

# Language

- [Execution model](#)
  - [Calculation based on historical bars](#)
  - [Calculation based on realtime bars](#)
  - [Events triggering the execution of a script](#)
  - [More information](#)
  - [Execution of Pine Script™ functions and historical context inside function blocks](#)
    - [Why this behavior?](#)
    - [Exceptions](#)
- [Time series](#)
- [Script structure](#)
  - [Version](#)
  - [Declaration statement](#)
  - [Code](#)
  - [Comments](#)
  - [Line wrapping](#)
  - [Compiler annotations](#)
- [Identifiers](#)
- [Operators](#)
  - [Introduction](#)
  - [Arithmetic operators](#)
  - [Comparison operators](#)
  - [Logical operators](#)
  - [`?:` ternary operator](#)
  - [`\[\]` history-referencing operator](#)
  - [Operator precedence](#)
  - [`= ` assignment operator](#)
  - [`:=` reassignment operator](#)
- [Variable declarations](#)
  - [Introduction](#)
    - [Initialization with `na`](#)
    - [Tuple declarations](#)
  - [Variable reassignment](#)
  - [Declaration modes](#)
    - [On each bar](#)
    - [`var`](#)
    - [`varip`](#)
- [Conditional structures](#)
  - [Introduction](#)
  - [`if` structure](#)
    - [`if` used for its side effects](#)
    - [`if` used to return a value](#)
  - [`switch` structure](#)
    - [`switch` with an expression](#)
    - [`switch` without an expression](#)
  - [Matching local block type requirement](#)
- [Loops](#)
  - [Introduction](#)

- When loops are not needed
  - When loops are necessary
- `for`
- `while`
- Type system
  - Introduction
    - Forms
    - Types
  - Using forms and types
    - Forms
    - Types
  - `na` value
  - Type templates
  - Type casting
  - Tuples
- Built-ins
  - Introduction
  - Built-in variables
  - Built-in functions
- User-defined functions
  - Introduction
  - Single-line functions
  - Multi-line functions
  - Scopes in the script
  - Functions that return multiple results
  - Limitations
- Arrays
  - Introduction
  - Declaring arrays
    - Using the `var` keyword
  - Reading and writing array values
  - Looping through array elements
  - Scope
  - History referencing
  - Inserting and removing array elements
    - Inserting
    - Removing
    - Using an array as a stack
    - Using an array as a queue
  - Calculations on arrays
  - Manipulating arrays
    - Concatenation
    - Copying
    - Joining
    - Sorting
    - Reversing
    - Slicing
  - Searching arrays
  - Error handling
    - Index xx is out of bounds. Array size is yy
    - Cannot call array methods when ID of array is `na`
    - Array is too large. Maximum size is 100000
    - Cannot create an array with a negative size
    - Cannot use shift() if array is empty.

- Cannot use pop() if array is empty.
- Index 'from' should be less than index 'to'
- Slice is out of bounds of the parent array

- [Objects](#)

- [Introduction](#)
- [Creating objects](#)
- [Changing field values](#)
- [Collecting objects](#)
- [Copying objects](#)
- [Shadowing](#)

- [Methods](#)

- [Introduction](#)
- [Built-in methods](#)
- [User-defined methods](#)
- [Method overloading](#)
- [Advanced example](#)

---

[Next steps](#)[Execution model](#)

Options

v: v5

© Copyright 2023, TradingView.