Pine Script v6 Reference Manual

This document serves as a comprehensive reference guide for Pine Script version 6, the latest iteration of TradingView's proprietary programming language for creating custom indicators and trading strategies.

## bar_index
Current bar index. Numbering is zero-based, index of the first bar is 0.

### Type
series int

### Example
Note that bar_index has replaced n variable in version 4.

### Remarks
Note that bar_index has replaced n variable in version 4.

### See also
Returns true if the script is calculating the last (closing) update of the current bar. The next script calculation will be on the new bar data.

### See also
- last_bar_index
- barstate.isfirst
- barstate.islast
- barstate.isrealtime

## barstate.isconfirmed
Returns true if the script is calculating the last (closing) update of the current bar. The next script calculation will be on the new bar data.

### Type
series bool

### Remarks
Pine Script® code that uses this variable could calculate differently on history and real-time data.

### See also
Returns true if current bar is first bar in barset, false otherwise.

### See also
- barstate.isfirst
- barstate.islast
- barstate.ishistory
- barstate.isrealtime
- barstate.isnew
- barstate.islastconfirmedhistory

## barstate.isfirst
Returns true if current bar is first bar in barset, false otherwise.

### Type
series bool

### Remarks
Pine Script® code that uses this variable could calculate differently on history and real-time data.

### See also
Returns true if current bar is a historical bar, false otherwise.

### See also
- barstate.islast
- barstate.ishistory
- barstate.isrealtime
- barstate.isnew
- barstate.isconfirmed
- barstate.islastconfirmedhistory

## barstate.ishistory
Returns true if current bar is a historical bar, false otherwise.

### Type
series bool

### Remarks
Pine Script® code that uses this variable could calculate differently on history and real-time data.

### See also
Returns true if current bar is the last bar in barset, false otherwise. This condition is true for all real-time bars in barset.

### See also
- barstate.isfirst
- barstate.islast
- barstate.isrealtime
- barstate.isnew
- barstate.isconfirmed
- barstate.islastconfirmedhistory

## barstate.islast
Returns true if current bar is the last bar in barset, false otherwise. This condition is true for all real-time bars in barset.

### Type
series bool

### Remarks
Pine Script® code that uses this variable could calculate differently on history and real-time data.

### See also
Returns true if script is executing on the dataset's last bar when market is closed, or script is executing on the bar immediately preceding the real-time bar, if market is open. Returns false otherwise.

### See also
- barstate.isfirst
- barstate.ishistory
- barstate.isrealtime
- barstate.isnew
- barstate.isconfirmed
- barstate.islastconfirmedhistory

## barstate.islastconfirmedhistory
Returns true if script is executing on the dataset's last bar when market is closed, or script is executing on the bar immediately preceding the real-time bar, if market is open. Returns false otherwise.

### Type
series bool

### Remarks
Pine Script® code that uses this variable could calculate differently on history and real-time data.

### See also
Returns true if script is currently calculating on new bar, false otherwise. This variable is true when calculating on historical bars or on first update of a newly generated real-time bar.

### See also
- barstate.isfirst
- barstate.islast
- barstate.ishistory
- barstate.isrealtime
- barstate.isnew

## barstate.isnew
Returns true if script is currently calculating on new bar, false otherwise. This variable is true when calculating on historical bars or on first update of a newly generated real-time bar.

### Type
series bool

### Remarks
Pine Script® code that uses this variable could calculate differently on history and real-time data.

### See also
Returns true if current bar is a real-time bar, false otherwise.

### See also
- barstate.isfirst
- barstate.islast
- barstate.ishistory
- barstate.isrealtime
- barstate.isconfirmed
- barstate.islastconfirmedhistory

## barstate.isrealtime
Returns true if current bar is a real-time bar, false otherwise.

### Type
series bool

### Remarks
Pine Script® code that uses this variable could calculate differently on history and real-time data.

### See also
Returns an array filled with all the current boxes drawn by the script.

### See also
- barstate.isfirst
- barstate.islast
- barstate.ishistory
- barstate.isnew
- barstate.isconfirmed
- barstate.islastconfirmedhistory

## box.all
Returns an array filled with all the current boxes drawn by the script.

### Type
array<box>

### Example
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### Remarks
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### See also
Returns the color of the chart's background from the "Chart settings/Appearance/Background" field. When a gradient is selected, the middle point of the gradient is returned.

### See also
- box.new
- line.all
- label.all
- table.all

## chart.bg_color
Returns the color of the chart's background from the "Chart settings/Appearance/Background" field. When a gradient is selected, the middle point of the gradient is returned.

### Type
input color

### See also
Returns a color providing optimal contrast with chart.bg_color.

### See also
- chart.fg_color

## chart.fg_color
Returns a color providing optimal contrast with chart.bg_color.

### Type
input color

### See also

simple bool

### See also
- chart.bg_color

## chart.is_heikinashi
simple bool

### Type
simple bool

### Returns
```
Returns true if the chart type is Heikin Ashi, false otherwise.
```

### See also
simple bool

### See also
- chart.is_renko
- chart.is_linebreak
- chart.is_kagi
- chart.is_pnf
- chart.is_range

## chart.is_kagi
simple bool

### Type
simple bool

### Returns
```
Returns true if the chart type is Kagi, false otherwise.
```

### See also
simple bool

### See also
- chart.is_renko
- chart.is_linebreak
- chart.is_heikinashi
- chart.is_pnf
- chart.is_range

## chart.is_linebreak
simple bool

### Type
simple bool

### Returns
```
Returns true if the chart type is Line break, false otherwise.
```

### See also
simple bool

### See also
- chart.is_renko
- chart.is_heikinashi
- chart.is_kagi
- chart.is_pnf
- chart.is_range

## chart.is_pnf
simple bool

### Type
simple bool

### Returns

```
Returns true if the chart type is Point & figure, false otherwise.
```

### See also
simple bool

### See also
- chart.is_renko
- chart.is_linebreak
- chart.is_kagi
- chart.is_heikinashi
- chart.is_range

## chart.is_range
simple bool

### Type
simple bool

### Returns
```
Returns true if the chart type is Range, false otherwise.
```

### See also
simple bool

### See also
- chart.is_renko
- chart.is_linebreak
- chart.is_kagi
- chart.is_pnf
- chart.is_heikinashi

## chart.is_renko
simple bool

### Type
simple bool

### Returns
```
Returns true if the chart type is Renko, false otherwise.
```

### See also
simple bool

### See also
- chart.is_heikinashi
- chart.is_linebreak
- chart.is_kagi
- chart.is_pnf
- chart.is_range

## chart.is_standard
simple bool

### Type
simple bool

### Returns
```
- Returns true if the chart type is not one of the following : Renko, Kagi, Line break, Point & figure, Range, Heikin Ashi; false otherwise.
```

### See also
The time of the leftmost bar currently visible on the chart.

### See also
- chart.is_renko
- chart.is_linebreak
- chart.is_kagi
- chart.is_pnf
- chart.is_range
- chart.is_heikinashi

## chart.left_visible_bar_time
The time of the leftmost bar currently visible on the chart.

### Type
input int

### Remarks
Scripts using this variable will automatically re-execute when its value updates to reflect changes in the chart, which can be caused by users scrolling the chart, or new real-time bars.

### See also
The time of the rightmost bar currently visible on the chart.

### See also
- chart.right_visible_bar_time

## chart.right_visible_bar_time
The time of the rightmost bar currently visible on the chart.

### Type
input int

### Remarks
Scripts using this variable will automatically re-execute when its value updates to reflect changes in the chart, which can be caused by users scrolling the chart, or new real-time bars.

### See also
Close price of the current bar when it has closed, or last traded price of a yet incomplete, realtime bar.

### See also
- chart.left_visible_bar_time

## close
Close price of the current bar when it has closed, or last traded price of a yet incomplete, realtime bar.

### Type
series float

### Remarks
Previous values may be accessed with square brackets operator [], e.g. close[1], close[2].

### See also
Date of current bar time in exchange timezone.

### See also
- open
- high
- low
- volume
- time
- hl2
- hlc3
- hlcc4
- ohlc4

## dayofmonth
Date of current bar time in exchange timezone.

### Type
series int

### Remarks
Note that this variable returns the day based on the time of the bar's open. For overnight sessions (e.g. EURUSD, where Monday session starts on Sunday, 17:00) this value can be lower by 1 than the day of the trading day.

### See also
Day of week for current bar time in exchange timezone.

### See also
- dayofmonth
- time
- year
- month
- weekofyear
- dayofweek
- hour
- minute
- second

## dayofweek
Day of week for current bar time in exchange timezone.

### Type
series int

### Remarks
Note that this variable returns the day based on the time of the bar's open. For overnight sessions (e.g. EURUSD, where Monday session starts on Sunday, 17:00) this value can be lower by 1 than the day of the trading day.

### See also
Returns the payment amount of the upcoming dividend in the currency of the current instrument, or na if this data isn't available.

### See also
- dayofweek
- time
- year
- month
- weekofyear
- dayofmonth
- hour
- minute
- second

## dividends.future_amount
Returns the payment amount of the upcoming dividend in the currency of the current instrument, or na if this data isn't available.

### Type
series float

### Remarks
This value is only fetched once during the script's initial calculation. The variable will return the same value until the script is recalculated, even after the expected Payment date of the next dividend.

## dividends.future_ex_date
Returns the Ex-dividend date (Ex-date) of the current instrument's next dividend payment, or na if this data isn't available. Ex-dividend date signifies when investors are no longer entitled to a payout from the most recent dividend. Only those who purchased shares before this day are entitled to the dividend payment.

### Type
series int

### Returns
```
UNIX time, expressed in milliseconds.
```

### Remarks
This value is only fetched once during the script's initial calculation. The variable will return the same value until the script is recalculated, even after the expected Payment date of the next dividend.

## dividends.future_pay_date
Returns the Payment date (Pay date) of the current instrument's next dividend payment, or na if this data isn't available. Payment date signifies the day when eligible investors will receive the dividend payment.

### Type
series int

### Returns
```
UNIX time, expressed in milliseconds.
```

### Remarks
This value is only fetched once during the script's initial calculation. The variable will return the same value until the script is recalculated, even after the expected Payment date of the next dividend.

## earnings.future_eps
Returns the estimated Earnings per Share of the next earnings report in the currency of the instrument, or na if this data isn't available.

### Type
series float

### Remarks
This value is only fetched once during the script's initial calculation. The variable will return the same value until the script is recalculated, even after the expected time of the next earnings report.

### See also
Checks the data for the next earnings report and returns the UNIX timestamp of the day when the financial period covered by those earnings ends, or na if this data isn't available.

### See also
- request.earnings

## earnings.future_period_end_time
Checks the data for the next earnings report and returns the UNIX timestamp of the day when the financial period covered by those earnings ends, or na if this data isn't available.

### Type
series int

### Returns
```
UNIX time, expressed in milliseconds.
```

### Remarks
This value is only fetched once during the script's initial calculation. The variable will return the same value until the script is recalculated, even after the expected time of the next earnings report.

### See also
Returns the estimated Revenue of the next earnings report in the currency of the instrument, or na if this data isn't available.

### See also
- request.earnings

## earnings.future_revenue
Returns the estimated Revenue of the next earnings report in the currency of the instrument, or na if this data isn't available.

### Type
series float

### Remarks
This value is only fetched once during the script's initial calculation. The variable will return the same value until the script is recalculated, even after the expected time of the next earnings report.

### See also
Returns a UNIX timestamp indicating the expected time of the next earnings report, or na if this data isn't available.

### See also
- request.earnings

## earnings.future_time

Returns a UNIX timestamp indicating the expected time of the next earnings report, or na if this data isn't available.

### Type
series int

### Returns
```
UNIX time, expressed in milliseconds.
```

### Remarks
This value is only fetched once during the script's initial calculation. The variable will return the same value until the script is recalculated, even after the expected time of the next earnings report.

### See also
Current high price.

### See also
- request.earnings

## high
Current high price.

### Type
series float

### Remarks
Previous values may be accessed with square brackets operator [], e.g. high[1], high[2].

### See also
Is a shortcut for (high + low)/2

### See also
- open
- low

- close
- volume
- time
- hl2
- hlc3
- hlcc4
- ohlc4

## hl2
Is a shortcut for (high + low)/2

### Type
series float

### See also
Is a shortcut for (high + low + close)/3

### See also
- open
- high
- low
- close
- volume
- time
- hlc3
- hlcc4
- ohlc4

## hlc3
Is a shortcut for (high + low + close)/3

### Type
series float

### See also
Is a shortcut for (high + low + close + close)/4

### See also
- open

- high
- low
- close
- volume
- time
- hl2
- hlcc4
- ohlc4

## hlcc4
Is a shortcut for (high + low + close + close)/4

### Type
series float

### See also
Current bar hour in exchange timezone.

### See also
- open
- high
- low
- close
- volume
- time
- hl2
- hlc3
- ohlc4

## hour
Current bar hour in exchange timezone.

### Type
series int

### See also
Returns an array filled with all the current labels drawn by the script.

### See also

- hour
- time
- year
- month
- weekofyear
- dayofmonth
- dayofweek
- minute
- second

## label.all
Returns an array filled with all the current labels drawn by the script.

### Type
array<label>

### Example
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### Remarks
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### See also
Bar index of the last chart bar. Bar indices begin at zero on the first bar.

### See also
- label.new
- line.all
- box.all
- table.all

## last_bar_index
Bar index of the last chart bar. Bar indices begin at zero on the first bar.

### Type
series int

### Example
Last historical bar index for closed markets, or the real-time bar index for open markets.

### Returns
```
Last historical bar index for closed markets, or the real-time bar index for open markets.
```

### Remarks
Please note that using this variable can cause indicator repainting.

### See also
Time in UNIX format of the last chart bar. It is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
- bar_index
- last_bar_time
- barstate.ishistory
- barstate.isrealtime

## last_bar_time
Time in UNIX format of the last chart bar. It is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### Type
series int

### Remarks
Please note that using this variable/function can cause indicator repainting.

### See also
Returns an array filled with all the current lines drawn by the script.

### See also
- time

- timenow
- timestamp
- last_bar_index

## line.all
Returns an array filled with all the current lines drawn by the script.

### Type
array<line>

### Example
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### Remarks
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### See also
Returns an array filled with all the current linefill objects drawn by the script.

### See also
- line.new
- label.all
- box.all
- table.all

## linefill.all
Returns an array filled with all the current linefill objects drawn by the script.

### Type
array<linefill>

### Remarks
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

## low
Current low price.

### Type
series float

### Remarks
Previous values may be accessed with square brackets operator [], e.g. low[1], low[2].

### See also
Current bar minute in exchange timezone.

### See also
- open
- high
- close
- volume
- time
- hl2
- hlc3
- hlcc4
- ohlc4

## minute
Current bar minute in exchange timezone.

### Type
series int

### See also
Current bar month in exchange timezone.

### See also
- minute
- time
- year
- month

- weekofyear
- dayofmonth
- dayofweek
- hour
- second

## month
Current bar month in exchange timezone.

### Type
series int

### Remarks
Note that this variable returns the month based on the time of the bar's open. For overnight sessions (e.g. EURUSD, where Monday session starts on Sunday, 17:00) this value can be lower by 1 than the month of the trading day.

### See also
A keyword signifying "not available", indicating that a variable has no assigned value.

### See also
- month
- time
- year
- weekofyear
- dayofmonth
- dayofweek
- hour
- minute
- second

na
A keyword signifying "not available", indicating that a variable has no assigned value.

### Type
simple na

### Example
Do not use this variable with comparison operators to test values for na, as it might lead to unexpected behavior. Instead, use the na function. Note that na can be used to initialize variables when the initialization statement also specifies the variable's type.

### Remarks
Do not use this variable with comparison operators to test values for na, as it might lead to unexpected behavior. Instead, use the na function. Note that na can be used to initialize variables when the initialization statement also specifies the variable's type.

### See also
Is a shortcut for (open + high + low + close)/4

### See also
- na
- nz
- fixnan

## ohlc4
Is a shortcut for (open + high + low + close)/4

### Type
series float

### See also
Current open price.

### See also
- open
- high
- low
- close
- volume
- time
- hl2
- hlc3
- hlcc4

## open
Current open price.

### Type
series float

### Remarks
Previous values may be accessed with square brackets operator [], e.g. open[1], open[2].

### See also
Returns an array containing all current polyline instances drawn by the script.

### See also
- high
- low
- close
- volume
- time
- hl2
- hlc3
- hlcc4
- ohlc4

## polyline.all
Returns an array containing all current polyline instances drawn by the script.

### Type
array<polyline>

### Remarks
The array is read-only. Index zero of the array references the ID of the oldest polyline object on the chart.

## second
Current bar second in exchange timezone.

### Type
series int

### See also
Returns true if the current bar is the first bar of the day's session, false otherwise. If extended session information is used, only returns true on the first bar of the pre-market bars.

### See also
- second
- time
- year
- month
- weekofyear
- dayofmonth
- dayofweek
- hour
- minute

## session.isfirstbar
Returns true if the current bar is the first bar of the day's session, false otherwise. If extended session information is used, only returns true on the first bar of the pre-market bars.

### Type
series bool

### Example
Returns true on the first regular session bar of the day, false otherwise. The result is the same whether extended session information is used or not.

### See also
Returns true on the first regular session bar of the day, false otherwise. The result is the same whether extended session information is used or not.

### See also
- session.isfirstbar_regular
- session.islastbar
- session.islastbar_regular

## session.isfirstbar_regular
Returns true on the first regular session bar of the day, false otherwise. The result is the same whether extended session information is used or not.

### Type
series bool

### Example
Returns true if the current bar is the last bar of the day's session, false otherwise. If extended session information is used, only returns true on the last bar of the post-market bars.

### See also
Returns true if the current bar is the last bar of the day's session, false otherwise. If extended session information is used, only returns true on the last bar of the post-market bars.

### See also
- session.isfirstbar
- session.islastbar

## session.islastbar
Returns true if the current bar is the last bar of the day's session, false otherwise. If extended session information is used, only returns true on the last bar of the post-market bars.

### Type
series bool

### Example
This variable is not guaranteed to return true once in every session because the last bar of the session might not exist if no trades occur during what should be the session's last bar.

### Remarks
This variable is not guaranteed to return true once in every session because the last bar of the session might not exist if no trades occur during what should be the session's last bar.

### See also
Returns true on the last regular session bar of the day, false otherwise. The result is the same whether extended session information is used or not.

### See also
- session.isfirstbar
- session.islastbar_regular

## session.islastbar_regular
Returns true on the last regular session bar of the day, false otherwise. The result is the same whether extended session information is used or not.

### Type
series bool

### Example
This variable is not guaranteed to return true once in every session because the last bar of the session might not exist if no trades occur during what should be the session's last bar.

### Remarks
This variable is not guaranteed to return true once in every session because the last bar of the session might not exist if no trades occur during what should be the session's last bar.

### See also
Returns true if the current bar is a part of the regular trading hours (i.e. market hours), false otherwise.

### See also
- session.isfirstbar
- session.islastbar
- session.isfirstbar_regular

## session.ismarket
Returns true if the current bar is a part of the regular trading hours (i.e. market hours), false otherwise.

### Type
series bool

### See also
Returns true if the current bar is a part of the post-market, false otherwise. On non-intraday charts always returns false.

### See also
- session.ispremarket
- session.ispostmarket

## session.ispostmarket
Returns true if the current bar is a part of the post-market, false otherwise. On non-intraday charts always returns false.

### Type
series bool

### See also
Returns true if the current bar is a part of the pre-market, false otherwise. On non-intraday charts always returns false.

### See also
- session.ismarket
- session.ispremarket

## session.ispremarket
Returns true if the current bar is a part of the pre-market, false otherwise. On non-intraday charts always returns false.

### Type
series bool

### See also
Returns the currency used to calculate results, which can be set in the strategy's properties.

### See also
- session.ismarket
- session.ispostmarket

## strategy.account_currency
Returns the currency used to calculate results, which can be set in the strategy's properties.

### Type
simple string

### See also
Returns the average amount of money lost per losing trade. Calculated as the sum of losses divided by the number of losing trades.

### See also
- strategy
- strategy.convert_to_account
- strategy.convert_to_symbol

## strategy.avg_losing_trade
Returns the average amount of money lost per losing trade. Calculated as the sum of losses divided by the number of losing trades.

### Type
series float

### See also
Returns the average percentage loss per losing trade. Calculated as the sum of loss percentages divided by the number of losing trades.

### See also
- strategy.avg_losing_trade_percent

## strategy.avg_losing_trade_percent

Returns the average percentage loss per losing trade. Calculated as the sum of loss percentages divided by the number of losing trades.

### Type
series float

### See also
Returns the average amount of money gained or lost per trade. Calculated as the sum of all profits and losses divided by the number of closed trades.

### See also
- strategy.avg_losing_trade

## strategy.avg_trade
Returns the average amount of money gained or lost per trade. Calculated as the sum of all profits and losses divided by the number of closed trades.

### Type
series float

### See also
Returns the average percentage gain or loss per trade. Calculated as the sum of all profit and loss percentages divided by the number of closed trades.

### See also
- strategy.avg_trade_percent

## strategy.avg_trade_percent
Returns the average percentage gain or loss per trade. Calculated as the sum of all profit and loss percentages divided by the number of closed trades.

### Type
series float

### See also
Returns the average amount of money gained per winning trade. Calculated as the sum of profits divided by the number of winning trades.

### See also
- strategy.avg_trade

## strategy.avg_winning_trade
Returns the average amount of money gained per winning trade. Calculated as the sum of profits divided by the number of winning trades.

### Type
series float

### See also
Returns the average percentage gain per winning trade. Calculated as the sum of profit percentages divided by the number of winning trades.

### See also
- strategy.avg_winning_trade_percent

## strategy.avg_winning_trade_percent
Returns the average percentage gain per winning trade. Calculated as the sum of profit percentages divided by the number of winning trades.

### Type
series float

### See also
Number of trades, which were closed for the whole trading range.

### See also
- strategy.avg_winning_trade

## strategy.closedtrades
Number of trades, which were closed for the whole trading range.

### Type

series int

### See also
The index, or trade number, of the first (oldest) trade listed in the List of Trades. This number is usually zero. If more trades than the allowed limit have been closed, the oldest trades are removed, and this number is the index of the oldest remaining trade.

### See also
- strategy.position_size
- strategy.opentrades
- strategy.wintrades
- strategy.losstrades
- strategy.eventrades

## strategy.closedtrades.first_index
The index, or trade number, of the first (oldest) trade listed in the List of Trades. This number is usually zero. If more trades than the allowed limit have been closed, the oldest trades are removed, and this number is the index of the oldest remaining trade.

### Type
series int

### See also
Current equity (strategy.initial_capital + strategy.netprofit + strategy.openprofit).

### See also
- strategy.position_size
- strategy.opentrades
- strategy.wintrades
- strategy.losstrades
- strategy.eventrades

## strategy.equity
Current equity (strategy.initial_capital + strategy.netprofit + strategy.openprofit).

### Type
series float

### See also
Number of breakeven trades for the whole trading range.

### See also
- strategy.netprofit
- strategy.openprofit
- strategy.position_size

## strategy.eventrades
Number of breakeven trades for the whole trading range.

### Type
series int

### See also
Total currency value of all completed losing trades.

### See also
- strategy.position_size
- strategy.opentrades
- strategy.closedtrades
- strategy.wintrades
- strategy.losstrades

## strategy.grossloss
Total currency value of all completed losing trades.

### Type
series float

### See also
The total value of all completed losing trades, expressed as a percentage of the initial capital.

### See also
- strategy.netprofit
- strategy.grossprofit

## strategy.grossloss_percent
The total value of all completed losing trades, expressed as a percentage of the initial capital.

### Type
series float

### See also
Total currency value of all completed winning trades.

### See also
- strategy.grossloss

## strategy.grossprofit
Total currency value of all completed winning trades.

### Type
series float

### See also
The total currency value of all completed winning trades, expressed as a percentage of the initial capital.

### See also
- strategy.netprofit
- strategy.grossloss

## strategy.grossprofit_percent
The total currency value of all completed winning trades, expressed as a percentage of the initial capital.

### Type

series float

### See also
The amount of initial capital set in the strategy properties.

### See also
- strategy.grossprofit

## strategy.initial_capital
The amount of initial capital set in the strategy properties.

### Type
series float

### See also
Number of unprofitable trades for the whole trading range.

### See also
- strategy

## strategy.losstrades
Number of unprofitable trades for the whole trading range.

### Type
series int

### See also
When margin is used in a strategy, returns the price point where a simulated margin call will occur and liquidate enough of the position to meet the margin requirements.

### See also
- strategy.position_size
- strategy.opentrades
- strategy.closedtrades
- strategy.wintrades
- strategy.eventrades

## strategy.margin_liquidation_price
When margin is used in a strategy, returns the price point where a simulated margin call will occur and liquidate enough of the position to meet the margin requirements.

### Type
series float

### Example
The variable returns na if the strategy does not use margin, i.e., the strategy declaration statement does not specify an argument for the margin_long or margin_short parameter.

### Remarks
The variable returns na if the strategy does not use margin, i.e., the strategy declaration statement does not specify an argument for the margin_long or margin_short parameter.

## strategy.max_contracts_held_all
Maximum number of contracts/shares/lots/units in one trade for the whole trading range.

### Type
series float

### See also
Maximum number of contracts/shares/lots/units in one long trade for the whole trading range.

### See also
- strategy.position_size
- strategy.max_contracts_held_long
- strategy.max_contracts_held_short

## strategy.max_contracts_held_long

Maximum number of contracts/shares/lots/units in one long trade for the whole trading range.

### Type
series float

### See also
Maximum number of contracts/shares/lots/units in one short trade for the whole trading range.

### See also
- strategy.position_size
- strategy.max_contracts_held_all
- strategy.max_contracts_held_short

## strategy.max_contracts_held_short
Maximum number of contracts/shares/lots/units in one short trade for the whole trading range.

### Type
series float

### See also
Maximum equity drawdown value for the whole trading range.

### See also
- strategy.position_size
- strategy.max_contracts_held_all
- strategy.max_contracts_held_long

## strategy.max_drawdown
Maximum equity drawdown value for the whole trading range.

### Type
series float

### See also

The maximum equity drawdown value for the whole trading range, expressed as a percentage and calculated by formula: Lowest Value During Trade / (Entry Price x Quantity) * 100.

### See also
- strategy.netprofit
- strategy.equity
- strategy.max_runup

## strategy.max_drawdown_percent
The maximum equity drawdown value for the whole trading range, expressed as a percentage and calculated by formula: Lowest Value During Trade / (Entry Price x Quantity) * 100.

### Type
series float

### See also
Maximum equity run-up value for the whole trading range.

### See also
- strategy.max_drawdown

## strategy.max_runup
Maximum equity run-up value for the whole trading range.

### Type
series float

### See also
The maximum equity run-up value for the whole trading range, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### See also
- strategy.netprofit
- strategy.equity
- strategy.max_drawdown

## strategy.max_runup_percent
The maximum equity run-up value for the whole trading range, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### Type
series float

### See also
Total currency value of all completed trades.

### See also
- strategy.max_runup

## strategy.netprofit
Total currency value of all completed trades.

### Type
series float

### See also
The total value of all completed trades, expressed as a percentage of the initial capital.

### See also
- strategy.openprofit
- strategy.position_size
- strategy.grossprofit
- strategy.grossloss

## strategy.netprofit_percent
The total value of all completed trades, expressed as a percentage of the initial capital.

### Type

series float

### See also
Current unrealized profit or loss for all open positions.

### See also
- strategy.netprofit

## strategy.openprofit
Current unrealized profit or loss for all open positions.

### Type
series float

### See also
The current unrealized profit or loss for all open positions, expressed as a percentage and calculated by formula: openPL / realizedEquity * 100.

### See also
- strategy.netprofit
- strategy.position_size

## strategy.openprofit_percent
The current unrealized profit or loss for all open positions, expressed as a percentage and calculated by formula: openPL / realizedEquity * 100.

### Type
series float

### See also
Number of market position entries, which were not closed and remain opened. If there is no open market position, 0 is returned.

### See also
- strategy.openprofit

## strategy.opentrades
Number of market position entries, which were not closed and remain opened. If there is no open market position, 0 is returned.

### Type
series int

### See also
Returns the capital amount currently held by open trades.

### See also
- strategy.position_size

## strategy.opentrades.capital_held
Returns the capital amount currently held by open trades.

### Type
series float

### Example
This variable returns na if the strategy does not simulate funding trades with a portion of the hypothetical account, i.e., if the strategy function does not include nonzero margin_long or margin_short arguments.

### Remarks
This variable returns na if the strategy does not simulate funding trades with a portion of the hypothetical account, i.e., if the strategy function does not include nonzero margin_long or margin_short arguments.

## strategy.position_avg_price
Average entry price of current market position. If the market position is flat, 'NaN' is returned.

### Type
series float

### See also
Name of the order that initially opened current market position.

### See also
- strategy.position_size

## strategy.position_entry_name
Name of the order that initially opened current market position.

### Type
series string

### See also
Direction and size of the current market position. If the value is > 0, the market position is long. If the value is < 0, the market position is short. The absolute value is the number of contracts/shares/lots/units in trade (position size).

### See also
- strategy.position_size

## strategy.position_size
Direction and size of the current market position. If the value is > 0, the market position is long. If the value is < 0, the market position is short. The absolute value is the number of contracts/shares/lots/units in trade (position size).

### Type
series float

### See also
Number of profitable trades for the whole trading range.

### See also
- strategy.position_avg_price

## strategy.wintrades

Number of profitable trades for the whole trading range.

### Type
series int

### See also
Returns a string containing the code representing the symbol's base currency (i.e., the traded currency or coin) if the instrument is a Forex or Crypto pair or a derivative based on such a pair. Otherwise, it returns an empty string. For example, this variable returns "EUR" for "EURJPY", "BTC" for "BTCUSDT", "CAD" for "CME:6C1!", and "" for "NASDAQ:AAPL".

### See also
- strategy.position_size
- strategy.opentrades
- strategy.closedtrades
- strategy.losstrades
- strategy.eventrades

## syminfo.basecurrency
Returns a string containing the code representing the symbol's base currency (i.e., the traded currency or coin) if the instrument is a Forex or Crypto pair or a derivative based on such a pair. Otherwise, it returns an empty string. For example, this variable returns "EUR" for "EURJPY", "BTC" for "BTCUSDT", "CAD" for "CME:6C1!", and "" for "NASDAQ:AAPL".

### Type
simple string

### See also
Returns the two-letter code of the country where the symbol is traded, in the ISO 3166-1 alpha-2 format, or na if the exchange is not directly tied to a specific country. For example, on "NASDAQ:AAPL" it will return "US", on "LSE:AAPL" it will return "GB", and on "BITSTAMP:BTCUSD it will return na.

### See also
- syminfo.currency
- syminfo.ticker

## syminfo.country
Returns the two-letter code of the country where the symbol is traded, in the ISO 3166-1 alpha-2 format, or na if the exchange is not directly tied to a specific country. For example, on "NASDAQ:AAPL" it will return "US", on "LSE:AAPL" it will return "GB", and on "BITSTAMP:BTCUSD it will return na.

### Type
simple string

## syminfo.currency
Returns a string containing the code representing the currency of the symbol's prices. For example, this variable returns "USD" for "NASDAQ:AAPL" and "JPY" for "EURJPY".

### Type
simple string

### See also
Description for the current symbol.

### See also
- syminfo.basecurrency
- syminfo.ticker
- currency.USD
- currency.EUR

## syminfo.description
Description for the current symbol.

### Type
simple string

### See also
The number of employees the company has.

### See also
- syminfo.ticker

- syminfo.prefix

## syminfo.employees
The number of employees the company has.

### Type
simple int

### Example
A UNIX timestamp representing the start of the last day of the current futures contract. This variable is only compatible with non-continuous futures symbols. On other symbols, it returns na.

### See also
A UNIX timestamp representing the start of the last day of the current futures contract. This variable is only compatible with non-continuous futures symbols. On other symbols, it returns na.

### See also
- syminfo.shareholders
- syminfo.shares_outstanding_float
- syminfo.shares_outstanding_total

## syminfo.expiration_date
A UNIX timestamp representing the start of the last day of the current futures contract. This variable is only compatible with non-continuous futures symbols. On other symbols, it returns na.

### Type
simple int

## syminfo.industry
Returns the industry of the symbol, or na if the symbol has no industry. Example: "Internet Software/Services", "Packaged software", "Integrated Oil", "Motor Vehicles", etc. These are the same values one can see in the chart's "Symbol info" window.

### Type
simple string

### Remarks
A sector is a broad section of the economy. An industry is a narrower classification. NASDAQ:CAT (Caterpillar, Inc.) for example, belongs to the "Producer Manufacturing" sector and the "Trucks/Construction/Farm Machinery" industry.

## syminfo.main_tickerid
A ticker identifier representing the current chart's symbol. The value contains an exchange prefix and a symbol name, separated by a colon (e.g., "NASDAQ:AAPL"). It can also include information about data modifications such as dividend adjustment, non-standard chart type, currency conversion, etc. Unlike syminfo.tickerid, this variable's value does not change when used in the expression argument of a request.*() function call.

### Type
simple string

### See also
The smallest amount of the current symbol that can be traded. This limit is set by the exchange. For cryptocurrencies, it is often less than 1 token. For most other types of asset, it is often 1.

### See also
- ticker.new
- timeframe.main_period
- syminfo.tickerid
- syminfo.ticker
- timeframe.period
- timeframe.multiplier
- syminfo.root

## syminfo.mincontract
The smallest amount of the current symbol that can be traded. This limit is set by the exchange. For cryptocurrencies, it is often less than 1 token. For most other types of asset, it is often 1.

### Type
simple float

### See also
Returns a whole number used to calculate the smallest increment between a symbol's price movements (syminfo.mintick). It is the numerator in the syminfo.mintick formula: syminfo.minmove / syminfo.pricescale = syminfo.mintick.

### See also
- syminfo.mintick
- syminfo.pointvalue

## syminfo.minmove
Returns a whole number used to calculate the smallest increment between a symbol's price movements (syminfo.mintick). It is the numerator in the syminfo.mintick formula: syminfo.minmove / syminfo.pricescale = syminfo.mintick.

### Type
simple int

### See also
Min tick value for the current symbol.

### See also
- ticker.new
- syminfo.ticker
- timeframe.period
- timeframe.multiplier
- syminfo.root

## syminfo.mintick
Min tick value for the current symbol.

### Type
simple float

### See also

Point value for the current symbol.

### See also
- syminfo.pointvalue
- syminfo.mincontract

## syminfo.pointvalue
Point value for the current symbol.

### Type
simple float

### See also
Prefix of current symbol name (i.e. for 'CME_EOD:TICKER' prefix is 'CME_EOD').

### See also
- syminfo.mintick
- syminfo.mincontract

## syminfo.prefix
Prefix of current symbol name (i.e. for 'CME_EOD:TICKER' prefix is 'CME_EOD').

### Type
simple string

### Example
Returns a whole number used to calculate the smallest increment between a symbol's price movements (syminfo.mintick). It is the denominator in the syminfo.mintick formula: syminfo.minmove / syminfo.pricescale = syminfo.mintick.

### See also
Returns a whole number used to calculate the smallest increment between a symbol's price movements (syminfo.mintick). It is the denominator in the syminfo.mintick formula: syminfo.minmove / syminfo.pricescale = syminfo.mintick.

### See also
- syminfo.ticker

- syminfo.tickerid

## syminfo.pricescale
Returns a whole number used to calculate the smallest increment between a symbol's price movements (syminfo.mintick). It is the denominator in the syminfo.mintick formula: syminfo.minmove / syminfo.pricescale = syminfo.mintick.

### Type
simple int

### See also
The number of analysts who gave the current symbol a "Buy" rating.

### See also
- ticker.new
- syminfo.ticker
- timeframe.period
- timeframe.multiplier
- syminfo.root

## syminfo.recommendations_buy
The number of analysts who gave the current symbol a "Buy" rating.

### Type
series int

### Example
The number of analysts who gave the current symbol a "Strong Buy" rating.

### See also
The number of analysts who gave the current symbol a "Strong Buy" rating.

### See also
- syminfo.recommendations_buy_strong
- syminfo.recommendations_date
- syminfo.recommendations_hold
- syminfo.recommendations_total
- syminfo.recommendations_sell

- syminfo.recommendations_sell_strong

## syminfo.recommendations_buy_strong
The number of analysts who gave the current symbol a "Strong Buy" rating.

### Type
series int

### Example
The starting date of the last set of recommendations for the current symbol.

### See also
The starting date of the last set of recommendations for the current symbol.

### See also
- syminfo.recommendations_buy
- syminfo.recommendations_date
- syminfo.recommendations_hold
- syminfo.recommendations_total
- syminfo.recommendations_sell
- syminfo.recommendations_sell_strong

## syminfo.recommendations_date
The starting date of the last set of recommendations for the current symbol.

### Type
series int

### Example
The number of analysts who gave the current symbol a "Hold" rating.

### See also
The number of analysts who gave the current symbol a "Hold" rating.

### See also
- syminfo.recommendations_buy
- syminfo.recommendations_buy_strong

- syminfo.recommendations_hold
- syminfo.recommendations_total
- syminfo.recommendations_sell
- syminfo.recommendations_sell_strong

## syminfo.recommendations_hold
The number of analysts who gave the current symbol a "Hold" rating.

### Type
series int

### Example
The number of analysts who gave the current symbol a "Sell" rating.

### See also
The number of analysts who gave the current symbol a "Sell" rating.

### See also
- syminfo.recommendations_buy
- syminfo.recommendations_buy_strong
- syminfo.recommendations_date
- syminfo.recommendations_total
- syminfo.recommendations_sell
- syminfo.recommendations_sell_strong

## syminfo.recommendations_sell
The number of analysts who gave the current symbol a "Sell" rating.

### Type
series int

### Example
The number of analysts who gave the current symbol a "Strong Sell" rating.

### See also
The number of analysts who gave the current symbol a "Strong Sell" rating.

### See also
- syminfo.recommendations_buy
- syminfo.recommendations_buy_strong
- syminfo.recommendations_date
- syminfo.recommendations_hold
- syminfo.recommendations_total
- syminfo.recommendations_sell_strong

## syminfo.recommendations_sell_strong
The number of analysts who gave the current symbol a "Strong Sell" rating.

### Type
series int

### Example
The total number of recommendations for the current symbol.

### See also
The total number of recommendations for the current symbol.

### See also
- syminfo.recommendations_buy
- syminfo.recommendations_buy_strong
- syminfo.recommendations_date
- syminfo.recommendations_hold
- syminfo.recommendations_total
- syminfo.recommendations_sell

## syminfo.recommendations_total
The total number of recommendations for the current symbol.

### Type
series int

### Example
Root for derivatives like futures contract. For other symbols returns the same value as syminfo.ticker.

### See also
Root for derivatives like futures contract. For other symbols returns the same value as syminfo.ticker.

### See also
- syminfo.recommendations_buy
- syminfo.recommendations_buy_strong
- syminfo.recommendations_date
- syminfo.recommendations_hold
- syminfo.recommendations_sell
- syminfo.recommendations_sell_strong

## syminfo.root
Root for derivatives like futures contract. For other symbols returns the same value as syminfo.ticker.

### Type
simple string

### Example
Returns the sector of the symbol, or na if the symbol has no sector. Example: "Electronic Technology", "Technology services", "Energy Minerals", "Consumer Durables", etc. These are the same values one can see in the chart's "Symbol info" window.

### See also
Returns the sector of the symbol, or na if the symbol has no sector. Example: "Electronic Technology", "Technology services", "Energy Minerals", "Consumer Durables", etc. These are the same values one can see in the chart's "Symbol info" window.

### See also
- syminfo.ticker
- syminfo.tickerid

## syminfo.sector
Returns the sector of the symbol, or na if the symbol has no sector. Example: "Electronic Technology", "Technology services", "Energy Minerals", "Consumer Durables", etc. These are the same values one can see in the chart's "Symbol info" window.

### Type
simple string

### Remarks
A sector is a broad section of the economy. An industry is a narrower classification. NASDAQ:CAT (Caterpillar, Inc.) for example, belongs to the "Producer Manufacturing" sector and the "Trucks/Construction/Farm Machinery" industry.

## syminfo.session
Session type of the chart main series. Possible values are session.regular, session.extended.

### Type
simple string

### See also
The number of shareholders the company has.

### See also
- session.regular
- session.extended

## syminfo.shareholders
The number of shareholders the company has.

### Type
simple int

### Example
The total number of shares outstanding a company has available, excluding any of its restricted shares.

### See also
The total number of shares outstanding a company has available, excluding any of

its restricted shares.

### See also
- syminfo.employees
- syminfo.shares_outstanding_float
- syminfo.shares_outstanding_total

## syminfo.shares_outstanding_float
The total number of shares outstanding a company has available, excluding any of its restricted shares.

### Type
simple float

### Example
The total number of shares outstanding a company has available, including restricted shares held by insiders, major shareholders, and employees.

### See also
The total number of shares outstanding a company has available, including restricted shares held by insiders, major shareholders, and employees.

### See also
- syminfo.employees
- syminfo.shareholders
- syminfo.shares_outstanding_total

## syminfo.shares_outstanding_total
The total number of shares outstanding a company has available, including restricted shares held by insiders, major shareholders, and employees.

### Type
simple int

### Example
The average of the last yearly price targets for the symbol predicted by analysts.

### See also
The average of the last yearly price targets for the symbol predicted by analysts.

### See also
- syminfo.employees
- syminfo.shareholders
- syminfo.shares_outstanding_float

## syminfo.target_price_average
The average of the last yearly price targets for the symbol predicted by analysts.

### Type
series float

### Example
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### Remarks
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### See also
The starting date of the last price target prediction for the current symbol.

### See also
- syminfo.target_price_date
- syminfo.target_price_estimates
- syminfo.target_price_high
- syminfo.target_price_low
- syminfo.target_price_median

## syminfo.target_price_date
The starting date of the last price target prediction for the current symbol.

### Type
series int

### Example
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### Remarks
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### See also
The latest total number of price target predictions for the current symbol.

### See also
- syminfo.target_price_average
- syminfo.target_price_estimates
- syminfo.target_price_high
- syminfo.target_price_low
- syminfo.target_price_median

## syminfo.target_price_estimates
The latest total number of price target predictions for the current symbol.

### Type
series float

### Example
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### Remarks
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### See also
The last highest yearly price target for the symbol predicted by analysts.

### See also
- syminfo.target_price_average
- syminfo.target_price_date

- syminfo.target_price_high
- syminfo.target_price_low
- syminfo.target_price_median

## syminfo.target_price_high
The last highest yearly price target for the symbol predicted by analysts.

### Type
series float

### Example
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### Remarks
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### See also
The last lowest yearly price target for the symbol predicted by analysts.

### See also
- syminfo.target_price_average
- syminfo.target_price_date
- syminfo.target_price_estimates
- syminfo.target_price_low
- syminfo.target_price_median

## syminfo.target_price_low
The last lowest yearly price target for the symbol predicted by analysts.

### Type
series float

### Example
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### Remarks
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### See also
The median of the last yearly price targets for the symbol predicted by analysts.

### See also
- syminfo.target_price_average
- syminfo.target_price_date
- syminfo.target_price_estimates
- syminfo.target_price_high
- syminfo.target_price_median

## syminfo.target_price_median
The median of the last yearly price targets for the symbol predicted by analysts.

### Type
series float

### Example
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### Remarks
If analysts supply the targets when the market is closed, the variable can return na until the market opens.

### See also
Symbol name without exchange prefix, e.g. 'MSFT'.

### See also
- syminfo.target_price_average
- syminfo.target_price_date
- syminfo.target_price_estimates
- syminfo.target_price_high
- syminfo.target_price_low

## syminfo.ticker
Symbol name without exchange prefix, e.g. 'MSFT'.

### Type
simple string

### See also
A ticker identifier representing the chart's symbol or a requested symbol, depending on how the script uses it. The variable's value represents a requested dataset's ticker ID when used in the expression argument of a request.*() function call. Otherwise, it represents the chart's ticker ID. The value contains an exchange prefix and a symbol name, separated by a colon (e.g., "NASDAQ:AAPL"). It can also include information about data modifications such as dividend adjustment, non-standard chart type, currency conversion, etc.

### See also
- syminfo.tickerid
- timeframe.period
- timeframe.multiplier
- syminfo.root

## syminfo.tickerid
A ticker identifier representing the chart's symbol or a requested symbol, depending on how the script uses it. The variable's value represents a requested dataset's ticker ID when used in the expression argument of a request.*() function call. Otherwise, it represents the chart's ticker ID. The value contains an exchange prefix and a symbol name, separated by a colon (e.g., "NASDAQ:AAPL"). It can also include information about data modifications such as dividend adjustment, non-standard chart type, currency conversion, etc.

### Type
simple string

### Remarks
Because the value of this variable does not always use a simple "prefix:ticker" format, it is a poor candidate for use in boolean comparisons or string manipulation functions. In those contexts, run the variable's result through ticker.standard to purify it. This will remove any extraneous information and return a ticker ID consistently formatted using the "prefix:ticker" structure.

### See also
Timezone of the exchange of the chart main series. Possible values see in timestamp.

### See also
- ticker.new
- syminfo.main_tickerid
- timeframe.main_period
- syminfo.ticker
- timeframe.period
- timeframe.multiplier
- syminfo.root

## syminfo.timezone
Timezone of the exchange of the chart main series. Possible values see in timestamp.

### Type
simple string

### See also
The type of market the symbol belongs to. The values are "stock", "fund", "dr", "right", "bond", "warrant", "structured", "index", "forex", "futures", "spread", "economic", "fundamental", "crypto", "spot", "swap", "option", "commodity".

### See also
- timestamp

## syminfo.type
The type of market the symbol belongs to. The values are "stock", "fund", "dr", "right", "bond", "warrant", "structured", "index", "forex", "futures", "spread", "economic", "fundamental", "crypto", "spot", "swap", "option", "commodity".

### Type
simple string

### See also
Volume type of the current symbol. Possible values are: "base" for base currency, "quote" for quote currency, "tick" for the number of transactions, and "n/a" when

there is no volume or its type is not specified.

### See also
- syminfo.ticker

## syminfo.volumetype
Volume type of the current symbol. Possible values are: "base" for base currency, "quote" for quote currency, "tick" for the number of transactions, and "n/a" when there is no volume or its type is not specified.

### Type
simple string

### Remarks
Only some data feed suppliers provide information qualifying volume. As a result, the variable will return a value on some symbols only, mostly in the crypto sector.

### See also
Accumulation/distribution index.

### See also
- syminfo.type

## ta.accdist
Accumulation/distribution index.

### Type
series float

## ta.iii
Intraday Intensity Index.

### Type
series float

### Example
Negative Volume Index.

## ta.nvi
Negative Volume Index.

### Type
series float

### Example
On Balance Volume.

## ta.obv
On Balance Volume.

### Type
series float

### Example
Positive Volume Index.

## ta.pvi
Positive Volume Index.

### Type
series float

### Example
Price-Volume Trend.

## ta.pvt
Price-Volume Trend.

### Type
series float

### Example
True range, equivalent to ta.tr(handle_na = false). It is calculated as math.max(high - low, math.abs(high - close[1]), math.abs(low - close[1])).

## ta.tr
True range, equivalent to ta.tr(handle_na = false). It is calculated as math.max(high - low, math.abs(high - close[1]), math.abs(low - close[1])).

### Type
series float

### See also
Volume Weighted Average Price. It uses hlc3 as its source series.

### See also
- ta.tr
- ta.atr

## ta.vwap
Volume Weighted Average Price. It uses hlc3 as its source series.

### Type
series float

### See also
Williams Accumulation/Distribution.

### See also
- ta.vwap

## ta.wad
Williams Accumulation/Distribution.

### Type
series float

### Example
Williams Variable Accumulation/Distribution.

## ta.wvad
Williams Variable Accumulation/Distribution.

### Type
series float

### Example
Returns an array filled with all the current tables drawn by the script.

## table.all
Returns an array filled with all the current tables drawn by the script.

### Type
array<table>

### Example
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### Remarks
The array is read-only. Index zero of the array is the ID of the oldest object on the chart.

### See also

Current bar time in UNIX format. It is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
- table.new
- line.all
- label.all
- box.all

## time
Current bar time in UNIX format. It is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### Type
series int

### Remarks
Note that this variable returns the timestamp based on the time of the bar's open. Because of that, for overnight sessions (e.g. EURUSD, where Monday session starts on Sunday, 17:00) this variable can return time before the specified date of the trading day. For example, on EURUSD, dayofmonth(time) can be lower by 1 than the date of the trading day, because the bar for the current day actually opens one day prior.

### See also
The time of the current bar's close in UNIX format. It represents the number of milliseconds elapsed since 00:00:00 UTC, 1 January 1970. On non-standard price-based chart types (Renko, Line break, Kagi, Point & Figure, and Range), this variable returns na on the chart's realtime bars.

### See also
- time
- time_close
- timenow
- year
- month
- weekofyear
- dayofmonth
- dayofweek
- hour
- minute
- second

## time_close
The time of the current bar's close in UNIX format. It represents the number of milliseconds elapsed since 00:00:00 UTC, 1 January 1970. On non-standard price-based chart types (Renko, Line break, Kagi, Point & Figure, and Range), this variable returns na on the chart's realtime bars.

### Type
series int

### See also
The beginning time of the trading day the current bar belongs to, in UNIX format (the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970).

### See also
- time
- timenow
- year
- month
- weekofyear
- dayofmonth
- dayofweek
- hour
- minute
- second

## time_tradingday
The beginning time of the trading day the current bar belongs to, in UNIX format (the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970).

### Type
series int

### Remarks
The variable is useful for overnight sessions, where the current day's session can start on the previous calendar day (e.g., on FXCM:EURUSD the Monday session will start on Sunday, 17:00 in the exchange timezone). Unlike time, which would return the timestamp for Sunday at 17:00 for the Monday daily bar, time_tradingday will return the timestamp for Monday, 00:00 UTC.

### See also
Returns true if current resolution is a daily resolution, false otherwise.

### See also
- time
- time_close

## timeframe.isdaily
Returns true if current resolution is a daily resolution, false otherwise.

### Type
simple bool

### See also
Returns true if current resolution is a daily or weekly or monthly resolution, false otherwise.

### See also
- timeframe.isdwm
- timeframe.isintraday
- timeframe.isminutes
- timeframe.isseconds
- timeframe.isticks
- timeframe.isweekly
- timeframe.ismonthly

## timeframe.isdwm
Returns true if current resolution is a daily or weekly or monthly resolution, false otherwise.

### Type
simple bool

### See also
Returns true if current resolution is an intraday (minutes or seconds) resolution, false otherwise.

### See also
- timeframe.isintraday
- timeframe.isminutes
- timeframe.isseconds
- timeframe.isticks
- timeframe.isdaily
- timeframe.isweekly
- timeframe.ismonthly

## timeframe.isintraday
Returns true if current resolution is an intraday (minutes or seconds) resolution, false otherwise.

### Type
simple bool

### See also
Returns true if current resolution is a minutes resolution, false otherwise.

### See also
- timeframe.isminutes
- timeframe.isseconds
- timeframe.isticks
- timeframe.isdwm
- timeframe.isdaily
- timeframe.isweekly
- timeframe.ismonthly

## timeframe.isminutes
Returns true if current resolution is a minutes resolution, false otherwise.

### Type
simple bool

### See also
Returns true if current resolution is a monthly resolution, false otherwise.

### See also
- timeframe.isdwm

- timeframe.isintraday
- timeframe.isseconds
- timeframe.isticks
- timeframe.isdaily
- timeframe.isweekly
- timeframe.ismonthly

## timeframe.ismonthly
Returns true if current resolution is a monthly resolution, false otherwise.

### Type
simple bool

### See also
Returns true if current resolution is a seconds resolution, false otherwise.

### See also
- timeframe.isdwm
- timeframe.isintraday
- timeframe.isminutes
- timeframe.isseconds
- timeframe.isticks
- timeframe.isdaily
- timeframe.isweekly

## timeframe.isseconds
Returns true if current resolution is a seconds resolution, false otherwise.

### Type
simple bool

### See also
Returns true if current resolution is a ticks resolution, false otherwise.

### See also
- timeframe.isdwm
- timeframe.isintraday
- timeframe.isminutes
- timeframe.isticks

- timeframe.isdaily
- timeframe.isweekly
- timeframe.ismonthly

## timeframe.isticks
Returns true if current resolution is a ticks resolution, false otherwise.

### Type
simple bool

### See also
Returns true if current resolution is a weekly resolution, false otherwise.

### See also
- timeframe.isdwm
- timeframe.isintraday
- timeframe.isminutes
- timeframe.isseconds
- timeframe.isdaily
- timeframe.isweekly
- timeframe.ismonthly

## timeframe.isweekly
Returns true if current resolution is a weekly resolution, false otherwise.

### Type
simple bool

### See also
A string representation of the script's main timeframe. If the script is an indicator that specifies a timeframe value in its declaration statement, this variable holds that value. Otherwise, its value represents the chart's timeframe. Unlike timeframe.period, this variable's value does not change when used in the expression argument of a request.*() function call.

### See also
- timeframe.isdwm
- timeframe.isintraday
- timeframe.isminutes

- timeframe.isseconds
- timeframe.isticks
- timeframe.isdaily
- timeframe.ismonthly

## timeframe.main_period
A string representation of the script's main timeframe. If the script is an indicator that specifies a timeframe value in its declaration statement, this variable holds that value. Otherwise, its value represents the chart's timeframe. Unlike timeframe.period, this variable's value does not change when used in the expression argument of a request.*() function call.

### Type
simple string

### See also
Multiplier of resolution, e.g. '60' - 60, 'D' - 1, '5D' - 5, '12M' - 12.

### See also
- timeframe.period
- syminfo.main_tickerid
- syminfo.ticker
- syminfo.tickerid
- timeframe.multiplier

## timeframe.multiplier
Multiplier of resolution, e.g. '60' - 60, 'D' - 1, '5D' - 5, '12M' - 12.

### Type
simple int

### See also
A string representation of the script's main timeframe or a requested timeframe, depending on how the script uses it. The variable's value represents the timeframe of a requested dataset when used in the expression argument of a request.*() function call. Otherwise, its value represents the script's main timeframe (timeframe.main_period), which equals either the timeframe argument of the indicator declaration statement or the chart's timeframe.

### See also
- syminfo.ticker
- syminfo.tickerid
- timeframe.period

## timeframe.period
A string representation of the script's main timeframe or a requested timeframe, depending on how the script uses it. The variable's value represents the timeframe of a requested dataset when used in the expression argument of a request.*() function call. Otherwise, its value represents the script's main timeframe (timeframe.main_period), which equals either the timeframe argument of the indicator declaration statement or the chart's timeframe.

### Type
simple string

### Remarks
To always access the script's main timeframe, even within another context, use the timeframe.main_period variable.

### See also
Current time in UNIX format. It is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
- timeframe.main_period
- syminfo.main_tickerid
- syminfo.ticker
- syminfo.tickerid
- timeframe.multiplier

## timenow
Current time in UNIX format. It is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### Type
series int

### Remarks

Please note that using this variable/function can cause indicator repainting.

### See also
Current bar volume.

### See also
- timestamp
- time
- time_close
- year
- month
- weekofyear
- dayofmonth
- dayofweek
- hour
- minute
- second

## volume
Current bar volume.

### Type
series float

### Remarks
Previous values may be accessed with square brackets operator [], e.g. volume[1], volume[2].

### See also
Week number of current bar time in exchange timezone.

### See also
- open
- high
- low
- close
- time
- hl2
- hlc3
- hlcc4
- ohlc4

## weekofyear
Week number of current bar time in exchange timezone.

### Type
series int

### Remarks
Note that this variable returns the week based on the time of the bar's open. For overnight sessions (e.g. EURUSD, where Monday session starts on Sunday, 17:00) this value can be lower by 1 than the week of the trading day.

### See also
Current bar year in exchange timezone.

### See also
- weekofyear
- time
- year
- month
- dayofmonth
- dayofweek
- hour
- minute
- second

## year
Current bar year in exchange timezone.

### Type
series int

### Remarks
Note that this variable returns the year based on the time of the bar's open. For overnight sessions (e.g. EURUSD, where Monday session starts on Sunday, 17:00) this value can be lower by 1 than the year of the trading day.

### See also

Constant for dividends adjustment type (dividends adjustment is applied).

### See also
- year
- time
- month
- weekofyear
- dayofmonth
- dayofweek
- hour
- minute
- second

## adjustment.dividends
Constant for dividends adjustment type (dividends adjustment is applied).

### Type
const string

### See also
Constant for none adjustment type (no adjustment is applied).

### See also
- adjustment.none
- adjustment.splits
- ticker.new

## adjustment.none
Constant for none adjustment type (no adjustment is applied).

### Type
const string

### See also
Constant for splits adjustment type (splits adjustment is applied).

### See also
- adjustment.splits

- adjustment.dividends
- ticker.new

## adjustment.splits
Constant for splits adjustment type (splits adjustment is applied).

### Type
const string

### See also
A named constant for use with the freq parameter of the alert() function.

### See also
- adjustment.none
- adjustment.dividends
- ticker.new

## alert.freq_all
A named constant for use with the freq parameter of the alert() function.

### Type
const string

### See also
A named constant for use with the freq parameter of the alert() function.

### See also
- alert

## alert.freq_once_per_bar
A named constant for use with the freq parameter of the alert() function.

### Type
const string

### See also
A named constant for use with the freq parameter of the alert() function.

### See also
- alert

## alert.freq_once_per_bar_close
A named constant for use with the freq parameter of the alert() function.

### Type
const string

### See also
A constant to specify the value of the backadjustment parameter in ticker.new and ticker.modify functions.

### See also
- alert

## backadjustment.inherit
A constant to specify the value of the backadjustment parameter in ticker.new and ticker.modify functions.

### Type
const backadjustment

### See also
A constant to specify the value of the backadjustment parameter in ticker.new and ticker.modify functions.

### See also
- ticker.new
- ticker.modify
- backadjustment.on
- backadjustment.off

## backadjustment.off
A constant to specify the value of the backadjustment parameter in ticker.new and ticker.modify functions.

### Type
const backadjustment

### See also
A constant to specify the value of the backadjustment parameter in ticker.new and ticker.modify functions.

### See also
- ticker.new
- ticker.modify
- backadjustment.on
- backadjustment.inherit

## backadjustment.on
A constant to specify the value of the backadjustment parameter in ticker.new and ticker.modify functions.

### Type
const backadjustment

### See also
Merge strategy for requested data. Data is merged continuously without gaps, all the gaps are filled with the previous nearest existing value.

### See also
- ticker.new
- ticker.modify
- backadjustment.inherit
- backadjustment.off

## barmerge.gaps_off
Merge strategy for requested data. Data is merged continuously without gaps, all the gaps are filled with the previous nearest existing value.

### Type
const barmerge_gaps

### See also
Merge strategy for requested data. Data is merged with possible gaps (na values).

### See also
- request.security
- barmerge.gaps_on

## barmerge.gaps_on
Merge strategy for requested data. Data is merged with possible gaps (na values).

### Type
const barmerge_gaps

### See also
Merge strategy for the requested data position. Requested barset is merged with current barset in the order of sorting bars by their close time. This merge strategy disables effect of getting data from "future" on calculation on history.

### See also
- request.security
- barmerge.gaps_off

## barmerge.lookahead_off
Merge strategy for the requested data position. Requested barset is merged with current barset in the order of sorting bars by their close time. This merge strategy disables effect of getting data from "future" on calculation on history.

### Type
const barmerge_lookahead

### See also
Merge strategy for the requested data position. Requested barset is merged with current barset in the order of sorting bars by their opening time. This merge strategy can lead to undesirable effect of getting data from "future" on calculation on history. This is unacceptable in backtesting strategies, but can be

useful in indicators.

### See also
- request.security
- barmerge.lookahead_on

## barmerge.lookahead_on
Merge strategy for the requested data position. Requested barset is merged with current barset in the order of sorting bars by their opening time. This merge strategy can lead to undesirable effect of getting data from "future" on calculation on history. This is unacceptable in backtesting strategies, but can be useful in indicators.

### Type
const barmerge_lookahead

### See also
Is a named constant for #00BCD4 color.

### See also
- request.security
- barmerge.lookahead_off

## color.aqua
Is a named constant for #00BCD4 color.

### Type
const color

### See also
Is a named constant for #363A45 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon

- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.orange

## color.black
Is a named constant for #363A45 color.

### Type
const color

### See also
Is a named constant for #2962ff color.

### See also
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.blue
Is a named constant for #2962ff color.

### Type
const color

### See also
Is a named constant for #E040FB color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.teal
- color.aqua
- color.orange

## color.fuchsia
Is a named constant for #E040FB color.

### Type
const color

### See also
Is a named constant for #787B86 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple

- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.gray
Is a named constant for #787B86 color.

### Type
const color

### See also
Is a named constant for #4CAF50 color.

### See also
- color.black
- color.silver
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.green
Is a named constant for #4CAF50 color.

### Type

const color

### See also
Is a named constant for #00E676 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.lime
Is a named constant for #00E676 color.

### Type
const color

### See also
Is a named constant for #880E4F color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green

- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.maroon
Is a named constant for #880E4F color.

### Type
const color

### See also
Is a named constant for #311B92 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.navy
Is a named constant for #311B92 color.

### Type
const color

### See also
Is a named constant for #808000 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.blue
- color.teal
- color.aqua
- color.orange

## color.olive
Is a named constant for #808000 color.

### Type
const color

### See also
Is a named constant for #FF9800 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.yellow

- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.orange
Is a named constant for #FF9800 color.

### Type
const color

### See also
Is a named constant for #9C27B0 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua

## color.purple
Is a named constant for #9C27B0 color.

### Type
const color

### See also

Is a named constant for #F23645 color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.red
Is a named constant for #F23645 color.

### Type
const color

### See also
Is a named constant for #B2B5BE color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue

- color.teal
- color.aqua
- color.orange

## color.silver
Is a named constant for #B2B5BE color.

### Type
const color

### See also
Is a named constant for #089981 color.

### See also
- color.black
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## color.teal
Is a named constant for #089981 color.

### Type
const color

### See also
Is a named constant for #FFFFFF color.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.aqua
- color.orange

## color.white
Is a named constant for #FFFFFF color.

### Type
const color

### See also
Is a named constant for #FDD835 color.

### See also
- color.black
- color.silver
- color.gray
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.yellow
- color.navy
- color.blue
- color.teal
- color.aqua

- color.orange

## color.yellow
Is a named constant for #FDD835 color.

### Type
const color

### See also
Australian dollar.

### See also
- color.black
- color.silver
- color.gray
- color.white
- color.maroon
- color.red
- color.purple
- color.fuchsia
- color.green
- color.lime
- color.olive
- color.navy
- color.blue
- color.teal
- color.aqua
- color.orange

## currency.AUD
Australian dollar.

### Type
const string

### See also
Bitcoin.

### See also

- strategy

## currency.BTC
Bitcoin.

### Type
const string

### See also
Canadian dollar.

### See also
- strategy

## currency.CAD
Canadian dollar.

### Type
const string

### See also
Swiss franc.

### See also
- strategy

## currency.CHF
Swiss franc.

### Type
const string

### See also
Ethereum.

### See also
- strategy

## currency.ETH
Ethereum.

### Type
const string

### See also
Euro.

### See also
- strategy

## currency.EUR
Euro.

### Type
const string

### See also
Pound sterling.

### See also
- strategy

## currency.GBP
Pound sterling.

### Type
const string

### See also

Hong Kong dollar.

### See also
- strategy

## currency.HKD
Hong Kong dollar.

### Type
const string

### See also
Indian rupee.

### See also
- strategy

## currency.INR
Indian rupee.

### Type
const string

### See also
Japanese yen.

### See also
- strategy

## currency.JPY
Japanese yen.

### Type
const string

### See also
South Korean won.

### See also
- strategy

## currency.KRW
South Korean won.

### Type
const string

### See also
Malaysian ringgit.

### See also
- strategy

## currency.MYR
Malaysian ringgit.

### Type
const string

### See also
Norwegian krone.

### See also
- strategy

## currency.NOK
Norwegian krone.

### Type

const string

### See also
Unspecified currency.

### See also
- strategy

## currency.NONE
Unspecified currency.

### Type
const string

### See also
New Zealand dollar.

### See also
- strategy

## currency.NZD
New Zealand dollar.

### Type
const string

### See also
Russian ruble.

### See also
- strategy

## currency.RUB
Russian ruble.

### Type
const string

### See also
Swedish krona.

### See also
- strategy

## currency.SEK
Swedish krona.

### Type
const string

### See also
Singapore dollar.

### See also
- strategy

## currency.SGD
Singapore dollar.

### Type
const string

### See also
Turkish lira.

### See also
- strategy

## currency.TRY

Turkish lira.

### Type
const string

### See also
United States dollar.

### See also
- strategy

## currency.USD
United States dollar.

### Type
const string

### See also
Tether.

### See also
- strategy

## currency.USDT
Tether.

### Type
const string

### See also
South African rand.

### See also
- strategy

## currency.ZAR
South African rand.

### Type
const string

### See also
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### See also
- strategy

## dayofweek.friday
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### Type
const int

### See also
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### See also
- dayofweek.sunday
- dayofweek.monday
- dayofweek.tuesday
- dayofweek.wednesday
- dayofweek.thursday
- dayofweek.saturday

## dayofweek.monday
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### Type

const int

### See also
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### See also
- dayofweek.sunday
- dayofweek.tuesday
- dayofweek.wednesday
- dayofweek.thursday
- dayofweek.friday
- dayofweek.saturday

## dayofweek.saturday
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### Type
const int

### See also
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### See also
- dayofweek.sunday
- dayofweek.monday
- dayofweek.tuesday
- dayofweek.wednesday
- dayofweek.thursday
- dayofweek.friday

## dayofweek.sunday
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### Type
const int

### See also
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### See also
- dayofweek.monday
- dayofweek.tuesday
- dayofweek.wednesday
- dayofweek.thursday
- dayofweek.friday
- dayofweek.saturday

## dayofweek.thursday
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### Type
const int

### See also
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### See also
- dayofweek.sunday
- dayofweek.monday
- dayofweek.tuesday
- dayofweek.wednesday
- dayofweek.friday
- dayofweek.saturday

## dayofweek.tuesday
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### Type
const int

### See also
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### See also
- dayofweek.sunday
- dayofweek.monday
- dayofweek.wednesday
- dayofweek.thursday
- dayofweek.friday
- dayofweek.saturday

## dayofweek.wednesday
Is a named constant for return value of dayofweek function and value of dayofweek variable.

### Type
const int

### See also
A named constant for use with the display parameter of plot*() and input*() functions. Displays plotted or input values in all possible locations.

### See also
- dayofweek.sunday
- dayofweek.monday
- dayofweek.tuesday
- dayofweek.thursday
- dayofweek.friday
- dayofweek.saturday

## display.all
A named constant for use with the display parameter of plot*() and input*() functions. Displays plotted or input values in all possible locations.

### Type
const plot_simple_display

### See also
A named constant for use with the display parameter of plot*() and input*() functions. Displays plotted or input values in the Data Window, a menu accessible from the chart's right sidebar.

### See also
- plot
- plotshape
- plotchar
- plotarrow
- plotbar
- plotcandle

## display.data_window
A named constant for use with the display parameter of plot*() and input*() functions. Displays plotted or input values in the Data Window, a menu accessible from the chart's right sidebar.

### Type
const plot_display

### See also
A named constant for use with the display parameter of plot*() and input*() functions. plot*() functions using this will not display their plotted values anywhere. However, alert template messages and fill functions can still use the values, and they will appear in exported chart data. input*() functions using this constant will only display their values within the script's settings.

### See also
- plot
- plotshape
- plotchar
- plotarrow
- plotbar
- plotcandle

## display.none
A named constant for use with the display parameter of plot*() and input*() functions. plot*() functions using this will not display their plotted values anywhere. However, alert template messages and fill functions can still use the values, and they will appear in exported chart data. input*() functions using this

constant will only display their values within the script's settings.

### Type
const plot_simple_display

### See also
A named constant for use with the display parameter of plot*() functions. Displays plotted values in the chart pane used by the script.

### See also
- plot
- plotshape
- plotchar
- plotarrow
- plotbar
- plotcandle

## display.pane
A named constant for use with the display parameter of plot*() functions. Displays plotted values in the chart pane used by the script.

### Type
const plot_display

### See also
A named constant for use with the display parameter of plot*() functions. Displays the plot’s label and value on the price scale if the chart's settings allow it.

### See also
- plot
- plotshape
- plotchar
- plotarrow
- plotbar
- plotcandle

## display.price_scale
A named constant for use with the display parameter of plot*() functions. Displays the plot’s label and value on the price scale if the chart's settings allow it.

### Type
const plot_display

### See also
A named constant for use with the display parameter of plot*() and input*()
functions. Displays plotted or input values in the status line next to the script's
name on the chart if the chart's settings allow it.

### See also
- plot
- plotshape
- plotchar
- plotarrow
- plotbar
- plotcandle

## display.status_line
A named constant for use with the display parameter of plot*() and input*()
functions. Displays plotted or input values in the status line next to the script's
name on the chart if the chart's settings allow it.

### Type
const plot_display

### See also
A named constant for the request.dividends function. Is used to request the
dividends return on a stock before deductions.

### See also
- plot
- plotshape
- plotchar
- plotarrow
- plotbar
- plotcandle

## dividends.gross
A named constant for the request.dividends function. Is used to request the

dividends return on a stock before deductions.

### Type
const string

### See also
A named constant for the request.dividends function. Is used to request the dividends return on a stock after deductions.

### See also
- request.dividends

## dividends.net
A named constant for the request.dividends function. Is used to request the dividends return on a stock after deductions.

### Type
const string

### See also
A named constant for the request.earnings function. Is used to request the earnings value as it was reported.

### See also
- request.dividends

## earnings.actual
A named constant for the request.earnings function. Is used to request the earnings value as it was reported.

### Type
const string

### See also
A named constant for the request.earnings function. Is used to request the estimated earnings value.

### See also
- request.earnings

## earnings.estimate
A named constant for the request.earnings function. Is used to request the estimated earnings value.

### Type
const string

### See also
A named constant for the request.earnings function. Is used to request the standardized earnings value.

### See also
- request.earnings

## earnings.standardized
A named constant for the request.earnings function. Is used to request the standardized earnings value.

### Type
const string

### See also
A named constant for line.new and line.set_extend functions.

### See also
- request.earnings

## extend.both
A named constant for line.new and line.set_extend functions.

### Type
const string

### See also
A named constant for line.new and line.set_extend functions.

### See also
- line.new
- line.set_extend
- extend.none
- extend.left
- extend.right

## extend.left
A named constant for line.new and line.set_extend functions.

### Type
const string

### See also
A named constant for line.new and line.set_extend functions.

### See also
- line.new
- line.set_extend
- extend.none
- extend.right
- extend.both

## extend.none
A named constant for line.new and line.set_extend functions.

### Type
const string

### See also
A named constant for line.new and line.set_extend functions.

### See also

- line.new
- line.set_extend
- extend.left
- extend.right
- extend.both

## extend.right
A named constant for line.new and line.set_extend functions.

### Type
const string

### See also
Literal representing a bool value, and result of a comparison operation.

### See also
- line.new
- line.set_extend
- extend.none
- extend.left
- extend.both

## false
Literal representing a bool value, and result of a comparison operation.

### Remarks
See the User Manual for comparison operators and logical operators.

### See also
Default text font for box.new, box.set_text_font_family, label.new,
label.set_text_font_family, table.cell and table.cell_set_text_font_family
functions.

### See also
- bool

## font.family_default

Default text font for box.new, box.set_text_font_family, label.new, label.set_text_font_family, table.cell and table.cell_set_text_font_family functions.

### Type
const string

### See also
Monospace text font for box.new, box.set_text_font_family, label.new, label.set_text_font_family, table.cell and table.cell_set_text_font_family functions.

### See also
- box.new
- box.set_text_font_family
- label.new
- label.set_text_font_family
- table.cell
- table.cell_set_text_font_family
- font.family_monospace

## font.family_monospace
Monospace text font for box.new, box.set_text_font_family, label.new, label.set_text_font_family, table.cell and table.cell_set_text_font_family functions.

### Type
const string

### See also
Is a named constant for selecting the formatting of the script output values from the parent series in the indicator function.

### See also
- box.new
- box.set_text_font_family
- label.new
- label.set_text_font_family
- table.cell
- table.cell_set_text_font_family
- font.family_default

## format.inherit
Is a named constant for selecting the formatting of the script output values from the parent series in the indicator function.

### Type
const string

### See also
Is a named constant to use with the str.tostring function. Passing a number to str.tostring with this argument rounds the number to the nearest value that can be divided by syminfo.mintick, without the remainder, with ties rounding up, and returns the string version of said value with trailing zeros.

### See also
- indicator
- format.price
- format.volume
- format.percent

## format.mintick
Is a named constant to use with the str.tostring function. Passing a number to str.tostring with this argument rounds the number to the nearest value that can be divided by syminfo.mintick, without the remainder, with ties rounding up, and returns the string version of said value with trailing zeros.

### Type
const string

### See also
Is a named constant for selecting the formatting of the script output values as a percentage in the indicator function. It adds a percent sign after values.

### See also
- indicator
- format.inherit
- format.price
- format.volume

## format.percent
Is a named constant for selecting the formatting of the script output values as a percentage in the indicator function. It adds a percent sign after values.

### Type
const string

### Remarks
The default precision is 2, regardless of the precision of the chart itself. This can be changed with the 'precision' argument of the indicator function.

### See also
Is a named constant for selecting the formatting of the script output values as prices in the indicator function.

### See also
- indicator
- format.inherit
- format.price
- format.volume

## format.price
Is a named constant for selecting the formatting of the script output values as prices in the indicator function.

### Type
const string

### Remarks
If format is format.price, default precision value is set. You can use the precision argument of indicator function to change the precision value.

### See also
Is a named constant for selecting the formatting of the script output values as volume in the indicator function, e.g. '5183' will be formatted as '5.183K'.

### See also

- indicator
- format.inherit
- format.volume
- format.percent

## format.volume
Is a named constant for selecting the formatting of the script output values as volume in the indicator function, e.g. '5183' will be formatted as '5.183K'.

### Type
const string

### See also
Is a named constant for dashed linestyle of hline function.

### See also
- indicator
- format.inherit
- format.price
- format.percent

## hline.style_dashed
Is a named constant for dashed linestyle of hline function.

### Type
const hline_style

### See also
Is a named constant for dotted linestyle of hline function.

### See also
- hline.style_solid
- hline.style_dotted

## hline.style_dotted
Is a named constant for dotted linestyle of hline function.

### Type
const hline_style

### See also
Is a named constant for solid linestyle of hline function.

### See also
- hline.style_solid
- hline.style_dashed

## hline.style_solid
Is a named constant for solid linestyle of hline function.

### Type
const hline_style

### See also
Label style for label.new and label.set_style functions.

### See also
- hline.style_dotted
- hline.style_dashed

## label.style_arrowdown
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign

- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_arrowup
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left

- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_circle
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_cross
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_diamond
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square

## label.style_flag
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_circle

- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_center
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right

- label.style_square
- label.style_diamond

## label.style_label_down
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_left
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_lower_left
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none

- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_lower_right
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left

- label.style_label_right
- label.style_label_lower_left
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_right
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_up

Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_upper_left
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_label_upper_right
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle

- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_none
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_square
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_diamond

## label.style_text_outline
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_triangledown
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangleup
- label.style_flag

- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right
- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_triangleup
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Label style for label.new and label.set_style functions.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_xcross
- label.style_cross
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_lower_left
- label.style_label_lower_right
- label.style_label_upper_left
- label.style_label_upper_right

- label.style_label_center
- label.style_square
- label.style_diamond

## label.style_xcross
Label style for label.new and label.set_style functions.

### Type
const string

### See also
Line style for line.new and line.set_style functions. Solid line with arrows on both points.

### See also
- label.new
- label.set_style
- label.set_textalign
- label.style_none
- label.style_cross
- label.style_triangleup
- label.style_triangledown
- label.style_flag
- label.style_circle
- label.style_arrowup
- label.style_arrowdown
- label.style_label_up
- label.style_label_down
- label.style_label_left
- label.style_label_right
- label.style_label_center
- label.style_square
- label.style_diamond

## line.style_arrow_both
Line style for line.new and line.set_style functions. Solid line with arrows on both points.

### Type
const string

### See also
Line style for line.new and line.set_style functions. Solid line with arrow on the first point.

### See also
- line.new
- line.set_style
- line.style_solid
- line.style_dotted
- line.style_dashed
- line.style_arrow_left
- line.style_arrow_right

## line.style_arrow_left
Line style for line.new and line.set_style functions. Solid line with arrow on the first point.

### Type
const string

### See also
Line style for line.new and line.set_style functions. Solid line with arrow on the second point.

### See also
- line.new
- line.set_style
- line.style_solid
- line.style_dotted
- line.style_dashed
- line.style_arrow_right
- line.style_arrow_both

## line.style_arrow_right
Line style for line.new and line.set_style functions. Solid line with arrow on the second point.

### Type
const string

### See also
Line style for line.new and line.set_style functions.

### See also
- line.new
- line.set_style
- line.style_solid
- line.style_dotted
- line.style_dashed
- line.style_arrow_left
- line.style_arrow_both

## line.style_dashed
Line style for line.new and line.set_style functions.

### Type
const string

### See also
Line style for line.new and line.set_style functions.

### See also
- line.new
- line.set_style
- line.style_solid
- line.style_dotted
- line.style_arrow_left
- line.style_arrow_right
- line.style_arrow_both

## line.style_dotted
Line style for line.new and line.set_style functions.

### Type
const string

### See also

Line style for line.new and line.set_style functions.

### See also
- line.new
- line.set_style
- line.style_solid
- line.style_dashed
- line.style_arrow_left
- line.style_arrow_right
- line.style_arrow_both

## line.style_solid
Line style for line.new and line.set_style functions.

### Type
const string

### See also
Location value for plotshape, plotchar functions. Shape is plotted above main series bars.

### See also
- line.new
- line.set_style
- line.style_dotted
- line.style_dashed
- line.style_arrow_left
- line.style_arrow_right
- line.style_arrow_both

## location.abovebar
Location value for plotshape, plotchar functions. Shape is plotted above main series bars.

### Type
const string

### See also
Location value for plotshape, plotchar functions. Shape is plotted on chart using

indicator value as a price coordinate.

### See also
- plotshape
- plotchar
- location.belowbar
- location.top
- location.bottom
- location.absolute

## location.absolute
Location value for plotshape, plotchar functions. Shape is plotted on chart using indicator value as a price coordinate.

### Type
const string

### See also
Location value for plotshape, plotchar functions. Shape is plotted below main series bars.

### See also
- plotshape
- plotchar
- location.abovebar
- location.belowbar
- location.top
- location.bottom

## location.belowbar
Location value for plotshape, plotchar functions. Shape is plotted below main series bars.

### Type
const string

### See also
Location value for plotshape, plotchar functions. Shape is plotted near the bottom chart border.

### See also
- plotshape
- plotchar
- location.abovebar
- location.top
- location.bottom
- location.absolute

## location.bottom
Location value for plotshape, plotchar functions. Shape is plotted near the bottom chart border.

### Type
const string

### See also
Location value for plotshape, plotchar functions. Shape is plotted near the top chart border.

### See also
- plotshape
- plotchar
- location.abovebar
- location.belowbar
- location.top
- location.absolute

## location.top
Location value for plotshape, plotchar functions. Shape is plotted near the top chart border.

### Type
const string

### See also
Is a named constant for Euler's number. It is equal to 2.7182818284590452.

### See also
- plotshape
- plotchar
- location.abovebar
- location.belowbar
- location.bottom
- location.absolute

## math.e
Is a named constant for Euler's number. It is equal to 2.7182818284590452.

### Type
const float

### See also
Is a named constant for the golden ratio. It is equal to 1.6180339887498948.

### See also
- math.phi
- math.pi
- math.rphi

## math.phi
Is a named constant for the golden ratio. It is equal to 1.6180339887498948.

### Type
const float

### See also
Is a named constant for Archimedes' constant. It is equal to 3.1415926535897932.

### See also
- math.e
- math.pi
- math.rphi

## math.pi

Is a named constant for Archimedes' constant. It is equal to 3.1415926535897932.

### Type
const float

### See also
Is a named constant for the golden ratio conjugate. It is equal to 0.6180339887498948.

### See also
- math.e
- math.phi
- math.rphi

## math.rphi
Is a named constant for the golden ratio conjugate. It is equal to 0.6180339887498948.

### Type
const float

### See also
Determines the sort order of the array from the smallest to the largest value.

### See also
- math.e
- math.pi
- math.phi

## order.ascending
Determines the sort order of the array from the smallest to the largest value.

### Type
const sort_order

### See also
Determines the sort order of the array from the largest to the smallest value.

### See also
- array.new_float
- array.sort

## order.descending
Determines the sort order of the array from the largest to the smallest value.

### Type
const sort_order

### See also
A named constant for the 'Area' style, to be used as an argument for the style parameter in the plot function.

### See also
- array.new_float
- array.sort

## plot.style_area
A named constant for the 'Area' style, to be used as an argument for the style parameter in the plot function.

### Type
const plot_style

### See also
A named constant for the 'Area With Breaks' style, to be used as an argument for the style parameter in the plot function. Similar to plot.style_area, except the gaps in the data are not filled.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond

- plot.style_histogram
- plot.style_areabr
- plot.style_cross
- plot.style_columns
- plot.style_circles

## plot.style_areabr
A named constant for the 'Area With Breaks' style, to be used as an argument for the style parameter in the plot function. Similar to plot.style_area, except the gaps in the data are not filled.

### Type
const plot_style

### See also
A named constant for the 'Circles' style, to be used as an argument for the style parameter in the plot function.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_histogram
- plot.style_cross
- plot.style_area
- plot.style_columns
- plot.style_circles

## plot.style_circles
A named constant for the 'Circles' style, to be used as an argument for the style parameter in the plot function.

### Type
const plot_style

### See also
A named constant for the 'Columns' style, to be used as an argument for the style

parameter in the plot function.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_histogram
- plot.style_cross
- plot.style_area
- plot.style_areabr
- plot.style_columns

## plot.style_columns
A named constant for the 'Columns' style, to be used as an argument for the style parameter in the plot function.

### Type
const plot_style

### See also
A named constant for the 'Cross' style, to be used as an argument for the style parameter in the plot function.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_histogram
- plot.style_cross
- plot.style_area
- plot.style_areabr
- plot.style_circles

## plot.style_cross
A named constant for the 'Cross' style, to be used as an argument for the style

parameter in the plot function.

### Type
const plot_style

### See also
A named constant for the 'Histogram' style, to be used as an argument for the style parameter in the plot function.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_histogram
- plot.style_area
- plot.style_areabr
- plot.style_columns
- plot.style_circles

## plot.style_histogram
A named constant for the 'Histogram' style, to be used as an argument for the style parameter in the plot function.

### Type
const plot_style

### See also
A named constant for the 'Line' style, to be used as an argument for the style parameter in the plot function.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_cross

- plot.style_area
- plot.style_areabr
- plot.style_columns
- plot.style_circles

## plot.style_line
A named constant for the 'Line' style, to be used as an argument for the style parameter in the plot function.

### Type
const plot_style

### See also
A named constant for the 'Line With Breaks' style, to be used as an argument for the style parameter in the plot function. Similar to plot.style_line, except the gaps in the data are not filled.

### See also
- plot
- plot.style_steplinebr
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_histogram
- plot.style_cross
- plot.style_area
- plot.style_areabr
- plot.style_columns
- plot.style_circles

## plot.style_linebr
A named constant for the 'Line With Breaks' style, to be used as an argument for the style parameter in the plot function. Similar to plot.style_line, except the gaps in the data are not filled.

### Type
const plot_style

### See also
A named constant for the 'Step Line' style, to be used as an argument for the style

parameter in the plot function.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_histogram
- plot.style_cross
- plot.style_area
- plot.style_areabr
- plot.style_columns
- plot.style_circles

## plot.style_stepline
A named constant for the 'Step Line' style, to be used as an argument for the style parameter in the plot function.

### Type
const plot_style

### See also
A named constant for the 'Step Line With Diamonds' style, to be used as an argument for the style parameter in the plot function. Similar to plot.style_stepline, except the data changes are also marked with the Diamond shapes.

### See also
- plot
- plot.style_line
- plot.style_steplinebr
- plot.style_stepline_diamond
- plot.style_linebr
- plot.style_histogram
- plot.style_cross
- plot.style_area
- plot.style_areabr
- plot.style_columns
- plot.style_circles

## plot.style_stepline_diamond

A named constant for the 'Step Line With Diamonds' style, to be used as an argument for the style parameter in the plot function. Similar to plot.style_stepline, except the data changes are also marked with the Diamond shapes.

### Type
const plot_style

### See also
A named constant for the 'Step line with Breaks' style, to be used as an argument for the style parameter in the plot function.

### See also
- plot
- plot.style_steplinebr
- plot.style_line
- plot.style_linebr
- plot.style_histogram
- plot.style_cross
- plot.style_area
- plot.style_areabr
- plot.style_columns
- plot.style_circles

## plot.style_steplinebr
A named constant for the 'Step line with Breaks' style, to be used as an argument for the style parameter in the plot function.

### Type
const plot_style

### See also
Table position is used in table.new, table.cell functions.

### See also
- plot
- plot.style_circles
- plot.style_line
- plot.style_linebr
- plot.style_stepline
- plot.style_stepline_diamond
- plot.style_histogram

- plot.style_cross
- plot.style_area
- plot.style_areabr
- plot.style_columns

## position.bottom_center
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position
- position.top_left
- position.top_center
- position.top_right
- position.middle_left
- position.middle_center
- position.middle_right
- position.bottom_left

## position.bottom_left
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position

- position.top_left
- position.top_center
- position.top_right
- position.middle_left
- position.middle_center
- position.middle_right
- position.bottom_center

## position.bottom_right
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position
- position.top_left
- position.top_center
- position.top_right
- position.middle_left
- position.middle_center
- position.middle_right
- position.bottom_left
- position.bottom_center

## position.middle_center
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position
- position.top_left
- position.top_center
- position.top_right
- position.middle_left
- position.middle_right
- position.bottom_left
- position.bottom_center

## position.middle_left
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position
- position.top_left
- position.top_center
- position.top_right
- position.middle_center
- position.middle_right
- position.bottom_left
- position.bottom_center

## position.middle_right
Table position is used in table.new, table.cell functions.

### Type
const string

### See also

Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position
- position.top_left
- position.top_center
- position.top_right
- position.middle_left
- position.middle_center
- position.bottom_left
- position.bottom_center

## position.top_center
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position
- position.top_left
- position.top_right
- position.middle_left
- position.middle_center
- position.middle_right
- position.bottom_left
- position.bottom_center

## position.top_left
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Table position is used in table.new, table.cell functions.

### See also
- table.new
- table.cell
- table.set_position
- position.top_center
- position.top_right
- position.middle_left
- position.middle_center
- position.middle_right
- position.bottom_left
- position.bottom_center

## position.top_right
Table position is used in table.new, table.cell functions.

### Type
const string

### See also
Scale value for indicator function. Indicator is added to the left price scale.

### See also
- table.new
- table.cell
- table.set_position
- position.top_left
- position.top_center
- position.middle_left
- position.middle_center
- position.middle_right
- position.bottom_left
- position.bottom_center

## scale.left
Scale value for indicator function. Indicator is added to the left price scale.

### Type
const scale_type

### See also
Scale value for indicator function. Indicator is added in 'No Scale' mode. Can be used only with 'overlay=true'.

### See also
- indicator

## scale.none
Scale value for indicator function. Indicator is added in 'No Scale' mode. Can be used only with 'overlay=true'.

### Type
const scale_type

### See also
Scale value for indicator function. Indicator is added to the right price scale.

### See also
- indicator

## scale.right
Scale value for indicator function. Indicator is added to the right price scale.

### Type
const scale_type

### See also
Constant for extended session type (with extended hours data).

### See also
- indicator

## session.extended
Constant for extended session type (with extended hours data).

### Type
const string

### See also
Constant for regular session type (no extended hours data).

### See also
- session.regular
- syminfo.session

## session.regular
Constant for regular session type (no extended hours data).

### Type
const string

### See also
A constant to specify the value of the settlement_as_close parameter in ticker.new and ticker.modify functions.

### See also
- session.extended
- syminfo.session

## settlement_as_close.inherit
A constant to specify the value of the settlement_as_close parameter in ticker.new and ticker.modify functions.

### Type
const settlement

### See also
A constant to specify the value of the settlement_as_close parameter in ticker.new

and ticker.modify functions.

### See also
- ticker.new
- ticker.modify
- settlement_as_close.on
- settlement_as_close.off

## settlement_as_close.off
A constant to specify the value of the settlement_as_close parameter in ticker.new and ticker.modify functions.

### Type
const settlement

### See also
A constant to specify the value of the settlement_as_close parameter in ticker.new and ticker.modify functions.

### See also
- ticker.new
- ticker.modify
- settlement_as_close.on
- settlement_as_close.inherit

## settlement_as_close.on
A constant to specify the value of the settlement_as_close parameter in ticker.new and ticker.modify functions.

### Type
const settlement

### See also
Shape style for plotshape function.

### See also
- ticker.new
- ticker.modify

- settlement_as_close.inherit
- settlement_as_close.off

## shape.arrowdown
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.arrowup
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.circle
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.cross
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.diamond
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.flag
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.labeldown
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.labelup
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.square
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.triangledown
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.triangleup
Shape style for plotshape function.

### Type
const string

### See also
Shape style for plotshape function.

### See also
- plotshape

## shape.xcross
Shape style for plotshape function.

### Type
const string

### See also
Size value for plotshape, plotchar functions. The size of the shape automatically adapts to the size of the bars.

### See also
- plotshape

## size.auto
Size value for plotshape, plotchar functions. The size of the shape automatically adapts to the size of the bars.

### Type
const string

### See also
Size value for plotshape, plotchar functions. The size of the shape constantly huge.

### See also
- plotshape
- plotchar
- label.set_size
- size.tiny
- size.small
- size.normal
- size.large
- size.huge

## size.huge
Size value for plotshape, plotchar functions. The size of the shape constantly huge.

### Type
const string

### See also
Size value for plotshape, plotchar functions. The size of the shape constantly large.

### See also
- plotshape
- plotchar
- label.set_size
- size.auto
- size.tiny
- size.small
- size.normal
- size.large

## size.large
Size value for plotshape, plotchar functions. The size of the shape constantly large.

### Type
const string

### See also
Size value for plotshape, plotchar functions. The size of the shape constantly normal.

### See also
- plotshape
- plotchar
- label.set_size
- size.auto
- size.tiny
- size.small
- size.normal
- size.huge

## size.normal
Size value for plotshape, plotchar functions. The size of the shape constantly normal.

### Type

const string

### See also
Size value for plotshape, plotchar functions. The size of the shape constantly small.

### See also
- plotshape
- plotchar
- label.set_size
- size.auto
- size.tiny
- size.small
- size.large
- size.huge

## size.small
Size value for plotshape, plotchar functions. The size of the shape constantly small.

### Type
const string

### See also
Size value for plotshape, plotchar functions. The size of the shape constantly tiny.

### See also
- plotshape
- plotchar
- label.set_size
- size.auto
- size.tiny
- size.normal
- size.large
- size.huge

## size.tiny
Size value for plotshape, plotchar functions. The size of the shape constantly tiny.

### Type
const string

### See also
A named constant for the request.splits function. Is used to request the denominator (the number below the line in a fraction) of a splits.

### See also
- plotshape
- plotchar
- label.set_size
- size.auto
- size.small
- size.normal
- size.large
- size.huge

## splits.denominator
A named constant for the request.splits function. Is used to request the denominator (the number below the line in a fraction) of a splits.

### Type
const string

### See also
A named constant for the request.splits function. Is used to request the numerator (the number above the line in a fraction) of a splits.

### See also
- request.splits

## splits.numerator
A named constant for the request.splits function. Is used to request the numerator (the number above the line in a fraction) of a splits.

### Type
const string

### See also
This is one of the arguments that can be supplied to the default_qty_type parameter in the strategy declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in strategy.entry or strategy.order function calls. It specifies that an amount of cash in the strategy.account_currency will be used to enter trades.

### See also
- request.splits

## strategy.cash
This is one of the arguments that can be supplied to the default_qty_type parameter in the strategy declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in strategy.entry or strategy.order function calls. It specifies that an amount of cash in the strategy.account_currency will be used to enter trades.

### Type
const string

### Example
Commission type for an order. Money displayed in the account currency per contract.

### See also
Commission type for an order. Money displayed in the account currency per contract.

### See also
- strategy

## strategy.commission.cash_per_contract
Commission type for an order. Money displayed in the account currency per contract.

### Type
const string

### See also

Commission type for an order. Money displayed in the account currency per order.

### See also
- strategy

## strategy.commission.cash_per_order
Commission type for an order. Money displayed in the account currency per order.

### Type
const string

### See also
Commission type for an order. A percentage of the cash volume of order.

### See also
- strategy

## strategy.commission.percent
Commission type for an order. A percentage of the cash volume of order.

### Type
const string

### See also
It allows strategy to open both long and short positions.

### See also
- strategy

## strategy.direction.all
It allows strategy to open both long and short positions.

### Type
const string

### See also
It allows strategy to open only long positions.

### See also
- strategy.risk.allow_entry_in

## strategy.direction.long
It allows strategy to open only long positions.

### Type
const string

### See also
It allows strategy to open only short positions.

### See also
- strategy.risk.allow_entry_in

## strategy.direction.short
It allows strategy to open only short positions.

### Type
const string

### See also
This is one of the arguments that can be supplied to the default_qty_type parameter in the strategy declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in strategy.entry or strategy.order function calls. It specifies that a number of contracts/shares/lots will be used to enter trades.

### See also
- strategy.risk.allow_entry_in

## strategy.fixed
This is one of the arguments that can be supplied to the default_qty_type parameter

in the strategy declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in strategy.entry or strategy.order function calls. It specifies that a number of contracts/shares/lots will be used to enter trades.

### Type
const string

### Example
A named constant for use with the direction parameter of the strategy.entry and strategy.order commands. It specifies that the command creates a buy order.

### See also
A named constant for use with the direction parameter of the strategy.entry and strategy.order commands. It specifies that the command creates a buy order.

### See also
- strategy

## strategy.long
A named constant for use with the direction parameter of the strategy.entry and strategy.order commands. It specifies that the command creates a buy order.

### Type
const strategy_direction

### See also
A named constant for use with the oca_type parameter of the strategy.entry and strategy.order commands. It specifies that the strategy cancels the unfilled order when another order with the same oca_name and oca_type executes.

### See also
- strategy.entry
- strategy.exit
- strategy.order

## strategy.oca.cancel
A named constant for use with the oca_type parameter of the strategy.entry and strategy.order commands. It specifies that the strategy cancels the unfilled order

when another order with the same oca_name and oca_type executes.

### Type
const string

### Remarks
Strategies cannot cancel or reduce pending orders from an OCA group if they execute on the same tick. For example, if the market price triggers two stop orders from strategy.order calls with the same oca_* arguments, the strategy cannot fully or partially cancel either one.

### See also
A named constant for use with the oca_type parameter of the strategy.entry and strategy.order commands. It specifies that the order executes independently of all other orders, including those with the same oca_name.

### See also
- strategy.entry
- strategy.exit
- strategy.order

## strategy.oca.none
A named constant for use with the oca_type parameter of the strategy.entry and strategy.order commands. It specifies that the order executes independently of all other orders, including those with the same oca_name.

### Type
const string

### See also
A named constant for use with the oca_type parameter of the strategy.entry and strategy.order commands. It specifies that when another order with the same oca_name and oca_type executes, the strategy reduces the unfilled order by that order's size. If the unfilled order's size reaches 0 after reduction, it is the same as canceling the order entirely.

### See also
- strategy.entry
- strategy.exit
- strategy.order

## strategy.oca.reduce
A named constant for use with the oca_type parameter of the strategy.entry and strategy.order commands. It specifies that when another order with the same oca_name and oca_type executes, the strategy reduces the unfilled order by that order's size. If the unfilled order's size reaches 0 after reduction, it is the same as canceling the order entirely.

### Type
const string

### Remarks
Strategies cannot cancel or reduce pending orders from an OCA group if they execute on the same tick. For example, if the market price triggers two stop orders from strategy.order calls with the same oca_* arguments, the strategy cannot fully or partially cancel either one.

### See also
This is one of the arguments that can be supplied to the default_qty_type parameter in the strategy declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in strategy.entry or strategy.order function calls. It specifies that a percentage (0-100) of equity will be used to enter trades.

### See also
- strategy.entry
- strategy.exit
- strategy.order

## strategy.percent_of_equity
This is one of the arguments that can be supplied to the default_qty_type parameter in the strategy declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in strategy.entry or strategy.order function calls. It specifies that a percentage (0-100) of equity will be used to enter trades.

### Type
const string

### Example
A named constant for use with the direction parameter of the strategy.entry and

strategy.order commands. It specifies that the command creates a sell order.

### See also
A named constant for use with the direction parameter of the strategy.entry and strategy.order commands. It specifies that the command creates a sell order.

### See also
- strategy

## strategy.short
A named constant for use with the direction parameter of the strategy.entry and strategy.order commands. It specifies that the command creates a sell order.

### Type
const strategy_direction

### See also
Vertical text alignment for box.new, box.set_text_valign, table.cell and table.cell_set_text_valign functions.

### See also
- strategy.entry
- strategy.exit
- strategy.order

## text.align_bottom
Vertical text alignment for box.new, box.set_text_valign, table.cell and table.cell_set_text_valign functions.

### Type
const string

### See also
Text alignment for box.new, box.set_text_halign, box.set_text_valign, label.new and label.set_textalign functions.

### See also

- table.cell
- table.cell_set_text_valign
- text.align_center
- text.align_left
- text.align_right

## text.align_center
Text alignment for box.new, box.set_text_halign, box.set_text_valign, label.new and label.set_textalign functions.

### Type
const string

### See also
Horizontal text alignment for box.new, box.set_text_halign, label.new and label.set_textalign functions.

### See also
- label.new
- label.set_style
- text.align_left
- text.align_right

## text.align_left
Horizontal text alignment for box.new, box.set_text_halign, label.new and label.set_textalign functions.

### Type
const string

### See also
Horizontal text alignment for box.new, box.set_text_halign, label.new and label.set_textalign functions.

### See also
- label.new
- label.set_style
- text.align_center
- text.align_right

## text.align_right
Horizontal text alignment for box.new, box.set_text_halign, label.new and label.set_textalign functions.

### Type
const string

### See also
Vertical text alignment for box.new, box.set_text_valign, table.cell and table.cell_set_text_valign functions.

### See also
- label.new
- label.set_style
- text.align_center
- text.align_left

## text.align_top
Vertical text alignment for box.new, box.set_text_valign, table.cell and table.cell_set_text_valign functions.

### Type
const string

### See also
A named constant for use with the text_formatting parameter of the label.new(), box.new(), table.cell(), and *set_text_formatting() functions. Makes the text bold.

### See also
- table.cell
- table.cell_set_text_valign
- text.align_center
- text.align_left
- text.align_right

## text.format_bold

A named constant for use with the text_formatting parameter of the label.new(), box.new(), table.cell(), and *set_text_formatting() functions. Makes the text bold.

### Type
const text_format

### See also
A named constant for use with the text_formatting parameter of the label.new(), box.new(), table.cell(), and *set_text_formatting() functions. Italicizes the text.

### See also
- label.new
- box.new
- table.cell

## text.format_italic
A named constant for use with the text_formatting parameter of the label.new(), box.new(), table.cell(), and *set_text_formatting() functions. Italicizes the text.

### Type
const text_format

### See also
A named constant for use with the text_formatting parameter of the label.new(), box.new(), table.cell(), and *set_text_formatting() functions. Signifies no special text formatting.

### See also
- label.new
- box.new
- table.cell

## text.format_none
A named constant for use with the text_formatting parameter of the label.new(), box.new(), table.cell(), and *set_text_formatting() functions. Signifies no special text formatting.

### Type

const text_format

### See also
Automatic wrapping mode for box.new and box.set_text_wrap functions.

### See also
- label.new
- box.new
- table.cell

## text.wrap_auto
Automatic wrapping mode for box.new and box.set_text_wrap functions.

### Type
const string

### See also
Disabled wrapping mode for box.new and box.set_text_wrap functions.

### See also
- box.new
- box.set_text
- box.set_text_wrap

## text.wrap_none
Disabled wrapping mode for box.new and box.set_text_wrap functions.

### Type
const string

### See also
Literal representing one of the values a bool variable can hold, or an expression can evaluate to when it uses comparison or logical operators.

### See also
- box.new
- box.set_text

- box.set_text_wrap

## true
Literal representing one of the values a bool variable can hold, or an expression can evaluate to when it uses comparison or logical operators.

### Remarks
See the User Manual for comparison operators and logical operators.

### See also
A constant that specifies how functions that create and modify Pine drawings interpret x-coordinates. If xloc = xloc.bar_index, the drawing object treats each x-coordinate as a bar_index value.

### See also
- bool

## xloc.bar_index
A constant that specifies how functions that create and modify Pine drawings interpret x-coordinates. If xloc = xloc.bar_index, the drawing object treats each x-coordinate as a bar_index value.

### Type
const string

### See also
A constant that specifies how functions that create and modify Pine drawings interpret x-coordinates. If xloc = xloc.bar_time, the drawing object treats each x-coordinate as a UNIX timestamp, expressed in milliseconds.

### See also
- xloc.bar_time
- line.new
- label.new
- box.new
- polyline.new
- line.set_xloc
- label.set_xloc

## xloc.bar_time
A constant that specifies how functions that create and modify Pine drawings interpret x-coordinates. If xloc = xloc.bar_time, the drawing object treats each x-coordinate as a UNIX timestamp, expressed in milliseconds.

### Type
const string

### See also
A named constant that specifies the algorithm of interpretation of y-value in function label.new.

### See also
- xloc.bar_index
- line.new
- label.new
- box.new
- polyline.new
- line.set_xloc
- label.set_xloc
- xloc.bar_index

## yloc.abovebar
A named constant that specifies the algorithm of interpretation of y-value in function label.new.

### Type
const string

### See also
A named constant that specifies the algorithm of interpretation of y-value in function label.new.

### See also
- label.new
- label.set_yloc
- yloc.price
- yloc.belowbar

## yloc.belowbar
A named constant that specifies the algorithm of interpretation of y-value in function label.new.

### Type
const string

### See also
A named constant that specifies the algorithm of interpretation of y-value in function label.new.

### See also
- label.new
- label.set_yloc
- yloc.price
- yloc.abovebar

## yloc.price
A named constant that specifies the algorithm of interpretation of y-value in function label.new.

### Type
const string

### See also
Creates an alert trigger for an indicator or strategy, with a specified frequency, when called on the latest realtime bar. To activate alerts for a script containing calls to this function, open the "Create Alert" dialog box, then select the script name and "Any alert() function call" in the "Condition" section.

### See also
- label.new
- label.set_yloc
- yloc.abovebar
- yloc.belowbar

## alert()

Creates an alert trigger for an indicator or strategy, with a specified frequency, when called on the latest realtime bar. To activate alerts for a script containing calls to this function, open the "Create Alert" dialog box, then select the script name and "Any alert() function call" in the "Condition" section.

### Syntax
```
message (series string) The message to send when the alert occurs.
```

### Arguments
```
message (series string) The message to send when the alert occurs.
```

### Example
The alert() function does not display information on the chart.

### Remarks
The alert() function does not display information on the chart.

### See also
Creates alert condition, that is available in Create Alert dialog. Please note, that alertcondition does NOT create an alert, it just gives you more options in Create Alert dialog. Also, alertcondition effect is invisible on chart.

### See also
- alertcondition

## alertcondition()
Creates alert condition, that is available in Create Alert dialog. Please note, that alertcondition does NOT create an alert, it just gives you more options in Create Alert dialog. Also, alertcondition effect is invisible on chart.

### Syntax
```
condition (series bool) Series of boolean values that is used for alert. True values mean alert fire, false - no alert. Required argument.
```

### Arguments
```
condition (series bool) Series of boolean values that is used for alert. True
values mean alert fire, false - no alert. Required argument.
```

### Example
Please note that an alertcondition call generates an additional plot. All such calls are taken into account when we calculate the number of the output series per script.

### Remarks
Please note that an alertcondition call generates an additional plot. All such calls are taken into account when we calculate the number of the output series per script.

### See also
Returns an array containing the absolute value of each element in the original array.

### See also
- alert

## array.abs()
Returns an array containing the absolute value of each element in the original array.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### See also
The function returns the mean of an array's elements.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.avg()
The function returns the mean of an array's elements.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
Mean of array's elements.

### Returns
```
Mean of array's elements.
```

### See also
The function returns the index of the value, or -1 if the value is not found. The array to search must be sorted in ascending order.

### See also
- array.new_float
- array.max
- array.min
- array.stdev

## array.binary_search()
The function returns the index of the value, or -1 if the value is not found. The

array to search must be sorted in ascending order.

### Syntax
```
id (array<int/float>) An array object.
```

### Arguments
```
id (array<int/float>) An array object.
```

### Example
A binary search works on arrays pre-sorted in ascending order. It begins by comparing an element in the middle of the array with the target value. If the element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array. If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### Remarks
A binary search works on arrays pre-sorted in ascending order. It begins by comparing an element in the middle of the array with the target value. If the element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array. If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### See also
The function returns the index of the value if it is found. When the value is not found, the function returns the index of the next smallest element to the left of where the value would lie if it was in the array. The array to search must be sorted in ascending order.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse

- order.ascending
- order.descending

## array.binary_search_leftmost()
The function returns the index of the value if it is found. When the value is not found, the function returns the index of the next smallest element to the left of where the value would lie if it was in the array. The array to search must be sorted in ascending order.

### Syntax
```
id (array<int/float>) An array object.
```

### Arguments
```
id (array<int/float>) An array object.
```

### Example
A binary search works on arrays pre-sorted in ascending order. It begins by comparing an element in the middle of the array with the target value. If the element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array. If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### Example
A binary search works on arrays pre-sorted in ascending order. It begins by comparing an element in the middle of the array with the target value. If the element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array. If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### Remarks
A binary search works on arrays pre-sorted in ascending order. It begins by comparing an element in the middle of the array with the target value. If the

element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array. If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### See also
The function returns the index of the value if it is found. When the value is not found, the function returns the index of the element to the right of where the value would lie if it was in the array. The array must be sorted in ascending order.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.binary_search_rightmost()
The function returns the index of the value if it is found. When the value is not found, the function returns the index of the element to the right of where the value would lie if it was in the array. The array must be sorted in ascending order.

### Syntax
```
id (array<int/float>) An array object.
```

### Arguments
```
id (array<int/float>) An array object.
```

### Example
A binary search works on sorted arrays in ascending order. It begins by comparing an element in the middle of the array with the target value. If the element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array.

If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### Example
A binary search works on sorted arrays in ascending order. It begins by comparing an element in the middle of the array with the target value. If the element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array. If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### Remarks
A binary search works on sorted arrays in ascending order. It begins by comparing an element in the middle of the array with the target value. If the element matches the target value, its position in the array is returned. If the element's value is greater than the target value, the search continues in the lower half of the array. If the element's value is less than the target value, the search continues in the upper half of the array. By doing this recursively, the algorithm progressively eliminates smaller and smaller portions of the array in which the target value cannot lie.

### See also
The function removes all elements from an array.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.clear()
The function removes all elements from an array.

### Syntax
```
id (any array type) An array object.
```

```

### Arguments
```
id (any array type) An array object.
```

### Example
The function is used to merge two arrays. It pushes all elements from the second array to the first array, and returns the first array.

### See also
The function is used to merge two arrays. It pushes all elements from the second array to the first array, and returns the first array.

### See also
- array.new_float
- array.insert
- array.push
- array.remove
- array.pop

## array.concat()
The function is used to merge two arrays. It pushes all elements from the second array to the first array, and returns the first array.

### Syntax
```
id1 (any array type) The first array object.
```

### Arguments
```
id1 (any array type) The first array object.
```

### Example
The first array with merged elements from the second array.
```

### Returns
```
The first array with merged elements from the second array.
```

### See also
The function creates a copy of an existing array.

### See also
- array.new_float
- array.insert
- array.slice

## array.copy()
The function creates a copy of an existing array.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
A copy of an array.

### Returns
```
A copy of an array.
```

### See also
The function returns the covariance of two arrays.

### See also
- array.new_float

- array.get
- array.slice
- array.sort

## array.covariance()
The function returns the covariance of two arrays.

### Syntax
```
id1 (array<int/float>) An array object.
```

### Arguments
```
id1 (array<int/float>) An array object.
```

### Example
The covariance of two arrays.

### Returns
```
The covariance of two arrays.
```

### Remarks
If biased is true, function will calculate using a biased estimate of the entire population, if false - unbiased estimate of a sample.

### See also
Returns true if all elements of the id array are true, false otherwise.

### See also
- array.new_float
- array.max
- array.stdev
- array.avg
- array.variance

## array.every()
Returns true if all elements of the id array are true, false otherwise.

### Syntax
```
id (array<bool>) An array object.
```

### Arguments
```
id (array<bool>) An array object.
```

### Remarks
This function also works with arrays of int and float types, in which case zero values are considered false, and all others true.

### See also
The function sets elements of an array to a single value. If no index is specified, all elements are set. If only a start index (default 0) is supplied, the elements starting at that index are set. If both index parameters are used, the elements from the starting index up to but not including the end index (default na) are set.

### See also
- array.some
- array.get

## array.fill()
The function sets elements of an array to a single value. If no index is specified, all elements are set. If only a start index (default 0) is supplied, the elements starting at that index are set. If both index parameters are used, the elements from the starting index up to but not including the end index (default na) are set.

### Syntax
```
id (any array type) An array object.
```

### Arguments

```
id (any array type) An array object.
```

### Example
Returns the array's first element. Throws a runtime error if the array is empty.

### See also
Returns the array's first element. Throws a runtime error if the array is empty.

### See also
- array.new_float
- array.set
- array.slice

## array.first()
Returns the array's first element. Throws a runtime error if the array is empty.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The function takes a variable number of arguments with one of the types: int, float, bool, string, label, line, color, box, table, linefill, and returns an array of the corresponding type.

### See also
The function takes a variable number of arguments with one of the types: int, float, bool, string, label, line, color, box, table, linefill, and returns an array of the corresponding type.

### See also

- array.last
- array.get

## array.from()
The function takes a variable number of arguments with one of the types: int, float, bool, string, label, line, color, box, table, linefill, and returns an array of the corresponding type.

### Syntax & Overloads
arg0, arg1, ... (<arg..._type>) Array arguments.

### Arguments
```
arg0, arg1, ... (<arg..._type>) Array arguments.
```

### Example
The array element's value.

### Returns
```
The array element's value.
```

### Remarks
This function can accept up to 4,000 'int', 'float', 'bool', or 'color' arguments. For all other types, including user-defined types, the limit is 999.

## array.get()
The function returns the value of the element at the specified index.

### Syntax
```
id (any array type) An array object.
```

### Arguments

```
id (any array type) An array object.
```

### Example
The array element's value.

### Returns
```
The array element's value.
```

### Remarks
If the index is positive, the function counts forwards from the beginning of the array to the end. The index of the first element is 0, and the index of the last element is array.size() - 1. If the index is negative, the function counts backwards from the end of the array to the beginning. In this case, the index of the last element is -1, and the index of the first element is negative array.size(). For example, for an array that contains three elements, all of the following are valid arguments for the index parameter: 0, 1, 2, -1, -2, -3.

### See also
The function returns true if the value was found in an array, false otherwise.

### See also
- array.new_float
- array.set
- array.slice
- array.sort

## array.includes()
The function returns true if the value was found in an array, false otherwise.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
```

id (any array type) An array object.
```

### Example
True if the value was found in the array, false otherwise.

### Returns
```
True if the value was found in the array, false otherwise.
```

### See also
The function returns the index of the first occurrence of the value, or -1 if the value is not found.

### See also
- array.new_float
- array.indexof
- array.shift
- array.remove
- array.insert

## array.indexof()
The function returns the index of the first occurrence of the value, or -1 if the value is not found.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The index of an element.
```

### Returns
```
The index of an element.
```

### See also
The function changes the contents of an array by adding new elements in place.

### See also
- array.lastindexof
- array.get
- array.lastindexof
- array.remove
- array.insert

## array.insert()
The function changes the contents of an array by adding new elements in place.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
If the index is positive, the function counts forwards from the beginning of the array to the end. The index of the first element is 0, and the index of the last element is array.size() - 1. If the index is negative, the function counts backwards from the end of the array to the beginning. In this case, the index of the last element is -1, and the index of the first element is negative array.size(). For example, for an array that contains three elements, all of the following are valid arguments for the index parameter: 0, 1, 2, -1, -2, -3.

### Remarks
If the index is positive, the function counts forwards from the beginning of the array to the end. The index of the first element is 0, and the index of the last element is array.size() - 1. If the index is negative, the function counts

backwards from the end of the array to the beginning. In this case, the index of the last element is -1, and the index of the first element is negative array.size(). For example, for an array that contains three elements, all of the following are valid arguments for the index parameter: 0, 1, 2, -1, -2, -3.

### See also
The function creates and returns a new string by concatenating all the elements of an array, separated by the specified separator string.

### See also
- array.new_float
- array.set
- array.push
- array.remove
- array.pop
- array.unshift

## array.join()
The function creates and returns a new string by concatenating all the elements of an array, separated by the specified separator string.

### Syntax
```
id (array<int/float/string>) An array object.
```

### Arguments
```
id (array<int/float/string>) An array object.
```

### Example
Returns the array's last element. Throws a runtime error if the array is empty.

### See also
Returns the array's last element. Throws a runtime error if the array is empty.

### See also
- array.new_float
- array.set

- array.insert
- array.remove
- array.pop
- array.unshift

## array.last()
Returns the array's last element. Throws a runtime error if the array is empty.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The function returns the index of the last occurrence of the value, or -1 if the value is not found.

### See also
The function returns the index of the last occurrence of the value, or -1 if the value is not found.

### See also
- array.first
- array.get

## array.lastindexof()
The function returns the index of the last occurrence of the value, or -1 if the value is not found.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The index of an element.

### Returns
```
The index of an element.
```

### See also
The function returns the greatest value, or the nth greatest value in a given array.

### See also
- array.new_float
- array.set
- array.push
- array.remove
- array.insert

## array.max()
The function returns the greatest value, or the nth greatest value in a given array.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
The greatest or the nth greatest value in the array.

### Returns
```
The greatest or the nth greatest value in the array.
```

### See also
The function returns the median of an array's elements.

### See also
- array.new_float
- array.min
- array.sum

## array.median()
The function returns the median of an array's elements.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
The median of the array's elements.

### Returns
```
The median of the array's elements.
```

### See also
The function returns the smallest value, or the nth smallest value in a given array.

### See also
- array.median

- array.avg
- array.variance
- array.min

## array.min()
The function returns the smallest value, or the nth smallest value in a given array.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
The smallest or the nth smallest value in the array.

### Returns
```
The smallest or the nth smallest value in the array.
```

### See also
The function returns the mode of an array's elements. If there are several values with the same frequency, it returns the smallest value.

### See also
- array.new_float
- array.max
- array.sum

## array.mode()
The function returns the mode of an array's elements. If there are several values with the same frequency, it returns the smallest value.

### Syntax & Overloads

id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
The most frequently occurring value from the id array. If none exists, returns the smallest value instead.

### Returns
```
The most frequently occurring value from the id array. If none exists, returns the smallest value instead.
```

### See also
The function creates a new array object of bool type elements.

### See also
- array.new_float
- ta.mode
- matrix.mode
- array.avg
- array.variance
- array.min

## array.new_bool()
The function creates a new array object of bool type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of box type elements.

### See also
- array.new_float
- array.get
- array.slice
- array.sort

## array.new_box()
The function creates a new array object of box type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns

```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of color type elements.

### See also
- array.new_float
- array.get
- array.slice

## array.new_color()
The function creates a new array object of color type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of float type elements.

### See also
- array.new_float
- array.get
- array.slice
- array.sort

## array.new_float()
The function creates a new array object of float type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of int type elements.

### See also
- array.new_color

- array.new_bool
- array.get
- array.slice
- array.sort

## array.new_int()
The function creates a new array object of int type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of label type elements.

### See also
- array.new_float
- array.get
- array.slice
- array.sort

## array.new_label()
The function creates a new array object of label type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of line type elements.

### See also
- array.new_float
- array.get
- array.slice

## array.new_line()
The function creates a new array object of line type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of linefill type elements.

### See also
- array.new_float
- array.get
- array.slice

## array.new_linefill()
The function creates a new array object of linefill type elements.

### Syntax
```
size (series int) Initial size of an array.
```

### Arguments
```
size (series int) Initial size of an array.
```

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

## array.new_string()
The function creates a new array object of string type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of table type elements.

### See also
- array.new_float
- array.get

- array.slice

## array.new_table()
The function creates a new array object of table type elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
The function creates a new array object of <type> elements.

### See also
- array.new_float
- array.get
- array.slice

## array.new<type>()
The function creates a new array object of <type> elements.

### Syntax
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Arguments
```
size (series int) Initial size of an array. Optional. The default is 0.
```

### Example
The ID of an array object which may be used in other array.*() functions.

### Example
The ID of an array object which may be used in other array.*() functions.

### Example
The ID of an array object which may be used in other array.*() functions.

### Example
The ID of an array object which may be used in other array.*() functions.

### Returns
```
The ID of an array object which may be used in other array.*() functions.
```

### Remarks
An array index starts from 0.

### See also
Returns the value for which the specified percentage of array values (percentile) are less than or equal to it, using linear interpolation.

### See also
- array.from
- array.push
- array.get
- array.size
- array.remove

- array.shift
- array.sum

## array.percentile_linear_interpolation()
Returns the value for which the specified percentage of array values (percentile) are less than or equal to it, using linear interpolation.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Remarks
In statistics, the percentile is the percent of ranking items that appear at or below a certain score. This measurement shows the percentage of scores within a standard frequency distribution that is lower than the percentile rank you're measuring. Linear interpolation estimates the value between two ranks.

### See also
Returns the value for which the specified percentage of array values (percentile) are less than or equal to it, using the nearest-rank method.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.percentile_nearest_rank()
Returns the value for which the specified percentage of array values (percentile) are less than or equal to it, using the nearest-rank method.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Remarks
In statistics, the percentile is the percent of ranking items that appear at or below a certain score. This measurement shows the percentage of scores within a standard frequency distribution that is lower than the percentile rank you're measuring.

### See also
Returns the percentile rank of the element at the specified index.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.percentrank()
Returns the percentile rank of the element at the specified index.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Remarks
Percentile rank is the percentage of how many elements in the array are less than or equal to the reference value.

### See also

The function removes the last element from an array and returns its value.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.pop()
The function removes the last element from an array and returns its value.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The value of the removed element.

### Returns
```
The value of the removed element.
```

### See also
The function appends a value to an array.

### See also
- array.new_float
- array.set
- array.push
- array.remove
- array.insert

- array.shift

## array.push()
The function appends a value to an array.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The function returns the difference between the min and max values from a given array.

### See also
The function returns the difference between the min and max values from a given array.

### See also
- array.new_float
- array.set
- array.insert
- array.remove
- array.pop
- array.unshift

## array.range()
The function returns the difference between the min and max values from a given array.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
The difference between the min and max values in the array.

### Returns
```
The difference between the min and max values in the array.
```

### See also
The function changes the contents of an array by removing the element with the specified index.

### See also
- array.new_float
- array.min
- array.max
- array.sum

## array.remove()
The function changes the contents of an array by removing the element with the specified index.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The value of the removed element.

### Returns
```
The value of the removed element.
```

### Remarks
If the index is positive, the function counts forwards from the beginning of the array to the end. The index of the first element is 0, and the index of the last element is array.size() - 1. If the index is negative, the function counts backwards from the end of the array to the beginning. In this case, the index of the last element is -1, and the index of the first element is negative array.size(). For example, for an array that contains three elements, all of the following are valid arguments for the index parameter: 0, 1, 2, -1, -2, -3.

### See also
The function reverses an array. The first array element becomes the last, and the last array element becomes the first.

### See also
- array.new_float
- array.set
- array.push
- array.insert
- array.pop
- array.shift

## array.reverse()
The function reverses an array. The first array element becomes the last, and the last array element becomes the first.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The function sets the value of the element at the specified index.

### See also
The function sets the value of the element at the specified index.

### See also
- array.new_float
- array.sort
- array.push
- array.set
- array.avg

## array.set()
The function sets the value of the element at the specified index.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
If the index is positive, the function counts forwards from the beginning of the array to the end. The index of the first element is 0, and the index of the last element is array.size() - 1. If the index is negative, the function counts backwards from the end of the array to the beginning. In this case, the index of the last element is -1, and the index of the first element is negative array.size(). For example, for an array that contains three elements, all of the following are valid arguments for the index parameter: 0, 1, 2, -1, -2, -3.

### Remarks
If the index is positive, the function counts forwards from the beginning of the array to the end. The index of the first element is 0, and the index of the last element is array.size() - 1. If the index is negative, the function counts backwards from the end of the array to the beginning. In this case, the index of the last element is -1, and the index of the first element is negative

array.size(). For example, for an array that contains three elements, all of the following are valid arguments for the index parameter: 0, 1, 2, -1, -2, -3.

### See also
The function removes an array's first element and returns its value.

### See also
- array.new_float
- array.get
- array.slice

## array.shift()
The function removes an array's first element and returns its value.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The value of the removed element.

### Returns
```
The value of the removed element.
```

### See also
The function returns the number of elements in an array.

### See also
- array.unshift
- array.set
- array.push

- array.remove
- array.includes

## array.size()
The function returns the number of elements in an array.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The number of elements in the array.

### Returns
```
The number of elements in the array.
```

### See also
The function creates a slice from an existing array. If an object from the slice changes, the changes are applied to both the new and the original arrays.

### See also
- array.new_float
- array.sum
- array.slice
- array.sort

## array.slice()
The function creates a slice from an existing array. If an object from the slice changes, the changes are applied to both the new and the original arrays.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
A shallow copy of an array's slice.

### Returns
```
A shallow copy of an array's slice.
```

### See also
Returns true if at least one element of the id array is true, false otherwise.

### See also
- array.new_float
- array.get
- array.slice
- array.sort

## array.some()
Returns true if at least one element of the id array is true, false otherwise.

### Syntax
```
id (array<bool>) An array object.
```

### Arguments
```
id (array<bool>) An array object.
```

### Remarks
This function also works with arrays of int and float types, in which case zero values are considered false, and all others true.

### See also
The function sorts the elements of an array.

### See also
- array.every
- array.get

## array.sort()
The function sorts the elements of an array.

### Syntax
```
id (array<int/float/string>) An array object.
```

### Arguments
```
id (array<int/float/string>) An array object.
```

### Example
Returns an array of indices which, when used to index the original array, will access its elements in their sorted order. It does not modify the original array.

### See also
Returns an array of indices which, when used to index the original array, will access its elements in their sorted order. It does not modify the original array.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.sort_indices()
Returns an array of indices which, when used to index the original array, will access its elements in their sorted order. It does not modify the original array.

### Syntax
```
id (array<int/float/string>) An array object.
```

### Arguments
```
id (array<int/float/string>) An array object.
```

### Example
The function returns the array of standardized elements.

### See also
The function returns the array of standardized elements.

### See also
- array.new_float
- array.insert
- array.slice
- array.reverse
- order.ascending
- order.descending

## array.standardize()
The function returns the array of standardized elements.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

```

### Example
The array of standardized elements.

### Returns
```
The array of standardized elements.
```

### See also
The function returns the standard deviation of an array's elements.

### See also
- array.max
- array.min
- array.mode
- array.avg
- array.variance
- array.stdev

## array.stdev()
The function returns the standard deviation of an array's elements.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
The standard deviation of the array's elements.

### Returns
```
The standard deviation of the array's elements.
```
```

### Remarks
If biased is true, function will calculate using a biased estimate of the entire population, if false - unbiased estimate of a sample.

### See also
The function returns the sum of an array's elements.

### See also
- array.new_float
- array.max
- array.min
- array.avg

## array.sum()
The function returns the sum of an array's elements.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

### Example
The sum of the array's elements.

### Returns
```
The sum of the array's elements.
```

### See also
The function inserts the value at the beginning of the array.

### See also
- array.new_float

- array.max
- array.min

## array.unshift()
The function inserts the value at the beginning of the array.

### Syntax
```
id (any array type) An array object.
```

### Arguments
```
id (any array type) An array object.
```

### Example
The function returns the variance of an array's elements.

### See also
The function returns the variance of an array's elements.

### See also
- array.shift
- array.set
- array.insert
- array.remove
- array.indexof

## array.variance()
The function returns the variance of an array's elements.

### Syntax & Overloads
id (array<int/float>) An array object.

### Arguments
```
id (array<int/float>) An array object.
```

```

### Example
The variance of the array's elements.

### Returns
```
The variance of the array's elements.
```

### Remarks
If biased is true, function will calculate using a biased estimate of the entire population, if false - unbiased estimate of a sample.

### See also
Set color of bars.

### See also
- array.new_float
- array.stdev
- array.min
- array.avg
- array.covariance

## barcolor()
Set color of bars.

### Syntax
```
- color (series color) Color of bars. You can use constants like 'red' or '#ff001a' as well as complex expressions like 'close >= open ? color.green : color.red'. Required argument.
```

### Arguments
```
- color (series color) Color of bars. You can use constants like 'red' or '#ff001a' as well as complex expressions like 'close >= open ? color.green : color.red'. Required argument.
```
```

### Example
Fill background of bars with specified color.

### See also
Fill background of bars with specified color.

### See also
- bgcolor
- plot
- fill

## bgcolor()
Fill background of bars with specified color.

### Syntax
```
- color (series color) Color of the filled background. You can use constants like 'red' or '#ff001a' as well as complex expressions like 'close >= open ? color.green : color.red'. Required argument.
```

### Arguments
```
- color (series color) Color of the filled background. You can use constants like 'red' or '#ff001a' as well as complex expressions like 'close >= open ? color.green : color.red'. Required argument.
```

### Example
Converts the x value to a bool value. Returns false if x is na, false, or an int/float value equal to 0. Returns true for all other possible values.

### See also
Converts the x value to a bool value. Returns false if x is na, false, or an int/float value equal to 0. Returns true for all other possible values.

### See also
- barcolor

- plot
- fill

## bool()
Converts the x value to a bool value. Returns false if x is na, false, or an int/float value equal to 0. Returns true for all other possible values.

### Syntax & Overloads
x (simple int/float/bool) The value to convert to the specified type, usually na.

### Arguments
```
x (simple int/float/bool) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to bool.
```

### See also
Casts na to box.

### See also
- float
- int
- color
- string
- line
- label

## box()
Casts na to box.

### Syntax
```
x (series box) The value to convert to the specified type, usually na.
```

### Arguments
```
x (series box) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to box.
```

### See also
Clones the box object.

### See also
- float
- int
- bool
- color
- string
- line
- label

## box.copy()
Clones the box object.

### Syntax
```
id (series box) Box object.
```

### Arguments
```
id (series box) Box object.
```

### Example
Deletes the specified box object. If it has already been deleted, does nothing.

### See also

Deletes the specified box object. If it has already been deleted, does nothing.

### See also
- box.new
- box.delete

## box.delete()
Deletes the specified box object. If it has already been deleted, does nothing.

### Syntax
```
id (series box) A box object to delete.
```

### Arguments
```
id (series box) A box object to delete.
```

### See also
Returns the price value of the bottom border of the box.

### See also
- box.new

## box.get_bottom()
Returns the price value of the bottom border of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### Returns
```
The price value.
```

### See also
Returns the bar index or the UNIX time (depending on the last value used for 'xloc') of the left border of the box.

### See also
- box.new
- box.set_bottom

## box.get_left()
Returns the bar index or the UNIX time (depending on the last value used for 'xloc') of the left border of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### Returns
```
A bar index or a UNIX timestamp (in milliseconds).
```

### See also
Returns the bar index or the UNIX time (depending on the last value used for 'xloc') of the right border of the box.

### See also
- box.new
- box.set_left

## box.get_right()
Returns the bar index or the UNIX time (depending on the last value used for 'xloc') of the right border of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### Returns
```
A bar index or a UNIX timestamp (in milliseconds).
```

### See also
Returns the price value of the top border of the box.

### See also
- box.new
- box.set_right

## box.get_top()
Returns the price value of the top border of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

```

### Returns
```
The price value.
```

### See also
Creates a new box object.

### See also
- box.new
- box.set_top

## box.new()
Creates a new box object.

### Syntax & Overloads
top_left (chart.point) A chart.point object that specifies the top-left corner location of the box.

### Arguments
```
top_left (chart.point) A chart.point object that specifies the top-left corner
location of the box.
```

### Example
The ID of a box object which may be used in box.set_*() and box.get_*() functions.

### Returns
```
The ID of a box object which may be used in box.set_*() and box.get_*() functions.
```

### See also
Sets the background color of the box.
```

### See also
- box.delete
- box.get_left
- box.get_top
- box.get_right
- box.get_bottom
- box.set_top_left_point
- box.set_left
- box.set_top
- box.set_bottom_right_point
- box.set_right
- box.set_bottom
- box.set_border_color
- box.set_bgcolor
- box.set_border_width
- box.set_border_style
- box.set_extend
- box.set_text
- box.set_text_formatting

## box.set_bgcolor()
Sets the background color of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the border color of the box.

### See also
- box.new

## box.set_border_color()
Sets the border color of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the border style of the box.

### See also
- box.new

## box.set_border_style()
Sets the border style of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the border width of the box.

### See also
- box.new
- line.style_solid

## box.set_border_width()
Sets the border width of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the bottom coordinate of the box.

### See also
- box.new

## box.set_bottom()
Sets the bottom coordinate of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the bottom-right corner location of the id box to point.

### See also
- box.new
- box.get_bottom

## box.set_bottom_right_point()
Sets the bottom-right corner location of the id box to point.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

## box.set_extend()
Sets extending type of the border of this box object. When extend.none is used, the horizontal borders start at the left border and end at the right border. With extend.left or extend.right, the horizontal borders are extended indefinitely to the left or right of the box, respectively. With extend.both, the horizontal borders are extended on both sides.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the left coordinate of the box.

### See also
- box.new
- extend.none

## box.set_left()
Sets the left coordinate of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the left and top coordinates of the box.

### See also
- box.new
- box.get_left

## box.set_lefttop()
Sets the left and top coordinates of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the right coordinate of the box.

### See also
- box.new

- box.get_left
- box.get_top

## box.set_right()
Sets the right coordinate of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the right and bottom coordinates of the box.

### See also
- box.new
- box.get_right

## box.set_rightbottom()
Sets the right and bottom coordinates of the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
The function sets the text in the box.

### See also
- box.new
- box.get_right
- box.get_bottom

## box.set_text()
The function sets the text in the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
The function sets the color of the text inside the box.

### See also
- box.set_text_color
- box.set_text_size
- box.set_text_valign
- box.set_text_halign
- box.set_text_formatting

## box.set_text_color()
The function sets the color of the text inside the box.

### Syntax
```
id (series box) A box object.
```

### Arguments

```
id (series box) A box object.
```

### See also
The function sets the font family of the text inside the box.

### See also
- box.set_text
- box.set_text_size
- box.set_text_valign
- box.set_text_halign

## box.set_text_font_family()
The function sets the font family of the text inside the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### Example
Sets the formatting attributes the drawing applies to displayed text.

### See also
Sets the formatting attributes the drawing applies to displayed text.

### See also
- box.new
- font.family_default
- font.family_monospace

## box.set_text_formatting()

Sets the formatting attributes the drawing applies to displayed text.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
The function sets the horizontal alignment of the box's text.

### See also
- box.set_text_color
- box.set_text_size
- box.set_text_valign
- box.set_text_halign
- box.set_text

## box.set_text_halign()
The function sets the horizontal alignment of the box's text.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
The function sets the size of the box's text.

### See also

- box.set_text
- box.set_text_size
- box.set_text_valign
- box.set_text_color

## box.set_text_size()
The function sets the size of the box's text.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
The function sets the vertical alignment of a box's text.

### See also
- box.set_text
- box.set_text_color
- box.set_text_valign
- box.set_text_halign

## box.set_text_valign()
The function sets the vertical alignment of a box's text.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
The function sets the mode of wrapping of the text inside the box.

### See also
- box.set_text
- box.set_text_size
- box.set_text_color
- box.set_text_halign

## box.set_text_wrap()
The function sets the mode of wrapping of the text inside the box.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the top coordinate of the box.

### See also
- box.set_text
- box.set_text_size
- box.set_text_valign
- box.set_text_halign
- box.set_text_color

## box.set_top()
Sets the top coordinate of the box.

### Syntax
```
```

```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

### See also
Sets the top-left corner location of the id box to point.

### See also
- box.new
- box.get_top

## box.set_top_left_point()
Sets the top-left corner location of the id box to point.

### Syntax
```
id (series box) A box object.
```

### Arguments
```
id (series box) A box object.
```

## chart.point.copy()
Creates a copy of a chart.point object with the specified id.

### Syntax
```
id (chart.point) A chart.point object.
```

### Arguments

```
id (chart.point) A chart.point object.
```

## chart.point.from_index()
Returns a chart.point object with index as its x-coordinate and price as its y-coordinate.

### Syntax
```
index (series int) The x-coordinate of the point, expressed as a bar index value.
```

### Arguments
```
index (series int) The x-coordinate of the point, expressed as a bar index value.
```

### Remarks
The time field values of chart.point instances returned from this function will be na, meaning drawing objects with xloc values set to xloc.bar_time will not work with them.

## chart.point.from_time()
Returns a chart.point object with time as its x-coordinate and price as its y-coordinate.

### Syntax
```
time (series int) The x-coordinate of the point, expressed as a UNIX time value, in milliseconds.
```

### Arguments
```
time (series int) The x-coordinate of the point, expressed as a UNIX time value, in milliseconds.
```

### Remarks
The index field values of chart.point instances returned from this function will be na, meaning drawing objects with xloc values set to xloc.bar_index will not work with them.

## chart.point.new()
Creates a new chart.point object with the specified time, index, and price.

### Syntax
```
time (series int) The x-coordinate of the point, expressed as a UNIX time value, in milliseconds.
```

### Arguments
```
time (series int) The x-coordinate of the point, expressed as a UNIX time value, in milliseconds.
```

### Remarks
Whether a drawing object uses a point's time or index field as an x-coordinate depends on the xloc type used in the function call that returned the drawing.

### See also
Returns a chart.point object with price as the y-coordinate

### See also
- polyline.new

## chart.point.now()
Returns a chart.point object with price as the y-coordinate

### Syntax
```
price (series int/float) The y-coordinate of the point. Optional. The default is
```

close.
```

### Arguments
```
price (series int/float) The y-coordinate of the point. Optional. The default is
close.
```

### Remarks
The chart.point instance returned from this function records values for its index and time fields on the bar it executed on, making it suitable for use with drawing objects of any xloc type.

## color()
Casts na to color

### Syntax & Overloads
x (const color) The value to convert to the specified type, usually na.

### Arguments
```
x (const color) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to color.
```

### See also
Retrieves the value of the color's blue component.

### See also
- float
- int
- bool
- string
- line
```

- label

## color.b()
Retrieves the value of the color's blue component.

### Syntax & Overloads
color (const color) Color.

### Arguments
```
color (const color) Color.
```

### Example
The value (0 to 255) of the color's blue component.

### Returns
```
The value (0 to 255) of the color's blue component.
```

## color.from_gradient()
Based on the relative position of value in the bottom_value to top_value range, the function returns a color from the gradient defined by bottom_color to top_color.

### Syntax
```
value (series int/float) Value to calculate the position-dependent color.
```

### Arguments
```
value (series int/float) Value to calculate the position-dependent color.
```

### Example
A color calculated from the linear gradient between bottom_color to top_color.

### Returns
```
A color calculated from the linear gradient between bottom_color to top_color.
```

### Remarks
Using this function will have an impact on the colors displayed in the script's "Settings/Style" tab. See the User Manual for more information.

## color.g()
Retrieves the value of the color's green component.

### Syntax & Overloads
color (const color) Color.

### Arguments
```
color (const color) Color.
```

### Example
The value (0 to 255) of the color's green component.

### Returns
```
The value (0 to 255) of the color's green component.
```

## color.new()
Function color applies the specified transparency to the given color.

### Syntax & Overloads
color (const color) Color to apply transparency to.

### Arguments
```
color (const color) Color to apply transparency to.
```

### Example
Color with specified transparency.

### Returns
```
Color with specified transparency.
```

### Remarks
Using arguments that are not constants (e.g., 'simple', 'input' or 'series') will have an impact on the colors displayed in the script's "Settings/Style" tab. See the User Manual for more information.

## color.r()
Retrieves the value of the color's red component.

### Syntax & Overloads
color (const color) Color.

### Arguments
```
color (const color) Color.
```

### Example
The value (0 to 255) of the color's red component.

### Returns
```
The value (0 to 255) of the color's red component.
```

## color.rgb()
Creates a new color with transparency using the RGB color model.

### Syntax & Overloads
red (const int/float) Red color component. Possible values are from 0 to 255.

### Arguments
```
red (const int/float) Red color component. Possible values are from 0 to 255.
```

### Example
Color with specified transparency.

### Returns
```
Color with specified transparency.
```

### Remarks
Using arguments that are not constants (e.g., 'simple', 'input' or 'series') will have an impact on the colors displayed in the script's "Settings/Style" tab. See the User Manual for more information.

## color.t()
Retrieves the color's transparency.

### Syntax & Overloads
color (const color) Color.

### Arguments
```
color (const color) Color.
```

### Example
The value (0-100) of the color's transparency.

### Returns
```
The value (0-100) of the color's transparency.
```

## dayofmonth()
time (series int) UNIX time in milliseconds.

### Syntax & Overloads
time (series int) UNIX time in milliseconds.

### Arguments
```
time (series int) UNIX time in milliseconds.
```

### Returns
```
Day of month (in exchange timezone) for provided UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
time (series int) UNIX time in milliseconds.

### See also
- dayofmonth
- time
- year
- month
- dayofweek
- hour
- minute
- second

## dayofweek()
time (series int) UNIX time in milliseconds.

### Syntax & Overloads
time (series int) UNIX time in milliseconds.

### Arguments
```
time (series int) UNIX time in milliseconds.
```

### Returns
```
Day of week (in exchange timezone) for provided UNIX time.
```

### Remarks
Note that this function returns the day based on the time of the bar's open. For overnight sessions (e.g. EURUSD, where Monday session starts on Sunday, 17:00) this value can be lower by 1 than the day of the trading day.

### See also
Fills background between two plots or hlines with a given color.

### See also
- dayofweek
- time
- year
- month
- dayofmonth
- hour
- minute
- second

## fill()
Fills background between two plots or hlines with a given color.

### Syntax & Overloads

hline1 (hline) The first hline object. Required argument.

### Arguments
```
hline1 (hline) The first hline object. Required argument.
```

### Example
Fill between two plots

### Example
Gradient fill between two horizontal lines

### Example
For a given series replaces NaN values with previous nearest non-NaN value.

### See also
For a given series replaces NaN values with previous nearest non-NaN value.

### See also
- plot
- barcolor
- bgcolor
- hline
- color.new

## fixnan()
For a given series replaces NaN values with previous nearest non-NaN value.

### Syntax & Overloads
source (series color) Source used for the calculation.

### Arguments
```
source (series color) Source used for the calculation.
```

### Returns

```
Series without na gaps.
```

### See also
Casts na to float

### See also
- na
- na
- nz

## float()
Casts na to float

### Syntax & Overloads
x (const int/float) The value to convert to the specified type, usually na.

### Arguments
```
x (const int/float) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to float.
```

### See also
Renders a horizontal line at a given fixed price level.

### See also
- int
- bool
- color
- string
- line
- label

## hline()
Renders a horizontal line at a given fixed price level.

### Syntax
```
price (input int/float) Price value at which the object will be rendered. Required argument.
```

### Arguments
```
price (input int/float) Price value at which the object will be rendered. Required argument.
```

### Example
An hline object, that can be used in fill

### Returns
```
An hline object, that can be used in fill
```

### See also
time (series int) UNIX time in milliseconds.

### See also
- fill

## hour()
time (series int) UNIX time in milliseconds.

### Syntax & Overloads
time (series int) UNIX time in milliseconds.

### Arguments
```
time (series int) UNIX time in milliseconds.
```

```

### Returns
```
Hour (in exchange timezone) for provided UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
This declaration statement designates the script as an indicator and sets a number of indicator-related properties.

### See also
- hour
- time
- year
- month
- dayofmonth
- dayofweek
- minute
- second

## indicator()
This declaration statement designates the script as an indicator and sets a number of indicator-related properties.

### Syntax
```
title (const string) The title of the script. It is displayed on the chart when no shorttitle argument is used, and becomes the publication's default title when publishing the script.
```

### Arguments
```
title (const string) The title of the script. It is displayed on the chart when no shorttitle argument is used, and becomes the publication's default title when publishing the script.

```

### Example
Every indicator script must have one indicator call.

### Remarks
Every indicator script must have one indicator call.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function automatically detects the type of the argument used for 'defval' and uses the corresponding input widget.

### See also
- strategy
- library

## input()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function automatically detects the type of the argument used for 'defval' and uses the corresponding input widget.

### Syntax & Overloads
defval (const int/float/bool/string/color or source-type built-ins) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where script users can change it. Source-type built-ins are built-in series float variables that specify the source of the calculation: close, hlc3, etc.

### Arguments
```
- defval (const int/float/bool/string/color or source-type built-ins) Determines
the default value of the input variable proposed in the script's "Settings/Inputs"
tab, from where script users can change it. Source-type built-ins are built-in
series float variables that specify the source of the calculation : close, hlc3,
etc.
```

### Example
Value of input variable.
```

### Returns
```
Value of input variable.
```

### Remarks
Result of input function always should be assigned to a variable, see examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a checkmark to the script's inputs.

### See also
- input.bool
- input.color
- input.int
- input.float
- input.string
- input.symbol
- input.timeframe
- input.text_area
- input.session
- input.source
- input.time

## input.bool()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a checkmark to the script's inputs.

### Syntax
```
defval (const bool) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it.
```

### Arguments
```
defval (const bool) Determines the default value of the input variable proposed in
```

```
the script's "Settings/Inputs" tab, from where the user can change it.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.bool function always should be assigned to a variable, see examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a color picker that allows the user to select a color and transparency, either from a palette or a hex value.

### See also
- input.int
- input.float
- input.string
- input.text_area
- input.symbol
- input.timeframe
- input.session
- input.source
- input.color
- input.time
- input

## input.color()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a color picker that allows the user to select a color and transparency, either from a palette or a hex value.

### Syntax

```
defval (const color) Determines the default value of the input variable proposed in
the script's "Settings/Inputs" tab, from where the user can change it.
```

### Arguments
```
defval (const color) Determines the default value of the input variable proposed in
the script's "Settings/Inputs" tab, from where the user can change it.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.color function always should be assigned to a variable, see
examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to
provide configuration options to script users. This function adds a dropdown with
options based on the enum fields passed to its defval and options parameters.

### See also
- input.bool
- input.int
- input.float
- input.string
- input.text_area
- input.symbol
- input.timeframe
- input.session
- input.source
- input.time
- input

## input.enum()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a dropdown with options based on the enum fields passed to its defval and options parameters.

### Syntax
```
defval (const enum) Determines the default value of the input, which users can
change in the script's "Settings/Inputs" tab. When the options parameter has a
specified tuple of enum fields, the tuple must include the defval.
```

### Arguments
```
defval (const enum) Determines the default value of the input, which users can
change in the script's "Settings/Inputs" tab. When the options parameter has a
specified tuple of enum fields, the tuple must include the defval.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
All fields included in the defval and options arguments must belong to the same enum.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for a float input to the script's inputs.

### See also
- input.text_area
- input.bool
- input.int
- input.float
- input.symbol

- input.timeframe
- input.session
- input.source
- input.color
- input.time
- input

## input.float()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for a float input to the script's inputs.

### Syntax & Overloads
defval (const int/float) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where script users can change it. When a list of values is used with the options parameter, the value must be one of them.

### Arguments
```
defval (const int/float) Determines the default value of the input variable
proposed in the script's "Settings/Inputs" tab, from where script users can change
it. When a list of values is used with the options parameter, the value must be one
of them.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.float function always should be assigned to a variable, see examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for an

integer input to the script's inputs.

### See also
- input.bool
- input.int
- input.string
- input.text_area
- input.symbol
- input.timeframe
- input.session
- input.source
- input.color
- input.time
- input

## input.int()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for an integer input to the script's inputs.

### Syntax & Overloads
defval (const int) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where script users can change it. When a list of values is used with the options parameter, the value must be one of them.

### Arguments
```
defval (const int) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where script users can change it. When a list of values is used with the options parameter, the value must be one of them.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks

Result of input.int function always should be assigned to a variable, see examples above.

### See also
Adds a price input to the script's "Settings/Inputs" tab. Using confirm = true activates the interactive input mode where a price is selected by clicking on the chart.

### See also
- input.bool
- input.float
- input.string
- input.text_area
- input.symbol
- input.timeframe
- input.session
- input.source
- input.color
- input.time
- input

## input.price()
Adds a price input to the script's "Settings/Inputs" tab. Using confirm = true activates the interactive input mode where a price is selected by clicking on the chart.

### Syntax
```
defval (const int/float) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it.
```

### Arguments
```
defval (const int/float) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
When using interactive mode, a time input can be combined with a price input if both function calls use the same argument for their inline parameter.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds two dropdowns that allow the user to specify the beginning and the end of a session using the session selector and returns the result as a string.

### See also
- input.bool
- input.int
- input.float
- input.string
- input.text_area
- input.symbol
- input.resolution
- input.session
- input.source
- input.color
- input

## input.session()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds two dropdowns that allow the user to specify the beginning and the end of a session using the session selector and returns the result as a string.

### Syntax
```
defval (const string) Determines the default value of the input variable proposed
in the script's "Settings/Inputs" tab, from where the user can change it. When a
list of values is used with the options parameter, the value must be one of them.
```

### Arguments

```
defval (const string) Determines the default value of the input variable proposed
in the script's "Settings/Inputs" tab, from where the user can change it. When a
list of values is used with the options parameter, the value must be one of them.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.session function always should be assigned to a variable, see examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a dropdown that allows the user to select a source for the calculation, e.g. close, hl2, etc. The user can also select an output from another indicator on their chart as the source.

### See also
- input.bool
- input.int
- input.float
- input.string
- input.text_area
- input.symbol
- input.timeframe
- input.source
- input.color
- input.time
- input

## input.source()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a dropdown that allows the user to select a source for the calculation, e.g. close, hl2, etc. The user can also select an output from another indicator on their chart as the source.

### Syntax
```
defval (open/high/low/close/hl2/hlc3/ohlc4/hlcc4) Determines the default value of
the input variable proposed in the script's "Settings/Inputs" tab, from where the
user can change it.
```

### Arguments
```
defval (open/high/low/close/hl2/hlc3/ohlc4/hlcc4) Determines the default value of
the input variable proposed in the script's "Settings/Inputs" tab, from where the
user can change it.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.source function always should be assigned to a variable, see examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for a string input to the script's inputs.

### See also
- input.bool
- input.int
- input.float
- input.string
- input.text_area
- input.symbol
- input.timeframe
- input.session
- input.color

- input.time
- input

## input.string()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for a string input to the script's inputs.

### Syntax
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it. When a list of values is used with the options parameter, the value must be one of them.
```

### Arguments
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it. When a list of values is used with the options parameter, the value must be one of them.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.string function always should be assigned to a variable, see examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field that allows the user to select a specific symbol using the symbol search and returns that symbol, paired with its exchange prefix, as a string.

### See also
- input.text_area
- input.bool
- input.int
- input.float
- input.symbol
- input.timeframe
- input.session
- input.source
- input.color
- input.time
- input

## input.symbol()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field that allows the user to select a specific symbol using the symbol search and returns that symbol, paired with its exchange prefix, as a string.

### Syntax
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it.
```

### Arguments
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.symbol function always should be assigned to a variable, see examples above.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for a multiline text input.

### See also
- input.bool
- input.int
- input.float
- input.string
- input.text_area
- input.timeframe
- input.session
- input.source
- input.color
- input.time
- input

## input.text_area()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a field for a multiline text input.

### Syntax
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it.
```

### Arguments
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it.
```

### Example
Value of input variable.

### Returns
```
```

Value of input variable.
```

### Remarks
Result of input.text_area function always should be assigned to a variable, see examples above.

### See also
Adds a time input to the script's "Settings/Inputs" tab. This function adds two input widgets on the same line: one for the date and one for the time. The function returns a date/time value in UNIX format. Using confirm = true activates the interactive input mode where a point in time is selected by clicking on the chart.

### See also
- input.string
- input.bool
- input.int
- input.float
- input.symbol
- input.timeframe
- input.session
- input.source
- input.color
- input.time
- input

## input.time()
Adds a time input to the script's "Settings/Inputs" tab. This function adds two input widgets on the same line: one for the date and one for the time. The function returns a date/time value in UNIX format. Using confirm = true activates the interactive input mode where a point in time is selected by clicking on the chart.

### Syntax
```
defval (const int) Determines the default value of the input variable proposed in
the script's "Settings/Inputs" tab, from where the user can change it. The value
can be a timestamp function, but only if it uses a date argument in const string
format.
```

### Arguments
```

```
defval (const int) Determines the default value of the input variable proposed in
the script's "Settings/Inputs" tab, from where the user can change it. The value
can be a timestamp function, but only if it uses a date argument in const string
format.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
When using interactive mode, a price input can be combined with a time input if both function calls use the same argument for their inline parameter.

### See also
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a dropdown that allows the user to select a specific timeframe via the timeframe selector and returns it as a string. The selector includes the custom timeframes a user may have added using the chart's Timeframe dropdown.

### See also
- input.bool
- input.int
- input.float
- input.string
- input.text_area
- input.symbol
- input.timeframe
- input.session
- input.source
- input.color
- input

## input.timeframe()
Adds an input to the Inputs tab of your script's Settings, which allows you to provide configuration options to script users. This function adds a dropdown that allows the user to select a specific timeframe via the timeframe selector and

returns it as a string. The selector includes the custom timeframes a user may have added using the chart's Timeframe dropdown.

### Syntax
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it. When a list of values is used with the options parameter, the value must be one of them.
```

### Arguments
```
defval (const string) Determines the default value of the input variable proposed in the script's "Settings/Inputs" tab, from where the user can change it. When a list of values is used with the options parameter, the value must be one of them.
```

### Example
Value of input variable.

### Returns
```
Value of input variable.
```

### Remarks
Result of input.timeframe function always should be assigned to a variable, see examples above.

### See also
Casts na or truncates float value to int

### See also
- input.bool
- input.int
- input.float
- input.string
- input.text_area
- input.symbol
- input.session
- input.source
- input.color

- input.time
- input

## int()
Casts na or truncates float value to int

### Syntax & Overloads
x (const int/float) The value to convert to the specified type, usually na.

### Arguments
```
x (const int/float) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to int.
```

### See also
Casts na to label

### See also
- float
- bool
- color
- string
- line
- label

## label()
Casts na to label

### Syntax
```
x (series label) The value to convert to the specified type, usually na.
```

### Arguments
```
x (series label) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to label.
```

### See also
Clones the label object.

### See also
- float
- int
- bool
- color
- string
- line

## label.copy()
Clones the label object.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### Example
New label ID object which may be passed to label.setXXX and label.getXXX functions.

### Returns
```
New label ID object which may be passed to label.setXXX and label.getXXX functions.
```

~~~
```


### See also
Deletes the specified label object. If it has already been deleted, does nothing.


### See also
- label.new
- label.delete



## label.delete()
Deletes the specified label object. If it has already been deleted, does nothing.


### Syntax
```
id (series label) Label object to delete.
```


### Arguments
```
id (series label) Label object to delete.
```


### See also
Returns the text of this label object.


### See also
- label.new



## label.get_text()
Returns the text of this label object.


### Syntax
```
id (series label) Label object.
```


### Arguments
~~~

```
id (series label) Label object.
```

### Example
String object containing the text of this label.

### Returns
```
String object containing the text of this label.
```

### See also
Returns UNIX time or bar index (depending on the last xloc value set) of this label's position.

### See also
- label.new

## label.get_x()
Returns UNIX time or bar index (depending on the last xloc value set) of this label's position.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### Example
UNIX timestamp (in milliseconds) or bar index.

### Returns
```
UNIX timestamp (in milliseconds) or bar index.
```

```

### See also
Returns price of this label's position.

### See also
- label.new

## label.get_y()
Returns price of this label's position.

### Syntax
```
id (series label) Label object.
```

Arguments
```
id (series label) Label object.
```

Returns
```
Floating point value representing price.
```

See also
Creates new label object.

See also
- label.new

label.new()
Creates new label object.

Syntax & Overloads
point (chart.point) A chart.point object that specifies the label's location.
```

### Arguments
```
point (chart.point) A chart.point object that specifies the label's location.
```

### Example
Label ID object which may be passed to label.setXXX and label.getXXX functions.

### Returns
```
Label ID object which may be passed to label.setXXX and label.getXXX functions.
```

### See also
Sets label border and arrow color.

### See also
- label.delete
- label.set_x
- label.set_y
- label.set_xy
- label.set_xloc
- label.set_yloc
- label.set_color
- label.set_textcolor
- label.set_style
- label.set_size
- label.set_textalign
- label.set_tooltip
- label.set_text
- label.set_text_formatting

## label.set_color()
Sets label border and arrow color.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets the location of the id label to point.

### See also
- label.new

## label.set_point()
Sets the location of the id label to point.

### Syntax
```
id (series label) A label object.
```

### Arguments
```
id (series label) A label object.
```

## label.set_size()
Sets arrow and text size of the specified label object.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets label style.

### See also
- size.auto
- size.tiny
- size.small
- size.normal
- size.large
- size.huge
- label.new

## label.set_style()
Sets label style.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets label text

### See also
- label.new

## label.set_text()
Sets label text

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
The function sets the font family of the text inside the label.

### See also
- label.new
- label.set_text_formatting

## label.set_text_font_family()
The function sets the font family of the text inside the label.

### Syntax
```
id (series label) A label object.
```

### Arguments
```
id (series label) A label object.
```

### Example
Sets the formatting attributes the drawing applies to displayed text.

### See also
Sets the formatting attributes the drawing applies to displayed text.

### See also
- label.new
- font.family_default
- font.family_monospace

## label.set_text_formatting()
Sets the formatting attributes the drawing applies to displayed text.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets the alignment for the label text.

### See also
- label.new
- label.set_text

## label.set_textalign()
Sets the alignment for the label text.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets color of the label text.

### See also
- text.align_left
- text.align_center

- text.align_right
- label.new

## label.set_textcolor()
Sets color of the label text.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets the tooltip text.

### See also
- label.new

## label.set_tooltip()
Sets the tooltip text.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets bar index or bar time (depending on the xloc) of the label position.

### See also
- label.new

## label.set_x()
Sets bar index or bar time (depending on the xloc) of the label position.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets x-location and new bar index/time value.

### See also
- label.new

## label.set_xloc()
Sets x-location and new bar index/time value.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets bar index/time and price of the label position.

### See also
- xloc.bar_index
- xloc.bar_time
- label.new

## label.set_xy()
Sets bar index/time and price of the label position.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets price of the label position

### See also
- label.new

## label.set_y()
Sets price of the label position

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Sets new y-location calculation algorithm.

### See also
- label.new

## label.set_yloc()
Sets new y-location calculation algorithm.

### Syntax
```
id (series label) Label object.
```

### Arguments
```
id (series label) Label object.
```

### See also
Declaration statement identifying a script as a library.

### See also
- yloc.price
- yloc.abovebar
- yloc.belowbar
- label.new

## library()
Declaration statement identifying a script as a library.

### Syntax
```
title (const string) The title of the library and its identifier. It cannot contain
spaces, special characters or begin with a digit. It is used as the publication's
default title, and to uniquely identify the library in the import statement, when
another script uses it. It is also used as the script's name on the chart.
```

### Arguments
```
title (const string) The title of the library and its identifier. It cannot contain spaces, special characters or begin with a digit. It is used as the publication's default title, and to uniquely identify the library in the import statement, when another script uses it. It is also used as the script's name on the chart.
```

### Example
Casts na to line

### See also
Casts na to line

### See also
- indicator
- strategy

## line()
Casts na to line

### Syntax
```
x (series line) The value to convert to the specified type, usually na.
```

### Arguments
```
x (series line) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to line.
```

### See also
Clones the line object.

### See also
- float
- int
- bool
- color
- string
- label

## line.copy()
Clones the line object.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### Example
New line ID object which may be passed to line.setXXX and line.getXXX functions.

### Returns
```
New line ID object which may be passed to line.setXXX and line.getXXX functions.
```

### See also
Deletes the specified line object. If it has already been deleted, does nothing.

### See also
- line.new
- line.delete

## line.delete()

Deletes the specified line object. If it has already been deleted, does nothing.

### Syntax
```
id (series line) Line object to delete.
```

### Arguments
```
id (series line) Line object to delete.
```

### See also
Returns the price level of a line at a given bar index.

### See also
- line.new

## line.get_price()
Returns the price level of a line at a given bar index.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### Example
Price value of line 'id' at bar index 'x'.

### Returns
```
Price value of line 'id' at bar index 'x'.
```

### Remarks
The line is considered to have been created using 'extend=extend.both'.

### See also
Returns UNIX time or bar index (depending on the last xloc value set) of the first point of the line.

### See also
- line.new

## line.get_x1()
Returns UNIX time or bar index (depending on the last xloc value set) of the first point of the line.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### Example
UNIX timestamp (in milliseconds) or bar index.

### Returns
```
UNIX timestamp (in milliseconds) or bar index.
```

### See also
Returns UNIX time or bar index (depending on the last xloc value set) of the second point of the line.

### See also
- line.new

## line.get_x2()
Returns UNIX time or bar index (depending on the last xloc value set) of the second point of the line.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### Returns
```
UNIX timestamp (in milliseconds) or bar index.
```

### See also
Returns price of the first point of the line.

### See also
- line.new

## line.get_y1()
Returns price of the first point of the line.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### Returns
```
Price value.
```

### See also
Returns price of the second point of the line.

### See also
- line.new

## line.get_y2()
Returns price of the second point of the line.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### Returns
```
Price value.
```

### See also
Creates new line object.

### See also
- line.new

## line.new()

Creates new line object.

### Syntax & Overloads
first_point (chart.point) A chart.point object that specifies the line's starting coordinate.

### Arguments
```
first_point (chart.point) A chart.point object that specifies the line's starting coordinate.
```

### Example
Line ID object which may be passed to line.setXXX and line.getXXX functions.

### Returns
```
Line ID object which may be passed to line.setXXX and line.getXXX functions.
```

### See also
Sets the line color

### See also
- line.delete
- line.set_x1
- line.set_y1
- line.set_xy1
- line.set_x2
- line.set_y2
- line.set_xy2
- line.set_xloc
- line.set_color
- line.set_extend
- line.set_style
- line.set_width

## line.set_color()
Sets the line color

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets extending type of this line object. If extend=extend.none, draws segment starting at point (x1, y1) and ending at point (x2, y2). If extend is equal to extend.right or extend.left, draws a ray starting at point (x1, y1) or (x2, y2), respectively. If extend=extend.both, draws a straight line that goes through these points.

### See also
- line.new

## line.set_extend()
Sets extending type of this line object. If extend=extend.none, draws segment starting at point (x1, y1) and ending at point (x2, y2). If extend is equal to extend.right or extend.left, draws a ray starting at point (x1, y1) or (x2, y2), respectively. If extend=extend.both, draws a straight line that goes through these points.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets the first point of the id line to point.

### See also
- extend.none
- extend.right
- extend.left
- extend.both
- line.new

## line.set_first_point()
Sets the first point of the id line to point.

### Syntax
```
id (series line) A line object.
```

### Arguments
```
id (series line) A line object.
```

## line.set_second_point()
Sets the second point of the id line to point.

### Syntax
```
id (series line) A line object.
```

### Arguments
```
id (series line) A line object.
```

## line.set_style()
Sets the line style

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets the line width.

### See also
- line.style_solid
- line.style_dotted
- line.style_dashed
- line.style_arrow_left
- line.style_arrow_right
- line.style_arrow_both
- line.new

## line.set_width()
Sets the line width.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets bar index or bar time (depending on the xloc) of the first point.

### See also
- line.new

## line.set_x1()
Sets bar index or bar time (depending on the xloc) of the first point.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets bar index or bar time (depending on the xloc) of the second point.

### See also
- line.new

## line.set_x2()
Sets bar index or bar time (depending on the xloc) of the second point.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets x-location and new bar index/time values.

### See also

- line.new

## line.set_xloc()
Sets x-location and new bar index/time values.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets bar index/time and price of the first point.

### See also
- xloc.bar_index
- xloc.bar_time
- line.new

## line.set_xy1()
Sets bar index/time and price of the first point.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets bar index/time and price of the second point

### See also
- line.new

## line.set_xy2()
Sets bar index/time and price of the second point

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Sets price of the first point

### See also
- line.new

## line.set_y1()
Sets price of the first point

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also

Sets price of the second point.

### See also
- line.new

## line.set_y2()
Sets price of the second point.

### Syntax
```
id (series line) Line object.
```

### Arguments
```
id (series line) Line object.
```

### See also
Casts na to linefill.

### See also
- line.new

## linefill()
Casts na to linefill.

### Syntax
```
x (series linefill) The value to convert to the specified type, usually na.
```

### Arguments
```
x (series linefill) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to linefill.
```

### See also
Deletes the specified linefill object. If it has already been deleted, does nothing.

### See also
- float
- int
- bool
- color
- string
- line
- label

## linefill.delete()
Deletes the specified linefill object. If it has already been deleted, does nothing.

### Syntax
```
id (series linefill) A linefill object.
```

### Arguments
```
id (series linefill) A linefill object.
```

## linefill.get_line1()
Returns the ID of the first line used in the id linefill.

### Syntax
```
id (series linefill) A linefill object.
```

### Arguments
```
id (series linefill) A linefill object.
```

## linefill.get_line2()
Returns the ID of the second line used in the id linefill.

### Syntax
```
id (series linefill) A linefill object.
```

### Arguments
```
id (series linefill) A linefill object.
```

## linefill.new()
Creates a new linefill object and displays it on the chart, filling the space between line1 and line2 with the color specified in color.

### Syntax
```
line1 (series line) First line object.
```

### Arguments
```
line1 (series line) First line object.
```

### Returns
```
The ID of a linefill object that can be passed to other linefill.*() functions.
```

### Remarks
If any line of the two is deleted, the linefill object is also deleted. If the lines are moved (e.g. via line.set_xy functions), the linefill object is also moved.

## linefill.set_color()
The function sets the color of the linefill object passed to it.

### Syntax
```
id (series linefill) A linefill object.
```

### Arguments
```
id (series linefill) A linefill object.
```

## log.error()
Converts the formatting string and value(s) into a formatted string, and sends the result to the "Pine logs" menu tagged with the "error" debug level.

### Syntax & Overloads
message (series string) Log message.

### Arguments
```
message (series string) Log message.
```

### Example
The formatted string.

### Returns
```
```

The formatted string.
```

Remarks
Any curly braces within an unquoted pattern must be balanced. For example, "ab {0} de" and "ab '}' de" are valid patterns, but "ab {0'}' de", "ab } de" and "''{''" are not.

log.info()
Converts the formatting string and value(s) into a formatted string, and sends the result to the "Pine logs" menu tagged with the "info" debug level.

Syntax & Overloads
message (series string) Log message.

Arguments
```
message (series string) Log message.
```

Example
The formatted string.

Returns
```
The formatted string.
```

Remarks
Any curly braces within an unquoted pattern must be balanced. For example, "ab {0} de" and "ab '}' de" are valid patterns, but "ab {0'}' de", "ab } de" and "''{''" are not.

log.warning()
Converts the formatting string and value(s) into a formatted string, and sends the result to the "Pine logs" menu tagged with the "warning" debug level.
```

### Syntax & Overloads
message (series string) Log message.

### Arguments
```
message (series string) Log message.
```

### Example
The formatted string.

### Returns
```
The formatted string.
```

### Remarks
Any curly braces within an unquoted pattern must be balanced. For example, "ab {0} de" and "ab '}' de" are valid patterns, but "ab {0'}' de", "ab } de" and "''{''" are not.

## map.clear()
Clears the map, removing all key-value pairs from it.

### Syntax
```
id (any map type) A map object.
```

### Arguments
```
id (any map type) A map object.
```

### Example
Returns true if the key was found in the id map, false otherwise.

### See also
Returns true if the key was found in the id map, false otherwise.

### See also
- map.new<type,type>
- map.put_all
- map.keys
- map.values
- map.remove

## map.contains()
Returns true if the key was found in the id map, false otherwise.

### Syntax
```
id (any map type) A map object.
```

### Arguments
```
id (any map type) A map object.
```

### Example
Creates a copy of an existing map.

### See also
Creates a copy of an existing map.

### See also
- map.new<type,type>
- map.put
- map.keys
- map.values
- map.size

## map.copy()
Creates a copy of an existing map.

### Syntax
```
id (any map type) A map object to copy.
```

### Arguments
```
id (any map type) A map object to copy.
```

### Example
A copy of the id map.

### Returns
```
A copy of the id map.
```

### See also
Returns the value associated with the specified key in the id map.

### See also
- map.new<type,type>
- map.put
- map.keys
- map.values
- map.get
- map.size

## map.get()
Returns the value associated with the specified key in the id map.

### Syntax
```
id (any map type) A map object.
```

### Arguments
```
```

id (any map type) A map object.
```

Example
Returns an array of all the keys in the id map. The resulting array is a copy and any changes to it are not reflected in the original map.

See also
Returns an array of all the keys in the id map. The resulting array is a copy and any changes to it are not reflected in the original map.

See also
- map.new<type,type>
- map.put
- map.keys
- map.values
- map.contains

map.keys()
Returns an array of all the keys in the id map. The resulting array is a copy and any changes to it are not reflected in the original map.

Syntax
```
id (any map type) A map object.
```

Arguments
```
id (any map type) A map object.
```

Example
Maps maintain insertion order. The elements within the array returned by this function will also be in the insertion order.

Remarks
Maps maintain insertion order. The elements within the array returned by this function will also be in the insertion order.
```

### See also
Creates a new map object: a collection that consists of key-value pairs, where all keys are of the keyType, and all values are of the valueType.

### See also
- map.new<type,type>
- map.put
- map.get
- map.values
- map.size

## map.new<type,type>()
Creates a new map object: a collection that consists of key-value pairs, where all keys are of the keyType, and all values are of the valueType.

### Syntax
```
The ID of a map object which may be used in other map.*() functions.
```

### Example
The ID of a map object which may be used in other map.*() functions.

### Returns
```
The ID of a map object which may be used in other map.*() functions.
```

### Remarks
Each key is unique and can only appear once. When adding a new value with a key that the map already contains, that value replaces the old value associated with the key.

### See also
Puts a new key-value pair into the id map.

### See also
- map.put
- map.keys

- map.values
- map.get
- array.new<type>

## map.put()
Puts a new key-value pair into the id map.

### Syntax
```
id (any map type) A map object.
```

### Arguments
```
id (any map type) A map object.
```

### Example
The previous value associated with key if the key was already present in the map, or na if the key is new.

### Returns
```
The previous value associated with key if the key was already present in the map,
or na if the key is new.
```

### Remarks
Maps maintain insertion order. Note that the order does not change when inserting a pair with a key that's already in the map. The new pair replaces the existing pair with the key in such cases.

### See also
Puts all key-value pairs from the id2 map into the id map.

### See also
- map.new<type,type>
- map.put_all
- map.keys
- map.values

- map.remove

## map.put_all()
Puts all key-value pairs from the id2 map into the id map.

### Syntax
```
id (any map type) A map object to append to.
```

### Arguments
```
id (any map type) A map object to append to.
```

### Example
Removes a key-value pair from the id map.

### See also
Removes a key-value pair from the id map.

### See also
- map.new<type,type>
- map.put
- map.keys
- map.values
- map.remove

## map.remove()
Removes a key-value pair from the id map.

### Syntax
```
id (any map type) A map object.
```

### Arguments
```
```

id (any map type) A map object.
```

Example
The previous value associated with key if the key was present in the map, or na if there was no such key.

Returns
```
The previous value associated with key if the key was present in the map, or na if there was no such key.
```

See also
Returns the number of key-value pairs in the id map.

See also
- map.new<type,type>
- map.put
- map.keys
- map.values
- map.clear

map.size()
Returns the number of key-value pairs in the id map.

Syntax
```
id (any map type) A map object.
```

Arguments
```
id (any map type) A map object.
```

Example
Returns an array of all the values in the id map. The resulting array is a copy and any changes to it are not reflected in the original map.
```

### See also
Returns an array of all the values in the id map. The resulting array is a copy and any changes to it are not reflected in the original map.

### See also
- map.new<type,type>
- map.put
- map.keys
- map.values
- map.get

## map.values()
Returns an array of all the values in the id map. The resulting array is a copy and any changes to it are not reflected in the original map.

### Syntax
```
id (any map type) A map object.
```

### Arguments
```
id (any map type) A map object.
```

### Example
Maps maintain insertion order. The elements within the array returned by this function will also be in the insertion order.

### Remarks
Maps maintain insertion order. The elements within the array returned by this function will also be in the insertion order.

### See also
Absolute value of number is number if number >= 0, or -number otherwise.

### See also
- map.new<type,type>
- map.put

- map.get
- map.keys
- map.size

## math.abs()
Absolute value of number is number if number >= 0, or -number otherwise.

### Syntax & Overloads
number (const int) The number to use in the calculation.

### Arguments
```
number (const int) The number to use in the calculation.
```

### Returns
```
The absolute value of number.
```

## math.acos()
The acos function returns the arccosine (in radians) of number such that cos(acos(y)) = y for y in range [-1, 1].

### Syntax & Overloads
angle (const int/float) The value, in radians, to use in the calculation.

### Arguments
```
angle (const int/float) The value, in radians, to use in the calculation.
```

### Returns
```
The arc cosine of a value; the returned angle is in the range [0, Pi], or na if y
is outside of range [-1, 1].
```

## math.asin()
The asin function returns the arcsine (in radians) of number such that sin(asin(y)) = y for y in range [-1, 1].

### Syntax & Overloads
angle (const int/float) The value, in radians, to use in the calculation.

### Arguments
```
angle (const int/float) The value, in radians, to use in the calculation.
```

### Returns
```
The arcsine of a value; the returned angle is in the range [-Pi/2, Pi/2], or na if
y is outside of range [-1, 1].
```

## math.atan()
The atan function returns the arctangent (in radians) of number such that tan(atan(y)) = y for any y.

### Syntax & Overloads
angle (const int/float) The value, in radians, to use in the calculation.

### Arguments
```
angle (const int/float) The value, in radians, to use in the calculation.
```

### Returns
```
The arc tangent of a value; the returned angle is in the range [-Pi/2, Pi/2].
```

## math.avg()
Calculates average of all given series (elementwise).

### Syntax & Overloads
number0, number1, ... (simple int/float) A sequence of numbers to use in the calculation.

### Arguments
```
number0, number1, ... (simple int/float) A sequence of numbers to use in the calculation.
```

### Returns
```
Average.
```

### See also
Rounds the specified number up to the smallest whole number ("int" value) that is greater than or equal to it.

### See also
- math.sum
- ta.cum
- ta.sma

## math.ceil()
Rounds the specified number up to the smallest whole number ("int" value) that is greater than or equal to it.

### Syntax & Overloads
number (const int/float) The number to round.

### Arguments
```
number (const int/float) The number to round.
```

### Returns
```
The smallest "int" value that is greater than or equal to the number.
```

### See also
The cos function returns the trigonometric cosine of an angle.

### See also
- math.floor
- math.round

## math.cos()
The cos function returns the trigonometric cosine of an angle.

### Syntax & Overloads
angle (const int/float) Angle, in radians.

### Arguments
```
angle (const int/float) Angle, in radians.
```

### Returns
```
The trigonometric cosine of an angle.
```

## math.exp()
The exp function of number is e raised to the power of number, where e is Euler's number.

### Syntax & Overloads
number (const int/float) The number to use in the calculation.

### Arguments

```
number (const int/float) The number to use in the calculation.
```

### Returns
```
A value representing e raised to the power of number.
```

### See also
Rounds the specified number down to the largest whole number ("int" value) that is less than or equal to it.

### See also
- math.pow

## math.floor()
Rounds the specified number down to the largest whole number ("int" value) that is less than or equal to it.

### Syntax & Overloads
number (const int/float) The number to round.

### Arguments
```
number (const int/float) The number to round.
```

### Returns
```
The largest "int" value that is less than or equal to the number.
```

### See also
Natural logarithm of any number > 0 is the unique y such that e^y = number.

### See also
- math.ceil
- math.round

## math.log()
Natural logarithm of any number > 0 is the unique y such that e^y = number.

### Syntax & Overloads
number (const int/float) The number to use in the calculation.

### Arguments
```
number (const int/float) The number to use in the calculation.
```

### Returns
```
The natural logarithm of number.
```

### See also
The common (or base 10) logarithm of number is the power to which 10 must be raised to obtain the number. 10^y = number.

### See also
- math.log10

## math.log10()
The common (or base 10) logarithm of number is the power to which 10 must be raised to obtain the number. 10^y = number.

### Syntax & Overloads
number (const int/float) The number to use in the calculation.

### Arguments
```
number (const int/float) The number to use in the calculation.
```

### Returns

```
The base 10 logarithm of number.
```

### See also
Returns the greatest of multiple values.

### See also
- math.log

## math.max()
Returns the greatest of multiple values.

### Syntax & Overloads
number0, number1, ... (const int) A sequence of numbers to use in the calculation.

### Arguments
```
number0, number1, ... (const int) A sequence of numbers to use in the calculation.
```

### Example
The greatest of multiple given values.

### Returns
```
The greatest of multiple given values.
```

### See also
Returns the smallest of multiple values.

### See also
- math.min

## math.min()
Returns the smallest of multiple values.

### Syntax & Overloads
number0, number1, ... (const int) A sequence of numbers to use in the calculation.

### Arguments
```
number0, number1, ... (const int) A sequence of numbers to use in the calculation.
```

### Example
The smallest of multiple given values.

### Returns
```
The smallest of multiple given values.
```

### See also
Mathematical power function.

### See also
- math.max

## math.pow()
Mathematical power function.

### Syntax & Overloads
base (const int/float) Specify the base to use.

### Arguments
```
base (const int/float) Specify the base to use.
```

### Example
base raised to the power of exponent. If base is a series, it is calculated
elementwise.

### Returns
```
base raised to the power of exponent. If base is a series, it is calculated elementwise.
```

### See also
Returns a pseudo-random value. The function will generate a different sequence of values for each script execution. Using the same value for the optional seed argument will produce a repeatable sequence.

### See also
- math.sqrt
- math.exp

## math.random()
Returns a pseudo-random value. The function will generate a different sequence of values for each script execution. Using the same value for the optional seed argument will produce a repeatable sequence.

### Syntax
```
min (series int/float) The lower bound of the range of random values. The value is not included in the range. The default is 0.
```

### Arguments
```
min (series int/float) The lower bound of the range of random values. The value is not included in the range. The default is 0.
```

### Returns
```
A random value.
```

## math.round()

Returns the value of number rounded to the nearest integer, with ties rounding up. If the precision parameter is used, returns a float value rounded to that amount of decimal places.

### Syntax & Overloads
number (const int/float) The value to be rounded.

### Arguments
```
number (const int/float) The value to be rounded.
```

### Returns
```
The value of number rounded to the nearest integer, or according to precision.
```

### Remarks
Note that for 'na' values function returns 'na'.

### See also
Returns the value rounded to the symbol's mintick, i.e. the nearest value that can be divided by syminfo.mintick, without the remainder, with ties rounding up.

### See also
- math.ceil
- math.floor

## math.round_to_mintick()
Returns the value rounded to the symbol's mintick, i.e. the nearest value that can be divided by syminfo.mintick, without the remainder, with ties rounding up.

### Syntax & Overloads
number (simple int/float) The value to be rounded.

### Arguments
```
number (simple int/float) The value to be rounded.
```

### Returns
```
The number rounded to tick precision.
```

### Remarks
Note that for 'na' values function returns 'na'.

### See also
Sign (signum) of number is zero if number is zero, 1.0 if number is greater than zero, -1.0 if number is less than zero.

### See also
- math.ceil
- math.floor

## math.sign()
Sign (signum) of number is zero if number is zero, 1.0 if number is greater than zero, -1.0 if number is less than zero.

### Syntax & Overloads
number (const int/float) The number to use in the calculation.

### Arguments
```
number (const int/float) The number to use in the calculation.
```

### Returns
```
The sign of the argument.
```

## math.sin()
The sin function returns the trigonometric sine of an angle.

### Syntax & Overloads
angle (const int/float) Angle, in radians.

### Arguments
```
angle (const int/float) Angle, in radians.
```

### Returns
```
The trigonometric sine of an angle.
```

## math.sqrt()
Square root of any number >= 0 is the unique y >= 0 such that y^2 = number.

### Syntax & Overloads
number (const int/float) The number to use in the calculation.

### Arguments
```
number (const int/float) The number to use in the calculation.
```

### Returns
```
The square root of number.
```

### See also
The sum function returns the sliding sum of last y values of x.

### See also
- math.pow

## math.sum()

The sum function returns the sliding sum of last y values of x.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
Sum of source for length bars back.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
The tan function returns the trigonometric tangent of an angle.

### See also
- ta.cum
- for

## math.tan()
The tan function returns the trigonometric tangent of an angle.

### Syntax & Overloads
angle (const int/float) Angle, in radians.

### Arguments
```
angle (const int/float) Angle, in radians.
```

### Returns
```
The trigonometric tangent of an angle.
```

## math.todegrees()
Returns an approximately equivalent angle in degrees from an angle measured in radians.

### Syntax
```
radians (series int/float) Angle in radians.
```

### Arguments
```
radians (series int/float) Angle in radians.
```

### Returns
```
The angle value in degrees.
```

## math.toradians()
Returns an approximately equivalent angle in radians from an angle measured in degrees.

### Syntax
```
degrees (series int/float) Angle in degrees.
```

### Arguments
```
degrees (series int/float) Angle in degrees.
```

### Returns
```
The angle value in radians.
```

## matrix.add_col()
The function adds a column at the column index of the id matrix. The column can consist of na values, or an array can be used to provide values.

### Syntax & Overloads
id (any matrix type) A matrix object.

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
Adding an array as a column to the matrix

### Example
Rather than add columns to an empty matrix, it is far more efficient to declare a matrix with explicit dimensions and fill it with values. Adding a column is also much slower than adding a row with the matrix.add_row function.

### Remarks
Rather than add columns to an empty matrix, it is far more efficient to declare a matrix with explicit dimensions and fill it with values. Adding a column is also much slower than adding a row with the matrix.add_row function.

### See also
The function adds a row at the row index of the id matrix. The row can consist of na values, or an array can be used to provide values.

### See also
- matrix.new<type>
- matrix.get

- matrix.set
- matrix.columns
- matrix.rows
- matrix.add_row

## matrix.add_row()
The function adds a row at the row index of the id matrix. The row can consist of na values, or an array can be used to provide values.

### Syntax & Overloads
id (any matrix type) A matrix object.

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
Adding an array as a row to the matrix

### Example
Indexing of rows and columns starts at zero. Rather than add rows to an empty matrix, it is far more efficient to declare a matrix with explicit dimensions and fill it with values.

### Remarks
Indexing of rows and columns starts at zero. Rather than add rows to an empty matrix, it is far more efficient to declare a matrix with explicit dimensions and fill it with values.

### See also
The function calculates the average of all elements in the matrix.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows
- matrix.add_col

## matrix.avg()
The function calculates the average of all elements in the matrix.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
The average value from the id matrix.

### Returns
```
The average value from the id matrix.
```

### See also
The function creates a one-dimensional array from the elements of a matrix column.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.col()
The function creates a one-dimensional array from the elements of a matrix column.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
An array ID containing the column values of the id matrix.

### Returns
```
An array ID containing the column values of the id matrix.
```

### Remarks
Indexing of rows starts at 0.

### See also
The function returns the number of columns in the matrix.

### See also
- matrix.new<type>
- matrix.get
- array.get
- matrix.col
- matrix.columns

## matrix.columns()
The function returns the number of columns in the matrix.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
The number of columns in the matrix id.

### Returns
```
The number of columns in the matrix id.
```

### See also
The function appends the m2 matrix to the m1 matrix.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.col
- matrix.row
- matrix.rows

## matrix.concat()
The function appends the m2 matrix to the m1 matrix.

### Syntax
```
id1 (any matrix type) Matrix object to concatenate into.
```

### Arguments
```
id1 (any matrix type) Matrix object to concatenate into.
```

### Example
Returns the id1 matrix concatenated with the id2 matrix.

### Returns
```
Returns the id1 matrix concatenated with the id2 matrix.
```

### Remarks
The number of columns in both matrices must be identical.

### See also
The function creates a new matrix which is a copy of the original.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.copy()
The function creates a new matrix which is a copy of the original.

### Syntax
```
id (any matrix type) A matrix object to copy.
```

### Arguments
```
id (any matrix type) A matrix object to copy.
```

### Example
A new matrix object of the copied id matrix.

### Returns
```
A new matrix object of the copied id matrix.
```

### See also
The function returns the determinant of a square matrix.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.det()
The function returns the determinant of a square matrix.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
The determinant value of the id matrix.

### Returns
```
The determinant value of the id matrix.
```

### Remarks
Function calculation based on the LU decomposition algorithm.

### See also
The function returns a new matrix resulting from the subtraction between matrices id1 and id2, or of matrix id1 and an id2 scalar (a numerical value).

### See also
- matrix.new<type>
- matrix.set
- matrix.is_square

## matrix.diff()
The function returns a new matrix resulting from the subtraction between matrices id1 and id2, or of matrix id1 and an id2 scalar (a numerical value).

### Syntax & Overloads
id1 (matrix<int>) Matrix to subtract from.

### Arguments
```
id1 (matrix<int>) Matrix to subtract from.
```

### Example
Difference between a matrix and a scalar value

### Example
A new matrix object containing the difference between id2 and id1.

### Returns
```
A new matrix object containing the difference between id2 and id1.
```

### See also
The function returns an array containing the eigenvalues of a square matrix.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.eigenvalues()
The function returns an array containing the eigenvalues of a square matrix.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
An array containing the eigenvalues of the id matrix.

### Returns
```
An array containing the eigenvalues of the id matrix.
```

### Remarks
The function is calculated using "The Implicit QL Algorithm".

### See also
Returns a matrix of eigenvectors, in which each column is an eigenvector of the id matrix.

### See also
- matrix.new<type>
- matrix.set
- matrix.eigenvectors

## matrix.eigenvectors()
Returns a matrix of eigenvectors, in which each column is an eigenvector of the id matrix.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example

A new matrix containing the eigenvectors of the id matrix.

### Returns
```
A new matrix containing the eigenvectors of the id matrix.
```

### Remarks
The function is calculated using "The Implicit QL Algorithm".

### See also
The function returns the total number of all matrix elements.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.eigenvalues

## matrix.elements_count()
The function returns the total number of all matrix elements.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### See also
The function fills a rectangular area of the id matrix defined by the indices from_column to to_column (not including it) and from_row to to_row(not including it) with the value.

### See also
- matrix.new<type>

- matrix.columns
- matrix.rows

## matrix.fill()
The function fills a rectangular area of the id matrix defined by the indices from_column to to_column (not including it) and from_row to to_row(not including it) with the value.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
The function returns the element with the specified index of the matrix.

### See also
The function returns the element with the specified index of the matrix.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.get()
The function returns the element with the specified index of the matrix.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
The value of the element at the row and column index of the id matrix.

### Returns
```
The value of the element at the row and column index of the id matrix.
```

### Remarks
Indexing of the rows and columns starts at zero.

### See also
The function returns the inverse of a square matrix.

### See also
- matrix.new<type>
- matrix.set
- matrix.columns
- matrix.rows

## matrix.inv()
The function returns the inverse of a square matrix.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
A new matrix, which is the inverse of the id matrix.

### Returns
```
A new matrix, which is the inverse of the id matrix.
```

### Remarks
The function is calculated using the LU decomposition algorithm.

### See also
The function determines if the matrix is anti-diagonal (all elements outside the secondary diagonal are zero).

### See also
- matrix.new<type>
- matrix.set
- matrix.pinv
- matrix.copy
- str.tostring

## matrix.is_antidiagonal()
The function determines if the matrix is anti-diagonal (all elements outside the secondary diagonal are zero).

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments
```
id (matrix<int/float>) Matrix object to test.
```

### Returns
```
Returns true if the id matrix is anti-diagonal, false otherwise.
```

### Remarks
Returns false with non-square matrices.

### See also
The function determines if a matrix is antisymmetric (its transpose equals its negative).

### See also
- matrix.new<type>
- matrix.set
- matrix.is_square
- matrix.is_identity
- matrix.is_diagonal

## matrix.is_antisymmetric()
The function determines if a matrix is antisymmetric (its transpose equals its negative).

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments
```
id (matrix<int/float>) Matrix object to test.
```

### Returns
```
Returns true, if the id matrix is antisymmetric, false otherwise.
```

### Remarks
Returns false with non-square matrices.

### See also
The function determines if the matrix is binary (when all elements of the matrix are 0 or 1).

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.is_square

## matrix.is_binary()
The function determines if the matrix is binary (when all elements of the matrix are 0 or 1).

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments
```
id (matrix<int/float>) Matrix object to test.
```

### Returns
```
Returns true if the id matrix is binary, false otherwise.
```

### See also
The function determines if the matrix is diagonal (all elements outside the main diagonal are zero).

### See also
- matrix.new<type>
- matrix.get
- matrix.set

## matrix.is_diagonal()
The function determines if the matrix is diagonal (all elements outside the main diagonal are zero).

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments
```
id (matrix<int/float>) Matrix object to test.
```

### Returns
```
Returns true if the id matrix is diagonal, false otherwise.
```

### Remarks
Returns false with non-square matrices.

### See also
The function determines if a matrix is an identity matrix (elements with ones on the main diagonal and zeros elsewhere).

### See also
- matrix.new<type>
- matrix.set
- matrix.is_square
- matrix.is_identity
- matrix.is_antidiagonal

## matrix.is_identity()
The function determines if a matrix is an identity matrix (elements with ones on the main diagonal and zeros elsewhere).

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments
```
```

id (matrix<int/float>) Matrix object to test.
```

Returns
```
Returns true if id is an identity matrix, false otherwise.
```

Remarks
Returns false with non-square matrices.

See also
The function determines if the matrix is square (it has the same number of rows and columns).

See also
- matrix.new<type>
- matrix.is_square
- matrix.is_diagonal

matrix.is_square()
The function determines if the matrix is square (it has the same number of rows and columns).

Syntax
```
id (any matrix type) Matrix object to test.
```

Arguments
```
id (any matrix type) Matrix object to test.
```

Returns
```
Returns true if the id matrix is square, false otherwise.
```
```

### See also
The function determines if the matrix is stochastic.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.is_stochastic()
The function determines if the matrix is stochastic.

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments
```
id (matrix<int/float>) Matrix object to test.
```

### Returns
```
Returns true if the id matrix is stochastic, false otherwise.
```

### See also
The function determines if a square matrix is symmetric (elements are symmetric with respect to the main diagonal).

### See also
- matrix.new<type>
- matrix.set

## matrix.is_symmetric()
The function determines if a square matrix is symmetric (elements are symmetric with respect to the main diagonal).

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments
```
id (matrix<int/float>) Matrix object to test.
```

### Returns
```
Returns true if the id matrix is symmetric, false otherwise.
```

### Remarks
Returns false with non-square matrices.

### See also
The function determines if the matrix is triangular (if all elements above or below the main diagonal are zero).

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.is_square

## matrix.is_triangular()
The function determines if the matrix is triangular (if all elements above or below the main diagonal are zero).

### Syntax
```
id (matrix<int/float>) Matrix object to test.
```

### Arguments

```
id (matrix<int/float>) Matrix object to test.
```

### Returns
```
Returns true if the id matrix is triangular, false otherwise.
```

### Remarks
Returns false with non-square matrices.

### See also
The function determines if all elements of the matrix are zero.

### See also
- matrix.new<type>
- matrix.set
- matrix.is_square

## matrix.is_zero()
The function determines if all elements of the matrix are zero.

### Syntax
```
id (matrix<int/float>) Matrix object to check.
```

### Arguments
```
id (matrix<int/float>) Matrix object to check.
```

### Returns
```
Returns true if all elements of the id matrix are zero, false otherwise.
```

### See also

The function returns the Kronecker product for the id1 and id2 matrices.

### See also
- matrix.new<type>
- matrix.get
- matrix.set

## matrix.kron()
The function returns the Kronecker product for the id1 and id2 matrices.

### Syntax & Overloads
id1 (matrix<int/float>) First matrix object.

### Arguments
```
id1 (matrix<int/float>) First matrix object.
```

### Example
A new matrix containing the Kronecker product of id1 and id2.

### Returns
```
A new matrix containing the Kronecker product of id1 and id2.
```

### See also
The function returns the largest value from the matrix elements.

### See also
- matrix.new<type>
- matrix.mult
- str.tostring
- table.new

## matrix.max()
The function returns the largest value from the matrix elements.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
The maximum value from the id matrix.

### Returns
```
The maximum value from the id matrix.
```

### See also
The function calculates the median ("the middle" value) of matrix elements.

### See also
- matrix.new<type>
- matrix.min
- matrix.avg
- matrix.sort

## matrix.median()
The function calculates the median ("the middle" value) of matrix elements.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
Note that na elements of the matrix are not considered when calculating the median.

### Remarks
Note that na elements of the matrix are not considered when calculating the median.

### See also
The function returns the smallest value from the matrix elements.

### See also
- matrix.new<type>
- matrix.mode
- matrix.sort
- matrix.avg

## matrix.min()
The function returns the smallest value from the matrix elements.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
The smallest value from the id matrix.

### Returns
```
The smallest value from the id matrix.
```

### See also
The function calculates the mode of the matrix, which is the most frequently occurring value from the matrix elements. When there are multiple values occurring equally frequently, the function returns the smallest of those values.

### See also

- matrix.new<type>
- matrix.max
- matrix.avg
- matrix.sort

## matrix.mode()
The function calculates the mode of the matrix, which is the most frequently occurring value from the matrix elements. When there are multiple values occurring equally frequently, the function returns the smallest of those values.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
The most frequently occurring value from the id matrix. If none exists, returns the smallest value instead.

### Returns
```
The most frequently occurring value from the id matrix. If none exists, returns the
smallest value instead.
```

### Remarks
Note that na elements of the matrix are not considered when calculating the mode.

### See also
The function returns a new matrix resulting from the product between the matrices id1 and id2, or between an id1 matrix and an id2 scalar (a numerical value), or between an id1 matrix and an id2 vector (an array of values).

### See also
- matrix.new<type>
- matrix.set
- matrix.median

- matrix.sort
- matrix.avg

## matrix.mult()
The function returns a new matrix resulting from the product between the matrices id1 and id2, or between an id1 matrix and an id2 scalar (a numerical value), or between an id1 matrix and an id2 vector (an array of values).

### Syntax & Overloads
id1 (matrix<int>) First matrix object.

### Arguments
```
id1 (matrix<int>) First matrix object.
```

### Example
Product of a matrix and a scalar

### Example
Product of a matrix and an array vector

### Example
A new matrix object containing the product of id2 and id1.

### Returns
```
A new matrix object containing the product of id2 and id1.
```

### See also
The function creates a new matrix object. A matrix is a two-dimensional data structure containing rows and columns. All elements in the matrix must be of the type specified in the type template ("<type>").

### See also
- matrix.new<type>
- matrix.sum
- matrix.diff

## matrix.new<type>()
The function creates a new matrix object. A matrix is a two-dimensional data structure containing rows and columns. All elements in the matrix must be of the type specified in the type template ("<type>").

### Syntax
```
rows (series int) Initial row count of the matrix. Optional. The default value is 0.
```

### Arguments
```
rows (series int) Initial row count of the matrix. Optional. The default value is 0.
```

### Example
Create a matrix from array values

### Example
Create a matrix from an input.text_area() field

### Example
Create matrix from random values

### Example
The ID of the new matrix object.

### Returns
```
The ID of the new matrix object.
```

### See also
The function returns the pseudoinverse of a matrix.

### See also
- matrix.set
- matrix.fill
- matrix.columns
- matrix.rows
- array.new<type>

## matrix.pinv()
The function returns the pseudoinverse of a matrix.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
A new matrix containing the pseudoinverse of the id matrix.

### Returns
```
A new matrix containing the pseudoinverse of the id matrix.
```

### Remarks
The function is calculated using a Moore–Penrose inverse formula based on singular-value decomposition of a matrix. For non-singular square matrices this function returns the result of matrix.inv.

### See also
The function calculates the product of the matrix by itself power times.

### See also
- matrix.new<type>
- matrix.set
- matrix.inv

## matrix.pow()
The function calculates the product of the matrix by itself power times.

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
The product of the id matrix by itself power times.

### Returns
```
The product of the id matrix by itself power times.
```

### See also
The function calculates the rank of the matrix.

### See also
- matrix.new<type>
- matrix.set
- matrix.mult

## matrix.rank()
The function calculates the rank of the matrix.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

```

### Example
The rank of the id matrix.

### Returns
```
The rank of the id matrix.
```

See also
The function removes the column at column index of the id matrix and returns an array containing the removed column's values.

See also
- matrix.new<type>
- matrix.set
- str.tostring

matrix.remove_col()
The function removes the column at column index of the id matrix and returns an array containing the removed column's values.

Syntax
```
id (any matrix type) A matrix object.
```

Arguments
```
id (any matrix type) A matrix object.
```

Example
An array containing the elements of the column removed from the id matrix.

Returns
```
An array containing the elements of the column removed from the id matrix.

```

Remarks
Indexing of rows and columns starts at zero. It is far more efficient to declare matrices with explicit dimensions than to build them by adding or removing columns. Deleting a column is also much slower than deleting a row with the matrix.remove_row function.

See also
The function removes the row at row index of the id matrix and returns an array containing the removed row's values.

See also
- matrix.new<type>
- matrix.set
- matrix.copy
- matrix.remove_row

matrix.remove_row()
The function removes the row at row index of the id matrix and returns an array containing the removed row's values.

Syntax
```
id (any matrix type) A matrix object.
```

Arguments
```
id (any matrix type) A matrix object.
```

Example
An array containing the elements of the row removed from the id matrix.

Returns
```
An array containing the elements of the row removed from the id matrix.
```
```

### Remarks
Indexing of rows and columns starts at zero. It is far more efficient to declare matrices with explicit dimensions than to build them by adding or removing rows.

### See also
The function rebuilds the id matrix to rows x cols dimensions.

### See also
- matrix.new<type>
- matrix.set
- matrix.copy
- matrix.remove_col

## matrix.reshape()
The function rebuilds the id matrix to rows x cols dimensions.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
The function reverses the order of rows and columns in the matrix id. The first row and first column become the last, and the last become the first.

### See also
The function reverses the order of rows and columns in the matrix id. The first row and first column become the last, and the last become the first.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.add_row

- matrix.add_col

## matrix.reverse()
The function reverses the order of rows and columns in the matrix id. The first row and first column become the last, and the last become the first.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
The function creates a one-dimensional array from the elements of a matrix row.

### See also
The function creates a one-dimensional array from the elements of a matrix row.

### See also
- matrix.new<type>
- matrix.set
- matrix.columns
- matrix.rows
- matrix.reshape

## matrix.row()
The function creates a one-dimensional array from the elements of a matrix row.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments

```
id (any matrix type) A matrix object.
```

### Example
An array ID containing the row values of the id matrix.

### Returns
```
An array ID containing the row values of the id matrix.
```

### Remarks
Indexing of rows starts at 0.

### See also
The function returns the number of rows in the matrix.

### See also
- matrix.new<type>
- matrix.get
- array.get
- matrix.col
- matrix.rows

## matrix.rows()
The function returns the number of rows in the matrix.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example

The number of rows in the matrix id.

### Returns
```
The number of rows in the matrix id.
```

### See also
The function assigns value to the element at the row and column of the id matrix.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.row

## matrix.set()
The function assigns value to the element at the row and column of the id matrix.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
The function rearranges the rows in the id matrix following the sorted order of the values in the column.

### See also
The function rearranges the rows in the id matrix following the sorted order of the values in the column.

### See also

- matrix.new<type>
- matrix.get
- matrix.columns
- matrix.rows

## matrix.sort()
The function rearranges the rows in the id matrix following the sorted order of the values in the column.

### Syntax
```
id (matrix<int/float/string>) A matrix object to be sorted.
```

### Arguments
```
id (matrix<int/float/string>) A matrix object to be sorted.
```

### Example
The function extracts a submatrix of the id matrix within the specified indices.

### See also
The function extracts a submatrix of the id matrix within the specified indices.

### See also
- matrix.new<type>
- matrix.max
- matrix.min
- matrix.avg

## matrix.submatrix()
The function extracts a submatrix of the id matrix within the specified indices.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
A new matrix object containing the submatrix of the id matrix defined by the from_row, to_row, from_column and to_column indices.

### Returns
```
A new matrix object containing the submatrix of the id matrix defined by the
from_row, to_row, from_column and to_column indices.
```

### Remarks
Indexing of the rows and columns starts at zero.

### See also
The function returns a new matrix resulting from the sum of two matrices id1 and id2, or of an id1 matrix and an id2 scalar (a numerical value).

### See also
- matrix.new<type>
- matrix.set
- matrix.row
- matrix.col
- matrix.reshape

## matrix.sum()
The function returns a new matrix resulting from the sum of two matrices id1 and id2, or of an id1 matrix and an id2 scalar (a numerical value).

### Syntax & Overloads
id1 (matrix<int>) First matrix object.

### Arguments
```
id1 (matrix<int>) First matrix object.
```

```

### Example
Sum of a matrix and scalar

### Example
A new matrix object containing the sum of id2 and id1.

### Returns
```
A new matrix object containing the sum of id2 and id1.
```

See also
The function swaps the columns at the index column1 and column2 in the id matrix.

See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

matrix.swap_columns()
The function swaps the columns at the index column1 and column2 in the id matrix.

Syntax
```
id (any matrix type) A matrix object.
```

Arguments
```
id (any matrix type) A matrix object.
```

Example
Indexing of the rows and columns starts at zero.
```

### Remarks
Indexing of the rows and columns starts at zero.

### See also
The function swaps the rows at the index row1 and row2 in the id matrix.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.swap_rows()
The function swaps the rows at the index row1 and row2 in the id matrix.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
Indexing of the rows and columns starts at zero.

### Remarks
Indexing of the rows and columns starts at zero.

### See also
The function calculates the trace of a matrix (the sum of the main diagonal's elements).

### See also
- matrix.new<type>

- matrix.get
- matrix.set
- matrix.columns
- matrix.swap_columns

## matrix.trace()
The function calculates the trace of a matrix (the sum of the main diagonal's elements).

### Syntax & Overloads
id (matrix<int/float>) A matrix object.

### Arguments
```
id (matrix<int/float>) A matrix object.
```

### Example
The trace of the id matrix.

### Returns
```
The trace of the id matrix.
```

### See also
The function creates a new, transposed version of the id. This interchanges the row and column index of each element.

### See also
- matrix.new<type>
- matrix.get
- matrix.set
- matrix.columns
- matrix.rows

## matrix.transpose()
The function creates a new, transposed version of the id. This interchanges the row and column index of each element.

### Syntax
```
id (any matrix type) A matrix object.
```

### Arguments
```
id (any matrix type) A matrix object.
```

### Example
A new matrix containing the transposed version of the id matrix.

### Returns
```
A new matrix containing the transposed version of the id matrix.
```

### See also
Function sets the maximum number of bars that is available for historical reference of a given built-in or user variable. When operator '[]' is applied to a variable - it is a reference to a historical value of that variable.

### See also
- matrix.new<type>
- matrix.set
- matrix.columns
- matrix.rows
- matrix.reshape
- matrix.reverse

## max_bars_back()
Function sets the maximum number of bars that is available for historical reference of a given built-in or user variable. When operator '[]' is applied to a variable - it is a reference to a historical value of that variable.

### Syntax
```
- var (series int/float/bool/color/label/line) Series variable identifier for which
```

history buffer should be resized. Possible values are : 'open', 'high', 'low', 'close', 'volume', 'time', or any user defined variable id.
```

Arguments
```
- var (series int/float/bool/color/label/line) Series variable identifier for which history buffer should be resized. Possible values are : 'open', 'high', 'low', 'close', 'volume', 'time', or any user defined variable id.
```

Example
void

Returns
```
void
```

Remarks
At the moment 'max_bars_back' cannot be applied to built-ins like 'hl2', 'hlc3', 'ohlc4'. Please use multiple 'max_bars_back' calls as workaround here (e.g. instead of a single ‘max_bars_back(hl2, 100)’ call you should call the function twice: ‘max_bars_back(high, 100), max_bars_back(low, 100)’).

See also
time (series int) UNIX time in milliseconds.

See also
- indicator

minute()
time (series int) UNIX time in milliseconds.

Syntax & Overloads
time (series int) UNIX time in milliseconds.

Arguments
```

time (series int) UNIX time in milliseconds.
```

Returns
```
Minute (in exchange timezone) for provided UNIX time.
```

Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

See also
time (series int) UNIX time in milliseconds.

See also
- minute
- time
- year
- month
- dayofmonth
- dayofweek
- hour
- second

month()
time (series int) UNIX time in milliseconds.

Syntax & Overloads
time (series int) UNIX time in milliseconds.

Arguments
```
time (series int) UNIX time in milliseconds.
```

Returns
```
Month (in exchange timezone) for provided UNIX time.
```
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
Tests if x is na.

### See also
- month
- time
- year
- dayofmonth
- dayofweek
- hour
- minute
- second

## na()
Tests if x is na.

### Syntax & Overloads
x (simple int/float) Value to be tested.

### Arguments
```
x (simple int/float) Value to be tested.
```

### Example
Returns true if x is na, false otherwise.

### Returns
```
Returns true if x is na, false otherwise.
```

### See also
Replaces NaN values with zeros (or given value) in a series.

### See also
- na
- fixnan
- nz

## nz()
Replaces NaN values with zeros (or given value) in a series.

### Syntax & Overloads
source (simple color) Series of values to process.

### Arguments
```
source (simple color) Series of values to process.
```

### Example
The value of source if it is not na. If the value of source is na, returns zero, or the replacement argument when one is used.

### Returns
```
The value of source if it is not na. If the value of source is na, returns zero, or the replacement argument when one is used.
```

### See also
Plots a series of data on the chart.

### See also
- na
- na
- fixnan

## plot()
Plots a series of data on the chart.

### Syntax
```
series (series int/float) Series of data to be plotted. Required argument.
```

### Arguments
```
series (series int/float) Series of data to be plotted. Required argument.
```

### Example
A plot object, that can be used in fill

### Returns
```
A plot object, that can be used in fill
```

### See also
Plots up and down arrows on the chart. Up arrow is drawn at every indicator positive value, down arrow is drawn at every negative value. If indicator returns na then no arrow is drawn. Arrows has different height, the more absolute indicator value the longer arrow is drawn.

### See also
- plotshape
- plotchar
- plotarrow
- barcolor
- bgcolor
- fill

## plotarrow()
Plots up and down arrows on the chart. Up arrow is drawn at every indicator positive value, down arrow is drawn at every negative value. If indicator returns na then no arrow is drawn. Arrows has different height, the more absolute indicator value the longer arrow is drawn.

### Syntax
```
```

series (series int/float) Series of data to be plotted as arrows. Required argument.
```

Arguments
```
series (series int/float) Series of data to be plotted as arrows. Required argument.
```

Example
Use plotarrow function in conjunction with 'overlay=true' indicator parameter!

Remarks
Use plotarrow function in conjunction with 'overlay=true' indicator parameter!

See also
Plots ohlc bars on the chart.

See also
- plot
- plotshape
- plotchar
- barcolor
- bgcolor

plotbar()
Plots ohlc bars on the chart.

Syntax
```
open (series int/float) Open series of data to be used as open values of bars. Required argument.
```

Arguments
```
open (series int/float) Open series of data to be used as open values of bars. Required argument.
```
```

### Example
Even if one value of open, high, low or close equal NaN then bar no draw.

### Remarks
Even if one value of open, high, low or close equal NaN then bar no draw.

### See also
Plots candles on the chart.

### See also
- plotcandle

## plotcandle()
Plots candles on the chart.

### Syntax
```
open (series int/float) Open series of data to be used as open values of candles. Required argument.
```

### Arguments
```
open (series int/float) Open series of data to be used as open values of candles. Required argument.
```

### Example
Even if one value of open, high, low or close equal NaN then bar no draw.

### Remarks
Even if one value of open, high, low or close equal NaN then bar no draw.

### See also
Plots visual shapes using any given one Unicode character on the chart.

### See also
- plotbar

## plotchar()
Plots visual shapes using any given one Unicode character on the chart.

### Syntax
```
series (series int/float/bool) Series of data to be plotted as shapes. Series is
treated as a series of boolean values for all location values except
location.absolute. Required argument.
```

### Arguments
```
series (series int/float/bool) Series of data to be plotted as shapes. Series is
treated as a series of boolean values for all location values except
location.absolute. Required argument.
```

### Example
Use plotchar function in conjunction with 'overlay=true' indicator parameter!

### Remarks
Use plotchar function in conjunction with 'overlay=true' indicator parameter!

### See also
Plots visual shapes on the chart.

### See also
- plot
- plotshape
- plotarrow
- barcolor
- bgcolor

## plotshape()
Plots visual shapes on the chart.

### Syntax
```
series (series int/float/bool) Series of data to be plotted as shapes. Series is
treated as a series of boolean values for all location values except
location.absolute. Required argument.
```

### Arguments
```
series (series int/float/bool) Series of data to be plotted as shapes. Series is
treated as a series of boolean values for all location values except
location.absolute. Required argument.
```

### Example
Use plotshape function in conjunction with 'overlay=true' indicator parameter!

### Remarks
Use plotshape function in conjunction with 'overlay=true' indicator parameter!

### See also
Deletes the specified polyline object. It has no effect if the id doesn't exist.

### See also
- plot
- plotchar
- plotarrow
- barcolor
- bgcolor

## polyline.delete()
Deletes the specified polyline object. It has no effect if the id doesn't exist.

### Syntax
```
id (series polyline) The polyline ID to delete.
```

### Arguments

```
id (series polyline) The polyline ID to delete.
```

## polyline.new()
Creates a new polyline instance and displays it on the chart, sequentially connecting all of the points in the points array with line segments. The segments in the drawing can be straight or curved depending on the curved parameter.

### Syntax
```
points (array<chart.point>) An array of chart.point objects for the drawing to sequentially connect.
```

### Arguments
```
points (array<chart.point>) An array of chart.point objects for the drawing to sequentially connect.
```

### Example
The ID of a new polyline object that a script can use in other polyline.*() functions.

### Returns
```
The ID of a new polyline object that a script can use in other polyline.*() functions.
```

### See also
Provides a daily rate that can be used to convert a value expressed in the from currency to another in the to currency.

### See also
- chart.point.new

## request.currency_rate()
Provides a daily rate that can be used to convert a value expressed in the from currency to another in the to currency.

### Syntax
```
- from (series string) The currency in which the value to be converted is expressed. Possible values : a three-letter string with the currency code in the ISO 4217 format (e.g. "USD"), or one of the built-in variables that return currency codes, like syminfo.currency or currency.USD.
```

### Arguments

```
- from (series string) The currency in which the value to be converted is expressed. Possible values : a three-letter string with the currency code in the ISO 4217 format (e.g. "USD"), or one of the built-in variables that return currency codes, like syminfo.currency or currency.USD.
```

### Example
If from and to arguments are equal, function returns 1. Please note that using this variable/function can cause indicator repainting.

### Remarks
If from and to arguments are equal, function returns 1. Please note that using this variable/function can cause indicator repainting.

## request.dividends()
Requests dividends data for the specified symbol.

### Syntax
```
- ticker (series string) Symbol. Note that the symbol should be passed with a prefix. For example : "NASDAQ:AAPL" instead of "AAPL". Using syminfo.ticker will cause an error. Use syminfo.tickerid instead.
```

### Arguments
```
```

- ticker (series string) Symbol. Note that the symbol should be passed with a prefix. For example : "NASDAQ:AAPL" instead of "AAPL". Using syminfo.ticker will cause an error. Use syminfo.tickerid instead.
```

Example
Requested series, or n/a if there is no dividends data for the specified symbol.

Returns
```
Requested series, or n/a if there is no dividends data for the specified symbol.
```

See also
Requests earnings data for the specified symbol.

See also
- request.earnings
- request.splits
- request.security
- syminfo.tickerid

request.earnings()
Requests earnings data for the specified symbol.

Syntax
```
- ticker (series string) Symbol. Note that the symbol should be passed with a prefix. For example : "NASDAQ:AAPL" instead of "AAPL". Using syminfo.ticker will cause an error. Use syminfo.tickerid instead.
```

Arguments
```
- ticker (series string) Symbol. Note that the symbol should be passed with a prefix. For example : "NASDAQ:AAPL" instead of "AAPL". Using syminfo.ticker will cause an error. Use syminfo.tickerid instead.
```

Example
```

Requested series, or n/a if there is no earnings data for the specified symbol.

### Returns
```
Requested series, or n/a if there is no earnings data for the specified symbol.
```

### See also
Requests economic data for a symbol. Economic data includes information such as the state of a country's economy (GDP, inflation rate, etc.) or of a particular industry (steel production, ICU beds, etc.).

### See also
- request.dividends
- request.splits
- request.security
- syminfo.tickerid

## request.economic()
Requests economic data for a symbol. Economic data includes information such as the state of a country's economy (GDP, inflation rate, etc.) or of a particular industry (steel production, ICU beds, etc.).

### Syntax
```
country_code (series string) The code of the country (e.g. "US") or the region (e.g. "EU") for which the economic data is requested. The Help Center article lists the countries and their codes. The countries for which information is available vary with metrics. The Help Center article for each metric lists the countries for which the metric is available.
```

### Arguments
```
country_code (series string) The code of the country (e.g. "US") or the region (e.g. "EU") for which the economic data is requested. The Help Center article lists the countries and their codes. The countries for which information is available vary with metrics. The Help Center article for each metric lists the countries for which the metric is available.
```

### Example
Requested series.

### Returns
```
Requested series.
```

### Remarks
Economic data can also be accessed from charts, just like a regular symbol. Use "ECONOMIC" as the exchange name and {country_code}{field} as the ticker. The name of US GDP data is thus "ECONOMIC:USGDP".

### See also
Requests financial series for symbol.

### See also
- request.financial

## request.financial()
Requests financial series for symbol.

### Syntax
```
- symbol (series string) Symbol. Note that the symbol should be passed with a
prefix. For example : "NASDAQ:AAPL" instead of "AAPL".
```

### Arguments
```
- symbol (series string) Symbol. Note that the symbol should be passed with a
prefix. For example : "NASDAQ:AAPL" instead of "AAPL".
```

### Example
Requested series.

### Returns
```
```

Requested series.
```

See also
Note: This function has been deprecated due to the API change from NASDAQ Data Link. Requests for "QUANDL" symbols are no longer valid and requests for them return a runtime error.

See also
- request.security
- syminfo.tickerid

request.quandl()
Note: This function has been deprecated due to the API change from NASDAQ Data Link. Requests for "QUANDL" symbols are no longer valid and requests for them return a runtime error.

Syntax
```
- ticker (series string) Symbol. Note that the name of a time series and Quandl data feed should be divided by a forward slash. For example : "CFTC/SB_FO_ALL".
```

Arguments
```
- ticker (series string) Symbol. Note that the name of a time series and Quandl data feed should be divided by a forward slash. For example : "CFTC/SB_FO_ALL".
```

Example
Requested series.

Returns
```
Requested series.
```

Remarks
You can learn more about how to find ticker and index values in our Help Center.
```

### See also
Requests the result of an expression from a specified context (symbol and timeframe).

### See also
- request.security
- syminfo.tickerid

## request.security()
Requests the result of an expression from a specified context (symbol and timeframe).

### Syntax
```
symbol (series string) Symbol or ticker identifier of the requested data. Use an empty string or syminfo.tickerid to request data using the chart's symbol. To retrieve data with additional modifiers (extended sessions, dividend adjustments, non-standard chart types like Heikin Ashi and Renko, etc.), create a custom ticker ID for the request using the functions in the ticker.* namespace.
```

### Arguments
```
symbol (series string) Symbol or ticker identifier of the requested data. Use an empty string or syminfo.tickerid to request data using the chart's symbol. To retrieve data with additional modifiers (extended sessions, dividend adjustments, non-standard chart types like Heikin Ashi and Renko, etc.), create a custom ticker ID for the request using the functions in the ticker.* namespace.
```

### Example
A result determined by expression.

### Example
A result determined by expression.

### Returns
```
A result determined by expression.
```

### Remarks
Scripts using this function might calculate differently on historical and realtime bars, leading to repainting.

### See also
Requests the results of an expression from a specified symbol on a timeframe lower than or equal to the chart's timeframe. It returns an array containing one element for each lower-timeframe bar within the chart bar. On a 5-minute chart, requesting data using a timeframe argument of "1" typically returns an array with five elements representing the value of the expression on each 1-minute bar, ordered by time with the earliest value first.

### See also
- syminfo.ticker
- syminfo.tickerid
- timeframe.period
- ticker.new
- ticker.modify
- request.security_lower_tf
- request.dividends
- request.earnings
- request.splits
- request.financial

## request.security_lower_tf()
Requests the results of an expression from a specified symbol on a timeframe lower than or equal to the chart's timeframe. It returns an array containing one element for each lower-timeframe bar within the chart bar. On a 5-minute chart, requesting data using a timeframe argument of "1" typically returns an array with five elements representing the value of the expression on each 1-minute bar, ordered by time with the earliest value first.

### Syntax
```
symbol (series string) Symbol or ticker identifier of the requested data. Use an
empty string or syminfo.tickerid to request data using the chart's symbol. To
retrieve data with additional modifiers (extended sessions, dividend adjustments,
non-standard chart types like Heikin Ashi and Renko, etc.), create a custom ticker
ID for the request using the functions in the ticker.* namespace.
```

### Arguments
```
symbol (series string) Symbol or ticker identifier of the requested data. Use an empty string or syminfo.tickerid to request data using the chart's symbol. To retrieve data with additional modifiers (extended sessions, dividend adjustments, non-standard chart types like Heikin Ashi and Renko, etc.), create a custom ticker ID for the request using the functions in the ticker.* namespace.
```

### Example
An array of a type determined by expression, or a tuple of these.

### Returns
```
An array of a type determined by expression, or a tuple of these.
```

### Remarks
Scripts using this function might calculate differently on historical and realtime bars, leading to repainting.

### See also
Requests data from a user-maintained GitHub repository and returns it as a series. An in-depth tutorial on how to add new data can be found here.

### See also
- request.security
- syminfo.ticker
- syminfo.tickerid
- timeframe.period
- ticker.new
- request.dividends
- request.earnings
- request.splits
- request.financial

## request.seed()
Requests data from a user-maintained GitHub repository and returns it as a series. An in-depth tutorial on how to add new data can be found here.

### Syntax

```
source (series string) Name of the GitHub repository.
```

### Arguments
```
source (series string) Name of the GitHub repository.
```

### Example
Requested series or tuple of series, which may include array/matrix IDs.

### Returns
```
Requested series or tuple of series, which may include array/matrix IDs.
```

## request.splits()
Requests splits data for the specified symbol.

### Syntax
```
- ticker (series string) Symbol. Note that the symbol should be passed with a
prefix. For example : "NASDAQ:AAPL" instead of "AAPL". Using syminfo.ticker will
cause an error. Use syminfo.tickerid instead.
```

### Arguments
```
- ticker (series string) Symbol. Note that the symbol should be passed with a
prefix. For example : "NASDAQ:AAPL" instead of "AAPL". Using syminfo.ticker will
cause an error. Use syminfo.tickerid instead.
```

### Example
Requested series, or n/a if there is no splits data for the specified symbol.

### Returns
```
```

Requested series, or n/a if there is no splits data for the specified symbol.
```

See also
When called, causes a runtime error with the error message specified in the message argument.

See also
- request.earnings
- request.dividends
- request.security
- syminfo.tickerid

runtime.error()
When called, causes a runtime error with the error message specified in the message argument.

Syntax
```
message (series string) Error message.
```

Arguments
```
message (series string) Error message.
```

second()
time (series int) UNIX time in milliseconds.

Syntax & Overloads
time (series int) UNIX time in milliseconds.

Arguments
```
time (series int) UNIX time in milliseconds.
```
```

### Returns
```
Second (in exchange timezone) for provided UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
Returns true if the source string contains the str substring, false otherwise.

### See also
- second
- time
- year
- month
- dayofmonth
- dayofweek
- hour
- minute

## str.contains()
Returns true if the source string contains the str substring, false otherwise.

### Syntax & Overloads
source (const string) Source string.

### Arguments
```
source (const string) Source string.
```

### Example
True if the str was found in the source string, false otherwise.

### Returns
```
True if the str was found in the source string, false otherwise.
```

```

See also
Returns true if the source string ends with the substring specified in str, false otherwise.

See also
- str.pos
- str.match

str.endswith()
Returns true if the source string ends with the substring specified in str, false otherwise.

Syntax & Overloads
source (const string) Source string.

Arguments
```
source (const string) Source string.
```

Returns
```
True if the source string ends with the substring specified in str, false
otherwise.
```

See also
Converts the formatting string and value(s) into a formatted string. The formatting string can contain literal text and one placeholder in curly braces {} for each value to be formatted. Each placeholder consists of the index of the required argument (beginning at 0) that will replace it, and an optional format specifier. The index represents the position of that argument in the str.format argument list.

See also
- str.startswith
```

## str.format()
Converts the formatting string and value(s) into a formatted string. The formatting string can contain literal text and one placeholder in curly braces {} for each value to be formatted. Each placeholder consists of the index of the required argument (beginning at 0) that will replace it, and an optional format specifier. The index represents the position of that argument in the str.format argument list.

### Syntax & Overloads
formatString (simple string) Format string.

### Arguments
```
formatString (simple string) Format string.
```

### Example
The formatted string.

### Returns
```
The formatted string.
```

### Remarks
By default, formatted numbers will display up to three decimals with no trailing zeros.

## str.format_time()
Converts the time timestamp into a string formatted according to format and timezone.

### Syntax
```
time (series int) UNIX time, in milliseconds.
```

### Arguments
```
time (series int) UNIX time, in milliseconds.
```

~~~
```

Example
The formatted string.

Returns
```
The formatted string.
```

Remarks
The M, d, h, H, m and s tokens can all be doubled to generate leading zeros. For
example, the month of January will display as 1 with M, or 01 with MM.

str.length()
Returns an integer corresponding to the amount of chars in that string.

Syntax & Overloads
string (const string) Source string.

Arguments
```
string (const string) Source string.
```

Returns
```
The number of chars in source string.
```

str.lower()
Returns a new string with all letters converted to lowercase.

Syntax & Overloads
source (const string) String to be converted.
~~~

### Arguments
```
source (const string) String to be converted.
```

### Returns
```
A new string with all letters converted to lowercase.
```

### See also
Returns the new substring of the source string if it matches a regex regular expression, an empty string otherwise.

### See also
- str.upper

## str.match()
Returns the new substring of the source string if it matches a regex regular expression, an empty string otherwise.

### Syntax & Overloads
source (simple string) Source string.

### Arguments
```
source (simple string) Source string.
```

### Example
The new substring of the source string if it matches a regex regular expression, an empty string otherwise.

### Returns
```
The new substring of the source string if it matches a regex regular expression, an empty string otherwise.
```

### Remarks
Function returns first occurrence of the regular expression in the source string.

### See also
Returns the position of the first occurrence of the str string in the source string, 'na' otherwise.

### See also
- str.contains
- str.substring

## str.pos()
Returns the position of the first occurrence of the str string in the source string, 'na' otherwise.

### Syntax & Overloads
source (const string) Source string.

### Arguments
```
source (const string) Source string.
```

### Returns
```
Position of the str string in the source string.
```

### Remarks
Strings indexing starts at 0.

### See also
Constructs a new string containing the source string repeated repeat times with the separator injected between each repeated instance.

### See also
- str.contains
- str.match

- str.substring

## str.repeat()
Constructs a new string containing the source string repeated repeat times with the separator injected between each repeated instance.

### Syntax & Overloads
source (const string) String to repeat.

### Arguments
```
source (const string) String to repeat.
```

### Example
Returns na if the source is na.

### Remarks
Returns na if the source is na.

## str.replace()
Returns a new string with the Nth occurrence of the target string replaced by the replacement string, where N is specified in occurrence.

### Syntax & Overloads
source (const string) Source string.

### Arguments
```
source (const string) Source string.
```

### Example
Processed string.

### Returns

```
Processed string.
```

### See also
Replaces each occurrence of the target string in the source string with the replacement string.

### See also
- str.replace_all
- str.match

## str.replace_all()
Replaces each occurrence of the target string in the source string with the replacement string.

### Syntax & Overloads
source (simple string) Source string.

### Arguments
```
source (simple string) Source string.
```

### Returns
```
Processed string.
```

## str.split()
Divides a string into an array of substrings and returns its array id.

### Syntax
```
string (series string) Source string.
```

### Arguments
```
string (series string) Source string.
```

### Returns
```
The id of an array of strings.
```

## str.startswith()
Returns true if the source string starts with the substring specified in str, false otherwise.

### Syntax & Overloads
source (const string) Source string.

### Arguments
```
source (const string) Source string.
```

### Returns
```
True if the source string starts with the substring specified in str, false
otherwise.
```

### See also
Returns a new string that is a substring of the source string. The substring begins with the character at the index specified by begin_pos and extends to 'end_pos - 1' of the source string.

### See also
- str.endswith

## str.substring()
Returns a new string that is a substring of the source string. The substring begins

with the character at the index specified by begin_pos and extends to 'end_pos - 1' of the source string.

### Syntax & Overloads
source (const string) Source string from which to extract the substring.

### Arguments
```
source (const string) Source string from which to extract the substring.
```

### Example
The substring extracted from the source string.

### Returns
```
The substring extracted from the source string.
```

### Remarks
Strings indexing starts from 0. If begin_pos is equal to end_pos, the function returns an empty string.

### See also
Converts a value represented in string to its "float" equivalent.

### See also
- str.contains
- str.pos
- str.match

## str.tonumber()
Converts a value represented in string to its "float" equivalent.

### Syntax & Overloads
string (const string) String containing the representation of an integer or floating point value.

### Arguments
```
string (const string) String containing the representation of an integer or floating point value.
```

### Returns
```
A "float" equivalent of the value in string. If the value is not a properly formed integer or floating point value, the function returns na.
```

## str.tostring()
value (simple int/float) Value or array ID whose elements are converted to a string.

### Syntax & Overloads
value (simple int/float) Value or array ID whose elements are converted to a string.

### Arguments
```
value (simple int/float) Value or array ID whose elements are converted to a string.
```

### Returns
```
The string representation of the value argument.
```

### Remarks
The formatting of float values will also round those values when necessary, e.g. str.tostring(3.99, '#') will return "4".

## str.trim()
Constructs a new string with all consecutive whitespaces and other control characters (e.g., “\n”, “\t”, etc.) removed from the left and right of the source.

### Syntax & Overloads
source (const string) String to trim.

### Arguments
```
source (const string) String to trim.
```

### Example
Returns an empty string ("") if the result is empty after the trim or if the source is na.

### Remarks
Returns an empty string ("") if the result is empty after the trim or if the source is na.

## str.upper()
Returns a new string with all letters converted to uppercase.

### Syntax & Overloads
source (const string) String to be converted.

### Arguments
```
source (const string) String to be converted.
```

### Returns
```
A new string with all letters converted to uppercase.
```

### See also
This declaration statement designates the script as a strategy and sets a number of strategy-related properties.

### See also
- str.lower

## strategy()
This declaration statement designates the script as a strategy and sets a number of strategy-related properties.

### Syntax
```
title (const string) The title of the script. It is displayed on the chart when no shorttitle argument is used, and becomes the publication's default title when publishing the script.
```

### Arguments
```
title (const string) The title of the script. It is displayed on the chart when no shorttitle argument is used, and becomes the publication's default title when publishing the script.
```

### Example
You can learn more about strategies in our User Manual.

### Remarks
You can learn more about strategies in our User Manual.

### See also
Cancels a pending or unfilled order with a specific identifier. If multiple unfilled orders share the same ID, calling this command with that ID as the id argument cancels all of them. If a script calls this command with an id representing the ID of a filled order, it has no effect.

### See also
- indicator
- library

## strategy.cancel()
Cancels a pending or unfilled order with a specific identifier. If multiple

unfilled orders share the same ID, calling this command with that ID as the id argument cancels all of them. If a script calls this command with an id representing the ID of a filled order, it has no effect.

### Syntax
```
id (series string) The identifier of the unfilled order to cancel.
```

### Arguments
```
id (series string) The identifier of the unfilled order to cancel.
```

### Example
Cancels all pending or unfilled orders, regardless of their identifiers.

## strategy.cancel_all()
Cancels all pending or unfilled orders, regardless of their identifiers.

### Syntax
```
Creates an order to exit from the part of a position opened by entry orders with a specific identifier. If multiple entries in the position share the same ID, the orders from this command apply to all those entries, starting from the first open trade, when its calls use that ID as the id argument.
```

### Example
Creates an order to exit from the part of a position opened by entry orders with a specific identifier. If multiple entries in the position share the same ID, the orders from this command apply to all those entries, starting from the first open trade, when its calls use that ID as the id argument.

## strategy.close()
Creates an order to exit from the part of a position opened by entry orders with a specific identifier. If multiple entries in the position share the same ID, the orders from this command apply to all those entries, starting from the first open

trade, when its calls use that ID as the id argument.

### Syntax
```
id (series string) The entry identifier of the open trades to close.
```

### Arguments
```
id (series string) The entry identifier of the open trades to close.
```

### Example
When a position consists of several open trades and the close_entries_rule in the strategy declaration statement is "FIFO" (default), a strategy.close call exits from the position starting with the first open trade. This behavior applies even if the id value is the entry ID of different open trades. However, in that case, the maximum exit order size still depends on the trades opened by orders with the id identifier. For more information, see this section of our User Manual.

### Remarks
When a position consists of several open trades and the close_entries_rule in the strategy declaration statement is "FIFO" (default), a strategy.close call exits from the position starting with the first open trade. This behavior applies even if the id value is the entry ID of different open trades. However, in that case, the maximum exit order size still depends on the trades opened by orders with the id identifier. For more information, see this section of our User Manual.

## strategy.close_all()
Creates an order to close an open position completely, regardless of the identifiers of the entry orders that opened or added to it.

### Syntax & Overloads
comment (series string) Optional. Additional notes on the filled order. If the value is not an empty string, the Strategy Tester and the chart show this text for the order instead of the automatically generated exit identifier. The default is an empty string.

### Arguments
```
```

comment (series string) Optional. Additional notes on the filled order. If the value is not an empty string, the Strategy Tester and the chart show this text for the order instead of the automatically generated exit identifier. The default is an empty string.
```

### Example
Returns the sum of entry and exit fees paid in the closed trade, expressed in strategy.account_currency.

## strategy.closedtrades.commission()
Returns the sum of entry and exit fees paid in the closed trade, expressed in strategy.account_currency.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Example
Returns the bar_index of the closed trade's entry.

### See also
Returns the bar_index of the closed trade's entry.

### See also
- strategy
- strategy.opentrades.commission

## strategy.closedtrades.entry_bar_index()
Returns the bar_index of the closed trade's entry.
```

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Example
Returns the comment message of the closed trade's entry, or na
if there is no entry with this trade_num.

### See also
Returns the comment message of the closed trade's entry, or na
if there is no entry with this trade_num.

### See also
- strategy.closedtrades.exit_bar_index
- strategy.opentrades.entry_bar_index

## strategy.closedtrades.entry_comment()
Returns the comment message of the closed trade's entry, or na
if there is no entry with this trade_num.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Example
Returns the id of the closed trade's entry.

### See also
Returns the id of the closed trade's entry.

### See also
- strategy
- strategy.entry
- strategy.closedtrades

## strategy.closedtrades.entry_id()
Returns the id of the closed trade's entry.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Example
Returns the id of the closed trade's entry.

### Returns
```
Returns the id of the closed trade's entry.
```

### Remarks
The function returns na if trade_num is not in the range: 0 to
strategy.closedtrades-1.

### See also
Returns the price of the closed trade's entry.

### See also
- strategy.closedtrades.entry_bar_index
- strategy.closedtrades.entry_price
- strategy.closedtrades.entry_time

## strategy.closedtrades.entry_price()
Returns the price of the closed trade's entry.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Example
Returns the UNIX time of the closed trade's entry, expressed in milliseconds..

### Example
Returns the UNIX time of the closed trade's entry, expressed in milliseconds..

### See also
Returns the UNIX time of the closed trade's entry, expressed in milliseconds..

### See also
- strategy.closedtrades.entry_price
- strategy.closedtrades.exit_price
- strategy.closedtrades.size
- strategy.closedtrades

## strategy.closedtrades.entry_time()
Returns the UNIX time of the closed trade's entry, expressed in milliseconds..

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Example
Returns the bar_index of the closed trade's exit.

### See also
Returns the bar_index of the closed trade's exit.

### See also
- strategy.opentrades.entry_time
- strategy.closedtrades.exit_time
- time

## strategy.closedtrades.exit_bar_index()
Returns the bar_index of the closed trade's exit.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

```

### Example
Returns the comment message of the closed trade's exit, or
na if there is no entry with this trade_num.

### Example
Returns the comment message of the closed trade's exit, or
na if there is no entry with this trade_num.

### See also
Returns the comment message of the closed trade's exit, or
na if there is no entry with this trade_num.

### See also
- bar_index
- last_bar_index

## strategy.closedtrades.exit_comment()
Returns the comment message of the closed trade's exit, or
na if there is no entry with this trade_num.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Example
Returns the id of the closed trade's exit.

### See also
Returns the id of the closed trade's exit.
```

### See also
- strategy
- strategy.exit
- strategy.close
- strategy.closedtrades

## strategy.closedtrades.exit_id()
Returns the id of the closed trade's exit.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Example
Returns the id of the closed trade's exit.

### Returns
```
Returns the id of the closed trade's exit.
```

### Remarks
The function returns na if trade_num is not in the range: 0 to strategy.closedtrades-1.

### See also
Returns the price of the closed trade's exit.

### See also
- strategy.closedtrades.exit_bar_index

- strategy.closedtrades.exit_price
- strategy.closedtrades.exit_time

## strategy.closedtrades.exit_price()
Returns the price of the closed trade's exit.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Example
Returns the UNIX time of the closed trade's exit, expressed in milliseconds.

### Example
Returns the UNIX time of the closed trade's exit, expressed in milliseconds.

### See also
Returns the UNIX time of the closed trade's exit, expressed in milliseconds.

### See also
- strategy.closedtrades.entry_price

## strategy.closedtrades.exit_time()
Returns the UNIX time of the closed trade's exit, expressed in milliseconds.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Example
Returns the maximum drawdown of the closed trade, i.e., the maximum possible loss during the trade, expressed in strategy.account_currency.

### Example
Returns the maximum drawdown of the closed trade, i.e., the maximum possible loss during the trade, expressed in strategy.account_currency.

### See also
Returns the maximum drawdown of the closed trade, i.e., the maximum possible loss during the trade, expressed in strategy.account_currency.

### See also
- strategy.closedtrades.entry_time

## strategy.closedtrades.max_drawdown()
Returns the maximum drawdown of the closed trade, i.e., the maximum possible loss during the trade, expressed in strategy.account_currency.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Example

The function returns na if trade_num is not in the range: 0 to strategy.closedtrades - 1.

### Remarks
The function returns na if trade_num is not in the range: 0 to strategy.closedtrades - 1.

### See also
Returns the maximum drawdown of the closed trade, i.e., the maximum possible loss during the trade, expressed as a percentage and calculated by formula: Lowest Value During Trade / (Entry Price x Quantity) * 100.

### See also
- strategy.opentrades.max_drawdown
- strategy.max_drawdown

## strategy.closedtrades.max_drawdown_percent()
Returns the maximum drawdown of the closed trade, i.e., the maximum possible loss during the trade, expressed as a percentage and calculated by formula: Lowest Value During Trade / (Entry Price x Quantity) * 100.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### See also
Returns the maximum run up of the closed trade, i.e., the maximum possible profit during the trade, expressed in strategy.account_currency.

### See also
- strategy.closedtrades.max_drawdown
- strategy.max_drawdown

## strategy.closedtrades.max_runup()
Returns the maximum run up of the closed trade, i.e., the maximum possible profit during the trade, expressed in strategy.account_currency.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Example
Returns the maximum run-up of the closed trade, i.e., the maximum possible profit during the trade, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### See also
Returns the maximum run-up of the closed trade, i.e., the maximum possible profit during the trade, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### See also
- strategy.opentrades.max_runup
- strategy.max_runup

## strategy.closedtrades.max_runup_percent()
Returns the maximum run-up of the closed trade, i.e., the maximum possible profit during the trade, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the
```

first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### See also
Returns the profit/loss of the closed trade, expressed in
strategy.account_currency. Losses are expressed as negative values.

### See also
- strategy.closedtrades.max_runup
- strategy.max_runup

## strategy.closedtrades.profit()
Returns the profit/loss of the closed trade, expressed in
strategy.account_currency. Losses are expressed as negative values.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the
first trade is zero.
```

### Example
Returns the profit/loss value of the closed trade, expressed as a percentage.
Losses are expressed as negative values.

### See also
Returns the profit/loss value of the closed trade, expressed as a percentage.
Losses are expressed as negative values.
```

### See also
- strategy.opentrades.profit
- strategy.closedtrades.commission

## strategy.closedtrades.profit_percent()
Returns the profit/loss value of the closed trade, expressed as a percentage. Losses are expressed as negative values.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### See also
Returns the direction and the number of contracts traded in the closed trade. If the value is > 0, the market position was long. If the value is < 0, the market position was short.

### See also
- strategy.closedtrades.profit

## strategy.closedtrades.size()
Returns the direction and the number of contracts traded in the closed trade. If the value is > 0, the market position was long. If the value is < 0, the market position was short.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Example
Converts the value from the currency that the symbol on the chart is traded in (syminfo.currency) to the currency used by the strategy (strategy.account_currency).

### Example
Converts the value from the currency that the symbol on the chart is traded in (syminfo.currency) to the currency used by the strategy (strategy.account_currency).

### See also
Converts the value from the currency that the symbol on the chart is traded in (syminfo.currency) to the currency used by the strategy (strategy.account_currency).

### See also
- strategy.opentrades.size
- strategy.position_size
- strategy.closedtrades
- strategy.opentrades

## strategy.convert_to_account()
Converts the value from the currency that the symbol on the chart is traded in (syminfo.currency) to the currency used by the strategy (strategy.account_currency).

### Syntax
```
value (series int/float) The value to be converted.
```

### Arguments
```
```

value (series int/float) The value to be converted.
```

### Example
Converts the value from the currency used by the strategy (strategy.account_currency) to the currency that the symbol on the chart is traded in (syminfo.currency).

### Example
Converts the value from the currency used by the strategy (strategy.account_currency) to the currency that the symbol on the chart is traded in (syminfo.currency).

### See also
Converts the value from the currency used by the strategy (strategy.account_currency) to the currency that the symbol on the chart is traded in (syminfo.currency).

### See also
- strategy
- strategy.convert_to_symbol

## strategy.convert_to_symbol()
Converts the value from the currency used by the strategy (strategy.account_currency) to the currency that the symbol on the chart is traded in (syminfo.currency).

### Syntax
```
value (series int/float) The value to be converted.
```

### Arguments
```
value (series int/float) The value to be converted.
```

### Example
Calculates the default quantity, in units, of an entry order from strategy.entry or strategy.order if it were to fill at the specified fill_price value. The
```

calculation depends on several strategy properties, including default_qty_type, default_qty_value, currency, and other parameters in the strategy function and their representation in the "Properties" tab of the strategy's settings.

### See also
Calculates the default quantity, in units, of an entry order from strategy.entry or strategy.order if it were to fill at the specified fill_price value. The calculation depends on several strategy properties, including default_qty_type, default_qty_value, currency, and other parameters in the strategy function and their representation in the "Properties" tab of the strategy's settings.

### See also
- strategy
- strategy.convert_to_account

## strategy.default_entry_qty()
Calculates the default quantity, in units, of an entry order from strategy.entry or strategy.order if it were to fill at the specified fill_price value. The calculation depends on several strategy properties, including default_qty_type, default_qty_value, currency, and other parameters in the strategy function and their representation in the "Properties" tab of the strategy's settings.

### Syntax
```
fill_price (series int/float) The fill price for which to calculate the default
order quantity.
```

### Arguments
```
fill_price (series int/float) The fill price for which to calculate the default
order quantity.
```

### Example
This function does not consider open positions simulated by a strategy. For example, if a strategy script has an open position from a long order with a qty of 10 units, using the strategy.entry function to simulate a short order with a qty of 5 will prompt the script to sell 15 units to reverse the position. This function will still return 5 in such a case since it doesn't consider an open trade.

### Remarks
This function does not consider open positions simulated by a strategy. For example, if a strategy script has an open position from a long order with a qty of 10 units, using the strategy.entry function to simulate a short order with a qty of 5 will prompt the script to sell 15 units to reverse the position. This function will still return 5 in such a case since it doesn't consider an open trade.

## strategy.entry()
Creates a new order to open or add to a position. If an unfilled order with the same id exists, a call to this command modifies that order.

### Syntax
```
id (series string) The identifier of the order, which corresponds to an entry ID in the strategy's trades after the order fills. If the strategy opens a new position after filling the order, the order's ID becomes the strategy.position_entry_name value. Strategy commands can reference the order ID to cancel or modify pending orders and generate exit orders for specific open trades. The Strategy Tester and the chart display the order ID unless the command specifies a comment value.
```

### Arguments
```
id (series string) The identifier of the order, which corresponds to an entry ID in the strategy's trades after the order fills. If the strategy opens a new position after filling the order, the order's ID becomes the strategy.position_entry_name value. Strategy commands can reference the order ID to cancel or modify pending orders and generate exit orders for specific open trades. The Strategy Tester and the chart display the order ID unless the command specifies a comment value.
```

### Example
Creates price-based orders to exit from an open position. If unfilled exit orders with the same id exist, calls to this command modify those orders. This command can generate more than one type of exit order, depending on the specified parameters. However, it does not create market orders. To exit from a position with a market order, use strategy.close or strategy.close_all.

### Example
Creates price-based orders to exit from an open position. If unfilled exit orders with the same id exist, calls to this command modify those orders. This command can generate more than one type of exit order, depending on the specified parameters.

However, it does not create market orders. To exit from a position with a market order, use strategy.close or strategy.close_all.

## strategy.exit()
Creates price-based orders to exit from an open position. If unfilled exit orders with the same id exist, calls to this command modify those orders. This command can generate more than one type of exit order, depending on the specified parameters. However, it does not create market orders. To exit from a position with a market order, use strategy.close or strategy.close_all.

### Syntax
```
id (series string) The identifier of the orders, which corresponds to an exit ID in the strategy's trades after an order fills. Strategy commands can reference the order ID to cancel or modify pending exit orders. The Strategy Tester and the chart display the order ID unless the command includes a comment* argument that applies to the filled order.
```

### Arguments
```
id (series string) The identifier of the orders, which corresponds to an exit ID in the strategy's trades after an order fills. Strategy commands can reference the order ID to cancel or modify pending exit orders. The Strategy Tester and the chart display the order ID unless the command includes a comment* argument that applies to the filled order.
```

### Example
A single call to the strategy.exit command can generate exit orders for several entries in an open position, depending on the call's from_entry value. If the call does not include a from_entry argument, it creates exit orders for all open trades, even the ones opened after the call, until the position closes. See this section of our User Manual to learn more.

### Example
A single call to the strategy.exit command can generate exit orders for several entries in an open position, depending on the call's from_entry value. If the call does not include a from_entry argument, it creates exit orders for all open trades, even the ones opened after the call, until the position closes. See this section of our User Manual to learn more.

### Remarks
A single call to the strategy.exit command can generate exit orders for several entries in an open position, depending on the call's from_entry value. If the call does not include a from_entry argument, it creates exit orders for all open trades, even the ones opened after the call, until the position closes. See this section of our User Manual to learn more.

## strategy.opentrades.commission()
Returns the sum of entry and exit fees paid in the open trade, expressed in strategy.account_currency.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Example
Returns the bar_index of the open trade's entry.

### See also
Returns the bar_index of the open trade's entry.

### See also
- strategy
- strategy.closedtrades.commission

## strategy.opentrades.entry_bar_index()
Returns the bar_index of the open trade's entry.

### Syntax

```
trade_num (series int) The trade number of the open trade. The number of the first
trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first
trade is zero.
```

### Example
Returns the comment message of the open trade's entry, or
na if there is no entry with this trade_num.

### See also
Returns the comment message of the open trade's entry, or
na if there is no entry with this trade_num.

### See also
- strategy.closedtrades.entry_bar_index
- strategy.closedtrades.exit_bar_index

## strategy.opentrades.entry_comment()
Returns the comment message of the open trade's entry, or
na if there is no entry with this trade_num.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first
trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first
trade is zero.
```

### Example

Returns the id of the open trade's entry.

### See also
Returns the id of the open trade's entry.

### See also
- strategy
- strategy.entry
- strategy.opentrades

## strategy.opentrades.entry_id()
Returns the id of the open trade's entry.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Example
Returns the id of the open trade's entry.

### Returns
```
Returns the id of the open trade's entry.
```

### Remarks
The function returns na if trade_num is not in the range: 0 to strategy.opentrades-1.

### See also
Returns the price of the open trade's entry.

### See also
- strategy.opentrades.entry_bar_index
- strategy.opentrades.entry_price
- strategy.opentrades.entry_time

## strategy.opentrades.entry_price()
Returns the price of the open trade's entry.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Example
Returns the UNIX time of the open trade's entry, expressed in milliseconds.

### Example
Returns the UNIX time of the open trade's entry, expressed in milliseconds.

### See also
Returns the UNIX time of the open trade's entry, expressed in milliseconds.

### See also
- strategy.closedtrades.exit_price

## strategy.opentrades.entry_time()
Returns the UNIX time of the open trade's entry, expressed in milliseconds.

### Syntax

```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Example
Returns the maximum drawdown of the open trade, i.e., the maximum possible loss during the trade, expressed in strategy.account_currency.

### See also
Returns the maximum drawdown of the open trade, i.e., the maximum possible loss during the trade, expressed in strategy.account_currency.

### See also
- strategy.closedtrades.entry_time
- strategy.closedtrades.exit_time

## strategy.opentrades.max_drawdown()
Returns the maximum drawdown of the open trade, i.e., the maximum possible loss during the trade, expressed in strategy.account_currency.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Example

The function returns na if trade_num is not in the range: 0 to strategy.closedtrades - 1.

### Example
The function returns na if trade_num is not in the range: 0 to strategy.closedtrades - 1.

### Remarks
The function returns na if trade_num is not in the range: 0 to strategy.closedtrades - 1.

### See also
Returns the maximum drawdown of the open trade, i.e., the maximum possible loss during the trade, expressed as a percentage and calculated by formula: Lowest Value During Trade / (Entry Price x Quantity) * 100.

### See also
- strategy.closedtrades.max_drawdown
- strategy.max_drawdown

## strategy.opentrades.max_drawdown_percent()
Returns the maximum drawdown of the open trade, i.e., the maximum possible loss during the trade, expressed as a percentage and calculated by formula: Lowest Value During Trade / (Entry Price x Quantity) * 100.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### See also
Returns the maximum run up of the open trade, i.e., the maximum possible profit during the trade, expressed in strategy.account_currency.

### See also
- strategy.opentrades.max_drawdown
- strategy.max_drawdown

## strategy.opentrades.max_runup()
Returns the maximum run up of the open trade, i.e., the maximum possible profit during the trade, expressed in strategy.account_currency.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Example
Returns the maximum run-up of the open trade, i.e., the maximum possible profit during the trade, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### Example
Returns the maximum run-up of the open trade, i.e., the maximum possible profit during the trade, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### See also
Returns the maximum run-up of the open trade, i.e., the maximum possible profit during the trade, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### See also
- strategy.closedtrades.max_runup
- strategy.max_drawdown

## strategy.opentrades.max_runup_percent()
Returns the maximum run-up of the open trade, i.e., the maximum possible profit during the trade, expressed as a percentage and calculated by formula: Highest Value During Trade / (Entry Price x Quantity) * 100.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### See also
Returns the profit/loss of the open trade, expressed in strategy.account_currency. Losses are expressed as negative values.

### See also
- strategy.opentrades.max_runup
- strategy.max_runup

## strategy.opentrades.profit()
Returns the profit/loss of the open trade, expressed in strategy.account_currency. Losses are expressed as negative values.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

```

### Example
Returns the profit/loss of the open trade, expressed as a percentage. Losses are expressed as negative values.

### Example
Returns the profit/loss of the open trade, expressed as a percentage. Losses are expressed as negative values.

### See also
Returns the profit/loss of the open trade, expressed as a percentage. Losses are expressed as negative values.

### See also
- strategy.closedtrades.profit
- strategy.openprofit
- strategy.netprofit
- strategy.grossprofit

## strategy.opentrades.profit_percent()
Returns the profit/loss of the open trade, expressed as a percentage. Losses are expressed as negative values.

### Syntax
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the closed trade. The number of the first trade is zero.
```

### See also
Returns the direction and the number of contracts traded in the open trade. If the value is > 0, the market position was long. If the value is < 0, the market position was short.
```

### See also
- strategy.opentrades.profit

## strategy.opentrades.size()
Returns the direction and the number of contracts traded in the open trade. If the value is > 0, the market position was long. If the value is < 0, the market position was short.

### Syntax
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Arguments
```
trade_num (series int) The trade number of the open trade. The number of the first trade is zero.
```

### Example
Creates a new order to open, add to, or exit from a position. If an unfilled order with the same id exists, a call to this command modifies that order.

### Example
Creates a new order to open, add to, or exit from a position. If an unfilled order with the same id exists, a call to this command modifies that order.

### See also
Creates a new order to open, add to, or exit from a position. If an unfilled order with the same id exists, a call to this command modifies that order.

### See also
- strategy.closedtrades.size
- strategy.position_size
- strategy.opentrades
- strategy.closedtrades

## strategy.order()
Creates a new order to open, add to, or exit from a position. If an unfilled order with the same id exists, a call to this command modifies that order.

### Syntax
```
id (series string) The identifier of the order, which corresponds to an entry or exit ID in the strategy's trades after the order fills. If the strategy opens a new position after filling the order, the order's ID becomes the strategy.position_entry_name value. Strategy commands can reference the order ID to cancel or modify pending orders and generate exit orders for specific open trades. The Strategy Tester and the chart display the order ID unless the command specifies a comment value.
```

### Arguments
```
id (series string) The identifier of the order, which corresponds to an entry or exit ID in the strategy's trades after the order fills. If the strategy opens a new position after filling the order, the order's ID becomes the strategy.position_entry_name value. Strategy commands can reference the order ID to cancel or modify pending orders and generate exit orders for specific open trades. The Strategy Tester and the chart display the order ID unless the command specifies a comment value.
```

### Example
This function can be used to specify in which market direction the strategy.entry function is allowed to open positions.

### Example
This function can be used to specify in which market direction the strategy.entry function is allowed to open positions.

## strategy.risk.allow_entry_in()
This function can be used to specify in which market direction the strategy.entry function is allowed to open positions.

### Syntax
```
```

```
- value (simple string) The allowed direction. Possible values :
strategy.direction.all, strategy.direction.long, strategy.direction.short
```

### Arguments
```
- value (simple string) The allowed direction. Possible values :
strategy.direction.all, strategy.direction.long, strategy.direction.short
```

### Example
The purpose of this rule is to cancel all pending orders, close all open positions and stop placing orders after a specified number of consecutive days with losses. The rule affects the whole strategy.

## strategy.risk.max_cons_loss_days()
The purpose of this rule is to cancel all pending orders, close all open positions and stop placing orders after a specified number of consecutive days with losses. The rule affects the whole strategy.

### Syntax
```
count (simple int) A required parameter. The allowed number of consecutive days
with losses.
```

### Arguments
```
count (simple int) A required parameter. The allowed number of consecutive days
with losses.
```

### Example
The purpose of this rule is to determine maximum drawdown. The rule affects the whole strategy. Once the maximum drawdown value is reached, all pending orders are cancelled, all open positions are closed and no new orders can be placed.

## strategy.risk.max_drawdown()

The purpose of this rule is to determine maximum drawdown. The rule affects the whole strategy. Once the maximum drawdown value is reached, all pending orders are cancelled, all open positions are closed and no new orders can be placed.

### Syntax
```
value (simple int/float) A required parameter. The maximum drawdown value. It is specified either in money (base currency), or in percentage of maximum equity. For % of equity the range of allowed values is from 0 to 100.
```

### Arguments
```
value (simple int/float) A required parameter. The maximum drawdown value. It is specified either in money (base currency), or in percentage of maximum equity. For % of equity the range of allowed values is from 0 to 100.
```

### Example
The purpose of this rule is to determine maximum number of filled orders per 1 day (per 1 bar, if chart resolution is higher than 1 day). The rule affects the whole strategy. Once the maximum number of filled orders is reached, all pending orders are cancelled, all open positions are closed and no new orders can be placed till the end of the current trading session.

### Example
The purpose of this rule is to determine maximum number of filled orders per 1 day (per 1 bar, if chart resolution is higher than 1 day). The rule affects the whole strategy. Once the maximum number of filled orders is reached, all pending orders are cancelled, all open positions are closed and no new orders can be placed till the end of the current trading session.

## strategy.risk.max_intraday_filled_orders()
The purpose of this rule is to determine maximum number of filled orders per 1 day (per 1 bar, if chart resolution is higher than 1 day). The rule affects the whole strategy. Once the maximum number of filled orders is reached, all pending orders are cancelled, all open positions are closed and no new orders can be placed till the end of the current trading session.

### Syntax
```
```

```
count (simple int) A required parameter. The maximum number of filled orders per 1 day.
```

### Arguments
```
count (simple int) A required parameter. The maximum number of filled orders per 1 day.
```

### Example
The maximum loss value allowed during a day. It is specified either in money (base currency), or in percentage of maximum intraday equity (0 -100).

## strategy.risk.max_intraday_loss()
The maximum loss value allowed during a day. It is specified either in money (base currency), or in percentage of maximum intraday equity (0 -100).

### Syntax
```
value (simple int/float) A required parameter. The maximum loss value. It is specified either in money (base currency), or in percentage of maximum intraday equity. For % of equity the range of allowed values is from 0 to 100.
```

### Arguments
```
value (simple int/float) A required parameter. The maximum loss value. It is specified either in money (base currency), or in percentage of maximum intraday equity. For % of equity the range of allowed values is from 0 to 100.
```

### Example
The purpose of this rule is to determine maximum size of a market position. The rule affects the following function: strategy.entry. The 'entry' quantity can be reduced (if needed) to such number of contracts/shares/lots/units, so the total position size doesn't exceed the value specified in 'strategy.risk.max_position_size'. If minimum possible quantity still violates the rule, the order will not be placed.

### Example
The purpose of this rule is to determine maximum size of a market position. The rule affects the following function: strategy.entry. The 'entry' quantity can be reduced (if needed) to such number of contracts/shares/lots/units, so the total position size doesn't exceed the value specified in 'strategy.risk.max_position_size'. If minimum possible quantity still violates the rule, the order will not be placed.

### See also
The purpose of this rule is to determine maximum size of a market position. The rule affects the following function: strategy.entry. The 'entry' quantity can be reduced (if needed) to such number of contracts/shares/lots/units, so the total position size doesn't exceed the value specified in 'strategy.risk.max_position_size'. If minimum possible quantity still violates the rule, the order will not be placed.

### See also
- strategy
- strategy.percent_of_equity
- strategy.cash

## strategy.risk.max_position_size()
The purpose of this rule is to determine maximum size of a market position. The rule affects the following function: strategy.entry. The 'entry' quantity can be reduced (if needed) to such number of contracts/shares/lots/units, so the total position size doesn't exceed the value specified in 'strategy.risk.max_position_size'. If minimum possible quantity still violates the rule, the order will not be placed.

### Syntax
```
contracts (simple int/float) A required parameter. Maximum number of
contracts/shares/lots/units in a position.
```

### Arguments
```
contracts (simple int/float) A required parameter. Maximum number of
contracts/shares/lots/units in a position.
```

### Example

Casts na to string

## string()
Casts na to string

### Syntax & Overloads
x (const string) The value to convert to the specified type, usually na.

### Arguments
```
x (const string) The value to convert to the specified type, usually na.
```

### Returns
```
The value of the argument after casting to string.
```

### See also
Returns exchange prefix of the symbol, e.g. "NASDAQ".

### See also
- float
- int
- bool
- color
- line
- label

## syminfo.prefix()
Returns exchange prefix of the symbol, e.g. "NASDAQ".

### Syntax & Overloads
symbol (simple string) Symbol. Note that the symbol should be passed with a prefix. For example: "NASDAQ:AAPL" instead of "AAPL".

### Arguments

```
- symbol (simple string) Symbol. Note that the symbol should be passed with a
prefix. For example : "NASDAQ:AAPL" instead of "AAPL".
```

### Example
Returns exchange prefix of the symbol, e.g. "NASDAQ".

### Returns
```
Returns exchange prefix of the symbol, e.g. "NASDAQ".
```

### Remarks
The result of the function is used in the ticker.new/ticker.modify and
request.security.

### See also
Returns symbol name without exchange prefix, e.g. "AAPL".

### See also
- syminfo.tickerid
- syminfo.ticker
- syminfo.prefix
- syminfo.ticker
- ticker.new

## syminfo.ticker()
Returns symbol name without exchange prefix, e.g. "AAPL".

### Syntax & Overloads
symbol (simple string) Symbol. Note that the symbol should be passed with a prefix.
For example: "NASDAQ:AAPL" instead of "AAPL".

### Arguments
```
- symbol (simple string) Symbol. Note that the symbol should be passed with a
prefix. For example : "NASDAQ:AAPL" instead of "AAPL".
```

### Example
Returns symbol name without exchange prefix, e.g. "AAPL".

### Returns
```
Returns symbol name without exchange prefix, e.g. "AAPL".
```

### Remarks
The result of the function is used in the ticker.new/ticker.modify and request.security.

### See also
Arnaud Legoux Moving Average. It uses Gaussian distribution as weights for moving average.

### See also
- syminfo.tickerid
- syminfo.ticker
- syminfo.prefix
- syminfo.prefix
- ticker.new

## ta.alma()
Arnaud Legoux Moving Average. It uses Gaussian distribution as weights for moving average.

### Syntax & Overloads
series (series int/float) Series of values to process.

### Arguments
```
series (series int/float) Series of values to process.
```

### Example
Arnaud Legoux Moving Average.

### Returns
```
Arnaud Legoux Moving Average.
```

### Remarks
na values in the source series are included in calculations and will produce an na result.

### See also
Function atr (average true range) returns the RMA of true range. True range is max(high - low, abs(high - close[1]), abs(low - close[1])).

### See also
- ta.sma
- ta.ema
- ta.rma
- ta.wma
- ta.vwma
- ta.swma

## ta.atr()
Function atr (average true range) returns the RMA of true range. True range is max(high - low, abs(high - close[1]), abs(low - close[1])).

### Syntax
```
length (simple int) Length (number of bars back).
```

### Arguments
```
length (simple int) Length (number of bars back).
```

### Example
Average true range.

### Returns
```
```

Average true range.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Counts the number of bars since the last time the condition was true.

### See also
- ta.tr
- ta.rma

## ta.barssince()
Counts the number of bars since the last time the condition was true.

### Syntax
```
condition (series bool) The condition to check for.
```

### Arguments
```
condition (series bool) The condition to check for.
```

### Example
Number of bars since condition was true.

### Returns
```
Number of bars since condition was true.
```

### Remarks
If the condition has never been met prior to the current bar, the function returns na.
```

### See also
Bollinger Bands. A Bollinger Band is a technical analysis tool defined by a set of lines plotted two standard deviations (positively and negatively) away from a simple moving average (SMA) of the security's price, but can be adjusted to user preferences.

### See also
- ta.lowestbars
- ta.highestbars
- ta.valuewhen
- ta.highest
- ta.lowest

## ta.bb()
Bollinger Bands. A Bollinger Band is a technical analysis tool defined by a set of lines plotted two standard deviations (positively and negatively) away from a simple moving average (SMA) of the security's price, but can be adjusted to user preferences.

### Syntax
```
series (series int/float) Series of values to process.
```

### Arguments
```
series (series int/float) Series of values to process.
```

### Example
Bollinger Bands.

### Returns
```
Bollinger Bands.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Bollinger Bands Width. The Bollinger Band Width is the difference between the upper and the lower Bollinger Bands divided by the middle band.

### See also
- ta.sma
- ta.stdev
- ta.kc

## ta.bbw()
Bollinger Bands Width. The Bollinger Band Width is the difference between the upper and the lower Bollinger Bands divided by the middle band.

### Syntax
```
series (series int/float) Series of values to process.
```

### Arguments
```
series (series int/float) Series of values to process.
```

### Example
Bollinger Bands Width.

### Returns
```
Bollinger Bands Width.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
The CCI (commodity channel index) is calculated as the difference between the typical price of a commodity and its simple moving average, divided by the mean

absolute deviation of the typical price. The index is scaled by an inverse factor of 0.015 to provide more readable numbers.

### See also
- ta.bb
- ta.sma
- ta.stdev

## ta.cci()
The CCI (commodity channel index) is calculated as the difference between the typical price of a commodity and its simple moving average, divided by the mean absolute deviation of the typical price. The index is scaled by an inverse factor of 0.015 to provide more readable numbers.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
Commodity channel index of source for length bars back.
```

### Remarks
na values in the source series are ignored.

## ta.change()
Compares the current source value to its value length bars ago and returns the difference.

### Syntax & Overloads
source (series int) Source series.

### Arguments
```
source (series int) Source series.
```

### Example
The difference between the values when they are numerical. When a 'bool' source is used, returns true when the current source is different from the previous source.

### Returns
```
The difference between the values when they are numerical. When a 'bool' source is
used, returns true when the current source is different from the previous source.
```

### Remarks
na values in the source series are included in calculations and will produce an na result.

### See also
Chande Momentum Oscillator. Calculates the difference between the sum of recent gains and the sum of recent losses and then divides the result by the sum of all price movement over the same period.

### See also
- ta.mom
- ta.cross

## ta.cmo()
Chande Momentum Oscillator. Calculates the difference between the sum of recent gains and the sum of recent losses and then divides the result by the sum of all price movement over the same period.

### Syntax
```
series (series int/float) Series of values to process.
```

### Arguments
```
series (series int/float) Series of values to process.
```

### Example
Chande Momentum Oscillator.

### Returns
```
Chande Momentum Oscillator.
```

### Remarks
na values in the source series are ignored.

### See also
The cog (center of gravity) is an indicator based on statistics and the Fibonacci golden ratio.

### See also
- ta.rsi
- ta.stoch
- math.sum

## ta.cog()
The cog (center of gravity) is an indicator based on statistics and the Fibonacci golden ratio.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Center of Gravity.

### Returns
```
Center of Gravity.
```

### Remarks
na values in the source series are ignored.

### See also
Correlation coefficient. Describes the degree to which two series tend to deviate from their ta.sma values.

### See also
- ta.stoch

## ta.correlation()
Correlation coefficient. Describes the degree to which two series tend to deviate from their ta.sma values.

### Syntax
```
source1 (series int/float) Source series.
```

### Arguments
```
source1 (series int/float) Source series.
```

### Returns
```
Correlation coefficient.
```

### Remarks
na values in the source series are ignored; the function calculates on the length

quantity of non-na values.

### See also
source1 (series int/float) First data series.

### See also
- request.security

## ta.cross()
source1 (series int/float) First data series.

### Syntax
```
source1 (series int/float) First data series.
```

### Arguments
```
source1 (series int/float) First data series.
```

### Returns
```
true if two series have crossed each other, otherwise false.
```

### See also
The source1-series is defined as having crossed over source2-series if, on the current bar, the value of source1 is greater than the value of source2, and on the previous bar, the value of source1 was less than or equal to the value of source2.

### See also
- ta.change

## ta.crossover()
The source1-series is defined as having crossed over source2-series if, on the current bar, the value of source1 is greater than the value of source2, and on the previous bar, the value of source1 was less than or equal to the value of source2.

### Syntax
```
source1 (series int/float) First data series.
```

### Arguments
```
source1 (series int/float) First data series.
```

### Returns
```
true if source1 crossed over source2 otherwise false.
```

## ta.crossunder()
The source1-series is defined as having crossed under source2-series if, on the current bar, the value of source1 is less than the value of source2, and on the previous bar, the value of source1 was greater than or equal to the value of source2.

### Syntax
```
source1 (series int/float) First data series.
```

### Arguments
```
source1 (series int/float) First data series.
```

### Returns
```
true if source1 crossed under source2 otherwise false.
```

## ta.cum()
Cumulative (total) sum of source. In other words it's a sum of all elements of source.

### Syntax
```
source (series int/float) Source used for the calculation.
```

### Arguments
```
source (series int/float) Source used for the calculation.
```

### Returns
```
Total sum series.
```

### See also
Measure of difference between the series and it's ta.sma

### See also
- math.sum

## ta.dev()
Measure of difference between the series and it's ta.sma

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example

Deviation of source for length bars back.

### Returns
```
Deviation of source for length bars back.
```

### Remarks
na values in the source series are ignored.

### See also
The dmi function returns the directional movement index.

### See also
- ta.variance
- ta.stdev

## ta.dmi()
The dmi function returns the directional movement index.

### Syntax
```
diLength (simple int) DI Period.
```

### Arguments
```
diLength (simple int) DI Period.
```

### Example
Tuple of three DMI series: Positive Directional Movement (+DI), Negative Directional Movement (-DI) and Average Directional Movement Index (ADX).

### Returns
```
- Tuple of three DMI series : Positive Directional Movement (+DI), Negative
Directional Movement (-DI) and Average Directional Movement Index (ADX).
```

### See also
The ema function returns the exponentially weighted moving average. In ema weighting factors decrease exponentially. It calculates by using a formula: EMA = alpha * source + (1 - alpha) * EMA[1], where alpha = 2 / (length + 1).

### See also
- ta.rsi
- ta.tsi
- ta.mfi

## ta.ema()
The ema function returns the exponentially weighted moving average. In ema weighting factors decrease exponentially. It calculates by using a formula: EMA = alpha * source + (1 - alpha) * EMA[1], where alpha = 2 / (length + 1).

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Exponential moving average of source with alpha = 2 / (length + 1).

### Returns
```
Exponential moving average of source with alpha = 2 / (length + 1).
```

### Remarks
Please note that using this variable/function can cause indicator repainting.

### See also
Test if the source series is now falling for length bars long.

### See also
- ta.sma
- ta.rma
- ta.wma
- ta.vwma
- ta.swma
- ta.alma

## ta.falling()
Test if the source series is now falling for length bars long.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
true if current source value is less than any previous source value for length bars back, false otherwise.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Highest value for a given number of bars back.

### See also
- ta.rising

## ta.highest()
Highest value for a given number of bars back.

### Syntax & Overloads
length (series int) Number of bars (length).

### Arguments
```
length (series int) Number of bars (length).
```

### Returns
```
Highest value in the series.
```

### Remarks
Two args version: source is a series and length is the number of bars back.

### See also
Highest value offset for a given number of bars back.

### See also
- ta.lowest
- ta.lowestbars
- ta.highestbars
- ta.valuewhen
- ta.barssince

## ta.highestbars()
Highest value offset for a given number of bars back.

### Syntax & Overloads
length (series int) Number of bars (length).

### Arguments
```
length (series int) Number of bars (length).
```

### Returns
```
Offset to the highest bar.
```

### Remarks
Two args version: source is a series and length is the number of bars back.

### See also
The hma function returns the Hull Moving Average.

### See also
- ta.lowest
- ta.highest
- ta.lowestbars
- ta.barssince
- ta.valuewhen

## ta.hma()
The hma function returns the Hull Moving Average.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Hull moving average of 'source' for 'length' bars back.

### Returns
```
Hull moving average of 'source' for 'length' bars back.
```

### Remarks
na values in the source series are ignored.

### See also
Keltner Channels. Keltner channel is a technical analysis indicator showing a central moving average line plus channel lines at a distance above and below.

### See also
- ta.ema
- ta.rma
- ta.wma
- ta.vwma
- ta.sma

## ta.kc()
Keltner Channels. Keltner channel is a technical analysis indicator showing a central moving average line plus channel lines at a distance above and below.

### Syntax & Overloads
series (series int/float) Series of values to process.

### Arguments
```
series (series int/float) Series of values to process.
```

### Example
Keltner Channels.

### Returns
```
Keltner Channels.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Keltner Channels Width. The Keltner Channels Width is the difference between the upper and the lower Keltner Channels divided by the middle channel.

### See also
- ta.ema
- ta.atr
- ta.bb

## ta.kcw()
Keltner Channels Width. The Keltner Channels Width is the difference between the upper and the lower Keltner Channels divided by the middle channel.

### Syntax & Overloads
series (series int/float) Series of values to process.

### Arguments
```
series (series int/float) Series of values to process.
```

### Example
Keltner Channels Width.

### Returns
```
Keltner Channels Width.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Linear regression curve. A line that best fits the prices specified over a user-defined time period. It is calculated using the least squares method. The result of this function is calculated using the formula: linreg = intercept + slope * (length - 1 - offset), where intercept and slope are the values calculated with the least squares method on source series.

### See also
- ta.kc
- ta.ema
- ta.atr
- ta.bb

## ta.linreg()
Linear regression curve. A line that best fits the prices specified over a user-defined time period. It is calculated using the least squares method. The result of this function is calculated using the formula: linreg = intercept + slope * (length - 1 - offset), where intercept and slope are the values calculated with the least squares method on source series.

### Syntax
```
source (series int/float) Source series.
```

### Arguments
```
source (series int/float) Source series.
```

### Returns
```
Linear regression curve.
```

### Remarks
na values in the source series are included in calculations and will produce an na result.

## ta.lowest()
Lowest value for a given number of bars back.

### Syntax & Overloads
length (series int) Number of bars (length).

### Arguments
```
length (series int) Number of bars (length).
```

### Returns
```
Lowest value in the series.
```

### Remarks
Two args version: source is a series and length is the number of bars back.

### See also
Lowest value offset for a given number of bars back.

### See also
- ta.highest
- ta.lowestbars
- ta.highestbars
- ta.valuewhen
- ta.barssince

## ta.lowestbars()
Lowest value offset for a given number of bars back.

### Syntax & Overloads
length (series int) Number of bars back.

### Arguments
```
length (series int) Number of bars back.
```

### Returns
```
Offset to the lowest bar.
```

### Remarks
Two args version: source is a series and length is the number of bars back.

### See also
MACD (moving average convergence/divergence). It is supposed to reveal changes in the strength, direction, momentum, and duration of a trend in a stock's price.

### See also
- ta.lowest
- ta.highest
- ta.highestbars
- ta.barssince
- ta.valuewhen

## ta.macd()
MACD (moving average convergence/divergence). It is supposed to reveal changes in the strength, direction, momentum, and duration of a trend in a stock's price.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
If you need only one value, use placeholders '_' like this:

### Example
Tuple of three MACD series: MACD line, signal line and histogram line.

### Returns
```
- Tuple of three MACD series : MACD line, signal line and histogram line.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Returns the all-time high value of source from the beginning of the chart up to the current bar.

### See also
- ta.sma
- ta.ema

## ta.max()
Returns the all-time high value of source from the beginning of the chart up to the current bar.

### Syntax
```
source (series int/float) Source used for the calculation.
```

### Arguments
```
source (series int/float) Source used for the calculation.
```

### Remarks
na occurrences of source are ignored.

## ta.median()
Returns the median of the series.

### Syntax & Overloads
source (series int) Series of values to process.

### Arguments
```
source (series int) Series of values to process.
```

### Returns
```
The median of the series.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

## ta.mfi()
Money Flow Index. The Money Flow Index (MFI) is a technical oscillator that uses price and volume for identifying overbought or oversold conditions in an asset.

### Syntax
```
series (series int/float) Series of values to process.
```

### Arguments
```
series (series int/float) Series of values to process.
```

### Example
Money Flow Index.

### Returns
```
Money Flow Index.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Returns the all-time low value of source from the beginning of the chart up to the current bar.

### See also
- ta.rsi
- math.sum

## ta.min()
Returns the all-time low value of source from the beginning of the chart up to the current bar.

### Syntax
```
source (series int/float) Source used for the calculation.
```

### Arguments
```
source (series int/float) Source used for the calculation.
```

### Remarks
na occurrences of source are ignored.

## ta.mode()
Returns the mode of the series. If there are several values with the same frequency, it returns the smallest value.

### Syntax & Overloads
source (series int) Series of values to process.

### Arguments
```
source (series int) Series of values to process.
```

### Returns
```
The most frequently occurring value from the source. If none exists, returns the
smallest value instead.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

## ta.mom()
Momentum of source price and source price length bars ago. This is simply a difference: source - source[length].

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
Momentum of source price and source price length bars ago.
```

### Remarks
na values in the source series are included in calculations and will produce an na result.

### See also
Calculates percentile using method of linear interpolation between the two nearest ranks.

### See also
- ta.change

## ta.percentile_linear_interpolation()
Calculates percentile using method of linear interpolation between the two nearest ranks.

### Syntax
```
source (series int/float) Series of values to process (source).
```

### Arguments
```
source (series int/float) Series of values to process (source).
```

### Returns
```
P-th percentile of source series for length bars back.
```

### Remarks
Note that a percentile calculated using this method will NOT always be a member of the input data set.

### See also
Calculates percentile using method of Nearest Rank.

### See also
- ta.percentile_nearest_rank

## ta.percentile_nearest_rank()
Calculates percentile using method of Nearest Rank.

### Syntax
```
source (series int/float) Series of values to process (source).
```

```

### Arguments
```
source (series int/float) Series of values to process (source).
```

### Returns
```
P-th percentile of source series for length bars back.
```

### Remarks
Using the Nearest Rank method on lengths less than 100 bars back can result in the same number being used for more than one percentile.

### See also
Percent rank is the percents of how many previous values was less than or equal to the current value of given series.

### See also
- ta.percentile_linear_interpolation
-

## ta.percentrank()
Percent rank is the percents of how many previous values was less than or equal to the current value of given series.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```

```
Percent rank of source for length bars back.
```

### Remarks
na values in the source series are included in calculations and will produce an na result.

## ta.pivot_point_levels()
Calculates the pivot point levels using the specified type and anchor.

### Syntax
```
- type (series string) The type of pivot point levels. Possible values : "Traditional", "Fibonacci", "Woodie", "Classic", "DM", "Camarilla".
```

### Arguments
```
- type (series string) The type of pivot point levels. Possible values : "Traditional", "Fibonacci", "Woodie", "Classic", "DM", "Camarilla".
```

### Example
An array<float> with numerical values representing 11 pivot point levels: [P, R1, S1, R2, S2, R3, S3, R4, S4, R5, S5]. Levels absent from the specified type return na values (e.g., "DM" only calculates P, R1, and S1).

### Returns
```
- An array<float> with numerical values representing 11 pivot point levels : [P, R1, S1, R2, S2, R3, S3, R4, S4, R5, S5]. Levels absent from the specified type return na values (e.g., "DM" only calculates P, R1, and S1).
```

### Remarks
The developing parameter cannot be true when type is set to "Woodie", because the Woodie calculation for a period depends on that period's open, which means that the pivot value is either available or unavailable, but never developing. If used together, the indicator will return a runtime error.

## ta.pivothigh()
This function returns price of the pivot high point. It returns 'NaN', if there was no pivot high point.

### Syntax & Overloads
leftbars (series int/float) Left strength.

### Arguments
```
leftbars (series int/float) Left strength.
```

### Example
Price of the point or 'NaN'.

### Returns
```
Price of the point or 'NaN'.
```

### Remarks
If parameters 'leftbars' or 'rightbars' are series you should use max_bars_back function for the 'source' variable.

## ta.pivotlow()
This function returns price of the pivot low point. It returns 'NaN', if there was no pivot low point.

### Syntax & Overloads
leftbars (series int/float) Left strength.

### Arguments
```
leftbars (series int/float) Left strength.
```

### Example
Price of the point or 'NaN'.

### Returns
```
Price of the point or 'NaN'.
```

### Remarks
If parameters 'leftbars' or 'rightbars' are series you should use max_bars_back function for the 'source' variable.

## ta.range()
Returns the difference between the min and max values in a series.

### Syntax & Overloads
source (series int) Series of values to process.

### Arguments
```
source (series int) Series of values to process.
```

### Returns
```
The difference between the min and max values in the series.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

## ta.rci()
Calculates the Rank Correlation Index (RCI), which measures the directional consistency of price movements. It evaluates the monotonic relationship between a

source series and the bar index over length bars using Spearman's rank correlation coefficient. The resulting value is scaled to a range of -100 to 100, where 100 indicates the source consistently increased over the period, and -100 indicates it consistently decreased. Values between -100 and 100 reflect varying degrees of upward or downward consistency.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
The Rank Correlation Index, a value between -100 to 100.
```

### See also
Test if the source series is now rising for length bars long.

### See also

## ta.rising()
Test if the source series is now rising for length bars long.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
true if current source is greater than any previous source for length bars back,
false otherwise.
```

### Remarks
na values in the source series are ignored.

### See also
Moving average used in RSI. It is the exponentially weighted moving average with
alpha = 1 / length.

### See also
- ta.falling

## ta.rma()
Moving average used in RSI. It is the exponentially weighted moving average with
alpha = 1 / length.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Exponential moving average of source with alpha = 1 / length.

### Returns
```
Exponential moving average of source with alpha = 1 / length.
```

### Remarks

na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Calculates the percentage of change (rate of change) between the current value of source and its value length bars ago.

### See also
- ta.sma
- ta.ema
- ta.wma
- ta.vwma
- ta.swma
- ta.alma
- ta.rsi

## ta.roc()
Calculates the percentage of change (rate of change) between the current value of source and its value length bars ago.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
The rate of change of source for length bars back.
```

### Remarks
na values in the source series are included in calculations and will produce an na result.

## ta.rsi()
Relative strength index. It is calculated using the ta.rma() of upward and downward changes of source over the last length bars.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Relative strength index.

### Returns
```
Relative strength index.
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

### See also
Parabolic SAR (parabolic stop and reverse) is a method devised by J. Welles Wilder, Jr., to find potential reversals in the market price direction of traded goods.

### See also
- ta.rma

## ta.sar()
Parabolic SAR (parabolic stop and reverse) is a method devised by J. Welles Wilder, Jr., to find potential reversals in the market price direction of traded goods.

### Syntax

```
start (simple int/float) Start.
```

### Arguments
```
start (simple int/float) Start.
```

### Example
Parabolic SAR.

### Returns
```
Parabolic SAR.
```

## ta.sma()
The sma function returns the moving average, that is the sum of last y values of x, divided by y.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Simple moving average of source for length bars back.

### Returns
```
Simple moving average of source for length bars back.
```

### Remarks
na values in the source series are ignored.

### See also
source (series int/float) Series of values to process.

### See also
- ta.ema
- ta.rma
- ta.wma
- ta.vwma
- ta.swma
- ta.alma

## ta.stdev()
source (series int/float) Series of values to process.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Standard deviation.

### Returns
```
Standard deviation.
```

### Remarks
If biased is true, function will calculate using a biased estimate of the entire population, if false - unbiased estimate of a sample.

### See also
Stochastic. It is calculated by a formula: 100 * (close - lowest(low, length)) / (highest(high, length) - lowest(low, length)).

### See also
- ta.dev
- ta.variance

## ta.stoch()
Stochastic. It is calculated by a formula: 100 * (close - lowest(low, length)) / (highest(high, length) - lowest(low, length)).

### Syntax
```
source (series int/float) Source series.
```

### Arguments
```
source (series int/float) Source series.
```

### Returns
```
Stochastic.
```

### Remarks
na values in the source series are ignored.

### See also
The Supertrend Indicator. The Supertrend is a trend following indicator.

### See also
- ta.cog

## ta.supertrend()

The Supertrend Indicator. The Supertrend is a trend following indicator.

### Syntax
```
factor (series int/float) The multiplier by which the ATR will get multiplied.
```

### Arguments
```
factor (series int/float) The multiplier by which the ATR will get multiplied.
```

### Example
Tuple of two supertrend series: supertrend line and direction of trend. Possible values are 1 (down direction) and -1 (up direction).

### Returns
```
- Tuple of two supertrend series : supertrend line and direction of trend. Possible values are 1 (down direction) and -1 (up direction).
```

### See also
Symmetrically weighted moving average with fixed length: 4. Weights: [1/6, 2/6, 2/6, 1/6].

### See also
- ta.macd

## ta.swma()
Symmetrically weighted moving average with fixed length: 4. Weights: [1/6, 2/6, 2/6, 1/6].

### Syntax
```
source (series int/float) Source series.
```

### Arguments

```
source (series int/float) Source series.
```

### Example
Symmetrically weighted moving average.

### Returns
```
Symmetrically weighted moving average.
```

### Remarks
na values in the source series are included in calculations and will produce an na result.

### See also
Calculates the current bar's true range. Unlike a bar's actual range (high - low), true range accounts for potential gaps by taking the maximum of the current bar's actual range and the absolute distances from the previous bar's close to the current bar's high and low. The formula is: math.max(high - low, math.abs(high - close[1]), math.abs(low - close[1]))

### See also
- ta.sma
- ta.ema
- ta.rma
- ta.wma
- ta.vwma
- ta.alma

## ta.tr()
Calculates the current bar's true range. Unlike a bar's actual range (high - low), true range accounts for potential gaps by taking the maximum of the current bar's actual range and the absolute distances from the previous bar's close to the current bar's high and low. The formula is: math.max(high - low, math.abs(high - close[1]), math.abs(low - close[1]))

### Syntax
```
handle_na (simple bool) Defines how the function calculates the result when the
```

previous bar's close is na. If true, the function returns the bar's high - low
value. If false, it returns na.
```

### Arguments
```
handle_na (simple bool) Defines how the function calculates the result when the
previous bar's close is na. If true, the function returns the bar's high - low
value. If false, it returns na.
```

### Returns
```
True range. It is math.max(high - low, math.abs(high - close[1]), math.abs(low -
close[1])).
```

### Remarks
ta.tr(false) is exactly the same as ta.tr.

### See also
True strength index. It uses moving averages of the underlying momentum of a
financial instrument.

### See also
- ta.tr
- ta.atr

## ta.tsi()
True strength index. It uses moving averages of the underlying momentum of a
financial instrument.

### Syntax
```
source (series int/float) Source series.
```

### Arguments
```
source (series int/float) Source series.

```

### Returns
```
True strength index. A value in range [-1, 1].
```

### Remarks
na values in the source series are ignored; the function calculates on the length quantity of non-na values.

## ta.valuewhen()
Returns the value of the source series on the bar where the condition was true on the nth most recent occurrence.

### Syntax & Overloads
condition (series bool) The condition to search for.

### Arguments
```
condition (series bool) The condition to search for.
```

### Example
This function requires execution on every bar. It is not recommended to use it inside a for or while loop structure, where its behavior can be unexpected. Please note that using this function can cause indicator repainting.

### Remarks
This function requires execution on every bar. It is not recommended to use it inside a for or while loop structure, where its behavior can be unexpected. Please note that using this function can cause indicator repainting.

### See also
Variance is the expectation of the squared deviation of a series from its mean (ta.sma), and it informally measures how far a set of numbers are spread out from their mean.
```

### See also
- ta.lowestbars
- ta.highestbars
- ta.barssince
- ta.highest
- ta.lowest

## ta.variance()
Variance is the expectation of the squared deviation of a series from its mean (ta.sma), and it informally measures how far a set of numbers are spread out from their mean.

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Returns
```
Variance of source for length bars back.
```

### Remarks
If biased is true, function will calculate using a biased estimate of the entire population, if false - unbiased estimate of a sample.

### See also
Volume weighted average price.

### See also
- ta.dev
- ta.stdev

## ta.vwap()

Volume weighted average price.

### Syntax & Overloads
source (series int/float) Source used for the VWAP calculation.

### Arguments
```
source (series int/float) Source used for the VWAP calculation.
```

### Example
A VWAP series, or a tuple [vwap, upper_band, lower_band] if stdev_mult is specified.

### Example
A VWAP series, or a tuple [vwap, upper_band, lower_band] if stdev_mult is specified.

### Returns
```
A VWAP series, or a tuple [vwap, upper_band, lower_band] if stdev_mult is
specified.
```

### Remarks
Calculations only begin the first time the anchor condition becomes true. Until then, the function returns na.

### See also
The vwma function returns volume-weighted moving average of source for length bars back. It is the same as: sma(source * volume, length) / sma(volume, length).

### See also
- ta.vwap

## ta.vwma()
The vwma function returns volume-weighted moving average of source for length bars back. It is the same as: sma(source * volume, length) / sma(volume, length).

### Syntax
```
source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Volume-weighted moving average of source for length bars back.

### Returns
```
Volume-weighted moving average of source for length bars back.
```

### Remarks
na values in the source series are ignored.

### See also
The wma function returns weighted moving average of source for length bars back. In wma weighting factors decrease in arithmetical progression.

### See also
- ta.sma
- ta.ema
- ta.rma
- ta.wma
- ta.swma
- ta.alma

## ta.wma()
The wma function returns weighted moving average of source for length bars back. In wma weighting factors decrease in arithmetical progression.

### Syntax
```
```

source (series int/float) Series of values to process.
```

### Arguments
```
source (series int/float) Series of values to process.
```

### Example
Weighted moving average of source for length bars back.

### Returns
```
Weighted moving average of source for length bars back.
```

### Remarks
na values in the source series are ignored.

### See also
Williams %R. The oscillator shows the current closing price in relation to the high and low of the past 'length' bars.

### See also
- ta.sma
- ta.ema
- ta.rma
- ta.vwma
- ta.swma
- ta.alma

## ta.wpr()
Williams %R. The oscillator shows the current closing price in relation to the high and low of the past 'length' bars.

### Syntax
```
length (series int) Number of bars.
```
```

### Arguments
```
length (series int) Number of bars.
```

### Example
Williams %R.

### Returns
```
Williams %R.
```

### Remarks
na values in the source series are ignored.

### See also
Casts na to table

### See also
- ta.mfi
- ta.cmo

## table()
Casts na to table

### Syntax
```
x (series table) The value to convert to the specified type, usually na.
```

### Arguments
```
x (series table) The value to convert to the specified type, usually na.
```

### Returns
```
```

```
The value of the argument after casting to table.
```

### See also
The function defines a cell in the table and sets its attributes.

### See also
- float
- int
- bool
- color
- string
- line
- label

## table.cell()
The function defines a cell in the table and sets its attributes.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### Remarks
This function does not create the table itself, but defines the table’s cells. To use it, you first need to create a table object with table.new.

### See also
The function sets the background color of the cell.

### See also
- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text
- table.cell_set_text_formatting

- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip

## table.cell_set_bgcolor()
The function sets the background color of the cell.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the height of cell.

### See also
- table.cell_set_height
- table.cell_set_text
- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip

## table.cell_set_height()
The function sets the height of cell.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the text in the specified cell.

### See also
- table.cell_set_bgcolor
- table.cell_set_text
- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip

## table.cell_set_text()
The function sets the text in the specified cell.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### Example
The function sets the color of the text inside the cell.

### See also
The function sets the color of the text inside the cell.

### See also

- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip
- table.cell_set_text_formatting

## table.cell_set_text_color()
The function sets the color of the text inside the cell.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the font family of the text inside the cell.

### See also
- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip

## table.cell_set_text_font_family()
The function sets the font family of the text inside the cell.

### Syntax

```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### Example
Sets the formatting attributes the drawing applies to displayed text.

### See also
Sets the formatting attributes the drawing applies to displayed text.

### See also
- table.new
- font.family_default
- font.family_monospace

## table.cell_set_text_formatting()
Sets the formatting attributes the drawing applies to displayed text.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the horizontal alignment of the cell's text.

### See also
- table.cell_set_bgcolor
- table.cell_set_height

- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip
- table.cell_set_text

## table.cell_set_text_halign()
The function sets the horizontal alignment of the cell's text.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the size of the cell's text.

### See also
- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text
- table.cell_set_text_color
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip

## table.cell_set_text_size()
The function sets the size of the cell's text.

### Syntax
```
table_id (series table) A table object.
```

```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the vertical alignment of a cell's text.

### See also
- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text
- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_tooltip

## table.cell_set_text_valign()
The function sets the vertical alignment of a cell's text.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the tooltip in the specified cell.

### See also
- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text
```

- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_width
- table.cell_set_tooltip

## table.cell_set_tooltip()
The function sets the tooltip in the specified cell.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### Example
The function sets the width of the cell.

### See also
The function sets the width of the cell.

### See also
- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_width
- table.cell_set_text

## table.cell_set_width()
The function sets the width of the cell.

### Syntax

```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function removes a cell or a sequence of cells from the table. The cells are removed in a rectangle shape where the start_column and start_row specify the top-left corner, and end_column and end_row specify the bottom-right corner.

### See also
- table.cell_set_bgcolor
- table.cell_set_height
- table.cell_set_text
- table.cell_set_text_color
- table.cell_set_text_halign
- table.cell_set_text_size
- table.cell_set_text_valign
- table.cell_set_tooltip

## table.clear()
The function removes a cell or a sequence of cells from the table. The cells are removed in a rectangle shape where the start_column and start_row specify the top-left corner, and end_column and end_row specify the bottom-right corner.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### Example
The function deletes a table.

### See also
The function deletes a table.

### See also
- table.delete
- table.new

## table.delete()
The function deletes a table.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### Example
The function merges a sequence of cells in the table into one cell. The cells are merged in a rectangle shape where the start_column and start_row specify the top-left corner, and end_column and end_row specify the bottom-right corner.

### See also
The function merges a sequence of cells in the table into one cell. The cells are merged in a rectangle shape where the start_column and start_row specify the top-left corner, and end_column and end_row specify the bottom-right corner.

### See also
- table.new
- table.clear

## table.merge_cells()
The function merges a sequence of cells in the table into one cell. The cells are merged in a rectangle shape where the start_column and start_row specify the top-

left corner, and end_column and end_row specify the bottom-right corner.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### Example
This function will merge cells, even if their properties are not yet defined with table.cell.

### Remarks
This function will merge cells, even if their properties are not yet defined with table.cell.

### See also
The function creates a new table.

### See also
- table.delete
- table.new

## table.new()
The function creates a new table.

### Syntax
```
- position (series string) Position of the table. Possible values are :
position.top_left, position.top_center, position.top_right, position.middle_left,
position.middle_center, position.middle_right, position.bottom_left,
position.bottom_center, position.bottom_right.
```

### Arguments

```
- position (series string) Position of the table. Possible values are :
position.top_left, position.top_center, position.top_right, position.middle_left,
position.middle_center, position.middle_right, position.bottom_left,
position.bottom_center, position.bottom_right.
```

### Example
The ID of a table object that can be passed to other table.*() functions.

### Returns
```
The ID of a table object that can be passed to other table.*() functions.
```

### Remarks
This function creates the table object itself, but the table will not be displayed until its cells are populated. To define a cell and change its contents or attributes, use table.cell and other table.cell_*() functions.

### See also
The function sets the background color of a table.

### See also
- table.cell
- table.clear
- table.delete
- table.set_bgcolor
- table.set_border_color
- table.set_border_width
- table.set_frame_color
- table.set_frame_width
- table.set_position

## table.set_bgcolor()
The function sets the background color of a table.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the color of the borders (excluding the outer frame) of the table's cells.

### See also
- table.clear
- table.delete
- table.new
- table.set_border_color
- table.set_border_width
- table.set_frame_color
- table.set_frame_width
- table.set_position

## table.set_border_color()
The function sets the color of the borders (excluding the outer frame) of the table's cells.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the width of the borders (excluding the outer frame) of the table's cells.

### See also
- table.clear

- table.delete
- table.new
- table.set_frame_color
- table.set_border_width
- table.set_bgcolor
- table.set_frame_width
- table.set_position

## table.set_border_width()
The function sets the width of the borders (excluding the outer frame) of the table's cells.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the color of the outer frame of a table.

### See also
- table.clear
- table.delete
- table.new
- table.set_frame_color
- table.set_frame_width
- table.set_bgcolor
- table.set_border_color
- table.set_position

## table.set_frame_color()
The function sets the color of the outer frame of a table.

### Syntax
```
```

```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function set the width of the outer frame of a table.

### See also
- table.clear
- table.delete
- table.new
- table.set_border_color
- table.set_border_width
- table.set_bgcolor
- table.set_frame_width
- table.set_position

## table.set_frame_width()
The function set the width of the outer frame of a table.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
The function sets the position of a table.

### See also
- table.clear
- table.delete

- table.new
- table.set_frame_color
- table.set_border_width
- table.set_bgcolor
- table.set_border_color
- table.set_position

## table.set_position()
The function sets the position of a table.

### Syntax
```
table_id (series table) A table object.
```

### Arguments
```
table_id (series table) A table object.
```

### See also
Creates a ticker identifier for requesting Heikin Ashi bar values.

### See also
- table.clear
- table.delete
- table.new
- table.set_bgcolor
- table.set_border_color
- table.set_border_width
- table.set_frame_color
- table.set_frame_width

## ticker.heikinashi()
Creates a ticker identifier for requesting Heikin Ashi bar values.

### Syntax & Overloads
symbol (simple string) Symbol ticker identifier.

### Arguments
```
symbol (simple string) Symbol ticker identifier.
```

### Example
String value of ticker id, that can be supplied to request.security function.

### Returns
```
String value of ticker id, that can be supplied to request.security function.
```

### See also
Constructs a ticker ID for the specified symbol with additional parameters inherited from the ticker ID passed into the function call, allowing the script to request a symbol's data using the same modifiers that the from_tickerid has, including extended session, dividend adjustment, currency conversion, non-standard chart types, back-adjustment, settlement-as-close, etc.

### See also
- syminfo.tickerid
- syminfo.ticker
- request.security
- ticker.renko
- ticker.linebreak
- ticker.kagi
- ticker.pointfigure

## ticker.inherit()
Constructs a ticker ID for the specified symbol with additional parameters inherited from the ticker ID passed into the function call, allowing the script to request a symbol's data using the same modifiers that the from_tickerid has, including extended session, dividend adjustment, currency conversion, non-standard chart types, back-adjustment, settlement-as-close, etc.

### Syntax & Overloads
from_tickerid (simple string) The ticker ID to inherit modifiers from.

### Arguments
```
```

from_tickerid (simple string) The ticker ID to inherit modifiers from.
```

### Example
If the constructed ticker ID inherits a modifier that doesn't apply to the symbol (e.g., if the from_tickerid has Extended Hours enabled, but no such option is available for the symbol), the script will ignore the modifier when requesting data using the ID.

### Remarks
If the constructed ticker ID inherits a modifier that doesn't apply to the symbol (e.g., if the from_tickerid has Extended Hours enabled, but no such option is available for the symbol), the script will ignore the modifier when requesting data using the ID.

## ticker.kagi()
Creates a ticker identifier for requesting Kagi values.

### Syntax & Overloads
symbol (simple string) Symbol ticker identifier.

### Arguments
```
symbol (simple string) Symbol ticker identifier.
```

### Example
String value of ticker id, that can be supplied to request.security function.

### Returns
```
String value of ticker id, that can be supplied to request.security function.
```

### See also
Creates a ticker identifier for requesting Line Break values.

### See also
```

- syminfo.tickerid
- syminfo.ticker
- request.security
- ticker.heikinashi
- ticker.renko
- ticker.linebreak
- ticker.pointfigure

## ticker.linebreak()
Creates a ticker identifier for requesting Line Break values.

### Syntax & Overloads
symbol (simple string) Symbol ticker identifier.

### Arguments
```
symbol (simple string) Symbol ticker identifier.
```

### Example
String value of ticker id, that can be supplied to request.security function.

### Returns
```
String value of ticker id, that can be supplied to request.security function.
```

### See also
Creates a ticker identifier for requesting additional data for the script.

### See also
- syminfo.tickerid
- syminfo.ticker
- request.security
- ticker.heikinashi
- ticker.renko
- ticker.kagi
- ticker.pointfigure

## ticker.modify()
Creates a ticker identifier for requesting additional data for the script.

### Syntax & Overloads
tickerid (series string) Symbol name with exchange prefix, e.g. 'BATS:MSFT', 'NASDAQ:MSFT' or tickerid with session and adjustment from the ticker.new function.

### Arguments
```
- tickerid (series string) Symbol name with exchange prefix, e.g. 'BATS : MSFT',
'NASDAQ:MSFT' or tickerid with session and adjustment from the ticker.new function.
```

### Example
String value of ticker id, that can be supplied to request.security function.

### Returns
```
String value of ticker id, that can be supplied to request.security function.
```

### See also
Creates a ticker identifier for requesting additional data for the script.

### See also
- syminfo.tickerid
- syminfo.ticker
- syminfo.session
- session.extended
- session.regular
- ticker.heikinashi
- adjustment.none
- adjustment.splits
- adjustment.dividends
- backadjustment.inherit
- backadjustment.on
- backadjustment.off
- settlement_as_close.inherit
- settlement_as_close.on
- settlement_as_close.off

## ticker.new()
Creates a ticker identifier for requesting additional data for the script.

### Syntax & Overloads
prefix (series string) Exchange prefix. For example: 'BATS', 'NYSE', 'NASDAQ'. Exchange prefix of main series is syminfo.prefix.

### Arguments
```
- prefix (series string) Exchange prefix. For example : 'BATS', 'NYSE', 'NASDAQ'.
Exchange prefix of main series is syminfo.prefix.
```

### Example
String value of ticker id, that can be supplied to request.security function.

### Returns
```
String value of ticker id, that can be supplied to request.security function.
```

### Remarks
You may use return value of ticker.new function as input argument for ticker.heikinashi, ticker.renko, ticker.linebreak, ticker.kagi, ticker.pointfigure functions.

### See also
Creates a ticker identifier for requesting Point & Figure values.

### See also
- syminfo.tickerid
- syminfo.ticker
- syminfo.session
- session.extended
- session.regular
- ticker.heikinashi
- adjustment.none
- adjustment.splits
- adjustment.dividends
- backadjustment.inherit
- backadjustment.on
- backadjustment.off

- settlement_as_close.inherit
- settlement_as_close.on
- settlement_as_close.off

## ticker.pointfigure()
Creates a ticker identifier for requesting Point & Figure values.

### Syntax & Overloads
symbol (simple string) Symbol ticker identifier.

### Arguments
```
symbol (simple string) Symbol ticker identifier.
```

### Example
String value of ticker id, that can be supplied to request.security function.

### Returns
```
String value of ticker id, that can be supplied to request.security function.
```

### See also
Creates a ticker identifier for requesting Renko values.

### See also
- syminfo.tickerid
- syminfo.ticker
- request.security
- ticker.heikinashi
- ticker.renko
- ticker.linebreak
- ticker.kagi

## ticker.renko()
Creates a ticker identifier for requesting Renko values.

### Syntax & Overloads
symbol (simple string) Symbol ticker identifier.

### Arguments
```
symbol (simple string) Symbol ticker identifier.
```

### Example
String value of ticker id, that can be supplied to request.security function.

### Example
String value of ticker id, that can be supplied to request.security function.

### Returns
```
String value of ticker id, that can be supplied to request.security function.
```

### See also
Creates a ticker to request data from a standard chart that is unaffected by modifiers like extended session, dividend adjustment, currency conversion, and the calculations of non-standard chart types: Heikin Ashi, Renko, etc. Among other things, this makes it possible to retrieve standard chart values when the script is running on a non-standard chart.

### See also
- syminfo.tickerid
- syminfo.ticker
- request.security
- ticker.heikinashi
- ticker.linebreak
- ticker.kagi
- ticker.pointfigure

## ticker.standard()
Creates a ticker to request data from a standard chart that is unaffected by modifiers like extended session, dividend adjustment, currency conversion, and the calculations of non-standard chart types: Heikin Ashi, Renko, etc. Among other things, this makes it possible to retrieve standard chart values when the script is running on a non-standard chart.

### Syntax & Overloads
symbol (simple string) A ticker ID to be converted into its standard form. Optional. The default is syminfo.tickerid.

### Arguments
```
symbol (simple string) A ticker ID to be converted into its standard form.
Optional. The default is syminfo.tickerid.
```

### Example
A string representing the ticker of a standard chart in the "prefix:ticker" format. If the symbol argument does not contain the prefix and ticker information, the function returns the supplied argument as is.

### Returns
```
- A string representing the ticker of a standard chart in the "prefix : ticker"
format. If the symbol argument does not contain the prefix and ticker information,
the function returns the supplied argument as is.
```

### See also
The time function returns the UNIX time of the current bar for the specified timeframe and session or NaN if the time point is out of session.

### See also
- request.security

## time()
The time function returns the UNIX time of the current bar for the specified timeframe and session or NaN if the time point is out of session.

### Syntax & Overloads
timeframe (series string) Timeframe. An empty string is interpreted as the current timeframe of the chart.

### Arguments

```
timeframe (series string) Timeframe. An empty string is interpreted as the current timeframe of the chart.
```

### Example
While setting up a session you can specify not just the hours and minutes but also the days of the week that will be included in that session.

### Example
One session argument can include several different sessions, separated by commas. For example, the following script will highlight the bars from 10:00 to 11:00 and from 14:00 to 15:00 (workdays only):

### Example
UNIX time.

### Returns
```
UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
Returns the UNIX time of the current bar's close for the specified timeframe and session, or na if the time point is outside the session. On non-standard price-based chart types (Renko, Line break, Kagi, Point & Figure, and Range), this function returns na on the chart's realtime bars.

### See also
- time

## time_close()
Returns the UNIX time of the current bar's close for the specified timeframe and session, or na if the time point is outside the session. On non-standard price-based chart types (Renko, Line break, Kagi, Point & Figure, and Range), this function returns na on the chart's realtime bars.

### Syntax & Overloads
timeframe (series string) Resolution. An empty string is interpreted as the current resolution of the chart.

### Arguments
```
timeframe (series string) Resolution. An empty string is interpreted as the current resolution of the chart.
```

### Example
UNIX time.

### Returns
```
UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
Detects changes in the specified timeframe.

### See also
- time_close

## timeframe.change()
Detects changes in the specified timeframe.

### Syntax
```
timeframe (series string) String formatted according to the User manual's timeframe string specifications.
```

### Arguments
```
timeframe (series string) String formatted according to the User manual's timeframe string specifications.
```

### Example
Returns true on the first bar of a new timeframe, false otherwise.

### Returns
```
Returns true on the first bar of a new timeframe, false otherwise.
```

## timeframe.from_seconds()
Converts a number of seconds into a valid timeframe string.

### Syntax & Overloads
seconds (simple int) The number of seconds in the timeframe.

### Arguments
```
seconds (simple int) The number of seconds in the timeframe.
```

### Example
A timeframe string compliant with timeframe string specifications.

### Returns
```
A timeframe string compliant with timeframe string specifications.
```

### Remarks
If no valid timeframe exists for the quantity of seconds supplied, the next higher valid timeframe will be returned. Accordingly, one second or less will return "1S", 2-5 seconds will return "5S", and 604,799 seconds (one second less than 7 days) will return "7D".

### See also
Converts a timeframe string into seconds.

### See also
- timeframe.in_seconds
- request.security
- request.security_lower_tf

## timeframe.in_seconds()
Converts a timeframe string into seconds.

### Syntax & Overloads
timeframe (simple string) Timeframe string in timeframe string specifications format. Optional. The default is timeframe.period.

### Arguments
```
timeframe (simple string) Timeframe string in timeframe string specifications format. Optional. The default is timeframe.period.
```

### Example
The "int" representation of the number of seconds in the timeframe string.

### Returns
```
The "int" representation of the number of seconds in the timeframe string.
```

### Remarks
When the timeframe is "1M" or more, calculations use 2628003 as the number of seconds in one month, which represents 30.4167 (365/12) days.

### See also
Function timestamp returns UNIX time of specified date and time.

### See also
- input.timeframe

- timeframe.period
- timeframe.from_seconds

## timestamp()
Function timestamp returns UNIX time of specified date and time.

### Syntax & Overloads
dateString (const string) A string containing the date and, optionally, the time and time zone. Its format must comply with either the IETF RFC 2822 or ISO 8601 standards ("DD MMM YYYY hh:mm:ss ±hhmm" or "YYYY-MM-DDThh:mm:ss±hh:mm", so "20 Feb 2020" or "2020-02-20"). If no time is supplied, "00:00" is used. If no time zone is supplied, GMT+0 will be used. Note that this diverges from the usual behavior of the function where it returns time in the exchange's timezone.

### Arguments
```
- dateString (const string) A string containing the date and, optionally, the time and time zone. Its format must comply with either the IETF RFC 2822 or ISO 8601 standards ("DD MMM YYYY hh : mm:ss ±hhmm" or "YYYY-MM-DDThh:mm:ss±hh:mm", so "20 Feb 2020" or "2020-02-20"). If no time is supplied, "00:00" is used. If no time zone is supplied, GMT+0 will be used. Note that this diverges from the usual behavior of the function where it returns time in the exchange's timezone.
```

### Example
UNIX time.

### Returns
```
UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
time (series int) UNIX time in milliseconds.

### See also

- time

## weekofyear()
time (series int) UNIX time in milliseconds.

### Syntax & Overloads
time (series int) UNIX time in milliseconds.

### Arguments
```
time (series int) UNIX time in milliseconds.
```

### Returns
```
Week of year (in exchange timezone) for provided UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
time (series int) UNIX time in milliseconds.

### See also
- weekofyear
- time
- year
- month
- dayofmonth
- dayofweek
- hour
- minute
- second

## year()
time (series int) UNIX time in milliseconds.

### Syntax & Overloads
time (series int) UNIX time in milliseconds.

### Arguments
```
time (series int) UNIX time in milliseconds.
```

### Returns
```
Year (in exchange timezone) for provided UNIX time.
```

### Remarks
UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

### See also
Logical AND. Applicable to boolean expressions.

### See also
- year
- time
- month
- dayofmonth
- dayofweek
- hour
- minute
- second

## and
Logical AND. Applicable to boolean expressions.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns

```
Boolean value, or series of boolean values.
```

### Remarks
If expr1 evaluates to false, the and operator returns false without evaluating expr2.

## enum
This keyword allows the creation of an enumeration, enum for short. Enums are unique constructs that hold groups of predefined constants.

### Syntax
```
One can use an enum to quickly create a dropdown input with the help of the
input.enum function. The options that appear in the dropdown represent the titles
of the enum fields.
```

### Example
Additionally, one can use an enum in a collection's type template to restrict the values it will allow as elements. When used inside a type template, the collection will only accept fields that belong to the specified enum.

### Example
Used in libraries to prefix the declaration of functions or user-defined type definitions that will be available from other scripts importing the library.

## export
Used in libraries to prefix the declaration of functions or user-defined type definitions that will be available from other scripts importing the library.

### Example
Each library must have at least one exported function or user-defined type (UDT).

### Remarks
Each library must have at least one exported function or user-defined type (UDT).

### See also
The 'for' structure allows the repeated execution of a number of statements:

### See also
- library
- import
- simple
- series
- type

## for
The 'for' structure allows the repeated execution of a number of statements:

### Syntax
```
var_declaration - An optional variable declaration that will be assigned the value
of the loop's return_expression.
```

### Example
The for...in structure allows the repeated execution of a number of statements for each element in an array. It can be used with either one argument: array_element, or with two: [index, array_element]. The second form doesn't affect the functionality of the loop. It tracks the current iteration's index in the tuple's first variable.

### Example
The for...in structure allows the repeated execution of a number of statements for each element in an array. It can be used with either one argument: array_element, or with two: [index, array_element]. The second form doesn't affect the functionality of the loop. It tracks the current iteration's index in the tuple's first variable.

### See also
The for...in structure allows the repeated execution of a number of statements for each element in an array. It can be used with either one argument: array_element, or with two: [index, array_element]. The second form doesn't affect the functionality of the loop. It tracks the current iteration's index in the tuple's first variable.

### See also
- for...in
- while

## for...in
The for...in structure allows the repeated execution of a number of statements for each element in an array. It can be used with either one argument: array_element, or with two: [index, array_element]. The second form doesn't affect the functionality of the loop. It tracks the current iteration's index in the tuple's first variable.

### Syntax
```
var_declaration - An optional variable declaration that will be assigned the value
of the loop's return_expression.
```

### Example
Here, we use the two-argument form of for...in to set the values of our isPos array to true when their corresponding value in our valuesArray array is positive:

### Example
Iterate through matrix rows as arrays.

### Example
If statement defines what block of statements must be executed when conditions of the expression are satisfied.

### See also
If statement defines what block of statements must be executed when conditions of the expression are satisfied.

### See also
- for
- while
- array.sum
- array.min
- array.max

if
If statement defines what block of statements must be executed when conditions of the expression are satisfied.

### Syntax
```
where
```

### Example
It is possible to omit the else block. In this case if the condition is false, an “empty” value (na, false, or “”) will be assigned to the var_declarationX variable:

### Example
It is possible to use either multiple “else if” blocks or none at all. The blocks “then”, “else if”, “else” are shifted by four spaces:

### Example
It is possible to ignore the resulting value of an if statement (“var_declarationX=“ can be omitted). It may be useful if you need the side effect of the expression, for example in strategy trading:

### Example
If statements can include each other:

### Example
Used to load an external library into a script and bind its functions to a namespace. The importing script can be an indicator, a strategy, or another library. A library must be published (privately or publicly) before it can be imported.

## import
Used to load an external library into a script and bind its functions to a namespace. The importing script can be an indicator, a strategy, or another library. A library must be published (privately or publicly) before it can be imported.

### Syntax

```
username (literal string) User name of the library's author.
```

### Arguments
```
username (literal string) User name of the library's author.
```

### Example
Using an alias that replaces a built-in namespace such as math.* or strategy.* is allowed, but if the library contains function names that shadow Pine Script®'s built-in functions, the built-ins will become unavailable. The same version of a library can only be imported once. Aliases must be distinct for each imported library. When calling library functions, casting their arguments to types other than their declared type is not allowed. An import statement cannot use 'as' or 'import' as username, libraryName, or alias identifiers.

### Remarks
Using an alias that replaces a built-in namespace such as math.* or strategy.* is allowed, but if the library contains function names that shadow Pine Script®'s built-in functions, the built-ins will become unavailable. The same version of a library can only be imported once. Aliases must be distinct for each imported library. When calling library functions, casting their arguments to types other than their declared type is not allowed. An import statement cannot use 'as' or 'import' as username, libraryName, or alias identifiers.

### See also
This keyword is used to prefix a function declaration, indicating it can then be invoked using dot notation by appending its name to a variable of the type of its first parameter and omitting that first parameter. Alternatively, functions declared as methods can also be invoked like normal user-defined functions. In that case, an argument must be supplied for its first parameter.

### See also
- library
- export

## method
This keyword is used to prefix a function declaration, indicating it can then be invoked using dot notation by appending its name to a variable of the type of its first parameter and omitting that first parameter. Alternatively, functions

declared as methods can also be invoked like normal user-defined functions. In that case, an argument must be supplied for its first parameter.

### Syntax
```
Logical negation (NOT). Applicable to boolean expressions.
```

### Example
Logical negation (NOT). Applicable to boolean expressions.

## not
Logical negation (NOT). Applicable to boolean expressions.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

or
Logical OR. Applicable to boolean expressions.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

### Remarks
If expr1 evaluates to true, the or operator returns true without evaluating expr2.

## switch
The switch operator transfers control to one of the several statements, depending on the values of a condition and expressions.

### Syntax
```
- Switch with an expression :
```

### Example
Switch without an expression:

### Example
The value of the last expression in the local block of statements that is executed.

### Returns
```
The value of the last expression in the local block of statements that is executed.
```

### Remarks
Only one of the local_block instances or the default_local_block can be executed. The default_local_block is introduced with the => token alone and is only executed when none of the preceding blocks are executed. If the result of the switch statement is assigned to a variable and a default_local_block is not specified, the statement returns na if no local_block is executed. When assigning the result of the switch statement to a variable, all local_block instances must return the same type of value.

### See also
This keyword allows the declaration of user-defined types (UDT) from which scripts can instantiate objects. UDTs are composite types that contain an arbitrary number of fields of any built-in or user-defined type, including the defined UDT itself. The syntax to define a UDT is:

### See also
- if
- ?:

## type
This keyword allows the declaration of user-defined types (UDT) from which scripts can instantiate objects. UDTs are composite types that contain an arbitrary number of fields of any built-in or user-defined type, including the defined UDT itself. The syntax to define a UDT is:

### Syntax
```
Once a UDT is defined, scripts can instantiate objects from it with the UDT_identifier.new() construct. When creating a new type instance, the fields of the resulting object will initialize with the default values from the UDT's definition. Any type fields without specified defaults will initialize as na. Alternatively, users can pass initial values as arguments in the *.new() method to override the type's defaults. For example, newFooObject = foo.new(x = true) assigns a new foo object to the newFooObject variable with its x field initialized using a value of true.
```

### Example
var is the keyword used for assigning and one-time initializing of the variable.

## var
var is the keyword used for assigning and one-time initializing of the variable.

### Syntax
```
- where :
```

### Example
The variable 'a' keeps the closing price of the first bar for each bar in the series.

## varip
varip (var intrabar persist) is the keyword used for the assignment and one-time initialization of a variable or a field of a user-defined type. It’s similar to the var keyword, but variables and fields declared with varip retain their values between executions of the script on the same bar.

### Syntax
```
- where :
```

### Example
With var, v would equal the value of the bar_index. On historical bars, where the script calculates only once per chart bar, the value of v is the same as with var. However, on realtime bars, the script will evaluate the expression on each new chart update, producing a different result.

### Example
The same += operation applied to both the index and ticks fields results in different real-time values because ticks increases on every chart update, while index only does so once per bar. Note how the currBar object does not use the varip keyword. The ticks field of the object can increment on every tick, but the reference itself is defined once and then stays unchanged. If we were to declare currBar using varip, the behavior of index would remain unchanged because while the reference to the type instance would persist between chart updates, the index field of the object would not.

### Remarks
When using varip to declare variables in strategies that may execute more than once per historical chart bar, the values of such variables are preserved across successive iterations of the script on the same bar.

## while
The while statement allows the conditional iteration of a local code block.

### Syntax
```
- where :
```

### Example
The local code block after the initial while line must be indented with four spaces or a tab. For the while loop to terminate, the boolean expression following while must eventually become false, or a break must be executed.

### Remarks
The local code block after the initial while line must be indented with four spaces or a tab. For the while loop to terminate, the boolean expression following while must eventually become false, or a break must be executed.

## array
Keyword used to explicitly declare the "array" type of a variable or a parameter. Array objects (or IDs) can be created with the array.new<type>, array.from function.

### Example
Array objects are always of "series" form.

### Remarks
Array objects are always of "series" form.

### See also
Keyword used to explicitly declare the "bool" (boolean) type of a variable or a parameter. "Bool" variables can have values true or false.

### See also
- var
- line
- label
- table
- box
- array.new<type>
- array.from

## bool
Keyword used to explicitly declare the "bool" (boolean) type of a variable or a parameter. "Bool" variables can have values true or false.

### Example
Explicitly mentioning the type in a variable declaration is optional. Learn more about Pine Script® types in the User Manual page on the Type System.

### Remarks
Explicitly mentioning the type in a variable declaration is optional. Learn more about Pine Script® types in the User Manual page on the Type System.

### See also
Keyword used to explicitly declare the "box" type of a variable or a parameter. Box objects (or IDs) can be created with the box.new function.

### See also
- var
- varip
- int
- float
- color
- string
- true
- false

## box
Keyword used to explicitly declare the "box" type of a variable or a parameter. Box objects (or IDs) can be created with the box.new function.

### Example
Box objects are always of "series" form.

### Remarks
Box objects are always of "series" form.

### See also
Keyword to explicitly declare the type of a variable or parameter as chart.point. Scripts can produce chart.point instances using the chart.point.from_time, chart.point.from_index, chart.point.now, and chart.point.new functions.

### See also
- var
- line

- label
- table
- box.new

## chart.point
Keyword to explicitly declare the type of a variable or parameter as chart.point. Scripts can produce chart.point instances using the chart.point.from_time, chart.point.from_index, chart.point.now, and chart.point.new functions.

### Fields
index (series int) The x-coordinate of the point, expressed as a bar index value.

### See also
Keyword used to explicitly declare the "color" type of a variable or a parameter.

### See also
- polyline

## color
Keyword used to explicitly declare the "color" type of a variable or a parameter.

### Example
Color literals have the following format: #RRGGBB or #RRGGBBAA. The letter pairs represent 00 to FF hexadecimal values (0 to 255 in decimal) where RR, GG and BB pairs are the values for the color's red, green and blue components. AA is an optional value for the color's transparency (or alpha component) where 00 is invisible and FF opaque. When no AA pair is supplied, FF is used. The hexadecimal letters can be upper or lower case.

### Remarks
Color literals have the following format: #RRGGBB or #RRGGBBAA. The letter pairs represent 00 to FF hexadecimal values (0 to 255 in decimal) where RR, GG and BB pairs are the values for the color's red, green and blue components. AA is an optional value for the color's transparency (or alpha component) where 00 is invisible and FF opaque. When no AA pair is supplied, FF is used. The hexadecimal letters can be upper or lower case.

### See also
The const keyword explicitly assigns the "const" type qualifier to variables and

the parameters of non-exported functions. Variables and parameters with the "const" qualifier reference values established at compile time that never change in the script's execution.

### See also
- var
- varip
- int
- float
- string
- color.rgb
- color.new

## const
The const keyword explicitly assigns the "const" type qualifier to variables and the parameters of non-exported functions. Variables and parameters with the "const" qualifier reference values established at compile time that never change in the script's execution.

### Syntax
```
To learn more, see our User Manual's section on type qualifiers.
```

### Example
To learn more, see our User Manual's section on type qualifiers.

### Example
To learn more, see our User Manual's section on type qualifiers.

### Remarks
To learn more, see our User Manual's section on type qualifiers.

### See also
Keyword used to explicitly declare the "float" (floating point) type of a variable or a parameter.

### See also
- simple
- series

## float
Keyword used to explicitly declare the "float" (floating point) type of a variable or a parameter.

### Example
Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with na. Learn more about Pine Script® types in the User Manual page on the Type System.

### Remarks
Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with na. Learn more about Pine Script® types in the User Manual page on the Type System.

### See also
Keyword used to explicitly declare the "int" (integer) type of a variable or a parameter.

### See also
- var
- varip
- int
- bool
- color
- string

## int
Keyword used to explicitly declare the "int" (integer) type of a variable or a parameter.

### Example
Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with na. Learn more about Pine Script® types in the User Manual page on the Type System.

### Remarks
Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with na. Learn more about Pine Script® types in the User Manual

page on the Type System.

### See also
Keyword used to explicitly declare the "label" type of a variable or a parameter. Label objects (or IDs) can be created with the label.new function.

### See also
- var
- varip
- float
- bool
- color
- string

## label
Keyword used to explicitly declare the "label" type of a variable or a parameter. Label objects (or IDs) can be created with the label.new function.

### Example
Label objects are always of "series" form.

### Remarks
Label objects are always of "series" form.

### See also
Keyword used to explicitly declare the "line" type of a variable or a parameter. Line objects (or IDs) can be created with the line.new function.

### See also
- var
- line
- box
- label.new

## line
Keyword used to explicitly declare the "line" type of a variable or a parameter. Line objects (or IDs) can be created with the line.new function.

### Example
Line objects are always of "series" form.

### Remarks
Line objects are always of "series" form.

### See also
Keyword used to explicitly declare the "linefill" type of a variable or a parameter. Linefill objects (or IDs) can be created with the linefill.new function.

### See also
- var
- label
- box
- line.new

## linefill
Keyword used to explicitly declare the "linefill" type of a variable or a parameter. Linefill objects (or IDs) can be created with the linefill.new function.

### Example
Linefill objects are always of "series" form.

### Remarks
Linefill objects are always of "series" form.

### See also
Keyword used to explicitly declare the "map" type of a variable or a parameter. Map objects (or IDs) can be created with the map.new<type,type> function.

### See also
- var
- line
- label
- table
- box
- linefill.new

## map
Keyword used to explicitly declare the "map" type of a variable or a parameter. Map objects (or IDs) can be created with the map.new<type,type> function.

### Example
Map objects are always of series form.

### Remarks
Map objects are always of series form.

### See also
Keyword used to explicitly declare the "matrix" type of a variable or a parameter. Matrix objects (or IDs) can be created with the matrix.new<type> function.

### See also
- map.new<type,type>

## matrix
Keyword used to explicitly declare the "matrix" type of a variable or a parameter. Matrix objects (or IDs) can be created with the matrix.new<type> function.

### Example
Matrix objects are always of "series" form.

### Remarks
Matrix objects are always of "series" form.

### See also
Keyword to explicitly declare the type of a variable or parameter as polyline. Scripts can produce polyline instances using the polyline.new function.

### See also
- var
- matrix.new<type>
- array

## polyline

Keyword to explicitly declare the type of a variable or parameter as polyline. Scripts can produce polyline instances using the polyline.new function.

### See also
The series keyword explicitly assigns the "series" type qualifier to variables and function parameters. Variables and parameters that use the "series" qualifier can reference values that change throughout a script's execution.

### See also
- chart.point

## series
The series keyword explicitly assigns the "series" type qualifier to variables and function parameters. Variables and parameters that use the "series" qualifier can reference values that change throughout a script's execution.

### Syntax
```
To learn more, see our User Manual's section on type qualifiers.
```

### Example
To learn more, see our User Manual's section on type qualifiers.

### Example
To learn more, see our User Manual's section on type qualifiers.

### Remarks
To learn more, see our User Manual's section on type qualifiers.

### See also
The simple keyword explicitly assigns the "simple" type qualifier to variables and function parameters. Variables and parameters that use the "simple" qualifier can reference values established at the beginning of a script's execution that do not change later.

### See also
- simple
- const

## simple
The simple keyword explicitly assigns the "simple" type qualifier to variables and function parameters. Variables and parameters that use the "simple" qualifier can reference values established at the beginning of a script's execution that do not change later.

### Syntax
```
To learn more, see our User Manual's section on type qualifiers.
```

### Example
To learn more, see our User Manual's section on type qualifiers.

### Example
To learn more, see our User Manual's section on type qualifiers.

### Remarks
To learn more, see our User Manual's section on type qualifiers.

### See also
Keyword used to explicitly declare the "string" type of a variable or a parameter.

### See also
- series
- const

## string
Keyword used to explicitly declare the "string" type of a variable or a parameter.

### Example
Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with na. Learn more about Pine Script® types in the User Manual page on the Type System.

### Remarks

Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with na. Learn more about Pine Script® types in the User Manual page on the Type System.

### See also
Keyword used to explicitly declare the "table" type of a variable or a parameter. Table objects (or IDs) can be created with the table.new function.

### See also
- var
- varip
- int
- float
- bool
- str.tostring
- str.format

## table
Keyword used to explicitly declare the "table" type of a variable or a parameter. Table objects (or IDs) can be created with the table.new function.

### Example
Table objects are always of "series" form.

### Remarks
Table objects are always of "series" form.

### See also
Subtraction or unary minus. Applicable to numerical expressions.

### See also
- var
- line
- label
- box
- table.new

-
Subtraction or unary minus. Applicable to numerical expressions.

### Syntax
```
- Returns integer or float value, or series of values :
```

### Returns
```
- Returns integer or float value, or series of values :
```

### Remarks
You may use arithmetic operators with numbers as well as with series variables. In case of usage with series the operators are applied elementwise.

-=
Subtraction assignment. Applicable to numerical expressions.

### Syntax
```
Integer or float value, or series of values.
```

### Example
Integer or float value, or series of values.

### Returns
```
Integer or float value, or series of values.
```

:=
Reassignment operator. It is used to assign a new value to a previously declared variable.

### Syntax

```
Not equal to. Applicable to expressions of any type.
```

### Example
Not equal to. Applicable to expressions of any type.

!=
Not equal to. Applicable to expressions of any type.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

?:
Ternary conditional operator.

### Syntax
```
expr2 if expr1 is evaluated to true, expr3 otherwise. Zero value (0 and also NaN,
+Infinity, -Infinity) is considered to be false, any other value is true.
```

### Example
expr2 if expr1 is evaluated to true, expr3 otherwise. Zero value (0 and also NaN, +Infinity, -Infinity) is considered to be false, any other value is true.

### Returns
```
expr2 if expr1 is evaluated to true, expr3 otherwise. Zero value (0 and also NaN,
+Infinity, -Infinity) is considered to be false, any other value is true.
```

```
```

### Remarks
Use na for 'else' branch if you do not need it.

### See also
Series subscript. Provides access to previous values of series expr1. expr2 is the number of bars back, and must be numerical. Floats will be rounded down.

### See also
- na

[]
Series subscript. Provides access to previous values of series expr1. expr2 is the number of bars back, and must be numerical. Floats will be rounded down.

### Syntax
```
A series of values.
```

### Example
A series of values.

### Returns
```
A series of values.
```

### See also
Multiplication. Applicable to numerical expressions.

### See also
- math.floor

*
Multiplication. Applicable to numerical expressions.

### Syntax
```
Integer or float value, or series of values.
```

### Returns
```
Integer or float value, or series of values.
```

*=
Multiplication assignment. Applicable to numerical expressions.

### Syntax
```
Integer or float value, or series of values.
```

### Example
Integer or float value, or series of values.

### Returns
```
Integer or float value, or series of values.
```

/
Division. Applicable to numerical expressions.

### Syntax
```
Integer or float value, or series of values.
```

### Returns

```
Integer or float value, or series of values.
```

/=
Division assignment. Applicable to numerical expressions.

### Syntax
```
Integer or float value, or series of values.
```

### Example
Integer or float value, or series of values.

### Returns
```
Integer or float value, or series of values.
```

%
Modulo (integer remainder). Applicable to numerical expressions.

### Syntax
```
Integer or float value, or series of values.
```

### Returns
```
Integer or float value, or series of values.
```

### Remarks
In Pine Script®, when the integer remainder is calculated, the quotient is truncated, i.e. rounded towards the lowest absolute value. The resulting value will have the same sign as the dividend.

%=
Modulo assignment. Applicable to numerical expressions.

### Syntax
```
Integer or float value, or series of values.
```

### Example
Integer or float value, or series of values.

### Returns
```
Integer or float value, or series of values.
```

+
Addition or unary plus. Applicable to numerical expressions or strings.

### Syntax
```
Binary + for strings returns concatenation of expr1 and expr2
```

### Returns
```
Binary + for strings returns concatenation of expr1 and expr2
```

### Remarks
You may use arithmetic operators with numbers as well as with series variables. In case of usage with series the operators are applied elementwise.

+=
Addition assignment. Applicable to numerical expressions or strings.

### Syntax
```
For strings returns concatenation of expr1 and expr2. For numbers returns integer or float value, or series of values.
```

### Example
For strings returns concatenation of expr1 and expr2. For numbers returns integer or float value, or series of values.

### Returns
```
For strings returns concatenation of expr1 and expr2. For numbers returns integer or float value, or series of values.
```

### Remarks
You may use arithmetic operators with numbers as well as with series variables. In case of usage with series the operators are applied elementwise.

<
Less than. Applicable to numerical expressions.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

<=

Less than or equal to. Applicable to numerical expressions.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

==
Equal to. Applicable to expressions of any type.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

=>
The '=>' operator is used in user-defined function declarations and in switch statements.

### Syntax
```
A <local_block> is zero or more Pine Script® statements.
```

### Example
You can learn more about user-defined functions in the User Manual's pages on Declaring functions and Libraries.

### Remarks
You can learn more about user-defined functions in the User Manual's pages on Declaring functions and Libraries.

>
Greater than. Applicable to numerical expressions.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

>=
Greater than or equal to. Applicable to numerical expressions.

### Syntax
```
Boolean value, or series of boolean values.
```

### Returns
```
Boolean value, or series of boolean values.
```

## @description
Sets a custom description for scripts that use the library declaration statement. The text provided with this annotation will be used to pre-fill the "Description" field in the publication dialogue.

### Example
If placed above an enum declaration, it adds a custom description for the enum. The Pine Editor's autosuggest uses this description and displays it when a user hovers over the enum name. When used in library scripts, the descriptions of all enums using the export keyword will pre-fill the "Description" field in the publication dialogue.

## @enum
If placed above an enum declaration, it adds a custom description for the enum. The Pine Editor's autosuggest uses this description and displays it when a user hovers over the enum name. When used in library scripts, the descriptions of all enums using the export keyword will pre-fill the "Description" field in the publication dialogue.

### Example
If placed above a type or enum declaration, it adds a custom description for a field of the type/enum. After the annotation, users should specify the field name, followed by its description.

## @field
If placed above a type or enum declaration, it adds a custom description for a field of the type/enum. After the annotation, users should specify the field name, followed by its description.

### Example
If placed above a function declaration, it adds a custom description for the function.

## @function
If placed above a function declaration, it adds a custom description for the function.

### Example
If placed above a function declaration, it adds a custom description for a function parameter. After the annotation, users should specify the parameter name, then its

description.

## @param
If placed above a function declaration, it adds a custom description for a function parameter. After the annotation, users should specify the parameter name, then its description.

### Example
If placed above a function declaration, it adds a custom description for what that function returns.

## @returns
If placed above a function declaration, it adds a custom description for what that function returns.

### Example
If used within a strategy script, it provides a default message to pre-fill the "Message" field in the alert creation dialogue.

## @strategy_alert_message
If used within a strategy script, it provides a default message to pre-fill the "Message" field in the alert creation dialogue.

### Example
If placed above a type declaration, it adds a custom description for the type.

## @type
If placed above a type declaration, it adds a custom description for the type.

### Example
If placed above a variable declaration, it adds a custom description for the variable.

## @variable
If placed above a variable declaration, it adds a custom description for the variable.

### Example
Specifies the Pine Script® version that the script will use. The number in this annotation should not be confused with the script's version number, which updates on every saved change to the code.

## @version=
Specifies the Pine Script® version that the script will use. The number in this annotation should not be confused with the script's version number, which updates on every saved change to the code.

### Example
The version should always be specified. Otherwise, for compatibility reasons, the script will be compiled using Pine Script® v1, which lacks most of the newer features and is bound to confuse. This annotation can be anywhere within a script, but we recommend placing it at the top of the code for readability.

### Example
The version should always be specified. Otherwise, for compatibility reasons, the script will be compiled using Pine Script® v1, which lacks most of the newer features and is bound to confuse. This annotation can be anywhere within a script, but we recommend placing it at the top of the code for readability.

### Remarks
The version should always be specified. Otherwise, for compatibility reasons, the script will be compiled using Pine Script® v1, which lacks most of the newer features and is bound to confuse. This annotation can be anywhere within a script, but we recommend placing it at the top of the code for readability.