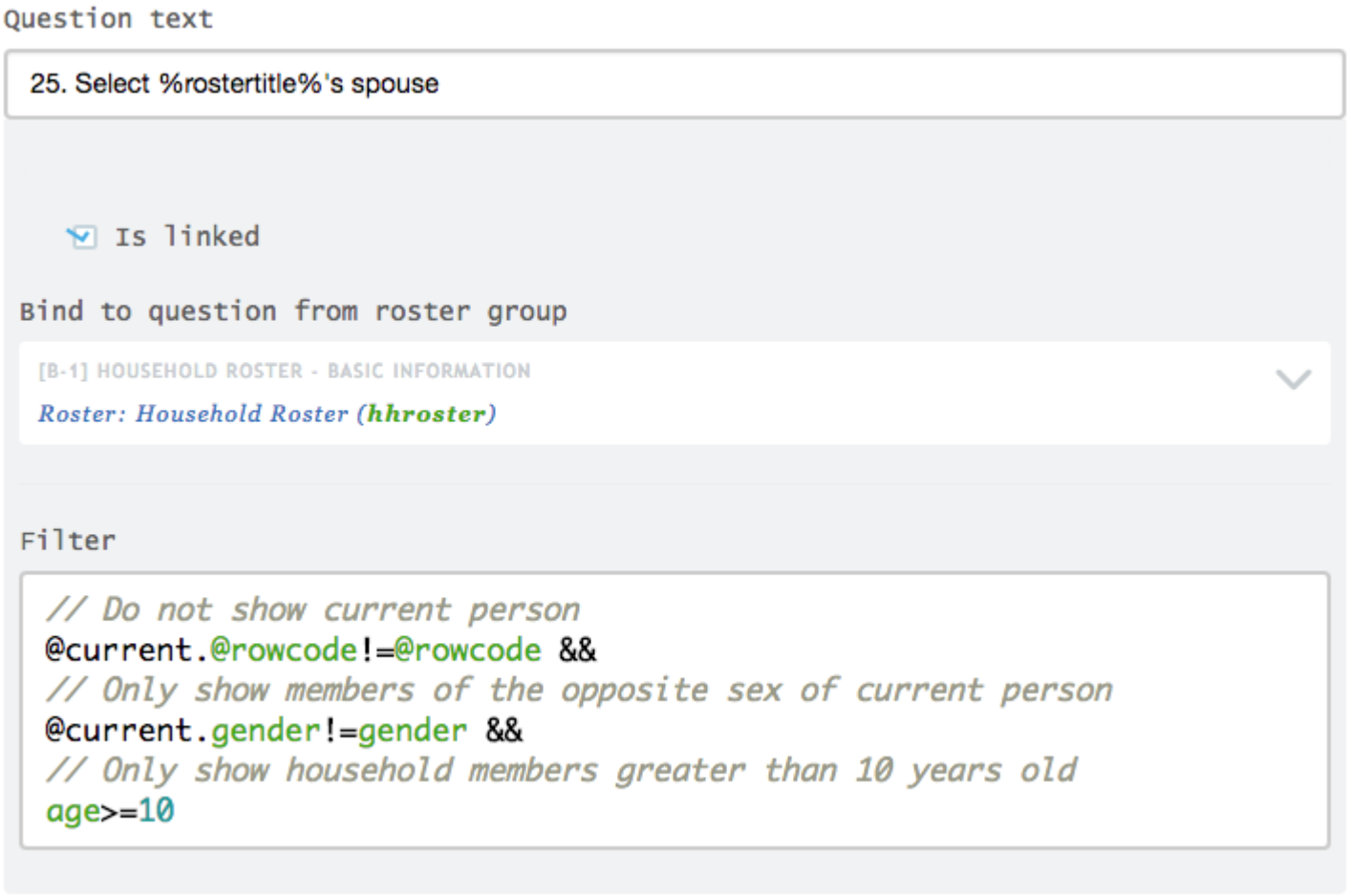
@current

This system generated variable, @current, is very useful in code to filter answer options. @current refers to the current line or occurence in the roster. For example, @current refers to the current household member that you are on in the household roster.

Example:
Assume you have a question that asks the interviewer to select the ID of the spouse of the current household member. You want to filter the answer options based on the following criteria:

- Do not show the current person
- Only show household members of the opposite gender
- Only show household members 10 years old or greater

This question is a single select linked to the household roster. For this example, assume that the variable for the gender of the household member is`gender`. You would write the filter for the answer options to look like this:



To see an example of this code in the public questionnaire, see this question.