

Language Elements

PowerLanguage is a system of expressing financial instrument trading rules in a systematic and logical way that can be executed by a computer.

A well-thought-out trading strategy, properly expressed in PowerLanguage, will be carried out with a far greater speed, accuracy, and persistence than could ever be possible for a human trader.

Implementing trading strategies in PowerLanguage requires familiarity with the basic rules and structure of the language, outlined below, as well as with the keywords described in the keyword section of this reference guide.

PowerLanguage Studies and Statements

A complete PowerLanguage "program" is called a **Script**. An example of a script would be a Strategy, a study that issues trading orders, an Indicator, a study that draws plots on a chart to assist in making trading decisions, or a Function, an independent procedure (subroutine) that can be called from another script to carry out a specific task.

PowerLanguage scripts consist of at least one, but usually more, **statements**. A statement is a complete instruction, and ends with a semi-column (;).

An example of a statement (and of a one-line script):

```
Buy Next Bar At Open;
```

Statements are evaluated from left to right, and scripts are evaluated from top to bottom.

Price Charts

A data series consists of groups of price data points, based on a defined interval and arranged in a chronological order.

The most popular format for visually presenting a data series is a Bar Chart. Each bar, based on a group of price data points, is a vertical line connecting the High and the Low price points, and graphically represents the range of an instrument's price movement over a defined interval. The prices of the first (Open) and of the last (Close) price points of the group, on which the bar is based, are indicated on the line by marks known as the bar's Open and Close components.

PowerLanguage handles price data "packaged" in the form of bars. While bars can be based on a variety of resolutions, the point of reference is always a "bar".

For example, the following statement places a Buy order at the Open price of the next bar if the previous bar's High was less than the current bar's Close:

```
If High 1 Bar Ago < Close Then Buy Next Bar At Open;
```

Both time and price in the above statement are referenced in terms of bars.

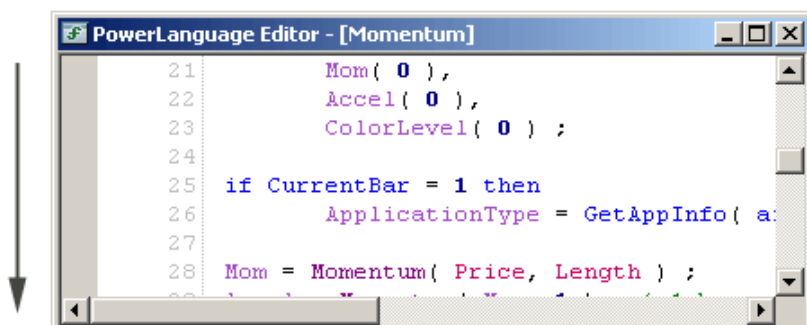
Price Chart Evaluation by PowerLanguage Studies

A PowerLanguage script evaluates a chart on bar-by-bar basis, starting with the very first (oldest) bar on the chart. The exact same script is executed for every bar.

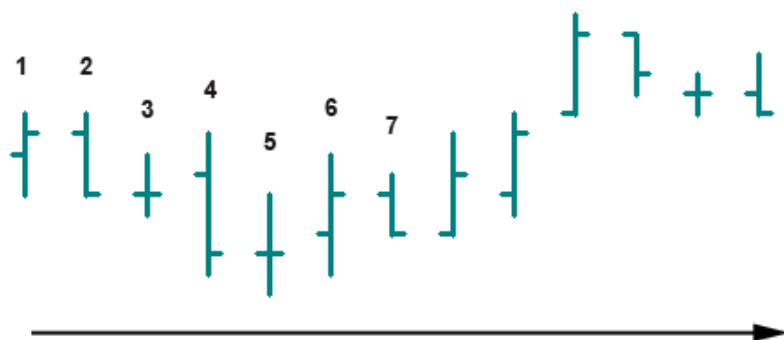
After the entire script is executed for the first bar, the second bar is evaluated, and so on to the last bar on the chart.

The bar being currently evaluated by the study is defined as the current bar, and the previous bar is the 1 Bar Ago. The current bar becomes the 1 Bar Ago once the study moves on to the next bar.

The study's entire script is executed from top to bottom for each and every bar



A study evaluates bars one-by-one, starting with the very first (oldest) bar on the chart (1, 2, 3...n)



This a basic overview is intended to give the initial understanding of how PowerLanguage studies function.

A more detailed description of how a chart is evaluated is presented in the How Scripts Work section.

PowerLanguage

Power language allows a combination of keywords, constants, variables, mathematical and logical operators, and punctuation to be used to express trading rules and bring about both conditional as well as unconditional actions and output.

There are tools to accomplish a variety of tasks, from the very basic, such as:

```
Print("My name is Joe");
```

all the way to array functions available in the advanced programming languages.

Each category of PowerLanguage components is discussed below:

Keywords

Most of the keywords are the instructions, "action words", of PowerLanguage.

Keywords instruct to buy or sell, plot or erase, compare or convert.

Some of the keywords are also "nouns", and return specific values or characteristics.

A list of keywords with detailed descriptions and examples is available in the Keyword Reference

Skip Words

Skip words are the interjections of PowerLanguage. They actually do nothing, and are skipped during code execution, but they make PowerLanguage easier to read.

In PowerLanguage Editor, skip words appear in red. A list of these words is available in the Skip Words section of this guide.

Operators

There are four types of operators: Mathematical, Relational, Logical, and String.

Mathematical operators are used to perform mathematical operations. There are five available mathematical operators:

+	Addition
-	Subtraction
*	Multiplication
/	Division
()	Parentheses

Relational operators are used to make comparisons. There are six relational operators:

<	Less than
>	Greater than
<=	Less than or equal to
=>	Greater than or equal to
=	Equal to
<>	Not equal to

Logical operators are used to perform logical (Boolean) operations. There are two available logical operators: AND and OR.

There is only one string operator, the plus sign (+), is used as follows to combine multiple string expressions in to one:

"String expression one" + "and string expression two"

The resulting string expression will be:

"String expression one and string expression two"

Referencing previous bars

Previous bars' values can be referenced by using the statement `N Bars Ago`, or by using the bar offset notation that consists of a numerical expression enclosed in square brackets:

`High 2 Bars Ago`

or

`High [2]`

Time and Date

Time values in PowerLanguage are specified in the 24-hour HHmm or HHmmss formats, where 1300 = 1:00 PM and 130000 = 1:00:00 PM, respectively.

Dates in PowerLanguage are specified in the YYYYMMdd format, where YYYY is the number of years since 1900, MM is the month, and dd is the day of the month.

This format is also known as the "EL Date" format.

Variables

Variables are used to store numerical, string, or logical (true/false) values. The value stored in a variable can be referenced throughout a script by the variable's name, and can be modified by the script at any time. Variables must be declared before use.

For more information see [Variable](#)

Arrays

Arrays are multiple-element variables.

For more information see [Array](#)

Article Sources and Contributors

Language Elements *Source:* <http://www.multicharts.com/trading-software/index.php?oldid=7376> *Contributors:* Admin, Henry Multicharts, Roman MultiCharts

Image Sources, Licenses and Contributors

File:bars.gif *Source:* <http://www.multicharts.com/trading-software/index.php?title=File:Bars.gif> *License:* unknown *Contributors:* Roman MultiCharts