IntelliSense.

- Down,
- Up Down,
- Enter,

 SHIFT+ENTER

SHIFT Enter
Code:

```plaintext
>    Close>Open
<    Open[i]<Open[i-1]
>=    Close=Open
<=    Close<=EMA

==true  ==true
==false  l==false
!=true    !=true
!=false   !=false
```

Code:

```
&&
Close>Open && Open[i]<Open[i-1]
```

Code:

```
Open+Close
EMA-EMA1
Close*
```

Code:

```
EMA>Close?((Close+Open)/2):Open
```

Code:

```
Close[i-1]
```
Code:
Close[i-8]>Close[i-9] && Close[i-9]>

, 9. == 9.

EMA[i-] EMA[i-], /, ( ). " " . 

: * Close[i] Close


, . . Math .:

* *

* . ..

* *

* ..

? :

Code:
Math._() 

. (EMA-EMA1).

Code:
Math.Abs(EMA-EMA1)

- , ,

TSLab / ?

Math


Math  Up  Down, .

Enter, .
Boolean expression
It is used to verify the truth of an expression. The output of the boolean expression is always 0 ("false") or 1 ("true"). It is usually used to record the entry/exit conditions of a position. Let's have a look at the syntax that can be used in the Boolean expression block.

In the "Boolean expression" block, the following comparison operations can be used (please see examples for each operation):

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>More</td>
<td>&gt;</td>
<td>Close&gt;Open</td>
</tr>
<tr>
<td>Less</td>
<td>&lt;</td>
<td>Open[i]&lt;Open[i-1]</td>
</tr>
<tr>
<td>More or equal</td>
<td>&gt;=</td>
<td>Close&gt;=Open</td>
</tr>
<tr>
<td>Less or equal</td>
<td>&lt;=</td>
<td>Close&lt;=EMA</td>
</tr>
<tr>
<td>Equal ?</td>
<td>==</td>
<td>Close==High</td>
</tr>
<tr>
<td>Equal to true</td>
<td>==true</td>
<td>Boolean expression==true</td>
</tr>
<tr>
<td>Equal to false</td>
<td>==false</td>
<td>Boolean expression==false</td>
</tr>
<tr>
<td>Not equal</td>
<td>!=</td>
<td>Close!=High</td>
</tr>
<tr>
<td>Not equal to true</td>
<td>!=true</td>
<td>Boolean expression!=true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>!Boolean expression</td>
</tr>
</tbody>
</table>
In addition to the comparison operations in Boolean expression block, you can simultaneously check other conditions as well. These conditions must be combined by the following operators:

**Code:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean «And»</td>
<td>&amp;&amp;</td>
<td>Close&gt;Open &amp;&amp; Open[i]&gt;Open[i-1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Close&gt;Open &amp; Open&lt;Open[-1]</td>
</tr>
<tr>
<td>Boolean «Or»</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EMA==EMA1</td>
</tr>
</tbody>
</table>

When using “AND”, the output of the Boolean expression block will be true only if all conditions recorded in the block are true. When using “OR”, the output of the block will be true if at least one of the conditions recorded in the block is true.

**IF THEN ELSE**

(Open>Open[-1])? (Close+Open)/2 : Open

Note. In the Boolean expression block it is possible to use mathematical functions: Addition, Subtraction, Multiplication, Division.

**Formula**

Used to calculate expression values. The output is always a numeric value. In practice, it is usually used for intermediate calculations, as well as for calculating entry/exit levels of a position.

The following operations are allowed in the Formula block:

**Code:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>Open+Close</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>EMA-EMA1</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>Close*Constant</td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>Close/Close[-1]</td>
</tr>
</tbody>
</table>

IF THEN ELSE The Formula block can use the boolean condition If. The result of it is still a number. The entry of the boolean condition If in the Formula block looks like this:

**Code:**

Logical_condition? Expression, _if_true: Expression, _if_false

As a result of checking the Logical condition, if it is true, at the output of the Formula block there will be “Expression,_if_true,” otherwise, “Expression,_if_false”.

**Example:**

**Code:**

EMA>Close ? ((Close+Open)/2) : Open
EMA>Close ? ((Close+Open)/2) : EMA>High ? ((Close+High)/2) : Open

**Common Features of Boolean expression and Formula Blocks**

1. The ability to work with the previous values of a certain sequence (price, indicator, etc.). In these blocks you can refer to previous values of prices or indicators. The current value is always indicated by the index i. In order to refer, for example, to the previous value of the close price, you should write

**Code:**
If you use the previous elements in calculations and checks, you should not forget about Start With parameter. The easiest way to remember what to put in the Start With field is to find out the maximum number used in square brackets and subtracted from the i index.

Example:

Code:


Looking at this entry, we see that the maximum value of the number in square brackets is 9. Therefore, in “Begin With” we put 9.

Why, when referring to the previous elements, do we need to change the value in Begin With? Everything is simple. A sequence of values has a beginning or a frontier. There is nothing beyond the beginning; it is empty. All calculations are based on the current value of i, therefore, when we start to count the value for \( i = 0 \), i.e. for the first value of the sequence, and we try to refer to the previous value, TSLab says that it is empty and cannot calculate it. However, when the recording is simple, for example \([i-9]\), it is not necessary to use Begin With, TSLab can detect it automatically.

When the EMA \([\text{Constant}]\) or EMA \([\text{Formula}]\) is used, **Begin With must be used**, it must be certainly larger than the Constant or Formula / indicator, but no more than the loaded bars in the script (or agent). Also, in the properties of the script, we recommend using the same value in “Trade with a bar”.

Axiom:

• \( \text{Close}[i] \) is equal to \( \text{Close} \)

2. The ability to use the functions of the Math library. It is possible to use functions from the Math library in expressions written in the Formula and Boolean expression blocks. A list of functions and their description is provided on the MSDN website: [http://msdn.microsoft.com/en-us/library/xaz41263.aspx](http://msdn.microsoft.com/en-us/library/xaz41263.aspx)

We need these functions when fundamental mathematical operations are not enough. The Math library functions allow to:

• Calculate an absolute value

• Raise a number to a power

• Calculate sine, cosine, etc.

• Round a number

• Calculate square root

• Etc.

How can these functions be used in the blocks? Use this to calculate a value with the help of the function:

Code:

\[ \text{Math.Function}_{\text{name}}(\text{Expression}) \]

Example. Calculation the absolute value for the (EMA-EMA1) expression:

Code:

\[ \text{Math.Abs}(\text{EMA}-\text{EMA1}) \]

Using other functions is similar, there can be difference in the number of function parameters that are indicated in brackets).

Does TSLab have a rounding up / down function?

There is a feature of the Formula block, when working in calculator mode:

For example, if you write

\[ \frac{13}{3} \]

it is expected that when the expression is displayed on the graph, we get 4.33.

But the expression will show 4.

Divide the integer by an integer, we get an integer.

Life hack: add .0 to the numerator or denominator .0

So write

\[ \frac{13}{3.0} \]

, not \[ \frac{13}{3} \]

The expression will return the close value of the first bar in the loaded history:

Code:

\[ \text{Close[-i]} \]

**String expression**

The syntax is the same, but returns words.

Example

\[ \text{Close > Open? "UP" : "Down"} \]

display string in the control panel.

when the bar is up, in the control panel there will be an inscription UP.

**Attachments**

Sample_logical_formula.xml (1589 downloads)
Sample_formula.xml (949 downloads)
Sample_logical_formula & formula_basic.xml (1083 downloads)